

8000

EasyBuilder8000 使用手冊

適用 EasyBuilder8000 Version 4.66.02



第一章 關於 EasyBuilder 安裝.....	12
1.1 安裝 EasyBuilder.....	12
1.2 安裝步驟	13
第二章 Project Manager	19
2.1 HMI 位址及密碼設定.....	20
2.2 編輯工具	21
2.2.1 建立儲存在 CF/SD 卡與 USB 中的下載資料	21
2.2.2 透過 USB 碟 / SD / CF 卡下載程式到 HMI 的步驟.....	22
2.3 傳輸	23
2.3.1 下載	23
2.3.2 上傳	25
2.4 模擬	26
2.4.1 離線模擬和連線模擬	26
2.5 穿透通訊	28
第三章 建立簡單的工程檔案.....	29
3.1 建立新的工程檔	29
3.2 儲存和編譯工程檔	31
3.3 離線模擬和連線模擬	32
3.4 下載工程檔至 HMI.....	33
第四章 硬體設定.....	38
4.1 I/O 埠	38
4.2 系統設定	39
4.2.1 系統重置	39
4.2.2 系統工具列	40
4.2.3 系統資訊	41
4.2.4 系統設定	41
4.3 System Setting Editor.....	44
第五章 系統參數設定.....	46
5.1 設備清單	47
5.1.1 如何控制一台本機 PLC.....	48
5.1.2 如何控制一台遠端 PLC.....	53
5.1.3 如何控制一台遠端 HMI	55
5.2 HMI 屬性.....	57
5.3 一般屬性	61
5.4 系統設定	64
5.5 使用者密碼	67

5.5.1 設定用戶可以操作的物件類別與密碼	67
5.6 字體	69
5.7 擴展記憶體	70
5.8 列印/備份伺服器	72
第六章 視窗.....	74
6.1 視窗類型	74
6.1.1 基本視窗	74
6.1.2 快選視窗	74
6.1.3 公共視窗	76
6.1.4 系統訊息視窗	76
6.2 視窗的建立、設定與刪除	78
6.2.1 視窗的建立與設定	78
6.2.2 視窗的開啟、關閉或刪除	80
第七章 事件登錄.....	81
7.1 事件登錄管理	81
7.1.1 Excel 編輯	83
7.2 建立一個新的事件記錄	84
7.2.1 事件登錄一般屬性	84
7.2.2 事件登錄訊息設定	86
第八章 資料取樣.....	88
8.1 資料取樣記錄管理	88
8.2 新增一個資料取樣	89
第九章 物件一般屬性.....	93
9.1 選擇 PLC.....	93
9.1.1 讀寫位址設定	93
9.2 向量圖庫與圖片庫的使用	96
9.2.1 向量圖庫設定項	97
9.2.2 圖片庫設定項	100
9.3 文字內容設定	102
9.4 輪廓調整	106
9.5 站號變數的使用	107
9.6 廣播站號的使用	108
第十章 使用者密碼與物件安全防護.....	109
10.1 用戶密碼與可操作物件類別設定	109
10.2 物件操作安全防護	110
10.3 物件安全防護範例	111

第十一章 索引 暫存器.....	114
11.1 概要	114
11.2 範例	115
第十二章 鍵盤的設計與使用.....	118
12.1 設計自製的彈出鍵盤	119
12.2 使用直接視窗的方式來設計鍵盤	122
12.3 將鍵盤固定在需要輸入的視窗上	124
12.4 製作 UNICODE 鍵盤	125
第十三章 物件.....	126
13.1 位元狀態指示燈	127
13.2 多狀態指示燈	130
13.3 位元狀態設定	135
13.4 多狀態設定	139
13.5 功能鍵	147
13.6 位元狀態切換開關	154
13.7 多狀態切換開關	157
13.8 滑動開關	161
13.9 數值輸入與數值顯示	166
13.10 字元輸入與字元顯示	175
13.11 間接視窗	179
13.12 直接視窗	184
13.13 移動圖形	187
13.14 動畫	193
13.15 棒圖	198
13.16 錶針	204
13.17 趨勢圖	212
13.18 歷史數據顯示	226
13.19 數據群組顯示	236
13.20 XY 曲線圖	246
13.21 報警條與報警顯示	256
13.22 事件顯示	260
13.23 觸發式資料傳輸	268
13.24 備份	271
13.25 媒體播放器	276
13.26 定時式資料傳輸	284
13.27 PLC 控制.....	287

13.28 排程	295
13.29 項目選單	313
13.30 計時器	320
13.31 影像輸入	325
13.32 系統訊息	328
13.33 事件序列顯示	331
13.34 QR 碼	334
第十四章 向量圖庫與圖片庫的與使用	335
14.1 向量圖庫的建立	335
14.2 圖片庫的建立	342
第十五章 文字標籤庫與多國語言使用	349
15.1 介紹	349
15.2 文字標籤庫的建立	351
15.3 文字標籤庫字體設定	352
15.4 文字標籤庫的使用	353
15.5 多國語言的使用 (系統暫存器 LW-9134)	354
第十六章 位址標籤庫的建立與使用	356
16.1 位址標籤庫的建立	356
16.2 位址標籤庫的使用	358
第十七章 配方資料傳送	359
17.1 使用乙太網路或 USB 線更新配方資料	360
17.2 使用 SD 卡或 USB 碟更新配方資料	361
17.3 配方資料傳輸	362
17.4 配方資料儲存準則	362
第十八章 巨集指令說明	363
18.1 巨集指令編輯器功能使用說明	363
18.2 巨集指令的結構	372
18.3 巨集指令的語法	373
18.3.1 常數和變數	373
18.3.1.1 常數	373
18.3.1.2 變數	373
18.3.2 運算符號	375
18.4 語句	378
18.4.1 定義語句	378
18.4.2 賦值語句	378
18.4.3 邏輯運算語句	378

18.4.4 多重判斷語句	381
18.4.5 迴圈語句	383
18.4.5.1 for-next 語句	383
18.4.5.2 while-wend 語句	384
18.4.5.3 其他控制命令	385
18.5 子函數	385
18.6 內置函數功能	388
18.6.1 數學運算函數	388
18.6.2 資料轉換函數	395
18.6.3 資料操作函數	400
18.6.4 位元狀態轉換	403
18.6.5 通訊有關的函數	405
18.6.6 字串處理函數	422
18.6.7 其他函數	449
18.7 怎樣建立和執行巨集指令	458
18.7.1 怎樣建立一個巨集指令	458
18.7.2 執行巨集指令	462
18.8 使用者自定義函數功能	463
18.8.1 汇入函數庫檔案	464
18.8.2 如何使用巨集函數庫	465
18.8.3 函數庫管理介面	467
18.8.3.1 新建函數	468
18.8.3.2 刪除函數	470
18.8.3.3 修改函數	471
18.8.3.4 汇入函數	472
18.8.3.5 汇出函數	473
18.9 使用巨集指令時的注意事項	474
18.10 使用自由協定去控制一個設備	475
18.11 編譯錯誤提示資訊	482
18.12 巨集指令範例程式	489
18.13 巨集指令 TRACE 函數	494
18.14 字串處理函式使用方法	501
18.15 巨集指令密碼保護	512
第十九章 如何將 HMI 設定成 MODBUS 裝置	513
19.1 將 HMI 設定成 MODBUS 裝置	513
19.1.1 建立一個 MODBUS Server 裝置	514

19.1.2 讀寫一個 MODBUS Server 裝置.....	517
19.2 線上更改 MODBUS Server 的站號.....	520
19.3 關於 MODBUS 各位址的說明	521
第二十章 如何使用條碼掃描器裝置.....	522
20.1 如何使用條碼掃描器裝置	522
第二十一章 乙太網路通訊與多台人機連線.....	526
21.1 HMI 與 HMI 間的通訊.....	527
21.2 PC 與 HMI 間的通訊	528
21.3 控制連接在其他 HMI 上的 PLC.....	529
第二十二章 位址暫存器.....	530
22.1 本機 HMI 記憶體位址範圍.....	531
22.1.1 位元位址	531
22.1.2 字元位址	531
22.2 系統暫存器	532
22.2.1 HMI 時間.....	532
22.2.2 HMI 操作.....	533
22.2.3 觸碰位置	534
22.2.4 本地 HMI 網路資訊	535
22.2.5 工程檔案資訊	536
22.2.6 儲存空間管理	536
22.2.7 配方及擴展記憶體	537
22.2.8 資料取樣	538
22.2.9 事件登錄	539
22.2.10 站號變數	541
22.2.11 索引暫存器	542
22.2.12 MODBUS Server 通訊	543
22.2.13 通訊參數設定	544
22.2.14 與 PLC (COM) 的通訊狀態與控制	547
22.2.15 與 PLC (乙太網路) 的通訊狀態與控制	550
22.2.16 與 PLC (USB) 的通訊狀態與控制.....	553
22.2.17 與遠端 HMI 的通訊狀態與控制	553
22.2.18 與遠端 PLC 的通訊狀態與控制	560
22.2.19 本地/遠端操作限制	563
22.2.20 通訊錯誤碼	563
22.2.21 驅動程式 ID	564
22.2.22 DLT645 控制器	564

22.2.23 [PLC No Response] 視窗控制	565
22.2.24 [快選] 視窗控制	565
22.2.25 EasyAccess	566
22.2.26 遠端列印/備份伺服器	567
22.2.27 穿透通訊設定	568
22.2.28 VNC 控制	568
22.2.29 HMI 和工程檔案識別碼	569
22.2.30 使用者名稱和密碼	570
22.2.31 巨集	571
22.2.32 輸入物件功能	572
22.2.33 其它功能	572
第二十三章 HMI 支援的印表機類型	574
23.1 支援的印表機類型	574
23.2 如何新增一台印表機設備並觸發列印	577
23.2.1 新增印表機類型	577
23.2.2 觸發列印功能	578
第二十四章 Recipe Editor	579
24.1 概要	579
24.2 Recipe/Extended Memory Editor 設定	579
第二十五章 EasyConverter	582
25.1 概要	582
25.2 將資料取樣記錄檔案輸出至 Excel	582
25.3 將事件記錄檔案輸出至 Excel	583
25.4 多檔案轉換	584
25.5 比例轉換功能	585
第二十六章 EasyPrinter	587
26.1 使用 EasyPrinter 為列印伺服器	588
26.1.1 EasyPrinter 設定程序	588
26.1.2 EasyBuilder 設定程序	589
26.2 使用 EasyPrinter 為備份伺服器	592
26.2.1 EasyPrinter 備份設定程序	592
26.2.2 EasyBuilder 備份設定程序	593
26.3 EasyPrinter 操作說明	597
26.3.1 視窗介面	597
26.3.2 操作說明	598
26.4 轉換批次檔	603

26.4.1 轉換批次檔預設值	603
26.4.2 特定標準	604
26.4.3 轉換批次檔格式	605
26.4.4 執行	605
第二十七章 EasySimulator.....	607
27.1 準備相關檔案	607
27.2 設定 xob_pos.def 內容	608
第二十八章 使用串列埠實現一機多屏功能 (主從模式)	609
28.1 如何設定主機所使用工程檔案的內容	610
28.2 如何設定從機所使用工程檔案的內容	611
28.3 如何連結從機的 MT500 工程	614
第二十九章 穿透通訊功能.....	617
29.1 乙太網路模式	618
29.1.1 如何安裝虛擬序列埠驅動程式	618
29.1.2 如何更改虛擬串列埠	619
29.1.3 如何使用乙太網路穿透通訊功能	620
29.2 序列埠模式	622
29.2.1 序列埠設定	622
29.2.2 HMI 工作模式	624
29.3 使用系統暫存器啟動穿透通訊功能	624
第三十章 工程檔案保護功能.....	626
30.1 XOB 密碼.....	627
30.2 禁止反編譯	628
30.3 禁止 XOB 檔案上傳功能.....	629
30.4 工程檔案識別碼	630
30.5 MTP 密碼	631
第三十一章 Memory Map 通訊協定	632
31.1 簡介	632
31.2 接腳設定	632
31.3 通信流程圖	633
31.4 通信資料格式	635
31.4.1 通信範例	636
31.5 實作範例	638
31.5.1 新增 Memory Map 設備	638
31.5.2 物件設定	640
31.5.3 執行結果	642

第三十二章 FTP 伺服器之運用.....	643
32.1 登入 FTP 伺服器.....	643
32.2 備份歷史資料及更新配方資料	645
第三十三章 EasyDiagnoser	647
33.1 簡介與設定方式	647
33.2 EasyDiagnoser 設定.....	650
33.3 錯誤代碼	656
33.4 另存新檔	657
33.5 視窗調整	658
第三十四章 Rockwell EtherNet/IP Free Tag Names	659
34.1 匯入使用者自訂 AB Tag CSV 檔至 EasyBulder	660
34.2 新增資料型態	662
34.3 貼上功能	665
34.4 其它功能	667
34.5 模組預設結構	671
第三十五章 EasyWatch.....	675
35.1 EasyWatch 概述.....	675
35.2 基本功能介紹	676
35.2.1 基本功能	676
35.2.2 快速工具	678
35.3 監視物件設定	679
35.3.1 新增監視物件	679
35.3.2 監視物件設定	680
35.3.3 新增監視設定	681
35.4 巨集物件設定	687
35.4.1 新增巨集物件	687
35.4.2 巨集物件設定	688
35.4.3 新增巨集設定	689
35.5 HMI 設定.....	691
35.5.1 HMI 設定.....	691
35.5.2 HMI 管理器設定.....	692
35.6 物件顯示列表	693
35.6.1 頁面設定	693
35.6.2 物件顯示欄位	694
第三十六章 事件序列記錄.....	695
36.1 介紹	695

36.2 事件序列設定	696
36.3 事件序列顯示物件	700
第三十七章 MODBUS TCP/IP 閘道功能	703
37.1 MODBUS TCP/IP 閘道簡介	703
37.2 位址對應設定	703
37.2.1 如何建立一個位址對應表	703
37.2.2 位址對應設定須知	705
第三十八章 EasyDownload	707
38.1 概要	707
38.2 設定	707

第一章 關於 EasyBuilder 安裝

1.1 安裝 EasyBuilder

軟體來源:

可由隨機光碟獲取，也可進入威綸公司網站 <http://www.weintek.com> 下載所有可用軟體語言版本（包括簡體中文、繁體中文、英文、義大利文、韓文、西班牙文、俄羅斯文及法文版本）及最新軟體更新資訊。

電腦硬體要求（建議配置）:

CPU: INTEL Pentium II 以上等級

記憶體: 256MB 以上

硬碟: 2.5GB 以上，最少有 500MB 以上的磁碟空間

光碟機: 4 倍速以上光碟機一個

顯示器: 支援解析度 1024 x 768 以上的彩色顯示器

鍵盤和滑鼠各一個

乙太網路埠: [工程下載] / [工程上傳] 時使用

USB 埠 2.0: [工程下載] / [工程上傳] 時使用

RS-232 COM 埠: 線上模擬時使用

印表機

作業系統:

Windows XP / Windows Vista / Windows 7 均可。

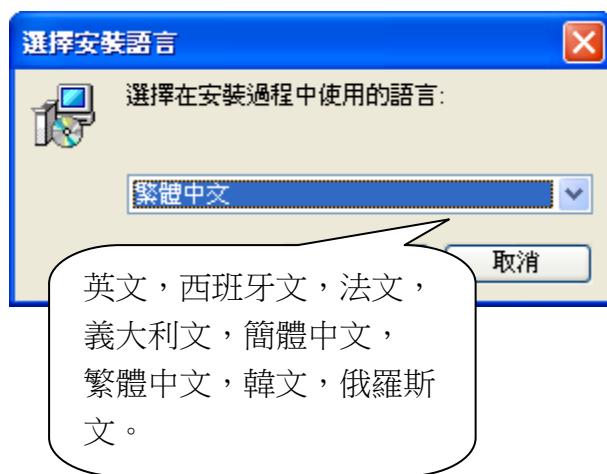
1.2 安裝步驟

1. 安裝 EasyBuilder:

將光碟放入光碟機，電腦將會自動執行安裝程式，或者您手動執行光碟根目錄下的 **[Autorun.exe]** 檔案，螢幕將顯示安裝程式如下：



2. 點選 **[Install]**，螢幕顯示如下，請選擇所需語言版本後，點選 **【下一步】**。

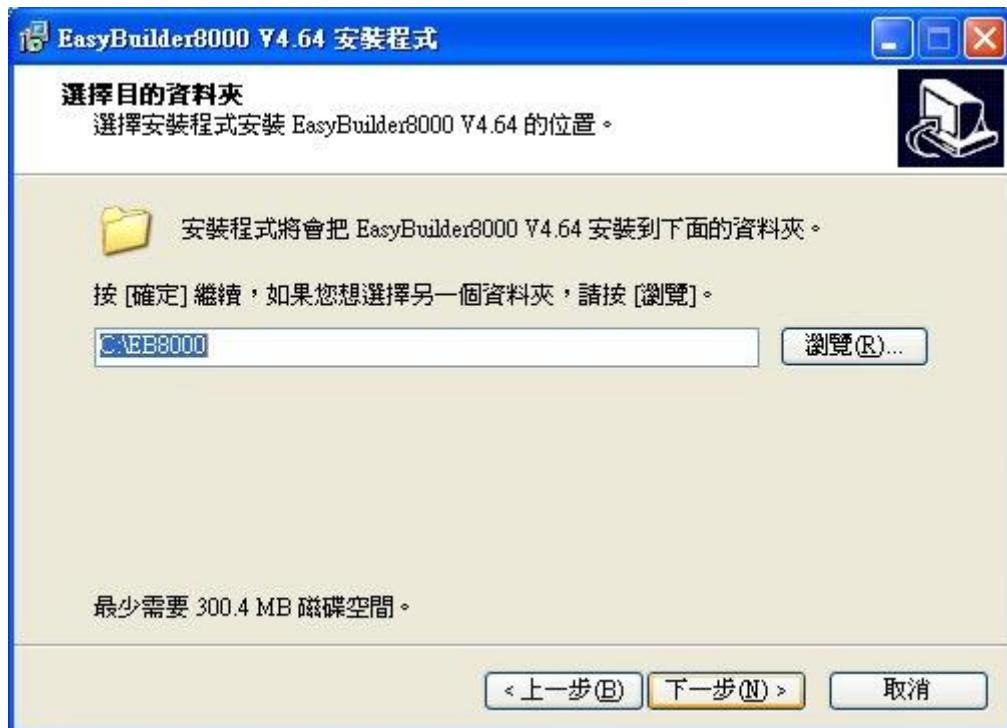




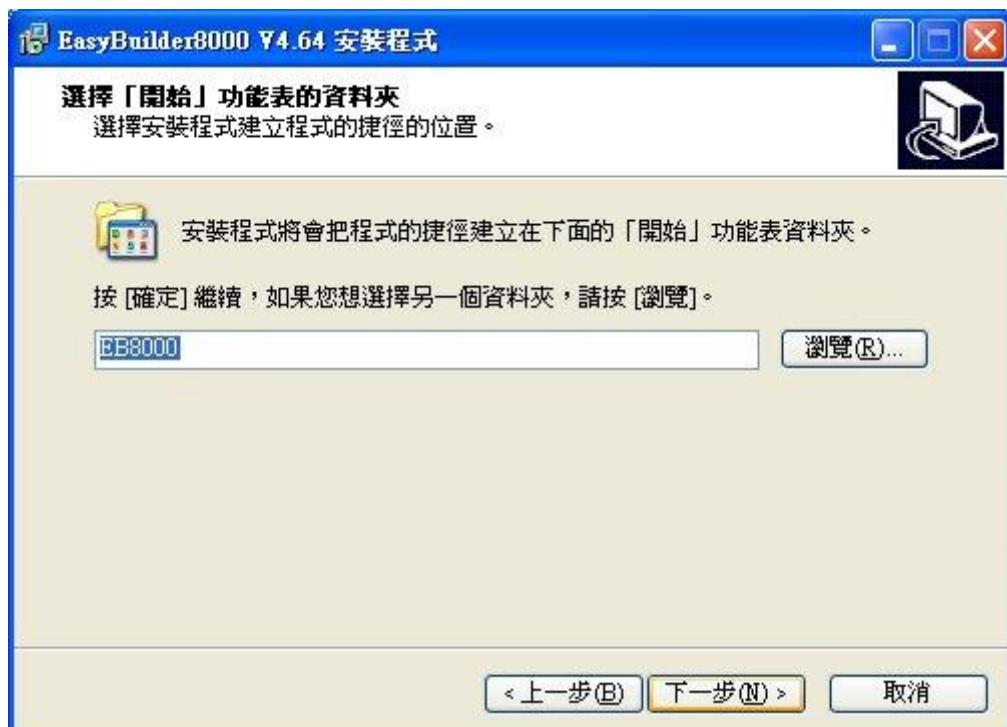
3. 系統將會詢問使用者是否想要移除電腦中已安裝過的 EasyBuilder 版本，請根據需求選擇後，點選 **[下一步]**。



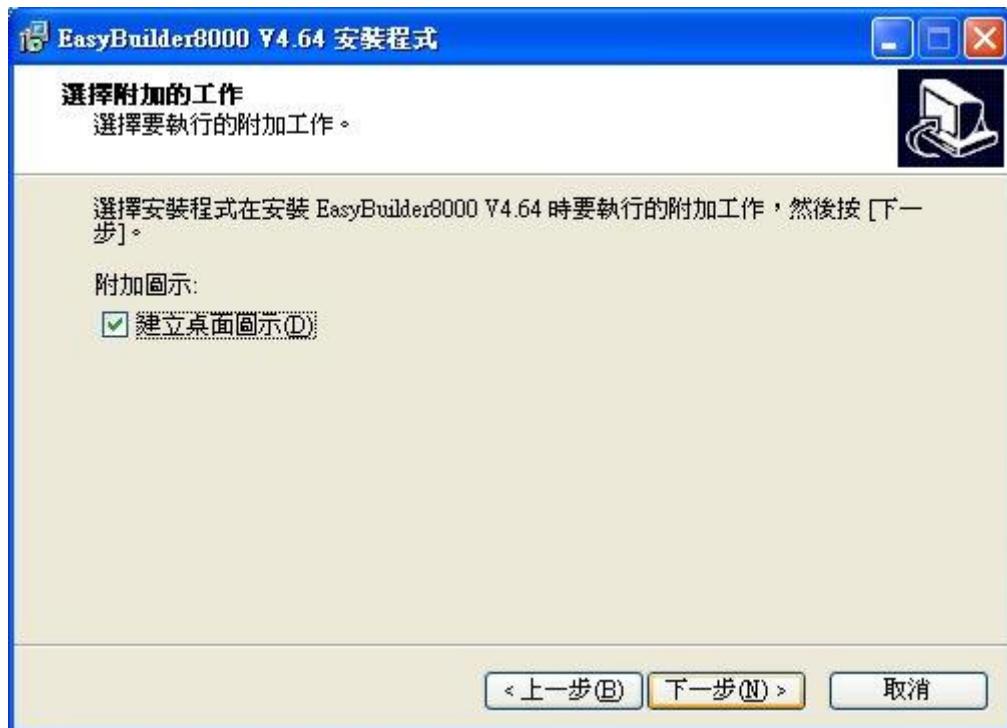
4. 指定一個全新的資料夾給 EasyBuilder 安裝資料，或是直接使用系統建議的資料夾，點選【下一步】。



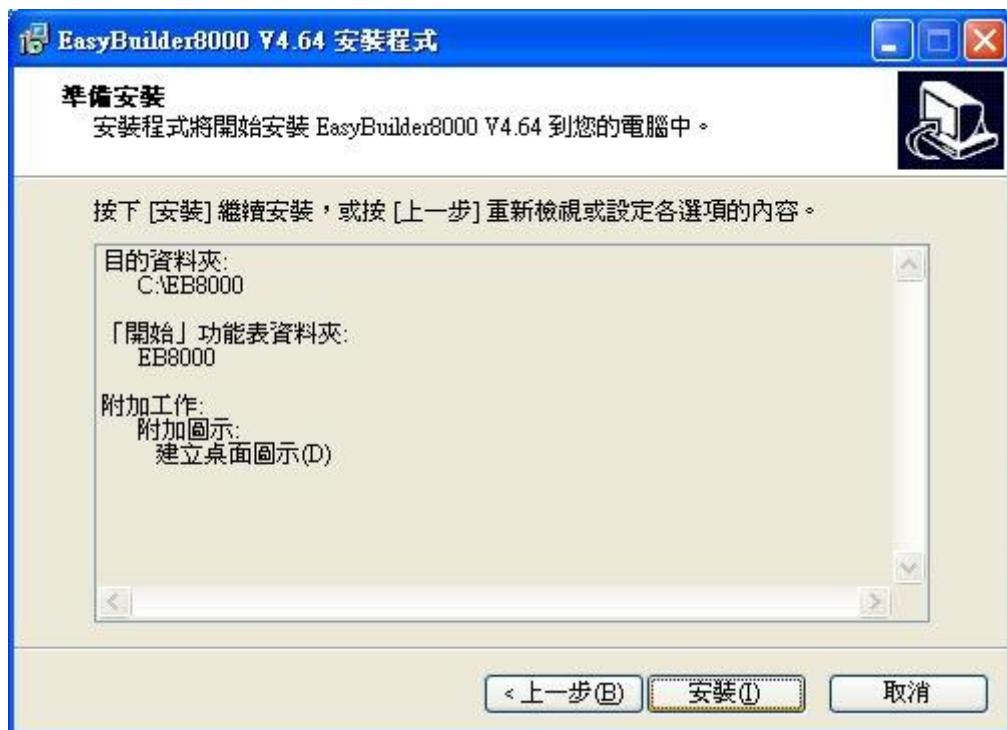
5. 系統會請使用者指定一個開始功能表資料夾用來建立程式捷徑，點選【瀏覽】指定一個資料夾，或是使用程式建議的資料夾，點選【下一步】繼續安裝程序。



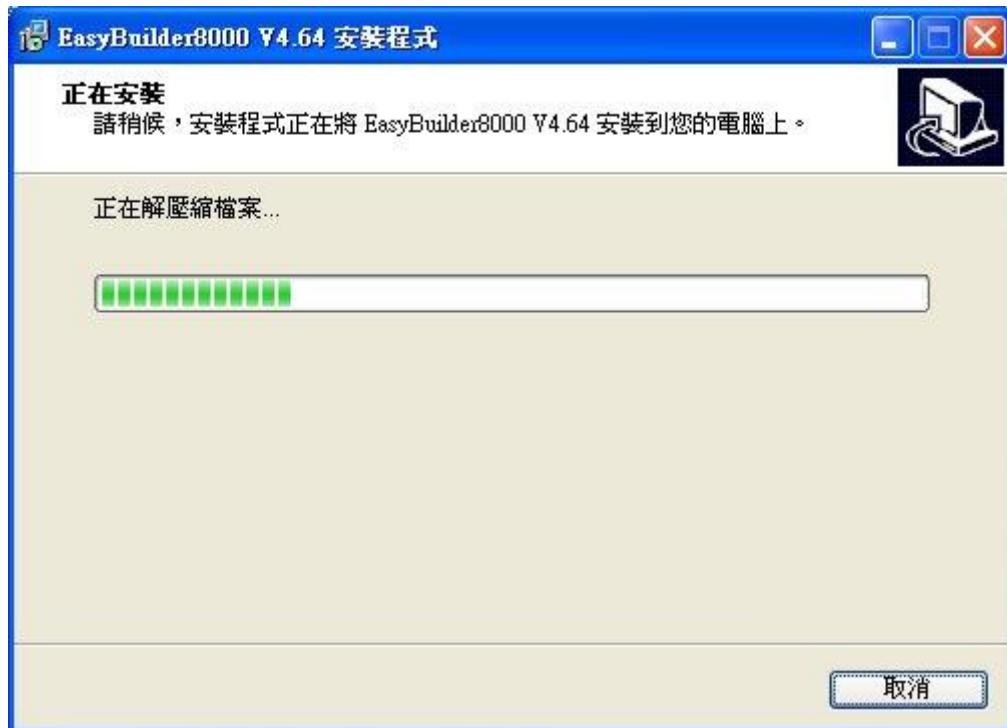
6. 系統會詢問使用者是否要執行附加工作，例如【建立桌面圖示】，若需要請勾選，然後按【下一步】繼續安裝程序。



7. 在此階段所有設定已完成，請檢查是否無誤，若有需要改選的部份，請按【上一步】，若全部正確，請按【安裝】開始安裝程序。



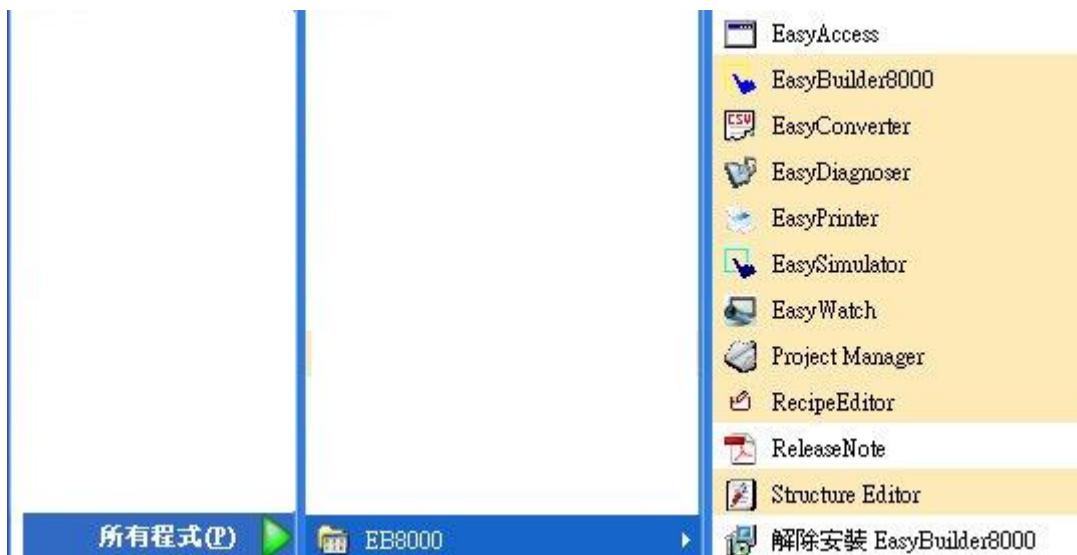
8. 安裝程序執行中。



9. 安裝完成，請按【完成】結束安裝。



10. 在 [開始] » [所有程式] » [EasyBuilder] 目錄下可看到 EasyBuilder 各功能捷徑。



軟體目錄下各選項的涵義如下：

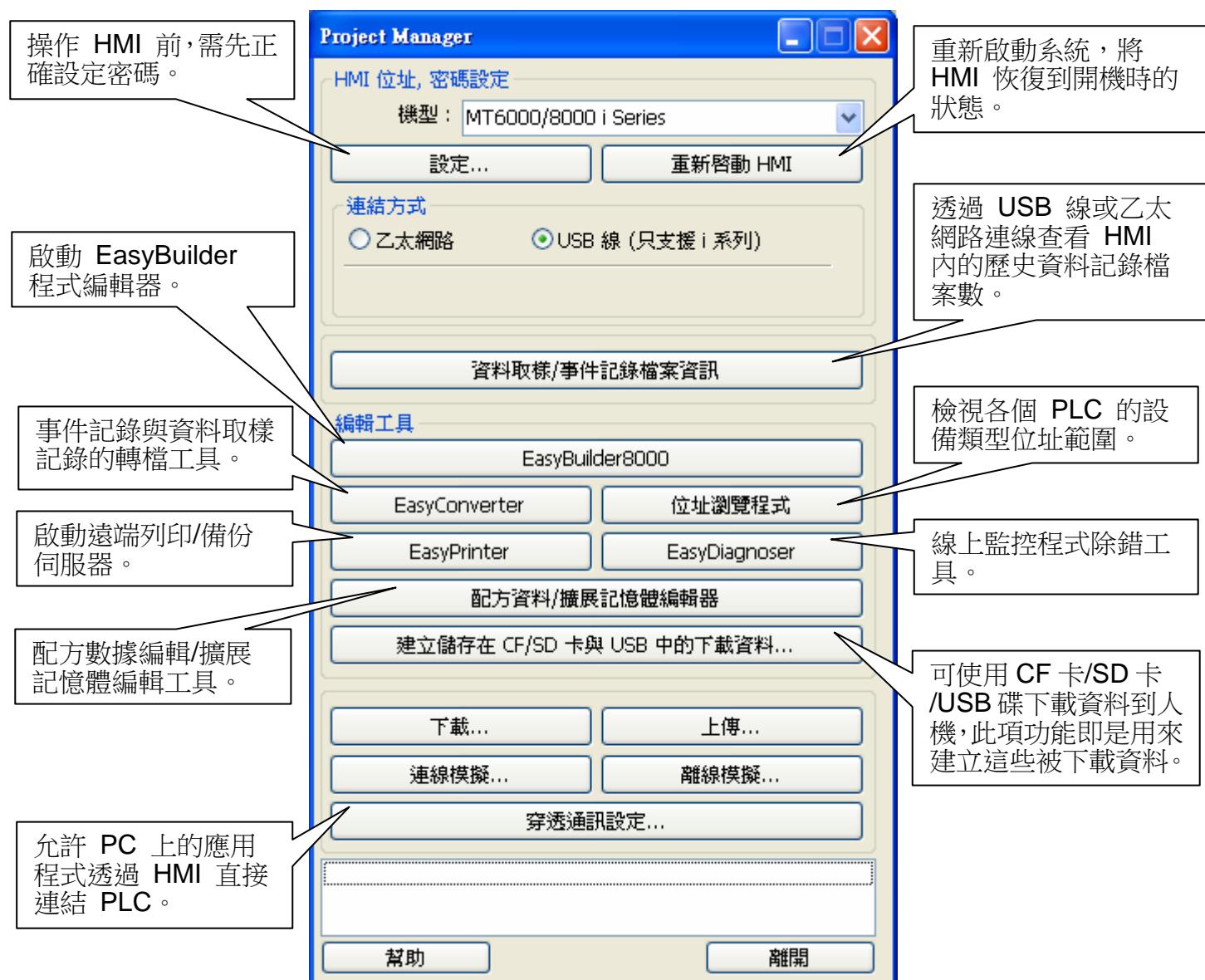
選項	敘述
EasyAccess	與網路連接中的 HMI 之管理工具。
EasyBuilder8000	EasyBuilder 程式編輯軟體。
EasyConverter	取樣數據與事件記錄檔轉檔工具。
EasyDiagnoser	線上監控程式通訊除錯工具。
EasyPrinter	遠端螢幕列印及資料備份的伺服器。
EasySimulator	透過此程式可不需安裝 EasyBuilder 即可執行程式模擬。
EasyWatch	可以透過 PC 監看或設定 HMI 和 PLC 內的位址數值，以達到監控的目的。
Project Manager	EasyBuilder 綜合管理軟體。
Recipe Editor	可以設定配方數據格式，開啟配方數據檔案及外部記憶體檔案。
Release Note	軟體版本及相關最新資訊說明。
Structure Editor	支援 AB TAG 結構，可增進物件讀/寫的活用性。



- HMI 支援使用 USB 線下載/上傳程式，完成安裝 EasyBuilder 後，可至 [電腦管理] » [裝置管理員] 確認 USB 驅動是否也一併安裝完成，若尚未被安裝，請手動完成安裝。

第二章 Project Manager

在 EasyBuilder 軟體安裝完成後，雙擊 PC 桌面上的 **[Project Manager]** 捷徑即可開啟。這是 EasyBuilder 軟體的綜合管理器，可當成獨立的程式來操作。若需使用 EasyWatch，就必須從 EasyBuilder 工程檔案編輯軟體開啟。



2.1 HMI 位址及密碼設定

[設定]

當使用者要用乙太網路或 USB 線操作 HMI 時，需正確設定操作 HMI 所需的密碼，避免沒有授權的使用者入侵 HMI 程式。



[重置/下載] 功能使用同一組密碼，**[上傳]** 功能則使用另外一組密碼。



- 請妥善保管密碼，若因忘記密碼而要將 HMI 恢復為原廠設定時，將導致 HMI 內部的所有程式資料被清除。



[重新啟動 HMI]

不需拔除電源即可重新啟動 HMI，並恢復到一開機時的狀態。若使用乙太網路重啟 HMI 時，請設定正確的 HMI IP 位址。

[資料取樣/事件記錄檔案資訊]

設定好連結方式，即可連線至 HMI 查看內部歷史資料檔案個數。



2.2 編輯工具

2.2.1 建立儲存在 CF/SD 卡與 USB 中的下載資料



1. 將 CF/SD 卡或 USB 碟接上 PC。
2. 指定檔案資料所要存放的路徑位置。
3. 指定所要建立的來源資料檔案存放位置。
4. 點選 [建立]。

所要建立的來源資料檔案將寫入所指定的外部裝置中，讓使用者可以透過該裝置下載工程檔案至 HMI，可以不需要透過乙太網路或 USB 線下載工程檔案。

2.2.2 透過 USB 碟 / SD / CF 卡下載程式到 HMI 的步驟

假設已經把來源資料檔案建立在 USB 碟裡的資料夾名稱 “**123**” (**K:\123**)

1. 將 USB 碟 (工程檔案已包含在內) 接上 HMI。
2. 彈出 [**Download / Upload**] 視窗，請選擇 [**Download**]。
3. 輸入下載密碼。
4. 在 [**Download Settings**] 視窗，勾選 [**Download project files**] 以及 [**Download history files**]。
5. 點擊 [**OK**]。
6. 在 [**Pick a Directory**] 視窗，選擇路徑：**usbdisk/device-0/123**。
7. 點擊 [**OK**]。

工程檔案將被自動更新。

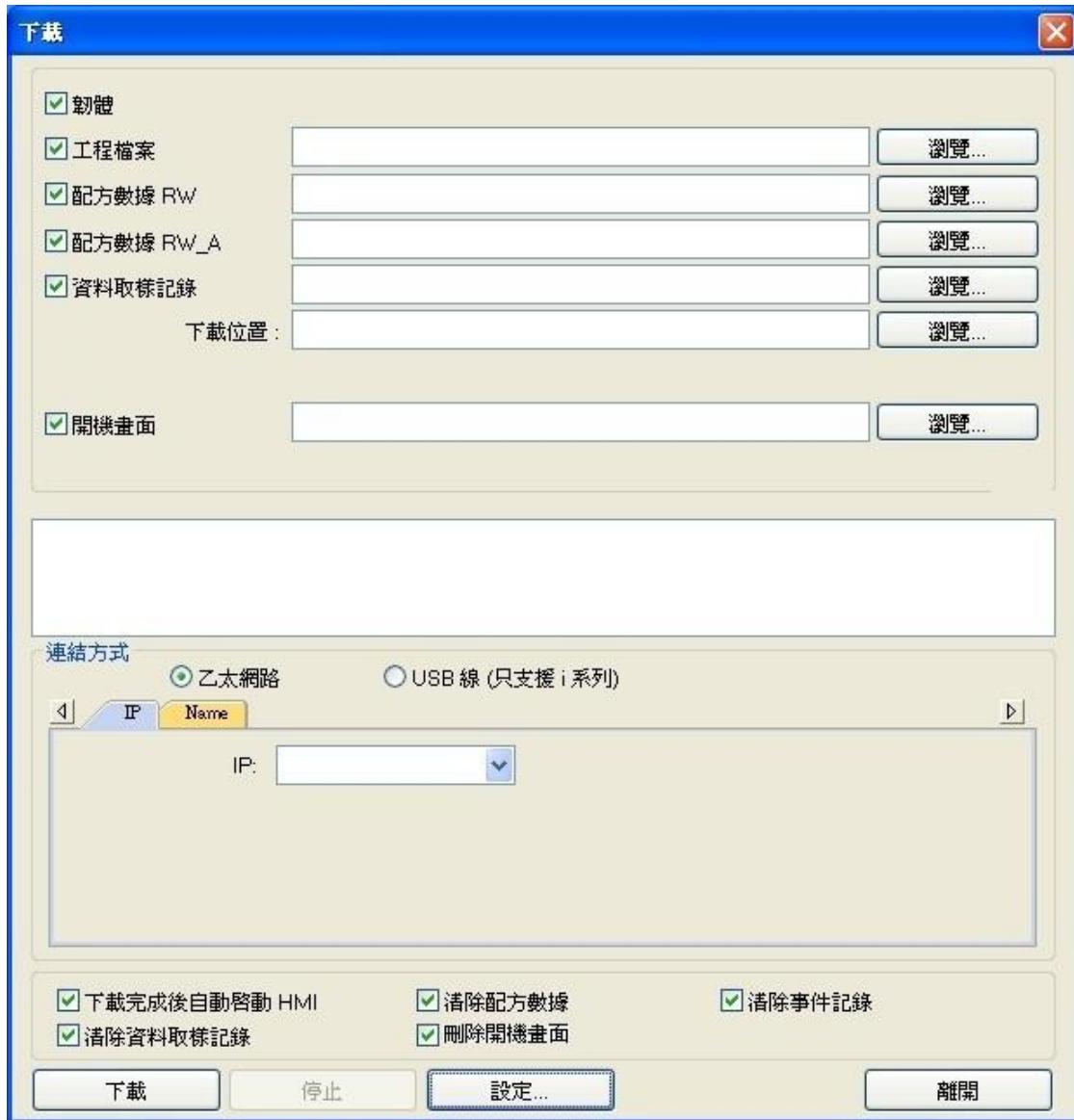


- 若沒有下載工程檔案而只有下載歷史資料，必須手動去重新啟動 HMI 更新檔案。

2.3 傳輸

2.3.1 下載

透過此功能可以使用乙太網路或 USB 線下載檔案到 HMI 上。



【韌體】

若勾選此項，表示要更新 HMI 所有核心程式。第一次下載檔案至 HMI 時，一定要下載韌體。

【工程檔案】

選擇 xob 格式的工程檔案。

[配方數據 RW / RW_A]

選擇 rcp 格式的配方檔案。

[資料取樣記錄]

先選擇 HMI 上資料取樣的資料夾名稱後，再選擇 dtl 格式的資料取樣檔案。

[安裝 X 系列媒體播放器驅動程式]

第一次使用 EasyBuilder 下載程式到 X 系列 HMI 前，請先下載此驅動程式。

[開機畫面]

將指定的 bmp 圖片下載到 HMI，HMI 啟動時，就會先顯示此圖片，再載入下載的程式。

[下載完成後自動啟動 HMI]

若勾選此項，HMI 將會在下載檔案成功後自動重新啟動。

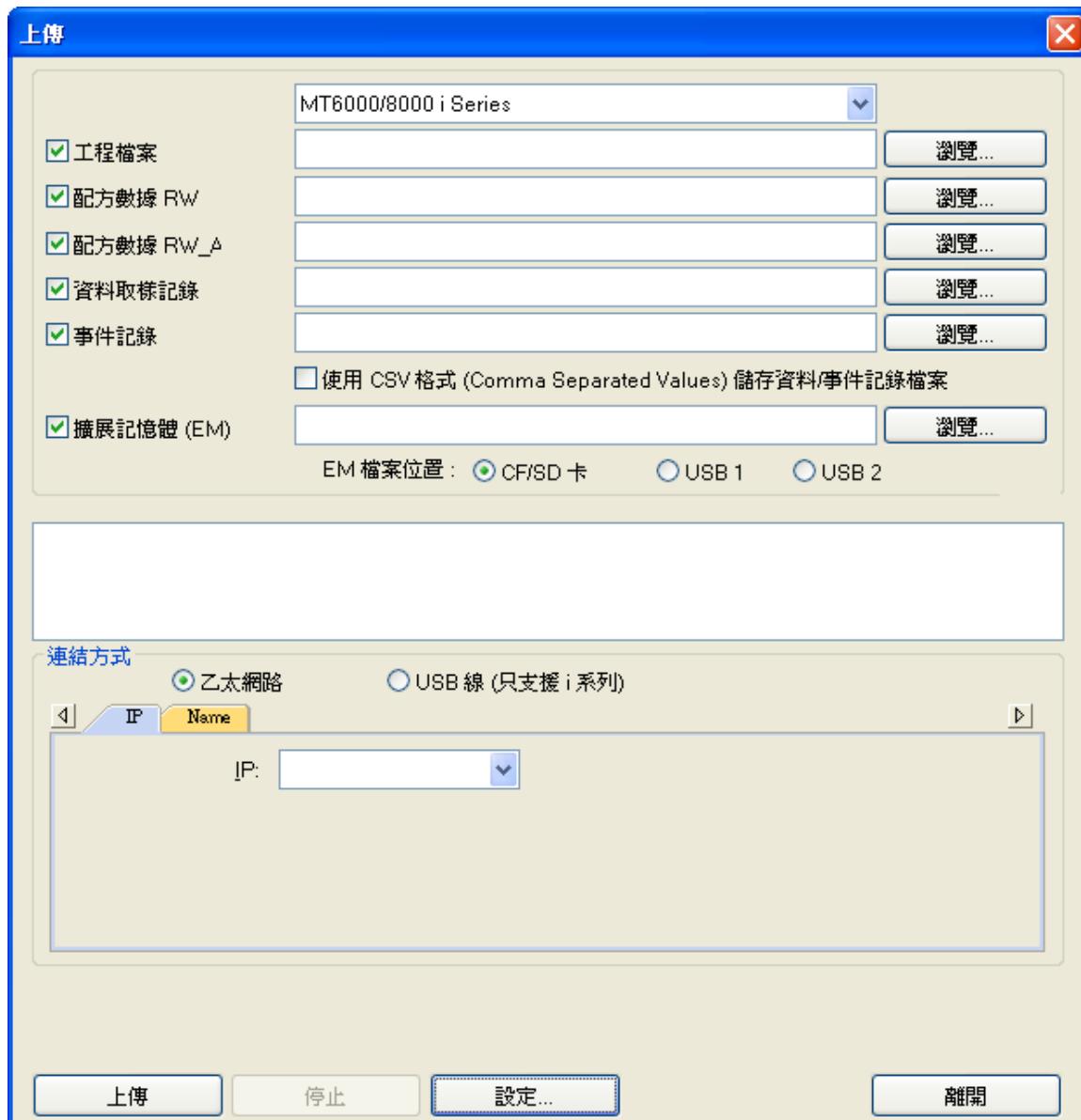
[清除配方數據 / 清除事件記錄 / 清除資料取樣記錄 / 刪除開機畫面]

若勾選此項，下載檔案前會先清除 HMI 所選取的檔案。

2.3.2 上傳

透過此功能可以使用乙太網路或 USB 線上傳 HMI 的檔案到 PC。

上傳前須先選擇存放檔案的路徑。按下 **[瀏覽]**，指定上傳檔案所要存放的位置。



[工程檔案]、[配方數據 RW / RW_A]、[資料取樣記錄] 可參考 2.3.1 下載介紹。

[事件記錄]

將 HMI 的 evt 檔案上傳到 PC。

[擴展記憶體(EM)]

將 HMI 上的 CF/SD 卡、USB 碟內的.emi 檔案上傳到 PC。



- 若將工程檔案上傳至 PC，由於上傳回來的檔案格式為 xob 檔案，使用者需先反編譯為 mtp 檔案，才能透過 EasyBuilder 編輯。
- 上傳功能無法支援上傳儲存於外部裝置的歷史資料，但仍可透過 FTP 伺服器的方式，詳細資訊請參考《第 32 章 FTP 伺服器之運用》。

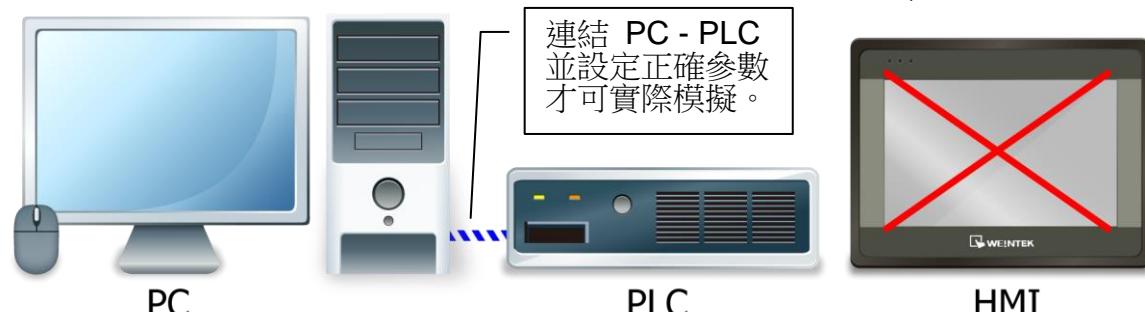
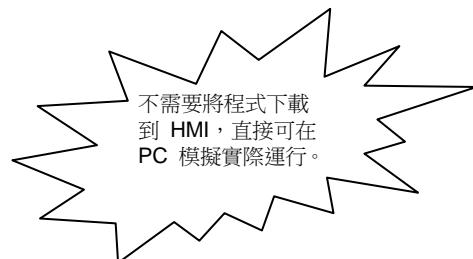
2.4 模擬

2.4.1 離線模擬和連線模擬

離線模擬 - 在 PC 上模擬工程檔案的運行，不與任何裝置連線。



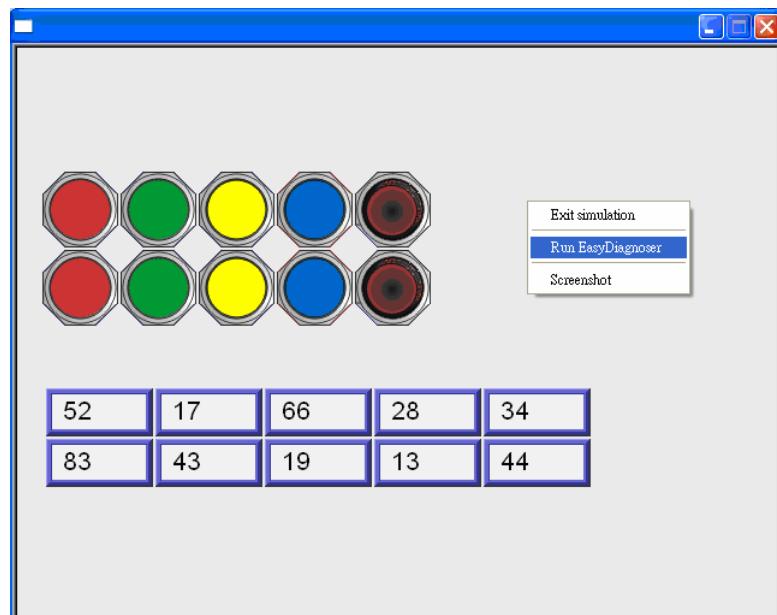
連線模擬 - 在 PC 上模擬工程檔案的運行，此時 PLC 是直接與 PC 連接。



在 PC 上進行【連線模擬】時，若監控設備是接在本地 PC 上的 PLC，監控時間會有 **10 分鐘**的限制。

執行連線模擬和離線模擬功能，需先選擇 **xob** 檔案的來源位置。

在執行連線模擬/離線模擬時，點選滑鼠右鍵後可以執行以下功能：



[Exit simulation]

關閉模擬狀態。

[Run EasyDiagnoser]

執行 EasyDiagnoser

監看目前的通訊狀態。

[Screenshot]

將目前的模擬畫面使用圖檔的方式
儲存到安裝路徑下的 **screenshot**
資料夾。

2.5 穿透通訊

穿透通訊功能允許 PC 上的應用程式透過 HMI 直接連結 PLC，此時 HMI 所扮演的角色類似轉接器。



穿透通訊功能包含 **【串列埠】** 模式與 **【乙太網路】** 模式。

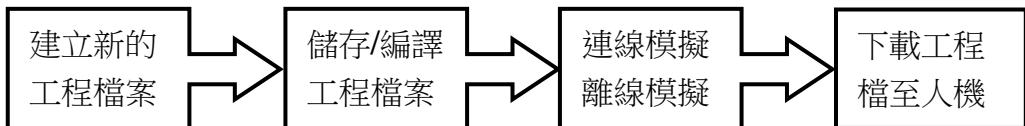
在使用 **【乙太網路】** 穿透通訊功能前，請先安裝虛擬串列埠驅動程式。



如欲知詳情，請參考《第 29 章 穿透通訊功能》。

第三章 建立簡單的工程檔案

以下為製作一個工程檔案的程序：



3.1 建立新的工程檔

1. 進入 EasyBuilder
2. 開啟新檔。
3. 選擇 **【型號】**。
4. 勾選 **【使用範本】**。
5. 點擊 **【確定】**。



9. **【設備清單】** 增加了一個新的裝置。

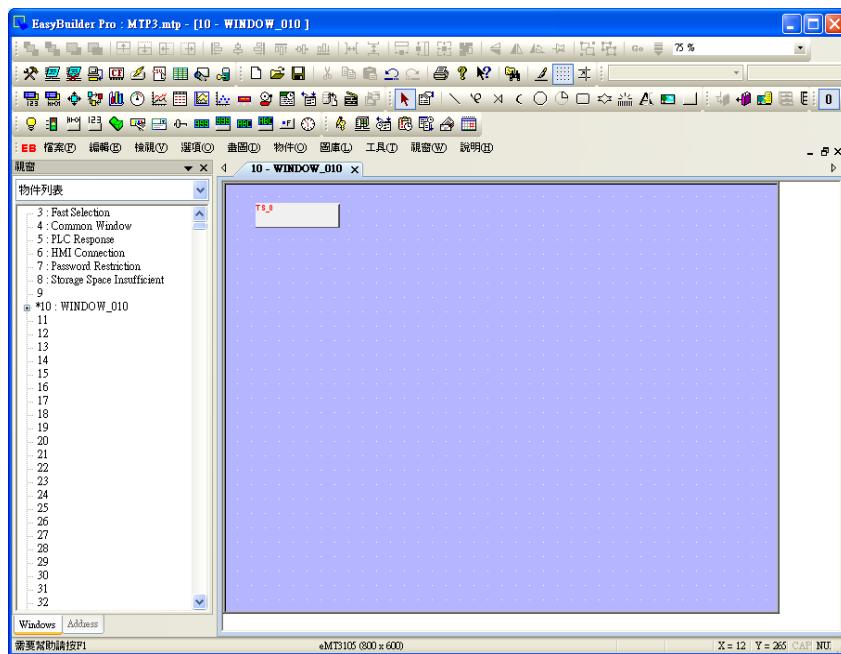


現在讓我們建立一個新物件：

1. 建立一個位元狀態切換開關物件。
2. 設定位址。



3. 將物件放置於編輯視窗中。
4. 一個含有單一物件的工程檔便完成了。



3.2 儲存和編譯工程檔

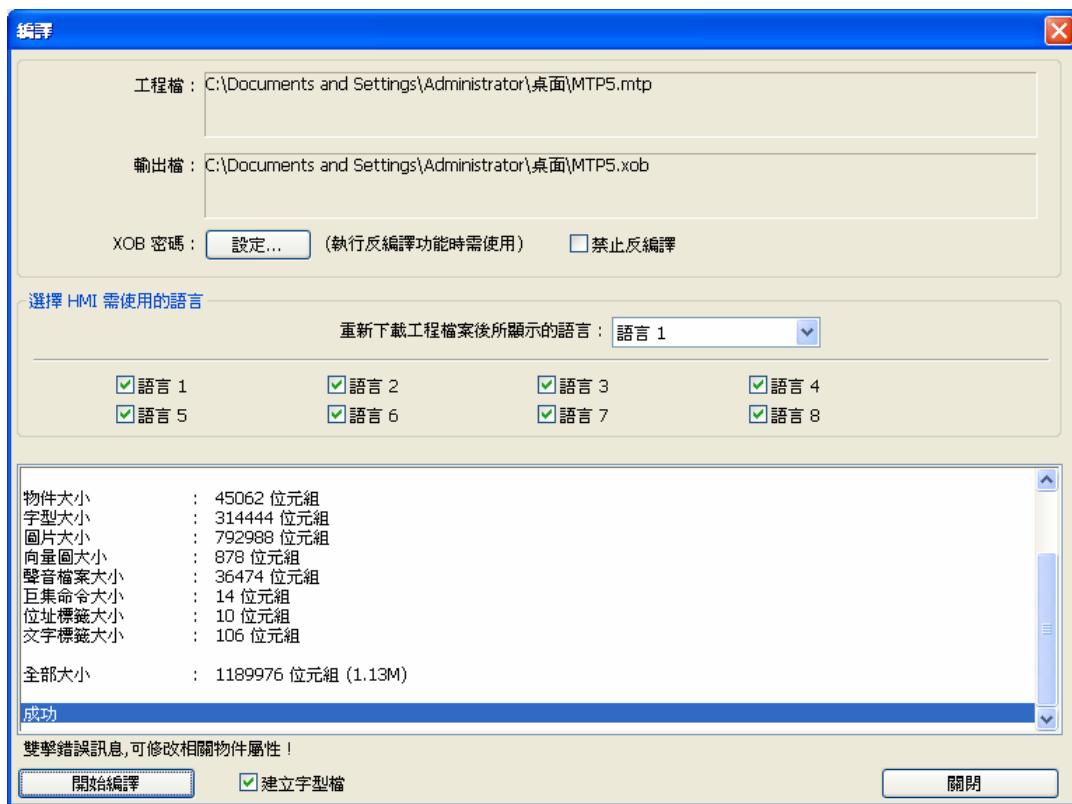
EasyBuilder 工具列上：



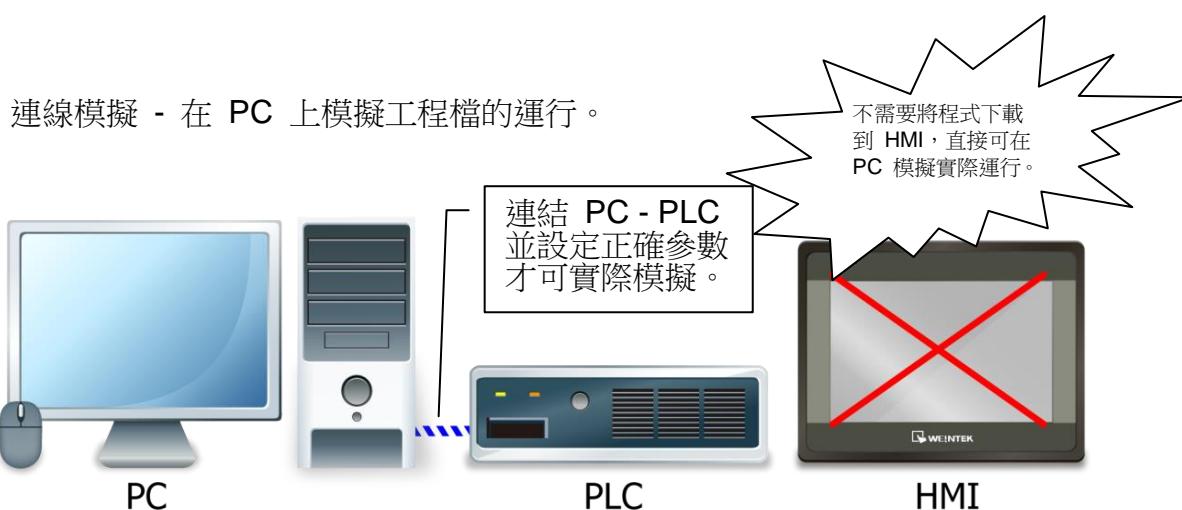
- 點擊【儲存檔案】存成 *.mtp 檔。

- 點擊【編譯】成為 *.xob 檔下載至 HMI，並檢查工程檔可否正常運行。

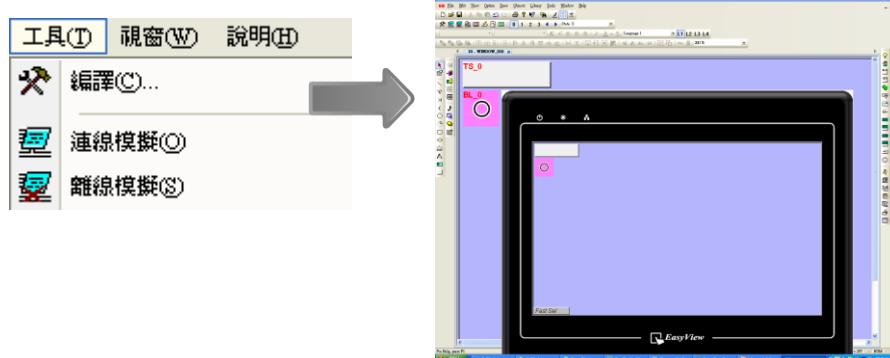
使用者需事先於文字標籤庫設定多國語言後，再選擇工程檔所需要的語言並下載至 HMI，最多可勾選 24 種語言。成功的編譯工程檔對話視窗如下。



3.3 離線模擬和連線模擬



- 在 PC 上進行 **【連線模擬】** 時，如監控對象為本地的 PLC (也就是接在 PC 上的 PLC)，則監控時間會有 **10 分鐘**的限制。



3.4 下載工程檔至 HMI

■ 方法 1 [乙太網路] » HMI IP » [下載] 之前，請確認所有設定是否正確。

設定 [密碼] 並
指定 [HMI IP]。

[Runtime (韌體)]

勾選此選項表示要更新機器上的所有核心程式。第一次下載檔案或更新 EasyBuilder 版本之後，當要下載檔案至 HMI 時，一定要下載此韌體。

[字型檔案]

將工程檔中選用的字型下載至 HMI。

[刪除開機畫面]

[清除配方數據]

[清除事件記錄]

[清除資料取樣記錄]

選項如被勾選，下載程式前會先清除機器上所選取存在的檔案。

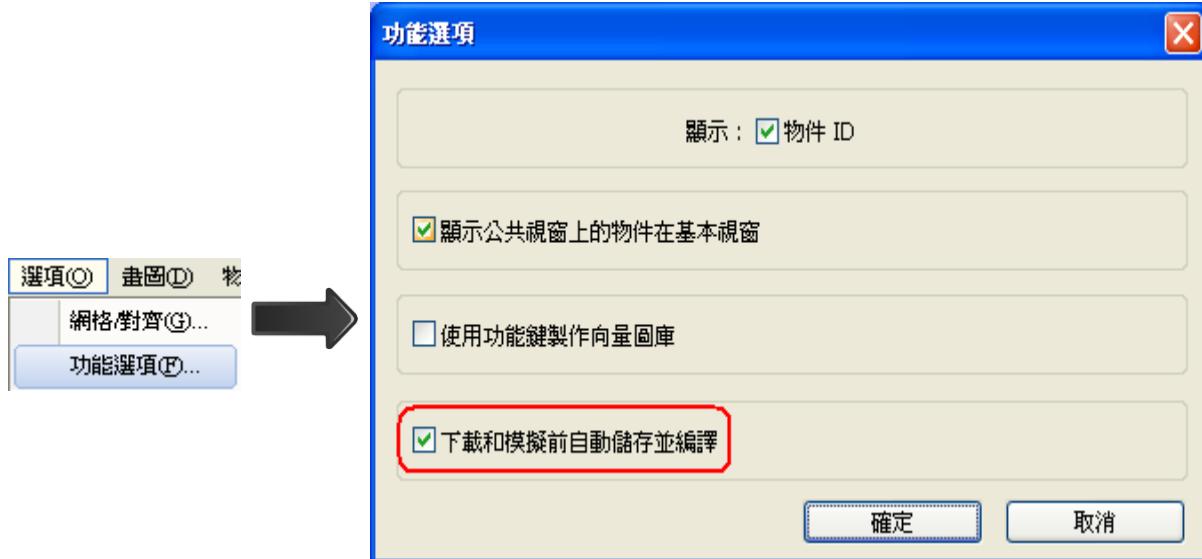
[下載後啟動程式畫面]

此選項如被勾選，下載程序完成後會自動重新啟動 HMI。

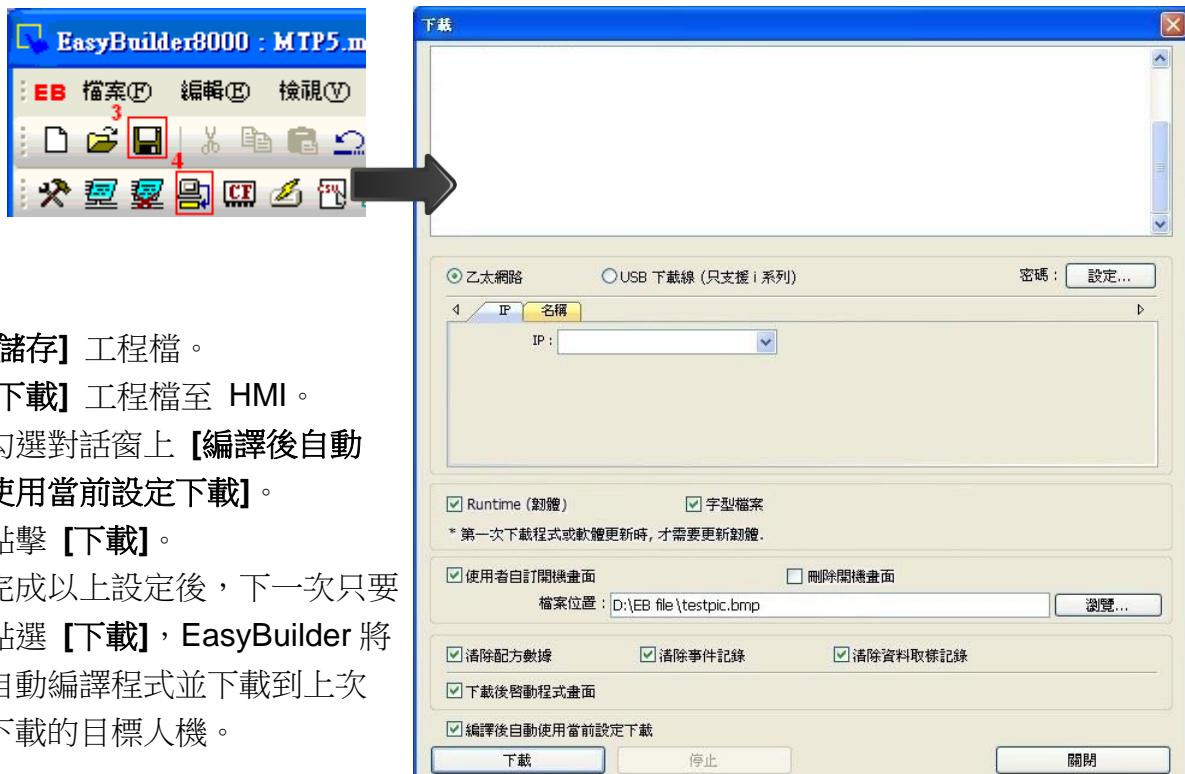


[編譯後自動使用當前設定下載]

如果勾選此項，下一次只要點選【下載】，EasyBuilder 將自動編譯程式並下載到上次下載的目標人機，請見下方說明：



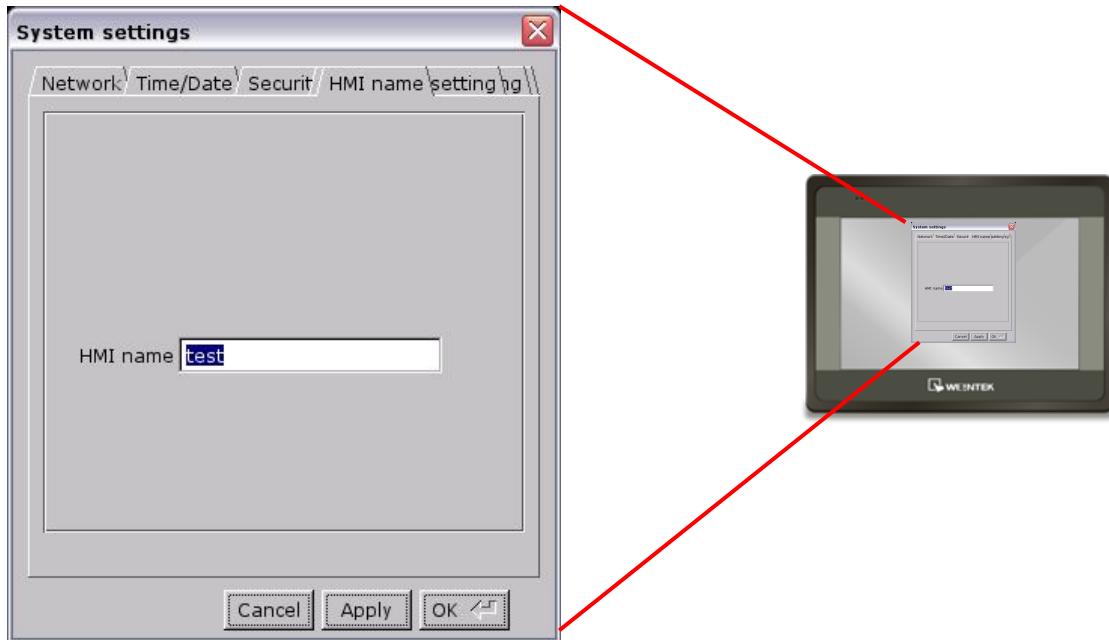
1. 點選【功能選項】。
2. 將【下載和模擬前自動儲存並編譯】啟用。



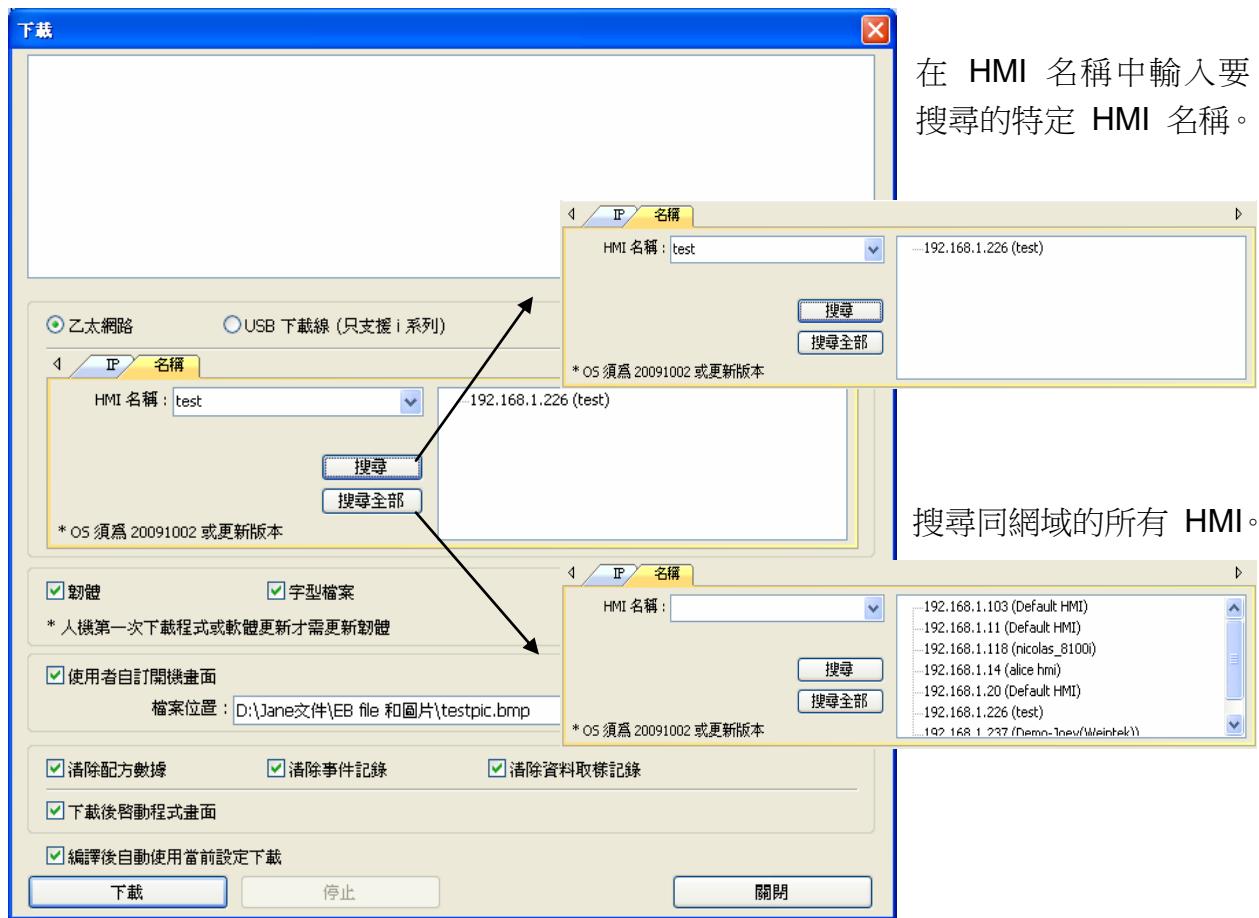
3. 【儲存】工程檔。
4. 【下載】工程檔至 HMI。
5. 勾選對話窗上【編譯後自動使用當前設定下載】。
6. 點擊【下載】。
7. 完成以上設定後，下一次只要點選【下載】，EasyBuilder 將自動編譯程式並下載到上次下載的目標人機。

■ 方法 2 [乙太網路] » HMI 名稱

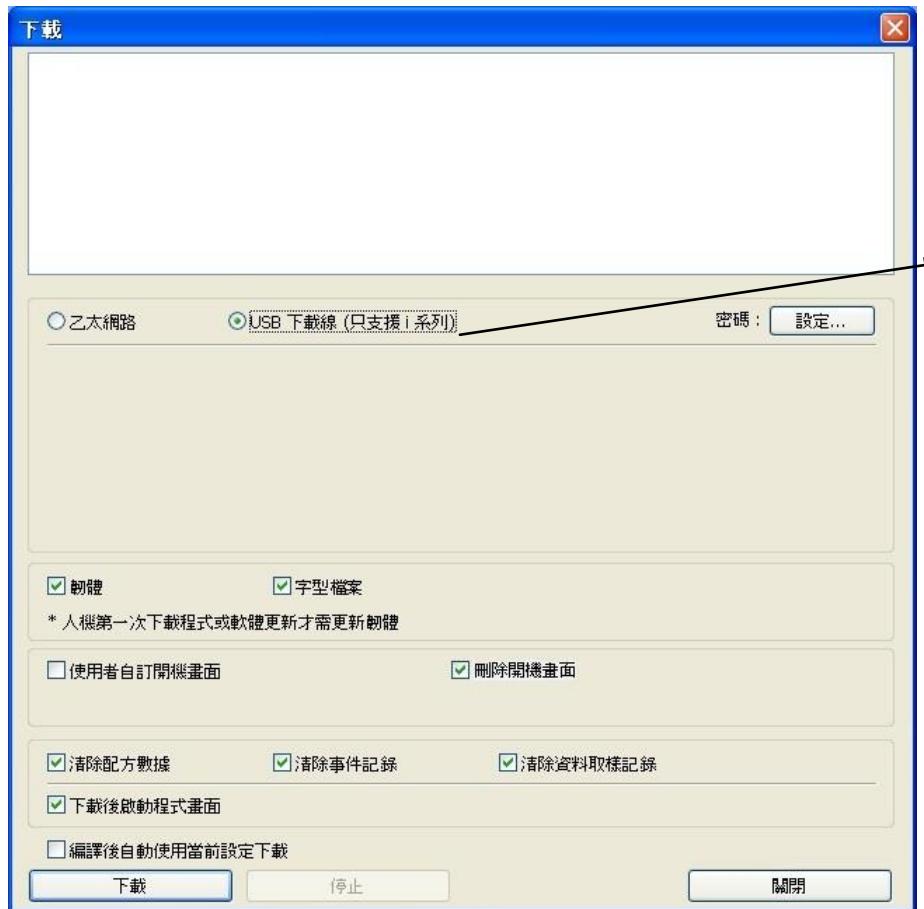
- 在人機上的 system setting 先設定 HMI name。



- 在電腦上，選擇先前設定的 HMI 名稱並開始下載。



■ 方法 3 [USB 下載線]

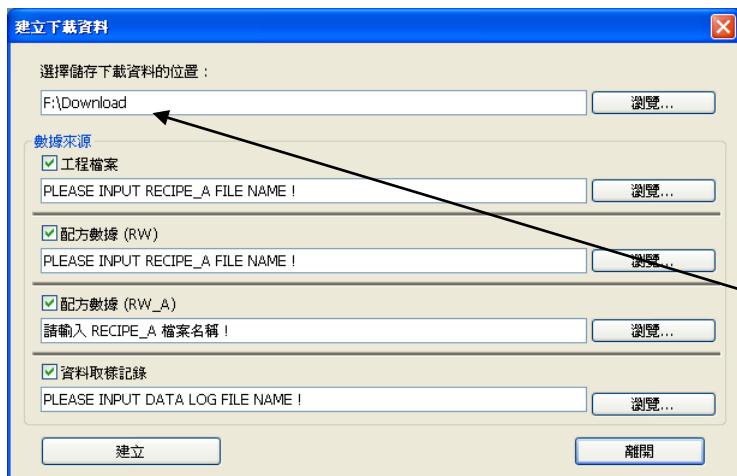


選擇 USB 線下載程式，其餘設定解說同
“方法 1”。

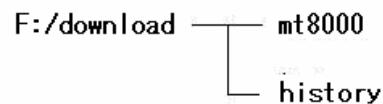


- 使用 USB 線傳輸程式前，可至 **[電腦管理] » [裝置管理員]** 確認 USB 驅動是否安裝完成，若尚未被安裝，請參閱 [安裝步驟](#) 手動完成安裝。

■ 方法 4 [USB 碟 / SD 卡下載]



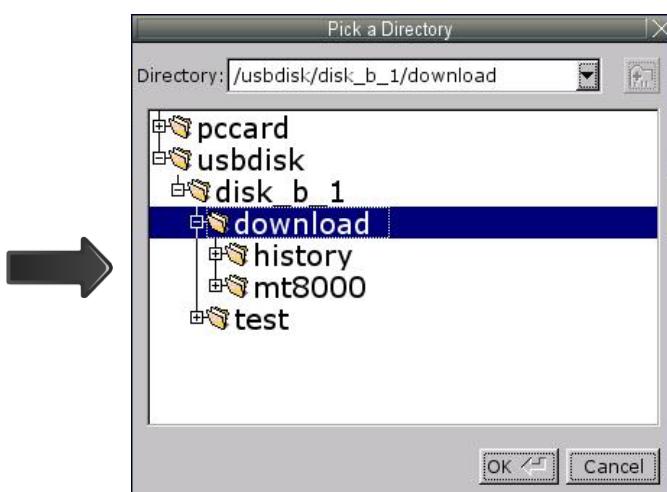
1. 使用 Project Manager 中的 **[建立使用在 USB 碟與 SD 卡所需的下載資料]** 先建立要下載的資料。一般分為兩個目錄，如果設定如左，則下載資料的存放結構為：



存放工程檔案需要下載歷史資料時，會產生此目錄。



2. 插入外部裝置如 SD 卡或 USB 碟至 HMI。
3. 選擇 **[Download]**，輸入密碼。
4. 密碼確認後會顯示外部裝置下的目錄名稱。
(pccard : SD 卡 ; usbdisk : USB 碟)
5. 選擇工程檔存放路徑按 **[OK]**，開始下載。

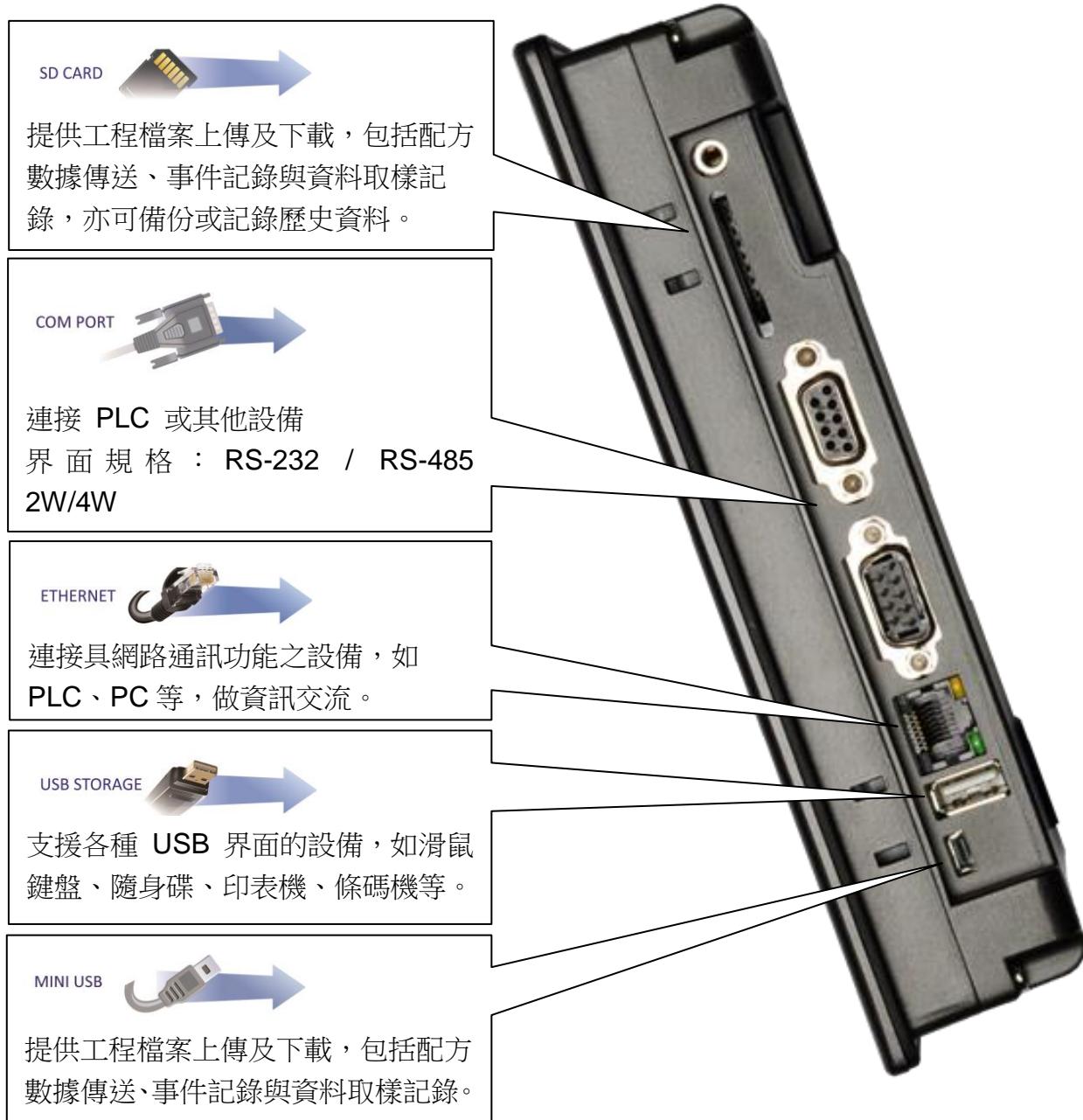


- 此時下載檔案時必須選擇存放下載資料的最上層路徑，也就是說，若以上圖為例，必須選擇 **download**，不可選擇 **history** 或 **mt8000**。

第四章 硬體設定

4.1 I/O 埠

HMI 的支援的連接界面 (依機種不同而相異)：

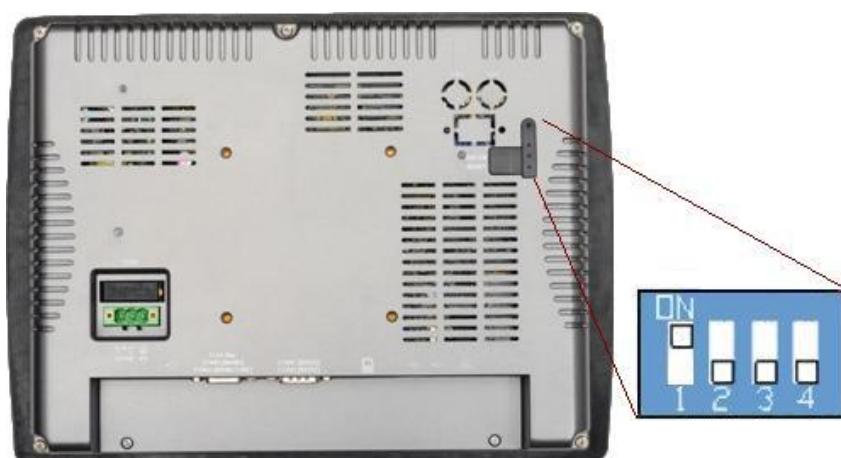


此外，威綸公司也提供 [MT8 - 1 對 2 接頭] 和 [MT8 - 1 對 3 接頭] 兩組連接線，將 HMI 上的串列埠擴展成為各自獨立的通訊埠，給使用者更方便更有效率的操作環境。

4.2 系統設定

當首次操作 HMI 前，必須先在 HMI 上完成以下各項系統設定，設定完成後即可使用 EasyBuilder 編輯軟體開發個人專屬的工程檔案。

4.2.1 系統重置



每台 HMI 背後都有一組重置按鈕及指撥開關，做不同模式切換時，將可觸發對應功能。若忘記 HMI 的系統設定密碼時，可以將 DIP Switch 1 切至 ON，其餘指撥開關保持為 OFF，然後重新啟動 HMI。

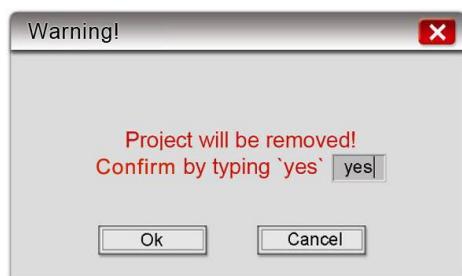
此時 HMI 將進入觸控校正模式，觸控校正完畢後系統將詢問是否要恢復為出廠預設值。



1. 在螢幕會出現一個“+”。使用觸控筆或者手指點選這個“+”的中心點進行五點校正。所有十字皆被準確觸控之後，“+”游標會消失。校準參數會保留在系統裡。



2. 完成校正動作後，系統會詢問使用者是否將 HMI 的系統設定密碼回復為出廠設定，選擇 [Yes]。

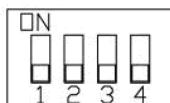


3. 再次確認使用者是否要將 HMI 的系統設定密碼回復為出廠設定，輸入 [yes] 按下 [OK]。HMI 內的程式檔案及所儲存的歷史資料將全部被清除。

(出廠時的 **[Local Password]** 預設密碼為 111111；但其他密碼，包括下載與上傳所使用的密碼於選擇恢復出廠值設定後，皆需重設)。

上述的說明為 T 和 i 系列的恢復原廠設定步驟，若要將 X 系列恢復至原廠設定，使用者需接 USB 鍵盤，在一開機後出現的第一個畫面即按任意鍵(或空白鍵)進入選單，選擇 **Factory Mode** 後，待系統進入程式畫面前即會彈出上述之視窗。若怕開機第一畫面時機點不好抓，使用者可以在一開機即一直持續按空白鍵確保進入系統設定成功。

Dip Switch



SW1	SW2	SW3	SW4	模式
ON	OFF	OFF	OFF	螢幕觸控校正模式 (X 系列除外)
OFF	ON	OFF	OFF	隱藏系統工具列 (T 系列除外)
OFF	OFF	ON	OFF	Boot 載入模式 (X 系列除外)
OFF	OFF	OFF	ON	開啟前面板電源開關 (限 X 系列)
OFF	OFF	OFF	OFF	正常模式



- Dip Switch 4 不一定為 ON 或是 OFF，需依照所使用的 LCD 為主，若所使用的 LCD 需將 Dip Switch 4 設 ON 的話，則 HMI 出廠時，Dip Switch 4 會撥到 ON 並且會將開關剪掉。若所用的 LCD 需將 Dip Switch 4 設 OFF 的話，則 HMI 出廠時，Dip Switch 4 會撥到 OFF 並且保留開關。

4.2.2 系統工具列

啟動 HMI 後可利用在螢幕下方的 **[工具列]** 做系統設定，一般情況下它是自動隱藏的，使用者只需點擊螢幕右下角底端即會彈出工具列，如圖示：



只有 X 系列配有校正模式捷徑，其他系列請用指撥開關 1 啟動。當 X 系列觸控已偏移，可接 USB 滑鼠選擇此模式。



■ 如何隱藏 HMI 系統設定列

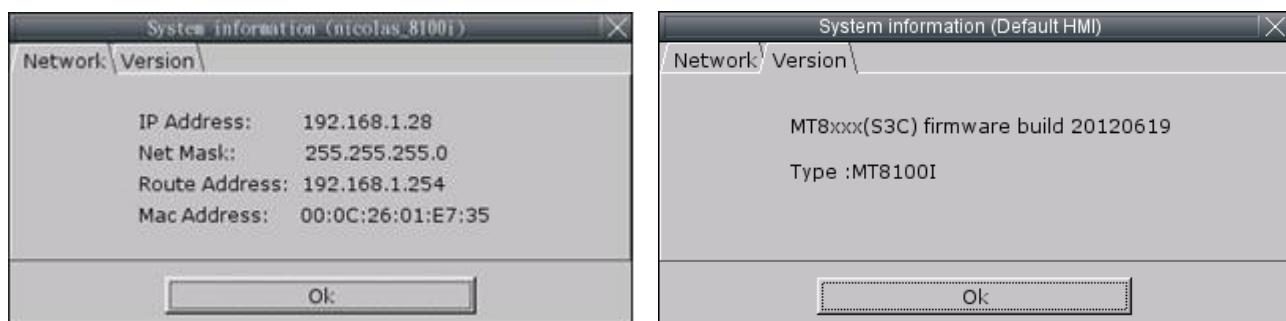
當 [Dip Switch 2] 設為 ON，系統設定列會被隱藏，設為 OFF，系統設定列便可被顯示並控制。使用者需重啟 HMI 來啟用/停止這個功能。

另外可使用系統暫存器 [LB-9020] 來顯示/隱藏系統設定列，當 [LB-9020] 設為 ON，此工具列會被顯示，當設為 OFF，則會被隱藏。

注意：[LB-9020] 可被用於所有系列 HMI。[指撥開關 2] 可被用於 i, X 系列 HMI。

4.2.3 系統資訊

Network: 顯示網路資訊，包含 HMI IP 位址等資訊。Version: 顯示 HMI 系統版本資訊。



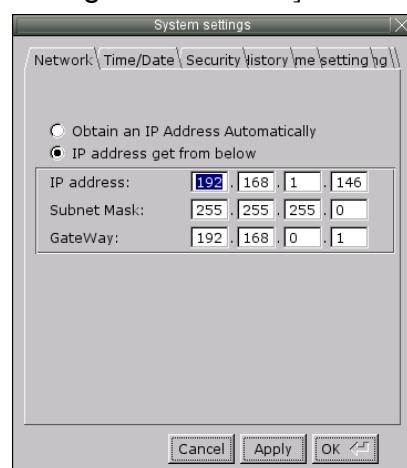
4.2.4 系統設定

設定或變更 HMI 的各項系統參數，基於安全考量必須進行密碼確認。



■ Network

用乙太網路下載工程檔案到 HMI 上，需正確設定操作對象(HMI)的 IP 位址。
[Obtain an IP Address Automatically] 或
[IP address get from below]。



■ Security

HMI 的密碼防護，預設密碼為 111111。



[進入系統設定的密碼]

[上傳工程檔案的密碼]

[下載工程檔案的密碼]

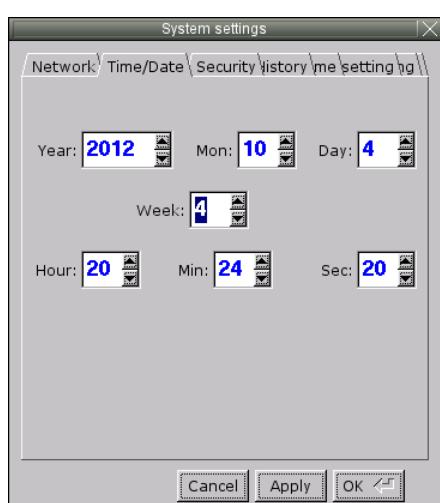
[上傳歷史記錄的密碼]

密碼確認視窗



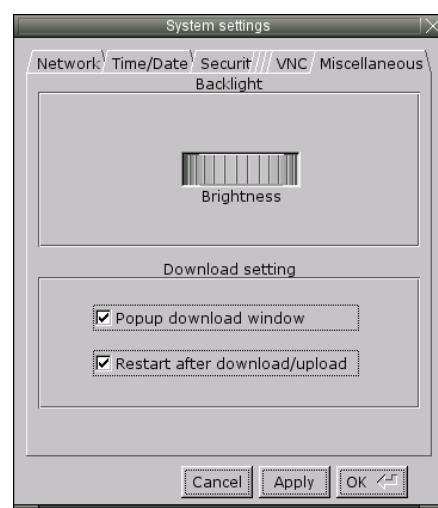
■ Time/Date

設定 HMI 內本地的日期時間。



■ Miscellaneous

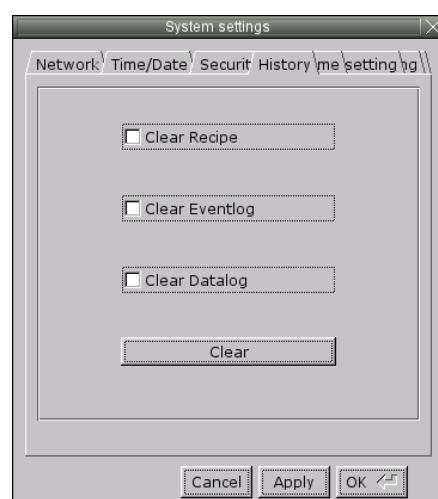
旋鈕可調整 LCD 畫面亮度。



■ History

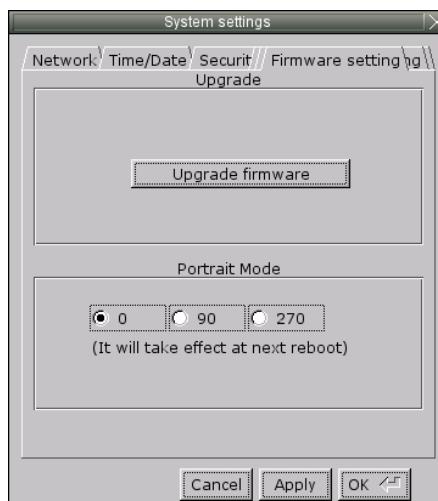
清除存於 HMI 內的歷史記錄：

[配方數據] / [事件記錄] / [資料取樣]



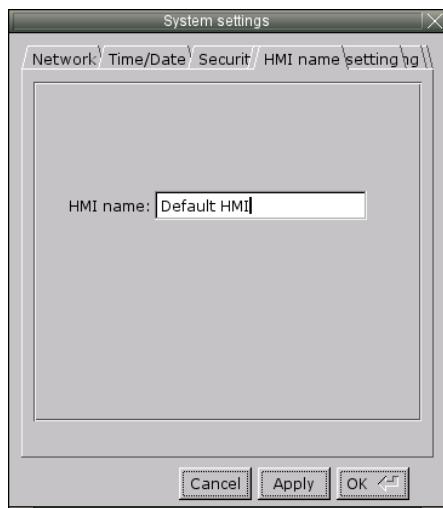
■ Firmware setting

更新系統韌體及啟用直立模式。



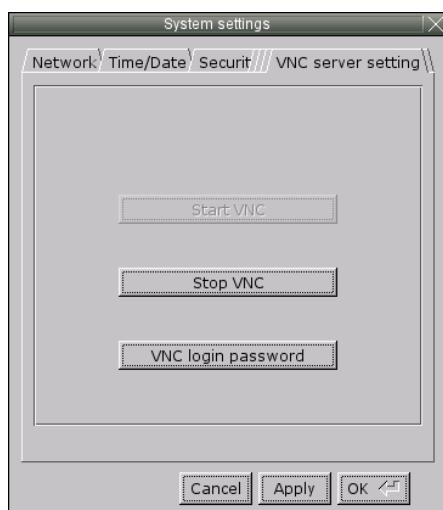
■ HMI name

設定 HMI 名稱以便下載/上傳工程檔案。



■ VNC server

啟用後，可透過乙太網路監控遠端 HMI。



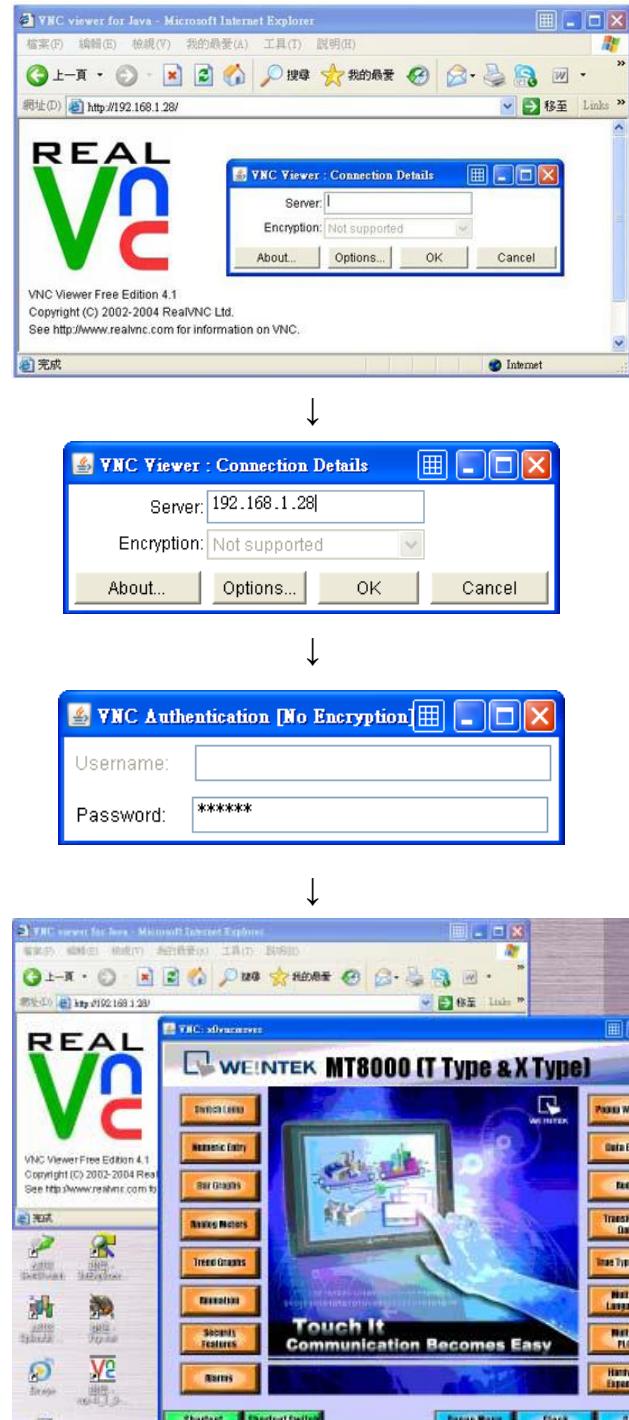
Step 1. 開啟 HMI 的 VNC server 並設置登入密碼。

Step 2. 安裝 Java IE 或 VNC viewer 於 PC。

Step 3-1. 安裝 Java IE 後可透過 IE 輸入遠端 HMI IP 位址，例如：

<http://192.168.1.28>。

Step 3-2. 或者透過 VNC viewer 輸入遠端 HMI IP 位址和密碼。



■ 一台 HMI 同時間只能允許一個使用者登入 VNC server，若持續一小時沒有操作，系統將自動登出。

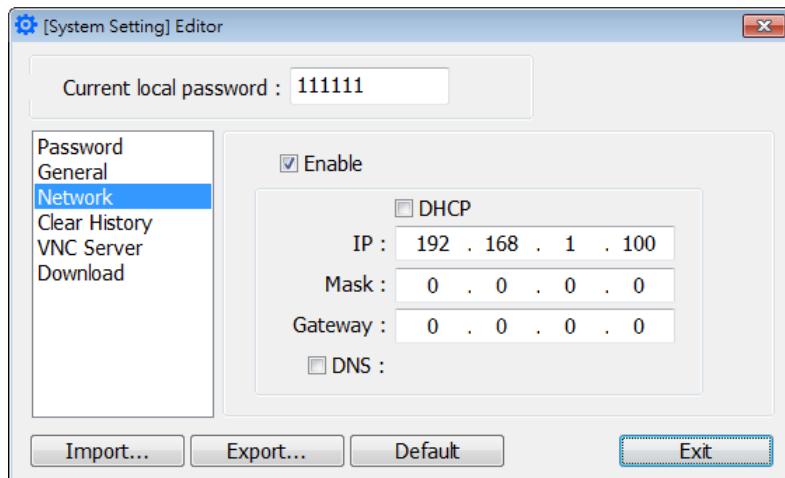
4.3 System Setting Editor

System Setting Editor 提供使用者利用 USB 碟或 SD 卡更新硬體設定資訊的功能。HMI 的韌體版本需使用 OS 20141119 或更新版本。以下介紹如何使用 USB 碟或 SD 卡更新 HMI 的 IP。

- 點選 [工具] » [建立使用在 USB 碟與 CF/SD 卡所需的下載資料]，勾選 [使用系統設定]。



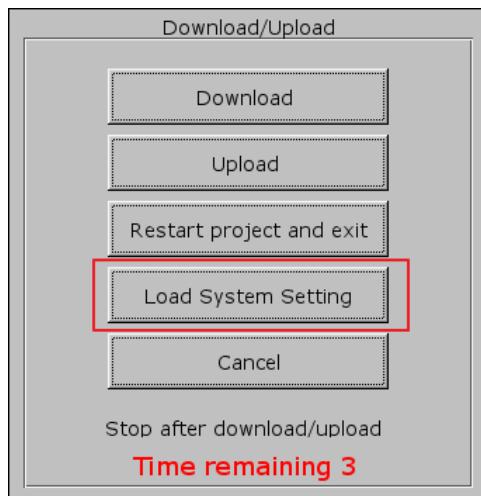
- 按下 [系統設定] 按鈕開啟 System Setting Editor，將 HMI 的 Network 資訊更新。



設定	描述
Import	將已儲存的.conf 檔案匯入並編輯。
Export	將完成設定的資訊匯出成.conf 檔案。
Default	將所有設定回復為預設值。

- 點選 [Export] 將產生一個 systemsetting.conf 檔案。
- 點選 [Exit] 關閉 System Setting Editor。
- 點選 [建立使用在 USB 碟與 CF/SD 卡所需的下載資料] 中的 [建立] 產生 USB 碟與 SD 卡下載檔案。

6. 將儲存該檔案的裝置插入 HMI，將彈出 Download/Upload 對話窗。



點選 [Load System Setting] 將顯示 [Download Config Settings] 訊息。完成更新系統設定後，將繼續更新工程檔案。

第五章 系統參數設定

在點選 EasyBuilder 表單【編輯】下的【系統參數設定】後，可以得到【系統參數設定】對話視窗。



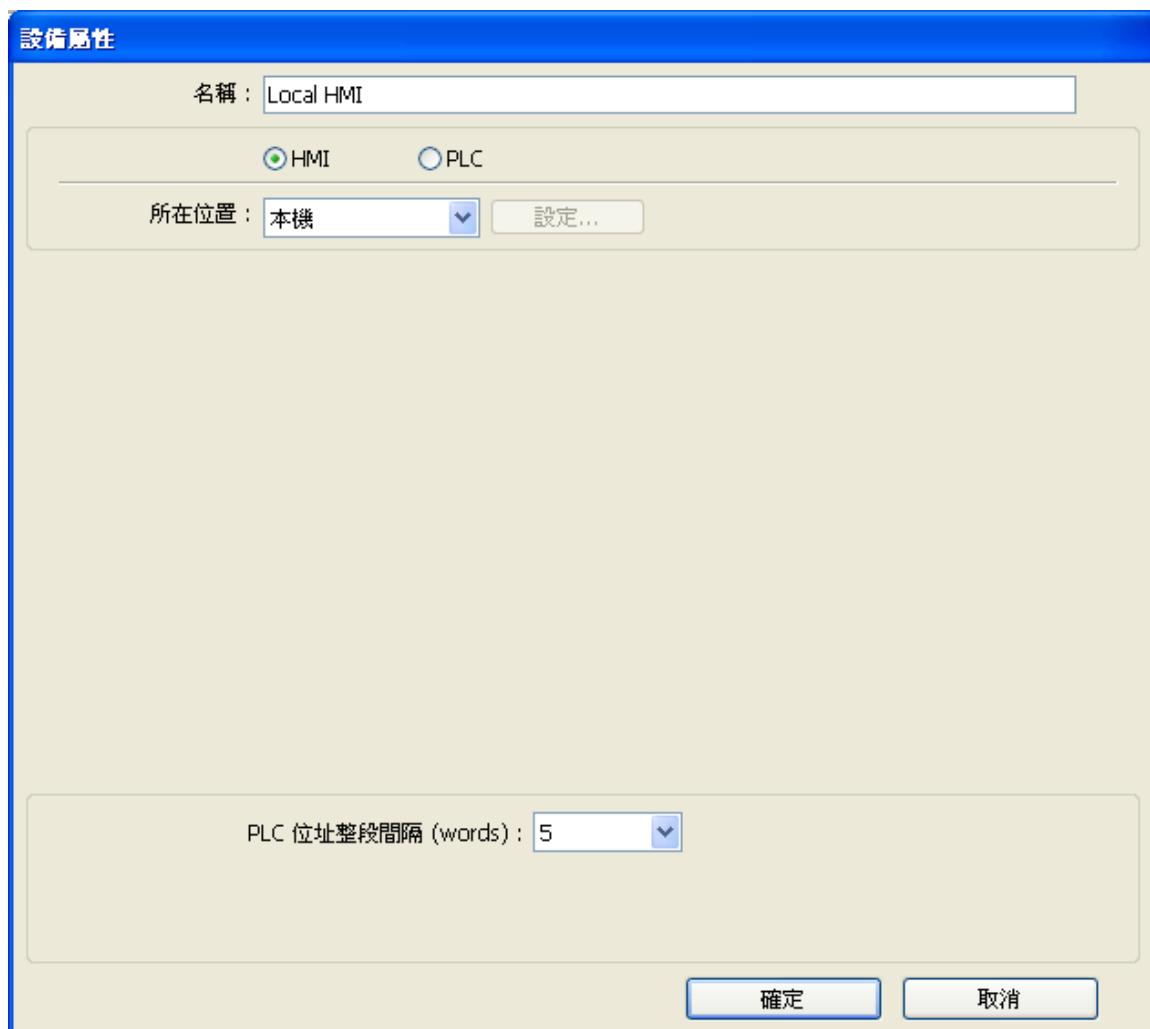
系統參數共分為 **【設備清單】**、**【HMI 屬性】**、**【一般屬性】**、**【系統設定】**、**【使用者密碼】**、**【字體】**、**【擴展記憶體】**、**【列印/備份伺服器】** 等部分，下面將說明各部分的內容。

5.1 設備清單

用來設定 HMI 連接裝置之屬性，這些裝置包括本機或遠端 HMI 和 PLC。

當開啟新的工程檔案時，會預設存在一個“Local HMI”裝置，用來辨識此機型。

若要修改設備內容，點選【系統參數設定】對話窗的【設定】，即可開啟【設備屬性】視窗，如下圖所示。

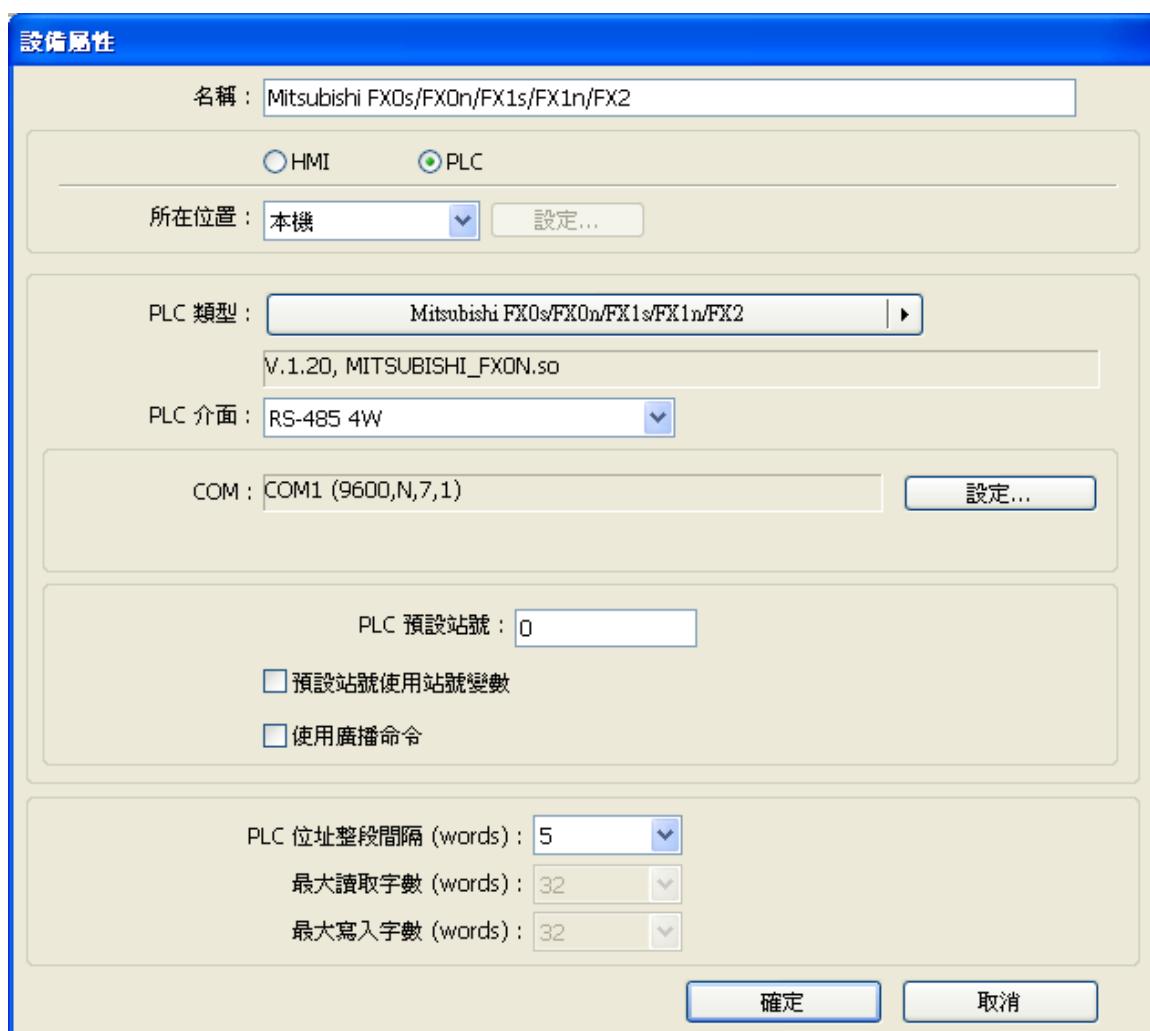


5.1.1 如何控制一台本機 PLC



“本機 PLC”是指本地 HMI 直接連接 PLC，若要控制本機 PLC 時，需先新增此種類型的裝置，點選【系統參數設定】對話窗的【新增】，即可開啟【設備屬性】視窗。

以 Mitsubishi FX0s/FX0n/FX1s/FX1n/FX2 為本機 PLC 為例：



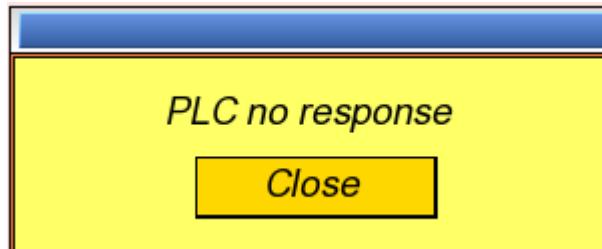
各項設定的說明如下：

設定	敘述
名稱	裝置名稱
HMI 或 PLC	此範例的連接裝置為 PLC，所以此時選擇 [PLC]。
所在位置	可以選擇 [本機] 或 [遠端]，此範例 PLC 連接在本機 HMI 上，所

	以選擇 [本機] 。
PLC 類型	選擇 PLC 的型號。
PLC 介面	<p>PLC 所使用的介面類型，可以選擇 [RS-232]、[RS-485 2W]、[RS-485 4W]、[乙太網路] 以及 [USB]。</p> <p>介面類型如果為 [RS-232]、[RS-485 2W]、[RS-485 4W]，點選 [設備屬性] 的 [設定]，可以開啟 [通訊埠設定] 對話視窗，用來設定通訊參數。</p> 

[超時]

通訊中斷超過此項設定值(單位為秒)，HMI 會使用 5 號視窗做為“PLC No Response”的提示。

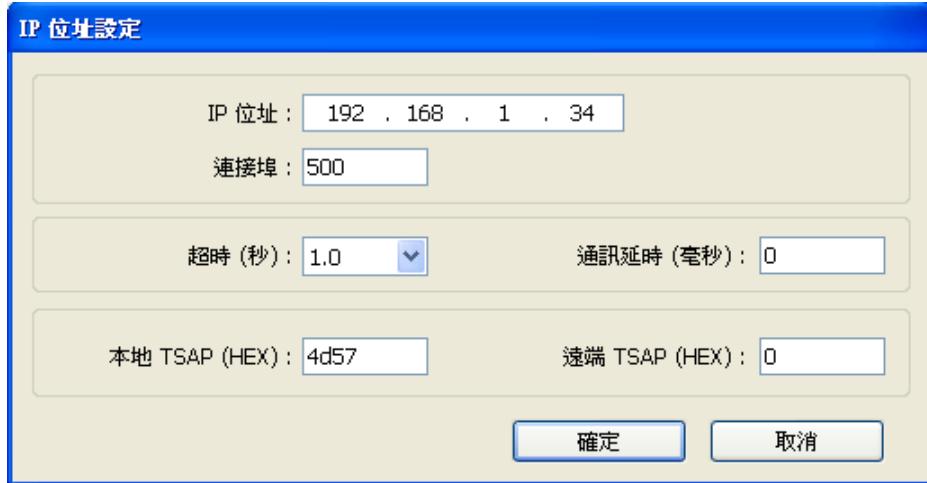


[通訊延時]

HMI 在送出下一個命令給 PLC 前，會刻意先延遲此項設定值(單位為毫秒)，再送出命令。此項設定值會降低 HMI 與 PLC 間的通訊效率，若無特殊需求，此項設定值設定為“0”即可。

若使用的 PLC 為 **Siemens S7-200** 系列，則不能忽略此項設定值，建議將 **[通訊延時]** 設定為“5”，**[ACK 訊號延時]** 設定為“30”。

介面類型如果為 **[乙太網路]**，點選 **[設備屬性]** 的 **[設定]** 即可開啟 **[IP 位址設定]** 視窗，使用者必須正確設定 PLC 的 IP 位址與連接埠。

	 <p>介面類型如果為 [USB]，就不需再設定，檢查 【設備屬性】 的各設定值是否正確即可。</p>
PLC 預設站號	<p>PLC 位址所使用的預設站號。當位址內容不包括站號資訊時，將使用此項設定值做為 PLC 的站號。</p> <p>也可將 PLC 站號資訊直接設定在位址內容中，此時位址格式為 ABC#DEFGH</p> <p>其中 ABC 表示 PLC 所使用的站號，必須大於等於 0，且小於等於 255。DEFGH 用來指定 PLC 的位址，兩個數據之間以 “#” 做區隔。</p> <p>以下圖為例，顯示此時將讀取 PLC 站號為 1 的 T20 位址之內容。</p> 
預設站號使 用站號變數	<p>PLC 預設站號可以選擇使用站號變數。</p> <p>可以利用 LW10000~LW10015 來設定站號變數。</p> <p>當 PLC 預設站號選擇使用站號變數時，若在 PLC 位址中未指定所使用的站號，則站號一律由預設站號所指定的站號變數來決定。</p>

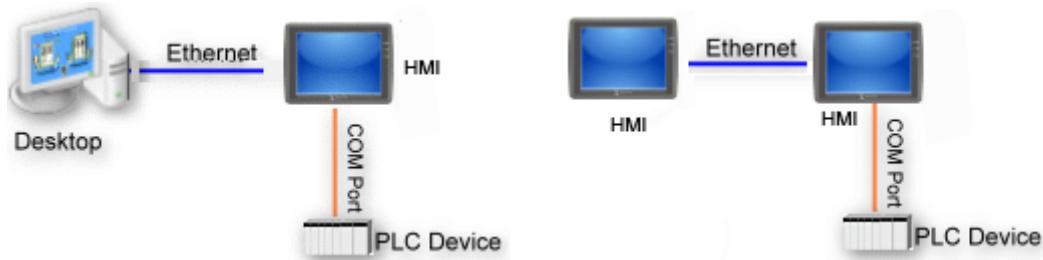
	<p>若 PLC 預設站號已選擇 var3：</p> <p>以下使用幾個例子說明。</p> <p>a. 所操作的 PLC 站號為 5</p> <p>b. 所操作的 PLC 站號由 var7 (LW-10007) 來決定</p> <p>c. PLC 的位址為 "111"，此時不指定 PLC 站號，但因為已使用預設站號 var3，所以所操作的 PLC 站號由 var3 (LW-10003) 來決定。</p>
使用廣播命令	<p>當勾選 【使用廣播命令】 後，依照所使用的 PLC 定義之廣播站號，填入至 【廣播命令所使用的站號】。當人機使用廣播站號發送命令時，PLC 將只接收命令而不回覆給人機。</p> <p>以下圖為例：</p> <p>假設廣播站號為 255，當人機發送命令至 255#200 這個位址時，所有的 PLC 會接收這個命令並且不回應給人機。</p> <p>note: 有支援廣播命令的 PLC 才適用此功能。</p>

PLC 位址整段間隔 (words)	不同讀取命令的讀取位址之間距若小於此項設定值，這些命令可以合併為同一筆命令。此項設定值如果設定為“0”，將取消命令合併功能。 若此項設定值為“5”，當分別需從 LW3 讀取 1 個 word 與從 LW6 讀取 2 個 word 的數據(即 LW6 與 LW7 的內容)時，因 LW3 與 LW6 的位址差距小於 5，此時可以將兩個命令合併為 1 個命令，合併後的命令內容為從 LW3 開始連續讀取 5 個 word 的數據(讀取 LW3~LW7)。需注意，可以被合併的命令之讀取數據大小將不會大於 【最大讀取字數 (words)】 。
最大讀取字數 (words)	一次可以從裝置讀取數據的最大量，單位為 word。
最大寫入字數 (words)	一次可以寫入到裝置的數據最大量，單位為 word。

完成上述的各項設定後，在 **【設備清單】** 中可以發現新增了一個名稱為 “本機 PLC 1”的裝置。



5.1.2 如何控制一台遠端 PLC



“遠端 PLC”是指遠端 HMI 直接連接的 PLC，若要控制遠端 PLC 需先增加此種類型的裝置。點選【系統參數設定】對話窗的【新增】，即可開啟【設備屬性】視窗設定各項屬性。以 Siemens S7-200 為遠端 PLC 為例：



各項設定的說明如下：

設定	敘述
HMI 或 PLC	連接裝置為 PLC，所以此時選擇 [PLC] 。
所在位置	可以選擇 [本機] 或 [遠端] ，因 PLC 連接在遠端 HMI 上，所以此時選擇 [遠端] ，並且必須設定遠端 HMI 的 IP 位址及連接埠。請在 [設備屬性] 視窗按下 [設定] 。
	
PLC 種類	選擇遠端 PLC 的型號。
PLC 介面	遠端 PLC 所使用的介面類型，若遠端 PLC 使用 COM 埠時，介面需選擇 [RS-232] ， [RS-485 2W] ， [RS485 4W] 任一種。
PLC 預設站號	遠端 PLC 所使用的站號。
COM	遠端 PLC 在遠端 HMI 上所使用的 COM 埠，此項內容必須正確設定。

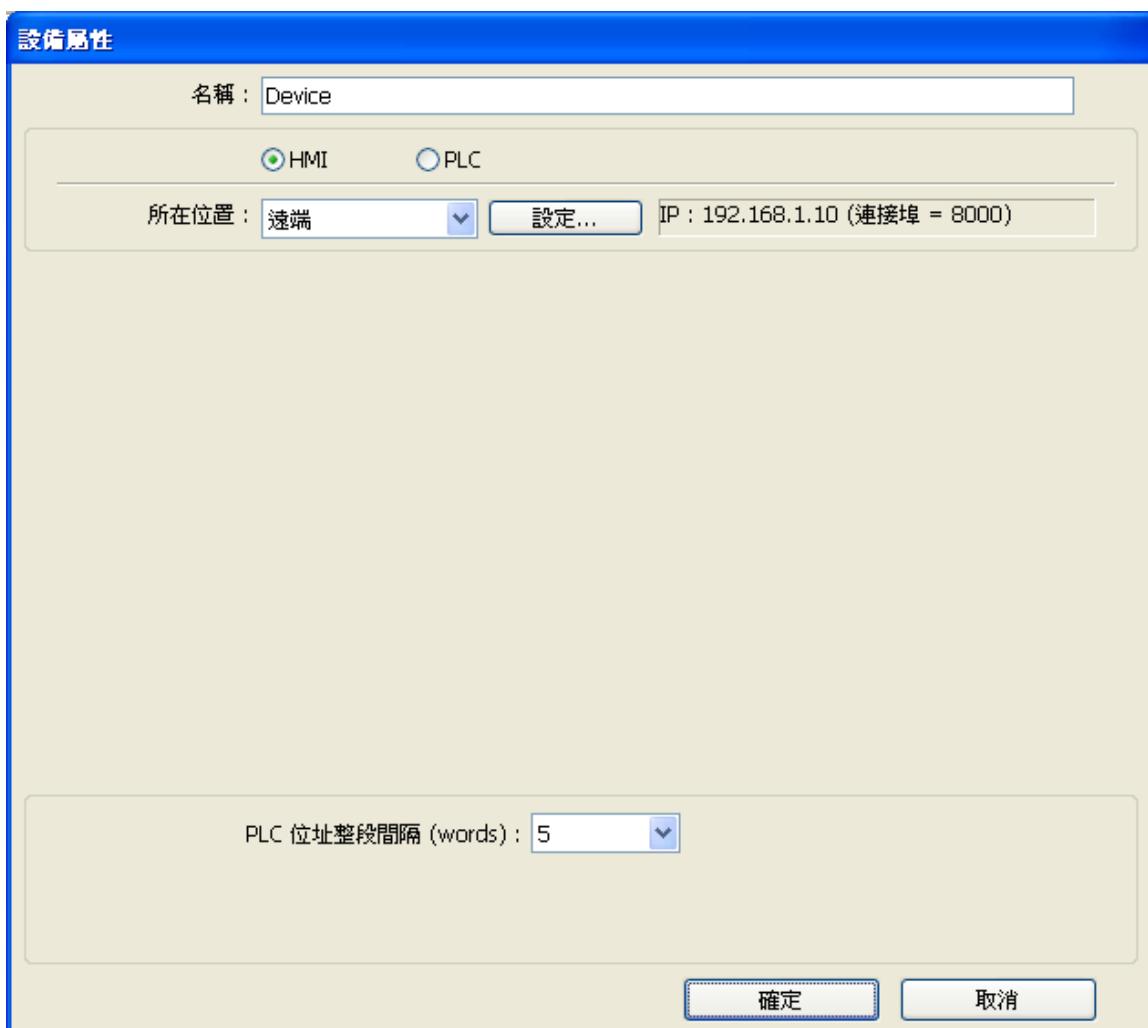
完成上述的各項設定後，在 **[設備清單]** 中可以發現新增了一個名稱為 “遠端 PLC 1”的裝置。



5.1.3 如何控制一台遠端 HMI



“遠端 HMI”是指非本地 HMI 的所有其他 HMI，PC 也被視為遠端 HMI 的一種，若要控制遠端 HMI 需先新增此種類型的裝置。點選【系統參數設定】對話窗的【新增】，即可開啟【設備屬性】視窗設定各項屬性。



各項設定的說明如下：

設定	敘述
HMI 或 PLC	連接裝置為 HMI，所以此時選擇 [HMI] 。
所在位置	可以選擇 【本機】 或 【遠端】 ，因為使用遠端 HMI，此時選擇 【遠端】 ，並且必須設定遠端 HMI 的 IP 位址及連接埠。請在 【設備屬性】 視窗按下 【設定】 。



完成上述的各項設定後，在 **【設備清單】** 即新增了一個為“遠端 HMI 1”的裝置。



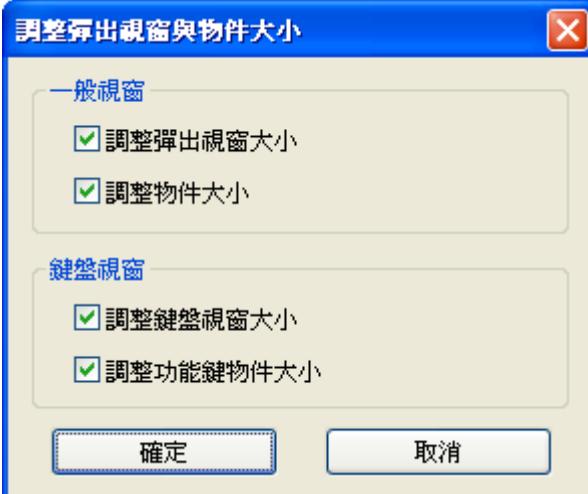
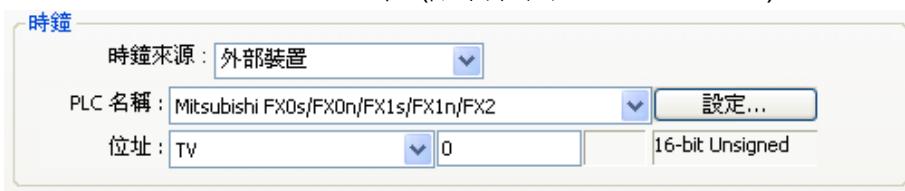
5.2 HMI 屬性

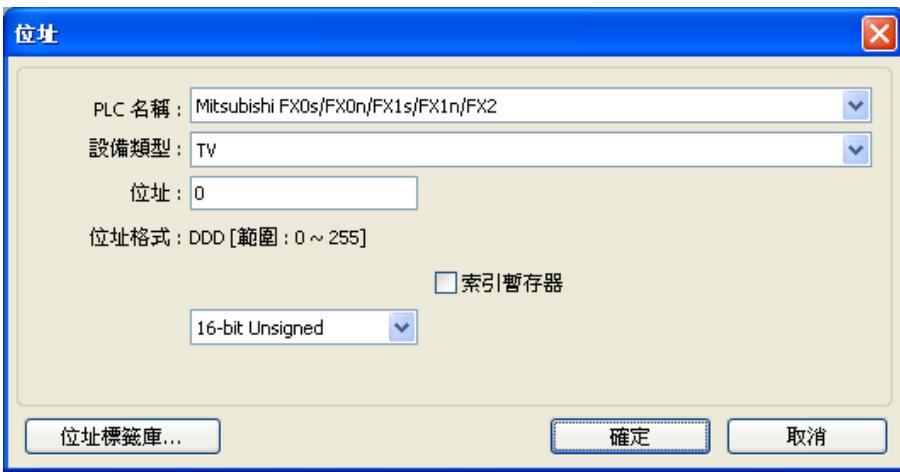
[HMI 屬性] 設定頁用來設定 **[HMI 型號]**、**[時鐘]** 的來源、**[印表機]** 相關的設定、以及 **[捲動軸]** 寬度設定。



各項設定的說明如下：

設定	敘述
HMI 型號	請選擇要使用的 HMI 之型號 HMI 型號 : MT6070iH/MT8070iH/MT6100i/MT8100i/WT3010 (800 x 480) HMI 站號 : MT6056T/MT8056T (320 x 234) MT6070T/MT8070T (480 x 234) MT6104T/MT8080T/MT8104T (640 x 480) MT8121T (800 x 600) MT8104X (640 x 480) MT8104XH/MT8121X (800 x 600) MT8150X (1024 x 768) MT6056i (320 x 234) MT6050i/MT8050i (480 x 272) MT6070iH/MT8070iH/MT6100i/MT8100i/WT3010 (800 x 480) MT8104iH (800 x 600)

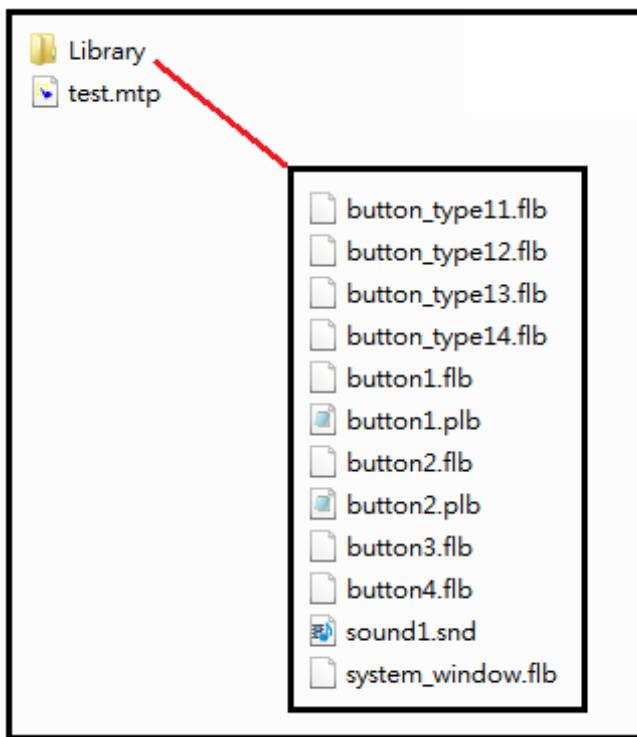
	<p>當使用者更換其他的 HMI 型號並按確認鍵時，【調整彈出視窗與物件大小】的對話窗會出現如下圖。按下確認後即完成變更 HMI 型號的設定。</p> 																		
HMI 站號	選擇 HMI 所使用的站號，若無特殊目的，採用預設值即可																		
連接埠	設定 HMI 所使用的通訊埠號碼，此號碼也用於 MODBUS 同伺服器，若無特殊目的，採用預設值即可																		
時鐘	<p>[時鐘來源] 設定計時器時間訊號的來源，計時器的時間被用在 【資料取樣】，【事件登錄】等需時間紀錄的物件上。</p> <p>a. 當選擇 [HMI RTC] 時，表示時間訊號來自 HMI 內含的計時器。</p> <p>b. 當選擇 [外部裝置] 時，表示時間訊號來自外部的裝置，此時需正確設定時間訊號的來源位址，以下圖的設定為例，表示時間來自“本機 PLC”的 TV，此時位址 TV 從位址 0 開始的連續 6 個 word 內容分別存放下列資訊：</p> <table style="margin-left: 20px;"> <tr><td>TV 0</td><td>-></td><td>秒 (限制範圍： 0~59)</td></tr> <tr><td>TV 1</td><td>-></td><td>分 (限制範圍： 0~59)</td></tr> <tr><td>TV 2</td><td>-></td><td>時 (限制範圍： 0~23)</td></tr> <tr><td>TV 3</td><td>-></td><td>日 (限制範圍： 1~31)</td></tr> <tr><td>TV 4</td><td>-></td><td>月 (限制範圍： 1~12)</td></tr> <tr><td>TV 5</td><td>-></td><td>年 (限制範圍： 1970~2037)</td></tr> </table> 	TV 0	->	秒 (限制範圍： 0~59)	TV 1	->	分 (限制範圍： 0~59)	TV 2	->	時 (限制範圍： 0~23)	TV 3	->	日 (限制範圍： 1~31)	TV 4	->	月 (限制範圍： 1~12)	TV 5	->	年 (限制範圍： 1970~2037)
TV 0	->	秒 (限制範圍： 0~59)																	
TV 1	->	分 (限制範圍： 0~59)																	
TV 2	->	時 (限制範圍： 0~23)																	
TV 3	->	日 (限制範圍： 1~31)																	
TV 4	->	月 (限制範圍： 1~12)																	
TV 5	->	年 (限制範圍： 1970~2037)																	

	 <p>位址</p> <p>PLC 名稱 : Mitsubishi FX0s/FX0n/FX1s/FX1n/FX2</p> <p>設備類型 : TV</p> <p>位址 : 0</p> <p>位址格式 : DDD [範圍 : 0 ~ 255]</p> <p><input type="checkbox"/> 索引暫存器</p> <p>16-bit Unsigned</p> <p>確定 取消</p>
印表機	<p>型號</p> <p>顯示目前支援的印表機類型，其中 HP PCL Series 需使用 USB 埠連接，其他類型印表機需使用 COM 埠連接。更詳細的機型描述請參考《第 23 章 印表機支援種類》。</p> <p>印表機</p>  <p>型號 : HP PCL 系列 (USB) * 只支援 USB</p> <p>紙張大小 : 無 SP-M, D, E, F EPSON ESC/P2 系列 HP PCL 系列 (USB) Axiohm A630 SPRT (SP-DIII, DIV, D5, D6, A, DN, T) EPSON TM-L90 EPSON TM-T70 BRIGHTEK WH-E19 BRIGHTEK WH-C1/C2</p> <p>比例 : 100%</p>
	<p>使用 [COM] 埠連接的印表機需正確設定 COM 埠的通訊參數。當印表機的型號為 [SP-M, D, E, F] 時，需小心設定 [每行點數]，此設定值不可以超過印表機每行可以列印的點數，否則將造成錯誤的列印結果。</p> <p>印表機</p>  <p>型號 : SP-M, D, E, F</p> <p>COM : COM 3</p> <p>傳輸速率 : 19200</p> <p>校驗 : None</p> <p>每行點數 : 100 像素</p> <p>數據位元 : 8 Bits</p> <p>停止位元 : 1 Bit</p> <p>螢幕列印縮放比例 : 100%</p> <p>* 100 像素 (1610 型號) 或 220 像素 (2407, 4004 型號)</p>
捲動軸	<p>寬度</p> <p>分為小、中、大，可依照所需捲動軸大小設定。</p> <p>設定捲動軸的尺寸，當物件的尺寸不足以顯示其內容時，捲動軸將顯示於相關物件上。</p>

存檔時儲存
至工程檔案
相同的目錄
下

向量圖/圖片/聲音/巨集指令函式庫存放位置
 存檔時儲存至工程檔案相同的目錄下

當使用此功能時，儲存工程檔案時向量圖/圖片/聲音/巨集指令函式庫將被儲存到與工程檔案相同目錄下的 Library 目錄中，如下圖所示：



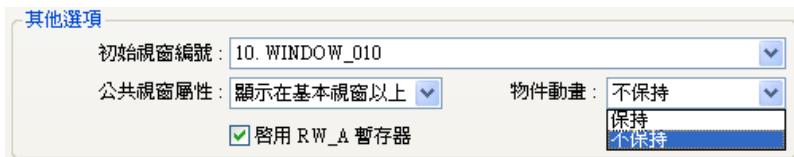
5.3 一般屬性

【一般屬性】 設定頁用來設定與畫面操作有關的各項屬性。



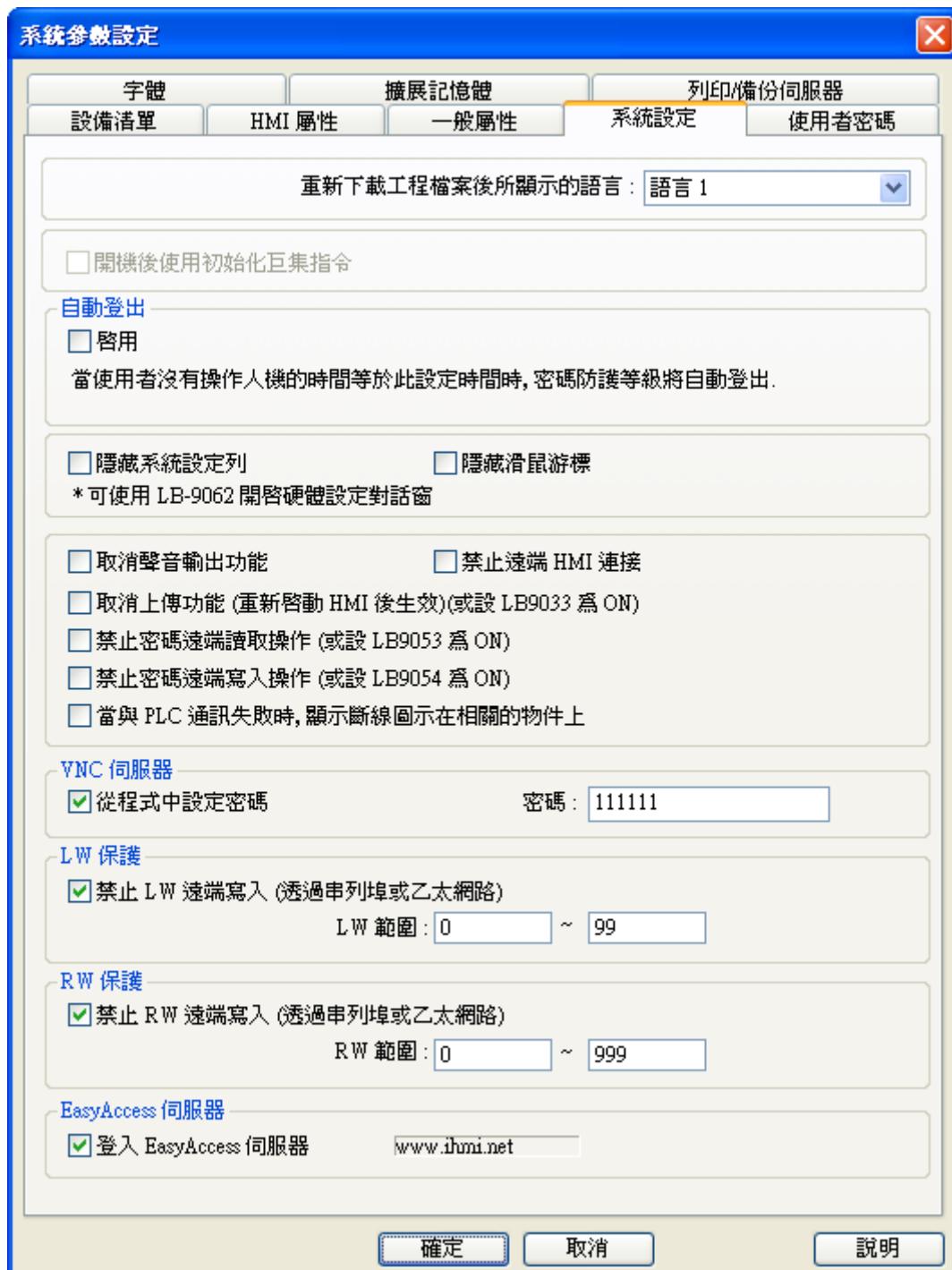
各項設定的說明如下：

設定	敘述
快選視窗按鍵設定	<p>設定快選視窗 (3 號視窗) 的各項屬性，若要使用快選視窗前，需先建立 3 號視窗。</p> <p>a. [屬性]</p>  <p>選擇是否使用快選視窗，在選擇 [啟用] 後，可以點選 [設定] 功能，設定快選視窗按鍵的各項屬性，包括標示於快選視窗按鍵上的顏色與文字。</p> <p>b. [位置]</p>  <p>選擇快選視窗按鈕的出現位置，選擇 [左] 則快選視窗按鈕出現在畫面的左下角；選擇 [右] 則快選視窗按鈕出現在畫面的右下角。</p>
螢幕保護設定	<p>a. [背光節能時間]</p> <p>當未碰觸螢幕的持續時間等於此設定值時，將關閉背光燈，設定的時間單位為分鐘。關閉背光燈後只需碰觸到螢幕，即可重開背光燈。當設定值選擇 [無] 時，HMI 將不使用背光節能的功能。</p> <p>b. [螢幕保護時間]</p> <p>當未碰觸螢幕的持續時間等於此設定值時，將自動切換到 [螢幕保護使用視窗] 所指定的視窗，設定的時間單位為分鐘。當設定值選擇 [無] 時，HMI 將不使用螢幕保護的功能。</p> <p>c. [螢幕保護使用視窗]</p> <p>指定螢幕保護功能執行時要顯示的頁面。</p>
其他選項	<p>a. [初始視窗編號]</p> <p>選擇 HMI 開機後的起始頁面。</p> <p>b. [公共視窗屬性]</p>  <p>公共視窗 (4 號視窗) 內的物件會出現在每個基本視窗中，此選項可設定公共視窗內的物件，是出現在基本視窗原來物件的上層或下層。</p>

	<p>c. [鍵盤游標顏色] 設定使用【數值輸入】或【字元輸入】物件時，輸入欄位中出現的游標顏色。</p> <p>d. [物件動畫]</p>  <p>如果選擇【保持】模式，則 HMI 在運作時，【動畫】與【移動圖形】物件將顯示在其他類型物件的上方，與物件的建立順序無關。如果選擇【不保持】模式，則物件的顯示順序依照物件的建立先後，先建立者先出現。</p> <p>e. [啟用 RW_A 暫存器] 可勾選是否啟用配方資料 RW_A。需在啟用 RW_A 後，物件才可以操作 RW_A 的內容，RW_A 的大小為 64K。</p>
事件	<p>額外事件數 系統預設的事件紀錄總數為 1000 筆，若要增加紀錄的筆數，可以更改此設定值，目前設定的上限為 10000。</p>
鍵盤	<p>顯示鍵盤存在的頁面，這些頁面代表在使用【數值輸入】與【字元輸入】物件時，可以選擇的鍵盤類型，最多可以新增到 32 個鍵盤。</p> <p>使用者要建立自訂的鍵盤時，需先在已存在的頁面規劃好要使用的鍵盤，並使用【新增】功能選擇這些頁面並加入到列表中即可。</p> <p>更詳細的功能請參考《第 12 章 鍵盤的設計與使用》。</p>
工程檔案保護(只支援 i 系列)	使用者的工程檔可被限定在特定的人機上執行 (只限 i 系列人機) 相關訊息請見《第 30 章 工程檔案保護功能》。

5.4 系統設定

[系統設定] 設定頁用來設定 EasyBuilder 中各種功能。



有些功能從系統暫存器複製而來，例如：[隱藏系統設定列 (LB-9020)]、[隱藏滑鼠遊標 (LB-9018)]、[取消聲音輸出功能 (LB-9019)]、[禁止遠端 HMI 連接 (LB-9044)] 以及 [取消上傳功能(LB-9033)]，使用者可以選擇系統暫存器來使用這些功能。

要選擇系統暫存器，使用者可在新增物件時，於設定頁勾選 **【位址】** 旁的 **【系統暫存器】** 並選 **【設備類型】**。

要檢視所有的系統暫存器，可於 EasyBuilder 點選 **【圖庫】** » **【位址標籤庫】** » **【系統暫存器】**。

【重新下載工程檔案後所顯示的語言】

選擇在重新下載工程檔並啟動人機時所顯示的語言。

【開機後使用初始化巨集指令】

指定當人機開機後所執行的巨集指令。

【自動登出】

如果閒置人機的時間超過此設定值，人機中有設定安全等級的物件將無法使用，需要重新再次輸入使用者 ID 及密碼才能操作物件。

【隱藏系統設定列】

將人機右下角系統設定頁關閉。

【隱藏滑鼠游標】

將人機上滑鼠游標關閉。

【取消聲音輸出功能】

將人機聲音關閉。

【禁止遠端 HMI 連接】

遠端 HMI 連接功能關閉，遠端人機將無法操作本機 HMI。

【取消上傳功能(重新啟動 HMI 後生效)(或設 LB9033 為 ON)】

將人機上傳 project 功能關閉，勾選此選項下載後，必須重新開機才會取消上傳 project 功能。

【禁止密碼遠端讀取操作(或設 LB9053 為 ON)】

禁止遠端 HMI 可以讀取本機 HMI 的密碼。

【禁止密碼遠端寫入操作(或設 LB9054 為 ON)】

禁止遠端 HMI 可以寫入本機 HMI 的密碼。

【當與 PLC 通訊失敗時，顯示斷線圖示在相關的物件上】

選擇是否與 PLC 通訊失敗時，使用斷線符號標示在相關物件上。



當使用此項功能且物件無法與 PLC 通訊時，斷線符號會顯示在物件右下角。



[VNC 伺服器]

設定登入 VNC 伺服器所使用的密碼。

[LW 保護]及[RW 保護]

如果使用者勾選 **[禁止 LW/RW 遠端寫入]**，並在 **[LW/RW 範圍]** 內設定所要保護的範圍，此範圍內的數值將不能透過遠端 HMI 調整。

[Easy Access 伺服器]

透過這項技術，使用者可以輕易透過網路監測網路連結的人機，並直接在電腦上操作，就像直接將人機拿在手上一樣。

特別的是，Easy Access 不需傳輸更新的圖檔，只需傳輸即時資料。這使得傳輸更快更有效率。更詳細的資訊請參閱《Easy Access》說明。

5.5 使用者密碼

【使用者密碼】 設定頁用來設定使用者的密碼以及可操控的物件類別。

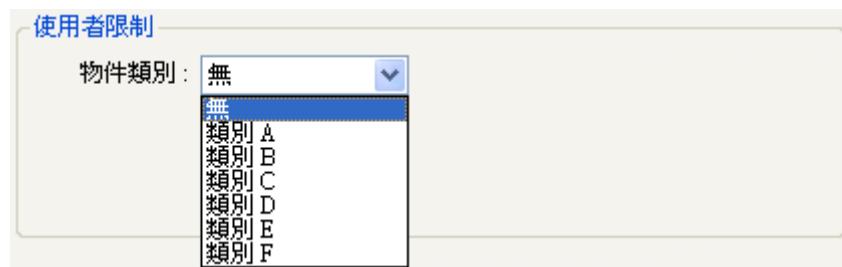
5.5.1 設定用戶可以操作的物件類別與密碼



可設定 12 組用戶密碼。密碼須為非負整數。

機器運作時，用戶在成功輸入密碼後，系統會依照用戶的設定內容決定用戶可以操作的物件類別。物件的類別被區分為 **【類別 A】** 至 **【類別 F】** 共 6 種。

類別屬於 **【無】** 的物件，開放給任何用戶使用。



當 “用戶 1”的設定內容如下時，則此位使用者只被允許使用類別屬於 “無” 與 “A、B、C”的物件。詳細說明請參考《第十章 物件安全防護》。

Setting user-operable object categories and passwords						
編號	啟用	密碼	類別 A	類別 B	類別 C	類別 D
1	<input checked="" type="checkbox"/>	1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

【工程檔密碼 (MTP 檔)】

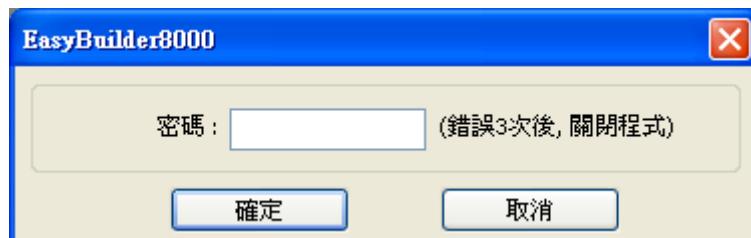


使用者可以設定保護工程檔案 (mtp 檔) 的密碼。若有設此密碼，當使用者想要編輯 MTP 檔時，必須輸入此密碼才能編輯。

勾選 **【啟用】** 再按 **【設定】** 會彈出下列視窗。

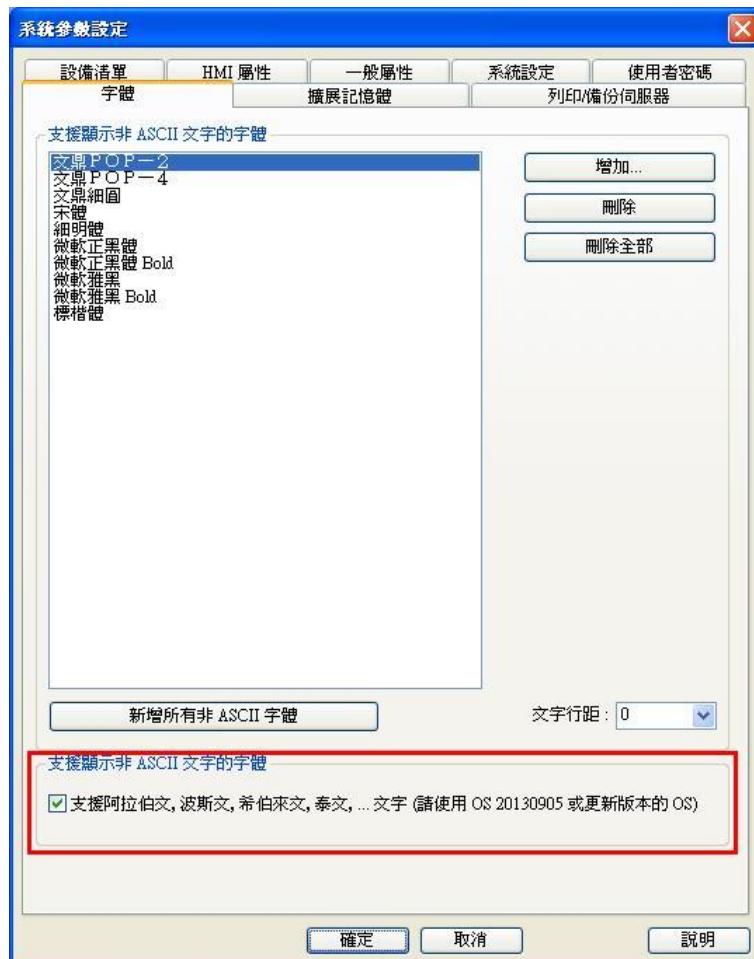


完成設定後，在編輯工程檔前，下面的視窗會彈出，要求使用者輸入密碼，輸入正確才能進入工程檔。



5.6 字體

[字體] 設定頁用來設定非 ASCII 文字所使用的字型。



[新增非 ASCII 字體]

在此項目表列出的字型為非 ASCII 的文字所使用的字型。當使用者使用了非 ASCII 的文字或雙字元文字(例如：簡體中文字、繁體中文字、日文、韓文等)，並使用了非表列中的字型時，EasyBuilder 會自動為這種文字選用表列中的字型。

使用者可以將 WINDOWS 的非 ASCII 文字的字型，新增到 **[新增非 ASCII 字體]** 列表中。

[文字行距]

決定行與行之間的間距。

[支援阿拉伯文、波斯文、希伯來文、泰文，... 文字]

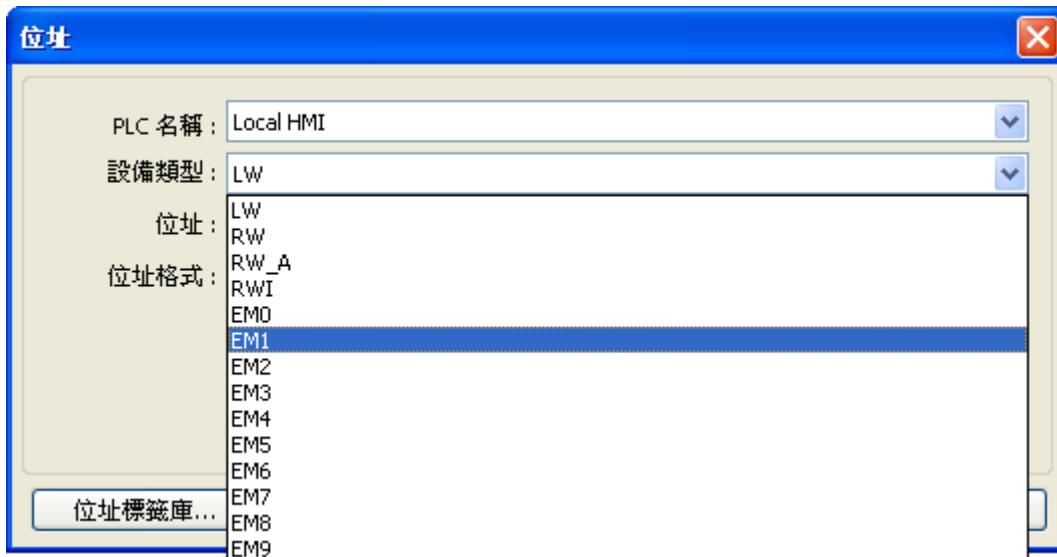
若勾選，將可正確顯示阿拉伯文、波斯文、希伯來文、泰文等文字。

5.7 擴展記憶體

【擴展記憶體】設定頁用來設定擴展記憶體的位置。



擴展記憶體包含 EM0 ~ EM9，使用方式與其他 HMI 上的設備類型相似(類似使用 LW 或 RW 位址類型)，使用者只需在建立物件時自 【設備類型】 中指定使用 EM0~EM9 即可，每個擴展記憶體最多可以存放 2G word 的資料。



擴展記憶體中的數據使用檔案的形式存放在 **[SD 卡]**、**[USB 碟]** 上，**[EM0] ~ [EM9]** 所使用的檔案名稱分別為 em0.emi ~ em9.emi，使用者可以使用 **RecipeEditor.exe** 開啟這些檔案並編輯擴展記憶體中的數據。

擴展記憶體中的數據不會因 HMI 斷電而消失，也就是下次開機後，擴展記憶體中的數據會恢復為關機前的狀態，與配方數據(RW、RW_A)類似。較特別的是使用者可以指定擴展記憶體的位置，可以選擇 SD 卡、USB 碟。

當做為擴展記憶體的裝置不存在時，若讀取擴展記憶體中的數據，內容將一律為 “0”；當做為擴展記憶體的裝置不存在時，若要將數據寫到擴展記憶體中，HMI 將會顯示 “PLC no response” 訊息。

HMI 支援“熱插拔”功能，在機器不需斷電的情況下，可以隨時插上或移除 SD 卡、USB 碟。使用者可以利用這項特性，更新或擷取擴展記憶體中的數據。

5.8 列印/備份伺服器

此設定頁用來設定遠端列印 / 備份伺服器的相關數據。



各項設定的說明如下：

設定	敘述
輸出設定	方向 設定圖文要 【水平】 或 【垂直】 列印。 列印大小 設定要照 【原始尺寸】 或 【配合印表機邊界】 列印。

	邊界 設定上下左右的邊界寬度。
通訊設定	IP 位址 透過網路指定印表機的 IP 位址。 連接埠、使用者名稱、密碼 指定登入印表機的資訊。 連接埠可設定為 1~65535。 使用者名稱或密碼最長可為 12 個字元。

更多資訊請參考《第 26 章 Easy Printer》。

第六章 視窗

視窗是 HMI 畫面程式的一個基本元素，也是很重要的一個元素。有了視窗後，畫面上的各種物件、圖形、文字等資訊才可以顯示在 HMI 上。EasyBuilder 提供視窗 3 ~ 1999，總共 1997 個視窗的建立和編輯功能。

6.1 視窗類型

依照功能與使用方式的不同，可將視窗分為下列四種類型：

(1) 基本視窗 (2) 快選視窗 (3) 公共視窗 (4) 系統訊息視窗

6.1.1 基本視窗

基本視窗是最常被使用的視窗類型，除了當作主畫面的用途之外，也被用在：

- 底層畫面，可提供其他視窗作為背景畫面。
- 鍵盤視窗。
- 功能鍵物件所選用的彈出視窗。
- 間接視窗與直接視窗物件所選用的彈出視窗。
- 螢幕保護視窗畫面。



■ 基本視窗必須是與螢幕的大小一樣，也就是基本視窗的解析度需要與所使用的 HMI 的解析度一致。

6.1.2 快選視窗

3 號視窗為預設的快選視窗，此種視窗可以與基本視窗同時存在，一般被用在置放常用的工作按鈕，位置為左下角或右下角。要使用快選視窗除了需先建立 3 號視窗外，需再設定快選視窗按鈕的各項屬性，快選視窗按鈕的各項設定在 **【系統參數設定】** » **【一般屬性】** 頁籤中。除了可以使用快選視窗按鈕來切換快選視窗的顯示與隱藏之外，系統暫存器也提供以下位址可用來控制快選視窗：

- [LB-9013] 快選視窗控制 [隱藏 (ON)/顯示 (OFF)]
- [LB-9014] 快選按鍵控制 [隱藏 (ON)/顯示 (OFF)]
- [LB-9015] 快選視窗/按鍵控制 [隱藏 (ON)/顯示 (OFF)]

6.1.3 公共視窗

4 號視窗為預設的公共視窗，此視窗中的物件也會出現在其他基本視窗中，但不包含彈出視窗，因此通常會將各視窗共用的物件放置在公共視窗中。

HMI 程式運行時，可以使用功能鍵物件的 [切換公共視窗] 模式，線上更改公共視窗的來源。

在 **[選項功能表] » [功能選項]**
可設定當編輯程式時，公共視窗上的物件是否會被顯示於基本視窗。有了此預覽功能，可避免當編輯程式時，將基本視窗的物件重疊到公共視窗的物件。

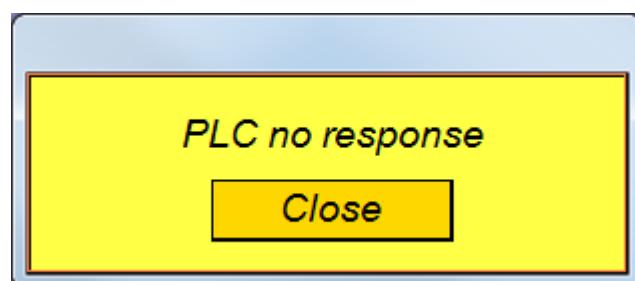


6.1.4 系統訊息視窗

5、6、7、8 號視窗為預設的系統提示訊息視窗：

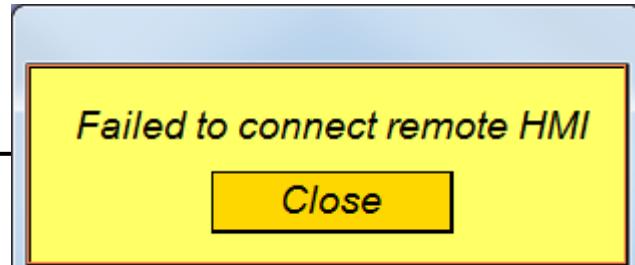
[5 號視窗為 PLC Response 視窗]

當 HMI 與 PLC 或控制器通訊中斷時，系統將自動彈出此視窗在 HMI 當前開啟的基本視窗上。



- 可使用系統暫存器所提供的相關位址來禁止彈出 “PLC no response” 視窗。詳情可查看第 22 章節 位址暫存器。

[6 號視窗為 HMI Connection 視窗]



當本地 HMI 無法連接到遠端的 HMI 時，系統將自動彈出此視窗。

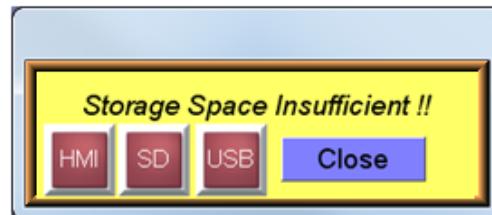
[7 號視窗為 Password Restriction 視窗]

當使用者的操作權限不足以操作所選物件時，可依物件的設定內容，決定是否彈出此視窗做為警示用途。



[8 號視窗為 Storage Space Insufficient 視窗]

當 HMI 記憶體、USB 碟或 SD 卡上的可用空間不足以儲存新的資料時，系統將自動彈出此視窗。
(當系統偵測到記憶體剩 4 MB 以下)



下列的系統暫存器可檢視 HMI、USB 碟或 SD 卡上目前可用的儲存空間：

- [LW-9072] HMI 目前的可用空間 (K bytes)
- [LW-9074] SD 卡目前的可用空間 (K bytes)
- [LW-9076] USB 碟 1 目前的可用空間 (K bytes)
- [LW-9078] USB 碟 2 目前的可用空間 (K bytes)

並可透過下列的系統暫存器檢視儲存裝置空間是否足夠，若系統偵測到記憶體剩 4 MB 以下，會將相關位址設為 ON：

- [LB-9035] HMI 可用空間不足警示 (當狀態為 ON)
- [LB-9036] SD 卡可用空間不足警示 (當狀態為 ON)
- [LB-9037] USB 碟 1 可用空間不足警示 (當狀態為 ON)
- [LB-9038] USB 碟 2 可用空間不足警示 (當狀態為 ON)

視窗 5 ~ 8 的內容提示，可以根據實際需要來修改，方便操作人員容易讀懂和識別故障資訊。

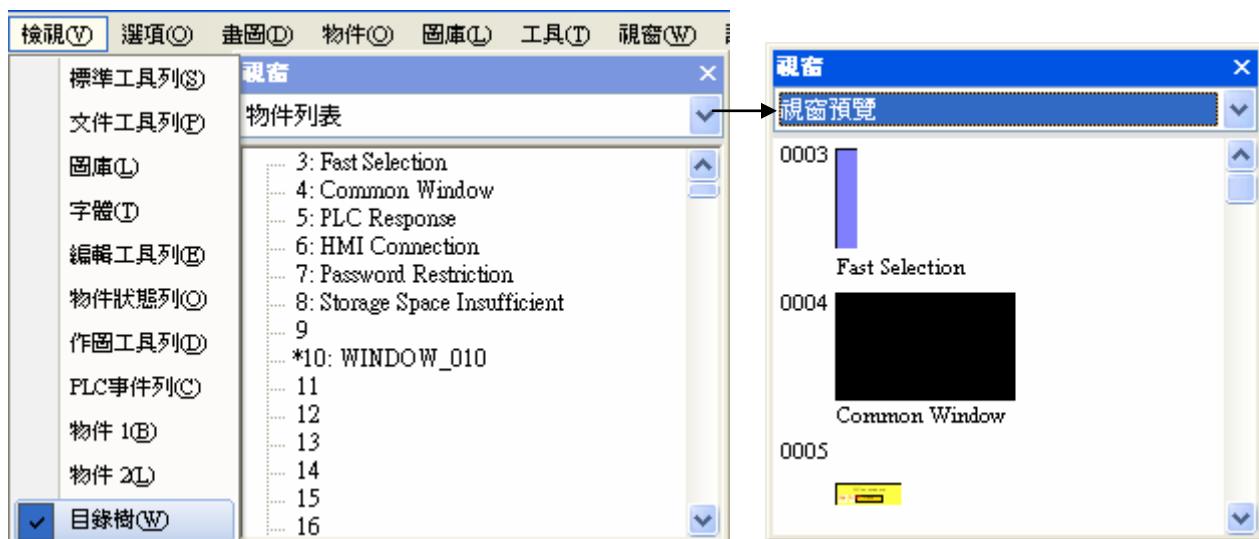


- 最多可同時開啟 16 個彈出視窗，包含系統訊息視窗、直接視窗和間接視窗。
- 系統不允許在一個基本視窗上使用 2 個直接(或間接) 視窗彈出同一個視窗。

- 視窗 3 ~ 9 為系統內部使用，視窗 10 ~ 1999 為使用者可任意編輯操作之視窗。

6.2 視窗的建立、設定與刪除

可透過 **【檢視功能表】»【目錄樹】** 查看已建立的視窗。



【物件列表】 顯示視窗編號，已定義的視窗名稱將被顯示。目前被開啟編輯的視窗編號前會有個 (*) 符號，按下視窗編號旁的 (+) 號可看到視窗含有哪些物件，包含物件 ID、位址與描述。

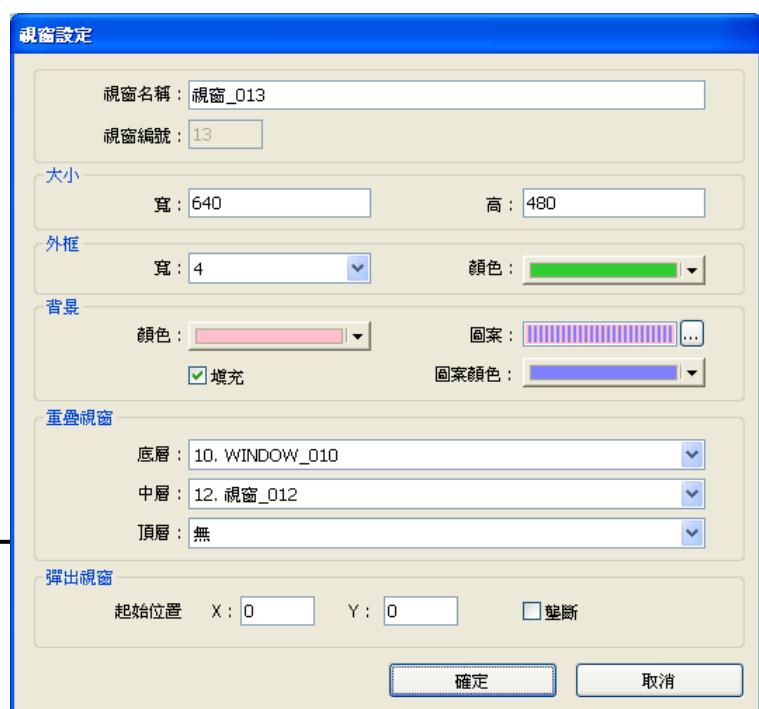
【視窗預覽】 用整體視窗外型的小圖來預覽視窗。

6.2.1 視窗的建立與設定

在目錄樹 **【物件列表】** 模式下，選擇欲建立的視窗編號後按下滑鼠右鍵選擇 **【新增】**。

【視窗名稱】 輸入的名稱將顯示在視窗的控制條，也會顯示在目錄樹中。

【視窗編號】 由 3 ~ 1999。



[大小] 設定視窗的大小，一般基本視窗的解析度與所選用的 HMI 的解析度一樣。

[重疊視窗] [彈出視窗]

請見下方說明

[重疊視窗]

重疊視窗功能可視為另一個額外的公共視窗，在設計程式時，同一物件可能被放置於許多的視窗中，但不是所有視窗時，便可使用重疊視窗。

每一個基本視窗最多可選擇三個視窗作為背景，從 **[底層]** 開始到 **[頂層]** 結束，這些背景視窗內的物件將在基本視窗中依序出現的。

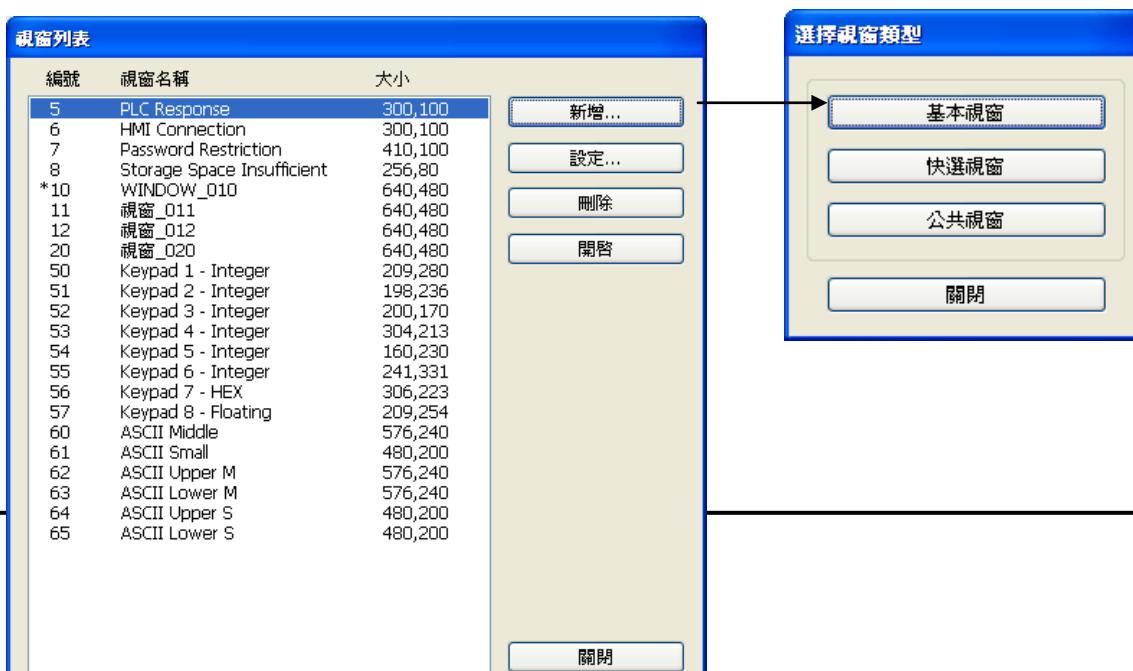
[彈出視窗] 基本視窗也可當作彈出視窗，**[X]** 與 **[Y]** 用來設定基本視窗在畫面彈跳出的座標位置。座標原點為畫面的左上角。

[壟斷] 若勾選，此視窗的顯示將獨佔所有的操作權。例如，當一個設定壟斷的視窗彈出，其他彈出視窗與背景視窗的操作將完全暫停，直到壟斷的視窗被關閉才可操作其他視窗。當基本視窗作為鍵盤視窗時，自動具有此屬性。



- 在重疊視窗中的物件無法從顯示它們的基本視窗上編輯，若要編輯重疊視窗的物件，均需開啟該視窗編輯。
- 若基本視窗所使用的重疊視窗編號與欲彈出的視窗編號相同，將無法彈出該視窗。
- 當基本視窗與彈出視窗皆使用相同的重疊視窗時，彈出的視窗將無法顯示重疊視窗上的物件。

或是在 **【視窗功能表】** » **【視窗列表】** 選擇 **【新增】**，並選擇欲建立的視窗類型。



呼叫 **【視窗設定】** 對話窗有以下方式：

- 於目錄樹選擇視窗編號，按下滑鼠右鍵選擇 **【設定】**。



- 在 **【視窗功能表】** » **【視窗列表】** 選擇欲設定的視窗後選擇 **【設定】**。
- 在該視窗中，未選擇任何物件時按下滑鼠右鍵選擇 **【屬性】**。



6.2.2 視窗的開啟、關閉或刪除

開啟現有的視窗：

- 使用滑鼠雙擊目錄樹物件列表上的視窗編號。
- 在目錄樹物件列表上選擇欲開啟的視窗後，按下滑鼠右鍵選擇 **【打開】**。
- 在 **【視窗功能表】** » **【視窗列表】** 選擇欲開啟的視窗後，選擇 **【開啟】**。

關閉或刪除現有的視窗：

- 在目錄樹物件列表上選擇欲關閉或刪除的視窗後，按下滑鼠右鍵選擇 **【關閉】** 或 **【刪除】**。
- 在 **【視窗功能表】** » **【視窗列表】** 選擇欲刪除的視窗後，選擇 **【刪除】**。

若要刪除某一個視窗，必須先將其關閉才可以刪除。

第七章 事件登錄



7.1 事件登錄管理

透過這些物件，可以得知事件從發生 → 等待處理 → 警報解除的時間。首先必需定義事件的內容。



事件登錄

目前類別：全部 [2]							
編號	類別	事件內容	位址類型	觸發條件	讀取位址	通知觸發位址	報警聲
1	0	Event 0	BIT	ON	Local HMI: LB-0	停用	停用
2	0	Event 0	BIT	ON	Local HMI: LB-0	停用	停用

報警時自動打開背光燈

歷史資料

保存到 HMI 保存到 USB 碟

檔案保留時間限制 保留時間： 7 天

列印格式

序號

事件發生時間 HH:MM:SS HH:MM DD:HH:MM

事件發生日期 MM/DD/YY DD/MM/YY DD.MM.YY YY/MM/DD

工具

新增... 插入... 刪除 設定... 動出... 備份...
複製 貼上 貼上 (新增模式) 關閉

各項設定的說明如下：

[目前類別]

提供事件分類功能，將事件分成 0 ~ 255 個類別，可選擇一個類別來輸入或列出事件資料。[] 中顯示此類別的事件資料數量。

[保存檔案]

指定事件記錄檔案的儲存位置，但若使用連線或離線模擬功能時，檔案一律存放在安裝目錄下的 HMI_memory / SD_card / USB 資料夾內。

[檔案保留時間限制]

此項設定值用來決定事件記錄檔案被保留的天數。若 **[保留時間]** 設定為兩天，也就是系統將只保留昨天與前天的事件記錄，這個時間範圍之前的檔案將自動被刪除，用來避免儲存空間被耗盡。

[列印格式]

需先在 **[系統參數設定] » [HMI 屬性]** 頁籤選擇印表機型號，才允許設定當事件觸發時，欲列印的格式。

[貼上]

新增之事件項目取代目前選擇的項目。在執行時會提供警示，避免產生錯誤的操作。

[貼上(新增模式)]

新增事件項目。

7.1.1 Excel 編輯

點擊事件登錄視窗右上角的 Excel 小圖示，可直接開啟 Excel 表格做為編輯時的範例參考。此範例為安裝目錄下的 EventLogExample.xls Excel 檔案，範例檔中有設計好的驗證機制和下拉式選單。

	A	B	C	D	E	F	G	H	I	J	K
1	Category	Priority level	Address type	PLC name	Device type	System tag	User-defined tag	Address	Index	Data Format	Enable
2	0	Middle	Word	Local HMI	LW	False	False	100	null	32-bit Signed	True
3	1	Low	Bit	Local HMI	LB-9009	True	False	9009	IDX 5	16-bit BCD	False
4										16-bit BCD	
5										22-bit BCD	
6										16-bit Unsigned	
7										16-bit Signed	
										32-bit Unsigned	
										32-bit Signed	
										32-bit Float	

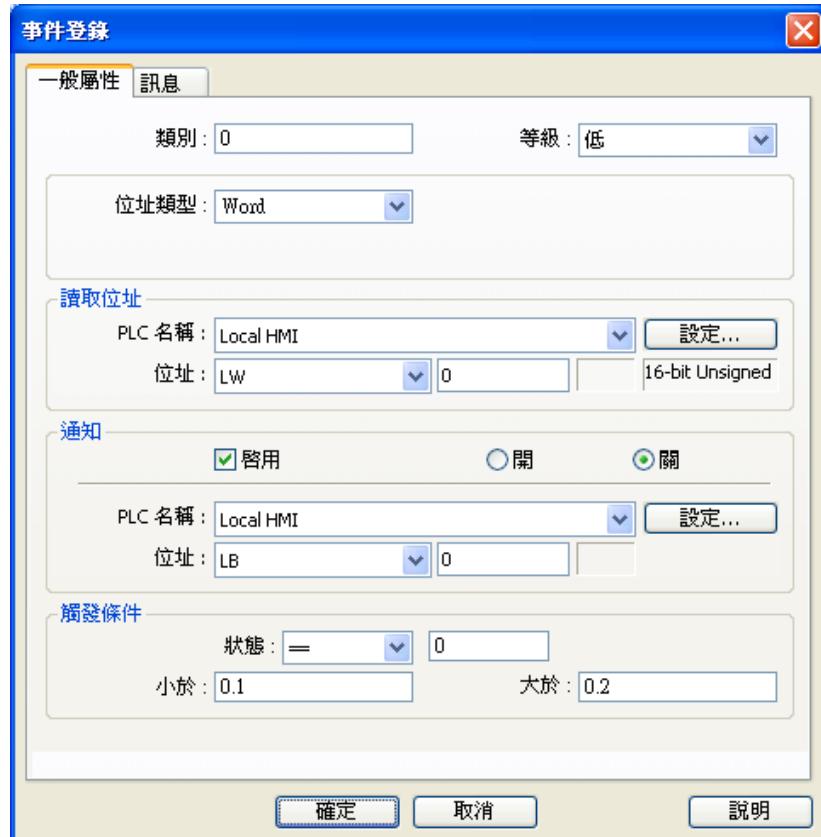


1. **[System tag]** 與 **[User-defined tag]** 請勿同時設為 true，否則系統會將其視為 **[System tag]**，並將 **[User-defined tag]** 視為 false。若在 **[Device type]** 所輸入的資料為 **[User-defined tag]**，請將 **[System tag]** 設為 false。
2. **[Color]** 格式為 R:G:B，各介於 0 ~ 255 之間的整數。
3. 若在 Excel 表中將 **[User-defined tag]** 設為 true，但若系統將 **[Device type]** 與系統中使用者自訂的 tag 做比對，卻找不到合適的 tag，則系統會自動將事件登錄中的 **[User-defined tag]** 設為 false。
4. 在匯入文字標籤庫或聲音庫前，請確認系統中已存在對應名稱。

7.2 建立一個新的事件記錄

7.2.1 事件登錄一般屬性

點選【事件登錄】視窗的【新增】，將出現【一般屬性】設定頁。



各項設定的說明如下：

[類別]

選擇事件類別，從 0 ~ 255。

[等級]

當已發生事件的數目等於系統允許的最大值數目（預設值為 1000）時，重要程度較低的事件將從事件記錄中被刪除，並加入新發生的事件。

[讀取位址]

系統將讀取此位址所獲得的數據，來檢查事件是否滿足觸發條件。

[通知]

若勾選，系統會在事件發生時，將指定暫存器狀態設為【開】或【關】。

[觸發條件]

當選擇 Bit 時，事件登錄將偵測一個位元位址的狀態。

當選擇 Word 時，事件登錄將偵測一個字元位址的值是否等於、大於或小於一個特定數值。

Example 1

觸發條件

狀態 :	=	30	
小於 :	1	大於 :	2

上面的設定內容表示

當 **[讀取位址]** 中的數據大於等於 $29 (= 30 - 1)$

或小於等於 $31 (= 30 + 1)$ 時，事件將被觸發。也就是事件被觸發的條件為

$$29 \leq [\text{讀取位址}] \text{ 中的數據} \leq 31$$

事件被觸發後，當 **[讀取位址]** 中的數據大於 $32 (= 30 + 2)$ 或小於 $28 (= 30 - 2)$ 時，系統將恢復為正常狀態。也就是系統恢復為正常狀態的條件為

$$[\text{讀取位址}] \text{ 中的數據} < 28 \text{ 或 } [\text{讀取位址}] \text{ 中的數據} > 32$$

Example 2

觸發條件

狀態 :	<>	30	
小於 :	1	大於 :	2

上面的設定內容表示

當 **[讀取位址]** 中的數據小於 $29 (= 30 - 1)$

或大於 $31 (= 30 + 1)$ 時，事件將被觸發。也就是事件被觸發的條件為

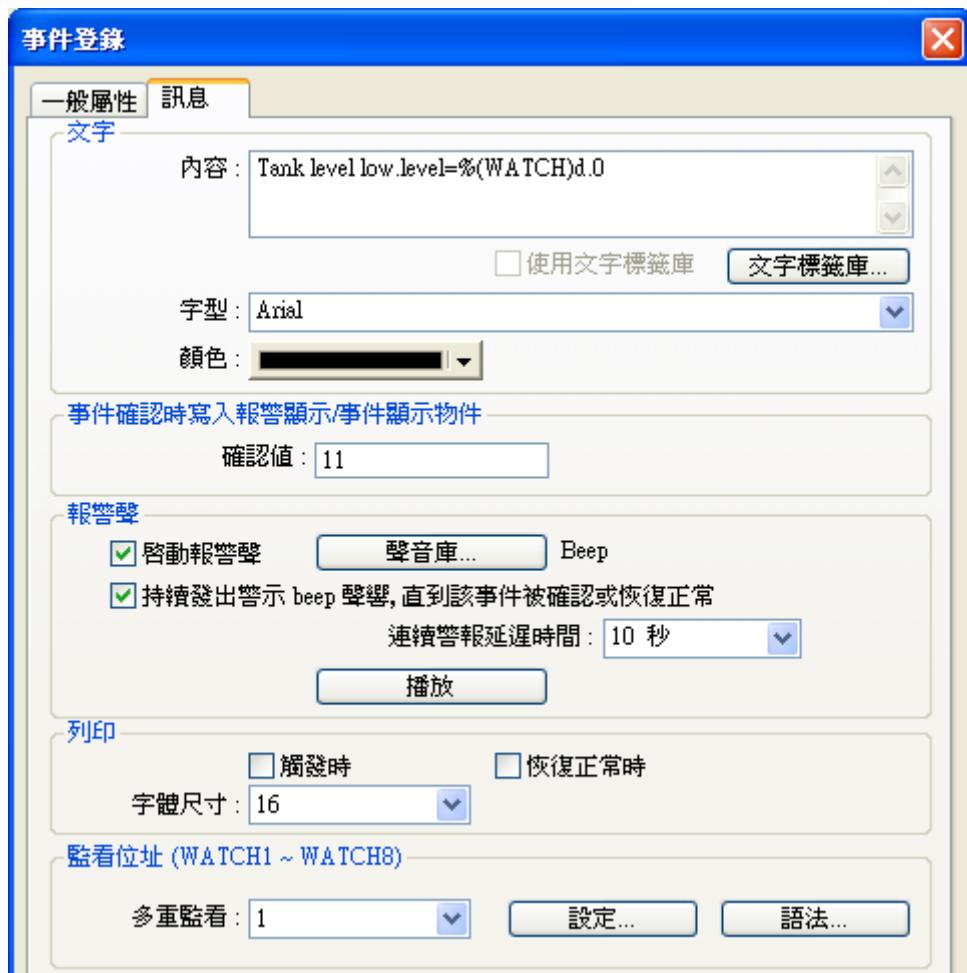
$$[\text{讀取位址}] \text{ 中的數據} < 29 \text{ 或 } [\text{讀取位址}] \text{ 中的數據} > 31$$

事件被觸發後，當 **[讀取位址]** 中的數據大於等於 $28 (= 30 - 2)$ 或小於等於 $32 (= 30 + 2)$ 時，系統將恢復為正常狀態。也就是系統恢復為正常狀態的條件為

$$28 \leq [\text{讀取位址}] \text{ 中的數據} \leq 32$$

7.2.2 事件登錄訊息設定

【事件登錄】 另一頁籤為 **【訊息】** 設定頁。



各項設定的說明如下：

[內容]

事件記錄在報警條、報警顯示與事件顯示物件中顯示的訊息內容。並可在內容中使用監看位址 WATCH1 至 WATCH8 來引用資料或下列兩個範例中的格式。

Example 1

可以在顯示內容中包含事件被觸發當時 LW 位址中的數據。

使用格式為：`%#d` (% -> 起始, # -> 位址, d -> 結束)

假設觸發時 LW-20 中的數值為 13：

設定為 “High Temperature = %20d”，則會顯示為 “High Temperature = 13”。

Example 2

可以在顯示內容中包含事件被觸發當時，特定 PLC 位址類型中的數據，此特定位址類型與事件登錄的【**讀取位址**】需為相同位址類型，假設選擇 MODBUS RTU 4x 位址類型。

使用格式為：**\$#d** (\$ -> 起始, # -> 位址, d -> 結束)

假設觸發時 MODBUS RTU 4x-15 中的數值為 42：

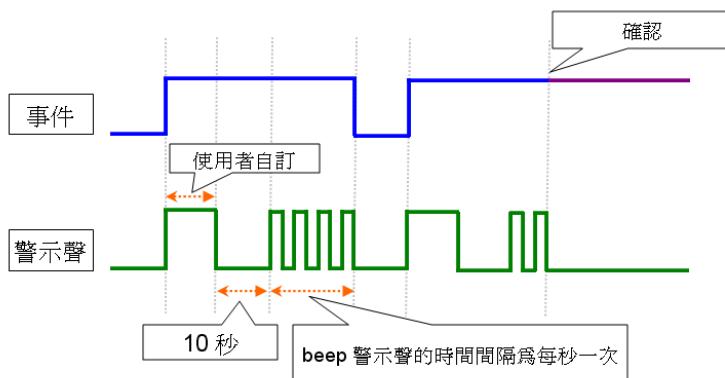
設定為 “High Temperature = \$15d” ，則顯示為 “High Temperature = 42”。

[字型] / [顏色]

每一個事件可以單獨設定字型與顏色，用於報警條、報警顯示與事件顯示中的字型顏色。

[事件確認時寫入]

當事件顯示與報警顯示物件中的事件項目被確認時，會將此數值寫入該物件指定的【**寫入位址**】。

[報警聲]

若勾選，當事件發生時會播放指定的聲音，並且可以選擇持續發出警報聲響，直到該事件被確認或恢復正常時，才會停止發出聲音。

若勾選使用持續警報聲響時，可以選擇在警報被觸發後，延遲所指定的時間後才開始連續發出聲音。

[監看位址]

使用者可以點選**【語法】**來編輯並顯示當事件觸發時，監看位址所設定暫存器內的數值。最多可同時監看八個位址。

第八章 資料取樣



定義「資料取樣」的取樣方式：

1. 取樣時間。
2. 取樣位址。
3. 字元長度。

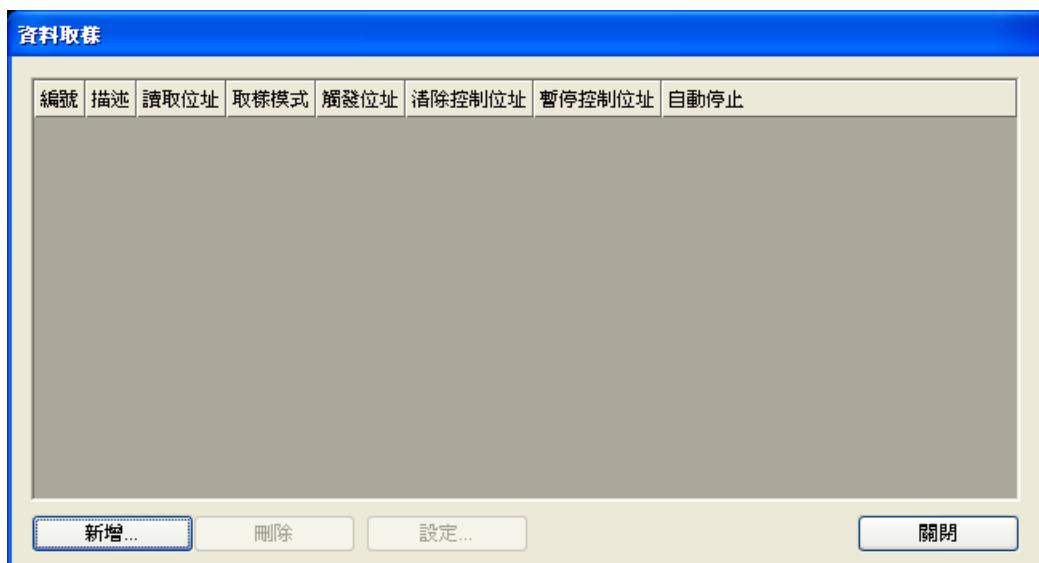
將已獲得的取樣資料儲存到指定的位置。可存至 HMI 記憶體、SD 卡或 USB 碟。

資料取樣可搭配使用
[趨勢圖] 或 [歷史數據顯示] 物件檢視資料取樣記錄的內容。

8.1 資料取樣記錄管理

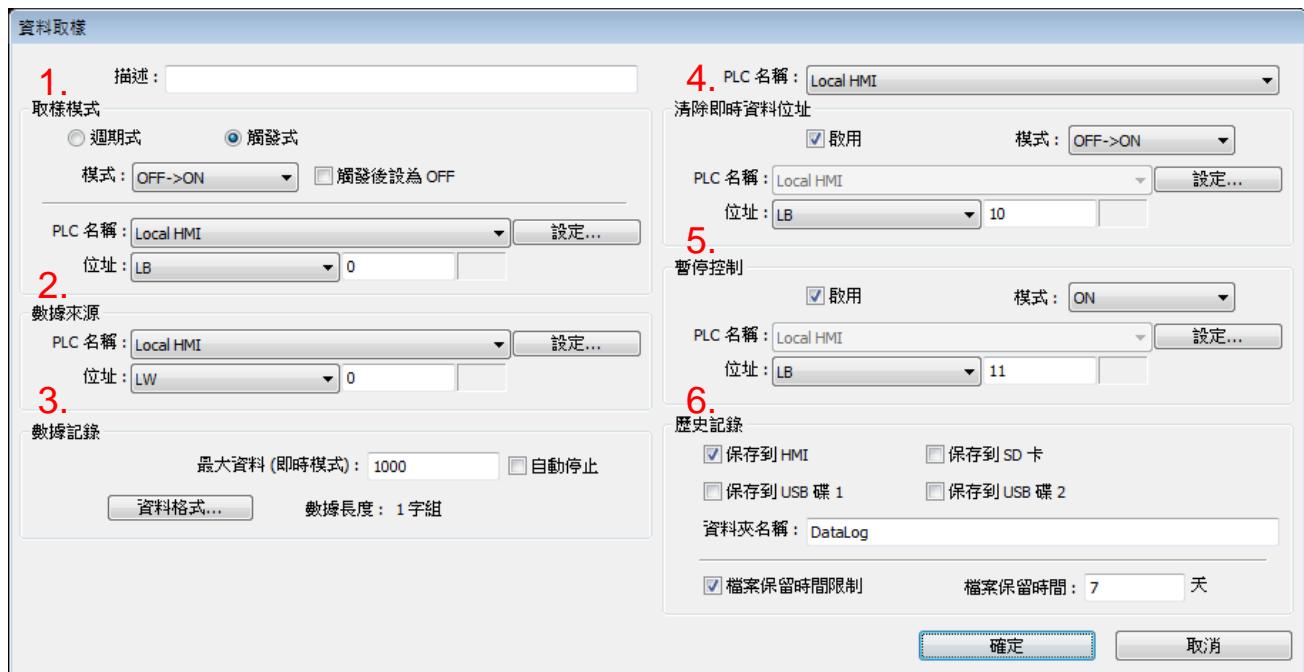
新增一個資料取樣，請依照下列步驟：

1. 點選功能表 **[物件]**，再點擊 **[資料取樣]**。
2. 點擊 **[新增]** 開始定義相關設定，如下圖：



8.2 新增一個資料取樣

以下介紹如何新增一筆資料取樣物件：



1. [取樣模式]

■ [週期式]

用固定的時間頻率進行資料取樣，[採樣週期] 可設定範圍從 0.1 秒至 120 分鐘。

■ [觸發式]

利用一個特定位元位址的狀態，來觸發取樣動作。

[模式] 可為：

[OFF -> ON] 若指定位址的狀態從 OFF 變為 ON，便會觸發資料取樣。

[ON -> OFF] 若指定位址的狀態從 ON 變為 OFF，便會觸發資料取樣。

[OFF <-> ON] 只要指定位址的狀態有所轉變，便會觸發資料取樣。

[觸發後設為 OFF]

若勾選，在觸發資料取樣後，系統會將觸發位元復歸為 ON/OFF。

2. [數據來源]

選擇一個設備位址作為取樣數據的來源。

3. [數據記錄]

一個資料取樣項目一天最多可以記錄的取樣資料筆數為 86400，即一天 24 小時，每秒採取樣一次。若 [採樣週期] 設為 “0.1 秒”，一天最多仍為 86400 筆。

■ [資料格式]

可從 [數據來源] 設定連續讀取多個不同資料，例如



■ [自動停止] 選項：

搭配不同物件，[自動停止] 產生的效用就不同，如下表：(假設最大資料設為 n)

搭配物件	未勾選 [自動停止]	勾選 [自動停止]
趨勢圖-即時模式	將刪除較舊的取樣資料，並顯示剛獲得最新的 n 筆資料在趨勢圖上。 請參考下列圖解。	至第 n 筆資料後停止動作。
趨勢圖-歷史模式	資料持續被取樣，並顯示所有歷史資料在趨勢圖上。	至第 n 筆資料後停止動作。
歷史資料顯示	資料持續被取樣，並顯示所有歷史資料在歷史資料顯示上。	至第 n 筆資料後停止動作。
資料取樣	持續取樣新記錄。	至第 n 筆資料後停止取樣。

圖解:

EX)	1.001	1.002
	2.002	2.003
	3.003	3.004
	4.004	4.005
	5.005	5.006
	6.006	6.007
	7.007	7.008
	8.008	8.009
	9.009	9.010
	10.010	10.011
	11.011	

如上圖，若設定資料長度數為 10 個，當第 11 個資料產生時，最舊的資料記錄將會被刪除，並增加最新的記錄。



- 一筆取樣資料可能包含超過一項以上的數據，資料取樣動作可以同時擷取不同型態的數據。使用者可以自行定義一筆取樣資料的內容。舉例來說，使用者共定義了三個數據，長度總共為 **4 words**。也就是說每次的取樣動作，系統會從指定的數據來源位址每次擷取長度為 **4 words** 的數據，作為一筆取樣資料的內容。
- 若執行模擬並將資料取樣記錄儲存，此時若要改變資料取樣的格式，須先刪除安裝目錄下的舊資料取樣記錄，以避免系統誤讀舊紀錄。

4. 【清除控制】

當指定位元位址的狀態由 [OFF -> ON] 或 [ON -> OFF] 時，將清除在趨勢圖 **【即時模式】** 下已取樣獲得的資料，取樣資料的數目也會被歸零，但不影響已存入檔案中的歷史取樣資料。

5. 【暫停控制】

當指定位元位址的狀態被設定為 ON 或 OFF 時，將暫停取樣動作，直到指定位址的狀態被恢復。

6. 【歷史紀錄】

■ [保存到 HMI]

將取樣資料儲存在 HMI 裡。資料必需要到達 4kb，才會被儲存；若少於 4kb，可以使用系統暫存器 **[LB-9034]** 來強迫儲存。

■ [保存到 SD 卡 / USB 1 / USB 2]

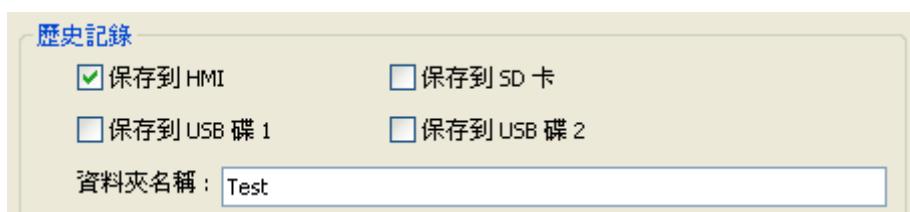
將資料取樣儲存到指定的外部裝置中。

■ [資料夾名稱]

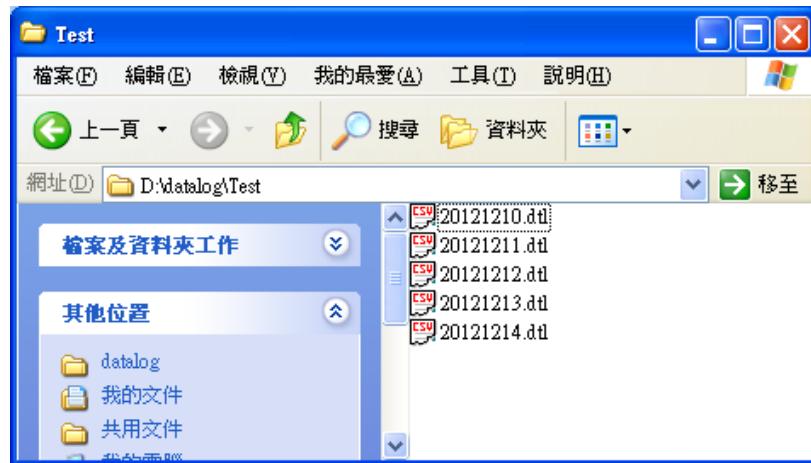
設定取樣資料夾的名稱，必須全部由 ASCII 字元所組成。

儲存的方式為：[保存位置] \ [取樣資料夾名稱] \ yyyyymmdd.dtl

檔案儲存是按日期記錄至指定檔名的資料夾內。



如下圖，檔案將會按照日期儲存於 Test 資料夾內。



■ [檔案保存時間限制]

此項設定值用來決定資料取樣記錄檔案被保留的時間。



- 假設 [檔案保存時間限制] 設定保留時間為兩天，也就是說系統將僅保留昨天與前天的資料取樣記錄檔案，超過這個時間範圍的檔案將自動被刪除，以避免儲存空間被耗盡。例如，今天是 7 月 1 號，USB 1 只會保存 6 月 30 號和 6 月 29 號的檔案，6 月 28 號的檔案將會被刪除。
- 在 PC 使用模擬功能時，資料取樣檔案一律儲存在安裝目錄下的「保存位置」下的 datalog 子目錄。

第九章 物件一般屬性

物件【一般屬性設定】的內容包含下面項目：

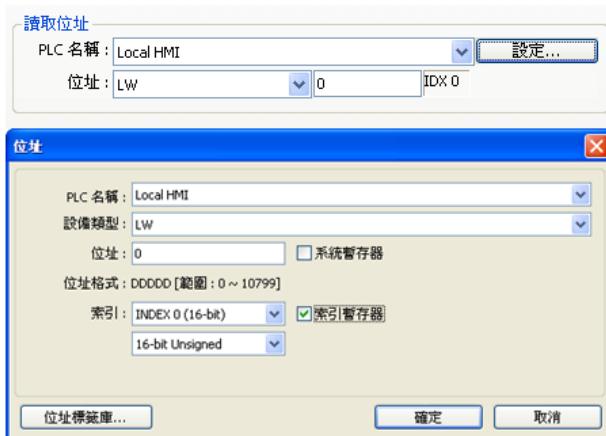
1. 選擇 PLC 裝置
2. 讀寫位址設定
3. 向量圖庫與圖片庫的使用
4. 標籤內容設定
5. 輪廓調整

9.1 選擇 PLC

某些物件的使用需選擇要操作的 PLC 對象，如下圖所示。【PLC 名稱】用來表示要控制的 PLC，下圖顯示目前存在的 PLC 名稱有“Local HMI”與“Mitsubishi FX0s/FX0n/FX1s/FX1n/FX2”，這些 PLC 名稱來自【系統參數】中【設備清單】的內容。



9.1.1 讀寫位址設定



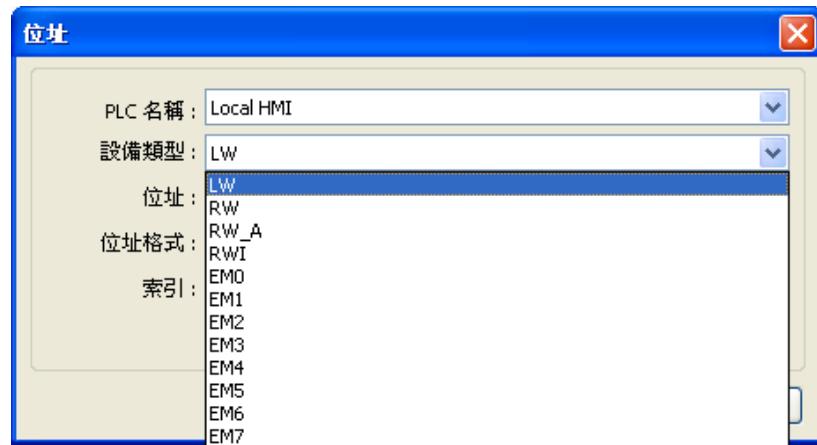
上圖可以看出一般位址的設定包含下列項目：

[PLC 名稱]

選擇 PLC 的型號。

[設備類型]

選擇位址類型，當 PLC 型號不同時，將出現不同的位址類型。

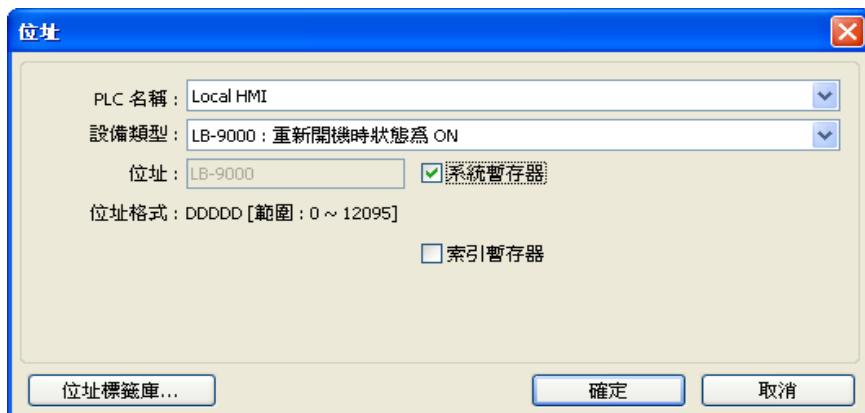
**[位址]**

設定讀寫的位址。

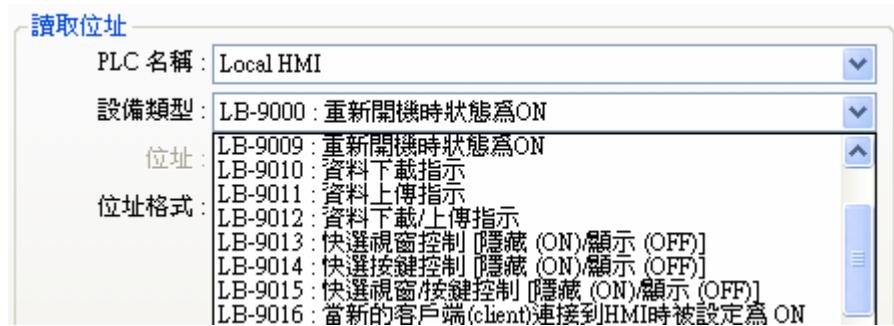
[系統暫存器]

位址標籤包含 **[系統暫存器]** 與 **[使用者定義位址標籤]**。此項目用來選擇是否使用 **[系統標籤]**。系統標籤為系統保留作為特殊用途的位址，分為 bit 位址系統暫存器與 word 位址系統暫存器 (LB 或 LW)。

在選擇使用 **[系統暫存器]** 後，除了 **[設備類型]** 將顯示系統暫存器的內容之外，**[位址]** 將顯示目前所選用的系統暫存器，如下圖所示。



下圖為系統暫存器的部分內容,其他內容可參考《第 16 章 位址標籤庫的建立與使用》或《第 22 章 系統保留暫存器位址和作用》 的說明。

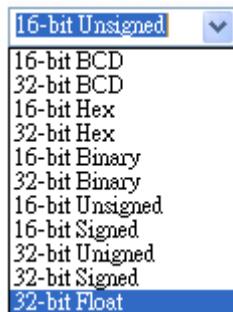


【索引暫存器】

選擇是否使用 【索引暫存器】，可參考 《第 11 章 索引暫存器》 的說明。

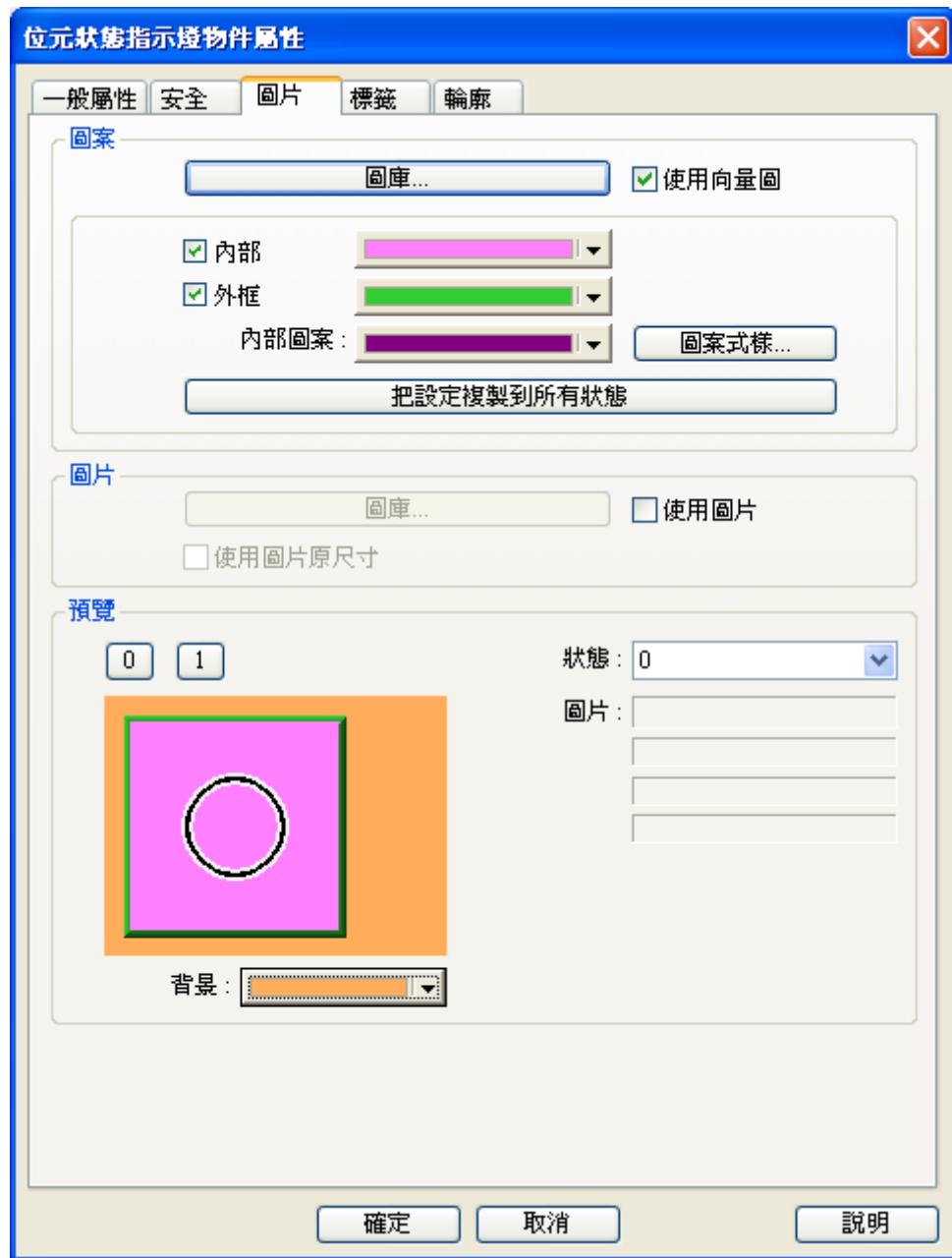
數據型態選擇

EasyBuilder 支援下列的數據型態，需正確選擇數據型態，尤其是在使用位址標籤時。



9.2 向量圖庫與圖片庫的使用

某些物件可以使用向量圖庫與圖片庫的圖形，增加物件的視覺效果。向量圖庫與圖片庫的使用在物件屬性頁中的【圖片】分頁中設定，見下圖。



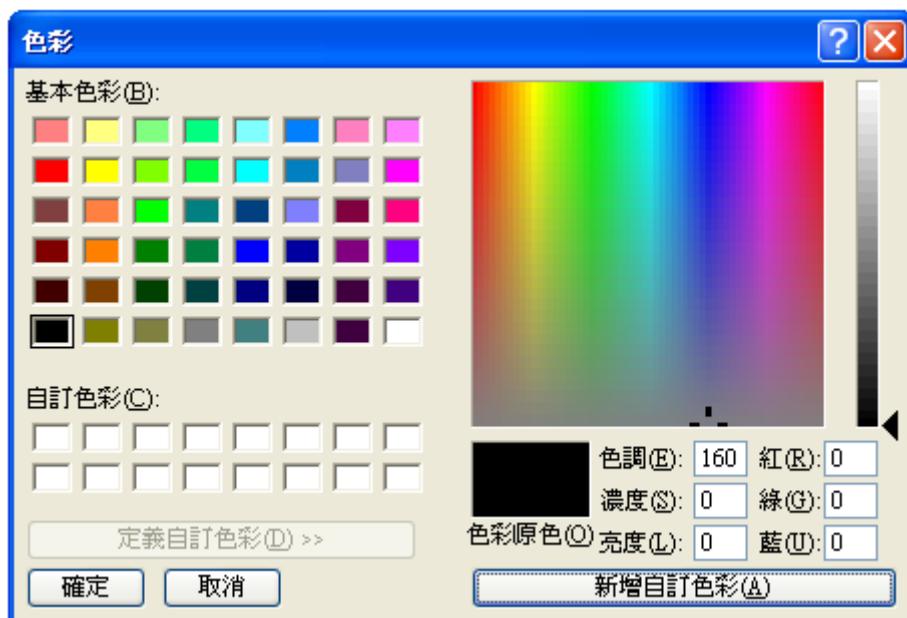
9.2.1 向量圖庫設定項

[圖庫]

勾選 [使用向量圖庫] 並從 [圖庫] 中選擇圖形樣式，此項目請參考後面的說明。

[內部]

選擇是否使用圖案的內底，按下顏色設定鈕後所出現的 [色彩] 對話窗，可來設定內底的顏色，如下圖所示。使用者也可以設定 [自定色彩]，按下 [新增自定色彩] 後 EasyBuilder 會記住使用者設定的 [自定色彩]。



[外框]

選擇是否使用圖案的外框，按下顏色設定鈕後所出現的 [色彩] 對話窗，可用來設定外框的顏色。

[內部圖案]

用來設定內底填充的圖案式樣與顏色。

[圖案式樣]

按下設定鈕後將出現下圖所示的對話窗，可用來選擇填充樣式。

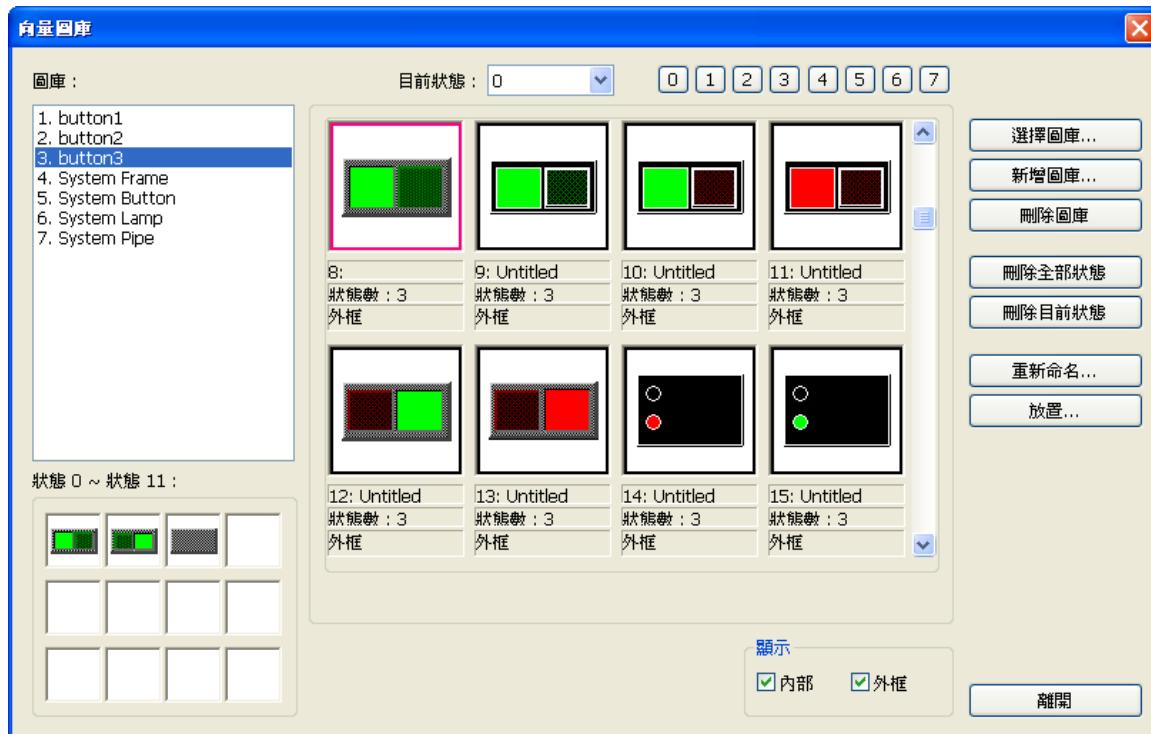


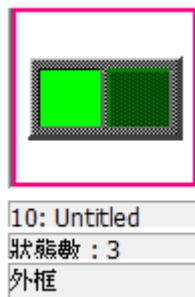
【把設定複製到所有狀態】

將目前狀態各項屬性設置到其他狀態。

如何使用向量圖庫

在按下 **【圖庫】** 按鈕後可以得到下面的 **【向量圖庫】** 對話窗，由對話窗中可看出目前選擇的樣式會使用紅色的外框加以標示。

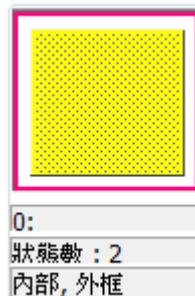




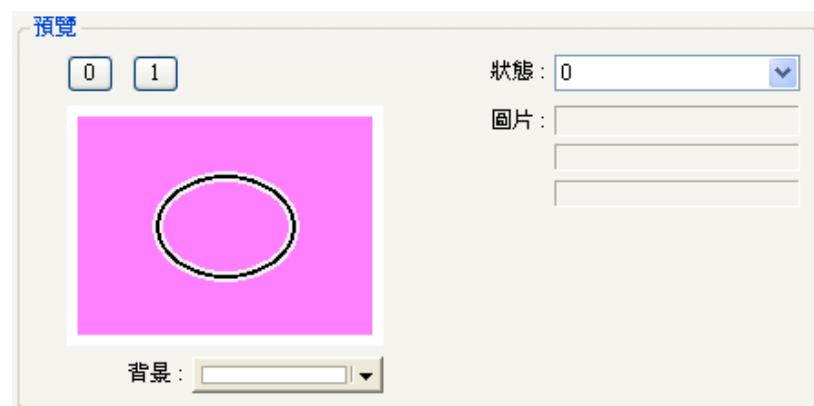
上圖顯示向量圖庫中某一樣式的資訊，這些資訊的意義如下：

- 10 : Untitled 表示此向量圖的名稱與向量圖庫編號
狀態數 : 3 此向量圖的狀態個數
外框 表示此向量圖只具備外框

下圖則顯示此向量圖具備外框與內底。



“向量圖管理對話窗”各項目的說明可參考《第十四章 向量圖、圖形庫的建立與使用》。在完成各項設定並按下確認鍵後，物件將使用目前所選擇的樣式，如下圖。



9.2.2 圖片庫設定項

【圖庫】

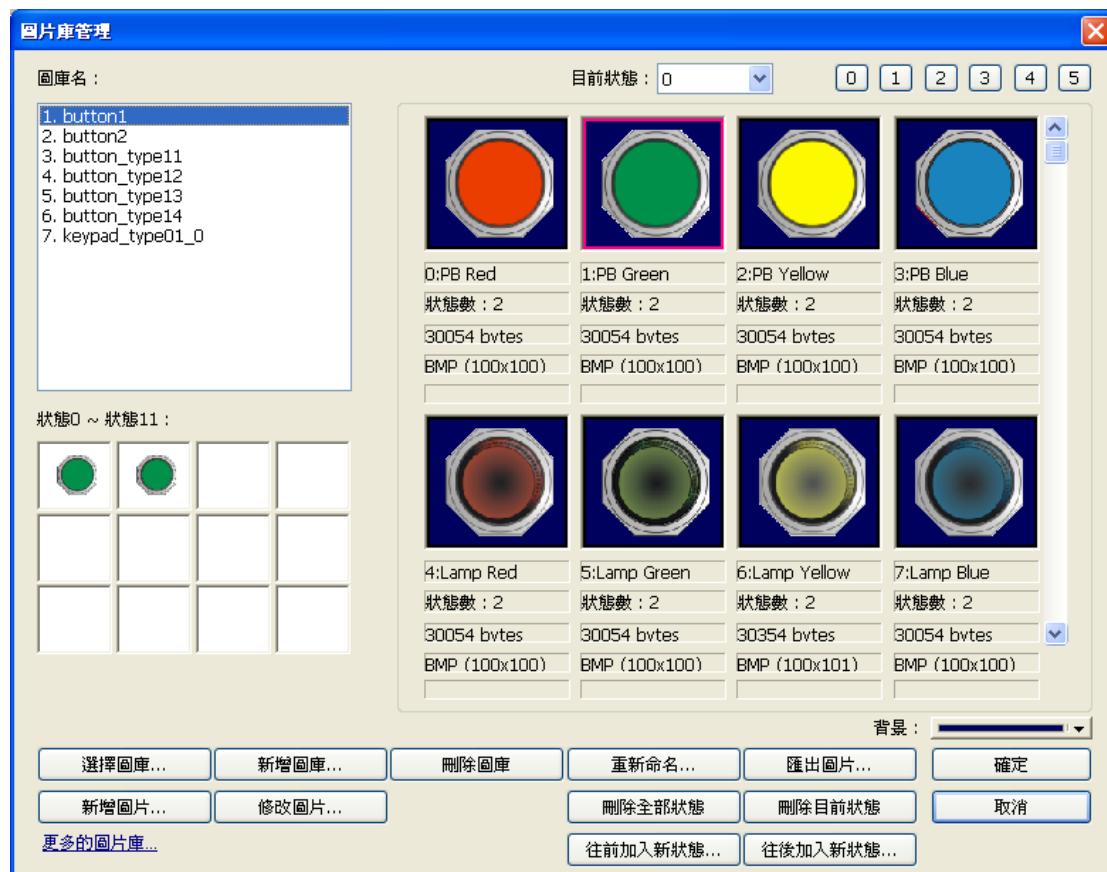
選擇圖形，此項目請參考後面的說明。

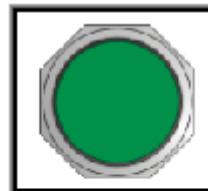
【使用圖片】

選擇樣式是否使用圖形庫的圖形。

如何使用圖形庫

在按下 【圖庫】 按鈕後可以得到下面的 【圖形庫管理】 對話窗，由對話窗中可看出目前選擇的圖形會使用紅色的外框加以標示。





1:PB Green
狀態數 : 2
30054 bytes
BMP (100x100)

上圖顯示圖形庫中某一圖形的資訊，這些資訊的意義如下：

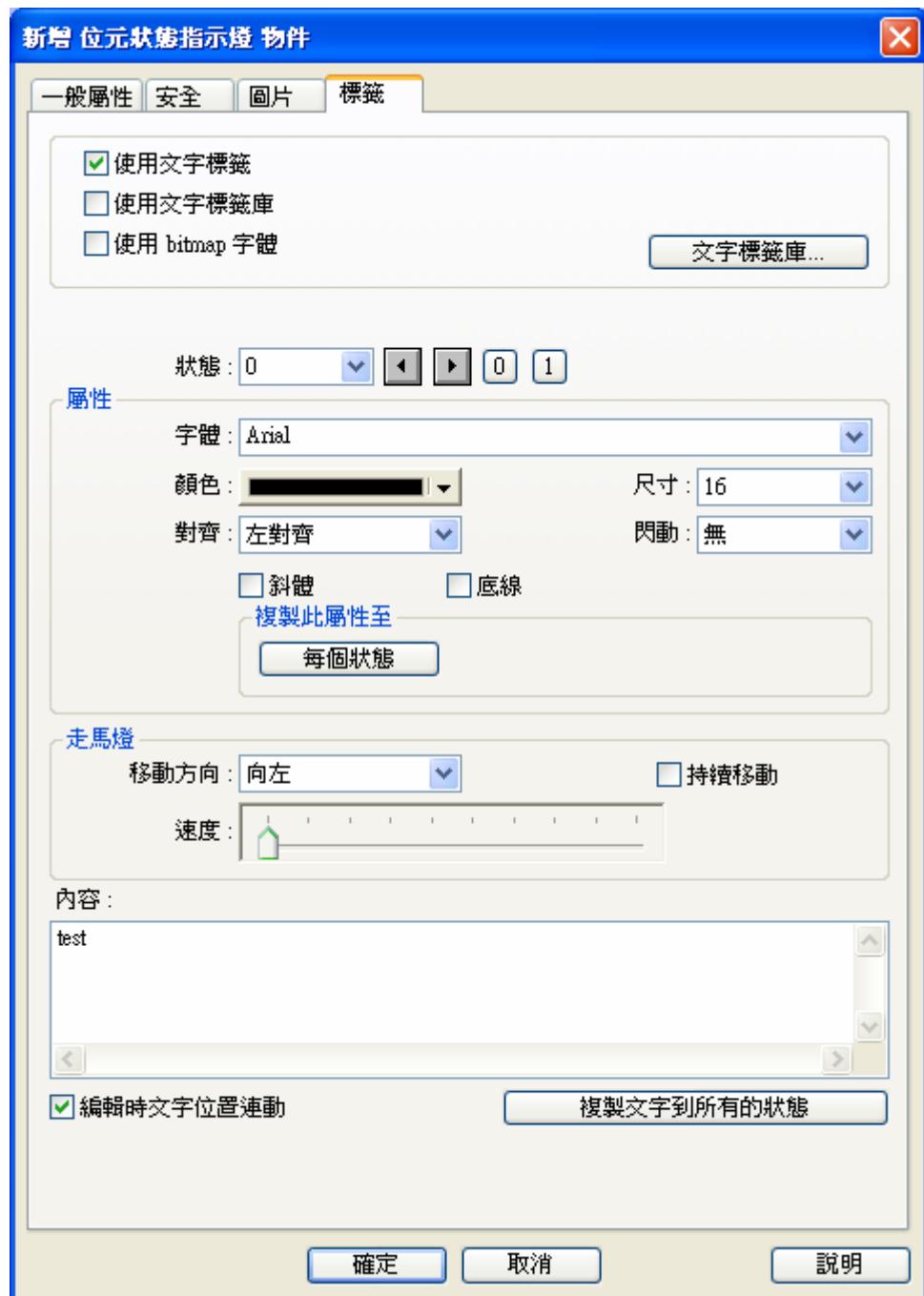
- 1 : PB Green 圖形編號與的名稱
狀態數 : 2 圖形的狀態個數
30054 bytes 圖形的大小
BMP (100 *100) 圖形的格式與原尺寸，BMP 表示圖形使用 bitmap 格式，圖形格式也可能為 JPG、PNG、DPD 或 GIF。100 * 100 表示圖形的原尺寸長為 100 pixels，高為 100 pixels。

“圖形庫對話窗”各項目的說明參考《第十四章 向量圖、圖形庫的建立與使用》。在完成各項設定並按下確認鍵後，物件將使用目前所選擇的圖形，如下圖所示。



9.3 文字內容設定

物件內文字的使用在物件屬性頁中的【標籤】分頁中設定，如下圖。



【使用文字標籤】

勾選此選項物件才允許使用文字標籤。 EasyBuilder 支援 Windows true-font。

[使用文字標籤庫]

勾選此選項表示文字內容將來自文字標籤庫，如下圖。

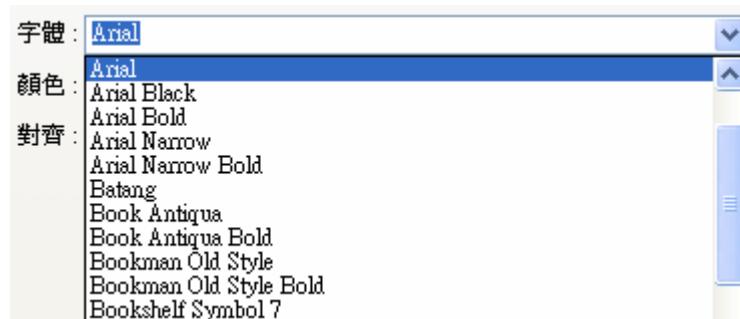


[文字標籤庫]

檢視文字庫的內容，可參考《第 15 章 文字標籤庫與多國語言使用》。

[字體]

選擇文字所使用的字型。EasyBuilder 支援 WINDOWS 的 true-font 字型，如下圖。



[顏色]

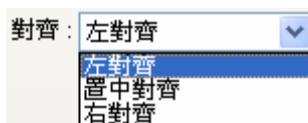
選擇文字所使用的顏色。

[尺寸]

選擇文字所使用的大小。

[對齊]

選擇多行文字的對齊方式，可選擇的方式如下：



下圖為選擇 [左對齊] 的對齊方式。

111
222222
33333333

下圖為選擇【置中對齊】的對齊方式。

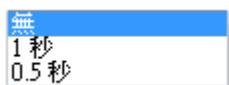
111
222222
33333333

下圖為選擇【右對齊】的對齊方式。

111
222222
33333333

【閃動】

選擇文字閃爍方式，可選擇不閃爍【無】，或閃爍時間間隔為【1秒】或【0.5秒】的閃爍方式。



【斜體】

使用斜體字體。

Italic Label

【底線】

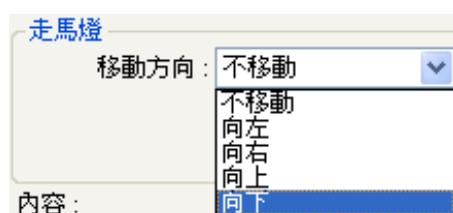
文字加上底線。

Underline Label

動作設定項

【走馬燈】

設定走馬燈的效果並選擇文字的移動方向，有下列的選擇：



[持續移動]

當文字選擇走馬燈的效果時，下圖的文字具有兩種顯示方式：



未勾選此選項，則文字需在全部消失後才出現後續的文字，如下圖。



有勾選此選項，則文字會連續出現，如下圖。

**[速度]**

選擇文字的移動速度。

[內容]

文字內容。如使用 **【文字標籤庫】**，此項內容將來自文字標籤庫。

[編輯時文字位置運動]

勾選此選項時，移動某個狀態的文字將連帶移動其他狀態的文字。

[複製文字到所有的狀態]

將目前的文字內容複製到其他所有的狀態。

9.4 輪廓調整

如下圖，物件的外型大小可以利用 **【輪廓】** 設定頁加以調整。



[位置] 設定項目

[圖釘]

鎖定設定，勾選此選項後將無法改變物件的位置與大小。

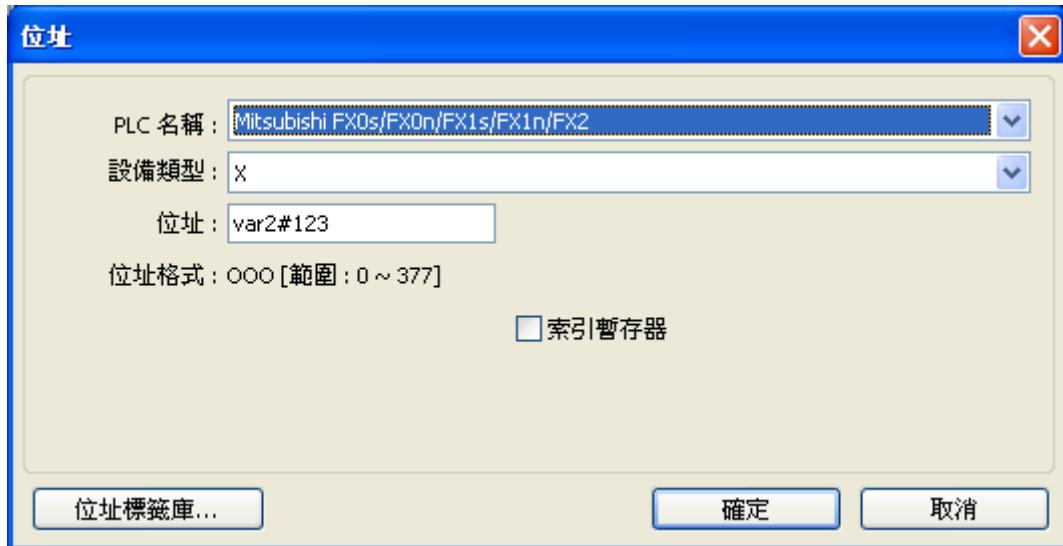
[X]、[Y] 物件左上角的座標。

[尺寸] 設定項目

設定物件的 **【寬度】** 和 **【高度】**。

9.5 站號變數的使用

在 PLC 的位址設定中允許使用站號變數，請參考下圖，其中 **[var 2]** 為 16 個站號變數中的一個。



站號變數的使用語法如下：`varN#address`

其中 N 的範圍為 0 ~ 15 的整數，address 為 PLC 的位址，

MT8000 目前共提供 16 個站號變數：var0 ~ var15，這些站號變數的實際數據讀取自 LW10000 ~ LW10015。下面為站號變數所對應的系統保留位址：

var0	LW10000	var8	LW10008
var1	LW10001	var9	LW10009
var2	LW10002	var10	LW10010
var3	LW10003	var11	LW10011
var4	LW10004	var12	LW10012
var5	LW10005	var13	LW10013
var6	LW10006	var14	LW10014
var7	LW10007	var15	LW10015

例如 **[var0]** 的數據因讀取自 LW10000，所以當 LW10000 中的數值等於 “32” 時，表示 **var0#234** 等同於使用 **32#234**，也就是此時的站號為 **32**；同樣的，**[var13]** 的數據因讀取自 LW10013，所以當 LW10013 中的數值等於 “5” 時，表示 **var13#234** 等同於使用 **5#234**。

9.6 廣播站號的使用

HMI 提供兩種方式讓用戶開啟廣播站號的使用。第一種方式是直接在 **【系統參數】** 中直接設定 PLC 的屬性，參考下圖：



第二種方式是使用系統保留位址開啟/關閉廣播站號功能，並更改廣播站號。
相關的系統保留位址的內容如下：

LB9065 開啟/關閉 COM 1 廣播站號功能

LB9066 開啟/關閉 COM 2 廣播站號功能

LB9067 開啟/關閉 COM 3 廣播站號功能

LW9565 COM 1 廣播站號

LW9566 COM 2 廣播站號

LW9567 COM 3 廣播站號

第十章 使用者密碼與物件安全防護

此章節介紹如何在 EasyBuilder8000 設定使用者的安全等級與密碼。後面的章節將詳細介紹相關的設定方式。

安全防護須完成以下兩項設定：

1. 用戶密碼與可操作物件類別設定
2. 物件操作安全防護



一個物件只能屬於一個安全等級，或將安全等級設定為“無”，使得任何人皆可操作該物件。

10.1 用戶密碼與可操作物件類別設定

在【編輯】»【系統參數設定】»【使用者密碼】頁籤可設定相關參數。

一般可規劃最多 12 個用戶，各別設定不同的用戶密碼，密碼需為非負整數，並規劃每個用戶可操作的物件類別分為“A ~ F”等共 6 個類別。

HMI 運作時，用戶在成功輸入密碼後，系統會依照設定內容決定用戶可以操作的物件類別。如下圖，“用戶 1”只被允許操作物件類別為“A、C”。



10.2 物件操作安全防護

[安全控制]

設定於避免誤觸物件即執行命令。

[最少按鍵時間]

只有當持續按壓物件時間大於此設定值才能成功啟動操作。

[操作前先確認] 按壓一個物件後，會出現一個確認對話窗，需點選 Yes 來確定操作。如果等待確認操作的時間超過 **[確認等待時間]**，對話窗會自動消失並取消動作。



[開啟/關閉]

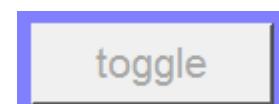
若勾選，此物件是否允許被操作，將決定於一個指定位元位址的狀態。

如圖，則必須在 LB-0 狀態為 ON 時，才允許操作此物件。

[關閉時隱藏] 當指定的位元位址於關閉狀態時物件會被隱藏。



[關閉時使用灰階效果顯示文字] 物件的標籤文字會在指定的位元位址於關閉狀態時以灰階樣式顯示。



[使用者限制]

設定物件類別，只允許可操作此類別的用戶操作。

[操作類別] “無”表示任意用戶皆可操作。

[操作完成後將使用限制取消] 一旦使用者的操作等級被允許操作該物件，系統便不再檢查該物件的安全等級，也就是說，即使是別的使用者，該物件也可被隨意操作。

[當使用者無權操作此類別時彈出提示視窗] 當用戶操作身份不符合此物件的操作等級時，將出現視窗 7 號作為警告。使用者可自行設定此視窗上的提示文字。

[當用戶無權操作此類別時隱藏此物件] 當用戶操作身份不符合此物件操作的等級時，物件會被隱藏。



10.3 物件安全防護範例

一般模式之物件安全防護的使用範例：

- 建立一個工程檔案，【系統參數設定】»【使用者密碼】»【一般模式】中啟用三個用戶，例如：
用戶 1 = 操作物件類別 A
用戶 2 = 操作物件類別 A, B
用戶 3 = 操作物件類別 A, B, C

- 在視窗 10 設計如圖：

【數值輸入】 物件

[LW-9219] 用戶編號 (1~12)

長度 = 1 word。

[LW-9220] 輸入用戶密碼，

長度 = 2 words。

【數值顯示】 物件 **[LW-9222]**

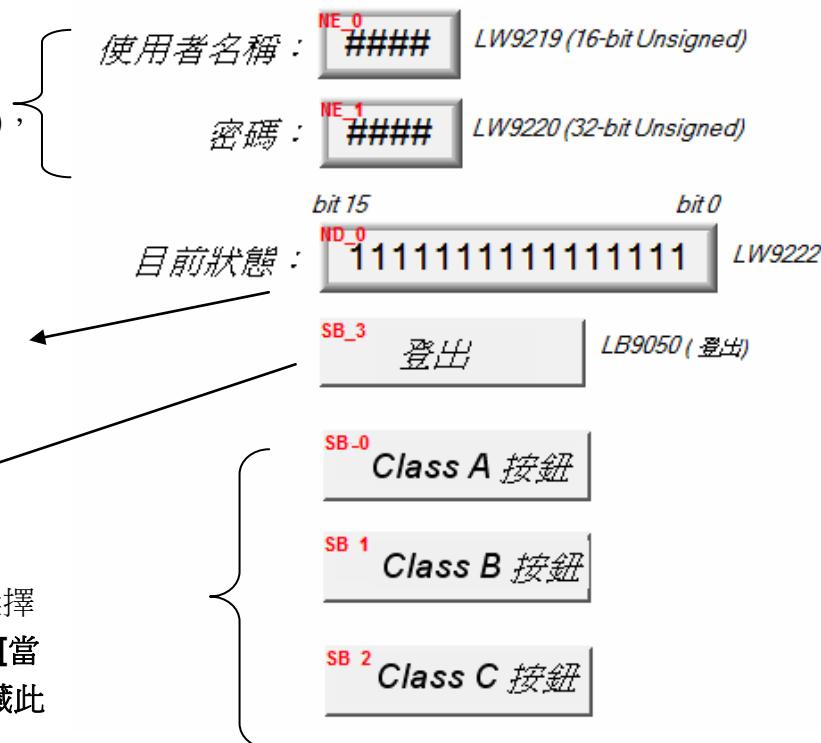
顯示目前用戶的狀態。16-bit

Binary

【位元狀態設定】 物件

[LB-9050] 用戶登出

三個 **【位元狀態設定】** 物件選擇不同的物件類別，但皆設定 **【當使用者無權操作此類別時隱藏此物件】**。



完成上述的各項設計並在存檔與編譯後即可執行離線模擬功能，下圖為離線模擬功能的起始畫面。

3. 因尚未輸入密碼，數值物件顯示“0000000000000000”，使用物件類別“無”。[Class A 按鈕] ~ [Class C 按鈕] 物件分別屬於類別“A ~ C”並設定【當使用者無權操作此類別時隱藏此物件】，所以皆被系統所隱藏。

使用者名稱 :	<input type="text" value="1"/>	LW9219 (16-bit Unsigned)
密碼 :	<input type="text" value="0"/>	LW9220 (32-bit Unsigned)
目前狀態 :	bit 15 bit 0 <div style="text-align: center;"> <input type="text" value="0000000000000000"/> LW9222 </div>	
	<input type="button" value="登出"/>	LB9050 (登出)

5. 輸入“用戶 3”的密碼(333)，因規劃允許使用類別 A, B, C 物件，[LW-9222] 的 bit 0 ~ bit 2 皆變為“1”，表示此時的用戶允許使用類別“A ~ C”的物件。

4. 輸入“用戶 1”的密碼(111)：

因規劃 “用戶 1” 允許使用類別 A 物件，所以此時 [Class A 按鈕] 物件將出現並允許操作。[LW-9222] 的 bit 0 變為 “1”，表示此時的用戶允許使用類別 “A”的物件。

使用者名稱 :	<input type="text" value="1"/>	<i>LW9219 (16-bit Unsigned)</i>
密碼 :	<input type="text" value="111"/>	<i>LW9220 (32-bit Unsigned)</i>
目前狀態 :	<div style="display: flex; justify-content: space-between;"> <i>bit 15</i> <i>bit 0</i> </div> <div style="border: 1px solid black; padding: 2px; margin-top: 5px;"> 0000000000000001 <i>LW9222</i> </div>	
<input type="button" value="登出"/>		<i>LW9050 (登出)</i>
<input type="button" value="Class A 按鈕"/>		

6. 此時若按下 [登出] 強迫用戶登出，可以發現系統將回復到起始狀態，此時只允許操作“無”類別的物件。

使用者名稱 :	<input type="text" value="3"/> LW9219 (16-bit Unsigned)				
密碼 :	<input type="text" value="333"/> LW9220 (32-bit Unsigned)				
目前狀態 :	<table><tr><td>bit 15</td><td>0000000000000000</td><td>bit 0</td><td>LW9222</td></tr></table>	bit 15	0000000000000000	bit 0	LW9222
bit 15	0000000000000000	bit 0	LW9222		
<input type="button" value="登出"/> LB9050 (登出)					



- **密碼輸入：**當密碼輸入錯誤時，[LB-9060] 的狀態將被設定為 ON 狀態；當密碼輸入成功時，[LB-9060] 的狀態將自動被恢復為 OFF 狀態。用戶 1 至用 戶 12 所有用 戶的密 碼可以利用 讀取 系統保留暫存器 [LW-9500] 至 [LW- 9522]，共 24 words 的內容取得。
- **線上更改密碼：**當 [LB-9061] 的狀態設定為 ON 時，系統將讀取 [LW-9500] 至 [LW-9522] 內的數值，更新用 戸的密 碼，往後並使用這些新的密 碼。此時用 戸可使用的物 件類別並不會因密 碼的變更而改變。

第十一章 索引暫存器

11.1 概要

索引暫存器是 EasyBuilder 提供用於變換位址的暫存器。有了索引暫存器後，使用者可以在不改變物件位址內容的情況下，在 HMI 直接線上修改物件的讀取與寫入位址。EasyBuilder 提供 32 組索引暫存器，分別為 16 組 16-bit 的索引暫存器和 16 組 32-bit 的索引暫存器。

位址



+ Index register →



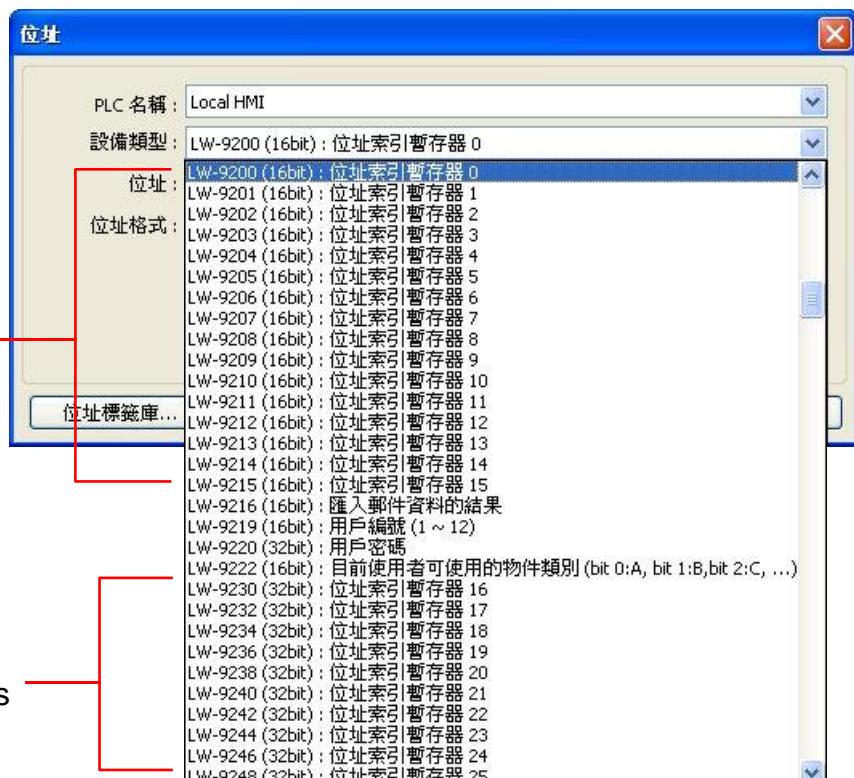
32 組索引暫存器的位址分別為：

16-bit 索引暫存器：

Index 0 [LW-9200] (16-bit)~

Index 15 [LW-9215] (16-bit)

最大定址範圍 65536 words



32-bit 索引暫存器：

Index 16 [LW-9230] (32-bit)~

Index 31 [LW-9260] (32-bit)

最大定址範圍 4294967296 words

使用【索引暫存器】後，所使用【設備類型】的位址則由下列公式決定：

“設定的常數位址 + 所選擇索引暫存器中的值”



索引暫存器對【系統參數設定】中所有的【設備列表】都有效，但只對字元格式的位址有效。

11.2 範例

以一個簡單設定說明索引暫存器的使用方式：

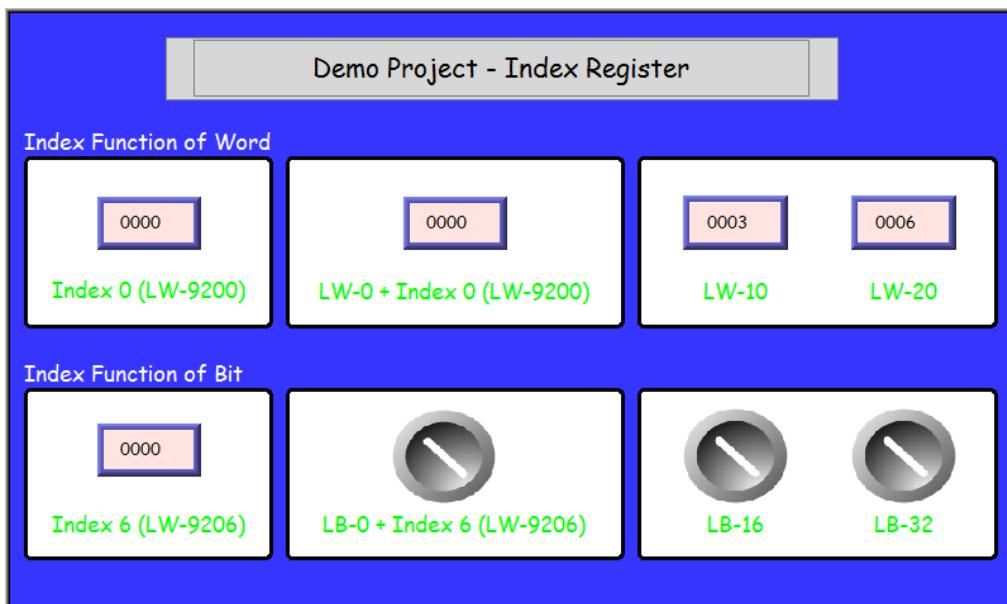
若未勾選 [索引暫存器]：
讀取位址為 [LW-10]。
系統直接對此位址做讀取/寫入
的動作。



若勾選 [索引暫存器]：
且選擇索引暫存器 [INDEX 0]：
讀取位置為 [LW-0 + INDEX 0]
INDEX 0 : 索引暫存器 0 為
[LW-9200]。
如果此時 [LW-9200] 位址中的
數值為 “5”，根據計算公式 “設定的常數位址 + 所選擇索引暫存
器中的值” 推得的結果為讀取位
址是 [LW(0+5)]，即是 [LW-5]。

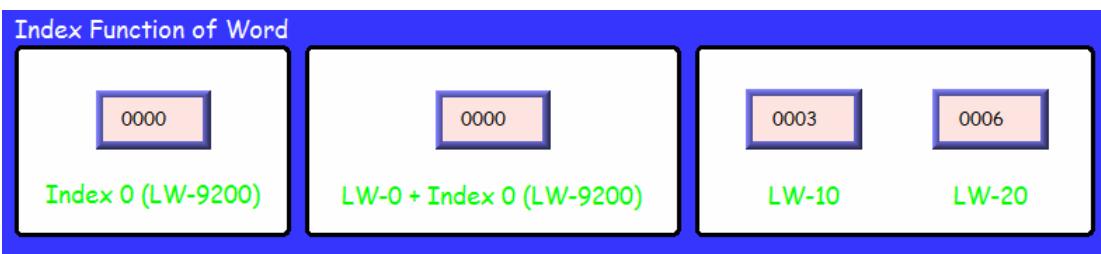


現在以一個實際範例作進一步說明：



Example 1

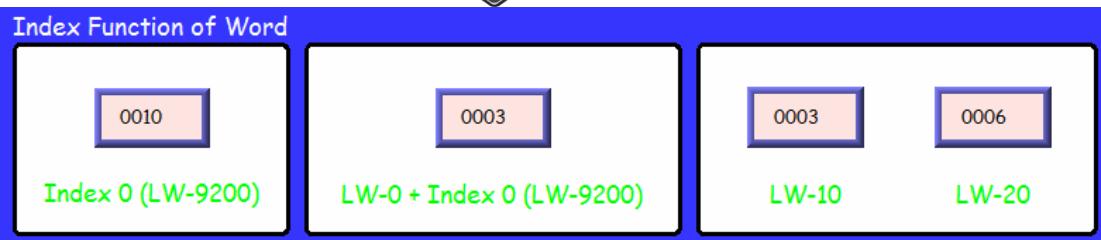
下圖顯示 Index Function of Word



Index 0 為 “0”
= [LW-9200] 位址
中的資料為“0”

讀取 [LW-0 + Index 0]
= 讀取 [LW-0] 內容

[LW-10] 設為 “3”
[LW-20] 設為 “6”



Index 0 [LW-9200]
設為 “10”

讀取 [LW-0 + Index 0]
= 讀取 [LW-10] = “3”

Example 2

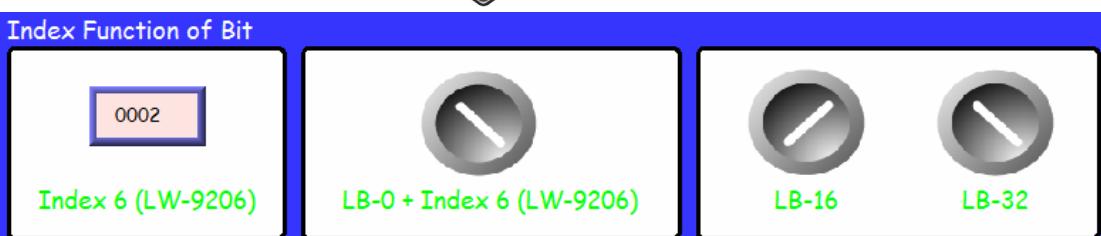
如下圖 Index Function of Bit 為例，同樣的，索引暫存器也可以用於位元位址。

1 個字元 = 16 個位元，所以索引暫存器數值改變 1 相當於 16 個位元。



Index 6 [LW-9206]
設為 “1”

開關 [LB-0 + Index 6] 讀取 LB-16 位址狀態 = ON



Index 6 設為 “2”

開關 [LB-0 + Index 6] 讀取 LB-32 位址狀態 = OFF



- 使用索引暫存器於位元位址時，所設定的位元位址將會以 16 個位元位址為一個單位。例如：以 LB-0 為目標，若是 `index` 裡的數值為 1，則 LB-16 將會動作，若是 `index` 裡的數值為 2，則 LB-32 會動作。

索引暫存器其實就是一個變址定址的暫存器，藉由索引暫存器可以在不改變設備位址的情況下，只要改變索引暫存器中的資料，即可改變同一個物件讀取或寫入不同的位址的資料。透過使用索引暫存器，可以實現不同位址之間資料的傳送或交換等功能。

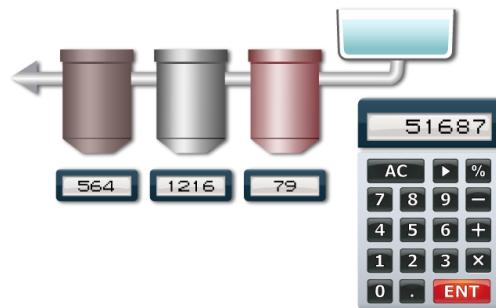


下載範例程式前，請先確定已連上網路線。

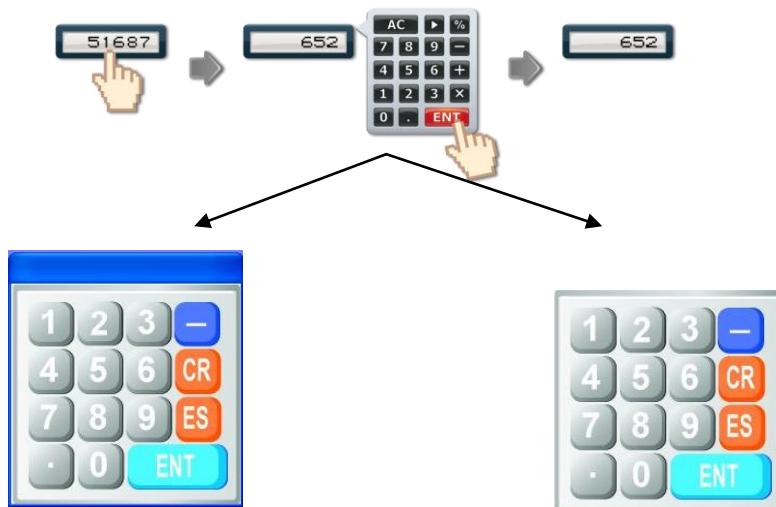
第十二章 鍵盤的設計與使用

數值輸入與字元輸入都需要使用鍵盤做為輸入工具，而數字鍵盤以及字元鍵盤均是使用功能鍵物件來製作的。鍵盤種類可分為：

1. 固定鍵盤，置於視窗固定位置。



2. 彈出鍵盤。



有視窗控制條的彈出視窗鍵盤。

沒有視窗控制條的彈出視窗鍵盤。

3. UNICODE 文字鍵盤。



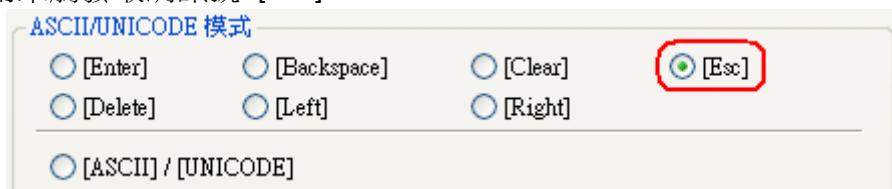
12.1 設計自製的彈出鍵盤

步驟 1. 先建立並開啟要作為鍵盤的視窗，假設為視窗 200。

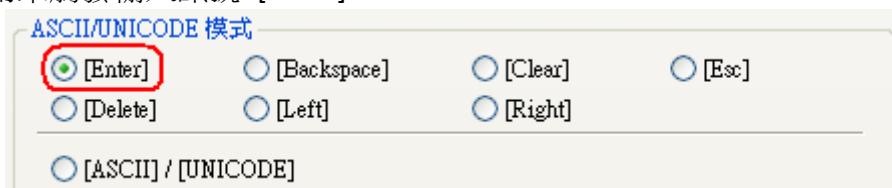


步驟 2. 調整視窗 200 的長度與寬度，建立各個功能鍵物件，並用 **[ASCII/UNICODE 模式]** 例如：

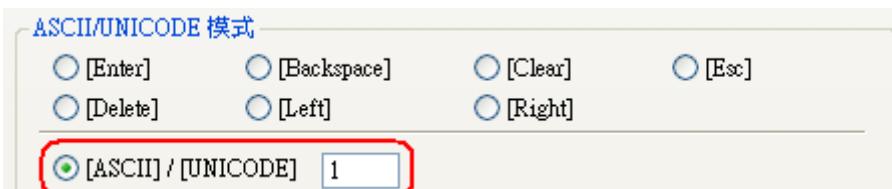
- [FK_11] 用來觸發取消訊號 [ESC]。



- [FK_14] 用來觸發輸入訊號 [Enter]。



- 其他大部分功能鍵用來觸發數值輸入訊號，例如 [FK_0] 是用來觸發數值 1 的輸入訊號。



步驟 3. 最後為功能鍵挑選適合的圖形，[GP_0] 為 圖片物件，置於所有物件最下層作為背景圖案。



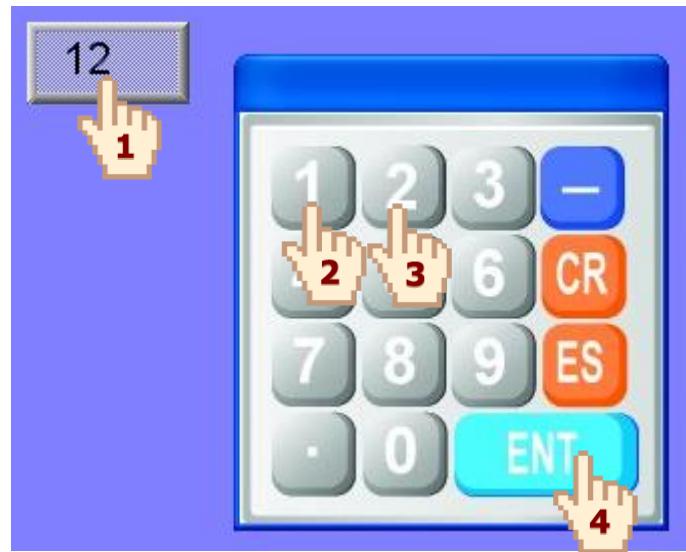
步驟 4. 在 [系統參數設定] » [一般屬性] » [鍵盤] 設定中，按下 [新增] 後選擇加入 [視窗 200]。最多可新增 32 個鍵盤視窗。



步驟 5. 在完成上述的所有步驟後，當使用者使用數值輸入與字元輸入物件的設定頁時，即可發現在 [數值輸入] » [鍵盤] 設定中的 [視窗編號]，增加了“200. Keyboard”的選項。
[鍵盤彈出位置] 可用來選擇鍵盤在 HMI 的出現位置，系統將 HMI 螢幕劃分為 9 個區域，如下圖：



步驟 6. 在選擇 **[200. Keyboard]** 後，當使用者按下數值輸入或字元輸入物件時，將自動彈出視窗 200，按其上的功能鍵將等同鍵盤輸入訊號。



12.2 使用直接視窗的方式來設計鍵盤

步驟 1. 新增一個直接視窗物件，設定讀取位址 LB-0 來啟動直接視窗。在【屬性】內選擇【隱藏視窗控制條】及設定鍵盤所在【視窗序號】。



步驟 2. 再次開啟此物件設定頁，在【輪廓】頁籤將物件尺寸設定等同鍵盤視窗的大小。



步驟 3. 新增數值輸入物件，在數值輸入頁籤內不要勾選 [使用彈出鍵盤]。



步驟 4. 設定一個位元狀態設定物件，寫入位址為 LB-0，開關類型為 [設為 ON]，並重疊在數值輸入物件上。當按下數值輸入物件的同時，也會將直接視窗即鍵盤開啟。



步驟 5. 在該需要彈出的“鍵盤”的 [Enter] 功能鍵和 [ESC] 功能鍵上，分別放置一個位元狀態設定物件，寫入位址為 LB-0，開關類型為 [設為 OFF]。在按下這兩個鍵的任意一個鍵時，可將直接視窗即鍵盤關閉。

12.3 將鍵盤固定在需要輸入的視窗上

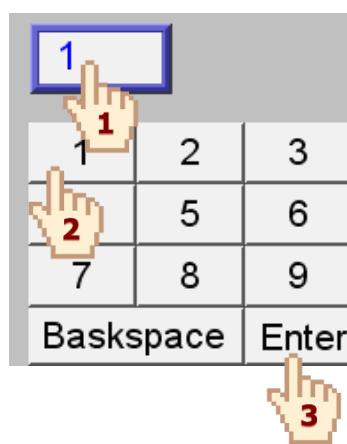
若不採用彈出鍵盤方式或是使用直接視窗來預設鍵盤所在位置，可採用此固定鍵盤方式，但這方式將無法移動或關閉鍵盤。

建立步驟如下：

步驟 1. 新增數值輸入物件，在【數值輸入】»【鍵盤】屬性中不要勾選【使用彈出鍵盤】。



步驟 2. 使用功能鍵物件將鍵盤按鍵設計好後，放置於視窗上即可使用，如下圖。



步驟 3. 當按下數值輸入物件時，使用者可以藉由鍵盤上的功能鍵來輸入數值。

12.4 製作 UNICODE 鍵盤

使用功能鍵物件製作 Unicode 鍵盤：

- 步驟 1. 放置一個字元輸入物件在視窗上，並勾選 [使用 UNICODE]。
- 步驟 2. 製作「威」、「綸」、「科」、「技」這四個文字輸入功能鍵，再製作一個 [Enter] 輸入功能鍵，即做好了一個簡單的文字鍵盤。
- 步驟 3.



- 使用者可以將自製的鍵盤設定群組為「群組圖片」並添加到「群組圖庫」中，以便於後續的調用。

第十三章 物件

本章說明各種物件的使用方式與設定。而物件通用的屬性請根據以下的指示參考相關的章節。

圖片、標籤、輪廓

請參考第九章「物件一般屬性」。

安全

請參考手冊第十章「使用者密碼與物件安全防護」。

索引暫存器

請參考手冊第十一章「索引暫存器」。

文字標籤庫

請參考手冊第十五章「文字標籤庫與多國語言使用」。

位址標籤庫

請參考手冊第十六章「位址標籤庫的建立與使用」。

13.1 位元狀態指示燈

概要

【位元狀態指示燈】 物件用來顯示位元暫存器的狀態。狀態 0 代表位元的狀態為 OFF；狀態 1 代表位元的狀態為 ON。



設定



按下工作列上的 **【位元狀態指示燈】** 按鈕後即會開啟 **【位元狀態指示燈】** 物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個 **【位元狀態指示燈】** 物件。

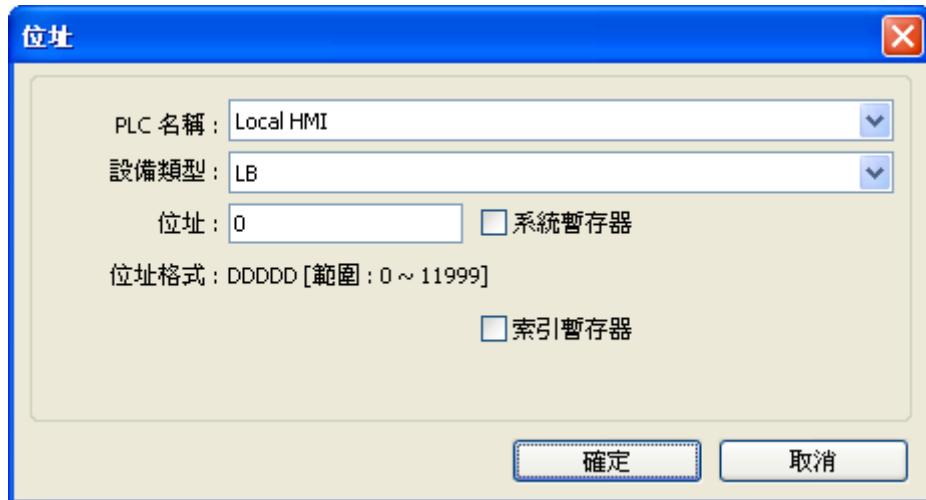


描述

使用者可為此物件描述相關訊息。

讀取位址

點選【設定】後選擇位元暫存器設備類型的 **[PLC 名稱]**、**[位址]**、**[設備類型]**、**[系統暫存器]**、**[索引暫存器]** 來控制位元狀態指示燈物件。使用者也可在 **【一般屬性】** 頁中設定位址。



【輸出反向】

可以將讀取的狀態作反向顯示，例如位元的狀態實際上為 OFF，但勾選了 **【輸出反向】** 後會顯示為 ON。

閃爍

設定物件在位元狀態為 ON 或 OFF 的顯示方式。

a. 無

不閃爍。

b. 狀態為 0 時顯示圖片

狀態為 OFF 時，圖片會進行狀態 0 及狀態 1 交互閃爍。

c. 狀態為 1 時顯示圖片

狀態為 ON 時，圖片會進行狀態 0 及狀態 1 交互閃爍。

d. 狀態為 0 時閃爍

狀態為 OFF 時，圖形 0 會進行出現與消失交互動作。

e. 狀態為 1 時閃爍

狀態為 ON 時，圖形 1 會進行出現與消失交互動作。

13.2 多狀態指示燈

概要

[多狀態指示燈] 物件利用字元暫存器內的數據，顯示相對的狀態與圖形(最高可支援 256 種狀態的顯示)。當暫存器內的數值為 0 時，顯示 [狀態 0]；當數值為 1 時，則顯示 [狀態 1]，依此類推。

數值顯示 (LW0) 多狀態顯示燈 (LW0)



數值顯示 (LW0) 多狀態顯示燈 (LW0)



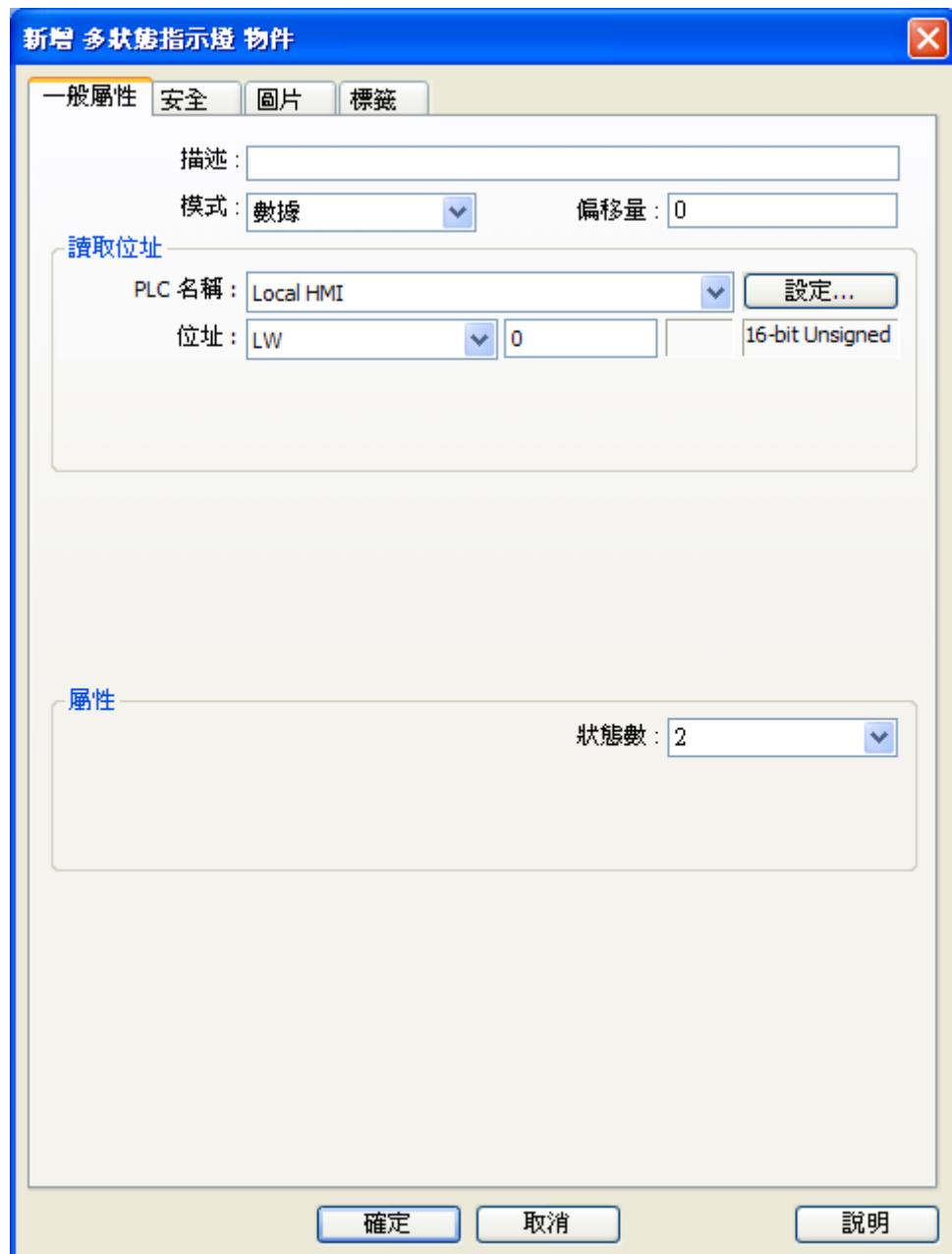
數值顯示 (LW0) 多狀態顯示燈 (LW0)



設定



按下工作列上的 [多狀態指示燈] 按鈕後即會開啟 [多狀態指示燈] 物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個 [多狀態指示燈] 物件。



[模式] / [偏移量]

[多狀態指示燈] 物件提供下列三種選擇模式：

a. 數據

直接利用暫存器內的數據加減 [偏移量] 的結果做為物件目前的狀態。例如下圖，當寫入一個數值 3 至暫存器地址 LW-200 時，因為有偏移量 3，所以地址 LW-200 的物件圖片會顯示狀態 6 (數值 3 + 偏移量 3)。



b. LSB

此模式首先會將暫存器內的數據先轉換為 2 進制，接著使用不為 0 的最低位元決定物件目前的狀態。以下表數據為例：

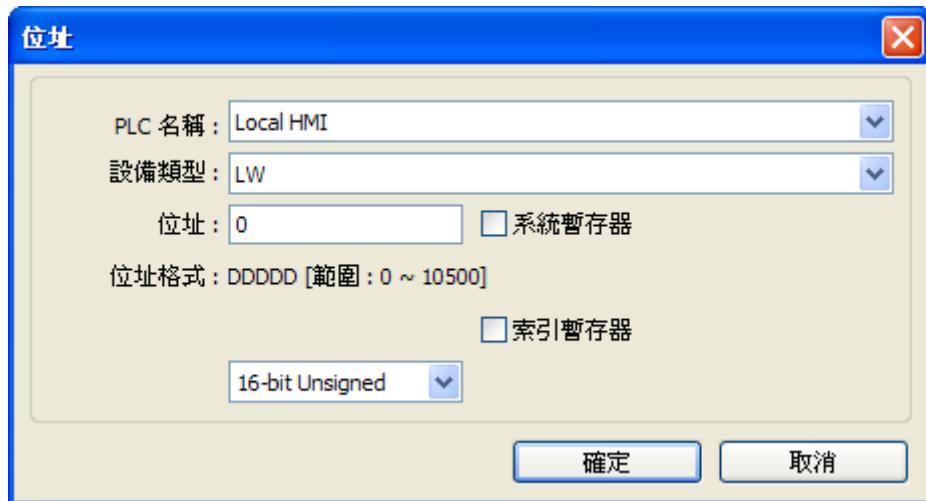
十進制	二進制	顯示的狀態
0	0000	全部 bit 皆為 0，則顯示狀態 0
1	0001	不為 0 的最低位元為 bit 0，此時顯示狀態 1
2	0010	不為 0 的最低位元為 bit 1，此時顯示狀態 2
3	0011	不為 0 的最低位元為 bit 0，此時顯示狀態 1
4	0100	不為 0 的最低位元為 bit 2，此時顯示狀態 3
5	0101	不為 0 的最低位元為 bit 0，此時顯示狀態 1
6	0110	不為 0 的最低位元為 bit 1，此時顯示狀態 2
7	0111	不為 0 的最低位元為 bit 0，此時顯示狀態 1
8	1000	不為 0 的最低位元為 bit 3，此時顯示狀態 4

c. 週期轉換狀態

物件的狀態會依照固定的頻率依序變換狀態。使用者可以利用 [頻率] 設定狀態改變頻率。

讀取位址

點選 **【設定】** 後選擇字元暫存器設備類型的 **【PLC 名稱】**、**【位址】**、**【設備類型】**、**【系統暫存器】**、**【索引暫存器】** 來控制多狀態指示燈物件。使用者也可在 **【一般屬性】** 頁中設定位址。



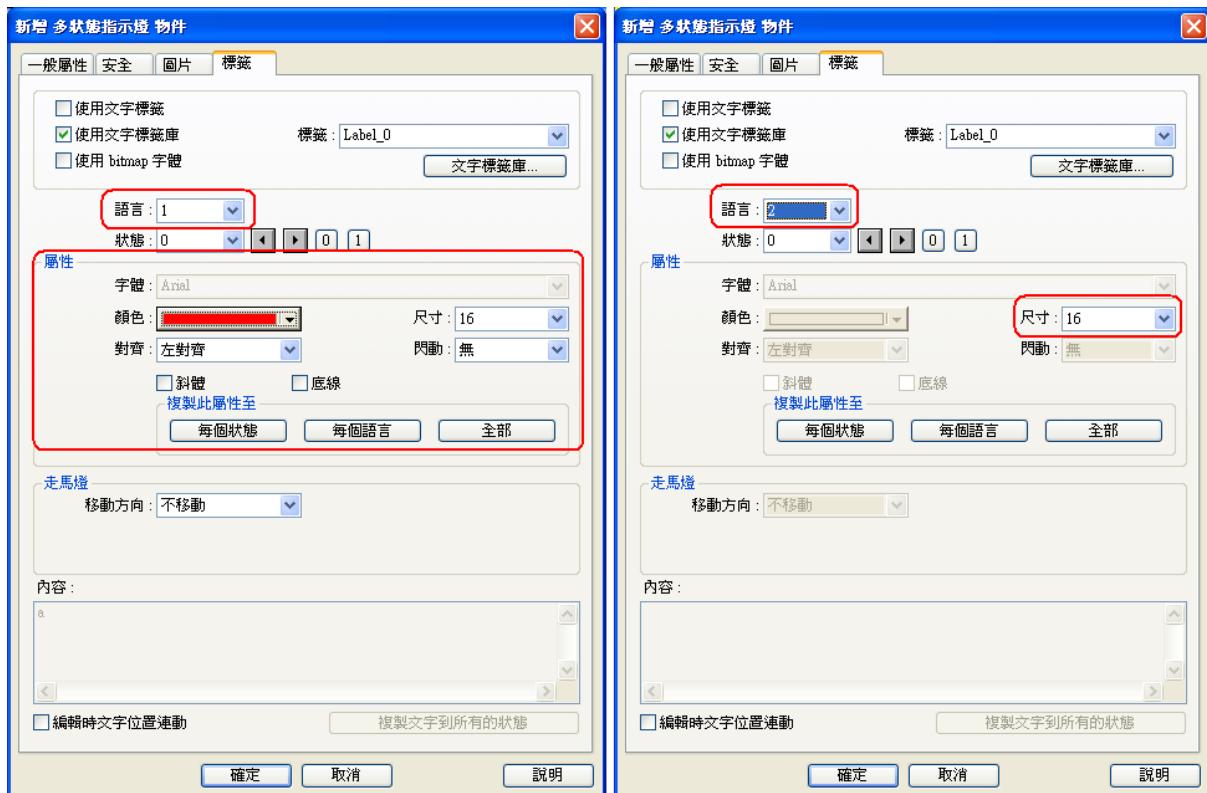
屬性

【狀態數】

物件顯示的狀態數目。狀態從 0 開始編號，能顯示的最大狀態編號為設定的 **【狀態數】 - 1**，當要求顯示超過設定的狀態數時，EasyBuilder 會顯示最後一個狀態。例如設定 **【狀態數】** 為 8，則顯示的狀態依序為 0, 1, 2, ..., 7，若要求暫存器顯示狀態 8 (含) 以上的狀態時，顯示的圖片僅顯示狀態 7。



在標籤頁中，語言 1 能夠改變字型相關屬性設定，但語言 2~8 只能改變字的尺寸，其他屬性設定皆與語言 1 相同。



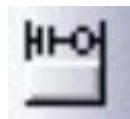
13.3 位元狀態設定

概要

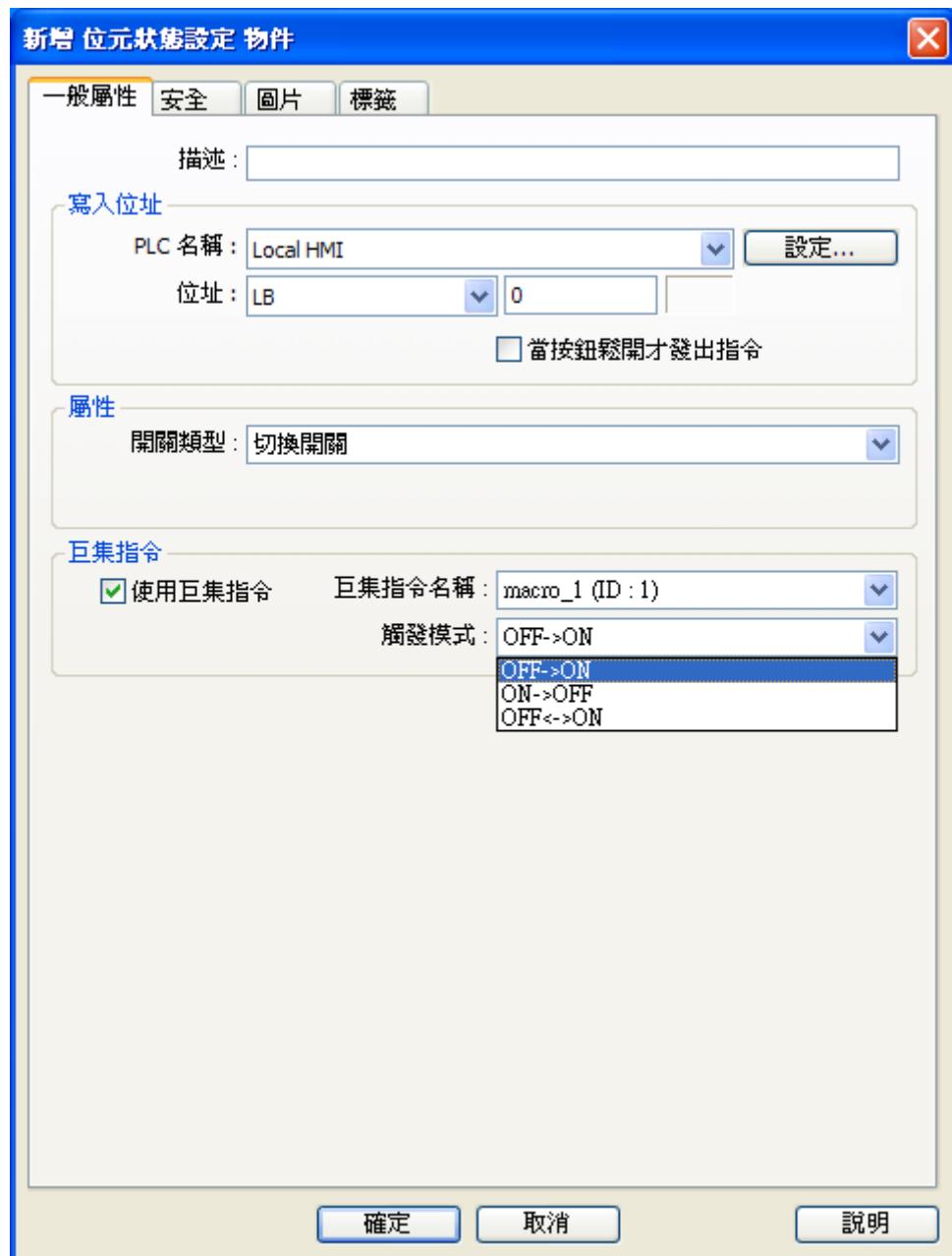
[位元狀態設定] 物件提供手動操作與自動執行兩種操作模式。使用手動操作模式，按壓此按鈕可以將暫存器的狀態設定為 ON 或 OFF。

若使用自動執行模式，則在某些特定條件下會自動執行指定的動作，使用此種操作模式，即使按壓此按鈕也不會有任何影響。

設定

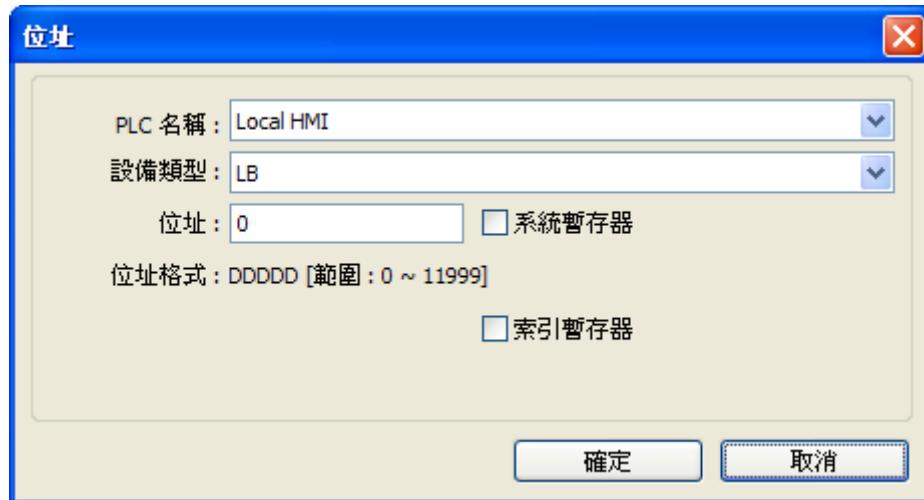


按下工作列上的 **[位元狀態設定]** 按鈕後即會開啟 **[位元狀態設定]** 物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個 **[位元狀態設定]** 物件。



寫入位址

點選【設定】後選擇位元暫存器設備類型的【PLC 名稱】、【位址】、【設備類型】、【系統暫存器】、【索引暫存器】來控制位元狀態設定物件。使用者也可在【一般屬性】頁中設定位址。



【當按鈕鬆開才發出指令】

使用此設定表示在按下物件後，必須完全鬆開按壓動作，物件定義的操作模式才會被執行。如未使用此項設定，只要一碰觸此區域，將立刻執行物件的動作。若選擇使用復歸型模式，將不支援此項功能。

屬性

【開關類型】

關於不同的開關類型請參閱以下表格敘述。

開關類型	敘述
設為 ON	按壓此物件後，所指定暫存器的狀態將被設定為 ON。
設為 OFF	按壓此物件後，所指定暫存器的狀態將被設定為 OFF。
切換開關	按壓此物件後，所指定暫存器的狀態將被反相。
復歸型	按壓此物件後，所指定暫存器的狀態將先被設定為 ON，但手放開後，狀態將被設定為 OFF。
週期切換開關	所指定暫存器的狀態將在 ON 與 OFF 間週期性切換，此模式為自動執行。可設定的週期為 0.1 秒 ~ 25.5 秒。
視窗打開時設 ON	物件所在位置的視窗被打開時，所指定暫存器的狀態將自動被設定為 ON。
視窗打開時設 OFF	物件所在位置的視窗被打開時，所指定暫存器的狀態將自動被設定為 OFF。

視窗關閉時設 ON	物件所在位置的視窗被關閉時，所指定暫存器的狀態將自動被設定為 ON。
視窗關閉時設 OFF	物件所在位置的視窗被關閉時，所指定暫存器的狀態將自動被設定為 OFF。
當背光燈開時設 ON	當背光燈打開時，所指定暫存器的狀態將自動被設定為 ON。
當背光燈開時設 OFF	當背光燈打開時，所指定暫存器的狀態將自動被設定為 OFF。
當背光燈關時設 ON	當背光燈關閉時，所指定暫存器的狀態將自動被設定為 ON。
當背光燈關時設 OFF	當背光燈關閉時，所指定暫存器的狀態將自動被設定為 OFF。

巨集指令

[位元狀態設定] 物件可以搭配執行巨集命令。選擇此項功能前需先建立巨集命令，如何建立巨集命令請參考第十八章-巨集指令說明。

觸發模式



當物件的操作模式，選擇 [切換開關] 時，設定執行巨集命令的條件，可以選擇狀態由 OFF 變為 ON 或由 ON 變為 OFF 時，才執行巨集命令，也可選擇狀態改變時(ON<>OFF)，即執行巨集命令。

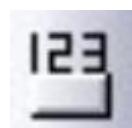
13.4 多狀態設定

概要

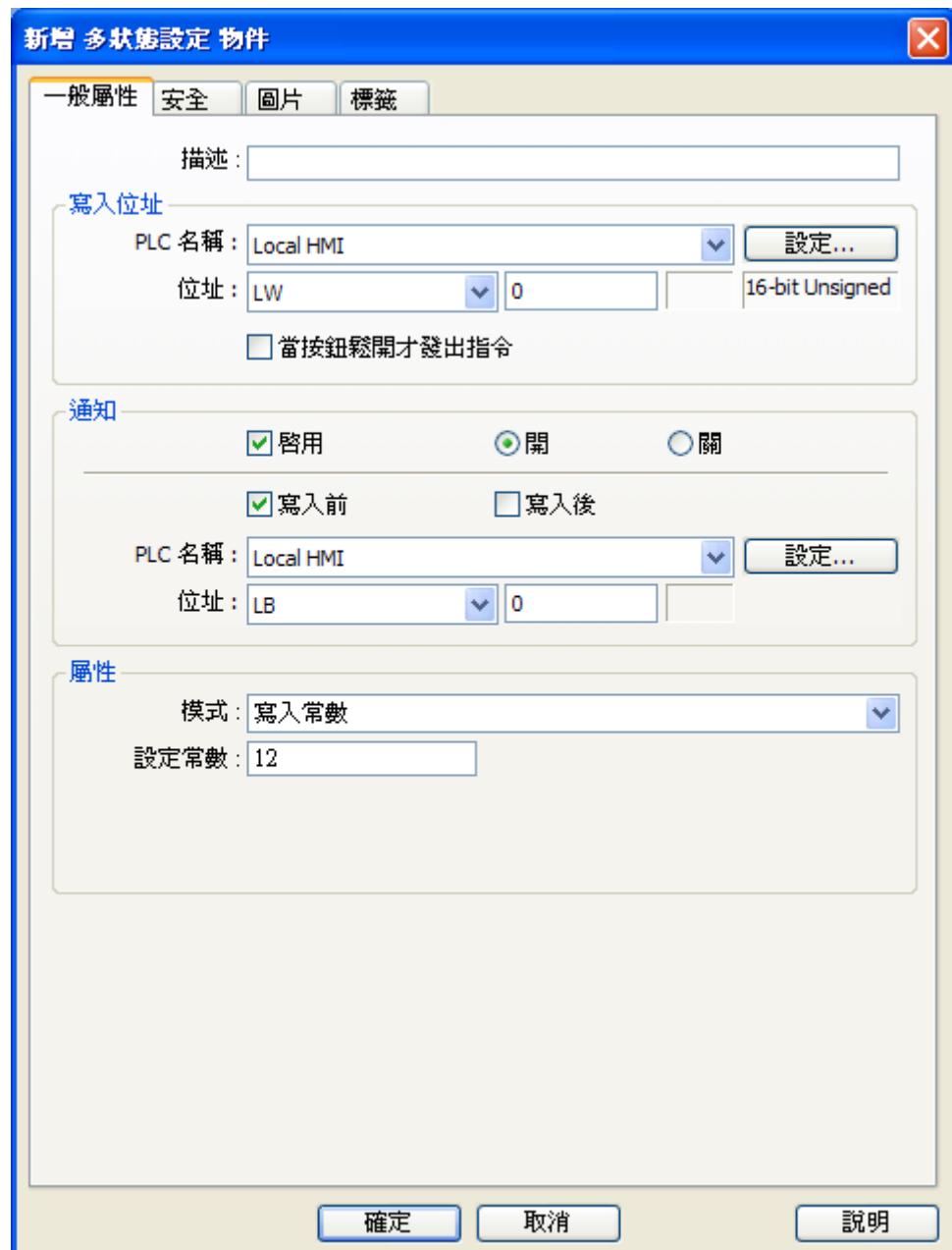
[多狀態設定] 物件提供手動操作與自動執行兩種操作模式。使用手動操作模式，按壓此按鈕可以設定暫存器內的數據。

若使用自動執行模式，則在某些特定條件下會自動執行指定的動作，使用此種操作模式，即使按壓此按鈕也不會有任何影響。

設定

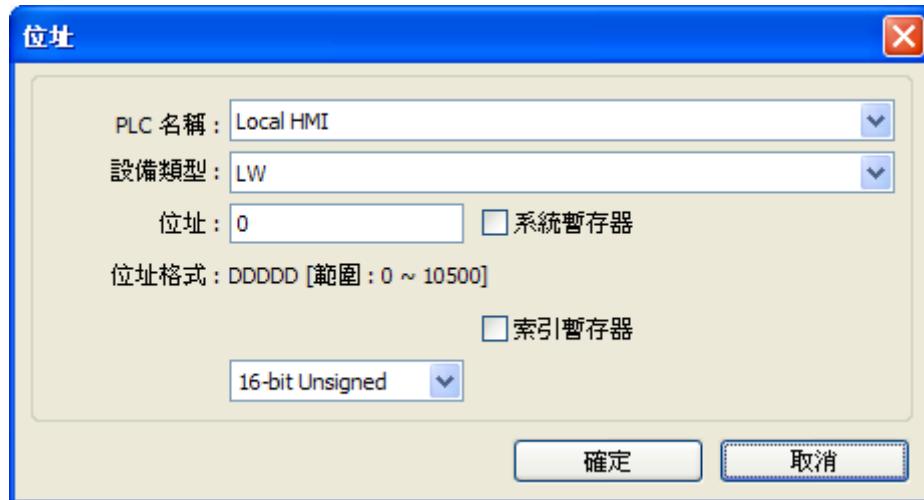


按下工作列上的**[多狀態設定]**按鈕後即會開啟**[多狀態設定]**物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個**[多狀態設定]**物件。



寫入位址

點選【設定】後選擇字元暫存器設備類型的【PLC 名稱】、【位址】、【設備類型】、【系統暫存器】、【索引暫存器】來控制多狀態設定物件。使用者也可在【一般屬性】頁中設定位址。



【當按鈕鬆開才發出指令】

使用此項設定表示在按壓此按鈕後，必須完全離開此區域才會執行物件定義的動作。如未使用此項設定，則只要一按壓此鈕，將立刻執行物件定義的動作。

通知

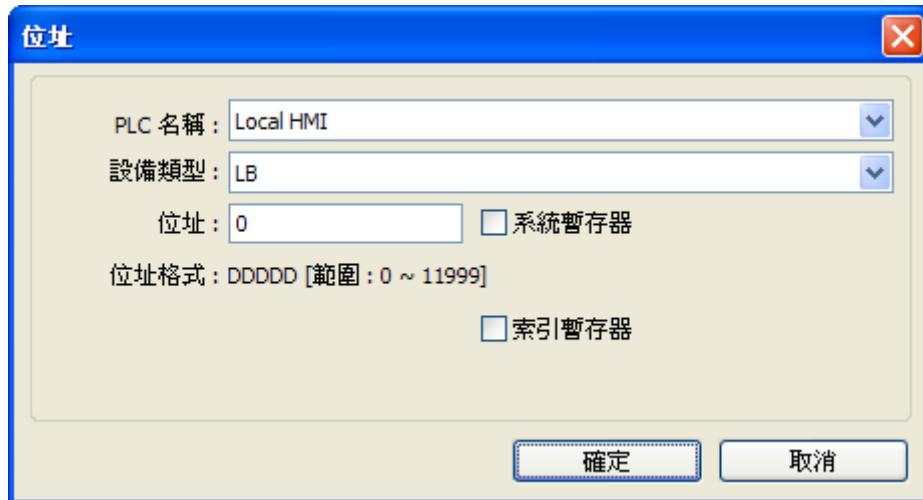
使用此項設定，則在使用手動操作模式時，在完成動作後可以連帶設定此項目所指定暫存器的狀態，使用 [開] 與 [關] 選擇要設定的狀態。



【寫入前】 / 【寫入後】

在寫入動作前 / 後設定所指定暫存器的狀態。

點選 [設定] 後選擇位元暫存器設備類型的 [PLC 名稱]、[位址]、[設備類型]、[系統暫存器]、[索引暫存器] 來控制位元狀態設定物件。使用者也可在 [一般屬性] 頁中設定位址。



屬性

[模式] 選擇物件的動作模式，可以選擇模式如下：

■ 寫入常數

設定常數功能。每按壓一次物件，[設定常數] 中的設定值將寫至指定的暫存器中。常數的型態可為 16-bit BCD、32-bit BCD、...、32-bit float 等。以右圖為例，當按壓此按鈕後，會將數值 12 寫入指定的暫存器中。



■ 遞加 (JOG+)

加值功能。每按壓一次物件，所指定暫存器內的數據將加上 [遞加值] 中設定的增量值，但增值的結果將不超過 [上限值] 中的設定值。以右圖為例，每按壓一次此按鈕後，會將指定的暫存器中的數值 +1，直至抵達上限值 10。



■ 遞減 (JOG-)

減值功能。每按壓一次物件，所指定暫存器內的數據將減去 [遞減值] 中設定的減量值，但是減值的結果不會低於 [下限值] 中的設定值。以右圖為例，每按壓一次此按鈕後，會將指定的暫存器中的數值 -1，直至抵達下限值 0。



■ 按住按鈕時遞加 (JOG++)

按住按鈕時遞加功能。若按壓物件超過 [遲滯時間] 設定時間，則所指定暫存器內的數據將以 [遞加速度] 所設定的速度，每次增加 [遞加值] 中設定的增量

值，但增量的結果將不超過 [上限值] 中的設定值。以右圖為例，每按壓一次此按鈕後，會將指定的暫存器中的數值+1，若按住此按鈕的時間超過 1.0 秒後，會以每 0.5 秒的速度持續+1 直到抵達上限值 10。

屬性

模式:	按住按鈕時遞加 (JOG++)		
遞加值:	1	上限值:	10
遲滯時間:	1.0 秒	遞加速度:	0.5 秒

■ 按住按鈕時遞減 (JOG--)

按住按鈕時遞減功能。若按壓物件超過 [遲滯時間] 的設定時間，則所指定暫存器內的數據將以 [遞加速度] 所設定的速度，每次減少 [遞減值] 中設定的減量

值，但減值的結果不會低於 [下限值] 中的設定值。以右圖為例，每按壓一次此按鈕後，會將指定的暫存器中的數值-1，若按住此按鈕的時間超過 1.0 秒後，會以每 0.5 秒的速度持續-1 到抵達下限值 0。

屬性

模式:	按住按鈕時遞減 (JOG--)		
遞減值:	1	下限值:	0
遲滯時間:	1.0 秒	遞加速度:	0.5 秒

■ 週期迴圈 (0->最大值->0)

週期性遞加功能。[多狀態設定] 物件會使用 [頻率] 設定的週期與 [遞加值] 中設定的增量值，自動增量所指定暫存器內的數據，但增量的結果將不超過 [上限值] 中的設定值。以右圖為例，系統會自動將指定的暫存器中的數值以每 0.5 秒的頻率+1，直到抵達上限值 10，接著數值會返回 0 再重新持續+1。

屬性

模式:	週期迴圈 (0->最大值->0...)		
遞加值:	1	上限值:	10
頻率:	0.5 秒		

■ 自動遞增 (增至上限值)

週期性遞增功能。[多狀態設定] 物件會使用 [頻率] 設定的週期，自動將所指定暫存器內的數據加上 [遞加值] 中設定的增量值，當結果等於 [上限值] 時自動停止。以右圖為例，系統會自動將指定的暫存器中的數值以每 0.5 秒的頻率+1，直到抵達上限值 10 後停止。

屬性

模式:	自動遞增 (增至上限值)		
遞加值:	1	上限值:	10
頻率:	0.5 秒		

■ 自動遞減 (減至下限值)

週期性遞減功能。[多狀態設定]

物件會使用 [頻率] 設定的週期，自動將所指定暫存器內的數據減去 [遞減值] 中設定的減量值，當結果等於 [下限值] 時自動停止。

以右圖為例，系統會自動將指定的暫存器中的數值以每 0.5 秒的頻率-1，直到抵達下限值 10 後停止。

屬性

模式 :	自動遞減 (減至下限值)
遞減值 :	1
下限值 :	10
頻率 :	0.5 秒

■ 週期迴圈 (自定範圍)

週期性迴圈功能。[多狀態設定]

物件會使用 [頻率] 設定的週期，每次將所指定暫存器內的數據加上 [遞加值] 中的設定值，直到暫存器內的數據等於 [上限值]；接著使用相同的週期，將暫

存器內的數據減去 [遞加值] 中的設定值，直到暫存器內的數據等於 [下限值]。以右圖為例，系統會自動將指定的暫存器中的數值以每 0.5 秒的頻率+1，直到抵達上限值 10，接著再以相同的頻率-1 直到等於下限值 0，如此週而復始的執行不停止。

屬性

模式 :	週期迴圈 (自定範圍)
下限值 :	0
上限值 :	10
遞加值 :	1
頻率 :	0.5 秒

■ 週期遞加 (從低到高)

步進功能。[多狀態設定] 物件會

使用 [頻率] 設定的週期，每次將所指定暫存器內的數據加上 [遞加值] 中的設定值，直到暫存器內的數據等於 [最大值]，接著會將暫存器內的數據復歸為

[最小值]，並重複先前的動作，讓數據一直保持動態變化。以右圖為例，系統會自動將指定的暫存器中的數值以每 0.5 秒的頻率+1，直到抵達上限值 10，接著再返回下限值 0 重新遞增，如此週而復始的執行不停止。

屬性

模式 :	週期遞加 (從低到高...)
最小值 :	0
最大值 :	10
遞加值 :	1
頻率 :	0.5 秒

■ 週期遞減 (從高到低)

步退功能。[多狀態設定] 物件會

使用 [頻率] 設定的週期，每次將所指定暫存器內的數據減去 [遞減值] 中的設定值，直到暫存器內的數據等於 [最小值]，接著

會將暫存器內的數據復歸為 [最大值]，並重複先前的動作，讓數據一直保持動態變化。以右圖

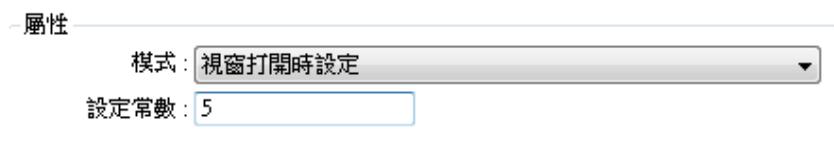
屬性

模式 :	週期遞減 (從高到低...)
最小值 :	0
最大值 :	10
遞減值 :	1
頻率 :	0.5 秒

為例，系統會自動將指定的暫存器中的數值以每 0.5 秒的頻率-1，直到抵達下限值 0，接著再返回上限值 10 重新遞減，如此週而復始的執行不停止。

■ 視窗打開時設定

開啟物件所在位置的視窗時，會將 [設定常數] 中的設定值自動寫至指定的暫存器中。以右圖為例，當當頁的視窗被開啟時，系統會自動將數值 5 寫入暫存器中。



■ 視窗關閉時設定

關閉物件所在位置的視窗時，會將 [設定常數] 中的設定值自動寫至指定的暫存器中。以右圖為例，當當頁的視窗被關閉時，系統會自動將數值 5 寫入暫存器中。



■ 當背光燈開時設定

當背光燈原處在關閉狀態，若恢復為開啟狀態時，會將 [設定常數] 中的設定值自動寫至指定的暫存器中。以右圖為例，當當頁的背光燈被開啟時，系統會自動將數值 5 寫入暫存器中。



■ 當背光燈關時設定

當背光燈原處在開啟狀態，若關閉背光燈時，會將 [設定常數] 中的設定值自動寫至指定的暫存器中。以右圖為例，當當頁的背光燈被關閉時，系統會自動將數值 5 寫入暫存器中。



■ 循環遞加 (JOG+)

加值功能。每按壓一次物件，所指定暫存器內的數據將加上 [遞加值] 中設定的增量值。當增量值達到上限時，會復歸回下限再重新遞增。以右圖為例，每按壓一次此物件後，會將指定的暫存器中的數值+1，當抵達上限值 10 後會自動復歸回 0 再遞增執行。

屬性

模式 : 循環遞加 (JOG+)	
下限值 : 0	上限值 : 10
遞加值 : 1	

■ 循環遞減 (JOG-)

減值功能。每按壓一次物件，所指定暫存器內的數據將減去 [遞減值] 中設定的減量值。當減量值達到下限時，會復歸回上限再重新遞減。以右圖為例，每按壓一次此物件後，會將指定的暫存器中的數值-1，當抵達下限值 0 後會自動復歸回 10 再遞減執行。

屬性

模式 : 循環遞減 (JOG-)	
下限值 : 0	上限值 : 10
遞減值 : 1	

■ 按住按鈕時循環遞加 (JOG++)

持續遞加功能。當按住按鈕的時間超過 [遲滯時間] 時，此物件會根據 [遞加速度] 的設定將指定暫存器的數據持續的遞加至上限值，之後會復歸回下限值。以右圖為例，每按壓一次此按鈕後，會將指定的暫存器中的數值+1，若按住此按鈕的時間超過 0.5 秒後，會以每 0.1 秒的速度持續+1 直到抵達上限值 10，接著會復歸回 0 再遞增執行。

屬性

模式 : 按住按鈕時循環遞加 (JOG++)	
下限值 : 0	上限值 : 10
遞加值 : 1	
遲滯時間 : 0.5 秒	遞加速度 : 0.1 秒

■ 按住按鈕時循環遞減 (JOG- -)

持續遞減功能。當按住按鈕的時間超過 [遲滯時間] 時，此物件會根據 [遞減速度] 的設定將指定暫存器的數據遞減至下限值，之後會復歸回上限值。以右圖為例，每按壓一次此按鈕後，會將指定的暫存器中的數值-1，若按住此按鈕的時間超過 0.5 秒後，會以每 0.1 秒的速度持續-1 直到抵達下限值 0，接著會復歸回 10 再遞增執行。

屬性

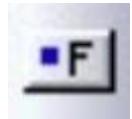
模式 : 按住按鈕時循環遞減 (JOG- -)	
下限值 : 0	上限值 : 10
遞減值 : 1	
遲滯時間 : 0.5 秒	遞加速度 : 0.1 秒

13.5 功能鍵

概要

[功能鍵] 物件提供視窗的切換、鍵盤的製作、巨集的執行及畫面列印等，也可用來設定 USB 安全金鑰使用。

設定



按下工作列上的 **[功能鍵]** 按鈕後即會開啟 **[功能鍵]** 物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個 **[功能鍵]** 物件。



【鬆開按鍵時觸發該指令】

使用此選項表示必須在釋放按壓物件的動作後，選擇的動作才會被執行。若未選擇，則在碰觸物件後，將立刻執行選擇的動作。

視窗切換

[切換基本視窗] 切換基本視窗。

[切換公共視窗] 切換公用視窗。

[彈出視窗]

呼叫其他視窗。此時呼叫出的視窗必定在基本視窗的上面。使用

此功能可以選擇是否使用 **[當父視窗被關閉時結束彈出視窗]**，參考右圖。選擇此屬性則呼叫出的視窗會在發生換頁動作時自動消失，否則使用者必須自行在被呼叫出的視窗上設計 **[關閉視窗]** 功能鍵來關閉此視窗。



[返回上一個視窗]

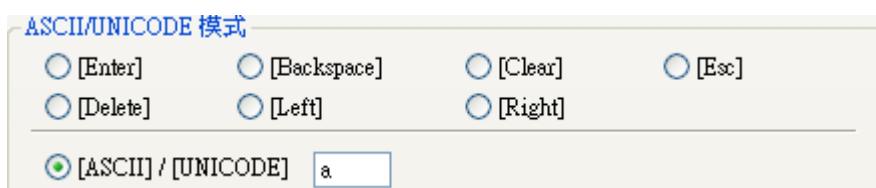
返回前一頁基本視窗。例如當由 “視窗 10” 切換到 “視窗 20” 時，使用此功能可以再返回”視窗 10”。此功能只對基本視窗有效。

[關閉視窗]

關閉在基本視窗上被呼叫出的視窗，包括訊息視窗。

ASCII/UNICODE 模式

被用來作為鍵盤的輸入訊號，主要用在 **[數值輸入]** 與 **[字元輸入]** 物件需要使用鍵盤來輸入數字或文字的場合。



[Enter] 與鍵盤的輸入 (enter) 動作相同。

[Backspace] 與鍵盤的後退刪除 (backspace) 動作相同。

[Clear] 清除暫存器中已輸入的資料。

[Esc] 與使用 **[關閉視窗]** 功能相同，可用來關閉彈跳出的鍵盤視窗。

[Delete] 與鍵盤的刪除 (delete) 動作相同，可將游標右方的一個字元刪除。

[Right] 與鍵盤的→動作相同，可將游標向右移動一個字元。

[Left] 與鍵盤的←動作相同，可將游標向左移動一個字元。

[ASCII/UNICODE] 設定鍵盤的輸入字元。

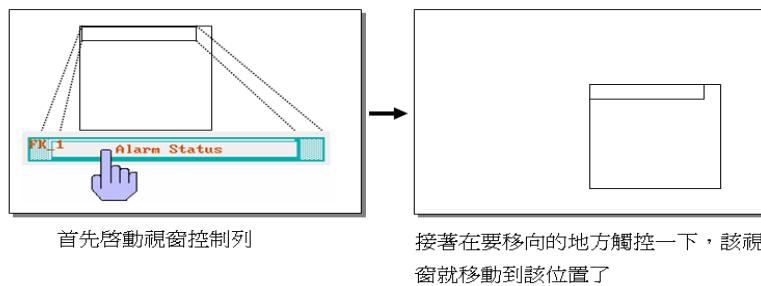
觸發巨集指令

選擇此項功能，將執行指定的巨集命令，選擇此項功能前需先建立巨集命令。如何建立巨集命令請參考手冊第十八章-巨集指令說明。



視窗控制條

當彈出的視窗無視窗控制條時，若需要移動視窗，則先點一下此物件，在移動的目的地再點一下，則視窗就會被移動到指定的位置。



畫面列印

此項功能用來列印當前的畫面。要選擇此項功能前需先在 **【系統參數設定】** » **【HMI 屬性】** 中選擇所使用的印表機類型。使用單色印表機時，勾選 **【灰階效果】** 可以提升畫面的辨識度，但也會影響文字的顯示效果，因此如果是強調文字的列印效果，請不要使用灰階功能。



通知

使用此項設定，則在完成動作後可以連帶設定此項目所指定暫存器的狀態，使用 **【開】** 與 **【關】** 選擇要設定的狀態。



1. 第六章 - 視窗的說明
2. 第十二章 - 鍵盤的設計與使用

Example 1 設計非 ASCII 文字鍵盤

下文說明如何在 HMI 上輸入與顯示非 ASCII 文字(例如繁體中文, 簡體中文, 日文, 希臘文等), 詳細說明各個步驟。

步驟 1: 設定非 ASCII 文字所使用的字體

首先規劃非 ASCII 文字所使用的字體，並將這些字體加入系統參數設定的【字體】設定頁中。請參考下圖。



步驟 2: 設計非 ASCII 文字輸入鍵盤

此時新增 "窗口 11" 作為非 ASCII 文字的輸入鍵盤，鍵盤的頁面規劃內容參考下圖。



此時窗口上的物件皆為功能鍵物件，並依實際用途選擇適當的屬性。以作為輸入 "簡" 此文字的功能鍵為例，需選擇 [ASCII] / [UNICODE] 模式，並將輸入內容設定為"簡"，參考下圖。



接著在 [標籤] 頁中，選擇 [使用文字標籤]，並將標籤內容設定為 ”簡”，最重要的是字體選擇為 ”AR MingtiM GB” 字體，必須與步驟一的設定內容相搭配，參考下圖。

另外，在同一個鍵盤中作為非 ASCII 文字輸入用途的功能鍵，所使用的字體必須都相同，以簡體中文鍵盤為例，此時作為非 ASCII 文字輸入用途的功能鍵都選擇使用 “AR MingtiM GB” 字體。



在完成所有按鍵的設定後，將窗口加入到系統參數設定的鍵盤列表中，參考下圖。



13.6 位元狀態切換開關

概要

【位元狀態切換開關】為【位元狀態指示燈】物件與【位元狀態設定】物件的組合。此物件除了可以用來顯示暫存器的狀態外，也可以利用這個物件在視窗上定義一個碰觸區域，按壓此區域可以設定所指定暫存器的狀態為 ON 或 OFF。

設定



按下工作列上的【位元狀態切換開關】按鈕後即會開啟【位元狀態切換開關】物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個【位元狀態切換開關】物件。



讀取位址

點選 **【設定】** 後選擇位元暫存器設備類型的 **【PLC 名稱】**、**【位址】**、**【設備類型】**、**【系統暫存器】**、**【索引暫存器】** 來控制顯示位元狀態切換開關狀態物件。使用者也可在 **【一般屬性】** 頁中設定位址。

[輸出反向]

可以將讀取的狀態作反向顯示，例如獲得的狀態實際上為 OFF，但物件會顯示為 ON。



不使用輸出反向



使用輸出反向

寫入位址

點選 **【設定】** 後選擇位元暫存器設備類型的 **【PLC 名稱】**、**【位址】**、**【設備類型】**、**【系統暫存器】**、**【索引暫存器】** 來控制位元狀態切換開關物件。使用者也可在 **【一般屬性】** 頁中設定位址。此暫存器可以與“讀取位址”所指定的暫存器相同亦或不同。

[鬆開按鍵時觸發該指令]

使用此選項表示必須在釋放按壓物件的動作後，選擇的動作才會被執行。若未選擇，則在碰觸物件後，將立刻執行選擇的動作。

屬性

開關類型	敘述
設為 ON	按壓此物件後，所指定暫存器的狀態將被設定為 ON。
設為 OFF	按壓此物件後，所指定暫存器的狀態將被設定為 OFF。
切換開關	按壓此物件後，所指定暫存器的狀態將被反向。當狀態為 ON 時，會被設定為 OFF。當狀態為 OFF 時，則會被設定為 ON。
復歸型	按壓此物件後，所指定暫存器的狀態將先被設定為 ON，但手放開後，狀態將被設定為 OFF。

巨集指令

操作 **【位元狀態切換開關】** 物件時，可以搭配執行巨集命令，但要選擇此項功能前需先建立巨集命令。

13.7 多狀態切換開關

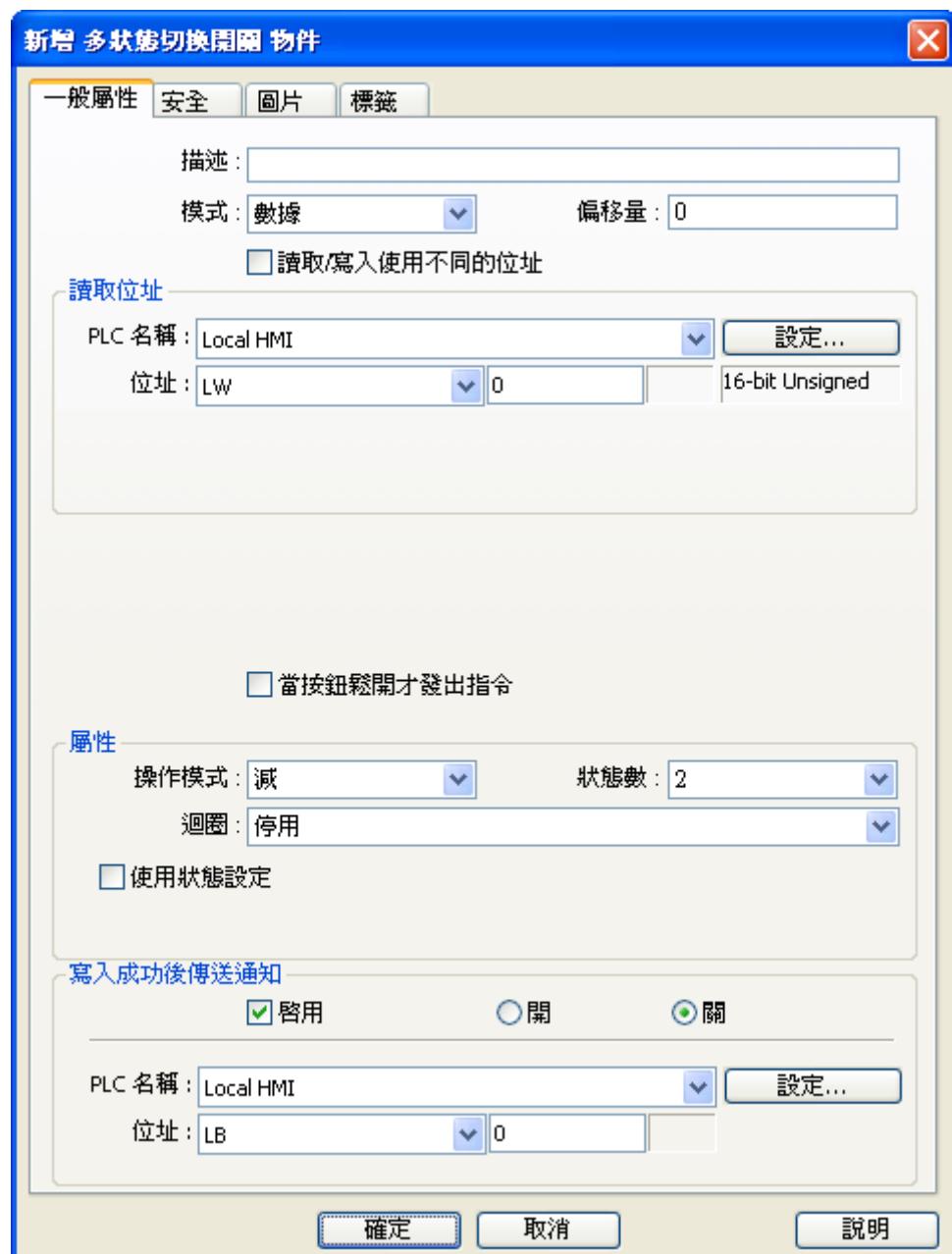
概要

【多狀態切換開關】物件為【多狀態指示燈】物件與【多狀態設定】物件的組合。此物件除了可以利用暫存器內的數據顯示不同的狀態外，也可以利用這個物件在視窗上定義一個碰觸區域，按壓此區域可以設定所指定暫存器內的數據。

設定



按下工作列上的【多狀態切換開關】按鈕後即會開啟【多狀態切換開關】物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個【多狀態切換開關】物件。



[模式] / [偏移量]

提供 [數據] 與 [LSB] 顯示模式，可參考 [多狀態指示燈] 物件的說明。

讀取位址

點選 [設定] 後選擇字元暫存器設備類型的 [PLC 名稱]、[位址]、[設備類型]、[系統暫存器]、[索引暫存器] 來控制顯示多狀態切換開關狀態物件。使用者也可在 [一般屬性] 頁中設定位址。

寫入位址

點選 [設定] 後選擇字元暫存器設備類型的 [PLC 名稱]、[位址]、[設備類型]、[系統暫存器]、[索引暫存器] 來控制多狀態切換開關物件。使用者也可在 [一般屬性] 頁中設定位址。此暫存器可以與 [讀取位址] 所指定的暫存器相同亦或不同。

【鬆開按鍵時觸發該指令】

使用此選項表示必須在釋放按壓物件的動作後，選擇的動作才會被執行。若未選擇，則在碰觸物件後，將立刻執行選擇的動作。

屬性

選擇物件的操作方式。



【操作模式】

選擇數據遞增或遞減，若 [偏移量] 不為 0，則顯示的狀態為 ([狀態數] -1) + 偏移量。

■ 加 (JOG+)

加值功能。每按壓一次物件，所指定暫存器內的數據+1，但增值的結果將不超過 [狀態數]。若 [啟用] 邊緣，則抵達最大狀態後會復歸回最低狀態 0。以下圖為例，若操作模式選擇 [加]，狀態數為 5 且 [啟用] 邊緣，則每按壓一次此物件，狀態會從狀態 0 會往上+1 直至狀態 4 ([狀態數] -1)，然後復歸回狀態 0 重新遞增。



■ 減 (JOG-)

減值功能。每按壓一次物件，所指定暫存器內的數據-1 直至 0。若 [啟用] 邊緣，則抵達最大狀態後會復歸回最高狀態。以下圖為例，若操作模式選擇 [減]，狀態數為 5 且 [啟用] 邊緣，則每按壓一次此物件，會往下-1 直至狀態 0，然後復歸回最高狀態 4 ([狀態數] -1) 重新遞減。



[使用狀態設定]

使用者可修改狀態對應的數值，亦可使用當有不合法的數值輸入時的動作狀態和通知指定位元切換狀態。

保持目前狀態：若輸入超出範圍的數值，多狀態切換開關會保持目前狀態。

跳至錯誤狀態：若輸入超出範圍的數值，多狀態切換開關會跳到錯誤狀態。

錯誤通知

當輸入無效的數值時，可以自動設定所指定位址的狀態。



寫入成功後傳送通知

啟用此項設定後，在命令傳送成功時會連帶將指定位元暫存器的狀態設定為 [開] 或 [關]。點選 [設定] 後選擇位元暫存器設備類型的 [PLC 名稱]、[位址]、[設備類型]、[系統暫存器]、[索引暫存器] 來控制位元狀態設定物件。使用者也可在 [一般屬性] 頁中設定位址。

13.8 滑動開關

概要

【滑動開關】物件是用來建立一個滑塊區域顯示數值或藉由拖曳滑軌改變指定暫存器內的數值。

設定



按下工作列上的【滑動開關】按鈕後即會開啟【滑動開關】物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個【滑動開關】物件。



寫入位址

點選 **[設定]** 後選擇字元暫存器設備類型的 **[PLC 名稱]**、**[位址]**、**[設備類型]**、**[系統暫存器]**、**[索引暫存器]** 來控制滑動開關物件。使用者也可在 **[一般屬性]** 頁中設定位址。

通知

使用此項設定，則在完成動作之前 / 之後可以連帶設定此項目所指定暫存器的狀態，使用**[開]**與**[關]**選擇要設定的狀態。

點選 **[設定]** 後選擇位元暫存器設備類型的 **[PLC 名稱]**、**[位址]**、**[設備類型]**、**[系統暫存器]**、**[索引暫存器]** 來控制通知位元項目。使用者也可在 **[一般屬性]** 頁中設定位址。

[寫入前] / [寫入後]

在寫入動作前 / 後設定所指定暫存器的狀態。

監看位址

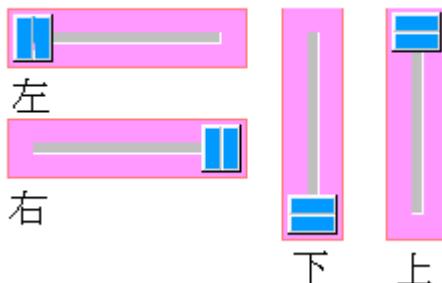
在滑塊被拖曳時，可以即時顯示當前寫入位址的設定值。



屬性

【方向】

滑動開關物件可以四個方向來顯示(朝右顯示，朝上顯示，朝左顯示，朝下顯示)



【最小刻度】

依照所填入之最小刻度值來顯示。例如：設【最小刻度】為 10，數值顯示為每一次都是依據 10 的刻度來跳動。

a. 常數

可直接設定字元暫存器的上下限常數值。例如：設【下限】為 5 和【上限】為 100，則設定的數值範圍為 5 ~ 100。

b. 位址

上下限可由指定暫存器設定。當寫入位址為 LW-n，則上/下限會根據以下的規則自動被設定為：

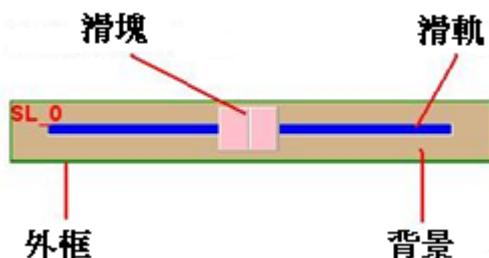
位址格式	16-bit	32-bit
位址	LW-n	LW-n
下限	LW-n	LW-n
上限	LW-n+1	LW-n+2

以下表為例，當【位址】為 LW-100 時，則上/下限的位址會自動被設定為：

位址格式	16-bit	32-bit
位址	LW-100	LW-100
下限	LW-100	LW-100
上限	LW-101	LW-102

【捲動模式】

不同於【最小刻度】拖拉滑動開關改變數值，只要輕觸一下【滑動開關】物件時，數值會被遞增/遞減，增減的多寡會根據【捲動值】的設定。



滑塊

共有四種滑塊樣式可供選擇，也可調整滑塊寬度。

顏色

用來選擇滑塊物件的外框，背景和滑軌顏色。

13.9 數值輸入與數值顯示

概要

[數值輸入] 與 [數值顯示] 物件皆可以用來顯示所指定字元暫存器內的數值，其中 [數值輸入] 物件可用鍵盤來輸入數值，更改暫存器內的數據。

設定



按下工作列上的 [數值輸入 / 顯示] 按鈕後即會開啟 [數值輸入 / 顯示] 物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個 [數值輸入 / 顯示] 物件。[數值輸入] 物件比 [數值顯示] 物件多了 [通知] 與 [鍵盤的設置]。



讀取/寫入使用不同的位址

數值輸入物件提供 [讀取/寫入使用不同的位址] 的選項，使用者可以分開設定數據的讀取位址與寫入位址。

讀取位址

點選 [設定] 後選擇字元暫存器設備類型的 **[PLC 名稱]**、**[位址]**、**[設備類型]**、**[系統暫存器]**、**[索引暫存器]** 來顯示數值。使用者也可在 **[一般屬性]** 頁中設定位址。

寫入位址

選擇字元相關的 **[PLC 名稱]**、**[設備類型]**、**[位址]** 作為數值寫入目標。

通知

使用此項設定，則在完成動作之前 / 之後可以連帶設定此項目所指定暫存器的狀態，使用[開]與[關]選擇要設定的狀態。

點選 [設定] 後選擇位元暫存器設備類型的 **[PLC 名稱]**、**[位址]**、**[設備類型]**、**[系統暫存器]**、**[索引暫存器]** 來控制通知位元項目。使用者也可在 **[一般屬性]** 頁中設定位址。

[寫入前]/[寫入後]

在寫入動作前 / 後設定所指定暫存器的狀態。

輸入無效通知

當輸入無效的數值時，可以自動設定所指定位址的狀態。



[模式]**a. 觸控**

使用者藉由觸碰物件來啟動輸入程序。

b. 位元控制

藉由指定位元暫存器的開或關來啟動及結束輸入程序。

[允許輸入位元位址]

指定控制輸入啟動及結束的位元暫存器位址。輸入順序必須遵照 [輸入次序] 的設定，輸入時須搭配外接 USB 鍵盤，不可使用螢幕觸控鍵盤。

輸入次序

設定輸入次序及輸入次序群組達成多個輸入物件連續輸入。

[使用準則]

- a. 輸入次序範圍: 1 ~ 511。群組範圍: 1 ~ 15。
- b. 若無勾選 [群組] 時，輸入次序群組為 0。
- c. 系統只尋找同一個輸入次序群組中的輸入物件。
- d. 愈小的輸入次序數值代表輸入順序排在愈前面，反之則愈後。
- e. 若兩輸入物件有同樣的輸入次序群組及輸入次序，則較下層的輸入物件將先輸入。

鍵盤**勾選[使用彈出鍵盤]**

指定鍵盤視窗及彈出位置。當啟動輸入時，系統將在指定位置彈出鍵盤視窗，並於結束輸入時關閉。

不勾選[使用彈出鍵盤]

啟動輸入時系統將不會彈出鍵盤視窗，使用者必須以下列方法進行輸入動作:

- a. 自行在視窗中設計鍵盤。
- b. 使用外接鍵盤。

隱藏視窗控制條

數值 / 字元輸入鍵盤可選擇不使用視窗控制條。

若輸入數值超出範圍時重新啟動鍵盤

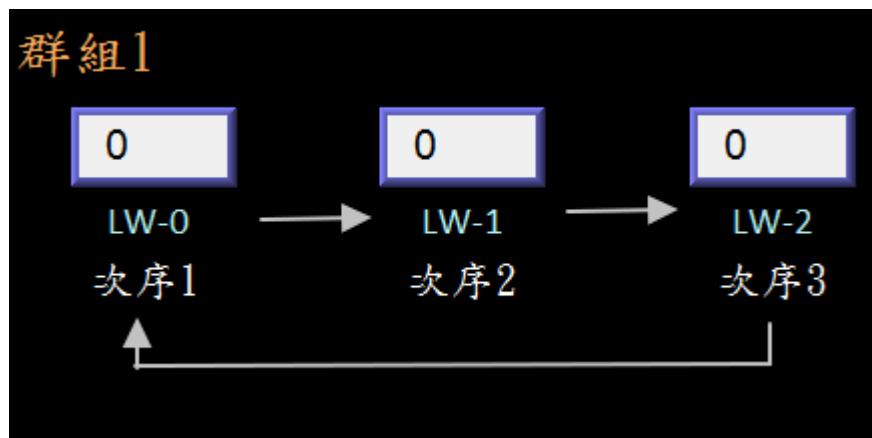
當使用輸入物件時，若輸入的數值超出設定範圍時系統將自動重新啟動鍵盤。

Example 1 設計群組式的數值輸入物件

藉由設定輸入次序及輸入次序群組達成多個輸入物件連續輸入。當完成目前物件輸入動作後，系統將自動跳至下一個同一群組的輸入物件繼續輸入。

- 建立三個數值輸入物件，皆使用【輸入次序】，次序分別為次序 1、次序 2、及次序 3，並設定為【群組一】。則輸入順序如下圖：

位址	設定
LW - 0	輸入次序 <input checked="" type="checkbox"/> 啟用 <input type="checkbox"/> 輸入完成後不再按次序輸入 輸入次序：1 <input type="button" value="↑"/> <input checked="" type="checkbox"/> 群組 1 <input type="button" value="↑"/>
LW - 1	輸入次序 <input checked="" type="checkbox"/> 啟用 <input type="checkbox"/> 輸入完成後不再按次序輸入 輸入次序：2 <input type="button" value="↑"/> <input checked="" type="checkbox"/> 群組 1 <input type="button" value="↑"/>
LW - 2	輸入次序 <input checked="" type="checkbox"/> 啟用 <input type="checkbox"/> 輸入完成後不再按次序輸入 輸入次序：3 <input type="button" value="↑"/> <input checked="" type="checkbox"/> 群組 1 <input type="button" value="↑"/>



- 若在輸入完最後一個數值輸入物件後，即停止跳出鍵盤結束所有的輸入動作，則勾選【輸入完成後不再按次序輸入】即可。

輸入次序	
<input checked="" type="checkbox"/> 啟用	
<input checked="" type="checkbox"/> 輸入完成後不再按次序輸入	
輸入次序：3 <input type="button" value="↑"/>	<input checked="" type="checkbox"/> 群組 1 <input type="button" value="↑"/>

下圖為【數值輸入】與【數值顯示】物件皆包含的【數字格式】設定頁，用來設定數值顯示的方式。



顯示格式

【資料格式】

選擇讀取位址字元暫存器內數據的形式，可選擇項目如下表。使用 16-bit 格式時，會佔用暫存器一個字元，使用 32-bit 格式時，則是佔用兩個字元。

格式
16-bit BCD
32-bit BCD
16-bit Hex
32-bit Hex
16-bit Binary
32-bit Binary
16-bit Unsigned
16-bit Signed
32-bit Unsigned
32-bit Signed
32-bit Float

【密碼】

數值顯示時將使用 “*” 號代替所有數字。

數字位元數

【小數點前位數】

小數點前的顯示位數。

【小數點後位數】

小數點後的顯示位數。

比例轉換

【使用比例轉換】

所顯示的數據是利用暫存器中的原始數據經過換算後所獲得。選擇此項功能必須設定 [比例最小值]，[比例最大值] 與 [限制] 項目中的 [輸入下限]、[輸入上限]。

假設原始數據使用 A 來表示，所顯示的數據使用 B 來表示，則數據 B 可以使用下列的換算公式獲得：

$$B = [\text{比例最小值}] + (A - [\text{輸入下限}]) * \text{ratio}$$

其中 $\text{ratio} = ([\text{比例最大值}] - [\text{比例最小值}]) / ([\text{輸入上限}] - [\text{輸入下限}])$

以下圖的設定為例，當原始數據是 15 時，則經過換算得到的數值為 $10 + (15 - 0) * (50 - 10) / (20 - 0) = 40$ ，數值物件上將顯示 40。

比例轉換

使用比例轉換

比例最小值： 比例最大值：

<= 預覽轉換結果

限制

輸入常數 取自暫存器

PLC 下限： PLC 上限：
 輸入下限： 輸入上限：

[測試]

利用測試功能檢查設定的轉換比例是否設定正確。輸入欲測試數值至 [PLC] 欄位，以下圖所示，輸入 15 為例，經過比例轉換後可得計算結果 40。

**限制**

用來設定輸入數值上下限的來源，並可設定警示顏色與警示效果。

[輸入常數]

選擇輸入數值的上下限分別來自 "輸入下限" (Input low) 與 "輸入上限" (Input high) 中的設定值。若輸入值不在上下限定義的範圍內，將無法更改暫存器內的數值。

[取自暫存器]

上下限可由指定暫存器設定。當寫入位址為 LW-n，則上/下限會根據以下的規則自動被設定為：

位址格式	16-bit	32-bit
暫存器位址	LW-n	LW-n
下限	LW-n	LW-n
上限	LW-n+1	LW-n+2

以下表為例，當 [暫存器位址] 為 LW-100 時，則上/下限的位址會自動被設定為：

位址格式	16-bit	32-bit
暫存器位址	LW-100	LW-100
下限	LW-100	LW-100
上限	LW-101	LW-102

[下限]

當暫存器內的數值小於下限值時，物件會使用此項顏色顯示數值。

[上限]

當暫存器內的數值大於上限值時，物件會使用此項顏色顯示數值。

[閃爍]

當暫存器內的數值小於下限值或大於上限值時，物件會使用閃爍的效果加以警示。

下圖為 [數值輸入] 與 [數值顯示] 物件的 [字型] 設定頁，用來設定數值顯示時所使用的字型、字型大小與顏色，另外也包括數字對齊的方式。



屬性

[顏色]

當數值在上下限的範圍內時，使用此項顏色顯示。

[對齊]

提供三種數字對齊方式：“左對齊” (left)、“前導零” (leading zero)、“右對齊” (right)，使用不同對齊方式的表現行為可參考右圖。

[尺寸]

設定字型大小

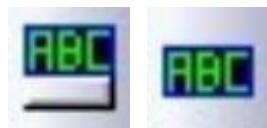
左對齊	12
前導零	0012
右對齊	12

13.10 字元輸入與字元顯示

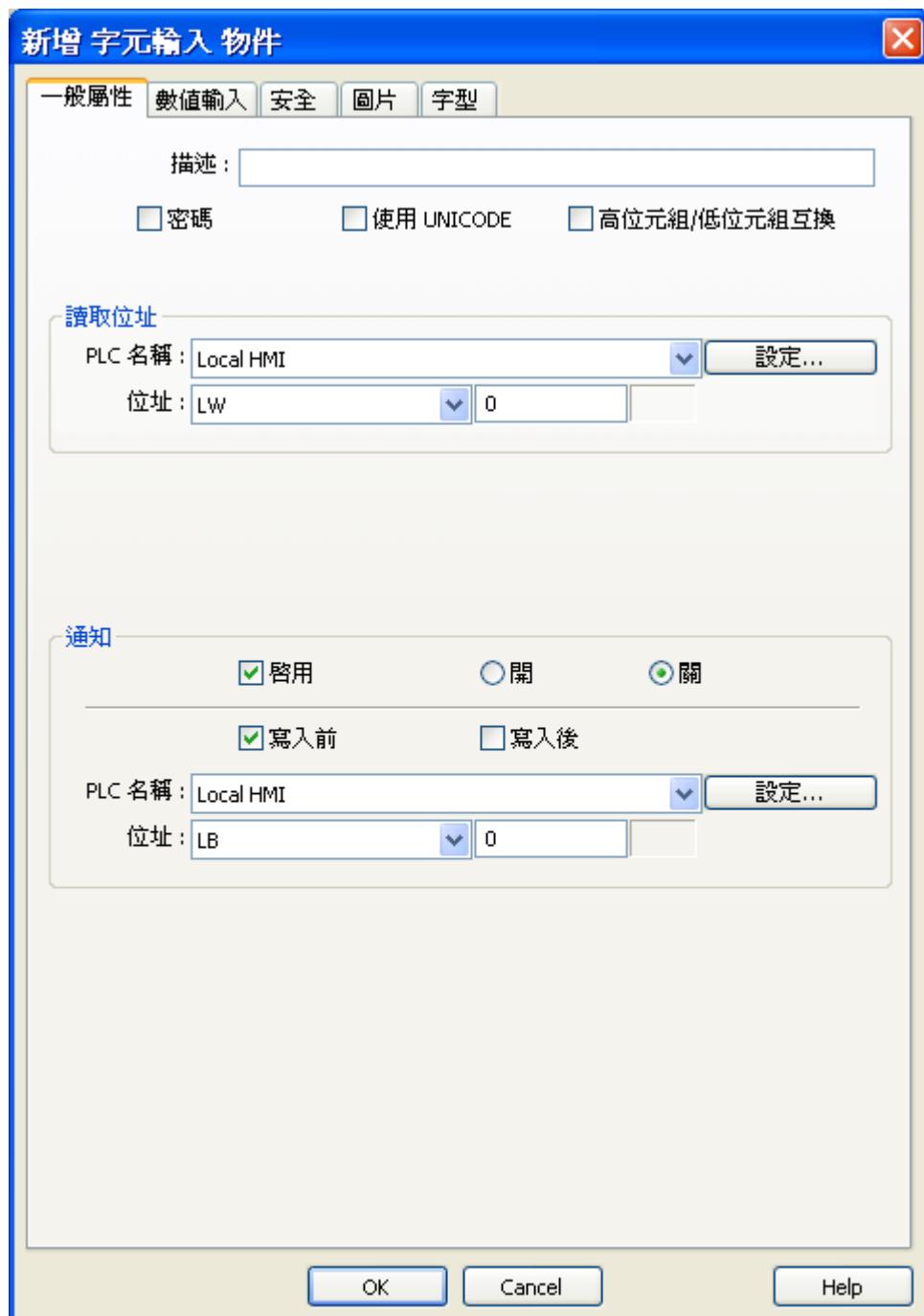
概要

[字元輸入] 與 [字元顯示] 物件使用 ASCII 編碼的方式顯示所指定暫存器中的數據，其中 [字元輸入] 物件可用鍵盤來輸入數值，更改暫存器內的數據。

設定



按下工作列上的 [字元輸入 / 顯示] 按鈕後即會開啟 [字元輸入 / 顯示] 物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個 [字元輸入 / 顯示] 物件。[字元輸入] 物件比 [字元顯示] 物件多了 [通知] 與 [鍵盤的設置]。



[密碼]

字元顯示時將使用“*”號代替所有字元。

[使用 UNICODE]

勾選 [使用 UNICODE] 可顯示 UNICODE 格式的資料。否則系統會顯示字元成 ASCII 格式。此功能可搭配功能鍵的 [ASCII/UNICODE]。

[高位元組/低位元組互換]

正常情況下，ASCII code 的顯示順序為 [高位元組] + [低位元組]。勾選此功能後，則顯示順序改為 [低位元組] + [高位元組]。

ABCD

BADC

高低位元不互換

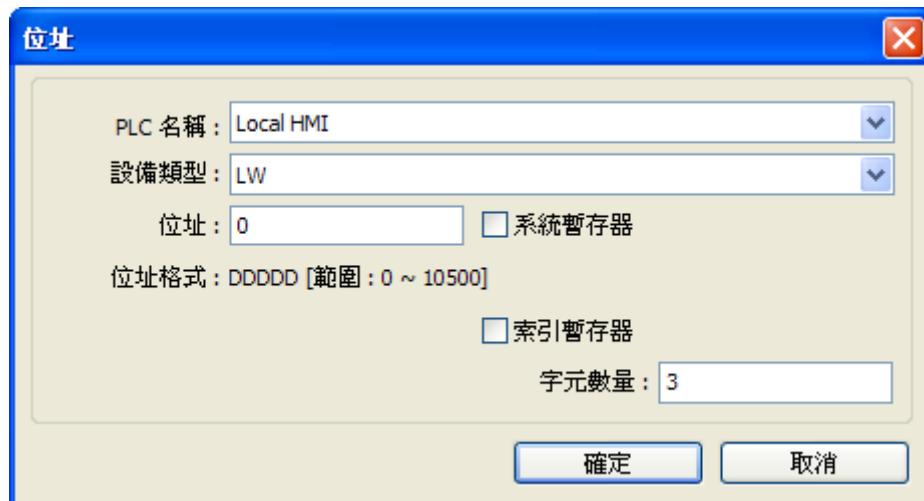
高低位元互換

讀取位址

點選 **【設定】** 後選擇字元暫存器設備類型的 **【PLC 名稱】**、**【位址】**、**【設備類型】**、**【系統暫存器】**、**【索引暫存器】** 來顯示字元。使用者也可在 **【一般屬性】** 頁中設定位址。

【字元數量】

點選 **【設定】** 後選擇文字最多可顯示的資料長度，單位為 word。



 使用 UNICODE 時，一個 UNICODE 文字等於一個字元 (word)；而使用 ASCII 時，一個 ASCII 文字等於一個位元組 (byte)，所以一個字元 (word) 可有兩個 ASCII 文字。(1 個字元 (word) 等於 2 個位元組 (byte))



屬性

下面圖片顯示字元輸入和字元顯示物件的 [字型] 設定頁面。使用者可設定文字顯示時所使用的字型、字型大小與顏色，另外也包括文字對齊的方式。

[對齊]

提供兩種文字對齊方式：“左對齊” (left)、“右對齊” (right)，使用不同對齊方式的表現行為可參考下圖。

[尺寸]

設定字型大小



13.11 間接視窗

概要

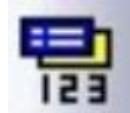
欲使用字元暫存器控制彈出的視窗編號時，可採用【間接視窗】物件。在間接視窗物件上設定控制的暫存器位址，於此位址寫入一數據，便會彈出與數據對應的視窗。

間接視窗物件有兩種使用方式：

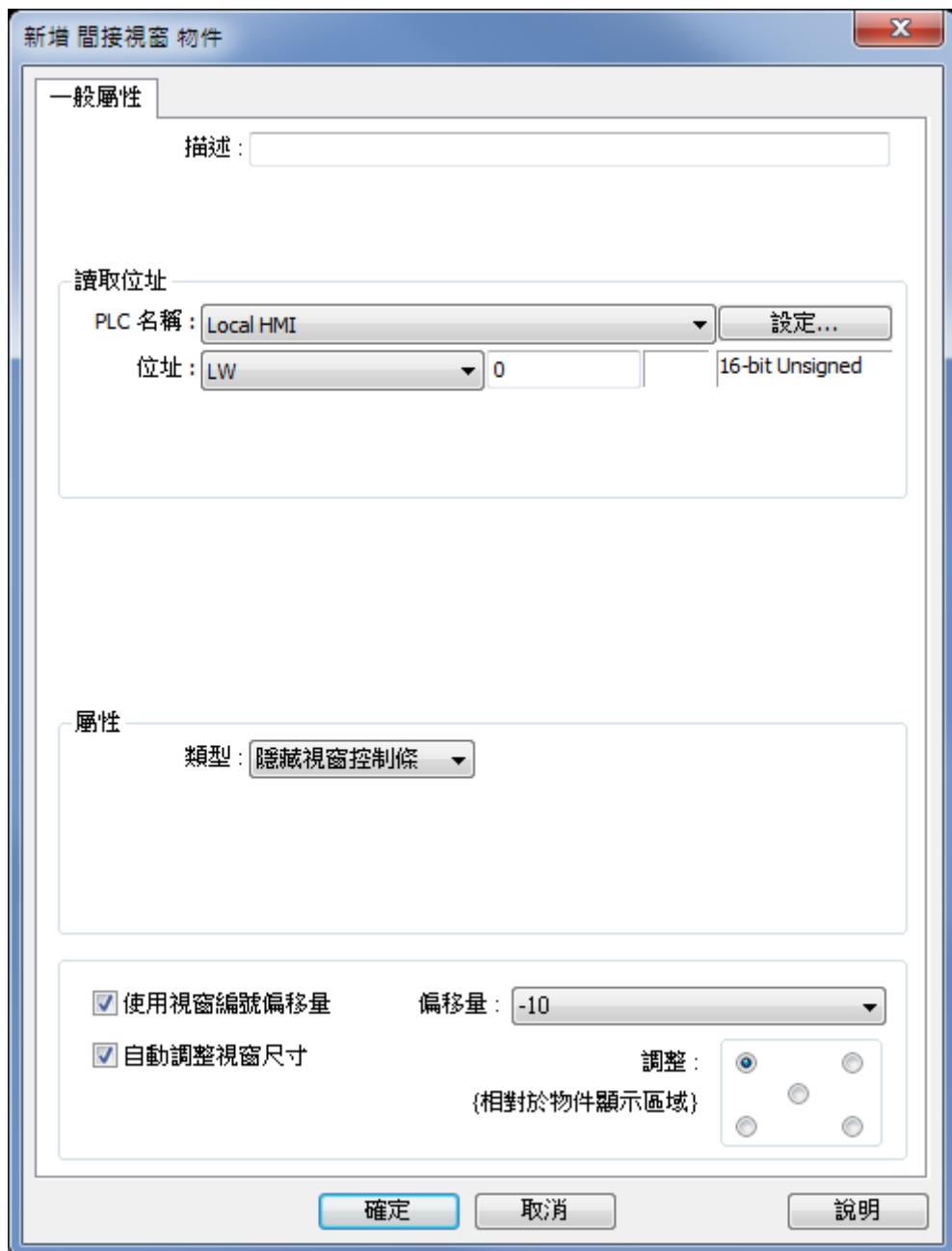
- 1.事先定義間接視窗的顯示區域，因此彈出的視窗都會有相同的尺寸大小。
- 2.間接視窗自動依據欲彈出的視窗尺寸調整其尺寸。

【直接視窗】與【間接視窗】的分別為直接視窗是由位元狀態來控制，而間接視窗則是由字元數值做控制。

設定



按下工作列上的【間接視窗】按鈕後即會開啟【間接視窗】物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個【間接視窗】物件。



讀取位址

點選【設定】後選擇字元暫存器設備類型的 **[PLC 名稱]**、**[位址]**、**[設備類型]**、**[系統暫存器]**、**[索引暫存器]** 來控制視窗彈出。使用者也可在 **[一般屬性]** 頁中設定位址。

屬性

[類型]

設定彈出視窗樣式。支援兩種樣式，**[隱藏視窗控制條]** 和 **[顯示視窗控制條]**。

a. 隱藏視窗控制條

彈出的子視窗不包含視窗控制條，它的視窗位置被固定住在預設位置無法拖曳。

**b. 顯示視窗控制條**

彈出的子視窗包含視窗控制條，它的視窗位置可藉由控制條任意被拖曳。

**[使用視窗編號偏移量]**

設定視窗的偏移量。彈出的視窗編號 = 讀取的字元數值 + 偏移量。偏移量的範圍: -1999 至 1999，不包含 0。

[自動調整視窗尺寸]

間接視窗自動調整成彈出視窗的大小，無須再特別規劃物件的尺寸。

[調整]

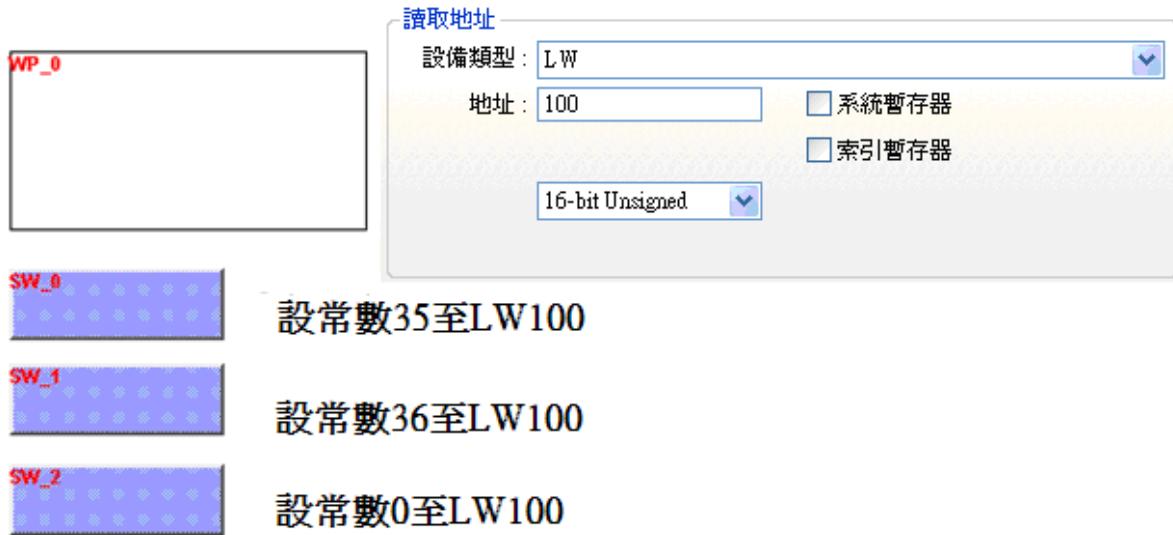
當勾選 [自動調整視窗尺寸] 後，系統會根據間接視窗的尺寸產生五個基準點。選擇一點後，彈出視窗就會從此間接視窗的基準點顯示。

範例：設定彈出視窗以間接視窗的左上角為基準點，如下圖。

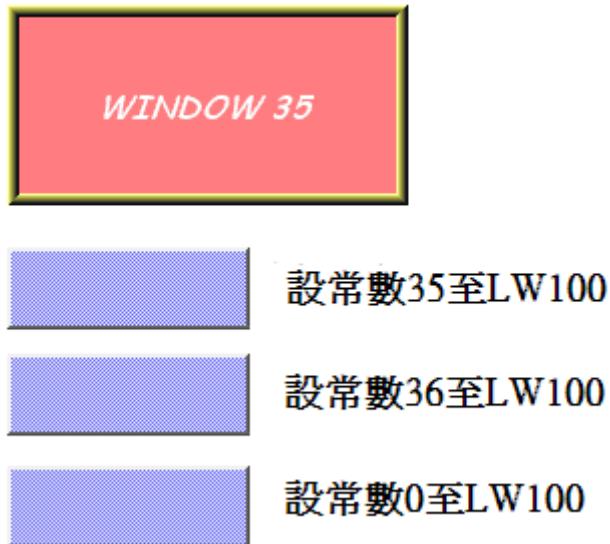


注意：使用此功能，設計者應注意彈出的視窗大小及方向，以免因視窗覆蓋原本頁面的重要物件，或彈出視窗超過主視窗範圍。

現在使用一個簡單的例子說明間接視窗的使用方式，下圖為間接視窗物件的設定內容，此時使用 LW-100 用來指定要出現的視窗號碼，並預先建立 ”視窗 35” 與 ”視窗 36”。



可以使用 [多狀態設定] 物件，將 [LW-100] 設定為寫入數值 35，此時視窗顯示的畫面如下。



如果繼續使用 [多狀態設定] 物件，將 [LW-100] 設定為寫入數值 36，則會關閉 ”視窗 35”，並且彈跳出 ”視窗 36”，參考下圖。



設常數35至LW100



設常數36至LW100



設常數0至LW100

要關閉 ”視窗 35” 或 ”視窗 36” 除了可以使用 [多狀態設定] 物件設定為寫入數值 0 之外，另一種方式是在 ”視窗 35” 與 ”視窗 36” 上設計一個功能鍵物件，並使用 [關閉視窗] 模式，在按下此物件後即可關閉這些視窗。



在程式運作時最多可同時顯示 16 個視窗。系統不允許在一個基本視窗上使用 2 個直接 (或間接) 視窗彈出同一個視窗。

13.12 直接視窗

概要

[直接視窗] 物件可以在視窗上定義一個顯示區域，當所指定暫存器的狀態改變時，將在此顯示區域內顯示此視窗的內容。所顯示視窗的長度與高度不會大於此顯示區域。要關閉此時所顯示的視窗，只需將用來觸發視窗出現的暫存器的狀態恢復即可。

[直接視窗] 與 **[間接視窗]** 物件的差別在於 **[直接視窗]** 已經事先設定好要顯示的視窗編號，系統運作時，將利用所指定暫存器的狀態決定是否顯示或關閉此視窗。

簡單來說，**[直接視窗]** 與 **[間接視窗]** 的分別為直接視窗是由位元狀態來控制，而間接視窗則是由字元數值做控制。

設定



按下工作列上的**[直接視窗]**按鈕後即會開啟**[直接視窗]**物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個**[直接視窗]**物件。



讀取位址

點選 **[設定]** 後選擇位元暫存器設備類型的**[PLC 名稱]**、**[位址]**、**[設備類型]**、**[系統暫存器]**、**[索引暫存器]** 來控制視窗彈出。使用者也可在 **[一般屬性]** 頁中設定位址。

屬性

[類型]

設定彈出視窗樣式。支援兩種樣式，**[隱藏視窗控制條]** 和 **[顯示視窗控制條]**。

[視窗序號]

設定要彈出的視窗號碼。

現在使用一個簡單的例子說明 **[直接視窗]** 的使用方式，下圖為 **[直接視窗]** 物件的設定內容，此時使用 **LB-10** 來決定是否顯示”視窗 35”。

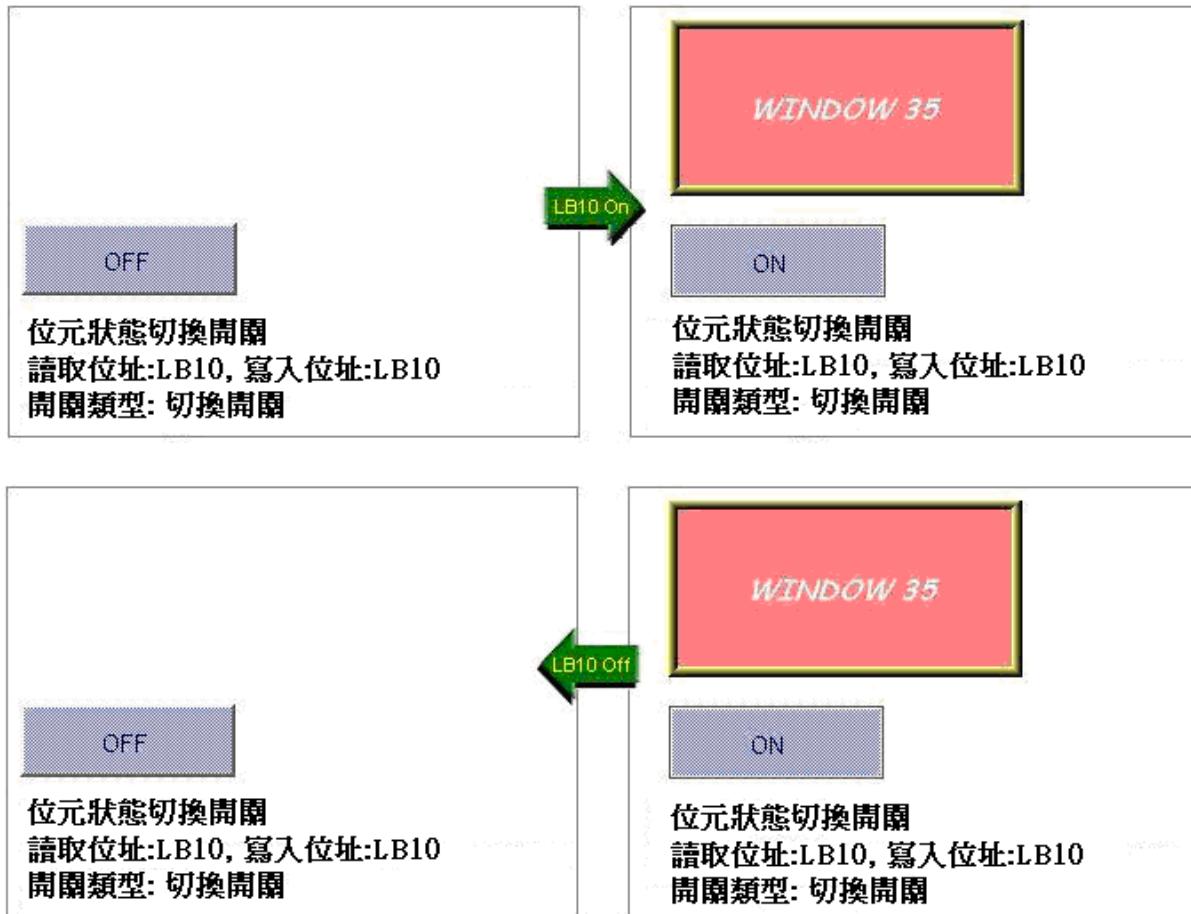


位元狀態切換開關

讀取位址:LB10, 寫入位址:LB10

開關類型: 切換開關

當 LB-10 狀態為 ON 時，"視窗 35"將出現；當 LB-10 狀態為 OFF 時，"視窗 35"將消失。參考下圖。



最多可同時開啟 16 個彈出視窗，包含系統訊息視窗、直接視窗和間接視窗。
系統不允許在一個基本視窗上使用 2 個直接(或間接) 視窗彈出同一個視窗。

13.13 移動圖形

概要

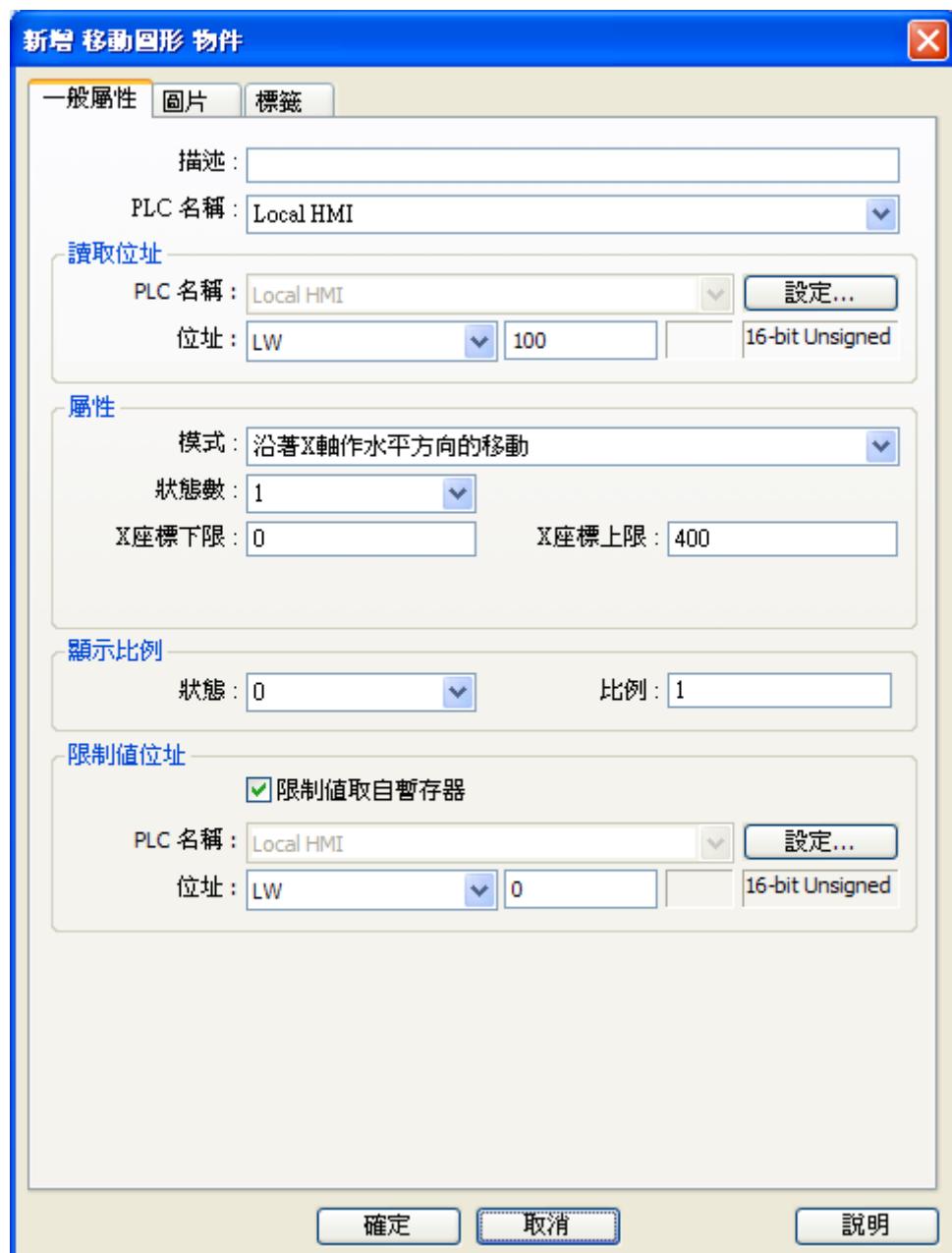
[移動圖形] 物件可定義物件的狀態和移動距離，物件會利用三個連續的暫存器內的數據，決定物件的狀態與物件的移動距離。第一個暫存器為控制物件的狀態，第二個暫存器為控制物件的水平位置移動距離(X)，第三個暫存器為控制物件的垂直位置移動距離(Y)。

資料格式	控制物件狀態位址	控制物件 X 軸移動距離位址	控制物件 Y 軸移動距離位址
16-bit 格式	位址 (LW-n)	位址+1 (LW-n+1)	位址+2 (LW-n+2)
32-bit 格式	位址 (LW-n)	位址+2 (LW-n+2)	位址+4 (LW-n+4)

設定



按下工作列上的 **[移動圖形]** 按鈕後即會開啟 **[移動圖形]** 物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個 **[移動圖形]** 物件。



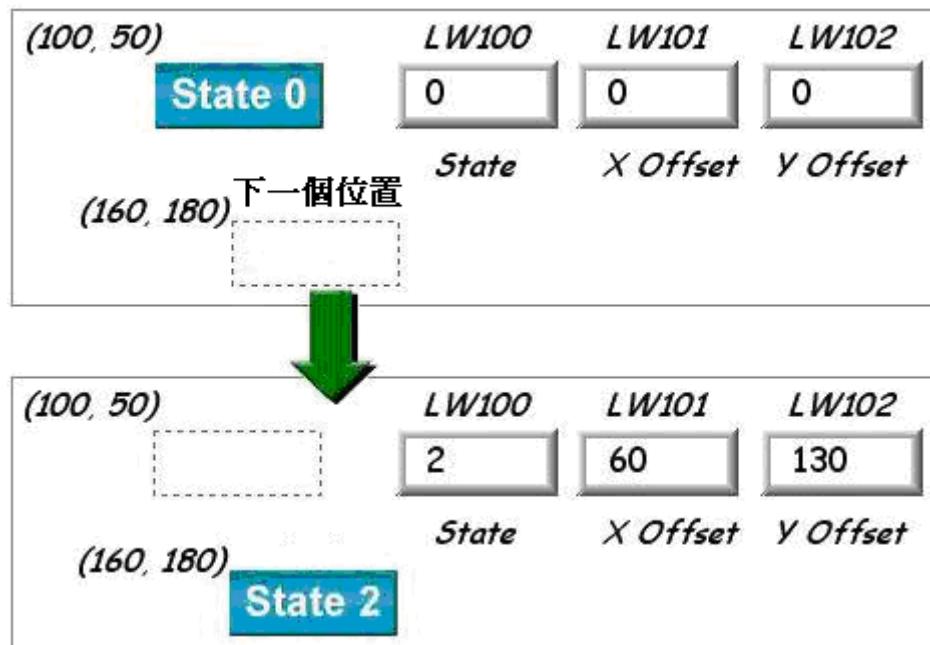
讀取位址

點選【設定】後選擇字元暫存器設備類型的【PLC 名稱】、【位址】、【設備類型】、【系統暫存器】、【索引暫存器】來做為控制移動圖形狀態和移動位置的目標。使用者也可在【一般屬性】頁中設定位址。下列表顯示不同格式時，需使用的控制位址。

資料格式	控制物件狀態 位址	控制物件 X 軸移動 距離位址	控制物件 Y 軸移動 距離位址
16-bit 格式	位址 (LW-n)	位址+1 (LW-n+1)	位址+2 (LW-n+2)
32-bit 格式	位址 (LW-n)	位址+2 (LW-n+2)	位址+4 (LW-n+4)

舉例來說，若暫存器為 LW-100，且變數型態使用[16-bit Unsigned]，則 LW-100 存放物件的狀態，LW-101 存放 X 軸方向的移動距離，LW-102 存放 Y 軸方向的移動距離。

以下圖為例，物件的位址為 LW-100 且起始位址為(100, 50)，假使現在要移動物件至(160, 180)且顯示狀態 2 的圖形，則 LW-100 需設定為 2， $LW101 = 160 - 100 = 60$ ， $[LW102] = 180 - 50 = 130$



屬性

選擇物件的移動方式、移動的範圍。

a. 沿著 X 軸作水平方向的移動

只允許物件沿著 X 軸作水平方向的移動。移動範圍由 [X 軸座標下限] 與 [X 軸座標上限] 來決定。

屬性

模式 :	沿著X軸作水平方向的移動
狀態數 :	8
X座標下限 :	0
X座標上限 :	600

資料格式	控制物件狀態位址	控制物件 X 軸移動距離位址
16-bit 格式	位址 (LW-n)	位址 + 1 (LW-n+1)
32-bit 格式	位址 (LW-n)	位址 + 2 (LW-n+2)

b. 沿著 Y 軸作垂直方向的移動

只允許物件沿著 Y 軸作垂直方向的移動。移動範圍由 [Y 軸座標下限] 與 [Y 軸座標上限] 來決定。

屬性

模式 :	沿著Y軸作垂直方向的移動
狀態數 :	8
Y座標下限 :	0
Y座標上限 :	480

資料格式	控制物件狀態位址	控制物件 Y 軸移動距離位址
16-bit 格式	位址 (LW-n)	位址 + 1 (LW-n+1)
32-bit 格式	位址 (LW-n)	位址 + 2 (LW-n+2)

c. 可同時作 X 軸與 Y 軸方向的移動

允許物件沿著 X 軸與 Y 軸移動。移動範圍由 [X 軸座標下限]、[X 軸座標上限] 與 [Y 軸座標下限]、[Y 軸座標上限] 來決定。

屬性

模式 :	可同時作X方向與Y方向的移動
狀態數 :	8
X座標下限 :	0
X座標上限 :	600
Y座標下限 :	0
Y座標上限 :	480

資料格式	控制物件狀態位址	控制物件 X 軸移動距離位址	控制物件 Y 軸移動距離位址
16-bit 格式	位址 (LW-n)	位址+1 (LW-n+1)	位址+2 (LW-n+2)
32-bit 格式	位址 (LW-n)	位址+2 (LW-n+2)	位址+4 (LW-n+4)

d. 沿著 X 軸按比例作水平方向的移動

只允許物件沿著 X 軸、按比例作水平方向的移動。假設暫存器中與 X 軸位移有關的數據為 data，則 X 軸的位移量可以使用下面的公式：

$$\text{X 軸位移} = (\text{data} - [\text{輸入下限}]) * ([\text{比例上限}-\text{比例下限}]) / ([\text{輸入上限}] - [\text{輸入下限}])$$



例如物件只允許做 200~500 大小的位移，但暫存器數據的大小範圍為 300~1000，此時可以將[輸入下限]設定為 300，[輸入上限]設定為 1000，[比例下限]設定為 200，[比例上限]設定為 500，物件即會在要求的範圍內移動。

資料格式	控制物件狀態位址	控制物件 X 軸移動距離位址
16-bit 格式	位址 (LW-n)	位址+1 (LW-n+1)
32-bit 格式	位址 (LW-n)	位址+2 (LW-n+2)

e. 沿著 Y 軸按比例作垂直方向的移動

只允許物件沿著 Y 軸按比例作垂直方向的移動，Y 軸位移量的換算公式與“沿著 X 軸按比例作水平方向的移動”相同。

資料格式	控制物件狀態位址	控制物件 Y 軸移動距離位址
16-bit 格式	位址 (LW-n)	位址+1 (LW-n+1)
32-bit 格式	位址 (LW-n)	位址+2 (LW-n+2)

f. 沿著 X 軸按反比例作水平方向的移動

此項功能與【沿著 X 軸按比例作水平方向的移動】相同，但移動方向相反。

g. 沿著 Y 軸按反比例作水平方向的移動

此項功能與【沿著 Y 軸按比例作垂直方向的移動】相同，但移動方向相反。

顯示比例

物件各個狀態的圖形在顯示時，可以分開設定縮放比例，參考下圖



限制值位址

物件的顯示區域除了可以直接設定[X 軸座標下限]、[X 軸座標上限]與[Y 軸座標下限]、[Y 軸座標上限]來決定外，也可以利用暫存器中的數據來決定。假設顯示區域由位址內的數據來決定，[X 軸座標下限]、[X 軸座標上限]與[Y 軸座標下限]、[Y 軸座標上限]的讀取位址可參考下表。

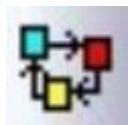
資料格式	[X 軸座標 下限] 位址	[X 軸座標上 限] 位址	[Y 軸座標下 限] 位址	[Y 軸座標上 限] 位址
16-bit 格式	位址	位址+ 1	位址+ 2	位址+ 3
32-bit 格式	位址	位址+ 2	位址+ 4	位址+ 6

13.14 動畫

概要

使用者可以預先定義【動畫】物件的移動軌跡，並利用更改暫存器內的數據，控制物件的狀態與物件在移動軌跡上的位置。系統將使用連續兩個暫存器內的數據來控制動畫物件，第一個暫存器為控制物件的狀態，第二個為控制物件的位置。

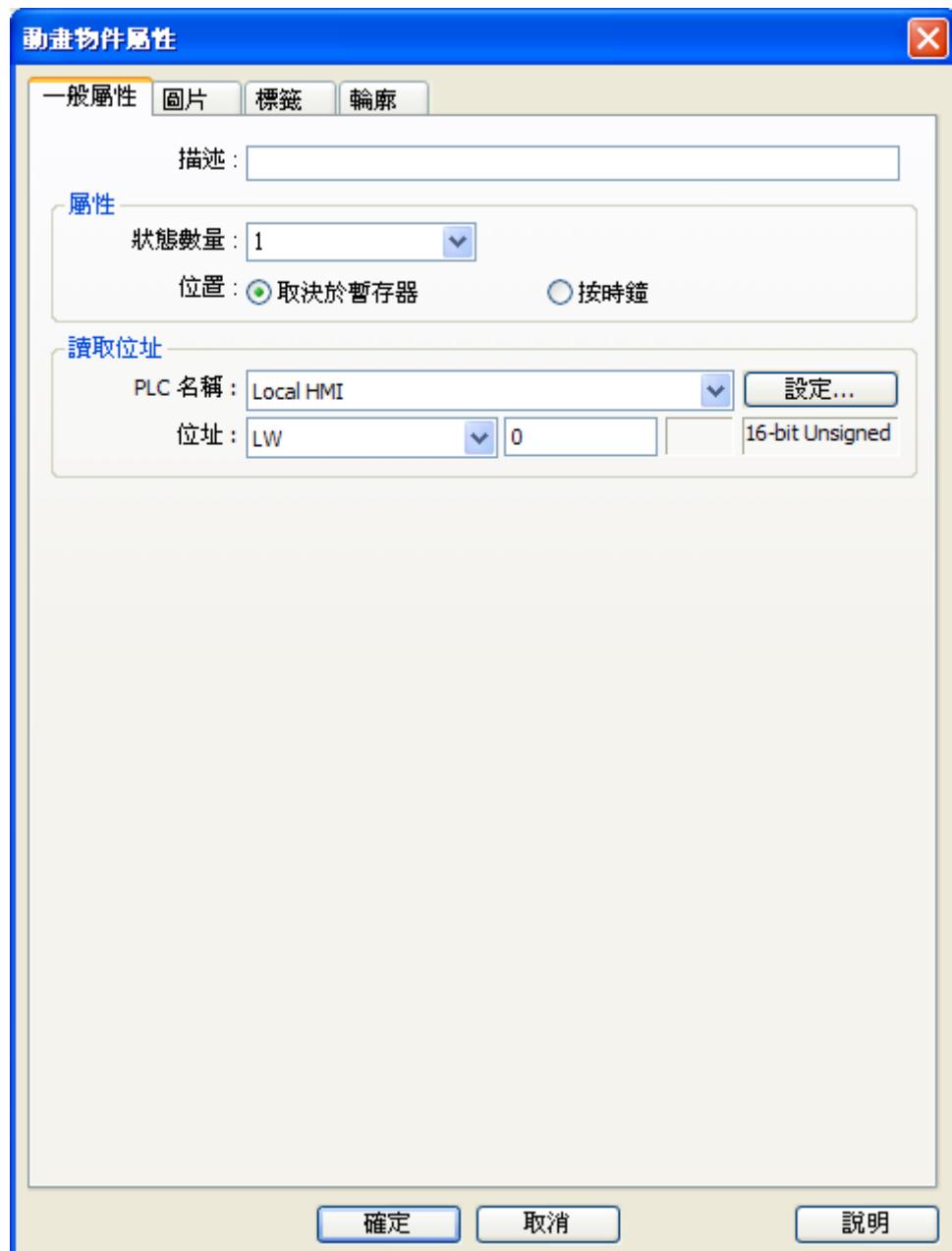
設定



按下工作列上的【動畫】按鈕。首先，設定物件的移動路徑。使用滑鼠在編輯畫面上點擊左鍵一一指定每個移動位置，然後點擊右鍵即會開啟【動畫】物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個【動畫】物件。



要更改物件的屬性，可以使用滑鼠左鍵雙擊物件所在位置，利用出現的【動畫】物件屬性對話窗，即可更改物件的各項屬性。



屬性

[狀態數量]

設定物件狀態數目。

a. 取決於暫存器

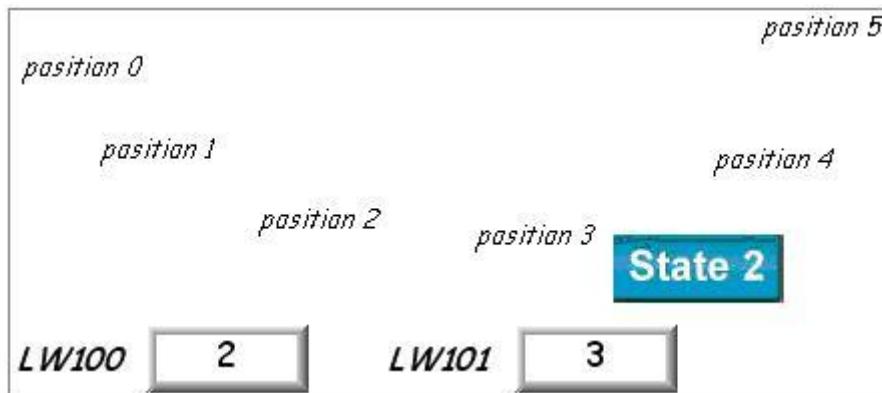
當選擇 [取決於暫存器] 時，由暫存器數據控制物件的狀態和位置。

讀取位址

如果物件的狀態與位置由暫存器中的數據決定，就必須正確設定物件狀態與位置的讀取位址。讀取位址格式如下表。

資料格式	控制物件狀態位址	控制物件位置位址
16-bit 格式	位址 (LW-n)	位址+1 (LW-n+1)
32-bit 格式	位址 (LW-n)	位址+2 (LW-n+2)

舉例來說，若暫存器為 LW-100，且格式使用[16-bit Unsigned]，則 LW-100 存放物件的狀態，LW-101 存放物件的顯示位置。以下圖為例，LW-100 = 2，LW-101 = 3，所以物件顯示狀態 2，並出現在位置 3。



b. 按時鐘

若選擇 [按時鐘] 的變化，則物件將自動改變狀態與顯示位置，[自動控制位置] 項目用來設定狀態與顯示位置改變方式。



[速度]

位置改變速度，單位為 0.1 秒。例如設定為 10，則物件每隔 1 秒鐘變換一個位置。

[返回]

假設物件有 4 個位置，分別為 position 0、position 1、position 2、position 3。若未選擇此項設定，當移動到最後一個位置(position 3)後，物件將移動到初始位置 position 0，再重複原來位置改變方式，移動位置整理順序如下。

position 0 → position 1 → position 2 → position 3 → position 0 → position 1 → position 2...

若選擇此項設定，當移動到最後一個位置(position 3)後，物件將使用反向的移動方式，移動到初始位置 position 0，再重複原來位置改變方式，移動位置整理順序如下。

position 0 → position 1→ position 2→ position 3→ position 2→ position 1→ position 0...

【狀態轉換】

狀態改變的方式，可以選擇 [基於位置] 與 [基於時間]。選擇 [基於位置] 表示位置改變，狀態也隨著改變。若選擇 [基於時間]，表示位置使用固定的頻率自動變換，變換頻率在[轉換週期]中設定。參考下圖。





向量圖尺寸

用來設定物件所顯示圖形的大小。

軌跡

來設定移動軌跡上各點的位置。



因一個動畫物件可設定多個不同的圖片，因此無法使用 [使用原尺寸] 將所有的圖片還原成原始尺寸。

13.15 棒圖

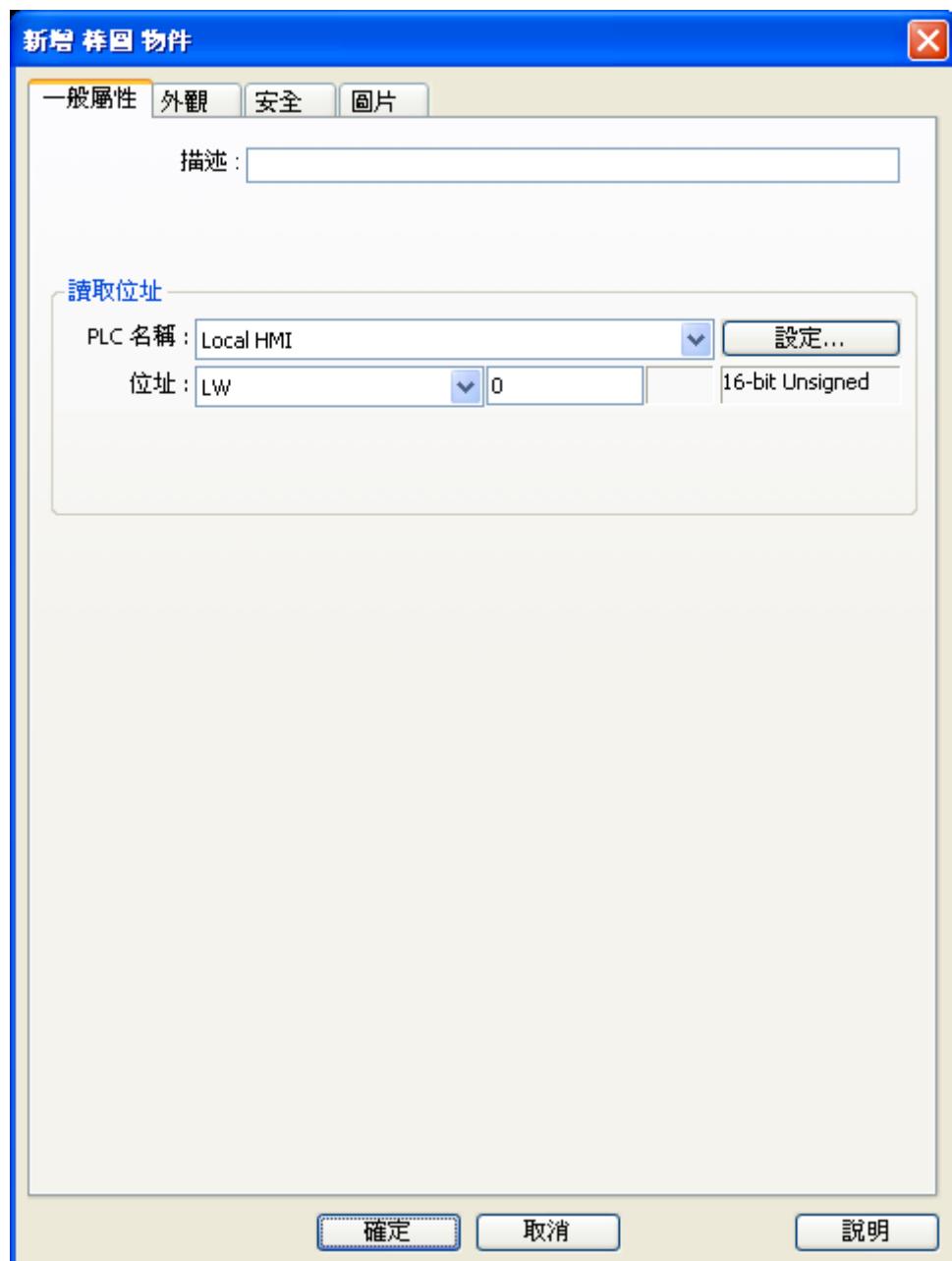
概要

[棒圖] 物件使用百分比例與棒圖的方式，顯示暫存器中的數據。

設定



按下工作列上的 [棒圖] 按鈕後即會開啟 [棒圖] 物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個 [棒圖] 物件。



讀取位址項目

點選【設定】後選擇字元暫存器設備類型的**[PLC 名稱]**、**[位址]**、**[設備類型]**、**[系統暫存器]**、**[索引暫存器]**來做為棒圖顯示的數據依據。使用者也可在**【一般屬性】**頁中設定位址。



屬性

[型式]

可以選擇 [一般型] 與 [偏差型]。當選擇 [偏差型] 時，需設定原點位置，參考下圖。



[顯示方向]

用來選擇棒圖的顯示方向，可以選擇 [朝上顯示]、[朝下顯示]、[朝右顯示]、[朝左顯示]。

[最小值]、[最大值]

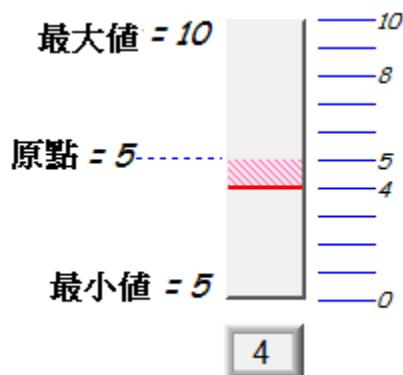
棒圖填充的百分比可以利用下列的公式換算而得：

$$\text{顯示區域百分比} = \frac{\text{暫存器數據} - \text{最小值}}{\text{最大值} - \text{最小值}} \times 100\%$$

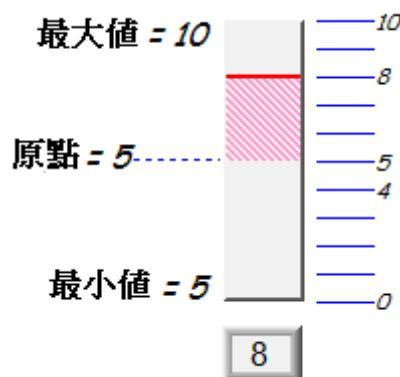
若(暫存器數據 - [最小值])大於 0，則棒圖將由[原點位置]的位置往上填充；若(暫存器數據 - [最小值])小於 0，則棒圖將由[原點位置]的位置往下填充。

範例，下圖顯示在[原點位置]設定為 5，[最大值]為 10，[最小值]為 0 並使用不同數據時，棒圖的填充情形。

當讀取數值為 4，

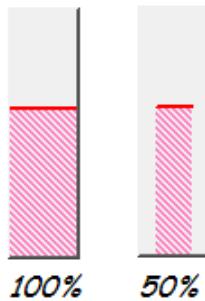


當讀取數值為 8，



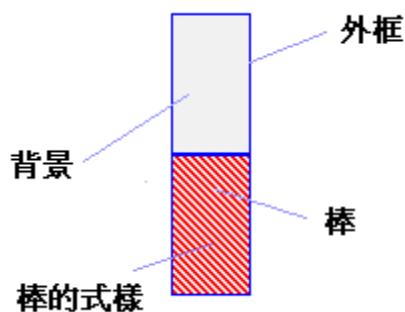
[棒寬度比例(%)]

設定棒圖的顯示寬度與物件寬度間的百分比率。以下圖示顯示不同比例，50% 以及 100%。



棒圖

來設定棒圖外框、背景顏色與填充區域的樣式與顏色，參考下圖。

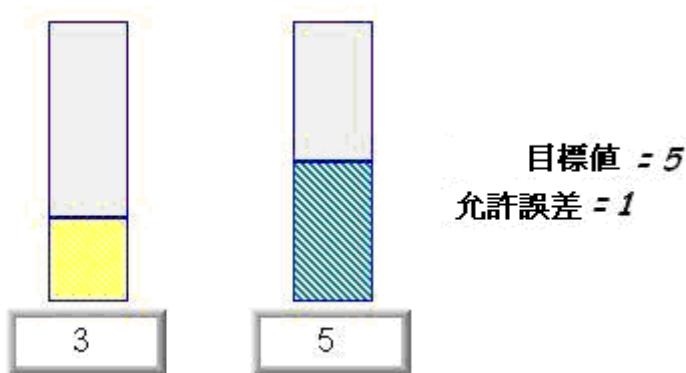


目標值

當暫存器內的數據符合下列的條件時，填充區域的顏色可以變更為此項目所定義的顏色。

$$[\text{目標值}] - [\text{允許誤差}] \leq \text{暫存器內的數據} \leq [\text{目標值}] + [\text{允許誤差}]$$

參考下圖，此時[目標值] = 5，[誤差值] = 1，則暫存器的值大於或等於 $5-1=4$ ，且小於或等於 $5+1=6$ ，填充區域的部分將改變為“目標值顏色”。

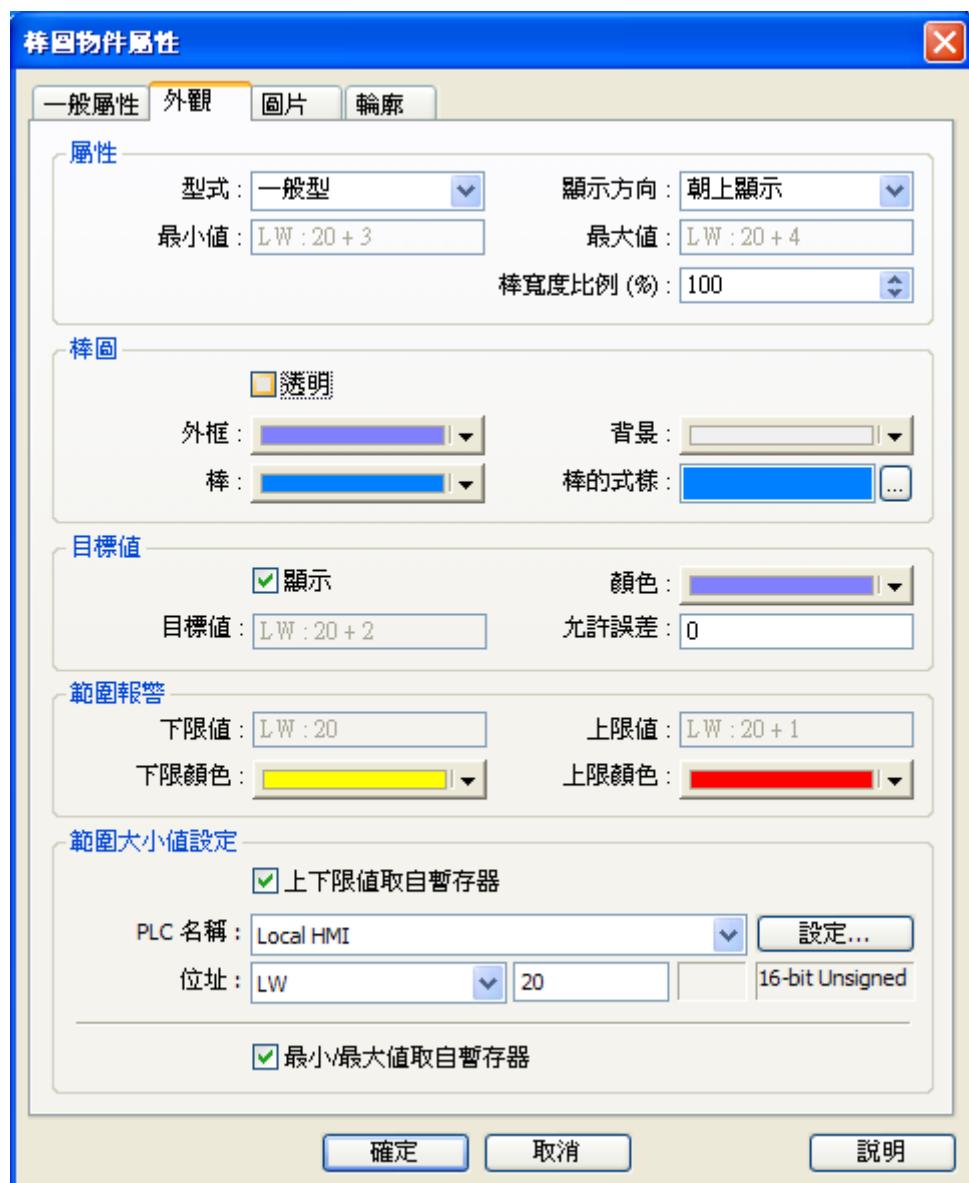


範圍警報

當數據大於 [上限值] 時，填充區域的顏色可以變更為 [上限顏色] 所定義的顏色；若當數據小於[下限值]時，填充區域的顏色可以變更為 [下限顏色] 所定義的顏色。

範圍上下限

當選擇[上下限值取自暫存器]，[範圍報警項目] 中所使用的 [下限值]、[上限值] 與 [目標值項目] 中的 [目標值] 皆讀取自指定的暫存器，參考下圖。



下表整理了當使用範圍取自暫存器，上下限與目標值的讀取位置，其中“位址”表示暫存器的位址值，例如暫存器為 LW-20 時，並使用 16-bit 資料格式時。

則下限值 LW-20 / 上限值 LW-21 / 目標值 LW-22 / 最小值 LW-23 / 最大值 LW-24

資料格式	下限值	上限值	目標值	最小值	最大值
16-bit 格式	位址	位址+1	位址+2	位址+3	位址+4
32-bit 格式	位址	位址+2	位址+4	位址+6	位址+8

13.16 錶針

概要

[錶針] 物件會使用儀表圖的方式，指示目前暫存器中的數據。

設定



按下工作列上的 **[錶針]** 按鈕後即會開啟 **[錶針]** 物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個 **[錶針]** 物件。

下面說明【錶針】物件屬性對話窗中各設定頁的內容。

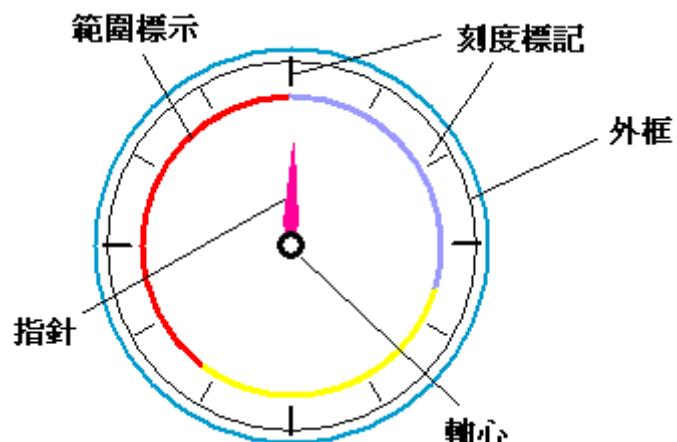


讀取位址項目

點選【設定】後選擇字元暫存器設備類型的**【PLC 名稱】**、**【位址】**、**【設備類型】**、**【系統暫存器】**、**【索引暫存器】**來做為錶針顯示的數據依據。使用者也可在**【一般屬性】**頁中設定位址。

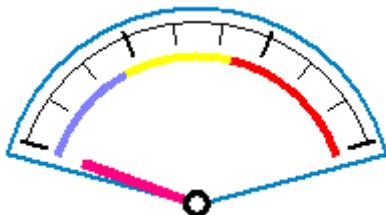


上圖的設定對話窗用來設定錶針物件的[外觀]，各部位的名稱可以參考下圖。

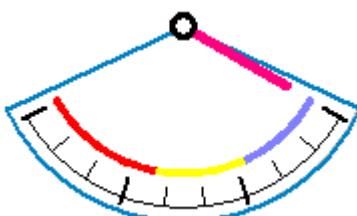


角度

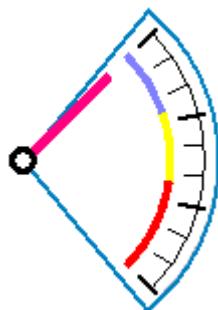
用來設定錶針物件的起始角度與結束角度，角度可設定範圍皆為 0~360 度。不同的設定值所顯示的結果，可參考下面的幾種不同的設定。



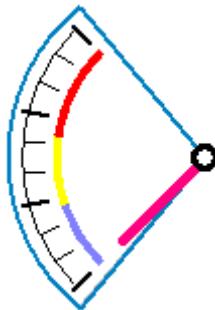
[起始角度] = 290 , [結束角度] = 70



[起始角度] = 120 , [結束角度] = 240



[起始角度] = 40 , [結束角度] = 140



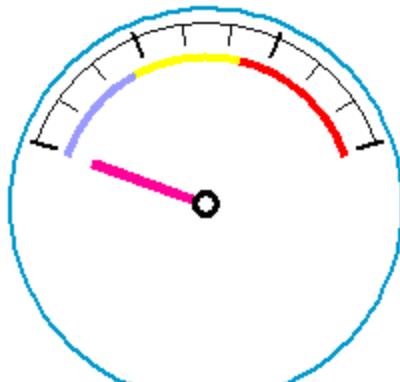
[起始角度] = 225 , [結束角度] = 315

背景

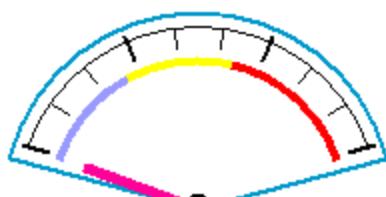
設定物件的背景與圓周的顏色。

[全圓]

當選擇 [全圓] 時，錶針物件將顯示整個圓形，反之則顯示被定義的角度範圍。



使用全圓



非使用全圓

[透明]

當選擇 [透明] 時，錶針物件將不會顯示背景與外框顏色。

刻度標記

設定標記的數量與顏色。

指針

設定指針的形狀，長寬度和顏色。

軸心

設定軸心的樣式與顏色。

下圖為[限制/標記]頁面設定對話窗



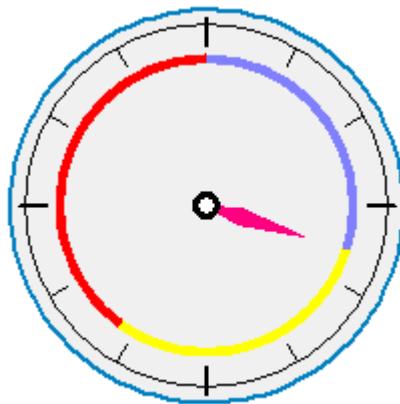
數值

設定物件所要顯示的數值範圍。[錶針] 物件會利用 [最小值] 與 [最大值] 的設定內容和由[讀取位址]所讀取的數值，換算指針的指示位置。舉例來說，假使[最小值] = 0，[最大值] = 100，若此時讀取的數據為 30，且[起始角度] = 0，[結束角度] = 360，則指針指示的角度為 (在[結束角度]大於[起始角度]的情形下)：

$$\{(30 - [\text{最小值}])/([\text{最大值}] - [\text{最小值}])\} * ([\text{結束角度}] - [\text{起始角度}]) =$$

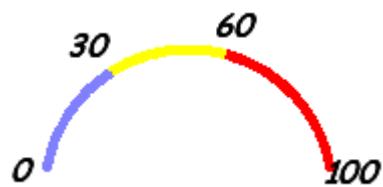
$$\{(30 - 0) / (100 - 0)\} * (360 - 0) = 108$$

指針將指示在 108 度的位置，參考下圖



範圍

設定高、低限值與高、低限標誌的顯示顏色與寬度



【使用者自定半徑】

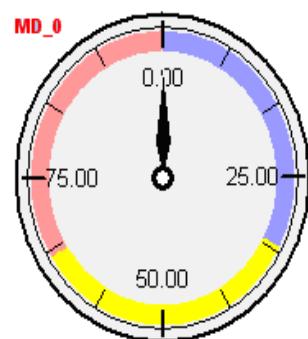
範圍

顯示數值範圍標示

下限: 範圍內: 上限:

寬度: 10

使用者自定半徑



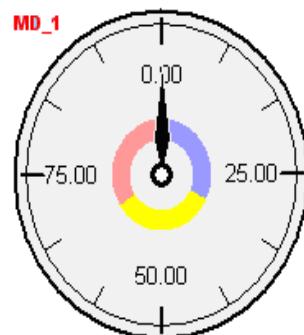
範圍

顯示數值範圍標示

下限: 範圍內: 上限:

寬度: 10

使用者自定半徑



【上下限取自暫存器】

上下限可由指定暫存器設定。當寫入位址為 LW-n，則上/下限會根據以下的規則自動被設定為：

位址格式	16-bit	32-bit
下限	LW-n	LW-n
上限	LW-n+1	LW-n+2

以下表為例，當[暫存器位址]為 LW-100 時，則上/下限的位址會自動被設定為：

位址格式	16-bit	32-bit
下限	LW-100	LW-100
上限	LW-101	LW-102

刻度符號

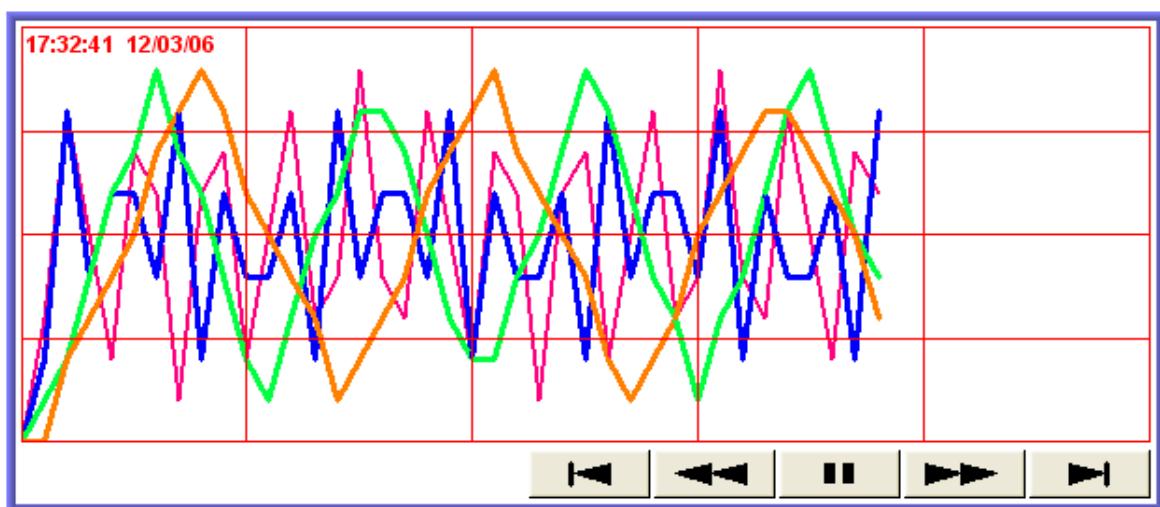
設定是否使用刻度符號於錶針上。



13.17 趨勢圖

概要

[趨勢圖] 物件會將設定在【資料取樣】中的資料利用連續的線段描繪出圖，使用者可藉由趨勢圖繪出的趨勢數據做分析。下圖為**【趨勢圖】**的使用情形。



其中各按鈕的功能描述如下：



按下後畫面將顯示最初的取樣資料，並關閉畫面自動捲動功能。

按下後畫面將顯示往前一個間隔的取樣資料，並關閉畫面自動捲動功能。

顯示此圖形表示目前已關閉畫面自動捲動功能，按下後將重新開啟此項功能。

按下後畫面將顯示往後一個間隔的取樣資料。

按下後畫面將顯示最新的取樣資料。

顯示此圖形表示目前畫面自動捲動功能已被開啟，按下後將關閉此項功能。

設定



按下工具列上的**【趨勢圖】**按鈕後即會出現**【趨勢圖】**物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個**【趨勢圖】**物件。

下圖為【趨勢圖】物件的一般屬性設定頁。



【資料取樣物件索引】

選擇【資料取樣】物件作為繪圖所需的數據來源。

【資料來源】

選擇數據來源的形式，可以選擇【即時】或【歷史】。

a. 即時

可顯示來自【資料取樣】物件從開機後到目前的取樣資料，如需顯示他日的資料，需選擇【歷史形式】，從歷史資料中讀取。

可以利用【暫停控制】功能暫停物件畫面更新的動作，但僅只暫停畫面刷新，並不會暫停【資料取樣】物件的取樣動作。

b. 歷史

歷史記錄來自【資料取樣】物件使用日期來分類並儲存的取樣資料。使用【歷史】模式可以利用【資料取樣物件索引】選定要顯示的歷史記錄，並利用【歷史控制】選擇不同日期的歷史記錄。

EasyBuilder 會將取樣資料的歷史記錄檔案依時間先後排序，日期最新的檔案為記錄 0(一般是今日已存檔的取樣資料)，日期次新的檔案為記錄 1，其餘記錄依此類推。

在【歷史控制】中所指定暫存器中的數據如果為 0，【趨勢圖】物件將顯示記錄 0 的數據；暫存器中的數據如果為 1，將顯示記錄 1 的數據，也就是說暫存器中的數據如果為 n，將顯示記錄 n 的數據。

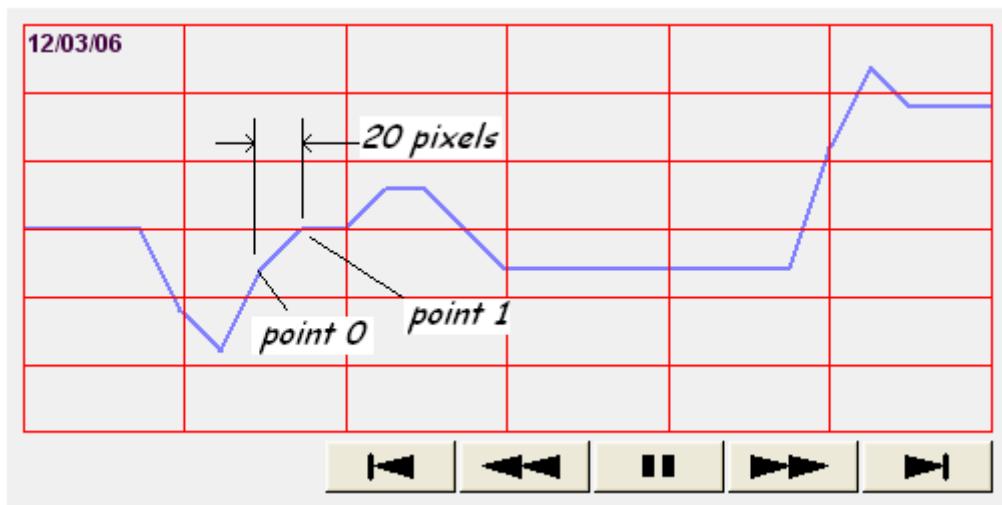
舉一個簡單的例子說明【歷史控制】的使用方式，若暫存器為 LW-0，假使目前的【資料取樣】物件已儲存的取樣資料檔案依時間先後分別為 pressure_20061120.dtl、pressure_20061123.dtl、pressure_20061127.dtl、pressure_20061203.dtl，共 4 筆檔案，並且今日時間為 2006/12/3，則依照 LW-0 中的數據內容，【趨勢圖】所顯示的取樣資料檔案整理如下：

LW-0 之數值	顯示的歷史資料取樣檔案
0	pressure_20061203.dtl
1	pressure_20061127.dtl
2	pressure_20061123.dtl
3	pressure_20061120.dtl

【兩取樣繪點間的距離】(選擇【像素】)



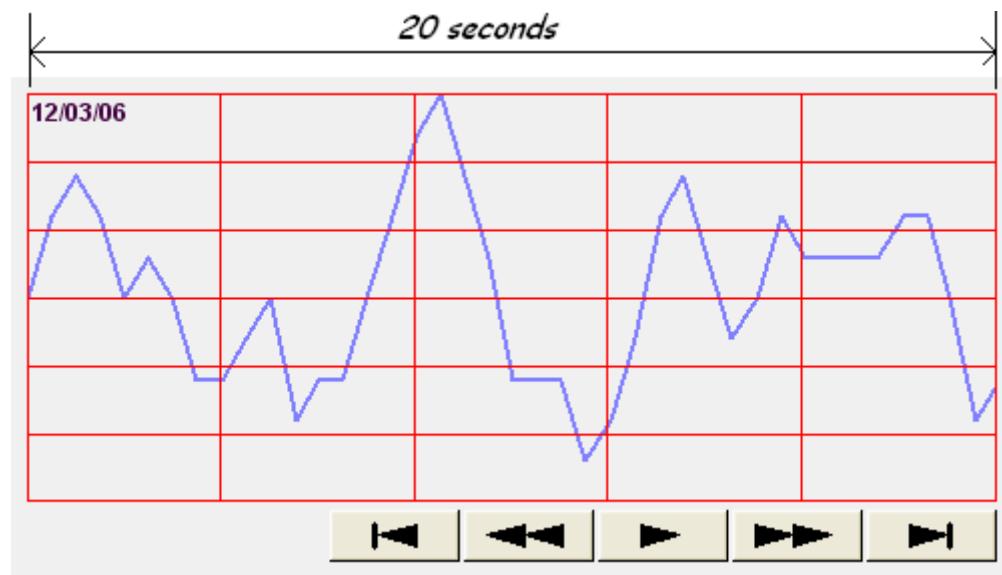
選擇【像素】，則【距離】用來設定各取樣點的描繪距離，參考下圖。



[X 軸表示時間範圍] (選擇 [時間])



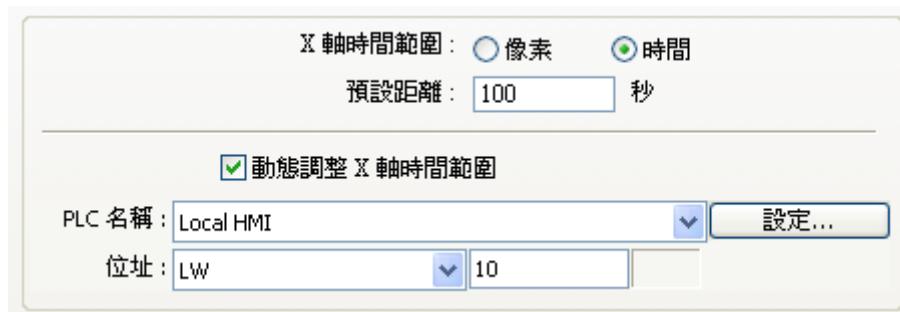
選擇[時間]，則 [距離] 用來設定物件寬度所顯示資料的時間範圍，如下圖。



此外，選擇 X 軸表示時間範圍，可在 [趨勢圖] 頁面的 [網格] 項目啟用 [時間刻度] 功能。

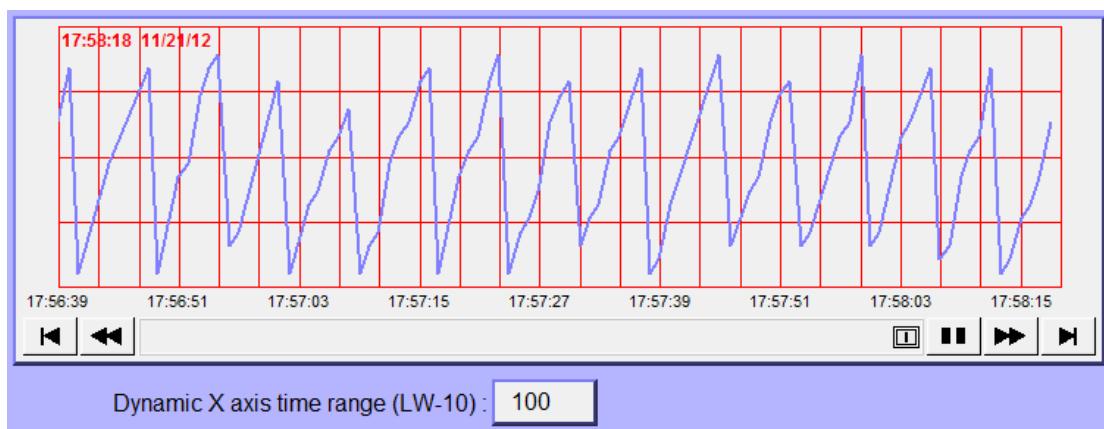
【動態調整兩取樣繪點間距 / 動態調整 X 軸時間範圍】

若勾選，可指定一個字元位址來線上動態調整 [像素] 或 [時間] 的距離。

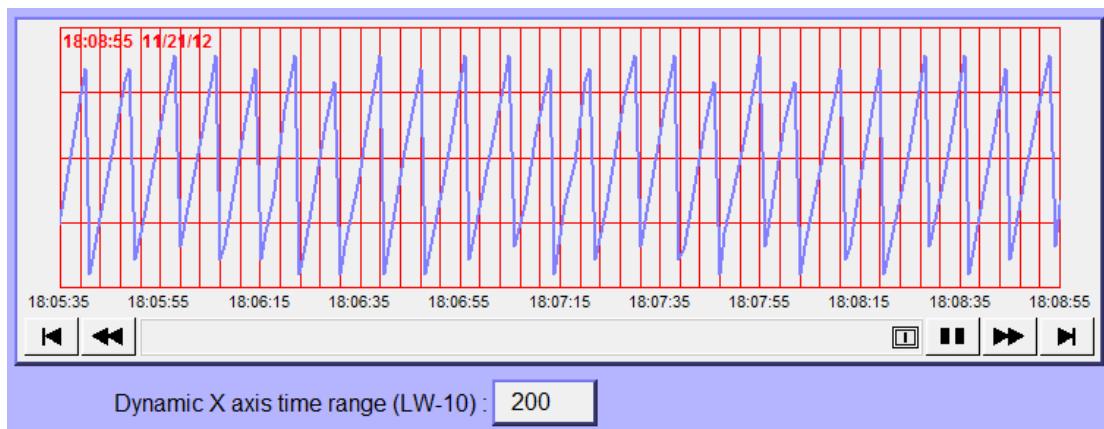


當啟用此項功能時，用戶需設定合理且有效的預設值，也就是當所指定暫存器中的數據為 0 時，將採用此項設定值來計算應顯示的數據。

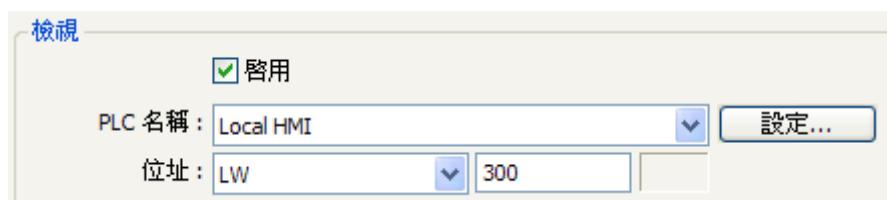
下圖為設定 X 軸顯示範圍為 100 秒的執行情形。



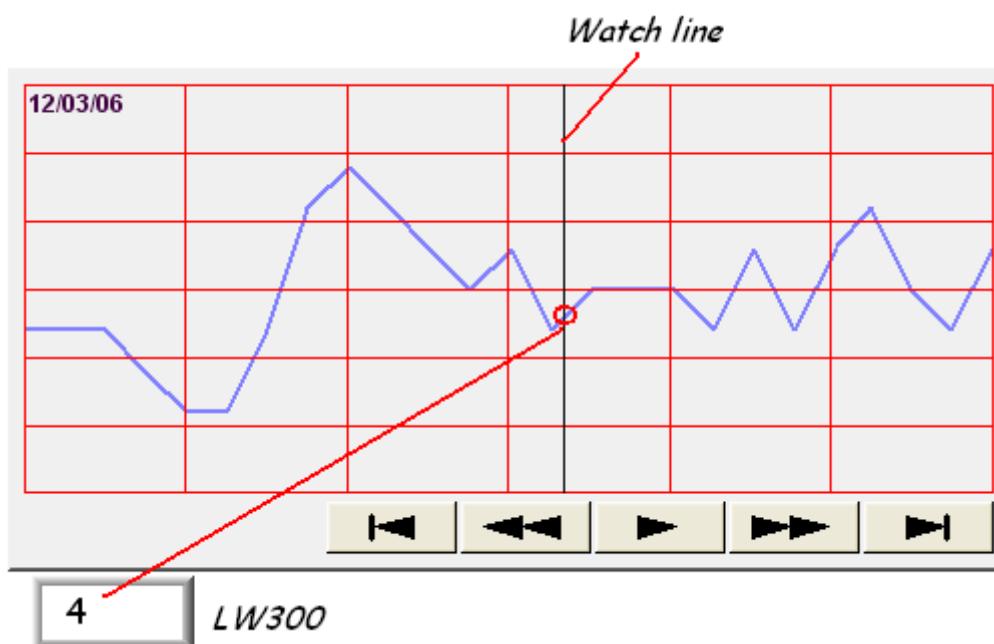
下圖則為設定 X 軸顯示範圍為 200 秒的執行情形。



檢視



使用 [檢視] 的功能可以讓使用者碰觸 [趨勢圖] 物件時產生標記符號，並可以將標記符號所在位置的取樣數據輸出到指定的位址，並使用 [數值顯示] 物件來顯示該數據。以下圖為例：



[檢視] 功能也可以輸出多條取樣曲線的取樣數據，系統會依照 [資料取樣] 物件中所定義的取樣資料數據格式，依序將標記所在位置的取樣數據，從 [檢視] 功能所定義的起始位置依序寫入。例如 [資料取樣] 物件的每個取樣資料包含數個數據，格式皆為不同，假設此時 LW-300 為 [檢視] 功能所定義的暫存器，則檢視線所標記的取樣資料的輸出位置如下。

[LW-300]	Ch. 0 : 16-bit Unsigned	(1 word)
[LW-301]	Ch. 1 : 32-bit Unsigned	(2 words)
[LW-303]	Ch. 2 : 32-bit float	(2 words)
[LW-305]	Ch. 3 : 16-bit Signed	(1 word)

時間標籤輸出

若啟用，系統將會以第一個取樣點的取樣時間作為時間原點並開始計數，並將最新取樣點之累計秒數輸出至 [時間標籤輸出位址 + 2]。

當點選趨勢圖物件上的曲線時，可將觸碰處最接近的取樣點之累計秒數輸出至 [時間標籤輸出位址]。

注意: [時間標籤輸出位址] 與 [時間標籤輸出位址 + 2] 皆須為 32-bit 格式。[時間標籤輸出位址 + 2] 只適用於趨勢圖 - 即時模式，而 [時間標籤輸出位址] 適用於趨勢圖 - 即時模式及歷史模式。

當趨勢圖頁籤的 [相對時間模式] 被勾選時，才可啟用此功能。

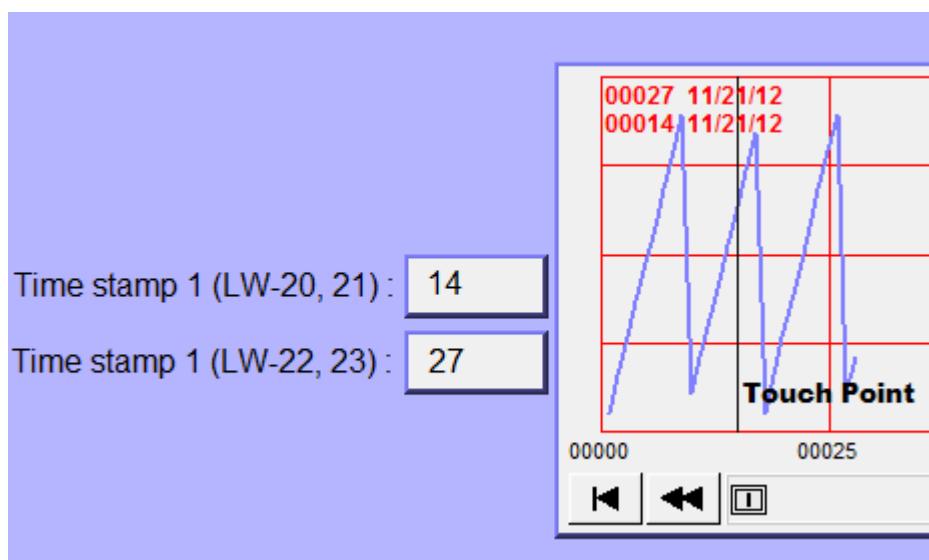
下面表格說明當所指定的暫存器為 16-bit 形態時，時間標籤在所指定暫存器中的數據排列情形。

暫存器地址	觸控處最接近的取樣點之取樣時間的低WORD數據
暫存器地址 + 1	觸控處最接近的取樣點之取樣時間的高WORD數據
暫存器地址 + 2	最新取樣時間的低WORD數據
暫存器地址 + 3	最新取樣時間的高WORD數據

下圖顯示執行 [時間標籤輸出] 功能的情形。

[LW-20, 21] 暫存器中所顯示的 14(秒) 為觸控處最接近取樣點的取樣時間。

[LW-22, 23] 暫存器中所顯示的 27(秒) 為目前最新取樣點的取樣時間。



下圖為 [趨勢圖] 物件的 [趨勢圖] 設定頁。



[外框]

物件的外框顏色。

[背景]

物件的背景顏色。

[使用畫面捲動控制按鈕]

啟用 / 取消 畫面捲動控制按鈕。



網格

設定格線的數目與顏色。

[水平]

設定格線水平線的數目。

[垂直]

a. 像素



當選擇 [像素] 來設定取樣點的描繪距離時，則 [垂直間隔] 用來選擇每兩個垂直格線間將包含幾個取樣點。

垂直 : 4 點

b. 時間

若選擇 [時間] 來設定物件寬度所顯示資料的時間範圍，則 [垂直間隔] 用來選擇每兩個垂直格線間所顯示資料的時間範圍。

垂直 : 4 秒

系統會利用這些設定，自動計算垂直格線的數目。

時間刻度

選擇 [顯示] 來顯示時間刻度於趨勢圖的底部。

[格式]

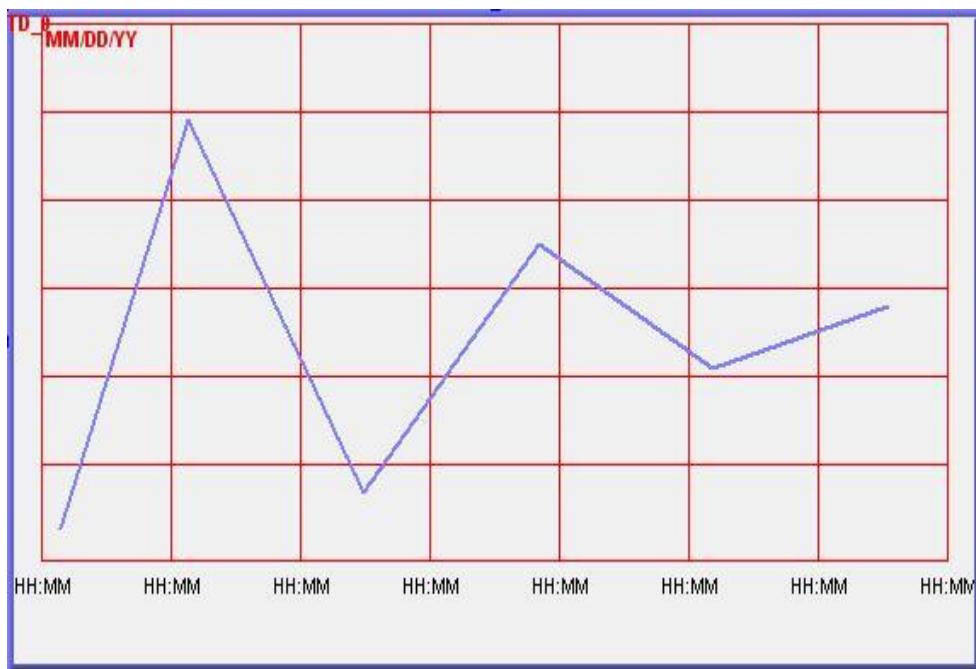
選擇時間刻度顯示的格式。

[字型]

選擇字的字型。

[尺寸]

選擇字型大小，預設值為 8。



相對時間模式

若勾選，系統將會以第一個取樣點的取樣時間作為時間原點，也就是左上角所顯示的取樣時間與X軸時間刻度的顯示將從 "00:00:00" 或 "00:00" 或 "0" 或 "00000" 開始計數。

此外，趨勢圖左上角之時間標籤與X軸時間刻度將可以選擇使用 [SSSSS] 或 [SSSSS (前導零)] 作為顯示格式，這些顯示格式皆以 "秒" 為單位。

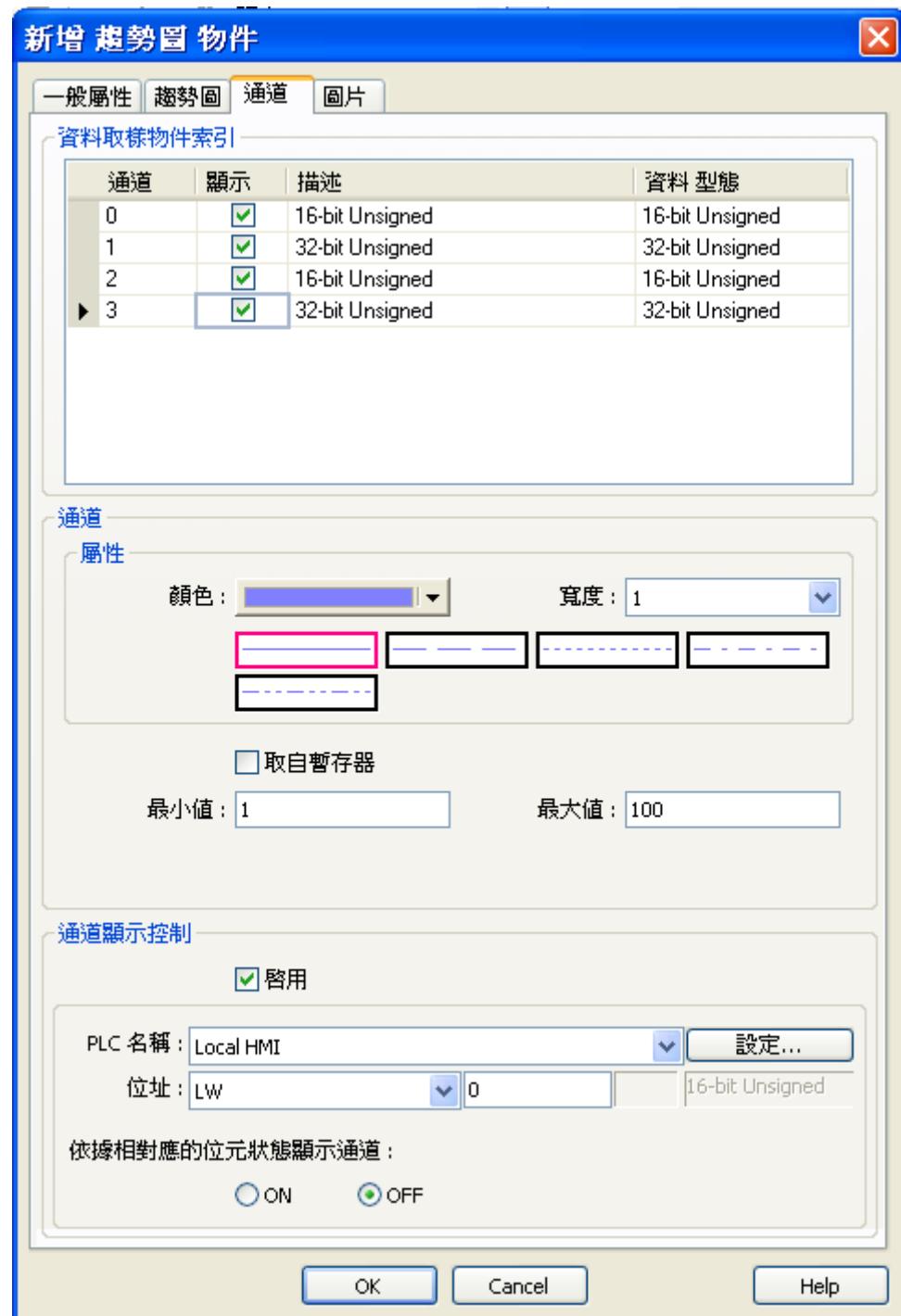
若使用【資料取樣】物件中的【清除控制】功能，除了可以清除目前的取樣數據之外，也可以重置取樣時間，也就是重新取樣後的第一個取樣時間點仍將作為時間原點。下圖為【清除控制】功能的設定內容。



時間 / 日期

最新的取樣資料所獲得的時間會被標示在物件的左上角，此項目用來設定時間的顯示格式與顏色。

下圖為 [趨勢圖] 物件的 [通道] 設定頁。



[通道]

設定各個曲線的樣式與顏色，與曲線所能描繪數據的上下限值。

最多可同時支援 20 個通道。

不勾選 [取自暫存器]

[最小值]、[最大值]

[最小值] 與 [最大值] 用來設定各曲線所描繪的取樣數據的最小值與最大值。也就是說如果存

在某一曲線所描繪的取樣數據最小值為 50，最大值為 100，則[最小值]與[最大值]需設定為[50]與[100]，如此所有的取樣數據才會完全被描繪在物件中。

勾選"取自暫存器"

上下限可由指定暫存器設定。當寫入位址為 LW-n，則上/下限會根據以下的規則自動被設定為：

位址格式	16-bit	32-bit
下限	LW-n	LW-n
上限	LW-n+1	LW-n+2

以下表為例，當暫存器為 LW-100 時，則上/下限的位址會自動被設定為：

位址格式	16-bit	32-bit
下限	LW-100	LW-100
上限	LW-101	LW-102

此設定常用於放大與縮小趨勢圖。

【通道顯示控制】

當選擇使用【通道顯示控制後】，則此位址中的各個位元將會被用來控制各個通道的顯示與否。Bit-1 控制通道 1，Bit-2 控制通道 2，依此類推。舉例來說，建立 5 個通道，並設定通道控制的位址為 LW-0，則各個通道會被以下控制位址控制：

通道編號	控制地址	位元狀態	是否顯示
1	LW_bit-000	OFF	YES
2	LW_bit-001	ON	NO
3	LW_bit-002	ON	NO
4	LW_bit-003	OFF	YES
5	LW_bit-004	OFF	NO

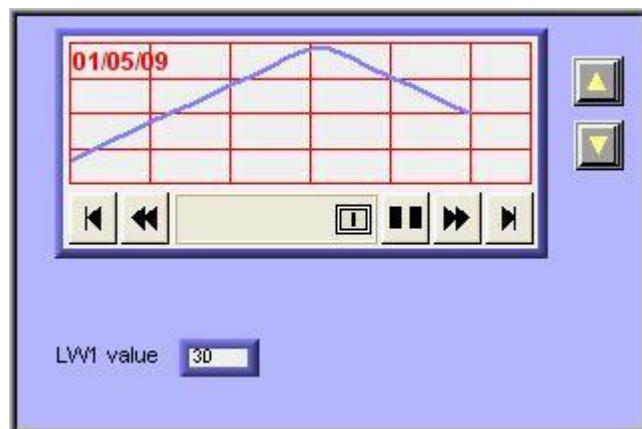
Example 1 如何縮放趨勢圖?

使用者需勾選【取自暫存器】以實現此功能。



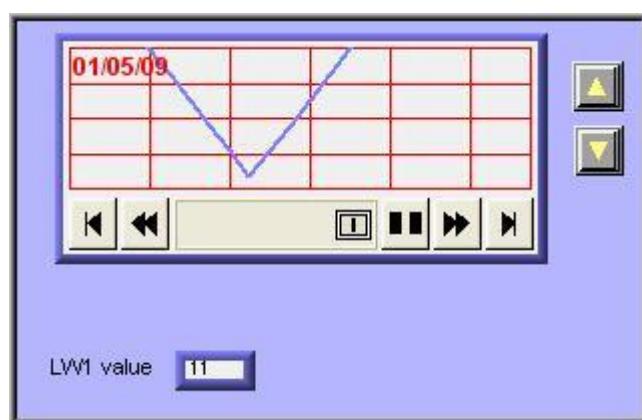
例如, LW0 和 LW1 用來控制最小值和最大值, 使用者可透過控制 LW1 的數值來縮放趨勢圖。

下圖表示原尺寸。趨勢圖範圍原本是介於 0~30。視窗右方的箭頭為 [多狀態設定] 物件 (LW1, 遞加(JOG+) 和 LW-1, 遞減(JOG-)) 來控制縮放功能。



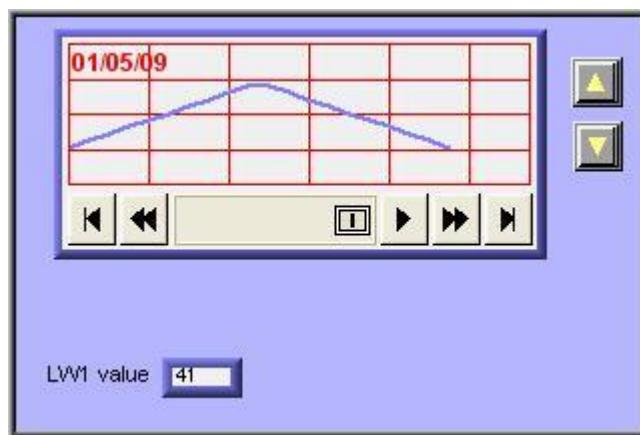
遞減 LW-1 的值來展示放大的功能:

下圖顯示 LW-1 的值被減到 11。



遞增 LW-1 的值來展示縮小的功能：

下圖顯示 LW-1 的值被加到 41。



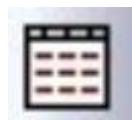
13.18 歷史數據顯示

概要

【歷史數據顯示】物件用來顯示已經儲存的取樣資料數據，跟趨勢圖不同的是【歷史數據顯示】物件使用表列的方式直接顯示這些數據的內容，由於是歷史數據，所以要顯示最新的資料需經由切換畫面來顯現，如下圖所示。

編號	時間	日期	Ch.0	Ch.1	Ch.2	▲
3577	21:52	16/09/07	0	0	0	▲
3576	21:52	16/09/07	0	0	0	▲
3575	21:52	16/09/07	0	0	0	▲
3574	21:52	16/09/07	0	0	0	▲
3573	21:52	16/09/07	0	0	0	▲
3572	21:52	16/09/07	0	0	0	▲
3571	21:52	16/09/07	0	0	0	▲
3570	21:52	16/09/07	0	0	0	▲
3569	21:52	16/09/07	0	0	0	▲
3568	21:52	16/09/07	0	0	0	▲

設定



按下工作列上的【歷史數據顯示】按鈕後即會出現【歷史數據顯示】物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個【歷史數據顯示】物件。



[資料取樣物件索引]

選擇 [資料取樣] 物件作為所需的數據來源。

格線

選擇物件是否使用格線區分每個欄位。

No.	Time	Date	Ch.0	Ch.1	Ch.2	▲
3982	22:02	16/09/07	0	0	0	◀
3981	22:02	16/09/07	0	0	0	▶
3980	22:02	16/09/07	0	0	0	◀
3979	22:02	16/09/07	0	0	0	▶
3978	22:02	16/09/07	0	0	0	◀
3977	22:02	16/09/07	0	0	0	▶
3976	22:02	16/09/07	0	0	0	◀
3975	22:02	16/09/07	0	0	0	▶
3974	22:02	16/09/07	0	0	0	◀
3973	22:02	16/09/07	0	0	0	▶

【顏色】

設定格線所使用的顏色。

【欄位距離】

此項設定值用來調整各欄位間的距離，下圖為使用不同【欄位距離】設定時的顯示情形。

No.	Time	Date	Ch.0	Ch.1	Ch.2	▲
3667	21:57	16/09/07	1	0	0	◀
3666	21:57	16/09/07	1	0	0	▶
3665	21:57	16/09/07	1	0	0	◀
3664	21:57	16/09/07	1	0	0	▶
3663	21:57	16/09/07	1	0	0	◀
3662	21:57	16/09/07	1	0	0	▶
3661	21:57	16/09/07	1	0	0	◀
3660	21:56	16/09/07	0	0	0	▶
3659	21:56	16/09/07	0	0	0	◀
3658	21:56	16/09/07	0	0	0	▶

No.	Time	Date	Ch.0	Ch.1	Ch.2	▲
3667	21:57	16/09/07				◀
3666	21:57	16/09/07				▶
3665	21:57	16/09/07				◀
3664	21:57	16/09/07				▶
3663	21:57	16/09/07				◀
3662	21:57	16/09/07				▶
3661	21:57	16/09/07				◀
3660	21:56	16/09/07				▶
3659	21:56	16/09/07				◀
3658	21:56	16/09/07				▶

外觀

設定物件的外框與背景顏色，若勾擇【透明】表示不使用外框與背景顏色。

時間 & 日期

用來選擇是否顯示資料的取樣時間與日期，並決定時間與日期的顯示格式。

【按時間順序】

選擇【按時間順序】表示將先顯示取樣時間較早的資料，參考下圖。

No.	Time	Date	Ch.0	Ch.1	Ch.2	▲
1	00:24:27	16/09/07	2	2		◀
2	00:24:28	16/09/07	4	4		▶
3	00:24:29	16/09/07	7	6		◀
4	00:24:30	16/09/07	9	8		▶
5	00:24:31	16/09/07	6	4		◀
6	00:24:32	16/09/07	4	2		▶
7	00:24:33	16/09/07	1	4		◀
8	00:24:34	16/09/07	3	6		▶
9	00:24:35	16/09/07	6	6		◀
10	00:24:36	16/09/07	8	4		▶

[按時間逆序]

選擇 [按時間逆序] 表示將先顯示取樣時間較晚的資料，參考下圖。

No.	Time	Date	Ch.0	Ch.1	C
4787	22:24:15	16/09/07	2	2	▲
4786	22:24:00	16/09/07	3	2	
4785	22:23:59	16/09/07	3	2	
4784	22:23:58	16/09/07	3	2	
4783	22:23:57	16/09/07	3	2	
4782	22:23:56	16/09/07	3	2	
4781	22:23:55	16/09/07	3	2	
4780	22:23:54	16/09/07	3	2	
4779	22:23:53	16/09/07	3	2	
4778	22:23:52	16/09/07	3	2	▼

歷史數據控制

系統會將取樣資料的歷史記錄檔案依時間先後排序，日期最新的檔案為記錄 0 (一般是今日已存檔的取樣資料)，日期次新的檔案為記錄 1，其餘記錄依此類推。[歷史數據控制] 項目則用來指定要顯示哪一個記錄。

下圖為歷史數據顯示物件的【編輯】設定頁。



啟用

若啟用，可於 HMI 線上修改資料取樣的歷史數據，並配合此物件立即檢視內容。

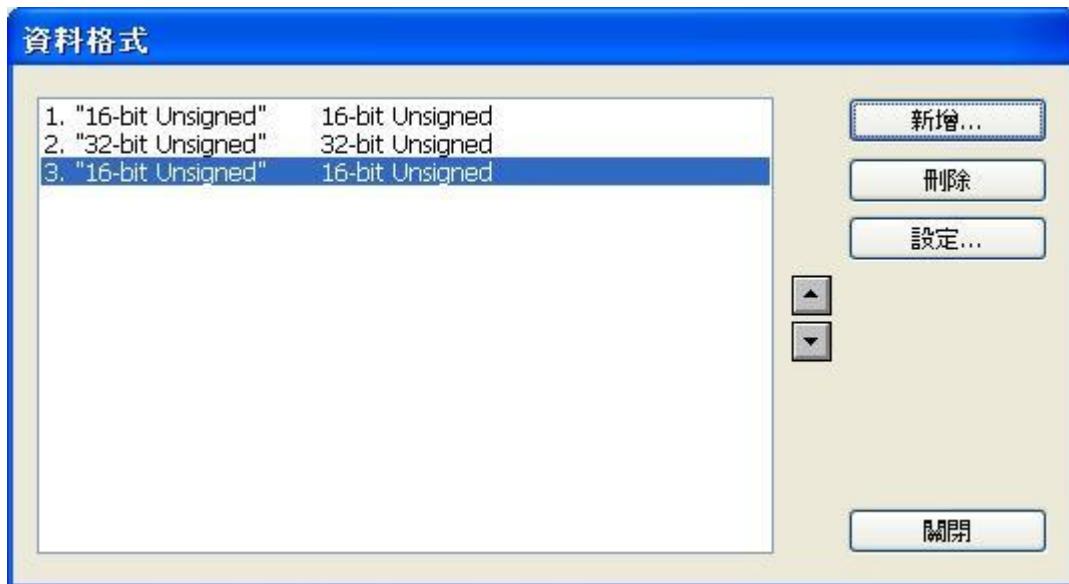
控制位址

當設數值為 1 時，可將【資料位址】內的數據覆寫至資料取樣。

資料位址

當選取歷史數據顯示物件中的某一行數時，系統會將該行數的數據依序讀出，此時即可修改資料取樣中的數據。

注意：此位址的編排格式需配合資料取樣中的資料格式設定。以下圖為例，第一個位址應為 16-Bit，第二個位址應為 32-Bit，第三個位址應為 16-Bit。



選擇位址

指示當前歷史數據顯示物件被選取的行數，修改此位址內的數值亦可移動選擇行數。

注意：當選擇位址內的數值為 0 時，將不會有行數被選擇，且 [資料位址] 內的數值將維持前一個選擇行數之數據。當選擇位址內的數值大於實際行數時，將選擇當前最大值之行數。

選擇顏色

指示當前歷史數據顯示物件被選取的行數之背景顏色。

注意：當資料取樣的儲存位置為外部裝置時，若將外部裝置移除將無法正確的使用此功能，須等到外部裝置再次插入且系統讀取到歷史數據後，才可繼續動作。

下圖為歷史數據顯示物件的【數據顯示格式】設定頁。



上圖的對話窗用來設定取樣資料的顯示格式，最多可顯示 20 個通道。由上圖可以發現目前使用的【資料取樣】物件執行一次取樣的動作將讀取 4 個數據(通道 0~通道 3)，由上圖也可以發現各數據的數值格式(例如通道 0 為 16-bit Unsigned)，這些皆定義在【資料取樣】物件中。由上圖可以看出目前只設定顯示通道 0 與通道 3 的數據，參考下圖。

編號	時間	日期	Ch.0	Ch.3
5272	22:43:09	16/09/07	4	1
5271	22:43:08	16/09/07	2	0
5270	22:33:42	16/09/07	0	0
5269	22:33:41	16/09/07	0	0
5268	22:33:40	16/09/07	0	0
5267	22:33:39	16/09/07	0	0
5266	22:33:38	16/09/07	0	0
5265	22:33:37	16/09/07	0	0
5264	22:33:36	16/09/07	0	0
5263	22:33:35	16/09/07	0	0

當使用歷史數據顯示物件顯示字組 [String] 格式時，可以選擇：

- 使用 [UNICODE] 模式顯示。
- 將數據的高位元組與低位元組資料互換後，再加以顯示。



下圖為歷史數據顯示物件的【標題】設定頁。



【使用標題】

選擇是否使用標題。

A screenshot of a history data display window showing a table with four columns: No., Time, Date, and Ch.0. The first row is highlighted with a red oval. The data in the table is as follows:

No.	Time	Date	Ch.0
5272	22:43:09	16/09/07	4
5271	22:43:08	16/09/07	2

標題背景

[透明]

勾選 [透明] 表示不使用標題文字的背景色。

[顏色]

設定標題文字的背景色。

[設定]

設定標題的文字。

No.	Time	Date	Ch.0
5272	22:43:09	16/09/07	4
5271	22:43:08	16/09/07	2

標題也可以使用文字標籤庫，顯示多國語言，只需使用[設定]。在出現設定對話窗後，選擇使用文字標籤庫即可。

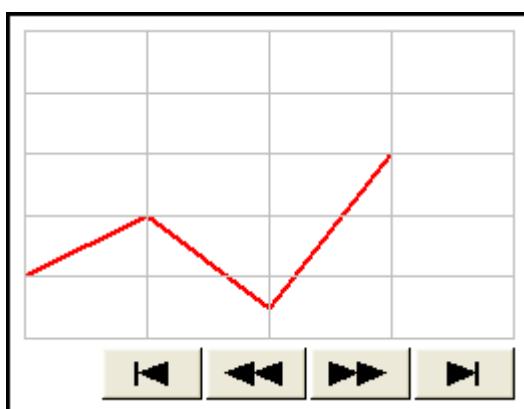
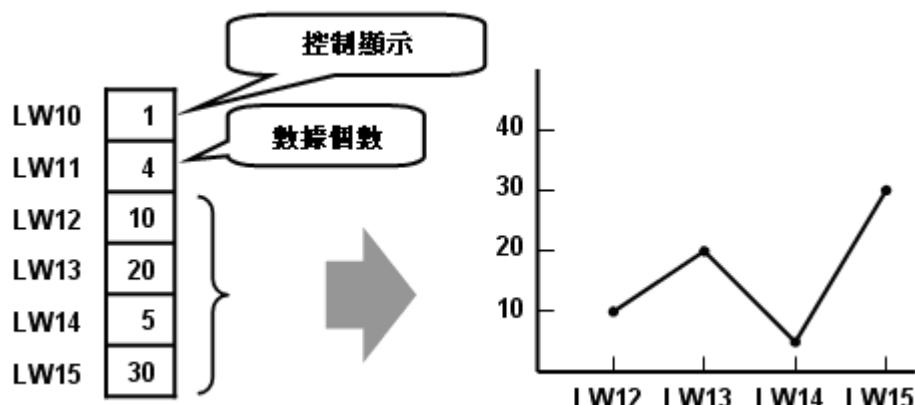


當改變過資料格式於資料取樣後，在執行離線模式後，不可再更改資料取樣，如果需要改變，請將 C:\EasyBuilder\HMI_memory\datalog 內記錄歷史數據顯示的檔案刪除，再進行離線模擬的動作。

13.19 數據群組顯示

概要

一個數據群組(或區塊)是指一組連續位址中的數據，例如 LW-12、LW-13、LW-14、LW-15 等。數據群組顯示物件可同時顯示多個數據群組的內容，例如同時顯示 LW-12~LW-15 與 RW-12~RW-15 兩個數據群組，使用者可藉由此方式來觀察及比較各暫存器中的資料。下圖為使用數據群組顯示物件顯示單一數據群組 LW-12~LW-15 中的數據。

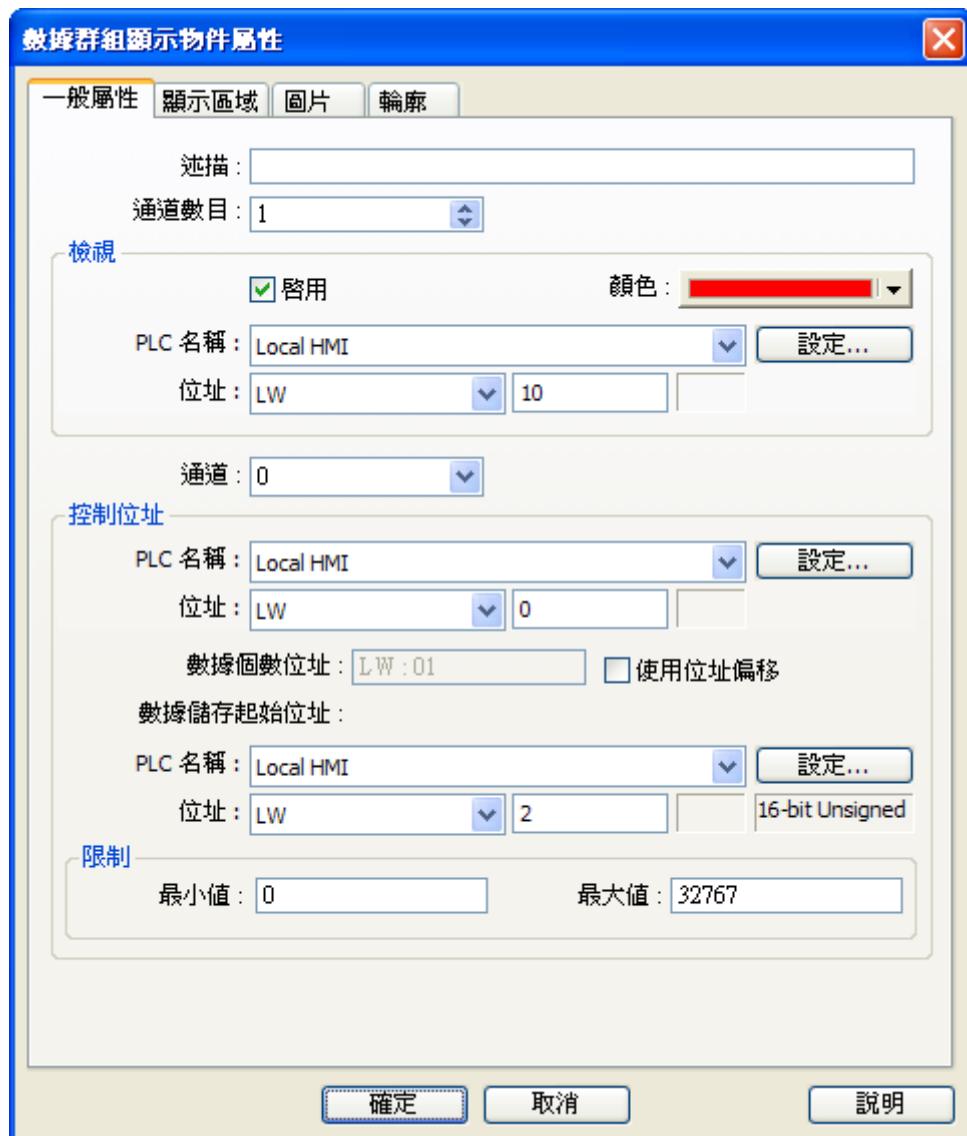


實際執行結果

設定



按下工作列上的 [數據群組顯示] 按鈕，隨即出現物件屬性對話窗。



[通道數目]

設定物件的通道數目。每個通道表示一組群組數據。最多可同時支援 12 組。

檢視

當使用 [檢視] 功能，使用者若觸碰此物件，將可顯示檢視線上的數據和位置於所指定的暫存器。

[通道]

用來指定要設定的數據群組屬性。

控制位址

[PLC 名稱]

選擇數據群組的數據來源。

[設備類型]

選擇目前所指定數據群組的位址類型。

[控制位址]

用來控制圖形的顯示及清除。

0 = 無動作 (預設值)

1 = 繪圖

2 = 清除

3 = 重新繪圖

當執行完上述動作後，系統會將控制位址重設為 0。

[數據個數位址]

定義為“控制位址 +1”。

用來存放群組的數據數量，可支援達 1024 個。

[數據儲存起始位址]

若取消 [使用位址偏移]，請設定儲存資料的起始位址。

[數據儲存偏移位址]

若啟用“使用位址偏移”，“數據儲存偏移位址”定義為“控制位址” + 2。

[格式]

若選擇 16-bit 格式，每個起始數據的位址為位址，起始位址 + 1，起始位址 + 2，等等。

若選擇 32-bit 格式，每個起始數據的位址為位址，起始位址 + 2，起始位址 + 4，等等。

限制

用來設定所顯示圖形之高、低限。

下圖為數據群組顯示物件的【顯示區域】設定頁。



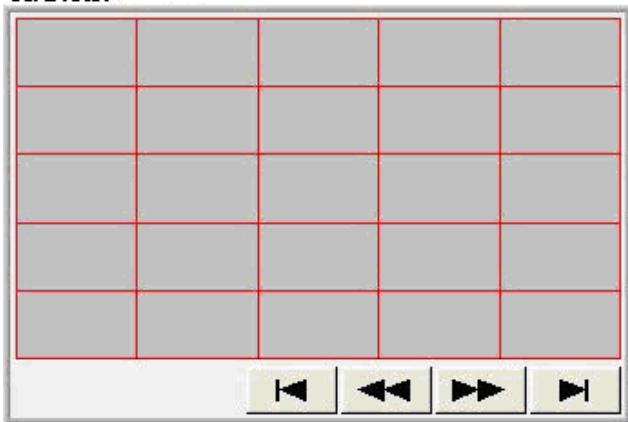
[顯示點數]

設定圖形一頁所能顯示最大資料筆數。

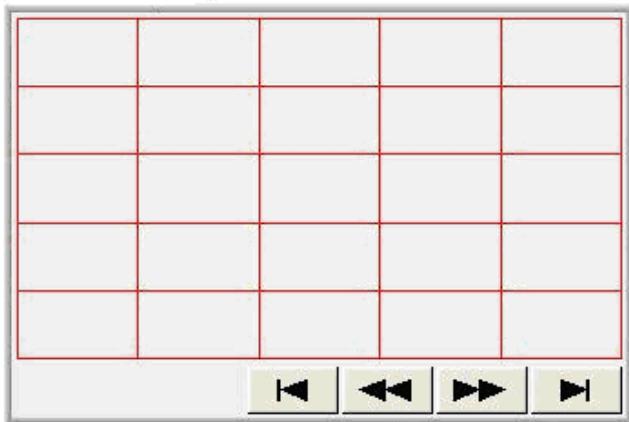
[捲動量]

左右捲動的資料筆數。

使用背景

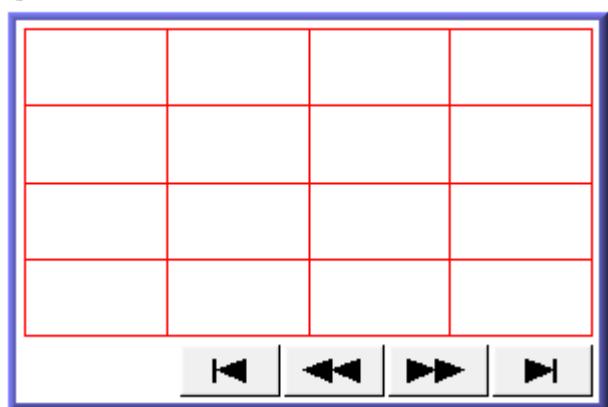


使用透明

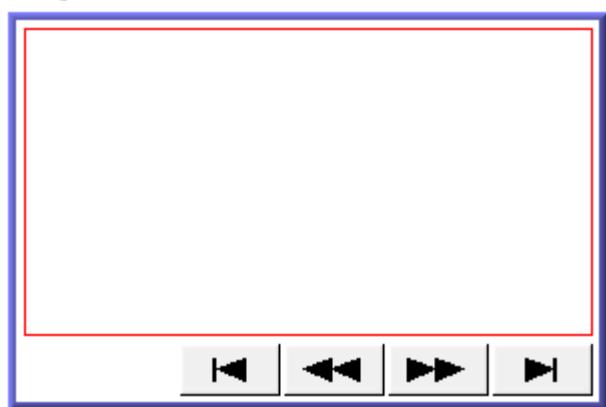


網格

使用網格



非使用網格



通道

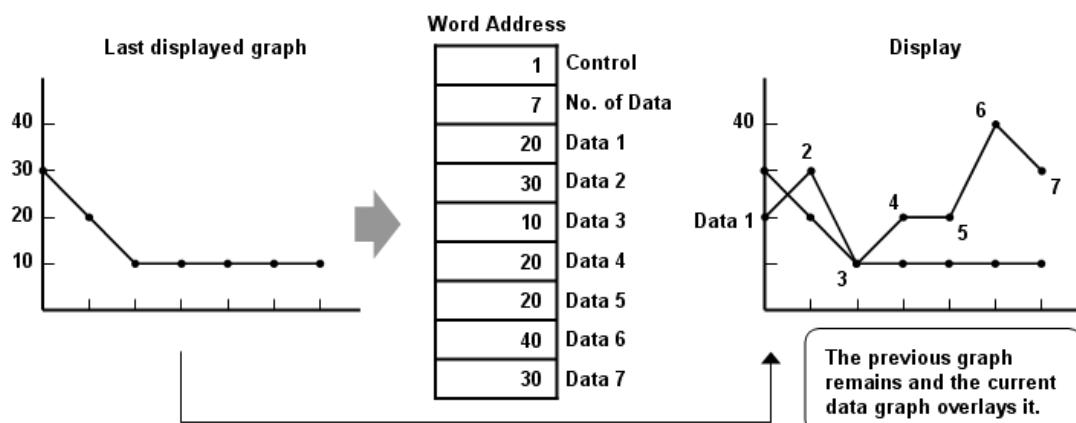
設定各數據群組圖形之線條顏色、粗細及樣式。



操作方式

1. 如何顯示數據群組的內容

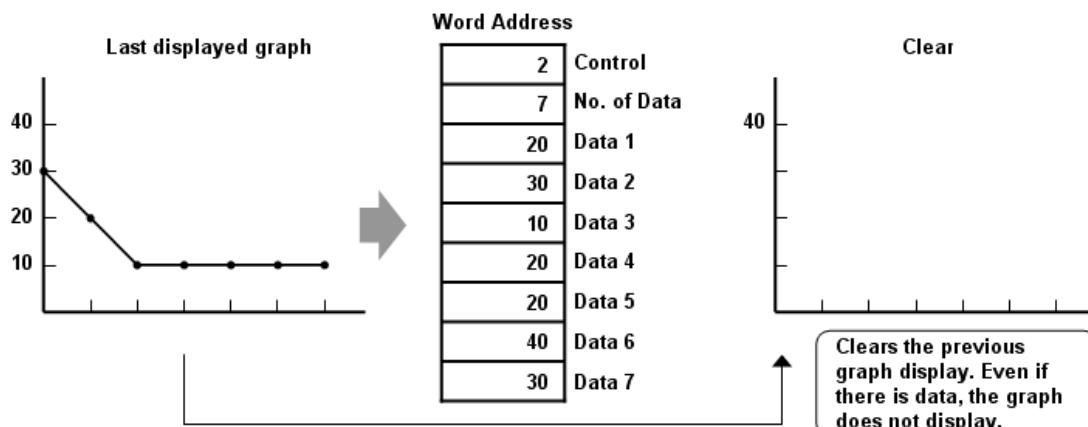
- 在 [數據個數位址] 寫入欲顯示的數據筆數，也就是“控制位址 +1”。
- 在 [數據儲存起始位址] 依序填入數據內容。
- 在 [控制位址] 寫入”1”；此時 HMI 將以折線圖畫出目前暫存器的內容 (並保留先前圖形)。
- HMI 在完成前項動作後將對[控制位址]寫入”0”。



- 在上述動作 c 和 d 之間，請勿更改 [控制位址]、[數據個數位址] 及 [數據儲存起始位址] 內容，否則可能產生非預期結果。

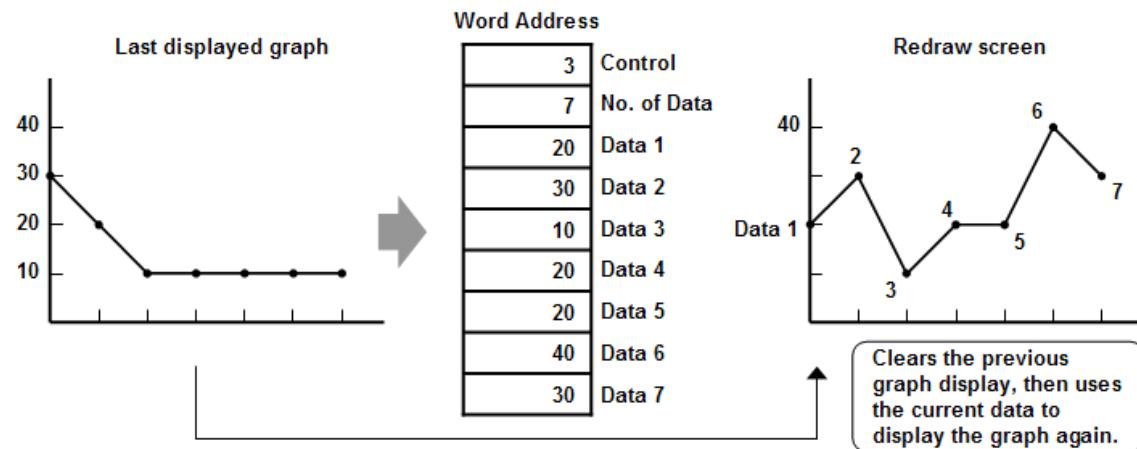
2. 如何清除已顯示的圖形

- 在[控制位址]寫入”2”；將清除先前所畫之線圖。
- HMI 在完成前項動作後將於 [控制位址] 寫入”0”。



3. 清除已顯示的圖形並顯示新數據的圖形

- 在[數據個數位址]寫入欲顯示的數據筆數，也就是“控制位址 +1”。
- 在[數據儲存起始位址]依序填入數據內容。
- 在[控制位址]寫入”3”；此時 HMI 會先將先前的線圖清除，再畫出目前位址內的內容。
- HMI 在完成前項動作後將於[控制位址]寫入”0”。



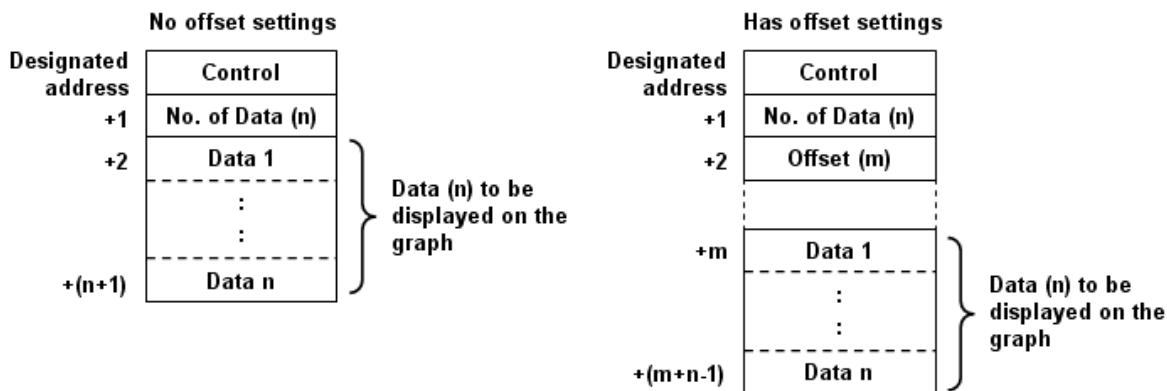
4. 位址偏移模式

啟用後，則各個通道的 [控制位址]、[數據個數位址]、[數據儲存偏移位址] 會使用連續的 3 個位址。

例如有 3 個通道(通道 0 至通道 2)，且 [控制位址] 分別為 LW-0、LW-100 和 LW-200，則各個通道的 [控制位址]、[數據個數位址]、[數據儲存偏移位址] 如下：(下表使用 3 個通道數，格式皆為 16-bit Unsigned 且數據儲存偏移位址設為 m)

項目	控制位址	數據個數位址	數據儲存偏移位址	資料 1	資料 2	...
通道 0	LW-0	LW-1	LW-2 (=m)	LW-0+m	LW-1+m	...
通道 1	LW-100	LW-101	LW-102 (=m)	LW-100+m	LW-101+m	...
通道 2	LW-200	LW-201	LW-202 (=m)	LW-200+m	LW-201+m	...

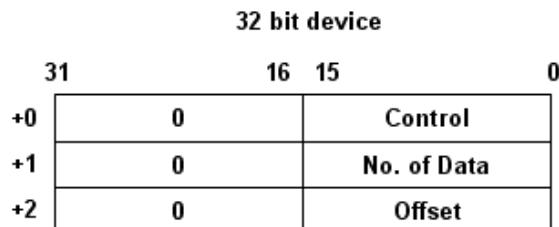
下圖左側代表未使用 [偏移模式] 的讀取方式，右側則是使用位址偏移模式的讀取方式。



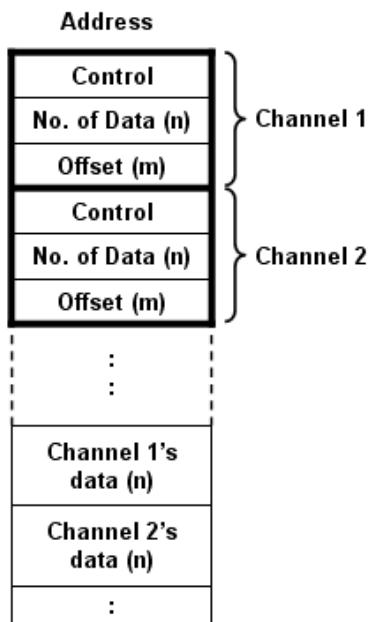
- 當【控制位址】設定為 LW-n 時，【數據個數位址】及【數據儲存偏移位址】會根據以下規則設定：

項目	16 位元	32 位元
控制位址	LW-n	LW-n
數據個數位址	LW-n+1	LW-n+2
數據儲存偏移位址	LW-n+2	LW-n+4

- 當【控制位址】為 32 位元時，只有較低的 16 位元產生作用，請將較高的 16 位元內容設為 0。

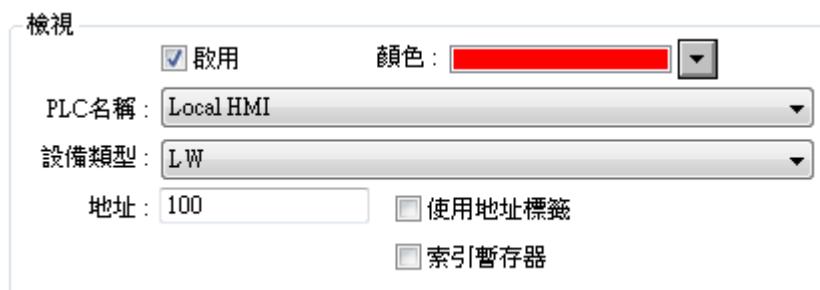


- 系統會在【控制位址】的數據不為 0 時讀取【數據個數位址】及【數據儲存偏移位址】的內容。
- 當使用了兩個同類型暫存器以上的通道，則啟用【使用位址偏移】並使用連續的位址做控制位址可減少系統讀取數據的時間。如下圖。當使用 16 位元格式時，設定通道 1 的控制位址為 LW-n，通道 2 的控制位址為 LW-n+3，依此類推。

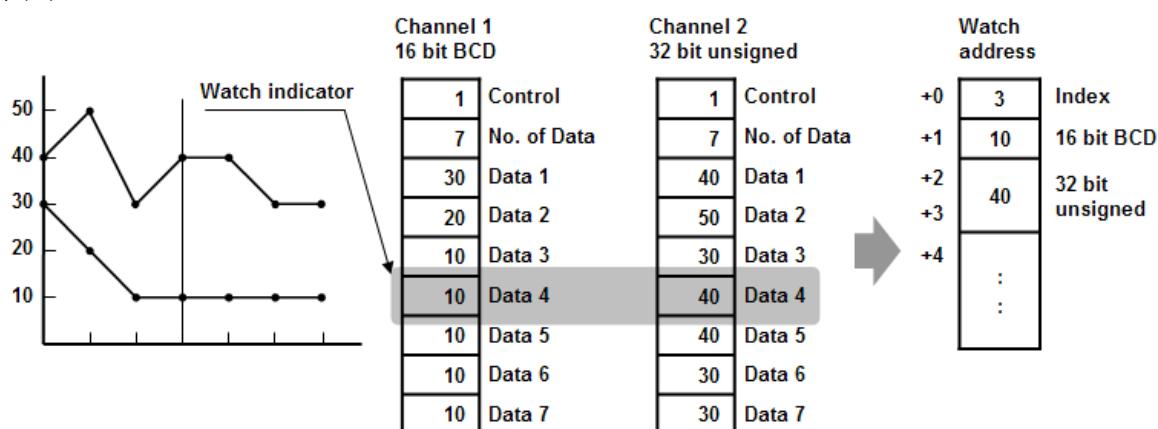


數據檢視功能

啟用【檢視】功能，則當觸控此物件上任一點時，會顯示一條垂直的檢視線，曲線與該檢視線交會形成的點所相對應的數值將會寫入至指定的暫存器中。



若設定檢視位址為 LW-n，則在 LW-n 寫入數值代表從各通道欲呼叫的索引編號 (從 0 計算)，如下圖：

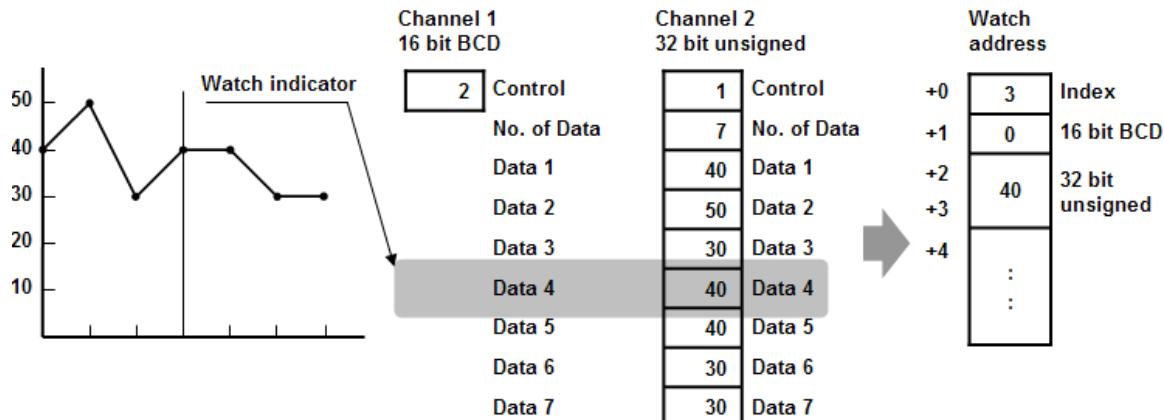




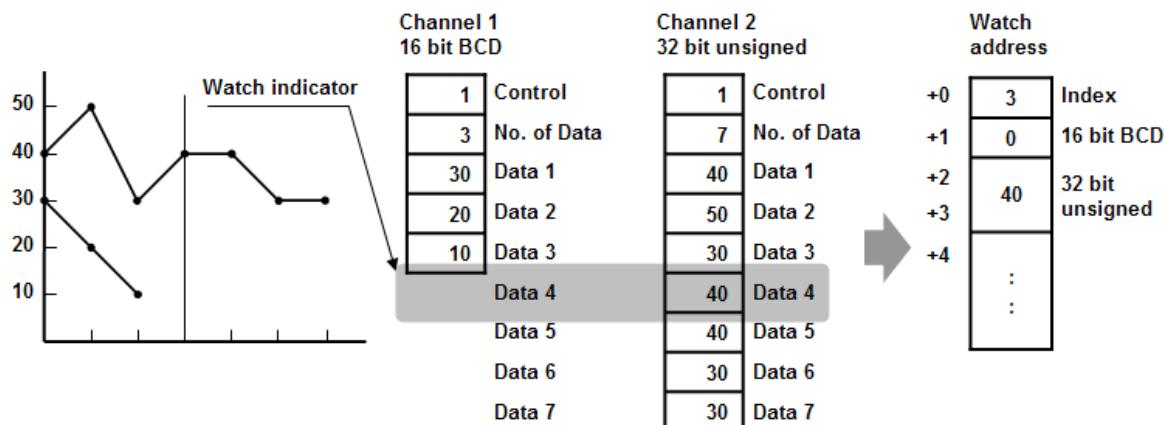
- 索引編號為 16 位元無號數整數。若指定之暫存器為 32 位元時，只有較低的 16 位元可作用，並忽略使用較高的 16 位元。

- 當檢視的通道無資料時，則會以 0 代替。

EX: 僅有通道二的資料，通道一無資料，LW-n+1 顯示為 0。



EX: 通道一僅有三筆資料，若檢視第四筆，則 LW-n+1 顯示為 0。



- 可支援的通道數目最多為 12 組。
- 重複繪圖上限次數為 32 除以通道數。
- 每個通道最多可顯示 1024 筆數據。

13.20 XY 曲線圖

概要

XY 曲線圖物件用來顯示二維座標的 XY 資料點，每個數據包含 X 和 Y 值，皆從暫存器中讀取。同時可顯示最多 16 組曲線。此功能可讓使用者觀察及分析各暫存器中的資料。負數亦可使用。

設定



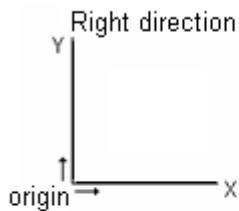
按下工作列上的 [XY 曲線圖] 按鈕，隨即出現物件屬性對話窗：



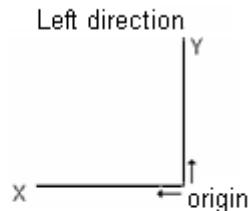
[方向]

XY 軸方向可選擇 [朝右]、[朝左]、[朝上] 或 [朝下] 顯示，如下圖：(O 代表原點)

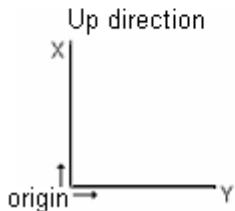
右:



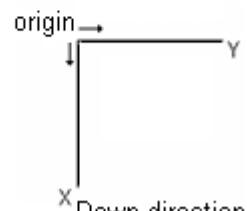
左:



上:



下:

**[通道數目]**

欲觀察的資料筆數。

控制位址**[控制位址]**

用來控制 XY 曲線圖的顯示或清除 XY，當控制位址設定為 LW-n，則對 LW-n 寫入不同的數值代表不同的命令。同時，LW-n+1 會被用來調整顯示的數據個數。

控制位址	數值	結果
LW-n	1	顯示目前圖形 (保留已繪製的圖形)
	2	清除圖形
	3	清除所有圖形，並重新顯示目前圖形
LW-n+1	任意數	顯示的數據個數

當 HMI 完成指定的動作後會將 [控制位址] 之值設為 0。

[數據個數位址]

此位址是用來儲存資料顯示的數量。每個通道可以有高達 1023 個 XY 數據。

[通道]

指定一個通道並設定讀取的相關屬性。

讀取位址**[PLC 名稱]**

選擇讀取的資料來源裝置。

存取暫存器資料時，須同時考慮是否啟用【X 軸數據和 Y 軸數據來自不同位址】和【上下限值取自暫存器】。以下以實例說明各情況（假設皆使用 16-bit 暫存器）：

1. 假設停用 X 軸數據和 Y 軸數據來自不同位址，當【讀取位址】設為 LW-0 時：

	啟用【上下限值取自暫存器】		停用【上下限值取自暫存器】	
	X 資料	Y 資料	X 資料	Y 資料
下限	LW-n	LW-n+1	常數	常數
上限	LW-n+2	LW-n+3	常數	常數
第一筆數據	LW-n+4	LW-n+5	LW-n+0	LW-n+2
第二筆數據	LW-n+6	LW-n+7	LW-n+1	LW-n+3
第三筆數據	LW-n+8	LW-n+9	LW-n+4	LW-n+5
第四筆數據	LW-n+10	LW-n+11	LW-n+6	LW-n+7

2. 假設啟用 X 軸數據和 Y 軸數據來自不同位址，當【X 資料】為 LW-m，【Y 資料】為 LW-n：

	啟用【上下限值取自暫存器】		停用【上下限值取自暫存器】	
	X 資料	Y 資料	X 資料	Y 資料
下限	LW-m+0	LW-n+0	常數	常數
上限	LW-m+1	LW-n+1	常數	常數
第一筆數據	LW-m+2	LW-n+2	LW-m+0	LW-n+0
第二筆數據	LW-m+3	LW-n+3	LW-m+1	LW-n+1
第三筆數據	LW-m+4	LW-n+4	LW-m+2	LW-n+2
第四筆數據	LW-m+5	LW-n+5	LW-m+3	LW-n+3

範圍上下限

當【上下限值取自暫存器】未勾選時，可自行設定上下限：

範圍上下限

X軸 上下限值取自暫存器

下限 : 0	上限 : 32767
--------	------------

Y軸 上下限值取自暫存器

下限 : 0	上限 : 32767
--------	------------

上下限是用於計算 X，Y 軸的刻度百分比。

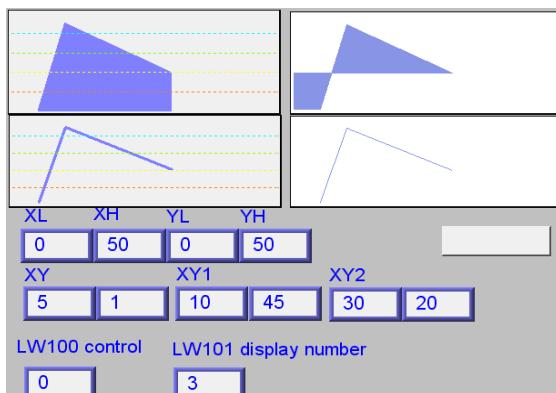
$$\text{刻度百分比} (\%) = \frac{\text{暫存器數據} - \text{下限}}{\text{上限} - \text{下限}}$$

【上下限取自暫存器】

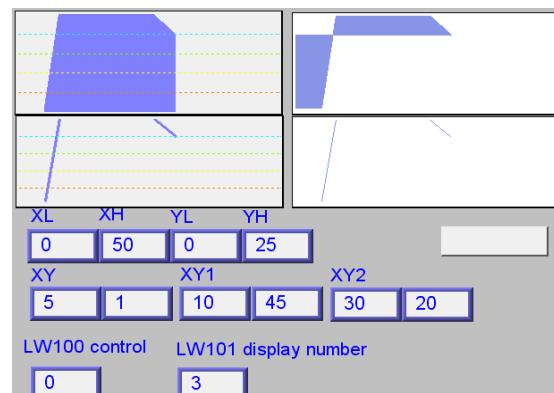
假設暫存器為 LW-100 (或以 n 表示) 且為 32 位元格式，則上下限及數據來源會根據以下方式設定：(停用 X 軸數據和 Y 軸數據來自不同位址)

內容	X 數據		Y 數據	
讀取來源	LW100 (n)			
下限	LW100	n+0	LW104	n+4
上限	LW102	n+2	LW106	n+6
數據 0	LW108	n+8	LW110	n+10
數據 1	LW112	n+12	LW114	n+14
數據 2	LW116	n+16	LW118	n+18

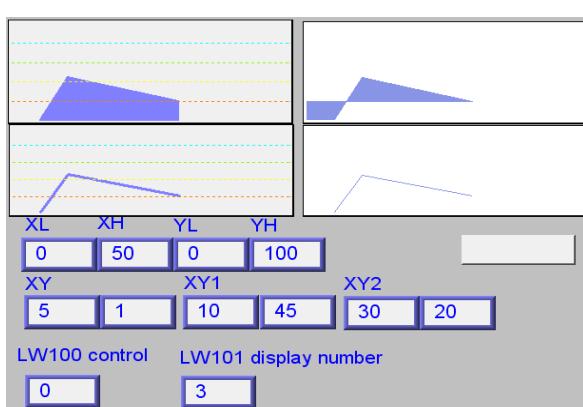
若勾選 【上下限值取自暫存器】，使用者可改變上下限來達到縮放效果。(請參閱趨勢圖物件)
如下範例，XL=X 下限，XH=X 上限，YL=Y 下限，YH=Y 上限，XY，XY1，XY2 為三個 XY 數據。此時改變 Y 軸的上限，即可觀察縮放效果。效果如下：



原圖



改變 Y 軸上限為 25 (放大效果)

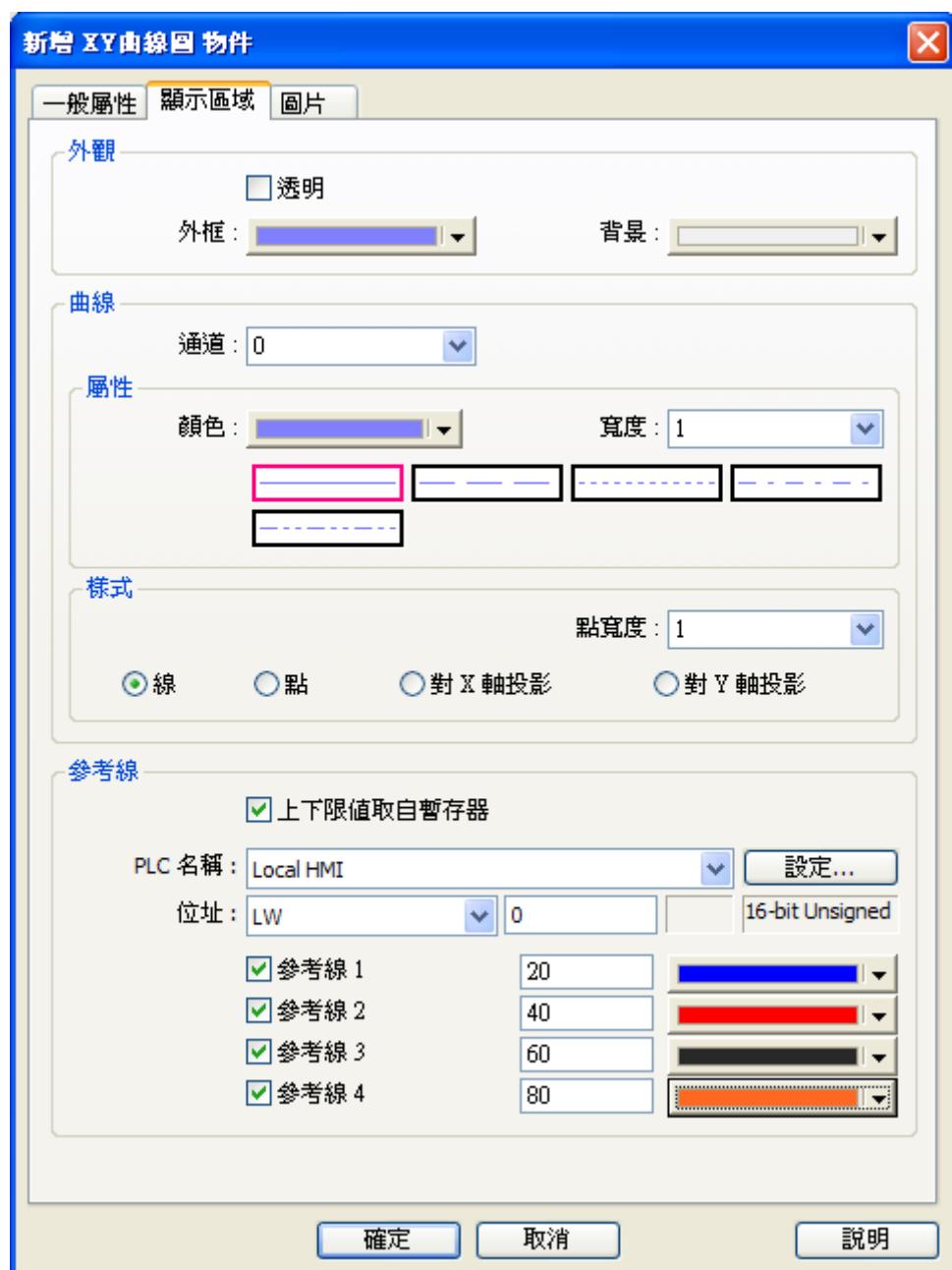


改變 Y 軸上限為 100 (縮小效果)



- X 和 Y 資料可使用不同格式，例如 X 資料使用 16-bit unsigned 而 Y 資料使用 32-bit signed，此時需特別留意位址的設置。
- 當 PLC 是 Tag PLC 時，例如 AB tag PLC，則 X 和 Y 一定要使用相同的位址格式。若選擇不同的格式會出現警示訊息。

[顯示區域]



外觀

勾選外觀時背景為透明，無勾選則依照所選擇的色彩來表現外框及背景。

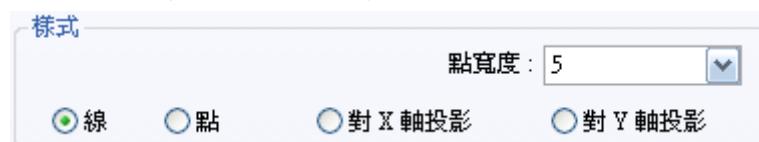
曲線

可在此設定通道所要顯示的屬性。

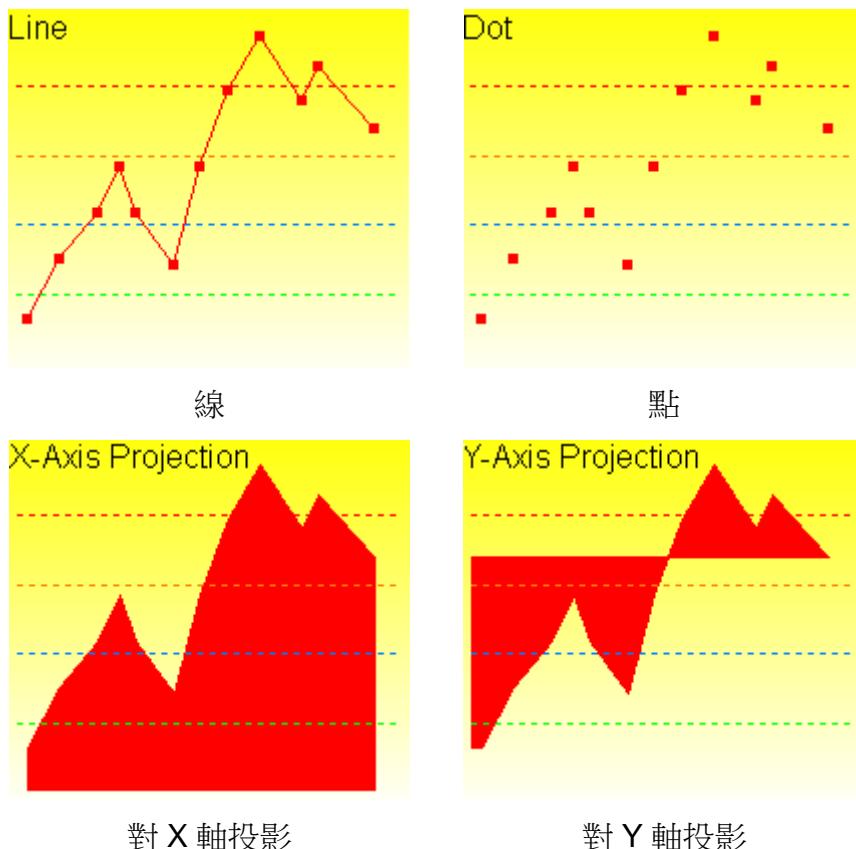


樣式

設定螢幕以線，點，對 X 軸投影或對 Y 軸投影顯示。



其對應的示意圖如下：

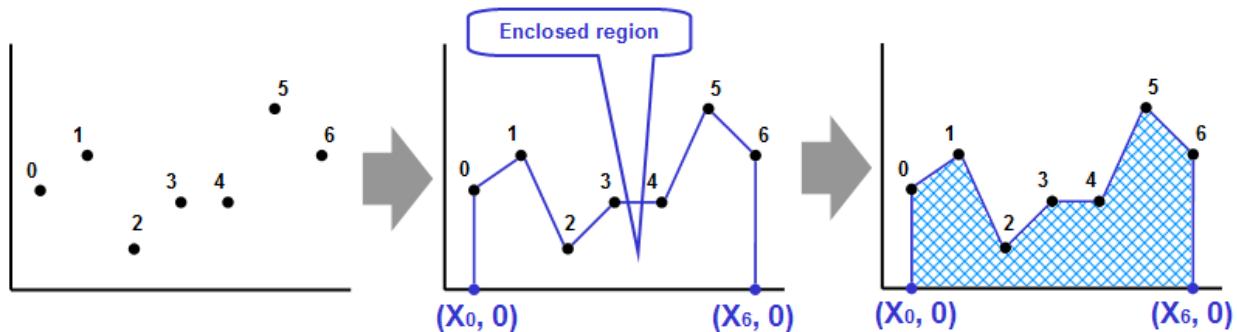


範例：

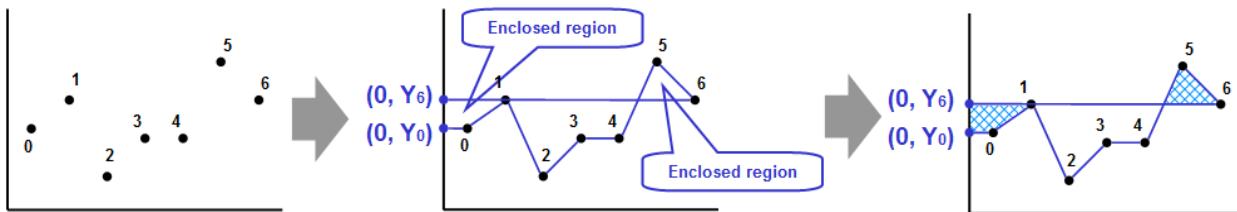
下圖中的曲線由 7 個點構成，由 P0 到 P6。

系統畫出 X 軸投影方式如以下步驟：

1. 自動計算出二個投影的點： X-axis – $(X_0, 0)$ 和 $(X_6, 0)$
2. 依照點出現的順序，連結所有的點 $(X_0, 0)$, P0, P1... P6, $(X_6, 0)$ 並且最後連結到第一個點 $(X_0, 0)$
3. 填滿封閉區域，結果如下：



同樣的對 Y 軸投影可得：



參考線

最多可畫四條參考線在曲線圖上，使用者可以自行選擇線條的色彩及參考的數值，並且依據所設定數值來顯示在螢幕上



若勾選上下限值取自暫存器，則需設定一個參考線之讀取位址。

參考線

上下限值取自暫存器

上下限

PLC 名稱 : Local HMI

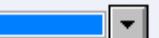
設備類型 : L W

地址 : 20

索引暫存器

16-bit Signed

參考線 1 20 

參考線 2 40 

參考線 3 60 

參考線 4 80 



- XY Plot 最多可以重覆畫 32 次。計算方式如下：
1 個通道可以重覆畫 32 次，若是有 2 個通道，則只能重覆畫 16 次。用 32 除以通道數可得最多重覆畫的次數。

13.21 報警條與報警顯示

概要

[報警條] 與 [報警顯示] 物件可以用來顯示已被定義在 [事件登錄] 中，且系統目前狀態滿足觸發條件的事件，此時這些事件也被稱為警示。[報警條] 與 [報警顯示] 物件將利用事件被觸發的時間先後，依序顯示這些警示，其中 [報警條] 物件使用單行跑馬燈型式呈現警示內容；[報警顯示] 物件則可同時顯示多行警示內容。下圖顯示不同物件對警示的表示方式。有關事件登錄的說明可以參考相關章節。

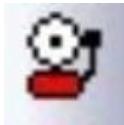
I (When LW 1 >= 10) 13:21:06 Event 0 (when LW0)

[報警條] 物件，單行顯示多個事件

13/12/06	13:21:38	Event 2 (when LB10 = ON)
13/12/06	13:21:38	Event 3 (when LB11 = ON)
13/12/06	13:21:38	Event 0 (when LW0 == 100)
13/12/06	13:21:38	Event 1 (When LW 1 >= 10)

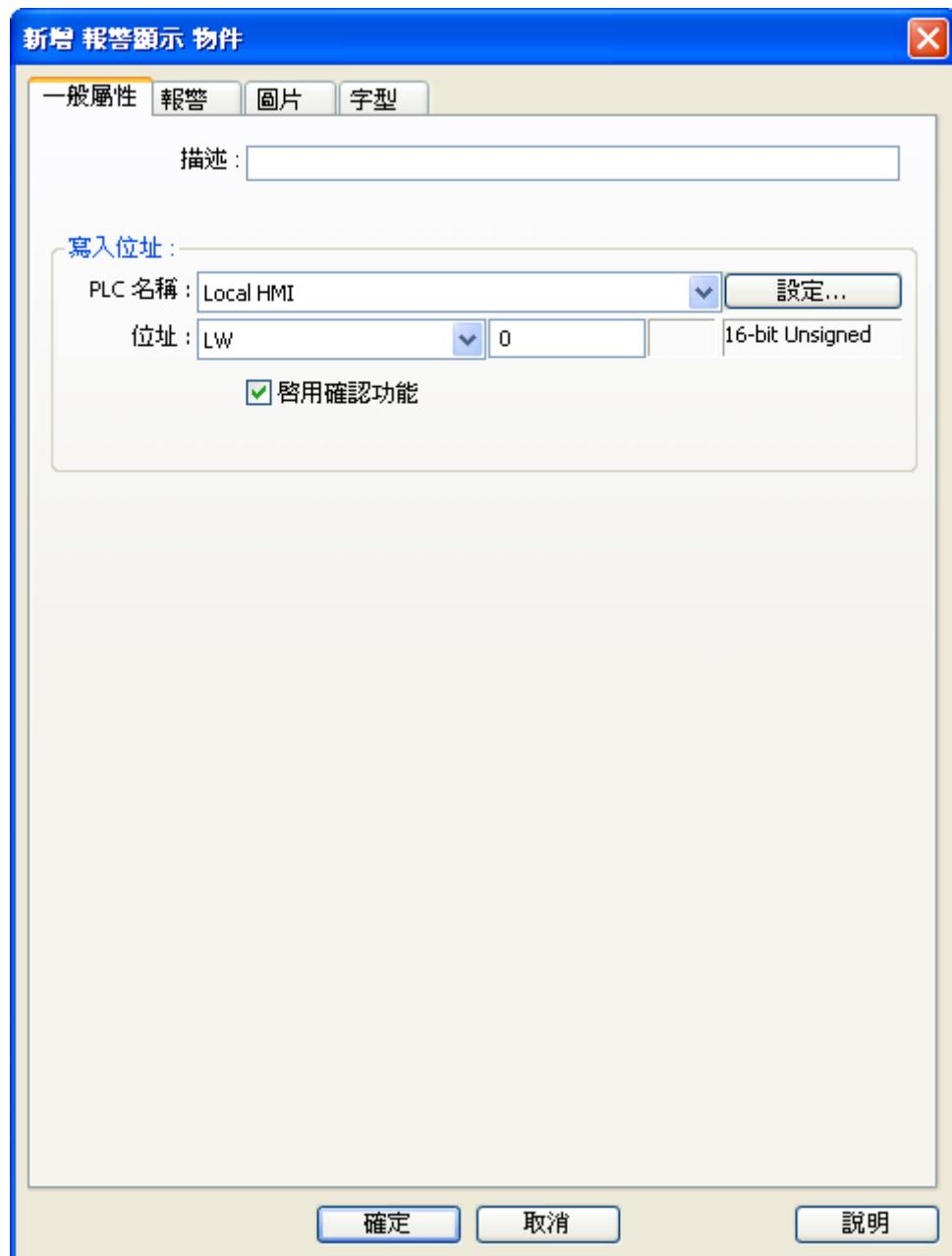
[報警顯示] 物件，可顯示多行

設定



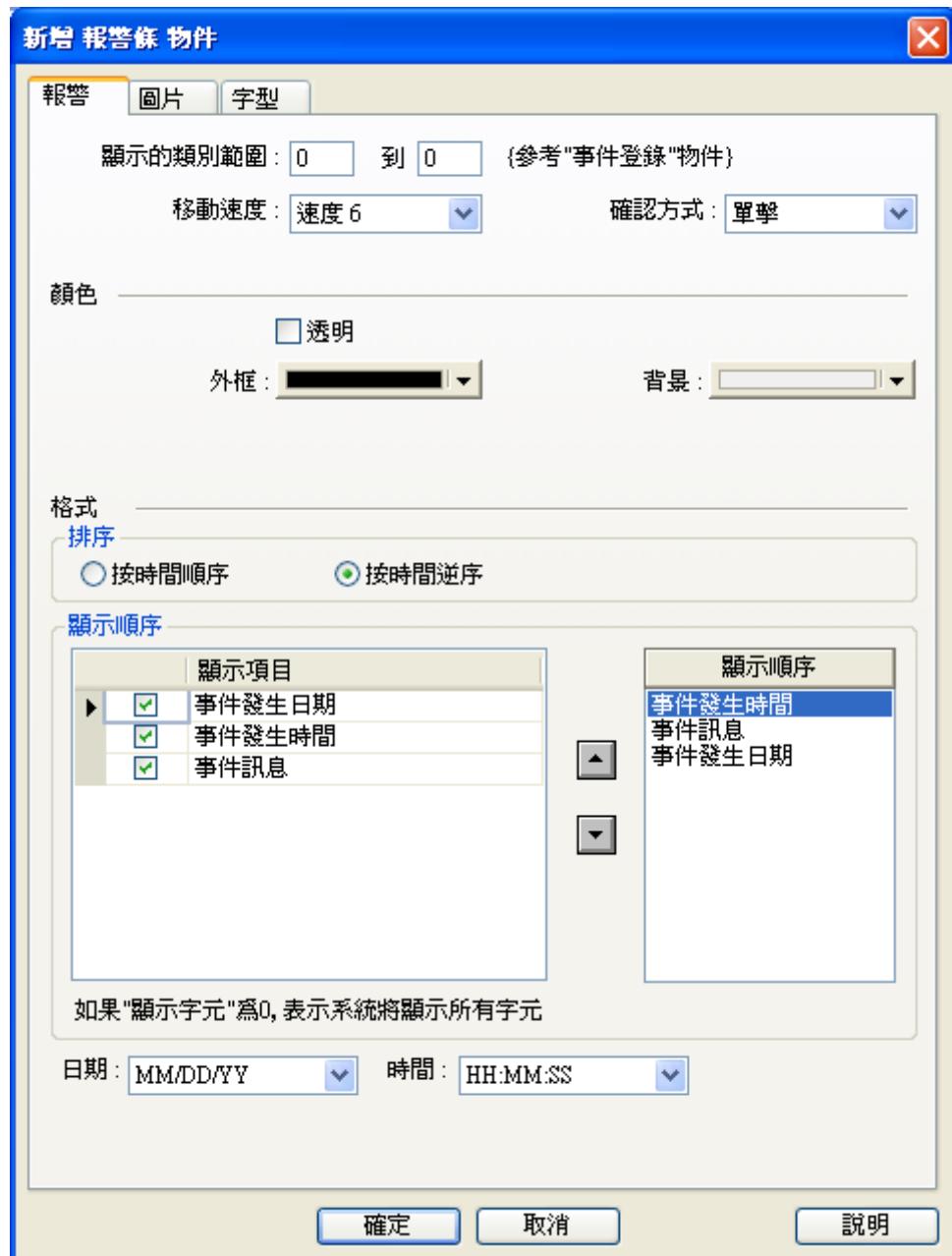
按下工具列上的 [報警條] 按鈕後，即會出現物件屬性對話窗；相同方式，按下工具列上的 [報警顯示] 按鈕後，即會出現物件屬性對話窗，正確設定各項屬性後按下確定鍵，即可新增一個物件。

兩者設定幾乎相同，唯一不同之是 [報警顯示] 較 [報警條] 多了一個 [寫入位址] 的設定功能，如下圖：



若點選【啟用確認功能】將會把在【事件登錄】中所指定的【在事件確認時寫入】之值寫入【寫入位址】。

接著說明兩物件之共同設定：



顯示的類別範圍

被觸發事件的 [類別] 需符合此處設定的顯示範圍才會被顯示(事件的類別在 [事件登錄] 中設定)。例如當 [報警條] 物件的 [類別] 此時被設定為 2~4，則僅有 [類別] 為 2 或 3 或 4 的事件，才會被顯示在報警條物件中。詳細說明請參考 [事件登錄] 說明中 [類別]。

移動速度 (僅 [報警條] 可設定)

設定 [報警條] 物件中所顯示文字的移動速度。

[排序]

設定警報顯示的順序，可以選擇 [按時間順序] 或 [按時間逆序]。

[按時間順序]

較晚發生的警報被排列在後(或下)。

[按時間逆序]

較晚發生的警示被排列在前(或上)。

[顯示順序]

可使用上下箭頭調整所要顯示的資訊及排列方式。

[日期]

選擇 [事件發生日期] 的日期格式，共有 4 種模式：

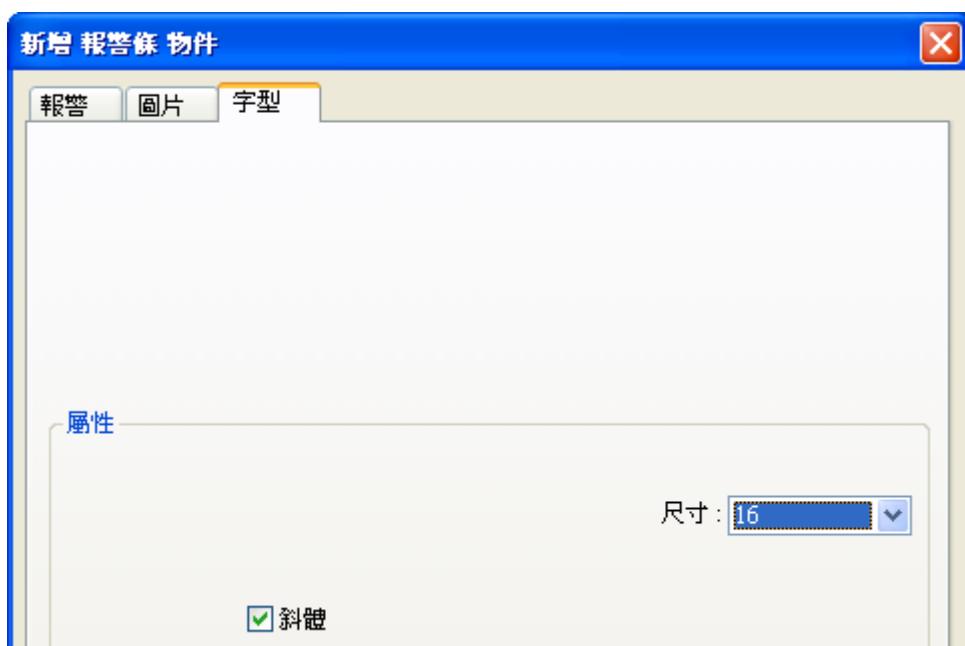
MM/DD/YY、DD/MM/YY、DD/MM/YY、YY/MM/DD

[時間]

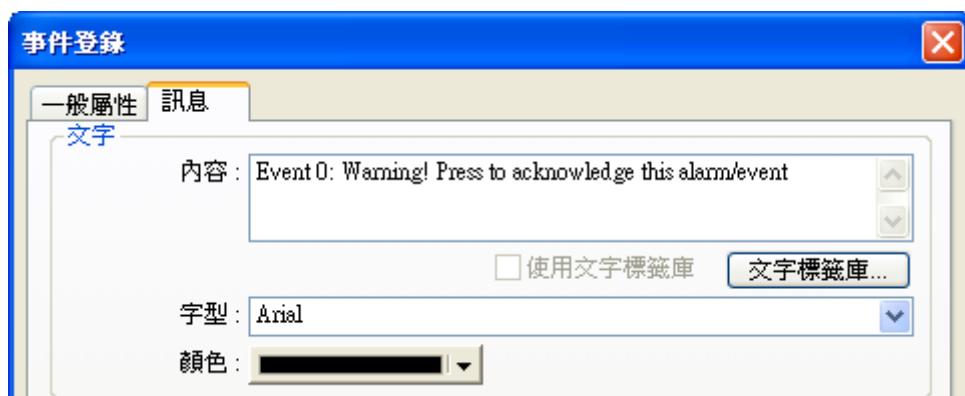
選擇 [事件發生時間] 的時間格式，共有以下 4 種模式：

HH:MM:SS、HH:MM、DD:HH:MM、HH

可以使用 [字型] 分頁設定物件文字之尺寸與斜體效果，參考下圖。



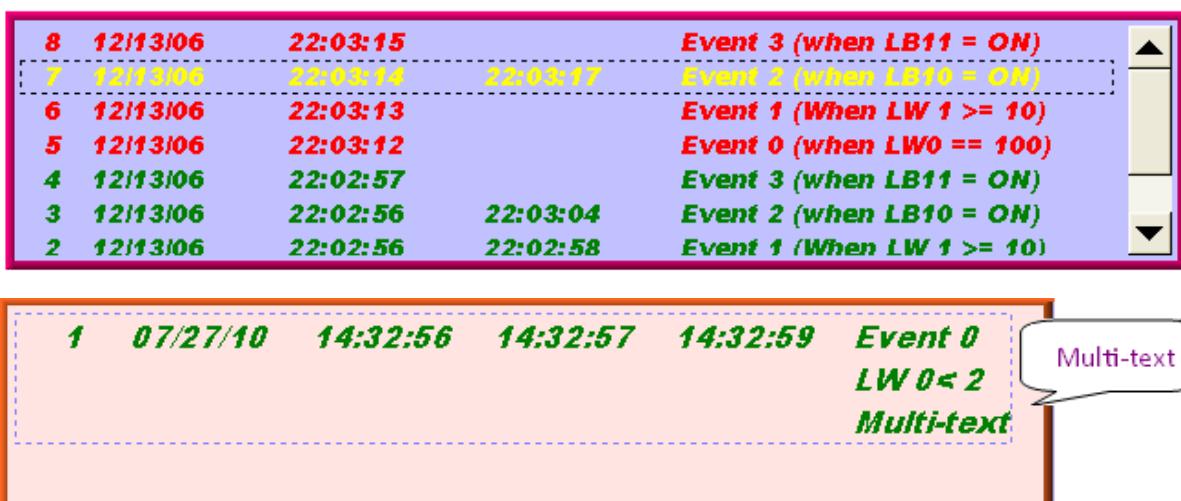
而各事件所使用的字型與顏色需在 [事件登錄] 中設定，如下圖：



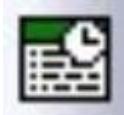
13.22 事件顯示

概要

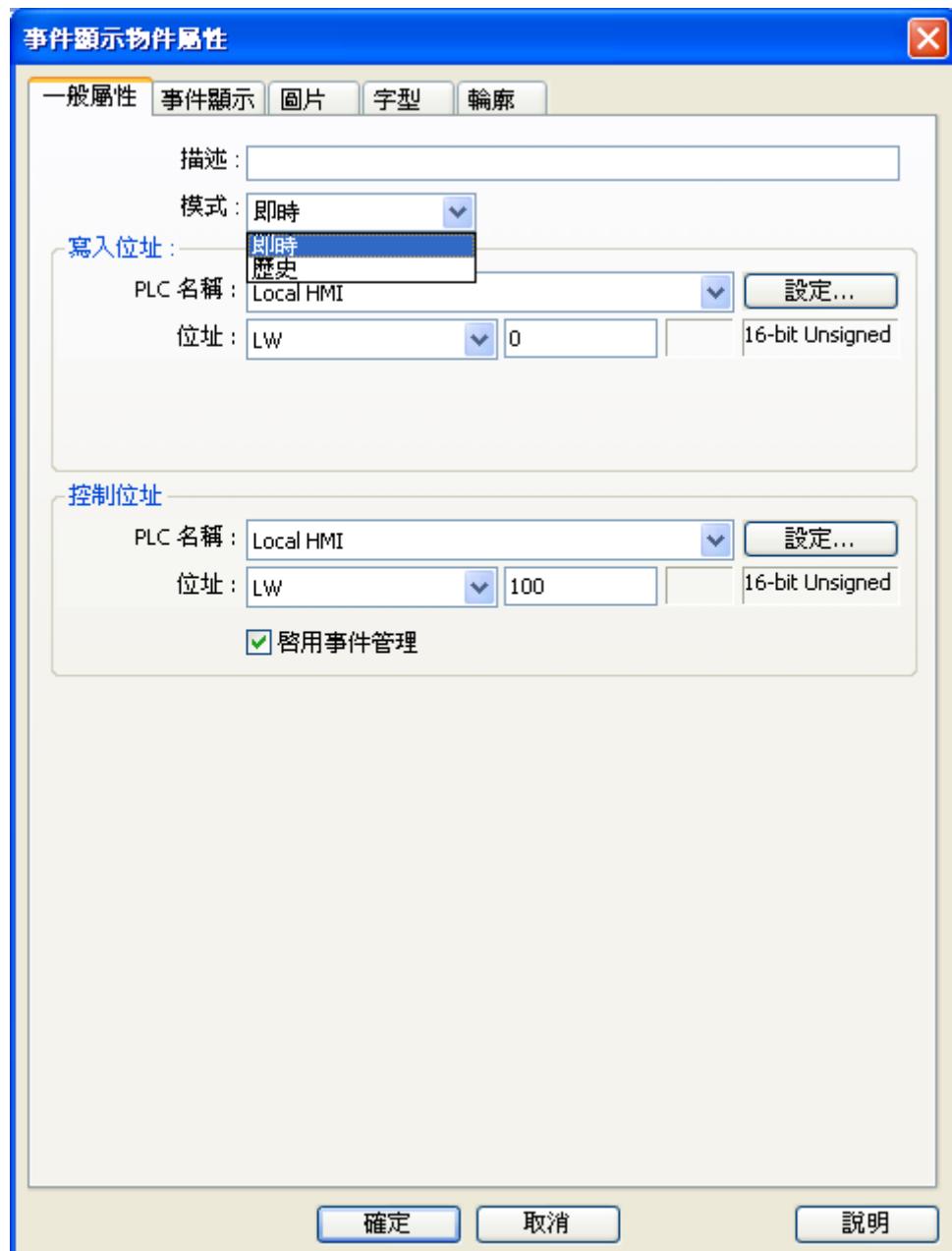
【事件顯示】物件可以用來顯示已被定義在【事件登錄】中，且曾經滿足觸發條件的事件，【事件顯示】物件將利用事件被觸發的時間先後，依序顯示這些事件，如下圖。【事件顯示】物件也可顯示事件被觸發、確認後與恢復正常狀態的時間。另外事件顯示物件支援多行文字顯示。



設定



按下工具列上的【事件顯示】按鈕後，即會出現【事件顯示】物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個【事件顯示】物件。



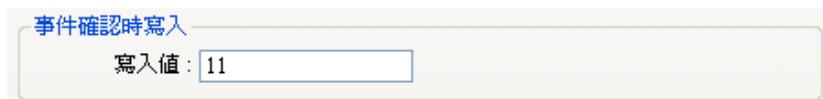
[模式]

選擇事件來源的形式，可以選擇 [即時] 或 [歷史]。

a. 即時

寫入位址

可顯示 [事件登錄] 中的事件從開機後到目前被觸發的事件。當事件被確認時，在 [事件登錄] 的訊息內的 [事件確認時寫入] 中的數值會被輸出到 [事件顯示物件] 的 [寫入位址]。請參考 [事件登錄] 章節。



啟用事件管理

啟用【事件管理功能】後，將特定的數值寫入暫存器 LW-n 及 LW-n+1 可對「事件顯示」物件給予不同的命令，命令條件及內容如下表：

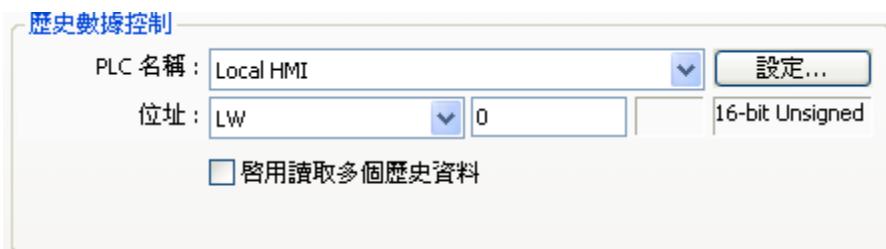
位址	數值	命令內容
LW-n	0	顯示所有事件
	1	隱藏「已確認」事件
	2	隱藏「已恢復」事件
	3	隱藏「已確認」或「已恢復」事件
	4	隱藏「已確認」與「已恢復」事件
LW-n+1	1	刪除選擇的單一事件

b. 歷史

● 無勾選【啟用讀取多個歷史資料】

在歷史模式下可顯示事件的歷史記錄，每一天的事件記錄被儲存在分開的檔案內。

下圖為設定歷史數據控制的指定位址。



系統透過索引來選擇歷史記錄 evt 檔：

輸入 0 則顯示最近一日的歷史資料

輸入 1 顯示第二近的一筆歷史資料

輸入 2 顯示第三近的一筆歷史資料

以此類推。

以下這個例子顯示如何使用歷史控制位址。假設在歷史數據控制位址是 LW-100，而目前歷史資料如下

- EL_20100720.evt
- EL_20100723.evt
- EL_20100727.evt
- EL_20100803.evt

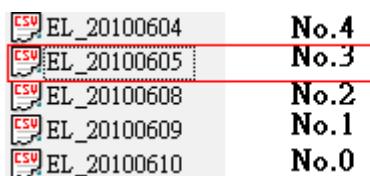
則當輸入數值到[LW100]時，所顯示的歷史資料如下表：

[LW100] 數值	相對應的歷史資料
0	EL_20100803.evt
1	EL_20100727.evt
2	EL_20100723.evt
3	EL_20100720.evt

- 勾選【啟用讀取多個歷史資料】

勾選後，可同時顯示多天歷史資料。假設【歷史數據控制】為 LW0，則 LW0 與 LW1 構成欲顯示之歷史資料範圍，LW0 的值代表開啟的第一筆歷史資料。

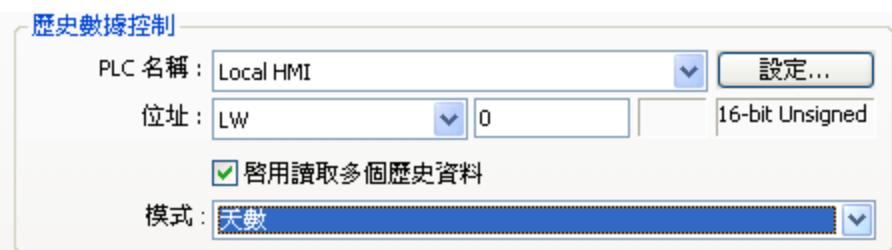
Ex: 如下圖，為了說明清楚，首先將歷史資料庫按日期優先作標記(No. 0、No. 1、No. 2...)，LW0 輸入數值為“3”，表示下圖歷史資料庫中標記 No. 3 的資料。



CSV EL_20100604	No.4	1 KB EVT 檔案
CSV EL_20100605	No.3	6 KB EVT 檔案
CSV EL_20100608	No.2	17 KB EVT 檔案
CSV EL_20100609	No.1	4 KB EVT 檔案
CSV EL_20100610	No.0	12 KB EVT 檔案

而 LW1 有以下模式：

1. 天數



歷史資料顯示範圍由 LW-0 標記開始算起，LW-1 的值表示往前推算幾天。

Ex: 如下圖，假設 LW-0 輸入數值為“1”，LW-1 輸入數值為“3”，則表示顯示的歷史資料範圍由 20100609 開始，往前推算三天(包括 20100609)，歷史資料庫中 20100607 資料不存在，所以顯示的歷史資料只有 20100609 與 20100608 等資料。



CSV EL_20100604	No.4	1 KB EVT 檔案
CSV EL_20100605	No.3	6 KB EVT 檔案
CSV EL_20100608	No.2	17 KB EVT 檔案
CSV EL_20100609	No.1	4 KB EVT 檔案
CSV EL_20100610	No.0	12 KB EVT 檔案

2. 最後歷史資料索引



歷史資料顯示範圍由 LW-0 標記開始算起，LW-1 的值表示資料標記結束。

Ex: 假設 LW-0 輸入數值為“1”，LW-1 輸入數值為“3”，顯示的歷史資料為下圖中 No. 1、No. 2、No. 3 的歷史資料。

EL_20100604	No.4	1 KB	EVT 檔案
EL_20100605	No.3	6 KB	EVT 檔案
EL_20100608	No.2	17 KB	EVT 檔案
EL_20100609	No.1	4 KB	EVT 檔案
EL_20100610	No.0	12 KB	EVT 檔案

系統最多可顯示 4MB 歷史資料，超出部份系統將略過。

以下顯示資料過大的例子。

Ex:

- 5 個歷史資料，每個 0.5 MB → 最多可顯示: $8 \times 0.5\text{MB}$
- 5 個歷史資料，每個 1 MB → 最多可顯示: $4 \times 1\text{MB}$
- 5 個歷史資料，每個 1.5 MB → 最多可顯示: $2 \times 1.5\text{MB} + 1 \times 1\text{MB}$ (部分)

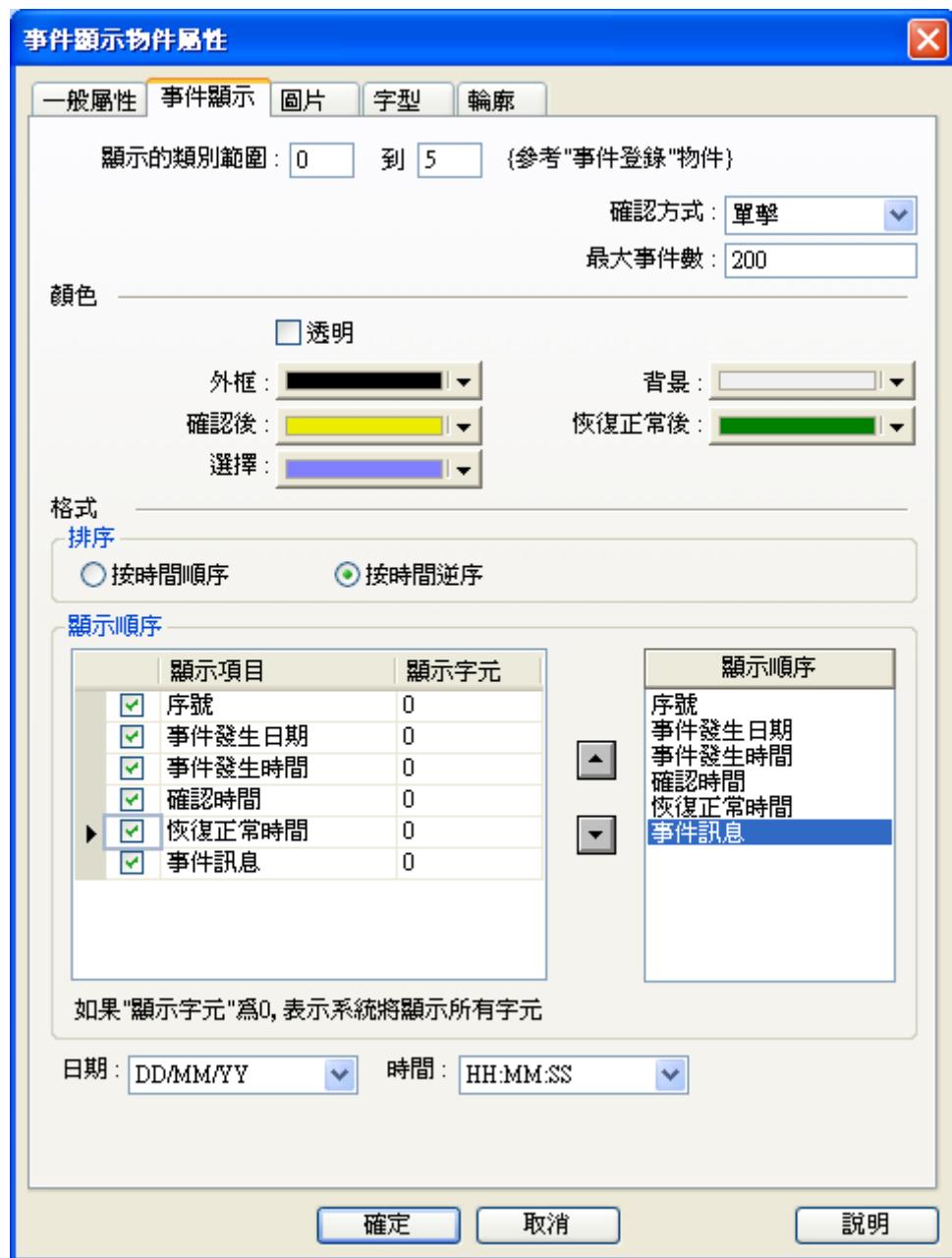
[事件管理功能]



啟用 [事件管理功能] 後，將特定的數值寫入暫存器 LW-n 及 LW-n+1 可對「事件顯示」物件給予不同的命令，命令條件及內容如下表：

位址	數值	命令內容
LW-n	0	顯示所有事件
	1	隱藏「已確認」事件
	2	隱藏「已恢復」事件
	3	隱藏「已確認」或「已恢復」事件
	4	隱藏「已確認」與「已恢復」事件
LW-n+1	1	刪除選擇的單一事件

事件顯示設定



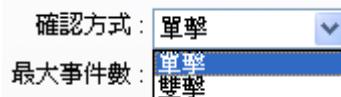
顯示的類別範圍

事件的 [類別] 需滿足此項設定範圍才會被顯示 (事件的 [類別] 在 [事件登錄] 中設定)。例如當 [事件顯示] 物件的 [類別] 此時被設定為 2 到 4，則僅有 [項目] 等於 2 或 3 或 4 的事件，才會被顯示在 [事件顯示] 物件中。可以參考 [事件登錄] 說明中有關 [類別] 的解釋。

確認方式

可選擇 [單擊] 或 [雙擊] 的確認方式。當事件發生時，使用者可依設定方式來做確認的動作。此處所謂“確認”的動作是指使用者對於已發生並顯示在 [事件顯示] 物件上的事件，此時除了

會將該事件的顯示顏色轉變為【確認】的顏色之外，也會將此事件預先設定的輸出值，寫至【寫入地址】中所設定的地址上。



以下二圖為例，當輸出地址為 LW-100，且事件確認時的寫入值為 31，則當使用者使用【確認】的動作時，LW-100 中的數據將被設定為 31，利用此項功能搭配【間接視窗】物件，可以讓不同的事件彈跳出不同的視窗，這些視窗通常被用來說明事件的內容。



最大事件數

物件所能顯示事件的最大數目。



- 當物件所顯示的事件已等於所設定的最大數目時，新發生的事件將取代已發生的事件中的第一筆資料。

顏色

設定事件在各種狀態下的顯示顏色。

- a. 確認後
- b. 恢復正常後
- c. 選擇事件 - 系統將會繪出虛線顯示剛被點選的事件

		確認	
序號	事件發生時間	事件訊息	點選事件
6	13:12:19	Event 1 (When LW 1 >= 10)	
5	13:12:18	Event 2 (when LB10 = ON)	
4	13:12:18	Event 3 (when LB11 = ON)	
3	13:12:15	Event 2 (when LB10 = ON)	
2	13:12:14	Event 1 (When LW 1 >= 10)	
1	13:12:14	Event 0 (when LW0 == 100)	

格式

序號	事件發生日期	事件發生時間	確認時間	恢復正常時間	事件訊息
0	12/14/06	15:26:21	15:26:31	15:26:36	Event 0 (when LW
1	12/14/06	15:26:47	15:26:50		Event 1 (When Lk
2	12/14/06	15:26:48			Event 2 (when LE

a. 排序

設定事件顯示的順序。

【按時間順序】

較晚發生的事件被排列在後(或在下)。

【按時間逆序】

較晚發生的事件被排列在前(或在上)。

b. 顯示順序

使用者可自行定義所要顯示的資訊及排列方式。

c. 日期【事件發生日期】模式

選擇顯示事件發生的日期的格式，共有以下 4 種模式

MM/DD/YY、DD/MM/YY、DD/MM/YY、YY/MM/DD

d. 時間【事件發生時間】模式

選擇顯示事件發生的日期，共有以下 4 種模式。

HH:MM:SS、HH:MM、DD:HH:MM、HH

各個事件所使用的字型請在【事件登錄】中設定。

13.23 觸發式資料傳輸

概要

[觸發式資料傳輸] 物件可以將指定地址中的數據傳送到其他地址中，可以使用手動按鈕的方式傳送數據，也可以利用特定地址的狀態改變，來觸發數據傳輸的動作。

設定



按下工具列上的**【觸發式資料傳輸】**按鈕後，即會出現**【觸發式資料傳輸】**物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個物件。



來源位址

設定被傳送數據的來源位址。

目標位址

設定數據傳送的目的位址。

屬性

[字元數量]

數據的傳送數量，單位為字元。

[模式]

可以選擇 [手動] 或 [觸發] 模式。

a. 手動模式

需使用手動的方式按下物件，才會進行資料傳送的動作。

b. 觸發模式

利用所指定暫存器狀態的改變來觸發資料傳送的動作，利用 [觸發模式] 選擇需要的觸發方式，這些觸發方式包括：

[ON → OFF]

當指定暫存器的狀態由 ON 變為 OFF，將執行資料傳輸動作。

[OFF → ON]

當指定暫存器的狀態由 OF 變為 ON，將執行資料傳輸動作。

[ON ← → OFF]

當指定暫存器的狀態改變，將執行資料傳輸動作。

觸發模式所使用的暫存器地址在 [觸發地址] 中設定，參考下圖。



13.24 備份

概要

利用備份物件可以將配方資料(RW, RW_A)、事件記錄或指定的資料取樣記錄複製到擴充裝置 (SD 卡或 USB 碟)，並可以指定備份的時間範圍。例如事件記錄原來儲存在 SD 卡，此時可以在不需關機的情形下插上 USB 碟，並利用備份物件複製一份相同的資料到 USB 碟，並在不需關機的情形下，直接拔取 USB 碟，這些數據即可移置 PC 做進一步的分析。當備份動作進行中時，[LB-9039]的狀態將維持在 ON。

設定



按下工作列上的【備份】按鈕後即會出現【備份】物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個【備份】物件。



來源

[RW]、[RW_A]、[事件記錄]、[資料取樣記錄]

這些選項用來選擇要複製的檔案來源，當選擇檔案來源為 [資料取樣記錄] 時，需使用**[資料取樣物件索引]** 選擇要複製哪一個資料取樣所產生的取樣記錄。

備份位置項目

設定要覆製來源檔案到那一個備份位置。

a. SD 卡或 USB 碟

外部裝置需事先連接在人機上。

b. 遠端列印/備份伺服器

若使用此選項，使用者需先在 **【系統參數的列印】>>【備份伺服器】** 中做設定

儲存格式

使用者可選擇想要儲存的格式。

a. HMI 事件記錄檔 (*.evt) / HMI 資料記錄檔 (*.dtl)

b. Comma Separated Values (*.csv)

➤ 事件記錄檔存成 csv 檔



➤ 資料記錄檔存成 csv 檔



當備份事件記錄檔為 CSV 格式時，可選擇在檔案開頭加入 BOM (Byte order mark)，讓 Excel 可以直接開啟包含非 ASCII 內容的.csv 檔案。



另外，事件記錄的備份檔案中有事件類別欄位如下圖所示。

- 0 -> 事件觸發
- 1 -> 事件確認
- 2 -> 事件恢復正常

	A	B	C	D	E
1	[Creation time]				
2	Fri Oct 29 10:59:28 2010				
3	[Data fields]				
4	event	category	time	message	
5	[Data]				
6	0	0	11:19:42	"Emergency"	
7	0	5	11:19:43	"5"	
8	2	0	11:19:46	"LOW"	
9	2	5	11:19:49	"5"	
10	1	0	11:19:52	"Word"	
11	2	0	11:19:52	"Word"	

範圍

[超始時間] 由今天或昨天

[幾天內]

設定儲存時間點。

例如，設定昨天並選擇 2 天。表示儲存昨天及前天的資料。

選擇全部，可儲存最多 90 天的資料。



觸發

選擇物件的執行方式，可以選擇 [手動] 與 [觸發] 兩種模式

a. 手動

使用者只需按壓物件，即可執行檔案複製動作。

b. 觸發 (位元)

當指定的暫存器狀態改變符合觸發條件時，物件將執行檔案複製動作。觸發條件包含下列幾種方式：

[ON → OFF]

當指定暫存器的狀態由 ON 變為 OFF，將執行檔案複製動作。

[OFF → ON]

當指定暫存器的狀態由 OFF 變為 ON，將執行檔案複製動作。

[ON ↔ OFF]

當指定暫存器的狀態改變，將執行檔案複製動作。

觸發位址

當使用 [觸發] 模式時，觸發地址被用來指定物件將使用哪一個暫存器來觸發檔案複製動作。



c. 觸發 (字元)

選擇使用【觸發 (字元)】模式時，用戶可以透過【觸發位址】設定所需備份資料的時間範圍。



【觸發位址】的用法如下(假使目前觸發地址被設定為 LW-0)：

LW-0：當此位址所指定暫存器中的由 0 變為 1，將觸發備份動作。

LW-1：此位址所指定暫存器中的數據用來指定備份的起始時間。

LW-2：此位址所指定暫存器中的數據用來指定備份的天數 (最多 90 天)。



1. 請注意所有的資料必須要被儲存在任意一記憶體之中(如：HMI memory 或 USB 碟或 SD 卡)，否則無法使用備份功能。
2. 單次備份最大天數為 90 天。

13.25 媒體播放器

第一次使用媒體播放器時，必需要使用 Ethernet 下載工程檔案到人機。EasyBuilder 會自動安裝媒體播放器的驅動。

概要

媒體播放器功能不只是播放影片，另外也提供額外操作功能例如搜尋，放大縮小，音量調整等作用。以動態影片來指示作業與維修保養，建立更容易瞭解且任何現場作業人員都能進行維修保養的環境。

注意：只有 MT8000X 系列支援此功能。

設定



按下工作列上的【媒體播放器】按鈕後即會開啟【媒體播放器】物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個【媒體播放器】物件。

一般屬性設定頁



設定		敘述
暫存 器控 制	啟用控制位址	<ul style="list-style-type: none"> ● 啓用時 <ul style="list-style-type: none"> a. 使用者可針對【媒體播放器】進行控制並且得到播放資訊。 b. 必須指定一位址用來控制物件行為。 ● 取消使用暫存器控制時 此設定無法手動控制影片的播放狀態。在視窗開啟時，系統會自動播放影片
	命令	控制【媒體播放器】的動作模式。 ➤ 命令 (控制位址 + 0)
	參數 1	相對於特定命令所傳入的參數 1。 ➤ 參數 1 (控制位址 + 1)
	參數 2	相對於特定命令所傳入的參數 2。 ➤ 參數 2 (控制位址 + 2)

	狀態	記錄檔案狀態、播放情況及錯誤代碼。 ➤ 狀態 (控制位址 + 3)
	檔案索引	播放的檔案位於指定目錄下的索引 (以檔名排序, 建議以數字為起始檔名)。 ➤ 檔案索引 (控制位址 + 4)
	開始時間	影片開始時間 (秒)。(通常為 0) ➤ 開始時間 (控制位址 + 5)
	結束時間	影片結束時間 (秒)。(影片的時間長度) ➤ 結束時間 (控制位址 + 6)
播放時間	更新影片播放時間	• 啓用時 HMI 每隔 [更新週期] (秒) 會將影片已播放時間寫入 [播放時間] 暫存器中。
	更新週期	[播放時間] 的更新週期，範圍由 1 至 60 秒。
	播放時間	影片已播放時間 (秒) (介於 [開始時間] 與 [結束時間] 之間)。 ➤ 已播放時間(控制位址 + 7)
影片存放位置	SD	選擇播放 SD 裡的檔案。
	USB 1/ USB 2	選擇播放 USB 1 或 USB 2 裡的檔案。
	資料夾名稱	影片檔案放置的資料夾名稱。檔案必須被放置於資料夾中且資料夾只能為一層, 多層資料夾將不會被接受 (例如指定 [資料夾名稱] 為 "video\ex" 將會出現錯誤)。 
屬性	自動重複	當所有的影片播放結束, 會自動跳回第一個影片從頭播放。 例如 : Video 1 > Video 2 > Video 1 > Video 2
	背景顏色	指定物件背景顏色。



預設的暫存器格式為 **16 位元無正負號**; 當指定之暫存器為 32 位元時, 只有較低的 16 位元產生作用, 並請將較高的 16 位元固定為 0。

控制命令：

a. 播放索引檔案

[命令] = 1

[參數 1] = 檔案索引

[參數 2] = 忽略(應設為 0)



- 檔案以檔名排序。
- 假如找不到檔案，則將 [狀態] 的位元 8 設為 ON。
- 若需中途切換影片，請先將正在播放的影片停止。

b. 播放上一個檔案

[命令] = 2

[參數 1] = 忽略(應設為 0)

[參數 2] = 忽略(應設為 0)



- 若 [檔案索引] 為 0，則輸入命令 2 會從頭播放原檔案。
- 假如找不到檔案，則將 [狀態] 的位元 8 會被設為 ON，代表命令錯誤。

c. 播放下一個檔案

[命令] = 3

[參數 1] = 忽略(應設為 0)

[參數 2] = 忽略(應設為 0)



- 如果找不到檔案，則播放索引值 0 的檔案。
- 假如找不到檔案，則將 [狀態] 的位元 8 會被設為 ON，代表命令錯誤。

d. 暫停/播放 切換

[命令] = 4

[參數 1] = 忽略(應設為 0)

[參數 2] = 忽略(應設為 0)

e. 停止播放並關閉檔案

[命令] = 5

[參數 1] = 忽略 (應設為 0)

[參數 2] = 忽略 (應設為 0)

f. 從指定位置開始播放

[命令] = 6

[參數 1] = 目標時間 (以秒為單位)

[參數 2] = 忽略 (應設為 0)



- 參數 1 (目標時間) 應小於結束時間，若超出結束時間則由結束時間前 1 秒開始播放。

g. 往後跳躍 (秒)

[命令] = 7

[參數 1] = 目標時間 (以秒為單位)

[參數 2] = 忽略 (應設為 0)



- 從目前時間往後跳躍 [參數 1] 指定秒數後開始播放。若系統目前為暫停播放影片狀態，則跳躍動作會在開始播放後才會進行。
- 若播放時間超過結束時間，則由結束時間前 1 秒開始播放。

h. 往前跳躍 (秒)

[命令] = 8

[參數 1] = 目標時間 (以秒為單位)

[參數 2] = 忽略 (應設為 0)



- 從目前時間往前跳躍 [參數 1] 指定秒數後開始播放。若系統目前為暫停播放影片狀態，則跳躍動作會在開始播放後才會進行。
- 若播放時間少於開始時間，系統會從頭播放影片。

i. 設定音量

[命令] = 9

[參數 1] = 音量 (0 ~ 128)

[參數 2] = 忽略(應設為 0)



- 預設為最大音量 (128)。

j. 設定影像放大倍率

[命令] = 10

[參數 1] = 影像大小 (0 ~ 16)

[參數 2] = 忽略(應設為 0)



- [參數 1 = 0]：配合物件大小。
- [參數 1 = 1 ~ 16]：放大倍率範圍 25% ~ 400%，設定 1 為放大 25%，2 為 50%，3 為 75% 以此類推。

k. 狀態(控制位址 + 3)

當 HMI 正在播放影片，則 [檔案開啓位元] 位元 00 及 [檔案播放位元] 位元 01 將被同時設為 ON (0 → 1)。相反地，若找不到檔案或輸入的命令不正確，則 [錯誤命令位元] 位元 08 將會被設為 ON (0 → 1)。若在播放過程中發現檔案格式錯誤或任何磁碟 I/O 錯誤(例：USB 磁碟被拔除)，則 [檔案錯誤位元] 位元 09 將被設為 ON(0→1)。

15	09 08	02 01 00	位元
保留 (0 固定)	0 0		0 0

位元 00： 檔案開啓位元(0：未開啓檔案；1：已開啓檔案)
 位元 01： 檔案播放位元(0：未播放檔案；1：檔案播放中)
 位元 08： 錯誤命令位元(0：正確命令格式；1：錯誤命令或參數)
 位元 09： 檔案錯誤位元(0：檔案格式及讀取正確；1：檔案格式或讀取錯誤)

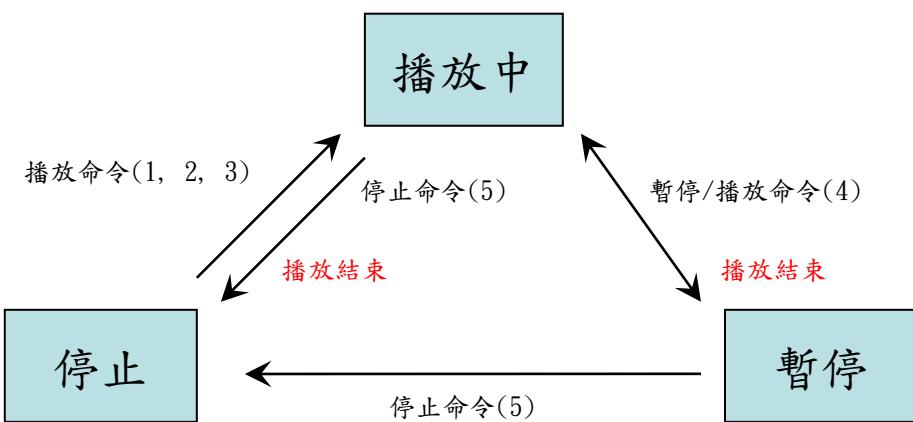


- 參考下圖 [媒體播放器] 之狀態轉換圖可知：

“停止”時狀態 = 0

“暫停”時狀態 = 1

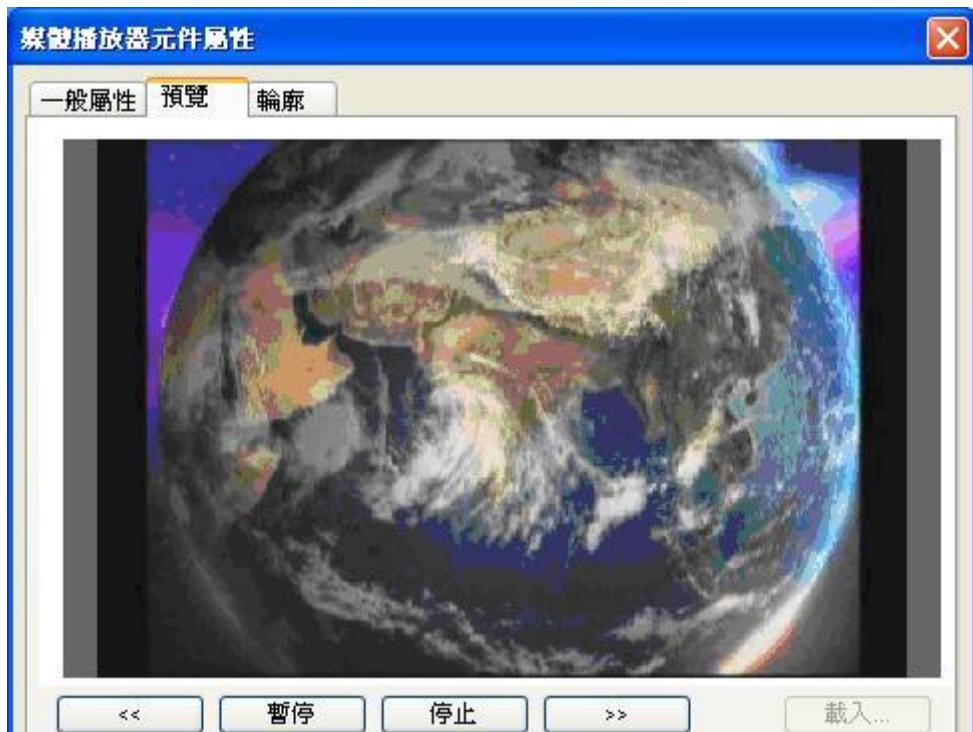
“播放”時狀態 = 3



- 請透過設定 [命令]，[參數 1] & [參數 2] 來操作物件，並將其它位址視為唯讀。

預覽設定頁

使用者可利用預覽功能來檢查 HMI 是否支援欲播放的影片格式。



[載入]

選擇要預覽的影片。

[開始 / 暫停]

播放

暫停

可選擇影片開始播放或暫停播放。

[前進 / 後退]

<<

>>

往前或往後快轉 (以 1 分鐘為單位)。

[停止]**停止**

停止播放影片並關閉檔案。若需測試另一影片，必須先停止播放目前影片。



1. 使用者需注意 HMI 上同一時間只能有一個影片檔被開啟。
2. 假如使用者沒有啓用 [控制位址] 且沒有設定 [自動重複]，則指定目錄下的第一個檔案播完一遍後，系統會自行將影片關閉。
3. 當沒有啓用 [控制位址] 時，物件生成後自動到指定目錄下找尋第一個檔案 (以檔名排序) 開始播放。
4. 當影片可以使用媒體播放器的預覽功能，表示人機支援此影片格式並且可以播放。若是在人機上有播放品質不佳的狀況請調整影片的解析度。
5. 支援檔案格式有: mpeg4, xvid, flv...等等。

13.26 定時式資料傳輸

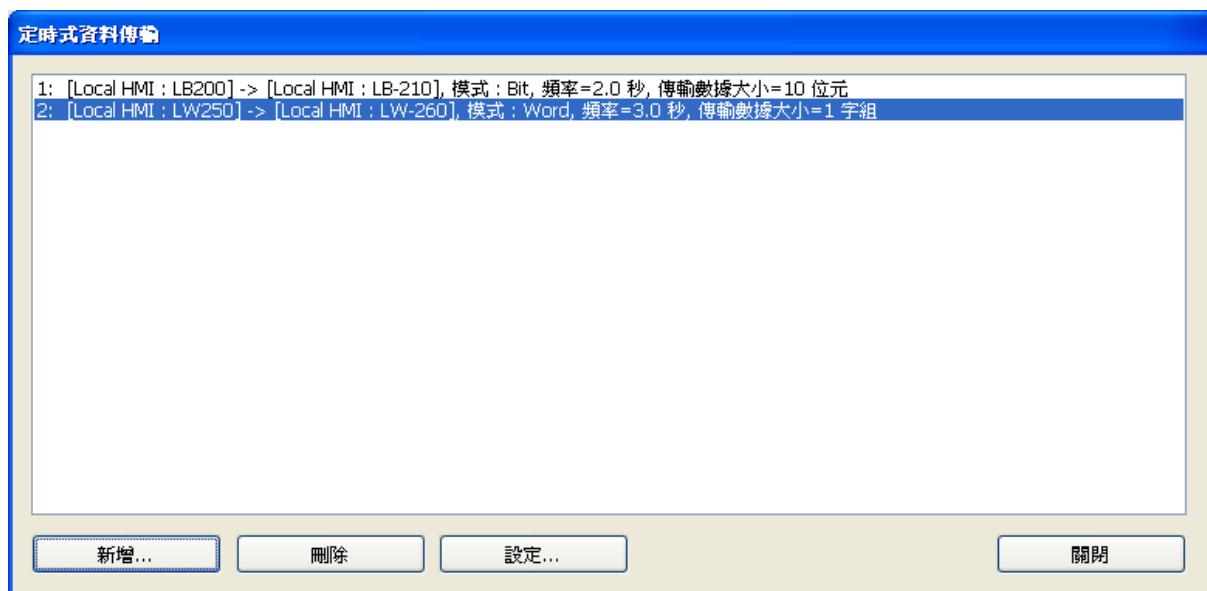
概要

【定時式資料傳輸】物件與【觸發式資料傳輸】物件相同，皆用來將指定地址中的數據傳送到其他地址中。與【觸發式資料傳輸】物件不同的是【定時式資料傳輸】物件使用固定的頻率、自動執行數據傳送的工作，並且可以傳送使用位元為計數單位的數據。

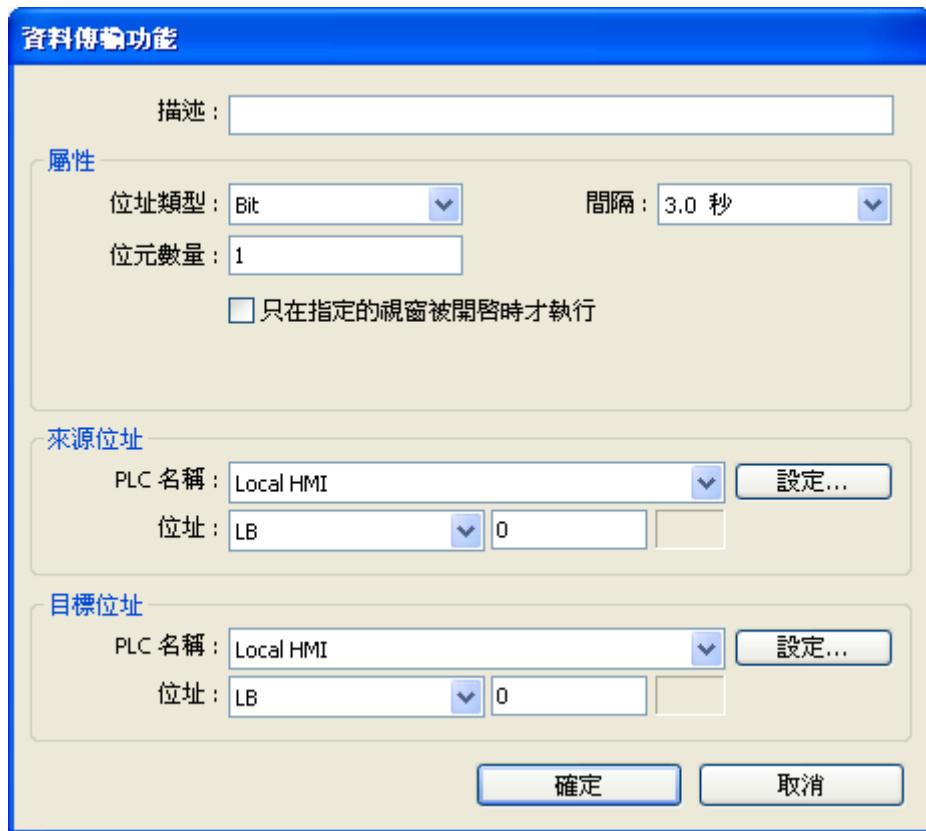
設定



要使用【定時式資料傳輸】物件首先需按下工具列上的  【定時式資料傳輸】按鈕打開【定時式資料傳輸】管理對話窗。



按下 [新增] 按鈕，即可打開 [資料傳輸功能] 設定對話窗，正確設定各項屬性後，即可新增一個 [定時式資料傳輸] 物件，參考下圖：



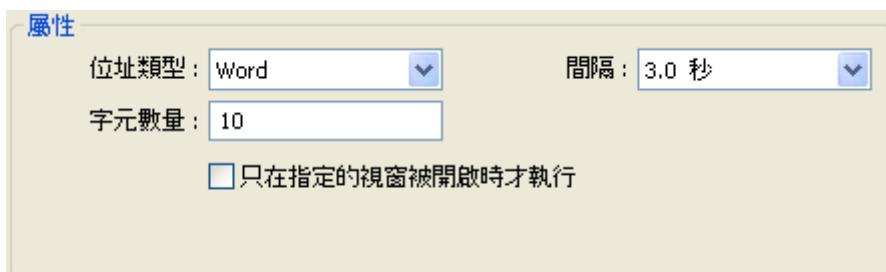
屬性

[位址類型]

選擇被傳送數據的類型，可以選擇位元型態或字元型態的數據。

[位數]或[字數]

當 [位址類型] 選擇 [Word] 型態時，數據傳送單位為 word，使用 [字元數量] 設定傳送數量，參考下圖。



當 [位址類型] 選擇 [Bit] 型態時，數據傳送單位為 bit，使用 [位元數量] 設定傳送數量。



[間隔]

選擇數據傳送頻率，例如選擇 3 秒，則每隔 3 秒，將傳送數據到指定的地址中。

備註：

1. 較小的間隔或是大量的資料傳輸可能會導致系統執行速度變慢，建議使用者拉長傳送的間隔或是一次傳送小量的資料，以避免系統執行速度變慢。
2. 當需要設定短時間的傳輸時，請注意設定間隔的時間要大於資料傳輸的時間。例如：數據傳輸操作需要 2 秒，則間隔時間須設置超過 2 秒。

資料來源地址項目

設定數據的傳送來源。

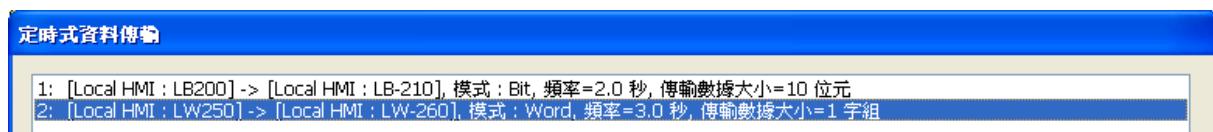
點選 [設定]，設定來源地址的 PLC 名稱和位址。

目標地址項目

設定數據傳送的目的地址。

點選 [設定]，設定來源地址的 PLC 名稱，位址。

在完成各項設定並按下 [確定] 鍵後，即可新增一個【定時式資料傳輸】物件，由【定時式資料傳輸】管理對話窗中可看出此新物件的內容。



13.27 PLC 控制

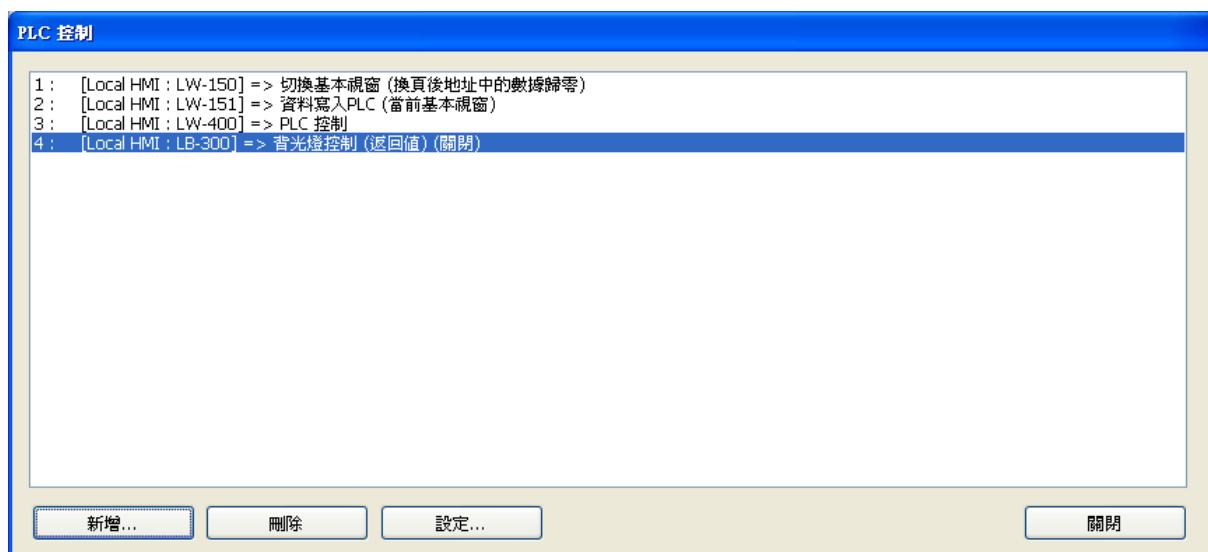
概要

當相應的控制命令被觸發時，[PLC 控制] 物件能啟動某個特定的動作。

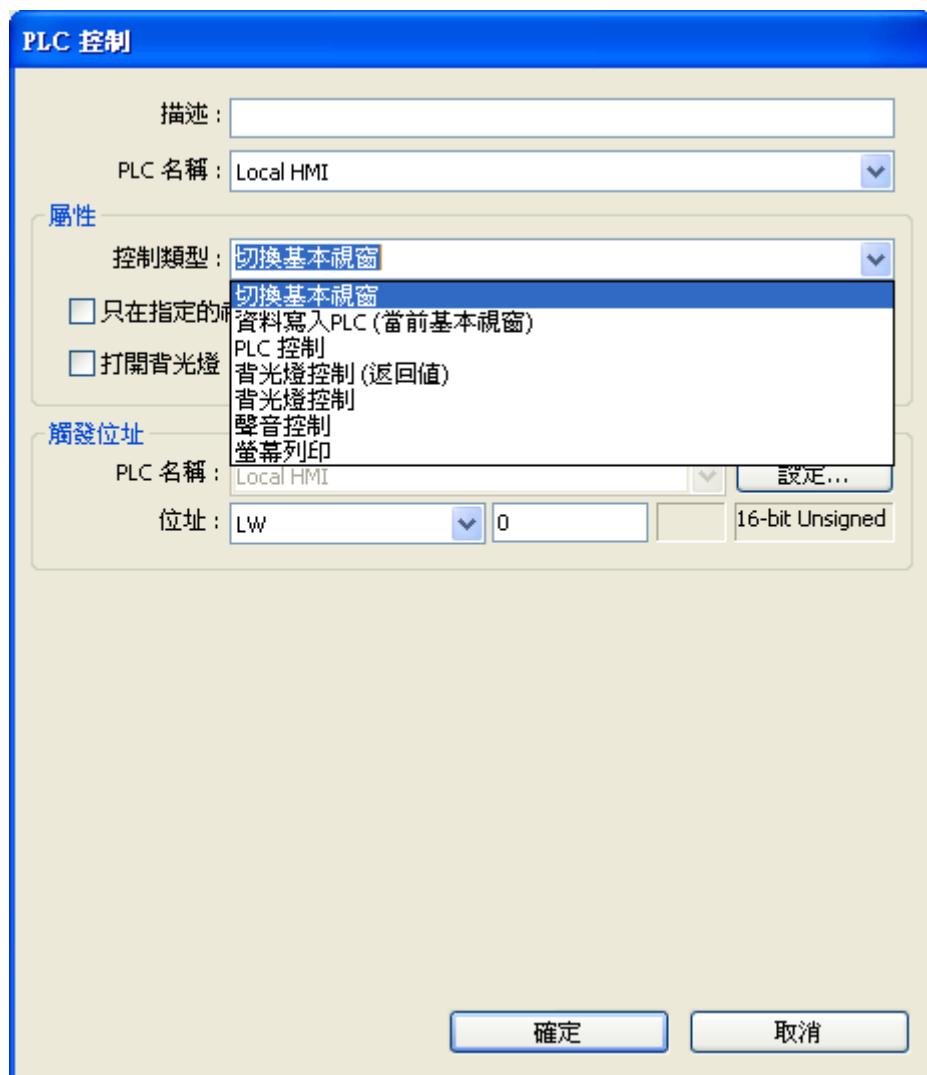
設定



按下工具列上的 [PLC 控制] 按鈕後即會出現 [PLC 控制] 物件管理對話窗，接著可按下 [新增] 按鍵，並利用出現的 [PLC 控制] 物件設定對話窗正確設定物件的各項屬性，最後按下確認鍵即可新增一個 [PLC 控制] 物件。



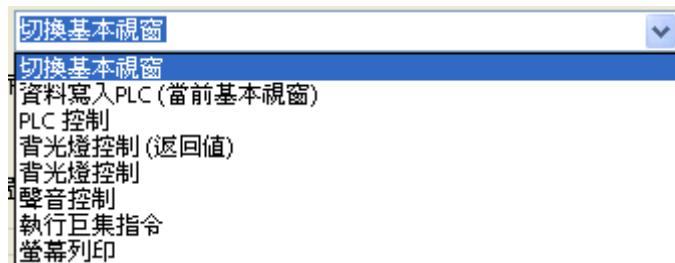
下圖為按下 [新增] 按鍵後所出現的設定對話窗。



屬性及控制位址

[控制類型]

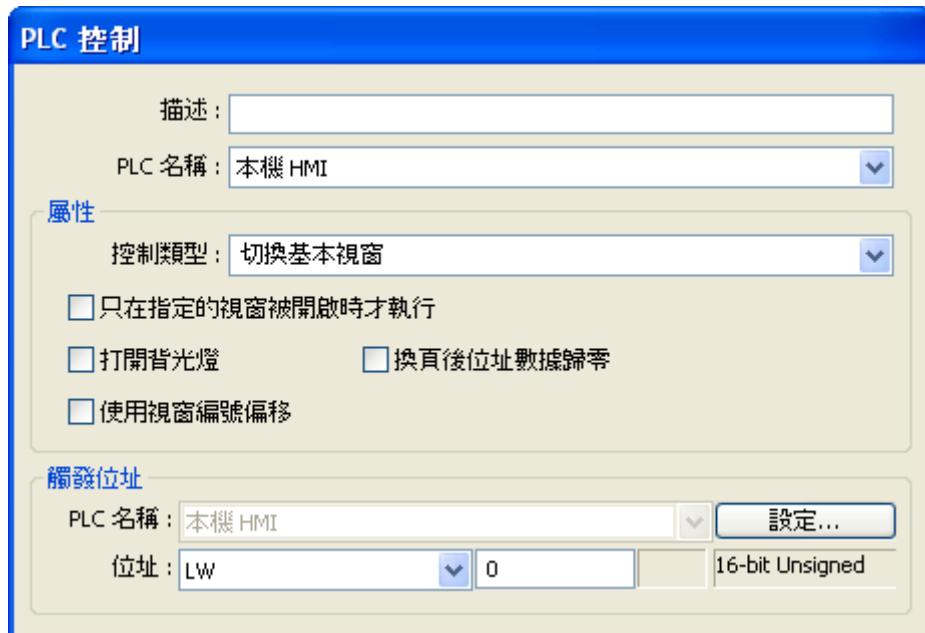
選擇 PLC 控制物件的控制類型，可選擇項目如下：



a. 切換基本視窗

切換基本視窗的功能。當 [觸發地址] 中的數據改變，且改變後的數據為一個有效的視窗編號時，將關閉目前的視窗並切換至 [觸發地址] 中數據所指定的視窗，並將此時切換後的視窗編號寫至指定的地址中，請見以下範例。

例如：目前的視窗編號為 10，且物件的設定內容如下圖：



當 LW - 0 中的數據由其他數據改變為 11 時，EasyBuilder 除了會將基本視窗切換到視窗 11 之外，也會將 LW - 1 中的數據更改為 11。

當切換視窗成功時，切換後的視窗編號的寫入地址與 [觸發地址] 中設定的讀取地址、變數型態皆有關係，見下表：

變數型態	目的視窗編號讀取地址 (觸發地址)	切換後視窗編號的寫入地址
16-bit BCD	位址	位址 + 1
32-bit BCD	位址	位址 + 2
16-bit Unsigned	位址	位址 + 1
16-bit Signed	位址	位址 + 1
32-bit Unsigned	位址	位址 + 2
32-bit Signed	位址	位址 + 2

【只在指定的視窗被開啟時才執行】

此切換視窗的功能將只在指定的視窗內才有作用。

【換頁後地址數據歸零】

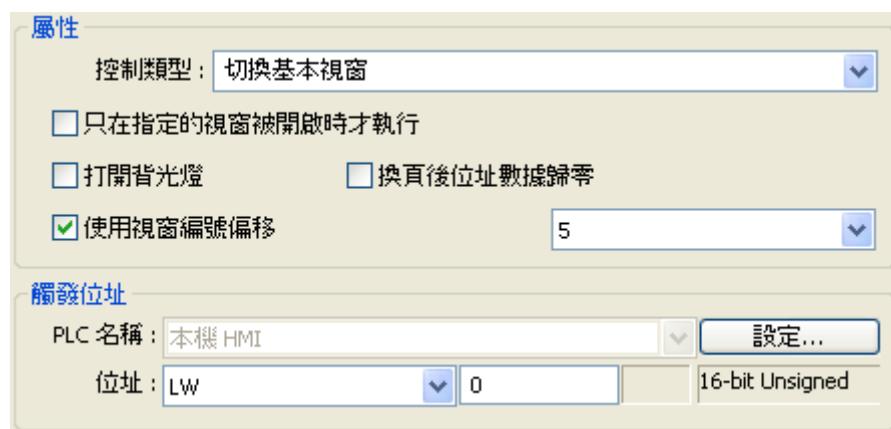
若啟用此選項，切換視窗成功後會將觸發地址中的數據歸零。

【換頁後打開背光燈】

若啟用此選項，當背光燈為關閉狀態時，切換視窗成功後會自動開啟背光燈。

[使用視窗編號偏移]

當勾選 [使用視窗編號偏移] 選項時，[觸發位址] 所指定暫存器中的數據，再加上 [視窗編號偏移]，才是實際切換的目的視窗號碼。例如：在下圖中，觸發位址為 LW - 0，偏移量為 5，當 LW - 0 的數值為 10 時，視窗將切換到視窗編號 15 (數值 10 + 偏移量 5)。偏移量範圍為 -1024 至 1024。



當 [LB - 9017] 的狀態被設定為 ON 時，切換後的視窗編號將不再寫至特定的地址中。

b. 資料寫入 PLC (當前基本視窗)

當切換基本視窗時，會將基本視窗的編號寫至 [觸發地址] 指定的地址中。

c. PLC 控制

此項功能提供使用者可以利用暫存器中的數據，控制 PLC 與 HMI 之間的資料傳輸，資料傳輸方向包含四種類型，參考下表的內容：

資料傳輸類型	資料傳輸方向
1	PLC 暫存器中的數據 → HMI 上的 RW 暫存器
2	PLC 暫存器中的數據 → HMI 上的 LW 暫存器
3	HMI 上的 RW 配方資料 → PLC 上的暫存器
4	HMI 上的 LW 暫存器 → PLC 上的暫存器

使用此項功能時，由 [觸發地址] 所設定的位址連續四個暫存器中的數據，決定資料傳輸類型、資料傳送數量、資料來源地址與資料傳送目的地址等。下表表示各暫存器中數據所表示的意義：

地址	用途	說明
[觸發地址]	存放資料傳輸類型，並決定資料傳輸的方向。	用來決定資料傳輸類型，如上所述，共有四種類型類型。當暫存器被寫入新的數據時，HMI 即執行相應的傳輸，傳輸完成後會將暫存器中的數據設為 0。
[觸發地址] + 1	欲傳輸資料大小，單位為 word	
[觸發地址] + 2	存放傳輸過程中數據來源的地址偏移量	傳輸的數據來源的起始地址為： [觸發地址] + 4 + 地址偏移量 以 OMRON PLC 為例，如果此時設定的 [觸發地址] 為 DM - 100，而在暫存器 [觸發地址] + 2 也就是 DM - 102 中的資料為“5”，則傳輸的數據來源的起始地址為 DM - 109，其中 $109 = (100 + 4) + 5$ 。
[觸發地址] + 3	存放傳輸過程中配方資料暫存器 (RW) 或者本地資料暫存器 (LW) 的起始地址	以 OMRON PLC 為例，如果此時設定的 [觸發地址] 為 DM - 100，而在暫存器 [觸發地址] + 3 也就是 DM - 103 中的資料為“100”，則傳輸過程中操作的 RW 或 LW 的起始地址為 RW - 100 或 LW - 100。

範例：

假使現在需要使用 [PLC 控制] 的功能，將 OMRON PLC 中從 DM - 100 起始的 16 words 的資料，傳輸到 HMI 配方記憶體 RW - 200 開始的地址中，設定方法如下：

(a) 首先，假設我們用 DM10 起始的四個資料暫存器來控制傳輸。先建立一個 PLC 控制物件，選擇類型為 [PLC 控制]，讀取地址設定為 DM - 10。

(b) 確定操作資料的大小和地址的偏移量。

將 DM - 11 設定為 16，表示傳輸資料的大小為 16 words；將 DM - 12 設定為 86，表示數據的來源地址為 DM - 100 ($100 = 10 + 4 + 86$)；將 DM - 13 設定值為 200，表示目標地址為 RW200。

(c) 最後，依照資料傳輸的方向，設定傳輸類型。

將 DM - 10 設定為 1，表示將傳輸 PLC 暫存器中的數據到 HMI 上的 RW 暫存器中。

如果設定 DM - 10 值為 3，則傳輸方向相反。

其他兩種傳輸方式具有類似的設定方法，差別只在 HMI 的資料記憶體變成了本地資料暫存器 LW。

d. 背光燈控制 (返回值)

當 [觸發地址] 的狀態由 OFF 變為 ON 時, HMI 將打開/關閉背光燈, 此時也會將 [觸發地址] 的狀態設定為 OFF。背光燈關閉時, 使用者只需碰觸螢幕, 背光燈即會再度打開



e. 背光燈控制

當 [觸發地址] 的狀態由 OFF 變為 ON 時, HMI 將打開/關閉背光燈。但因不具備 "返回值" (write back) 功能, 此時並不會將 [觸發地址] 的狀態設定為 OFF。

f. “聲音控制”



當 [觸發地址] 的狀態改變符合觸發條件時, 將播放預先指定的聲音檔案。

[觸發方式] 可以選擇：

- (a) 狀態由 OFF 變為 ON (OFF->ON)
- (b) 狀態由 ON 變為 OFF (ON->OFF)
- (c) 只需狀態改變 (OFF<->ON)

g. 執行巨集指令



當【觸發地址】的狀態改變符合觸發條件時，將執行指定的巨集指令。

【觸發方式】可以選擇：

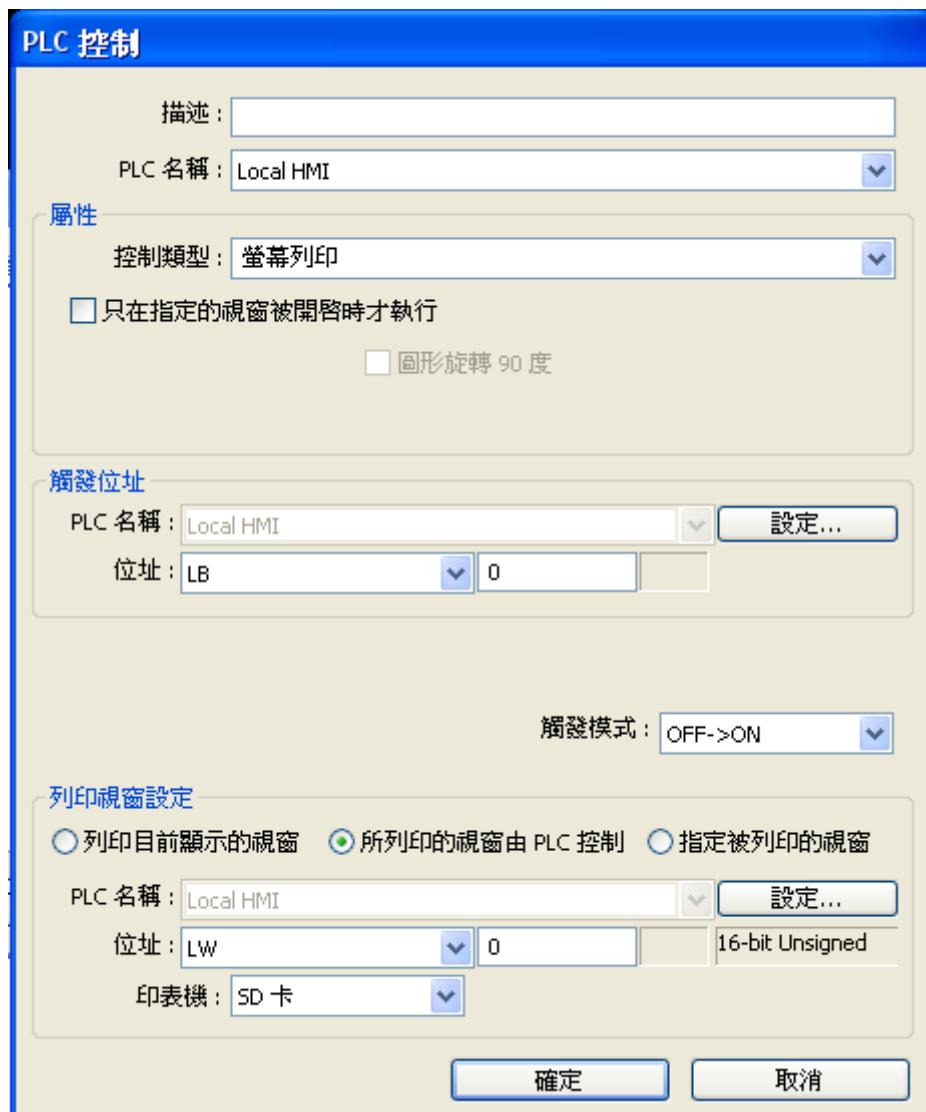
- (a) 狀態由 OFF 變為 ON (OFF->ON)
- (b) 狀態由 ON 變為 OFF (ON->OFF)
- (c) 只需狀態改變 (OFF<->ON)
- (d) 當狀態為 ON 時即執行：只需狀態維持在 ON，即可持續執行指定的巨集指令。

h. 螢幕列印

當【觸發地址】的狀態改變符合觸發條件時，將列印指定的畫面。【觸發方式】可以選擇：

- (a) 狀態由 OFF 變為 ON (OFF->ON)
- (b) 狀態由 ON 變為 OFF (ON->OFF)
- (c) 只需狀態改變 (OFF<->ON)

可以選擇要列印的畫面，共有三種指定方式，參考下圖。

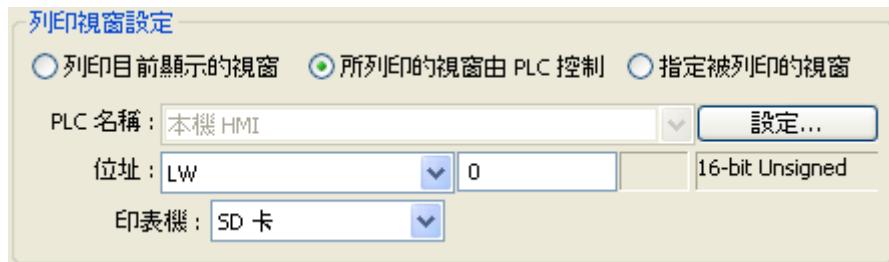


[列印目前顯示的視窗]

將列印目前顯示的畫面

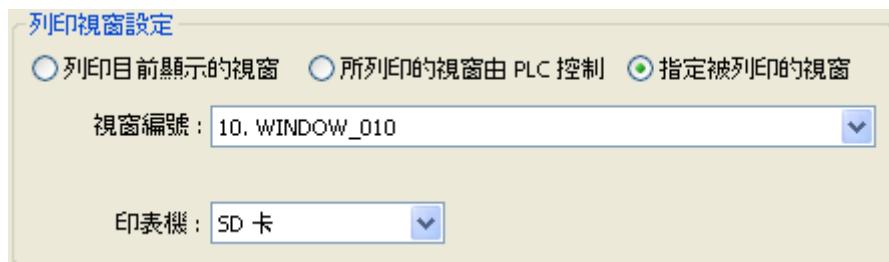
[所列印的視窗由 PLC 控制]

將利用指定的地址讀取數據，此數據即為視窗的號碼。如果此視窗存在，則列印這個視窗的內容。如下圖：



[指定被列印的視窗]

直接指定要列印的視窗，參考下圖。



[印表機]

若未設定印表機，可選擇使用 SD 卡或 USB 碟。

印表機可在 **【系統參數設定】** 的 **【HMI 屬性】** 頁面中設定。



- 當指定被列印的視窗不是當前視窗時，系統提供背景列印。
- 指定背景視窗時，該視窗之直接視窗或間接視窗將不會被列印。

13.28 排程

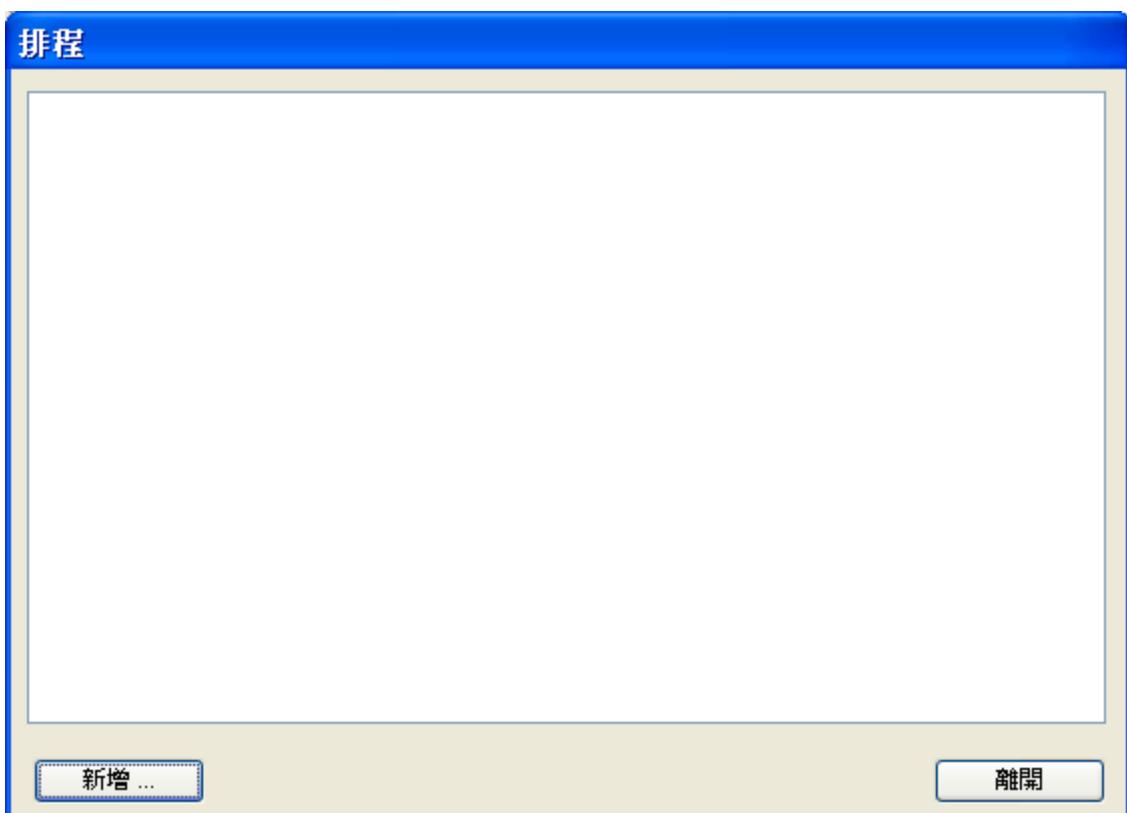
概要

[排程] 可用來設定時刻表，將位元設為 ON / OFF 或在字元位址寫入數值，適合用來規劃一週內的例行程序。

設定



按下工作列上的 [排程] 按鈕後即會出現對話窗，按下 [新增] 鍵，即可進入排程的設定頁。



因排程較為複雜，將先介紹兩個範例再詳細說明各項功能：

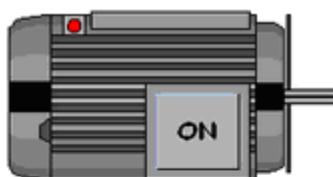
Example 1

範例 1—設定程序

馬達(位址：LB - 100)從星期一直運轉到星期五，時間由每天上午 9 點到下午 6 點。設定程序將為在起始時間 (早上 9 點) 將位址 LB - 100 設為 ON，在結束時間 (下午 6 點) 將位址 LB - 100 設為 OFF。



當設定開始時間到達時



開始運轉



當設定停止時間到達時



停止運轉

從物件選單中選取【排程】選項或按下工作列上的【排程】按鈕，即出現排程新增對話框。

【一般屬性】

- 選定【行動模式】為【位元設為 ON】。

行動模式

- 位元設為ON 位元設為OFF 字組寫入

- 設定【目標位址】(例如：LB - 100)

目標位址	
PLC 名稱 :	Local HMI
設備類型 :	LB
位址 :	100
<input type="checkbox"/> 系統暫存器 <input type="checkbox"/> 索引暫存器	

[時間設定頁]

3. 選擇 [時間設定] 頁籤，接著選擇 [常數]。



4. 設定 [起始時間及日期]。將時間設為 9 點 0 分 0 秒，接著勾選星期一到星期五，取消 [設定為單一日期]。



5. 設定 [結束時間及日期]。勾選 [啓用結束動作]，將結束時間設定為 18 點 0 分 0 秒。



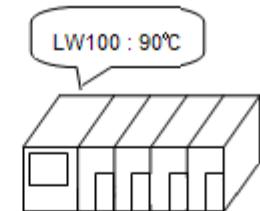
6. 按下 [確定] 鍵後，即可看到排程的日程表，如下圖。

Example 2**範例 2—設定程序**

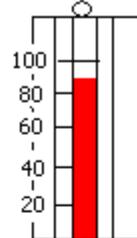
從星期一到星期五，在起始時間 8 點把溫度設定值 90 度寫入字組位址 LW - 100，此時系統進入運轉模式。在結束時間 17 點把溫度設定值 30 度寫入字組位址 LW - 100，此時系統進入等待模式。



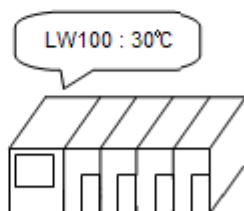
當指定的起始
時間到達...



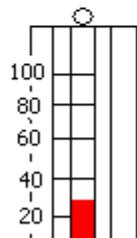
寫入運轉模式的溫度設定



當指定的結束
時間到達...



寫入等待模式的溫度設定



從物件選單中選取 [排程] 選項或按下工作列上的 [排程] 按鈕，隨即出現排程新增對話框。
按下 [新增] 按鈕，新增 [排程] 物件。



【一般屬性】

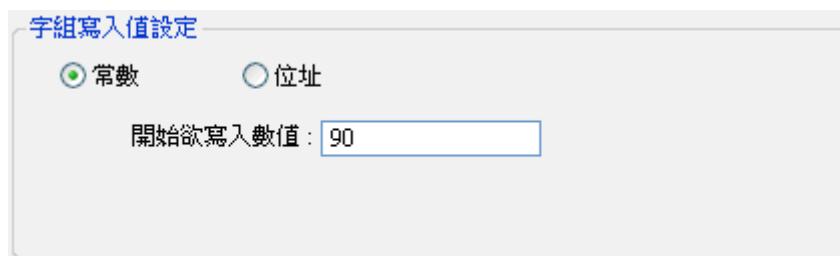
- 選定 [動作模式] 為 [字組寫入]。



2. 設定 [動作位址] (例如：LW - 100)。



3. 選擇[常數]，設定[開始欲寫入數值]為 90。



[時間設定]

4. 選擇 [時間設定] 頁籤，接著選擇 [常數]。



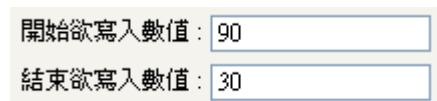
5. 設定 [起始時間及日期]。將時間設為 8 點 0 分 0 秒，接著勾選星期一到星期五。取消勾選 [設定為單一日期]。



6. 設定 [結束時間及日期]。勾選 [啓用結束動作]，將結束時間設定為 17 點 0 分 0 秒。



7. 選擇 [一般] 頁籤，設定 [結束欲寫入數值] 為 30。



8. 按下 [確定] 鍵，完成。

一般屬性設定



【電源開啓時執行開啟/結束動作】

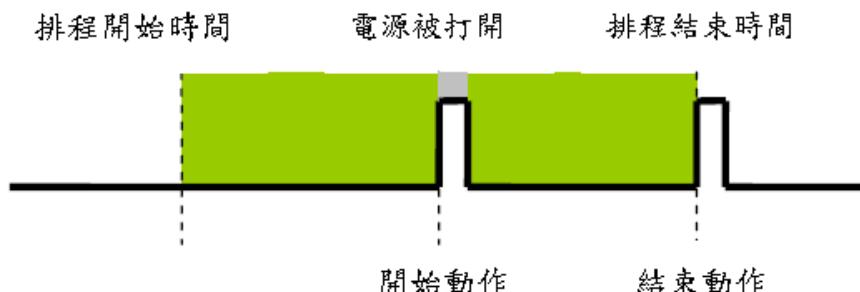
當電源打開時，執行已設定的動作。

- 啓用時

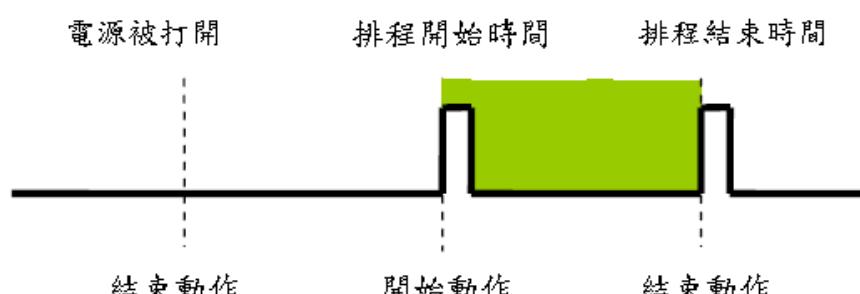
假如 HMI 的電源在排程區間內被打開，則開始動作會被執行。

假如 HMI 的電源在排程區間外被打開，則結束動作會被執行。

在排程區間內



在排程區間外



• 停用時

假如電源打開時晚於排程開始時間，則開始動作不會自動執行。然而結束動作會自動執行。
假如結束動作未被設定，由於無法正確地判定排程區間，結束動作將不會被執行。

行動模式

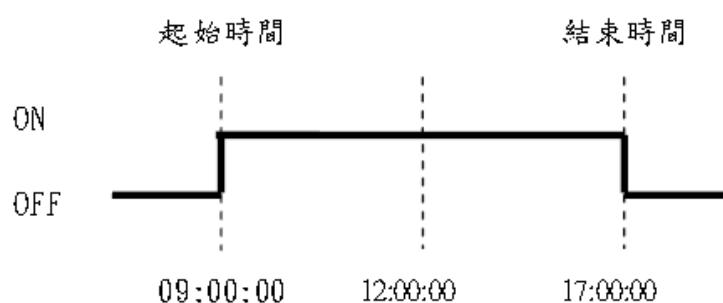
選擇在設定的時間要操作的類型。

[位元設為 ON]

在排程開始時，將指定位元位址的狀態設為 ON；在排程結束時，將指定位元位址的狀態設為 OFF。

例如：起始時間：09:00:00

結束時間：17:00:00

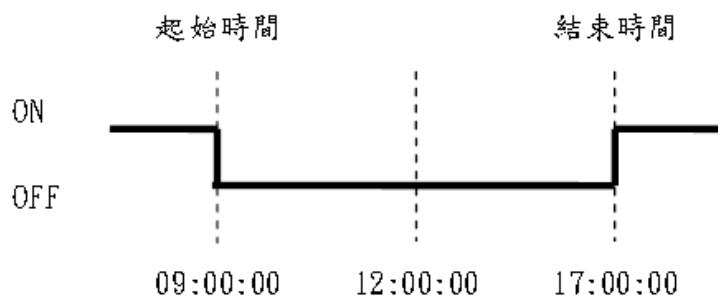


【位元設為 OFF】

在排程開始時，將指定位元位址的狀態設為 OFF；在排程結束時，將指定位元位址的狀態設為 ON。

例如：起始時間：09:00:00

結束時間：17:00:00



【字組寫入】

在排程開始時，將 [開始欲寫入數值] 寫入指定字元位址；在排程結束時，將 [結束欲寫入數值] 寫入指定字元位址。

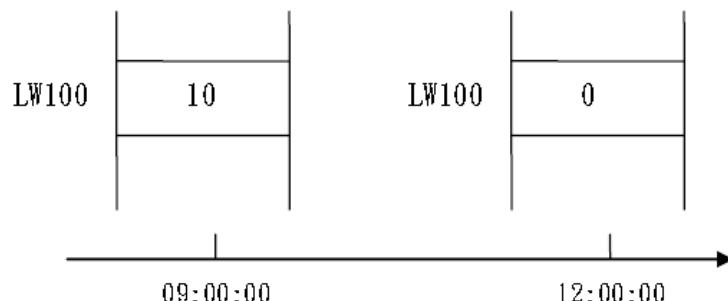
例如：字組寫入值設定位址：LW - 100

起始時間：09:00:00

結束時間：17:00:00

開始欲寫入數值：10

結束欲寫入數值：0



- 必須在 [時間設定] 頁籤中勾選 [啓用結束動作] 才能使用 [結束欲寫入數值]。

時間設定

常數 / 位址：選擇設定起始時間和結束時間的方法。[常數] 可指定一個固定的時間和日期，而 [位址] 將指定特定位址作為時間和日期的資訊。

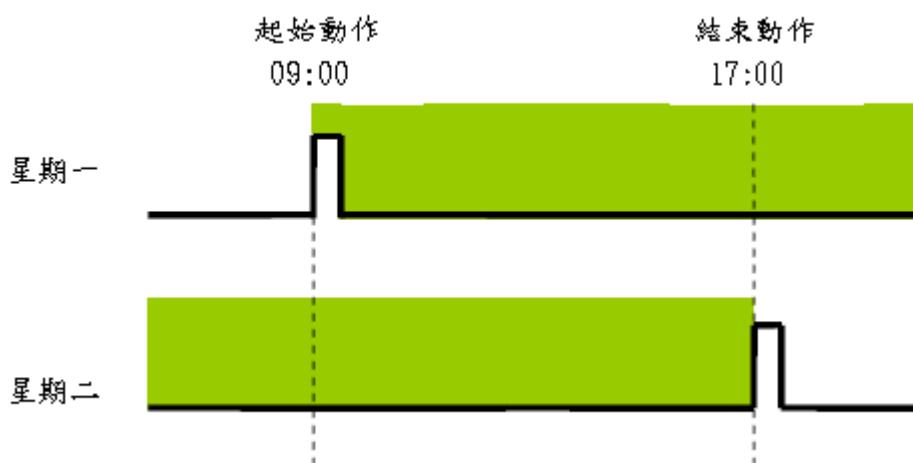
時間設定: [常數]



[設定為單一日期]

- 啓用時

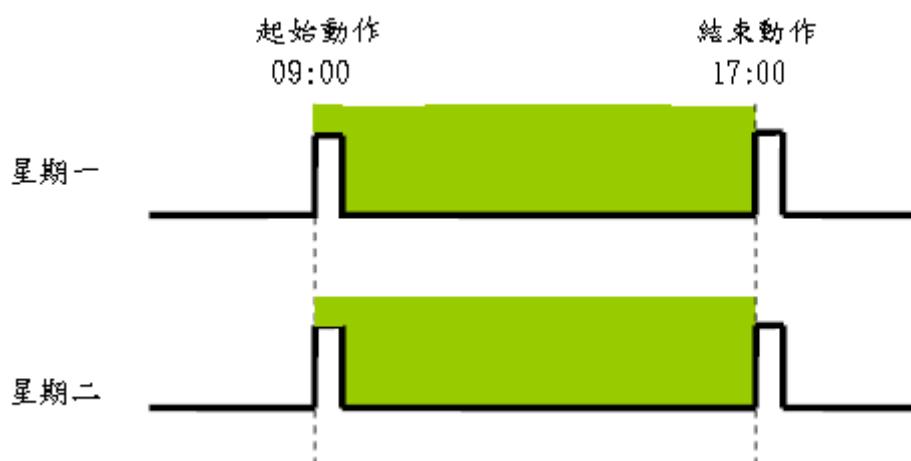
動作可在一周內指定的日期及時間被執行。當啟用後，則必須設定所有日期有相同的動作開始時間及結束時間。



- 必須輸入起始時間和結束時間。
- 不能在起始時間和結束時間欄位裡輸入一模一樣的時間和日期。

[設定為單一日期]

- 停用時
排程時間必須被限定在一天之內(起始時間和結束時間必須在 24 小時內)。



- 不能在起始時間和結束時間欄位裡輸入一模一樣的時間和日期。
- 此種時間排程只適用於一天之內的排程，因此如果所鍵入的結束時間早於起始時間，則結束動作將會等到下一天才會執行。

例如：..

起始日期：星期一..
起始時間：22:00:00..
結束時間：01:00:00..

星期一..

星期二..



時間設定：[位址]

當地址被選擇後，系統會顯示控制位址為字元的型態，如下圖。此時使用者可經由控制這些字元位址來手動設定開始及結束時間。



使用者只需定義時間設定位址，其餘的並 11 個控制字元會自動產生並列示出來。圖上之資料長度皆以 16 位元為例；當指定之暫存器為 32 位元時，只有較低的 16 位元產生作用，並請將較高的 16 位元固定為 0

以下說明各位址之使用說明：

a. 控制 (時間設定位址 + 0)

當 [更新時間位元] (見下圖) 被偵測為 ON ($0 \rightarrow 1$) 時，則讀出 [模式]、[開始時間] 和 [結束時間]。

15

0

位元

保留 (0 固定)

0

位元 00 : 更新時間位元(0:無動作, 1:讀取排程時間資料)



- HMI 並不會定期地讀取時間設定位址的【模式】(位址 + 2) 到【結束時間(秒)】(位址 + 10) 裡的資料。所以，當排程時間資料改變時，請務必把【控制】中的【更新時間位元】設為 ON (0→1)。

b. 狀態(時間設定位址 + 1)

在【控制】中的時間資料讀取完成之後，HMI 將會把【時間讀取完成位元】設為 ON (0→1)。同樣地，若輸入的時間資料不正確，【錯誤通知位元】將會同時被設為 ON (0→1)。

15

02 01 00

位元

保留 (0 固定)

0

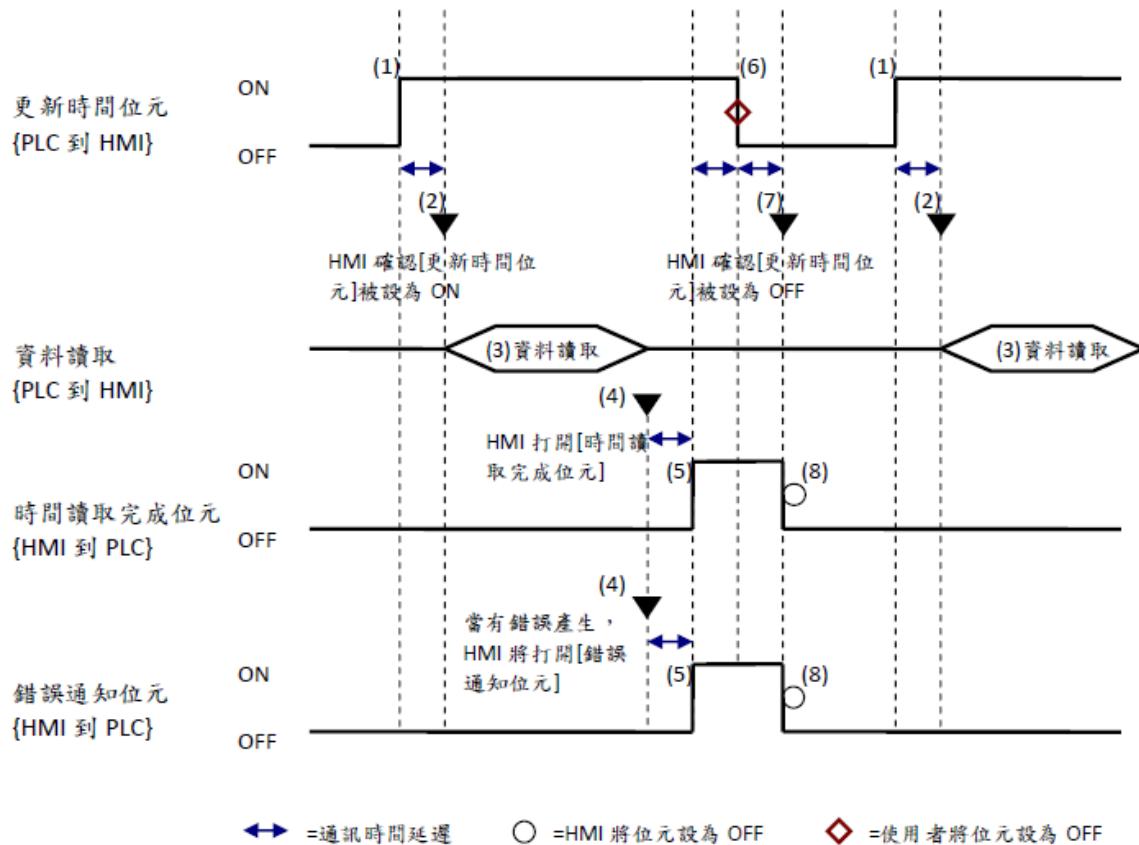
0

位元 00 : 時間讀取完成位元(0:還沒開始或是正在讀取時間資料；1:時間資料讀取完成)

位元 01 : 錯誤通知位元(0:時間資料被正確更新；1:時間資料中包含錯誤)



- 一旦發現【時間讀取完成位元】被觸發，請務必把【控制】中的【更新時間位元】設為 OFF(1→0)。一旦這個位元被設為 OFF(1→0)，則【狀態】中的【時間讀取完成位元】及【錯誤通知位元】將同時被設為 OFF(1→0)。



c. 模式(時間設定位址 + 2)

啓用或停用【結束時間動作設定】和【單一日期指定模式】。不管【結束時間動作設定】的狀態是如何，所有的時間資料 (時間設定位址) 中的 11 個字組位址) 都會被讀取。

15

02 01 00

位元

保留 (0 固定)

0 0

位元 00 : 結束時間動作設定(0:停用；1:啓用)

位元 01 : 單一日期指定模式(0:停用；1:啓用)



- 若【結束時間動作設定】輸入 0(停用)，仍會讀取結束時間資料但忽略其內容。
若【單一日期指定模式】輸入 1(啓用)，請確認你有輸入開始及結束時間資訊。假如有 2 個以上的開始/結束日期位元被同時設為 ON，則會產生錯誤。

d. 開始 / 結束日期 (開始日期：時間設定位址 + 3；結束日期：時間設定位址 + 7)

指定觸發開始/結束動作的日期。

15	07	06	05	04	03	02	01	00	位元
保留 (0 固定)	Sat	Fri	Thu	Wen	Tue	Mon	Sun		

位元 00：星期日(0：無；1：指定)

位元 01：星期一(0：無；1：指定)

位元 02：星期二(0：無；1：指定)

位元 03：星期三(0：無；1：指定)

位元 04：星期四(0：無；1：指定)

位元 05：星期五(0：無；1：指定)

位元 06：星期六(0：無；1：指定)

e. 開始/結束時間 (開始時間：時間設定位址 + 4 到 + 6；結束時間：時間設定位址 + 8 到 + 10)

時：0 - 23。

分：0 - 59。

秒：0 - 59。

假如你所指定的值超出上面的範圍，將會產生錯誤。



- 使用者所輸入的時間資料應為 **16 位元無正負號**格式，系統不接受 BCD 格式的時間資料。
- 結束時間取決於 [模式](位址+2) 設定。同樣地，[結束時間動作設定] (位元 00)有效與否取決於 [單一日期指定模式] (位元 01)的使用。

單一日期指定模式	使用	不使用	
結束時間動作設定	使用	使用	不使用

禁制



禁止

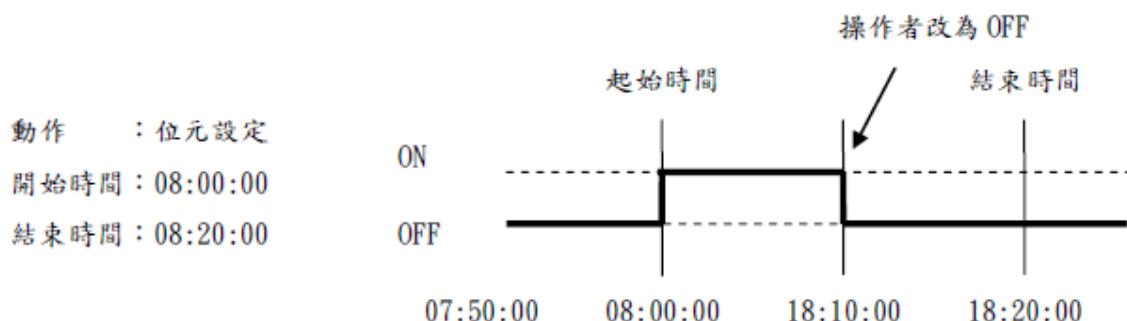
啓用時，在執行開始動作前 HMI 將讀取此位元狀態，若為 ON，則略過此次開始及結束動作(若存在)；反之則正常執行設定動作。

按鍵聲音

啓用時，在執行開始及結束動作(若存在)同時播放指定音效。

使用排程限制:

- 最多可註冊 32 個 [排程] 物件。
- 時間排程的特性為一次動作。當開始時間到達時，特定的設備位址只會被寫入一次，這個寫入的動作將不會重複。



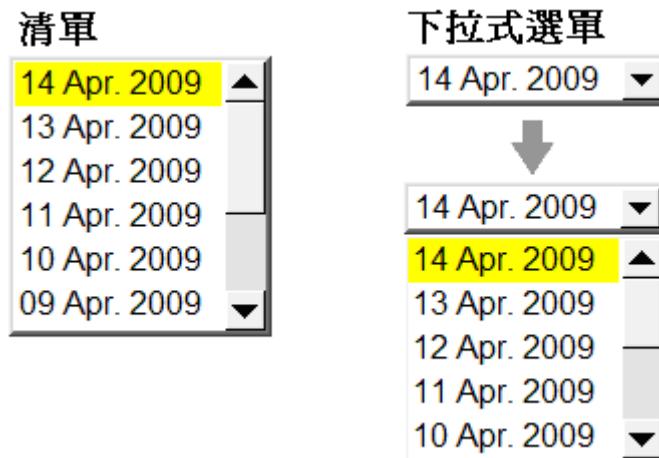
- [開始/結束寫入數值] 和 [禁制位元] 只會在執行開始動作前讀取一次。所以當開始動作執行後，就算再去改變 [禁制位元] 狀態或 [結束寫入數值] 都無法改變結束動作的執行與否及寫入數值。另外，為了讀取 [開始/結束寫入數值] 和 [禁制位元] 資料，起始動作可能因資料通訊而有少許延遲。
- 當使用者改變 HMI 的系統時間，系統將會重新確認排程中起始與結束時間的範圍。假如編輯的物件位於新範圍中，則開始動作會被執行。假如結束動作未被設定，系統無法確認新範圍，則這個動作將不會被執行。
- 當相同的起始和結束時間出現在多個排程物件中，他們將依編號由小到大順序被處理。
- 當 [時間設定] 指定為 [位址]，系統將會定期去讀取 [控制] 位址，時間長短視系統忙碌程度而定。
- 當 [時間設定] 指定為 [位址]，且指定開始時間和結束時間超過時間合法的範圍，則設定的時間可能不會正確地運作。注意：不能使用 BCD 當成輸入值。
- 當 [時間設定] 指定為 [位址]，排程物件直到第一次成功更新時間資料，才開始運作。

13.29 項目選單

概要

[項目選單] 物件可以顯示多樣項目成一列表，使用者可以藉此去檢視並選擇。一旦使用者選擇了某一項目，相對應的項目數值將被寫入到字元暫存器。

[項目選單] 有兩種顯示模式：**[清單]** 和 **[下拉式選單]**。清單可以完整顯示所有的項目，並把目前所選擇的項目標示出來。此外，下拉式選單在一般情況下只顯示目前所選擇之項目。但是當使用者點選下拉式選單時，系統則會列出所有完整項目(類似於清單的顯示法) 如下：



設定



點選 **[項目選單]** 圖示，物件屬性對話窗顯示如下：

項目選單分頁



屬性

1. [模式]

可選擇清單或下拉式選單。

2. [項目數]

設定需使用的項目數。每個項目代表一個狀態，當被選擇時，會將指定的數據寫入監看位址中。

3. [背景]

改變背景顏色。

4. [選擇]

改變選擇的項目的背景顏色。

5. [項目資料來源]

項目選單顯示的內容，共有三種模式：預設、歷史數據日期、項目位址模式。

6. [監看位址]

系統會將已選擇項目的數據寫入 [監看位址] 中。

[當按鈕鬆開才發出指令]

當啟用時，當按鈕鬆開時才會將指定的數據寫入監看位址中。

[項目資料來源]

(1) [預設]

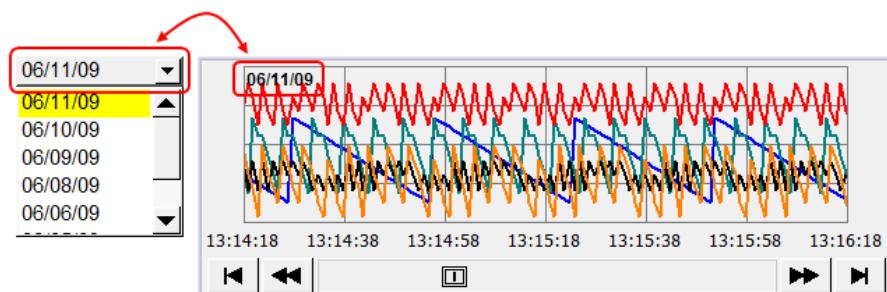
顯示的選項由使用者在 [狀態設定] 分頁中手動輸入。



可調整欲使用的 [項目數]，設定此物件的狀態數。每一個項目表示一個狀態並會顯示在列表上，此項目數值可被寫入至 [監看位址]。

(2) [歷史數據日期]

搭配使用在歷史模式下的事件顯示物件。例如，趨勢圖及歷史數據顯示物件，用來顯示上述物件的歷史檔案並顯示在螢幕上。顯示方式如下圖：

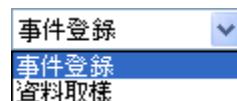


將使用現有的 [資料取樣] 或 [事件登錄] 物件，對話窗下方將顯示額外設定：



(a) [類型]

可選擇 [事件登錄] 或是 [資料取樣]。



(b) [日期]

共有 8 種日期模式可選擇，YYYYY 代表四位數年份(例如 2012)、YY 代表二位數年份(例如 12)、MM 代表月份、DD 代表日期。

(c) 若 [類型] 選擇 [資料取樣]，需在 [資料取樣索引] 設定欲顯示的資料取樣物件。

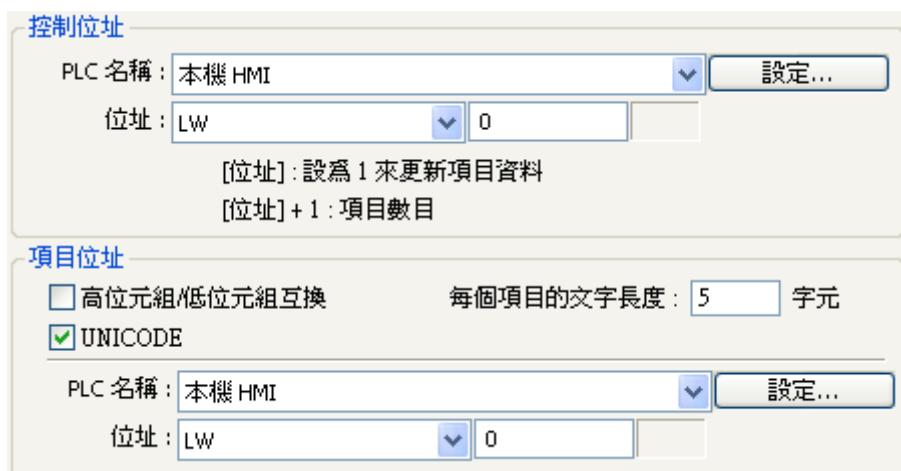
一般來說，選擇與搭配之趨勢圖的歷史模式或歷史數據物件相同即可。



- 當選用歷史數據來源(歷史資料日期)，由於系統在執行期間將自動產生狀態設定，故此處使用者不需再填寫。
- 當項目選單進入錯誤狀態時，項目選單將顯示「？」

(3) [項目位址模式]

當使用 [項目位址模式] 後，下方會出現 [控制位址] 和 [項目位址]，如下圖：



(a) [控制位址]

[位址]: 若將此位址所指定的暫存器中數據設定為 1，將更新物件所顯示的選項為 [項目位址] 的內容。更新完成後此暫存器中的數據會被恢復為 0。

[位址] + 1: 此位址中的數據用來設定選項的個數。

(b) [項目位址]

指定用來存放選項的位址。

◆ **[UNICODE]**

選項的內容使用 UNICODE 文字，例如中文字。

可能使用到的 UNICODE 文字必須也使用在其他物件上，EasyBuilder 才會預先編譯所需要的字型檔案，在下載時一併存放到 HMI 上，如此才能正常顯示 UNICODE 文字。

◆ **[每個項目的文字長度]**

每一個選項的文字長度。



- (選項的個數) * [每個項目的文字長度] 不得超過 1024。
- 若選擇項目位址模式，系統將自動取消狀態設定頁面。

寫入成功後傳送通知

啟用此項設定後，在命令傳送成功時會連帶將指定位元暫存器的狀態設定為 [開] 或 [關]。點選 **【設定】** 後選擇位元暫存器設備類型的 **[PLC 名稱]**、**[位址]**、**[設備類型]**、**[系統暫存器]**、**[索引暫存器]** 來控制位元狀態設定物件。使用者也可在 **【一般屬性】** 頁中設定位址。

狀態設定分頁



狀態設定

此設定頁顯示所有狀態項目、文字和數值，如果要改變項目數，請點選【項目選單】，在【屬性】區塊中選擇【項目數】。

[項目]

系統會列出目前所有使用的項目，每一個項目表示一個狀態並且會顯示在列表。此欄為唯讀。

[數值]

使用者可為每個項目設定數值，但須注意以下兩點：

1. 從監看位址讀取：如果系統偵測到【監看位址】的內容有任何改變，物件將會對照內容和其數值並選擇第一個吻合的項目。如果沒有項目吻合，將跳至錯誤狀態並觸發錯誤通知位元 (若已設定)

2. 寫入監看位址：當使用者選擇某項目，系統將數值寫入至 [監看位址]。

[項目資料]

使用者可為每個項目設定顯示文字，項目選單物件將顯示所有項目的文字在列表上供使用者檢視和選擇。

[錯誤狀態]

錯誤狀態的文字只能應用於 [下拉式選單] 模式，[清單] 模式無法使用錯誤狀態文字。

在錯誤狀態發生時，[清單] 模式將清除文字來表示沒有任何項目被選擇，而 [下拉式選單] 模式則會顯示錯誤狀態的文字。

例如，當 [項目數] 設 8 時，8 即為錯誤狀態。同樣的，如果設 [項目數] 為 11 時，狀態 11 即為錯誤狀態。(因狀態由 0 算起)

[設為預設值]

將所有項目數值還原為預設值，即：設項目 0 為 0，項目 1 為 1...等等。

錯誤通知

當項目選單偵測到不合法的數據寫入時，會觸發指定暫存器的狀態 [開] 或 [關] 以示警告。此外，使用者亦可搭配其他物件修正錯誤。

13.30 計時器

概要

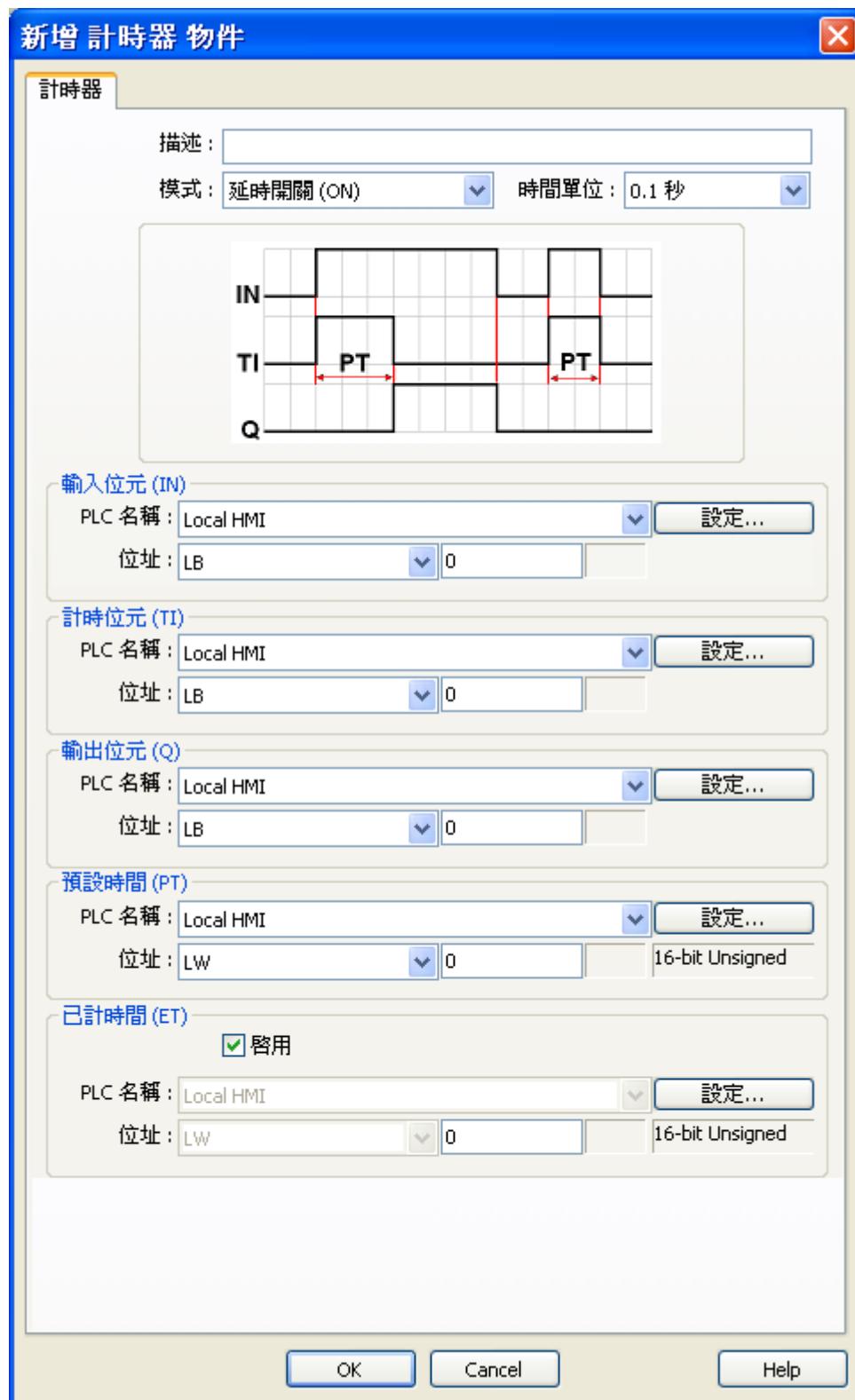
計時器相當於一計時開關，可做為延時開關、脈衝開關及累加式延時開關，其包含下列六項變數：

計時器變數	變數類型	敘述
輸入位元 (IN)	位元變數	計時器的總開關
測量位元 (TI)	位元變數	計時開始時設 ON
輸出位元 (Q)	位元變數	計時結束後啟動相關設定
預設時間 (PT)	字元變數	設定計時器時間數值
已計時間 (ET)	字元變數	顯示計時器目前已計時間
重置位元 (R)	位元變數	將目前計時器已計時間 (ET) 歸零

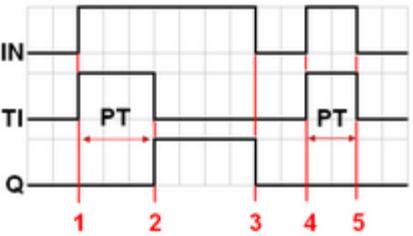
設定

點選 [計時器] 圖示，其 [計時器] 物件屬性對話窗顯示如下：

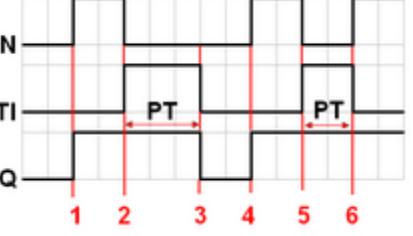




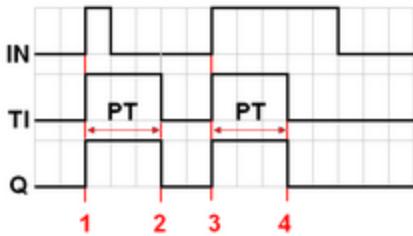
■ 延時開關 (ON)

電位圖	暫存器
	<ul style="list-style-type: none"> • 輸入位元 (IN): 計時器的總開關。 • 測量位元 (TI) • 輸出位元 (Q) • 預設時間 (PT) • 已計時間 (ET): 顯示計時器目前已計時間。
說明 (參照上圖)	
<p>時段 1: 輸入位元 IN 設 ON 時，測量位元 TI 開啟，已計時間 ET 開始計數，輸出位元 Q 保持 OFF。</p> <p>時段 2: 當已計時間 ET 等於預設時間 PT 時，測量位元 TI 被關閉，同時輸出位元 Q 被開啟。</p> <p>時段 3: 輸入位元 IN 設 OFF 時，輸出位元 Q 被關閉，同時已計時間 ET 歸零。</p> <p>時段 4: 輸入位元 IN 設 ON 時，測量位元 TI 被開啟，已計時間 ET 開始計數，輸出位元 Q 保持 OFF。</p> <p>時段 5: 在已計時間 ET 到達預設時間 PT 之前，將輸入位元 IN 設為 OFF，測量位元 TI 將被關閉，同時已計時間 ET 歸零。因為達 ET 仍小於 PT，輸出位元 Q 保持在 OFF。</p>	

■ 延時開關 (OFF)

電位圖	暫存器
	<ul style="list-style-type: none"> • 輸入位元 (IN): 計時器的總開關。 • 測量位元 (TI) • 輸出位元 (Q): 計時結束後設 ON。 • 預設時間 (PT) • 已計時間 (ET)
說明 (參照上圖)	
<p>時段 1: 輸入位元 IN 設 ON 時，測量位元 TI 保持 OFF，輸出位元 Q 被開啟，已計時間 ET 歸零。</p> <p>時段 2: 輸入位元 IN 設 OFF 時，測量位元 TI 被開啟，輸出位元 Q 保持 ON，已計時間 ET 開始計數。</p> <p>時段 3: 當已計時間 ET 等於預設時間 PT 時，輸出位元 Q 和測量位元 TI 被關閉。</p> <p>時段 4: 輸入位元 IN 設 ON 時，測量位元 TI 保持 OFF，輸出位元 Q 被開啟，已計時間歸零。</p> <p>時段 5: 當輸入位元 IN 設 OFF 時，測量位元 TI 被開啟，輸出位元 Q 保持 ON，已計時間 ET 開始計數。</p> <p>時段 6: 當在已計時間 ET 到達預設時間 PT 的數值前設輸入位元 IN 為 ON，測量位元 TI 被關閉，同時輸出位元 Q 保持 ON，已計時間 ET 歸零。</p>	

■ 脈衝啟動開關

電位圖	暫存器
	<ul style="list-style-type: none"> • 輸入位元 (IN): 計時器的總開關。 • 測量位元 (TI) • 輸出位元 (Q) • 預設時間 (PT) • 已計時間 (ET)
說明 (參照上圖)	

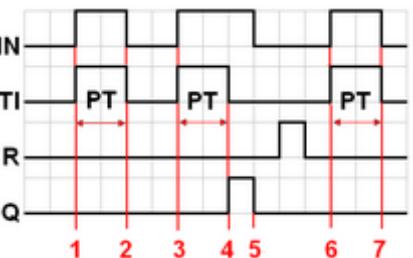
時段 1: 當輸入位元 IN 設 ON 時，測量位元 TI 和輸出位元 Q 同時被開啟，已計時間 ET 開始計數。

時段 2: 當已計時間 ET 等於預設時間 PT 時，輸出位元 Q 和測量位元 TI 同時被關閉。(因為在計數同時已先將輸入位元 IN 設 OFF，所以已計時間 ET 將被自動歸零。)

時段 3: 當輸入位元 IN 設 ON 時，測量位元 TI 和輸出位元 Q 同時被開啟，已計時間 ET 開始計數。

時段 4: 當已計時間 ET 等於預設時間 PT 時，輸出位元 Q 和測量位元 TI 同時被關閉。

■ 累加式延時開關 (ON)

電位圖	暫存器
	<ul style="list-style-type: none"> • 輸入位元 (IN): 計時器的總開關。 • 測量位元 (TI): 計時開始時設 ON。 • 輸出位元 (Q): 計時結束後設 ON。 • 預設時間 (PT): 設定計時器時間數值。 • 已計時間 (ET): 顯示計時器目前已計時間。 • 重置位元 (R): 將目前已計時間 (ET) 歸零。
說明 (參照上圖)	

時段 1: 當輸入位元 IN 設 ON 時，測量位元 TI 被開啟，已計時間 ET 開始計數，輸出位元 Q 保持 OFF。

時段 2: 當輸入位元 IN 設 OFF 時，如果已計時間 ET 未到達預設時間 PT，測量位元 TI 被關閉，同時輸出位元 Q 保持 OFF。已計時間 ET 保持現在的狀態數值。

時段 3: 當輸入位元 IN 再度設 ON 時，測量位元 TI 被開啟，同時已計時間 ET 再次由剛剛保持的狀態數值開始計數，同時輸出位元 Q 保持 OFF。

時段 4: 當已計時間 ET 等於預設時間 PT 時，測量位元 TI 被關閉，同時輸出位元 Q 被開啟。

時段 5: 設輸入位元 IN 為 OFF，同時輸出位元 Q 也被關閉。(此時設重置位元 ON 使已計時間 ET 歸零後再設為 OFF。)

■ 累加式延時開關 (OFF)

電位圖	暫存器
	<ul style="list-style-type: none"> • 輸入位元 (IN)：計時器的總開關。 • 測量位元 (TI)：計時開始時設 ON。 • 輸出位元 (Q)：計時結束後設 OFF。 • 預設時間 (PT)：設定計時器時間數值。 • 已計時間 (ET)：顯示計時器目前已計時間。 • 重置位元 (R) 將目前計時器已計時間 (ET) 歸零。
說明 (參照上圖)	
時段 1: 當輸入位元 IN 設 ON 時，測量位元 TI 保持 OFF，同時輸出位元 Q 被開啟。	
時段 2: 當輸入位元 IN 設 OFF 時，測量位元 TI 被開啟，同時輸出位元 Q 保持 ON。已計時間 ET 開始計數。	
時段 3: 當輸入位元 IN 再度設 ON 時，測量位元 TI 和輸出位元 Q 保持 ON，同時已計時間 ET 暫停計數。	
時段 4: 當輸入位元 IN 再度設 OFF 時，已計時間 ET 再次由剛剛保持的狀態數值開始計數。	
時段 5: 當已計時間 ET 等於預設時間 PT 時，測量位元 TI 和輸出位元 Q 同時被關閉。(此時設重置位元 ON 使已計時間 ET 歸零後再設為 OFF。)	

13.31 影像輸入

概要

如 HMI 型號提供影像輸入功能，使用者加裝監視鏡頭後，在透過監視鏡頭即可即時監看現場狀況，也能將畫面記錄到儲存裝置並且在電腦上做分析。

此功能可應用在各個層面，除了可監看現場狀況外，也能應用在行車裝置或是大樓監控。

在硬體上，具有影像輸入的 HMI 提供 2 個影像輸入通道，使用者可自由切換所欲監控的畫面，而且影像擷取功能不受暫停播放控制之影響，所擷取的圖片仍是外部影像輸入之即時畫面。可支援 NTSC 及 PAL 二種影像格式。

注意： MT 系列則是 MT8000X 系列才支援此功能。

設定



點選工具列上之 [影像輸入] 按鈕以新增影像輸入物件，其對話視窗如下：



[輸入通道]

可選擇使用 Video Input 1 或 Video Input 2

[編碼格式]

依據不同視訊格式，可選擇 NTSC 或 PAL 訊號

影像擷取位址

啟用 [使用影像擷取功能] 來擷取輸入影像畫面之功能。

**1. [影像擷取位址]**

觸發系統擷取圖片的控制位址

2. [儲存空間]

選擇圖片儲存媒體: SD 或 USB 碟。

- 影像輸入通道 1 影像將儲存在儲存媒體的影像輸入通道 1(VIP1)目錄中，影像輸入通道 2 則儲存在影像輸入通道 2(VIP2)目錄。

3. [記錄時間]

設定擷取畫面之時間範圍。

- 最大擷取範圍為 [影像擷取位址] 觸發時之前後 10 秒。
- 系統每秒擷圖一次。
- 圖片檔案命名規則:

[影像擷取位址] 觸發前後: YYYYMMDDhhmmss.jpg

[影像擷取位址] 觸發當時: YYYYMMDDhhmmss@.jpg

以上圖為例，設定記錄時間為前後 5 秒，當 [影像擷取位址] 由 OFF 轉為 ON 時，系統將從觸發時間點起算，每秒 1 張，擷取前後 5 秒之輸入畫面，共 11 張圖。

控制位址

啟用 [使用控制功能] 來使用 [控制位址] 控制影像輸出。



例如：指定控制位址為 LW100：

1. 用戶可藉由設定 [控制位址] 來啟動/停止影像輸入：

[LW100] = 0 → 停止播放

[LW100] = 1 → 開啟影像輸入通道 1 影像輸入並顯示於屏幕上

[LW100] = 2 → 開啟影像輸入通道 2 影像輸入並顯示於屏幕上

[LW100] = 3 → 開啟影像輸入通道 1 影像輸入但不顯示 (仍可執行影像擷取功能)

[LW100] = 4 → 開啟影像輸入通道 2 影像輸入但不顯示 (仍可執行影像擷取功能)

2. 用戶可藉由設定 [控制位址+1] 對影像顯示做額外控制：

[LW101] = 1 → 暫停/繼續播放

3. 用戶在變更 [控制位址+0] 的值後，系統將保留變更後的值。

4. 用戶在變更 [控制位址+1] 的值後，系統將在執行對應命令結束後將其清除為 0

5. 若不啟用 [使用控制功能]，系統將自動播放 [輸入通道] 指定之影像輸入

[顯示調整]

勾選此選項後可調整畫面的亮度及對比。若設定控制位址為 LW100，則：

1. 對比調整使用 [控制位址+2]：LW102，調整值為 1~100。
2. 亮度調整使用 [控制位址+3]：LW103，調整值為 1~100。

Important

1. 影像輸入物件僅可使用於 eMT 系列中的 eMT3120 / eMT3150。MT 系列則是 MT8000X 系列才支援此功能。
2. 系統中任何時間點只能有一組影像輸入通道開啟影像輸入。
3. 影像擷取功能不受暫停播放控制之影響，所擷取的圖片仍是外部影像輸入之即時畫面。
4. 推薦的格式類型和解析度：

	1:1	50%
NTSC	720 x 480	360 x 240
PAL	720 x 576	360 x 288

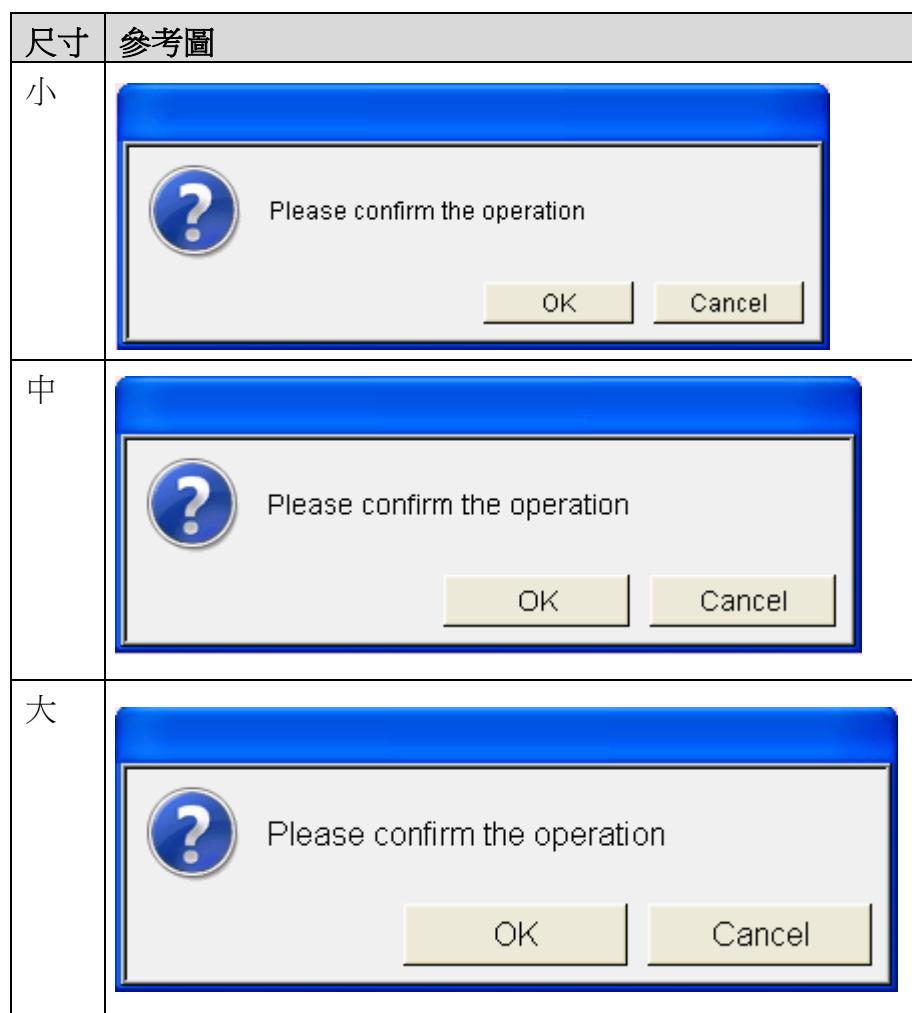
13.32 系統訊息

當物件選擇被使用前須彈出確認視窗或是否可使用遠端登入時，系統會先彈出訊息視窗【操作確認提示】【禁止寫入命令】、【允許寫入命令】的訊息，此三個訊息的內容可在【系統訊息】中編輯。



視窗尺寸

選擇提示的視窗和字體尺寸



操作確認提示

要操作受保護的物件時，顯示訊息向使用者確認這項操作。在設定對話窗中可以設定 [操作確認提示] 中的訊息與 [確認]、[取消] 兩個按鈕上的文字標籤。

在顯示 [確認] 與 [取消] 兩個按鈕上的文字標籤時，需使用相同的字型。另外，只有在 [訊息] 選擇使用文字標籤庫時，[確認] 與 [取消] 兩個按鈕上的文字標籤才允許使用文字標籤庫。



禁止寫入命令

當系統暫存器 LB-9196(唯本地 HMI 支援監控功能)設 ON 時，顯示此訊息。

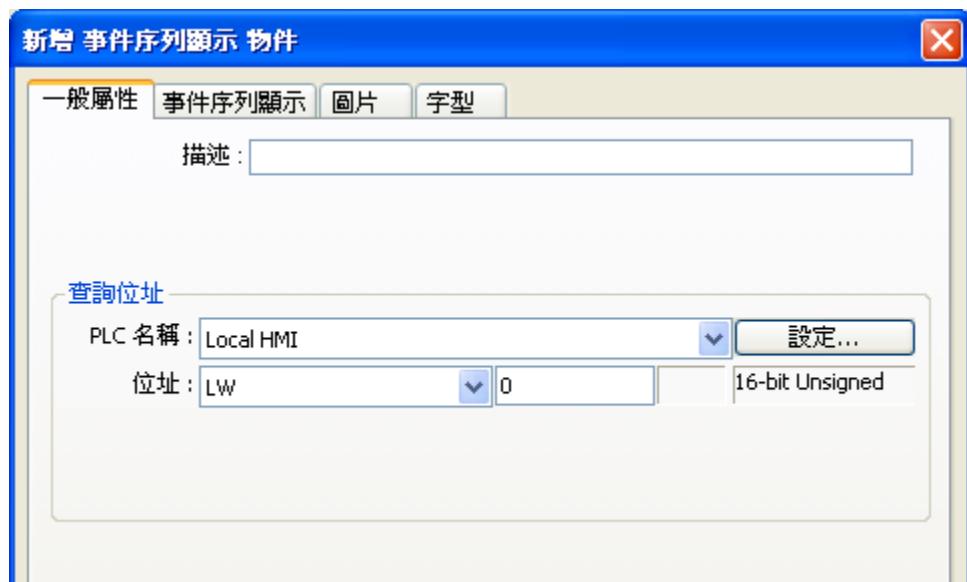
允許寫入命令

當系統暫存器 LB-9196(唯本地 HMI 支援監控功能)設 OFF 時，顯示此訊息。

13.33 事件序列顯示

透過事件序列顯示物件可以讓使用者瀏覽事件記錄。在設備屬性中啟用事件序列後，系統才會開放事件序列顯示物件的設定。

事件序列顯示物件的 [一般屬性] 頁籤設定如下：



一般屬性頁籤可設定事件序列的查詢位址。查詢位址將以連續 19 個字元作為相關控制與數據比對的來源。

名稱	位址	敘述
模式	查詢位址 n	(請參考下列表格)
狀態	查詢位址 n + 1	(請參考下列表格)
開始日期	查詢位址 n + 2 ~ 4	表示： 年.月.日
開始時間	查詢位址 n + 5 ~ 7	表示： 時.分.秒
開始毫秒	查詢位址 n + 8	
類型	查詢位址 n + 9	範圍： 0 ~ 9
設備	查詢位址 n + 10	範圍： 0 ~ 99
使用者名稱	查詢位址 n + 11	範圍： 1 ~ 12
結束日期	查詢位址 n + 12 ~ 14	表示： 年.月.日
結束時間	查詢位址 n + 15 ~ 17	表示： 時.分.秒
結束毫秒	查詢位址 n + 18	

[模式] 內的數值各代表如下：

數值	模式
1	以開始日期 (查詢位址 $n + 2 \sim 4$) 內的數值為查詢依據。當 [狀態] 設為 4 時，需搭配結束日期 (查詢位址 $n + 12 \sim 14$) 作為一個查詢範圍。
2	以開始時間 (查詢位址 $n + 5 \sim 7$) 內的數值為查詢依據。當 [狀態] 設為 4 時，需搭配結束時間 (查詢位址 $n + 15 \sim 17$) 作為一個查詢範圍。
4	以開始毫秒 (查詢位址 $n + 8$) 內的數值為查詢依據。當 [狀態] 設為 4 時，需搭配結束毫秒 (查詢位址 $n + 18$) 作為一個查詢範圍。
8	以類型 (查詢位址 $n + 9$) 內的數值為查詢依據。
16	以設備 (查詢位址 $n + 10$) 內的數值為查詢依據。
32	以使用者名稱 (查詢位址 $n + 11$) 內的數值為查詢依據。

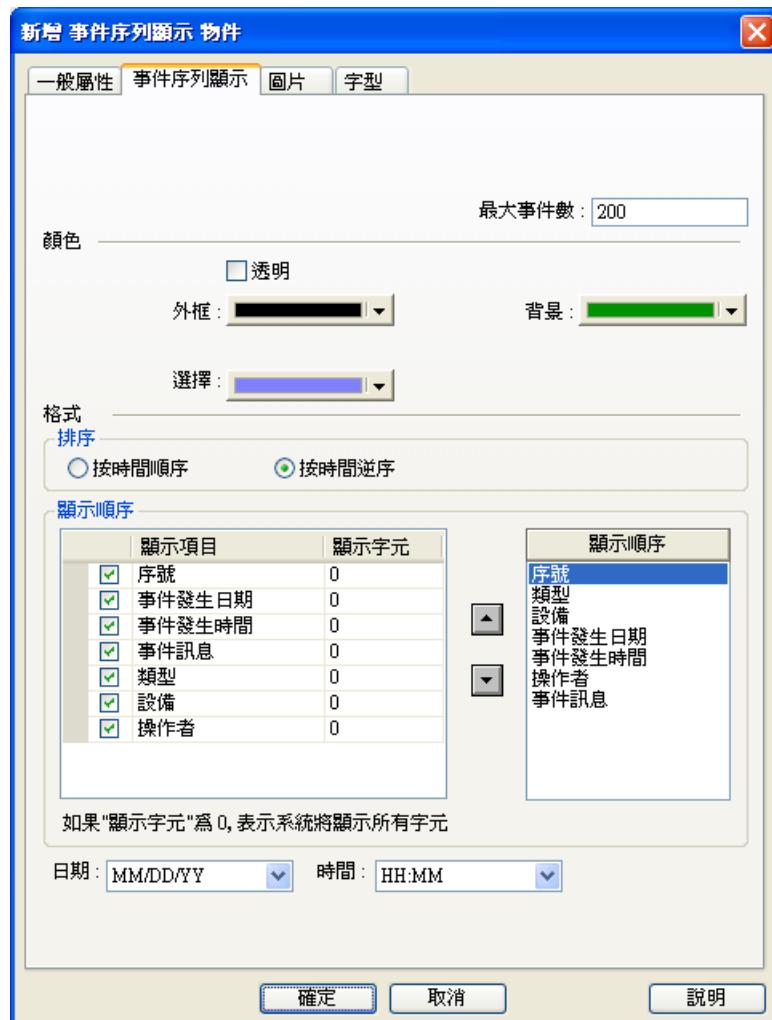
[狀態] 內的數值各代表如下：

數值	模式
0	顯示全部訊息。
1	只顯示等於某特定模式下的訊息。
2	只顯示大於等於某特定模式下的訊息。
3	只顯示小於等於某特定模式下的訊息。
4	只顯示某範圍之特定模式下的訊息。



- [類型]、[設備]、[使用者名稱] 僅支援查詢狀態為 0 (顯示全部) 與 1 (等於)。即無論 [狀態] 的設定為何，[類型]、[設備]、[使用者名稱] 都只會尋找與其數值相等的事件記錄。
- 當 [模式] 或 [狀態] 的數值改變時，才會重新觸發查詢功能。

事件序列顯示物件的【事件序列顯示】頁籤設定如下：



設定	敘述
最大事件數	設定欲顯示的事件數量，若事件觸發數量超出設定範圍，舊事件將依序被新事件覆蓋。
透明	若勾選，將不使用外框與背景的顏色。
外框	設定外框的顏色。
背景	設定背景的顏色。
選擇	設定事件被選取時，標示的框線顏色。
按時間順序	事件將由舊到新依序顯示，最近觸發的事件顯示於底部。
按時間逆序	事件將由新到舊依序顯示，最近觸發的事件顯示於頂部。
顯示順序	設定當事件觸發時，欲顯示的項目與字元數量。若字元設為 0，將顯示項目的所有內容。
日期	設定欲顯示日期資訊的格式。
時間	設定欲顯示時間資訊的格式。

13.34 QR 碼

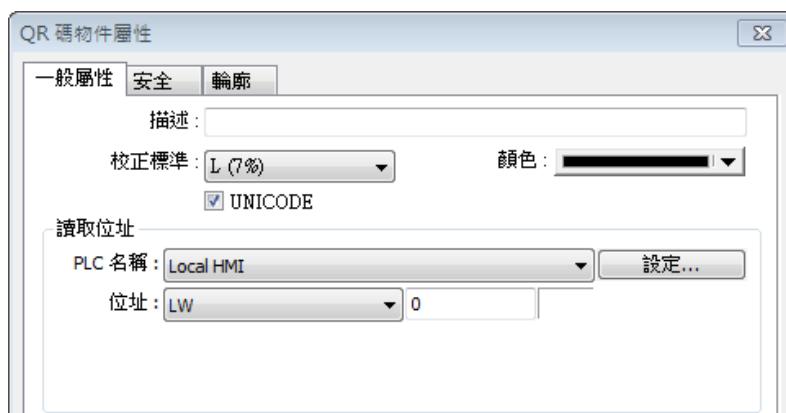
概要

透過輸入資料於指定的字元位址後，可產生相對於資料的 QR 碼於視窗供掃描。

設定



按下工作列上的 [QR 碼] 按鈕後即會開啟 [QR 碼] 物件屬性對話窗，正確設定各項屬性後按下確認鍵，即可新增一個 [QR 碼] 物件。



設定	敘述										
校正標準	QR 碼有容錯能力，QR 碼圖形如果有破損，仍然可以被機器讀取內容，錯誤修正容量分為四種：L、M、Q、H，請見以下說明： <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="2">錯誤修正容量</th> </tr> </thead> <tbody> <tr> <td>L</td><td>7% 的字碼可被修正</td></tr> <tr> <td>M</td><td>15% 的字碼可被修正</td></tr> <tr> <td>Q</td><td>25% 的字碼可被修正</td></tr> <tr> <td>H</td><td>30% 的字碼可被修正</td></tr> </tbody> </table>	錯誤修正容量		L	7% 的字碼可被修正	M	15% 的字碼可被修正	Q	25% 的字碼可被修正	H	30% 的字碼可被修正
錯誤修正容量											
L	7% 的字碼可被修正										
M	15% 的字碼可被修正										
Q	25% 的字碼可被修正										
H	30% 的字碼可被修正										
顏色	設定 QR 碼顯示的顏色。										
Unicode	在預設中，QR 碼的內容是以 ASCII 編碼形式產生。若勾選此選項，則 QR 碼的內容會以 UNICODE 的編碼形式產生。 預設 ASCII 編碼適用於一般英文字母與數字形式的內容；勾選 UNICODE 則可以使用符合 UNICODE 編碼的其他文字，例如中文、韓文等。										
讀取位址	透過輸入資料於指定的字元位址後，可產生相對於資料的 QR 碼於視窗供掃描。可輸入資料的字元長度為 1 ~ 1024。										

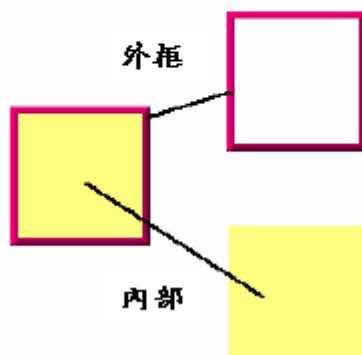
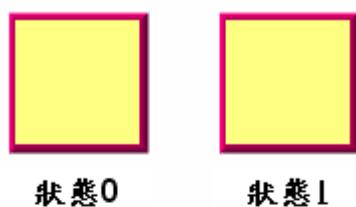
第十四章 向量圖庫與圖片庫的與使用

EasyBuilder 提供向量圖庫與圖片庫的使用，增加元件在視覺上的效果，每個向量圖與圖片最多可包含 256 個狀態。下文將說明如何建立向量圖庫與圖片庫。

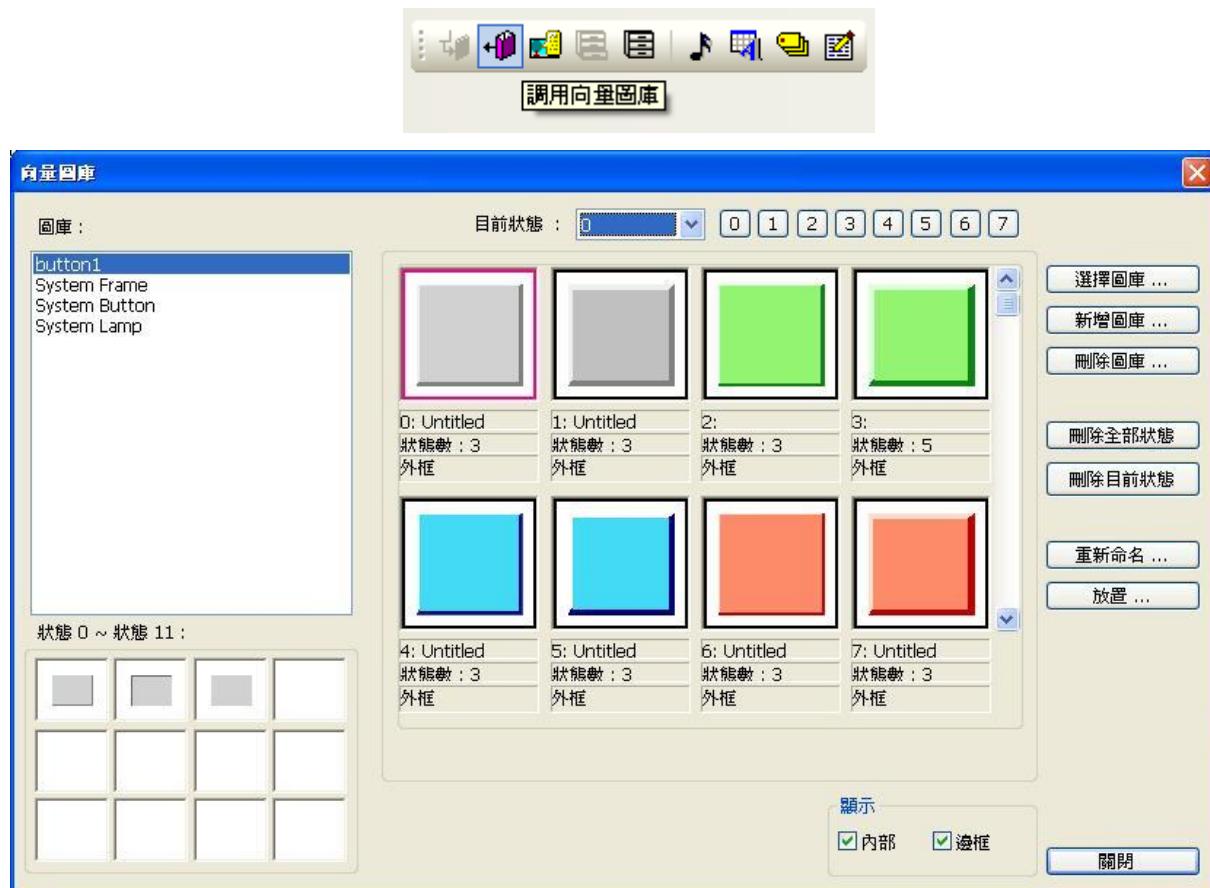
向量圖、圖片庫的使用請參考 《第九章 元件一般屬性》。

14.1 向量圖庫的建立

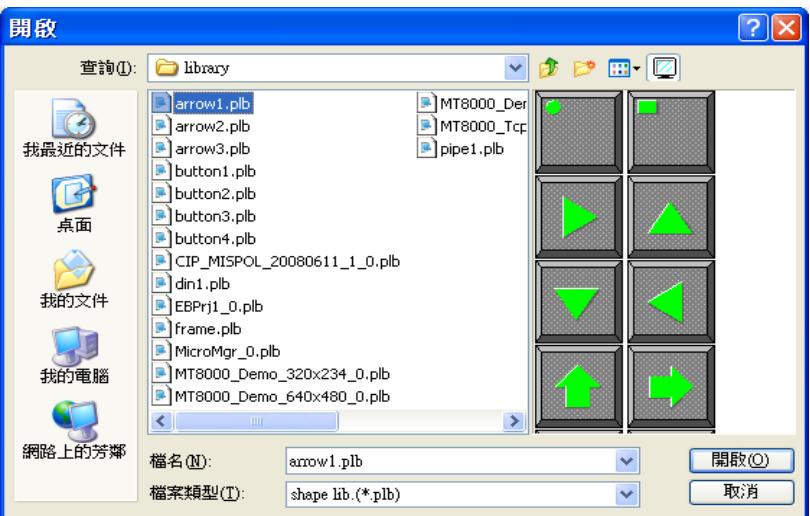
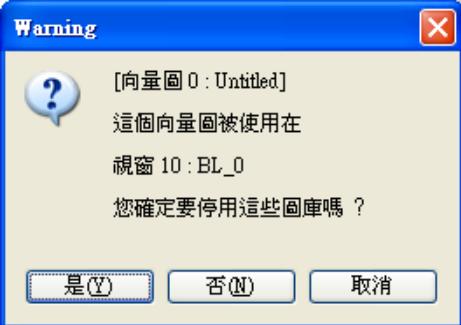
向量圖是由直線、矩形、圓等繪圖元件所構成的圖形；一個完整的向量圖可能具有一個以上的狀態，每個狀態都可包含兩個部分：外框與內部，如下圖所示。

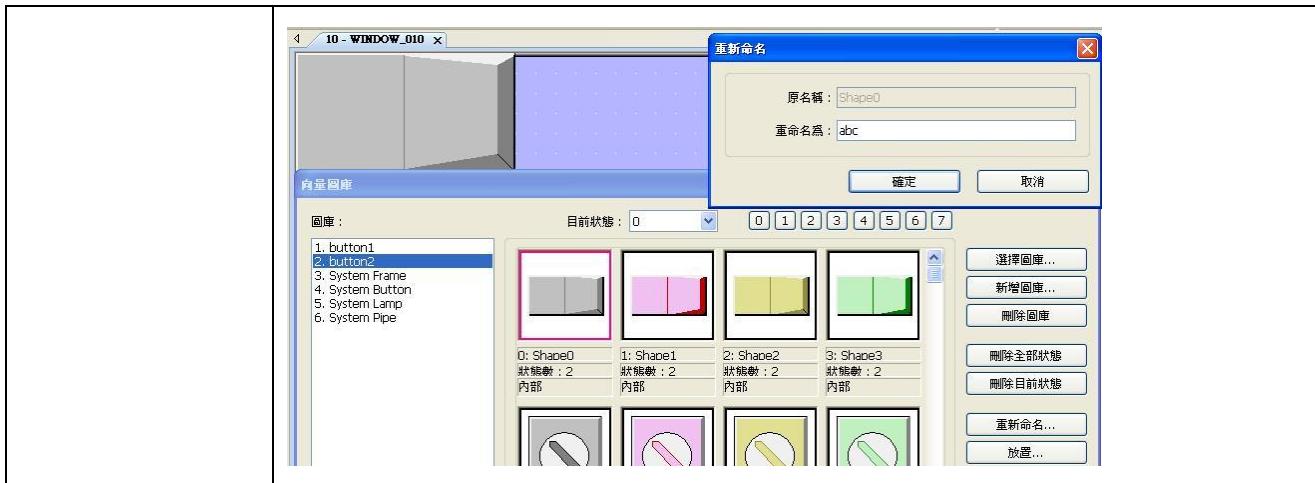


物件可以單獨選擇使用向量圖的外框或內部，或兩者同時使用。在按下工具列的【調用向量圖庫】按鈕，即可進入【向量圖庫】管理對話窗，如下圖所示。



設定	描述
[圖庫]	顯示已加入此工程檔案的向量圖庫，若要選擇使用哪一個向量圖庫只需點選圖庫名稱即可。
[目前狀態]	選擇向量圖目前要顯示的狀態，當視窗中未顯示向量圖時，表示該向量圖不存在，或此向量圖在目前的狀態尚未被定義。
[選擇圖庫]	按下按鈕後會出現下圖的畫面，可選擇要加入此工程檔案的向量圖庫。在視窗的右半部則可先預覽圖庫的內容，再將合適的圖庫加入。

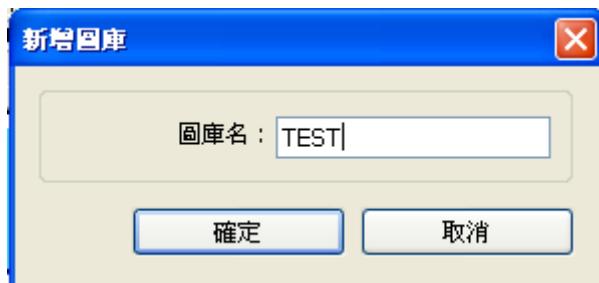
	
[新增圖庫]	按下按鈕後會出現下圖的畫面，可用來增加一個空的向量圖庫。
	
[刪除圖庫]	按下按鈕後可出現下圖的畫面，可將 【圖庫】 中顯示的向量圖庫從此工程檔案中移除。
	
[刪除全部狀態]	用來刪除目前所選擇向量圖的全部狀態。
[刪除目前狀態]	用來刪除目前所選擇向量圖所顯示的狀態。
[重新命名]	按下按鈕後將出現下圖的畫面，可重新命名目前所選擇的向量圖名稱。
	
[放置]	可將目前選擇的向量圖輸出到使用中的視窗上，如下圖所示。



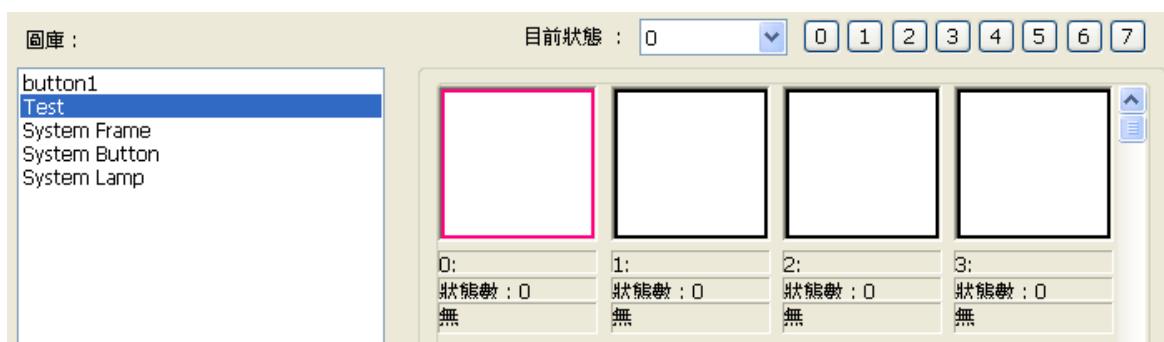
下面說明如何新建立一個新的向量圖庫，並在此圖庫中加入一個具有兩個狀態的向量圖。

步驟一

按下【新增圖庫】後，在對話窗中輸入新的向量圖庫名稱。

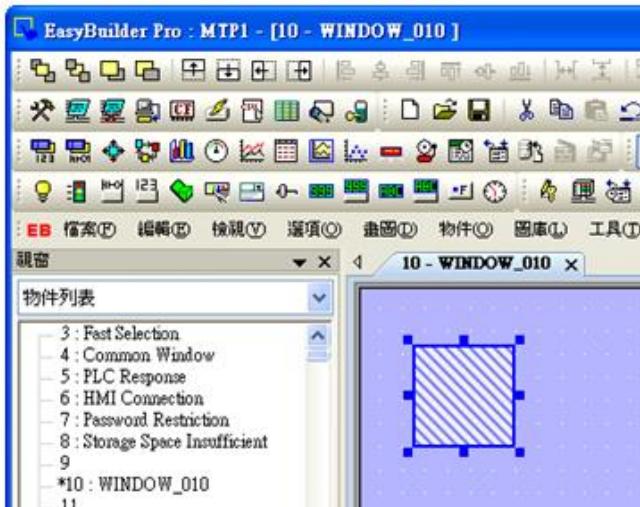


此時可以發現向量圖庫管理對話窗中增加一個新的向量圖庫“TEST”，且此新的向量圖庫中並未包含任何向量圖，如下圖所示。



步驟二

對特定向量圖，加入一個狀態。首先使用繪圖工具在視窗上繪出需要的圖形，並圈選要加到向量圖庫的圖形。



接著在圈選圖形元件後，按下工具列上的【儲存至向量圖庫】按鈕，可以得到下面的對話窗：



設定	描述
[圖庫]	選擇目前的圖形是要加到哪一個向量圖庫，此範例選擇“ TEST ”。
[描述]	向量圖狀態的描述。
[圖形編號]	選擇目前的圖形要加到“ TEST ”向量圖庫中的哪一個編號中。
[狀態]	選擇目前的圖形將作為向量圖的哪一個狀態，此範例是選擇狀態 0 。每個向量圖最多可有 256 個狀態。
[外框]	若勾選此項目，目前的圖形將做為向量圖的外框。
[內部]	若勾選此項目，目前的圖形將做為向量圖的內部。

下圖的資訊顯示“TEST”向量圖庫中編號 0 的向量圖，目前的狀態 (state 0)並未定義任何邊框與內部。



在按下確認鍵後可以發現圖形已經加入到向量圖庫中。由下圖可看出編號 0 的向量圖只具有一個狀態，且外框已被定義。

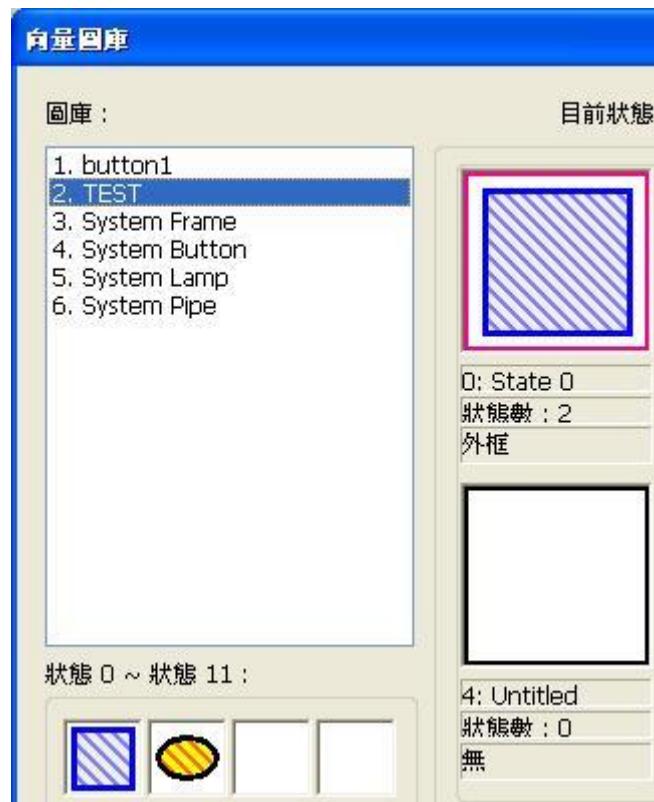


步驟三

使用步驟二相同的方式，但新加入的圖形需選擇做為狀態 1，如下圖所示。



一個完整的向量圖需建立至少 2 個狀態，如下圖所示。



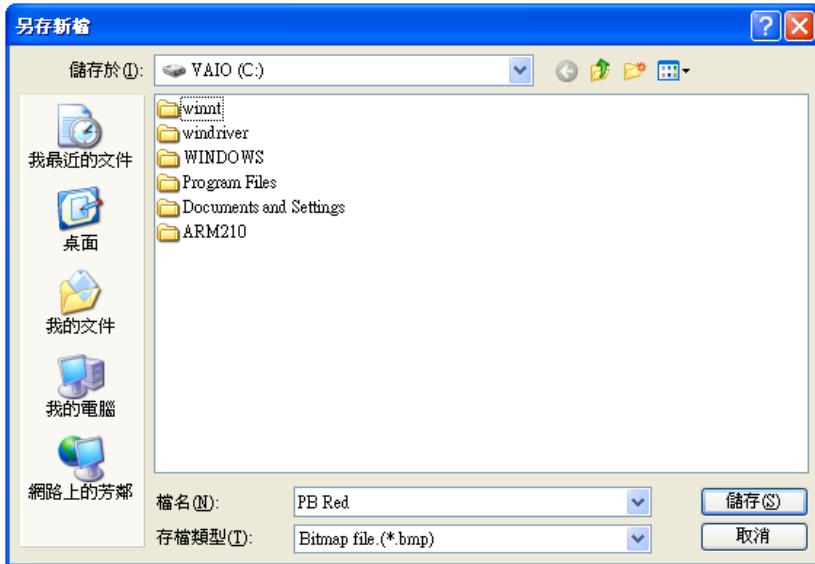
14.2 圖片庫的建立

按下工具列上的工作按鈕後將出現【調用圖片庫】對話，如下圖所示。



設定	描述
【圖庫名】	顯示已加入此工程檔案的圖形庫，要選擇使用哪一個圖形庫只需點選圖庫名稱即可。
【目前狀態】	選擇圖形目前要顯示的狀態，當視窗中未顯示圖形時，表示該圖形不存在，或此圖形在此狀態並未被定義。
【選擇圖庫】	按下按鈕後可出現下圖的畫面，可選擇要加入此工程檔案的圖片庫。在視窗的右半部則可先預覽圖庫的內容，再將合適的圖庫加入。

[新增圖庫]	按下按鈕後可出現下圖的畫面，可用來增加一個空的圖形庫。
[刪除圖庫]	按下按鈕後可出現下圖的畫面，可將 [圖庫] 中顯示的圖片庫從此工程檔案中移除。
[刪除全部狀態]	用來刪除目前所選擇圖片的全部狀態。
[刪除目前狀態]	用來刪除目前所選擇圖片所顯示的狀態。
[重新命名]	按下按鈕後將出現下圖的對話窗，可重新命名目前選擇的圖形。

	
[往前加入新狀態]	在目前所顯示的狀態前加入一個新狀態。
[往後加入新狀態]	在目前所顯示的狀態後加入一個新狀態。
[新增圖形]	在圖形庫中增加一個新的圖形。 
[修改圖形]	利用此項功能可以修改圖片目前的狀態。
[匯出圖形]	可將目前選擇的圖片輸出到指定的位置，如下圖所示，讓使用者可以獲得原始圖片。 



- 圖片庫支援的圖片格式為 *.bmp、*.jpg、*.gif、*.dpg 和 *.png。在圖片庫新增 GIF 格式的圖檔時，如果圖檔屬於動畫類型的檔案時，使用者可以設定動畫的播放次數。如下



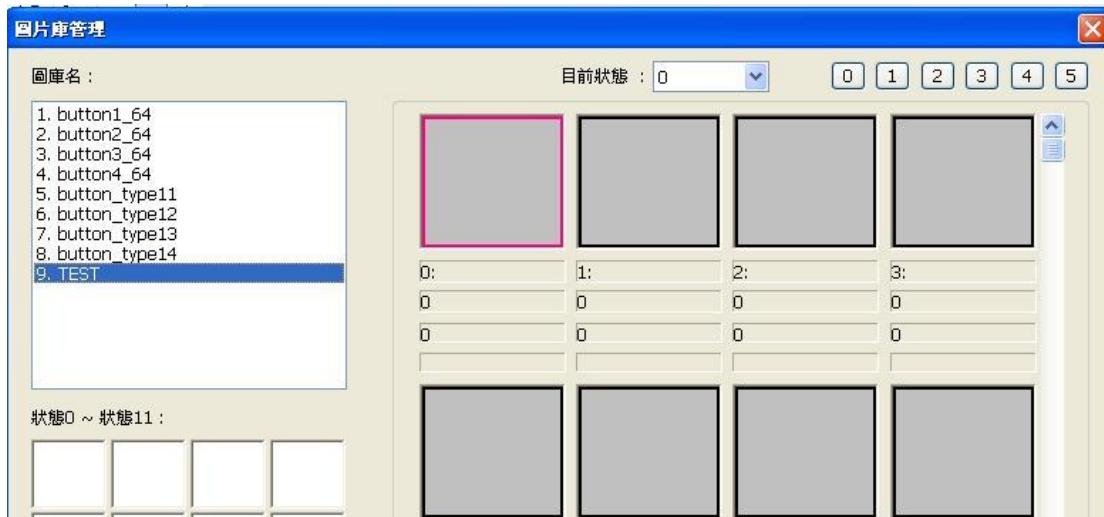
下面說明如何新建立一個新圖片庫，並在此圖庫中加入一個具兩個狀態的圖片。

步驟一

按下【新增圖庫】後，在對話窗中輸入新的圖形庫名稱。

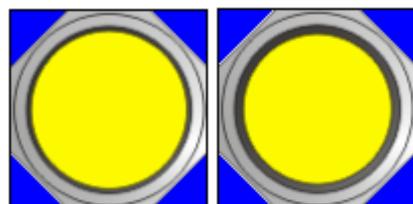


此時可以發現圖片庫管理對話窗中增加一個新的圖片庫“TEST”，且此新的圖片庫中並未包含任何圖片，如下圖所示。



步驟二

先用繪圖工具準備好要加入的圖片；假設現在要將下面的兩個圖片分別用來表示狀態 0 與狀態 1。



首先按下【新增圖片】按鍵，可出現下圖的對話窗，設定圖片編號與圖片名，最後按下【下一步】。



步驟三

在出現下圖的對話窗時，此時需挑選狀態 0 的圖片來源，若勾選**【使用透明色】**，並設定 RGB (250, 0, 0)，會將紅色區域設定為透明色。完成狀態 0 的設定後，因還包含另一個狀態，繼續執行**【下一步】**。



要挑選透明色首先需先勾選**【使用透明色】**，接著使用滑鼠點擊欲作為透明區域的位置，此時會自動顯示作為透明色的**RGB** 值，以上圖為例，實際顯示的圖片如下。

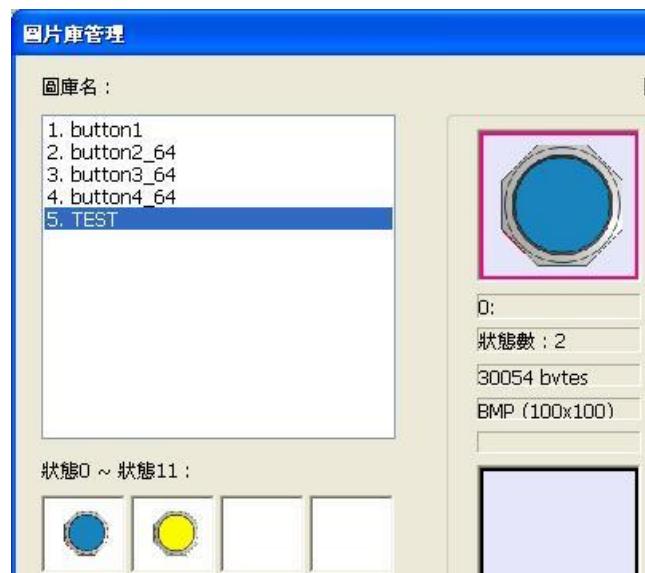


步驟四

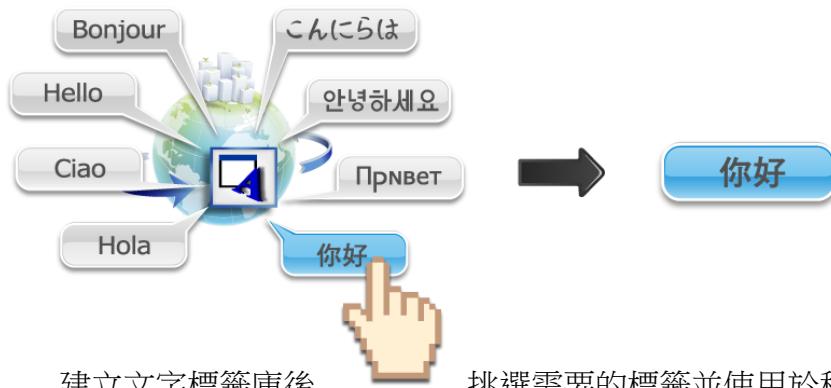
與上一步驟相同，挑選狀態 1 的圖片並設定透明色。最後按下**【完成】**即完成包含兩個狀態圖片的建立。



在完成上述的各項動作後，即建立一個完整的圖片，可參考下圖。這時在圖片管理對話窗中可以發現新加入的圖片“**F Yellow**”，由圖片資訊中也可看出此圖片為 bitmap 形式，且包含兩個狀態。



第十五章 文字標籤庫與多國語言使用

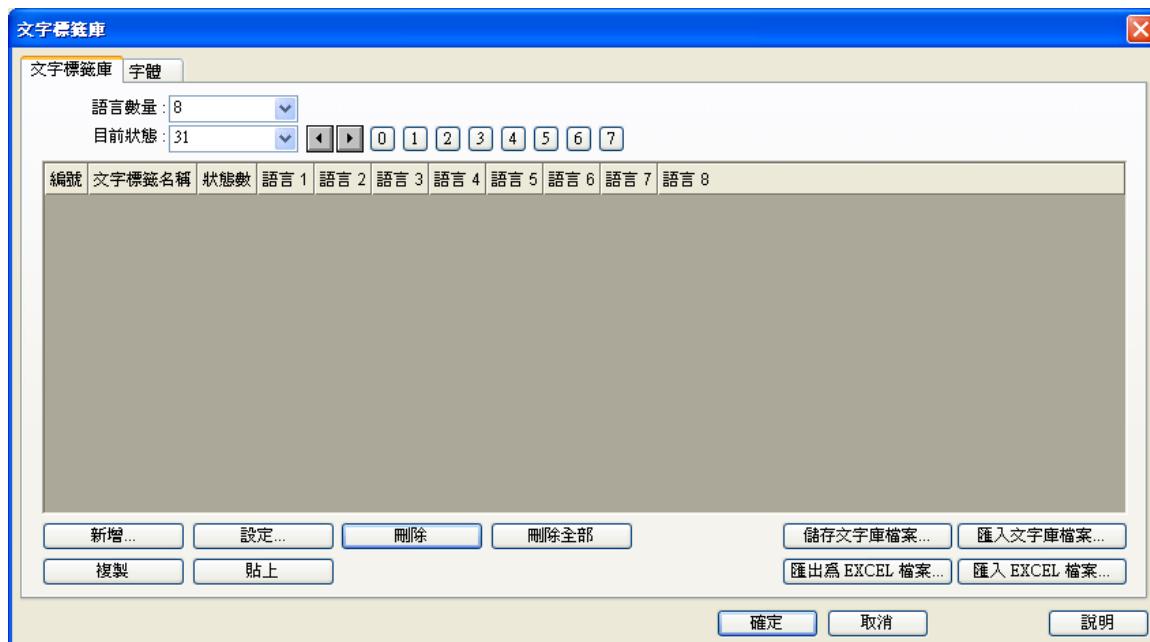


建立文字標籤庫後

挑選需要的標籤並使用於程式中。

15.1 介紹

系統在運作時會依照語言模式設定，於工程檔案中顯示所選擇語言模式相對應的文字。



[語言數量]

EasyBuilder 同時支援 24 種不同語言的文字顯示。

[目前狀態]

一個文字標籤最多可擁有 256 個狀態 (0 ~ 255)。

狀態數量將受 [語言數量] 使用的限制，若使用 1 ~ 3 個語言，每個語言最多可有 256 個狀態，若使用 4 個以上的語言數量，則可用 768 去除以語言數量後，即可得到最多狀態數。例如：語言數量為 24，則 $768/24 = 32$ (狀態數)。

[新增]

新增一筆文字標籤。

[設定]

設定所選文字標籤內容。

[儲存文字庫檔案]

儲存所有文字標籤為 *.lbl 格式檔案。

[匯入文字庫檔案]

將現存文字標籤 *.lbl 檔案匯入文字標籤庫。

[匯出為 EXCEL 檔案]

儲存所有文字標籤為 *.csv, *.xls 或 *.xlsx 格式檔案。

[匯入 EXCEL 檔案]

將現存文字標籤 *.csv, *.xls 或 *.xlsx 檔案匯入文字標籤庫。



- 匯入或匯出 Excel 檔案均不支援 Unicode。

15.2 文字標籤庫的建立

- 開啟【文字標籤庫】»【新增】。



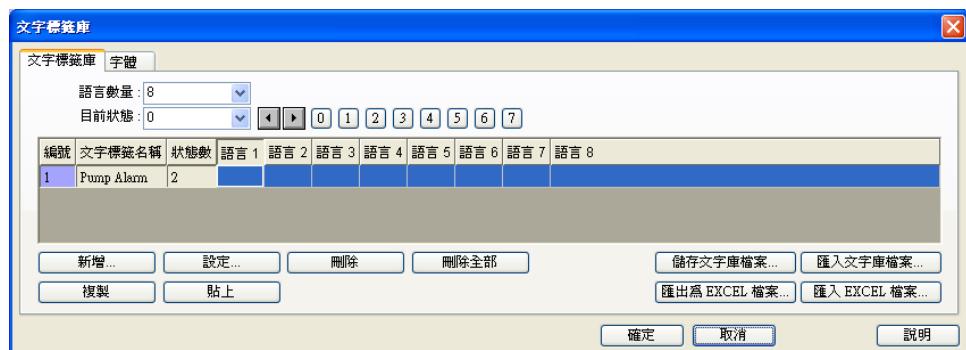
【標籤名】

文字標籤的名稱，請自行定義。

【狀態數】

文字標籤所要表現的狀態數。

- 按下【確定】會顯示一個新的空白標籤，選擇該標籤後按【設定】即可編輯標籤內容。



- 設定相關語言的內容。



15.3 文字標籤庫字體設定

在【文字標籤庫】»【字體】頁籤中可以設定目前已經存在的標籤所包含的語言字型，不同語言可選擇不同的字型：



[字體]

用來設定在使用多國語言文字時，各種語文所使用的字型。

[備註]

每種字型的註釋

15.4 文字標籤庫的使用

當文字標籤庫存在已定義完成的標籤，在物件的【標籤】頁籤中可以勾選【使用文字標籤庫】，在【標籤】的下拉選單會列示出已定義的文字標籤。



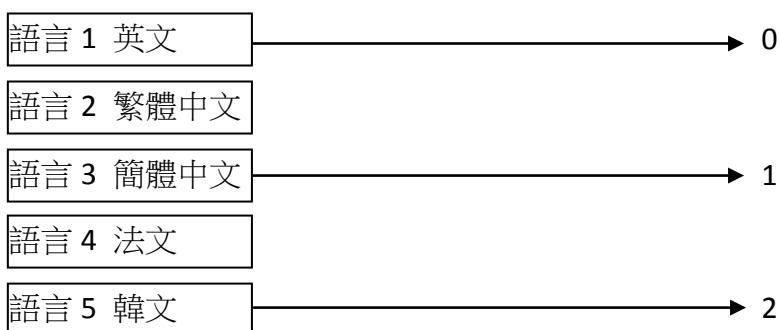
選擇標籤後，可以發現【內容】中的文字來源所顯示為文字標籤的內容，所使用的字型為文字標籤庫的設定內容。請注意，語言 2 ~ 24 的文字屬性設定只有文字尺寸是可被單獨設定的，其餘屬性設定，包含文字顏色、對齊和閃動等，均與語言 1 相同。

15.5 多國語言的使用 (系統暫存器 LW-9134)

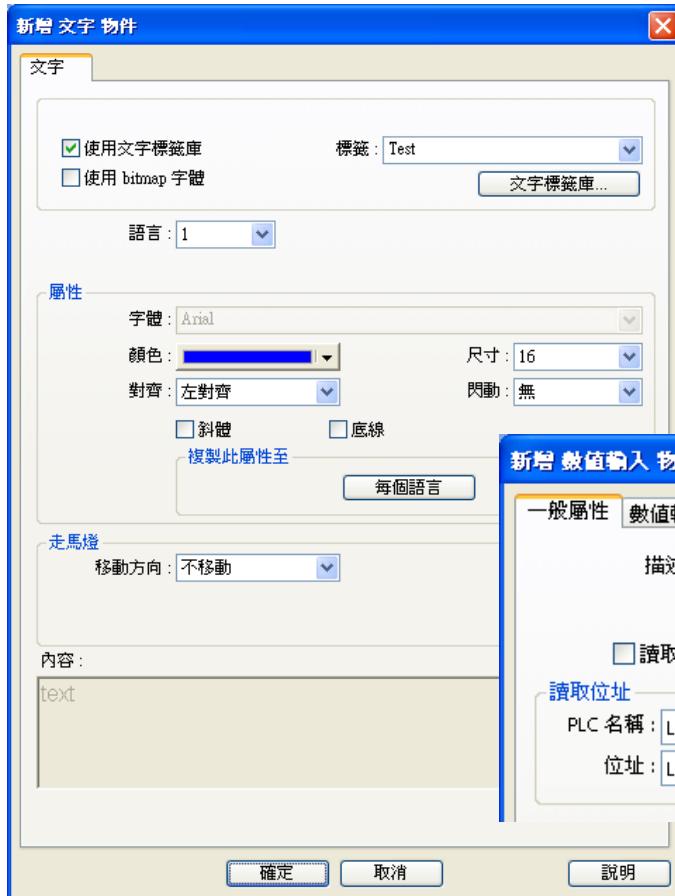
當物件的文字內容要求表現出多國語言的效果時，除了使用文字標籤外，也需搭配系統暫存器 [LW-9134] 的使用。

[LW-9134] 的有效可設定值範圍為 0 ~ 23，不同的數據對應到需顯示的語言，由於 EasyBuilder 可以設定 24 種語言，但是當編譯下載的檔案沒有勾選全部語言時，[LW-9134] 使用方式將有改變：

使用者建立 5 種語言如下，編譯時只勾選語言 1, 3, 5，[LW-9134] 對應數值為



如何使用多國語言：

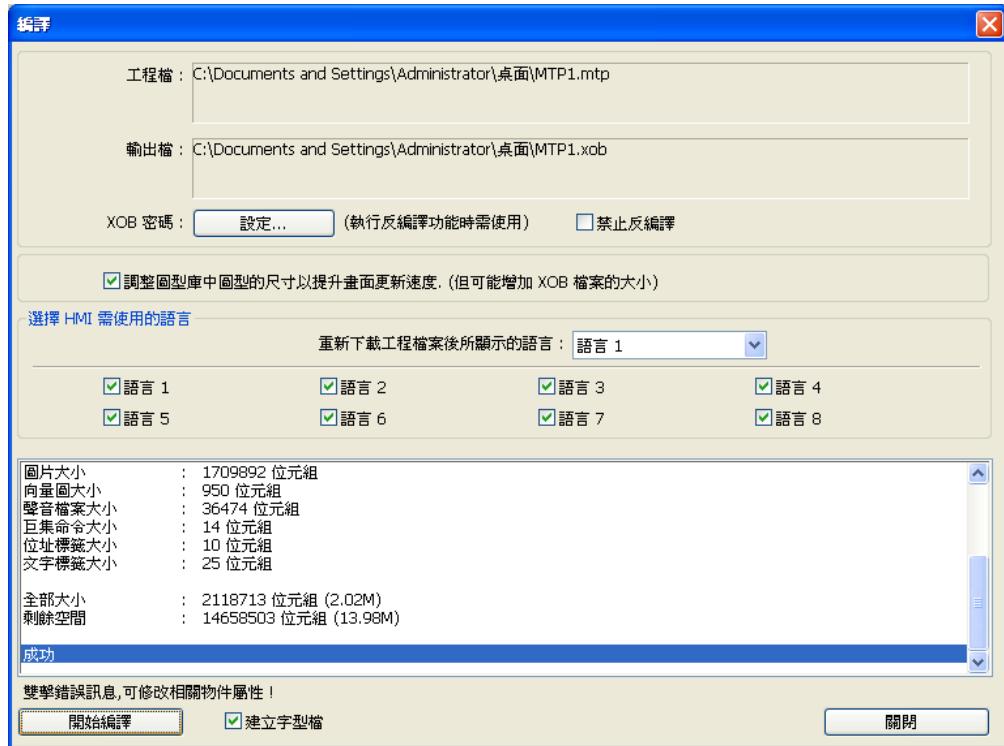


1. 先建立一個【文字物件】，勾選【使用文字標籤庫】。

2. 再建立一個【數值輸入物件】，位址設為系統暫存器 [LW-9134]。



編譯時勾選擬需要並已定義的語言：



模擬如下，當更改 [LW-9134] 的內容時，即可變換文字物件所顯示的內容：

English

LW9134 : 目前所使用的語言

0

简体中文 (SIMPLE)

LW9134 : 目前所使用的語言

2

한국어 웹 (KOREAN)

LW9134 : 目前所使用的語言

4

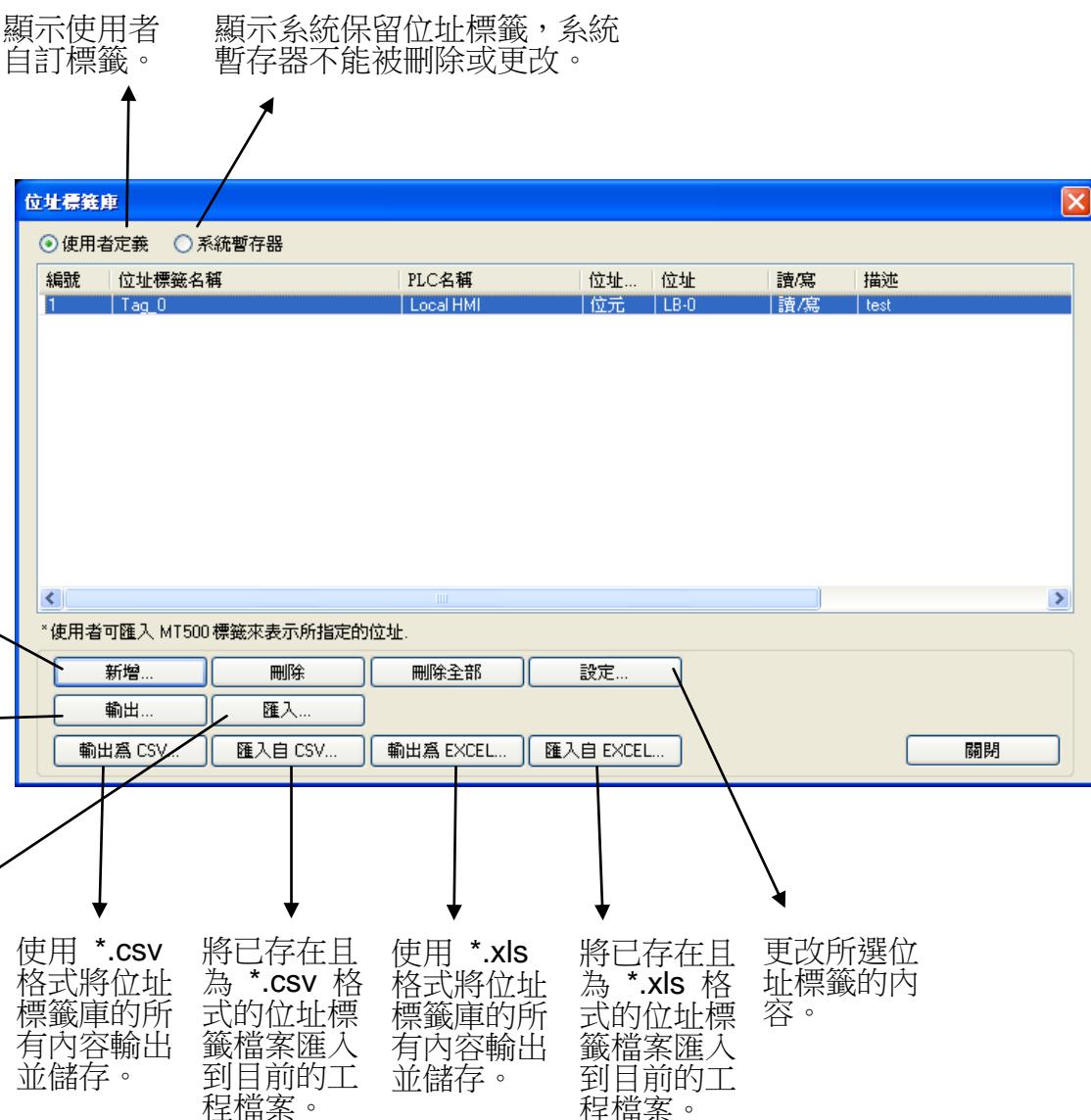


下載範例程式前，請先確定已連上網路線。

第十六章 位址標籤庫的建立與使用

16.1 位址標籤庫的建立

一般來說，建議使用者在程式設計開始時，先將常用的位址定義在位址標籤庫中，除了可以省去繁複的位址輸入外，也可以增加物件位址資訊的可讀性。



若勾選 [使用 UTF-8 格式匯出 CSV 檔案]，則所匯出的 CSV 格式檔案將為 UTF-8 格式，反之則為 ANSI。

按下【新增】後，即可設定相關屬性



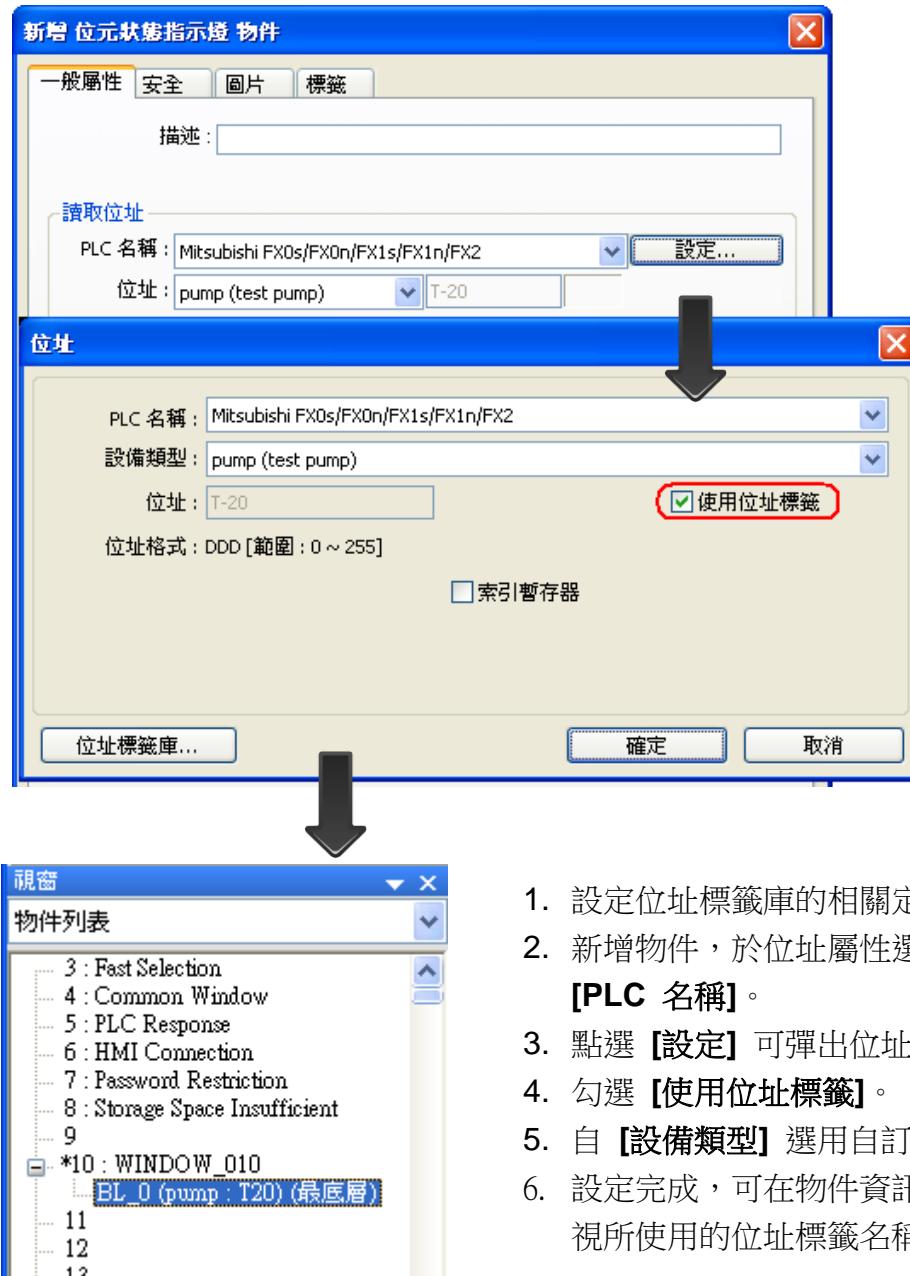
按下【確定】後，可在【使用者定義】位址標籤中發現一筆新增的位址標籤。

位址標籤庫						
<input checked="" type="radio"/> 使用者定義		<input type="radio"/> 系統暫存器				
編號	位址標籤名稱	PLC名稱	位址...	位址	讀/寫	描述
1	pump	Mitsubishi FX0s/FX0n/FX1s/FX1n/FX2	位元	T-20	讀/寫	test pump

*使用者可匯入 MT500 標籤來表示所指定的位址。

新增... 刪除 刪除全部 設定...
輸出... 匯入...
輸出為 CSV... 匯入自 CSV... 輸出為 EXCEL... 匯入自 EXCEL... 關閉

16.2 位址標籤庫的使用



1. 設定位址標籤庫的相關定義。
2. 新增物件，於位址屬性選擇相關 **[PLC 名稱]**。
3. 點選 **[設定]** 可彈出位址設定視窗。
4. 勾選 **[使用位址標籤]**。
5. 自 **[設備類型]** 選用自訂標籤。
6. 設定完成，可在物件資訊樹狀圖檢視所使用的位址標籤名稱。

第十七章 配方資料傳送

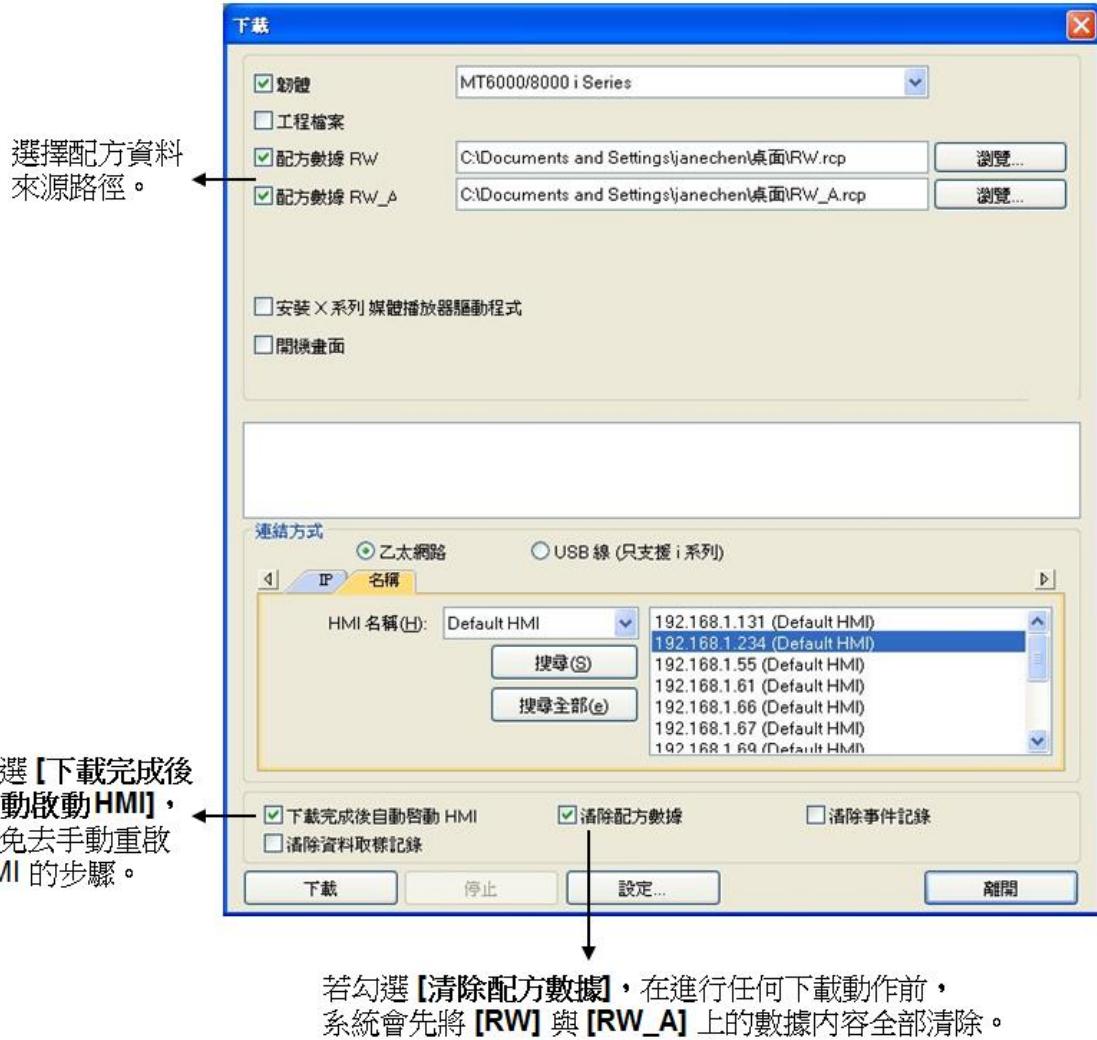
所謂配方資料是指存在 **RW** 與 **RW_A** 位址上的數據，讀寫這些位址的方式與讀寫一般字元位址的方式並無不同，配方資料的特性在於關機後這些數據將保存在 **HMI** 內，重新開機後 **RW** 與 **RW_A** 位址上的數據將維持前一次記錄的內容。

RW 位址可儲存的配方資料大小為 **1024 KB**，而 **RW_A** 位址則為 **128 KB**，使用者可以透過 **SD 卡**、**USB 碟**、**USB 線**或**乙太網路**更新配方資料，並利用這些資料更新 **PLC** 上的數據，同樣地也可以上傳配方資料至電腦；此外，使用者亦可以將 **PLC** 上的數據保存在配方資料中。下文將針對配方資料的各種操作做說明。



17.1 使用乙太網路或 USB 線更新配方資料

透過 Project Manager 的 [下載] 功能，勾選 [RW] 與 [RW_A] 後選擇要儲存的檔案路徑。下載成功後重新啟動 HMI，即可更新 RW 與 RW_A 的內容。



17.2 使用 SD 卡或 USB 碟更新配方資料

在 Project Manager 點選 [建立儲存在 SD 卡與 USB 中的下載資料]。將 SD 卡或 USB 碟插入電腦之後，點選 [瀏覽] 指定檔案資料所要存放的路徑名稱，之後點選 [建立]，EasyBuilder 會自動把所要建立的來源資料檔案寫入到 SD 卡或 USB 碟裡。



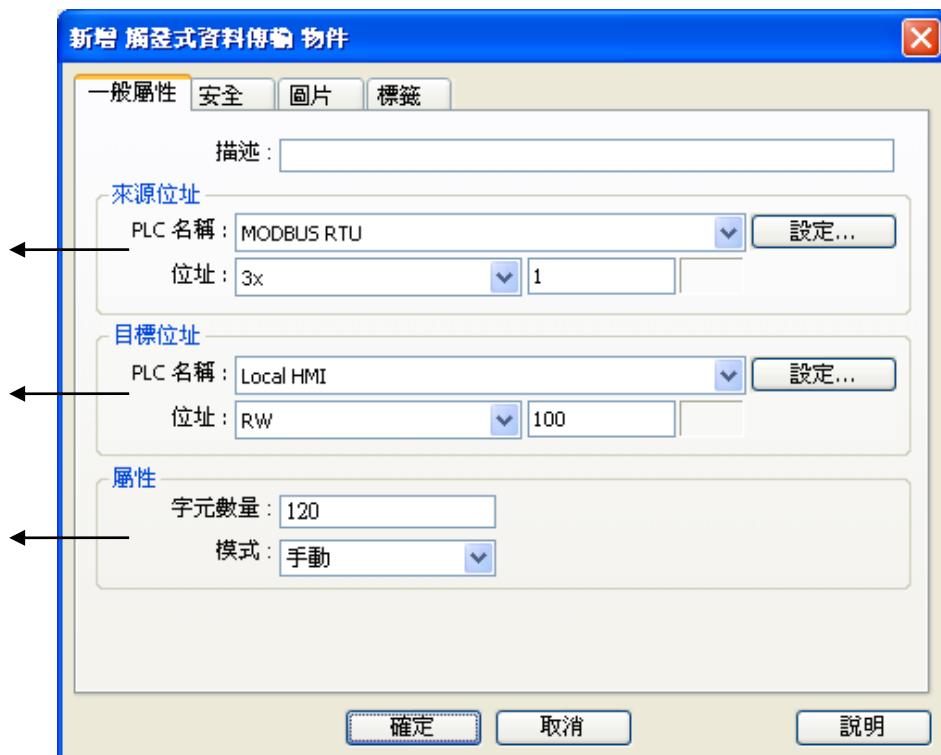
選擇配方資料來源路徑。



- 建立完成後將可發現兩個新資料夾，分別是 history 和 mt8000。mt8000 是存放工程檔案的資料夾，而 history 則是存放配方資料及資料取樣/事件記錄的資料夾。

17.3 配方資料傳輸

可以藉由 **【觸發式資料傳輸】** 物件，將配方資料傳送到特定位址；也可以將特定位址的數據保存在 **[RW]** 與 **[RW_A]** 中。



17.4 配方資料儲存準則

為了延長 HMI 上的記憶體使用壽命，系統以**每隔 1 分鐘**的時間間隔將配方資料保存在 HMI 上，且為了避免配方資料在兩次儲存動作間因關機而造成資料的流失，EasyBuilder 提供系統暫存器 **[LB-9029: 強迫儲存配方資料到 HMI]**，只需對其送出 ON 的訊號，系統即會執行一次配方資料儲存動作。另外如果對 **[LB-9028: 重置配方資料]** 送出 ON 的訊號，則會將所有的配方數據清除。

第十八章 巨集指令說明

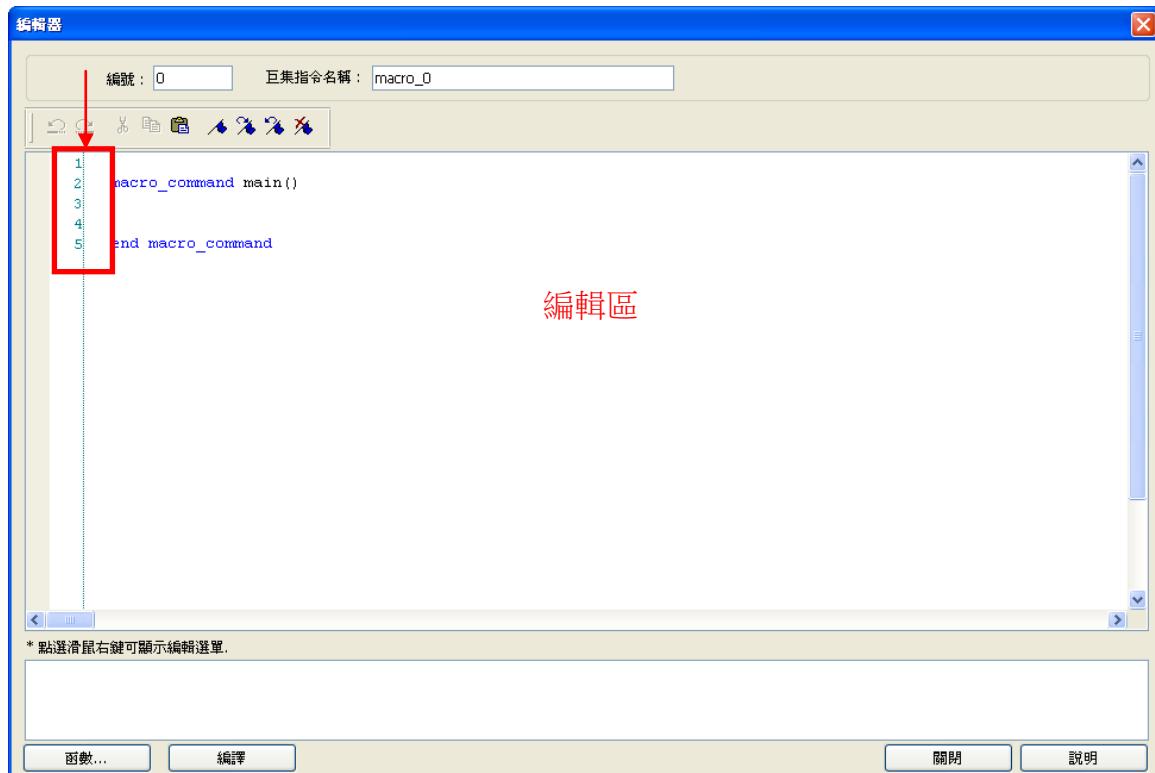
巨集指令提供了應用程式之外所需的附加功能。在 HMI 人機界面運行時，巨集指令可以自動的執行這些命令。它可以擔負執行譬如複雜的運算、字串處理，和使用者與工程之間的交流等功能。本章主要介紹巨集指令的語法、如何使用和編輯方法等功能。希望通過本章的說明，能夠使各位能夠快速的掌握 EasyBuilder 軟體提供的強大的巨集指令功能。

18.1 巨集指令編輯器功能使用說明

1. 巨集指令編輯器提供下列新功能：
 - a. 顯示行號
 - b. 復原 (Undo) / 重複 (Redo)
 - c. 剪下 (Cut) / 複製 (Copy) / 貼上 (Paste)
 - d. 全選 (Select All)
 - e. 建立 / 取消書籤 (Toggle Bookmark) / 上一個書籤 (Previous Bookmark) / 下一個書籤 (Next Bookmark) / 清除全部書籤 (Clear All Bookmarks)
 - f. 程式碼摺疊 (Toggle All Outlining)
 - g. 安全 -> 啟用執行條件
 - h. 週期執行
 - i. 當 HMI 啟動時即執行一次

以下將詳細描述如何使用各項功能。

2. 打開巨集指令編輯器，可以看到編輯區左邊將自動顯示行號。



3. 編輯區中按滑鼠右鍵，呼叫出右鍵選單如下圖：



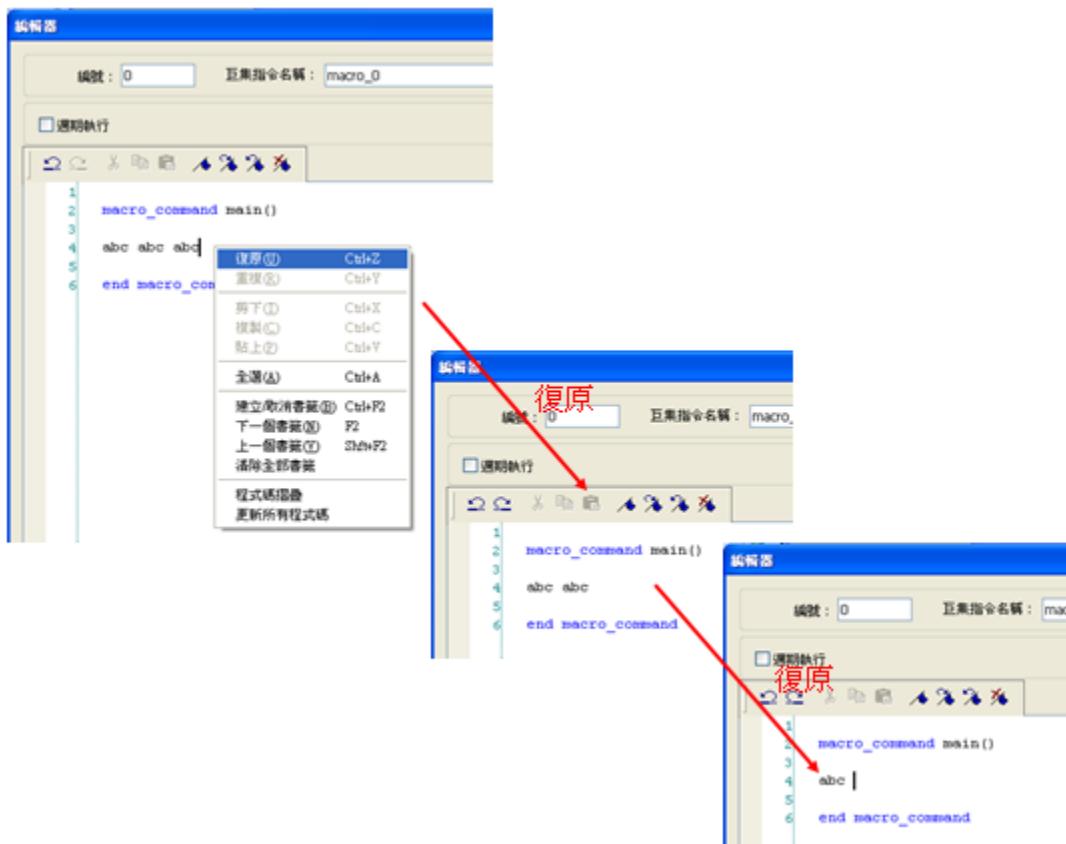
目前的狀態無法使用的功能將顯示灰色。例如必須在編輯區中選取一段文字才會開啟複製功能，因此未選取任何文字的狀態下，複製功能暫不開啟。

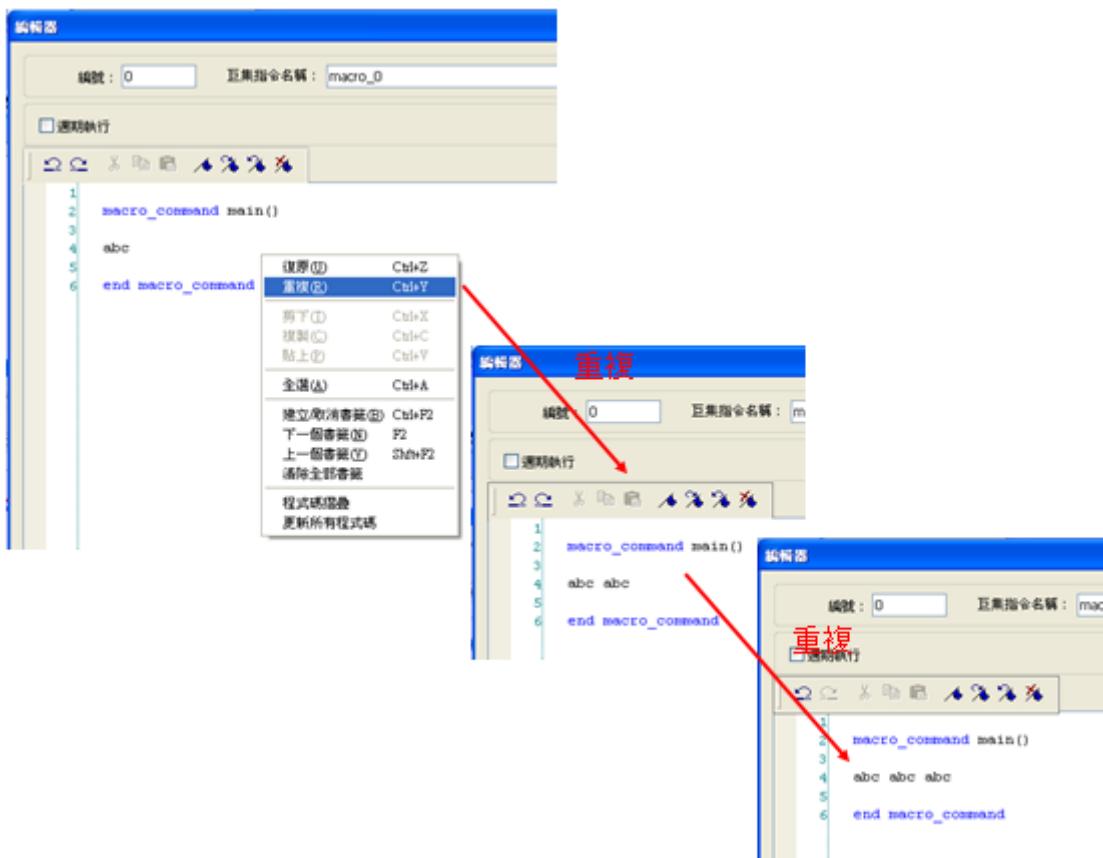
提供快速鍵，如選單內所提示。

4. 編輯區上方有工具列，提供【復原】、【重複】、【剪下】、【複製】、【貼上】、【建立/取消書籤】、【下一個書籤】、【上一個書籤】、【清除全部書籤】等按鈕方便快速選取。

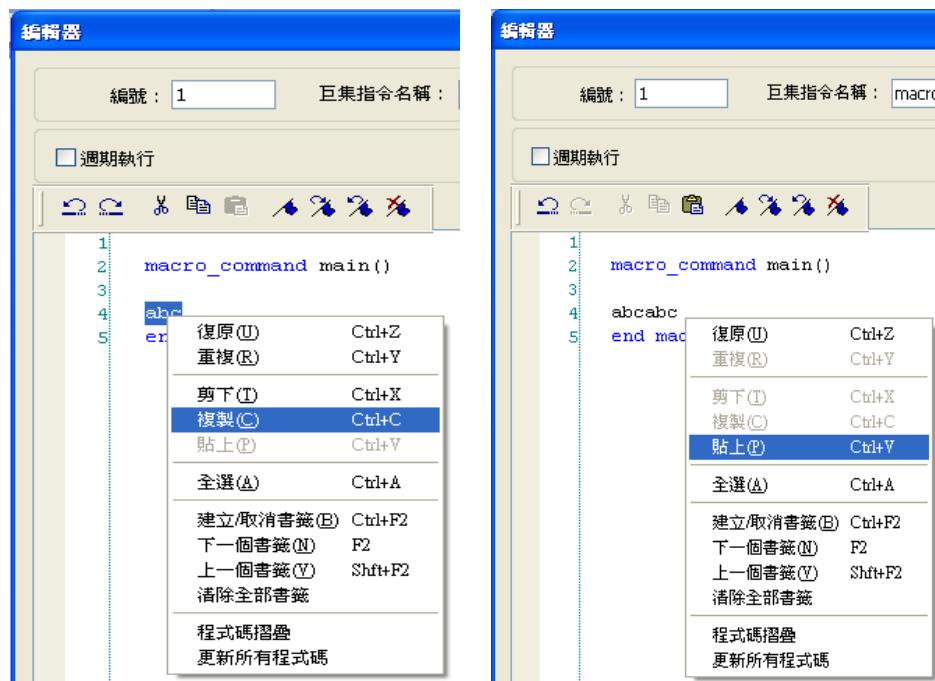


5. 改變編輯區內容將開啟【復原】功能，使用者執行復原後可用【重複】復原。使用者可從右鍵選單或是利用熱鍵 (Undo : Ctrl+Z , Redo : Ctrl+Y) 執行此功能。

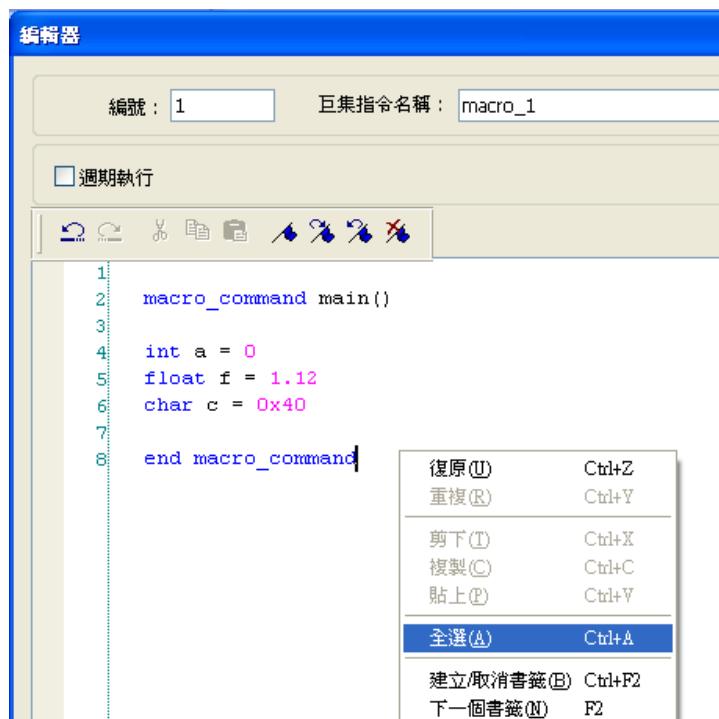




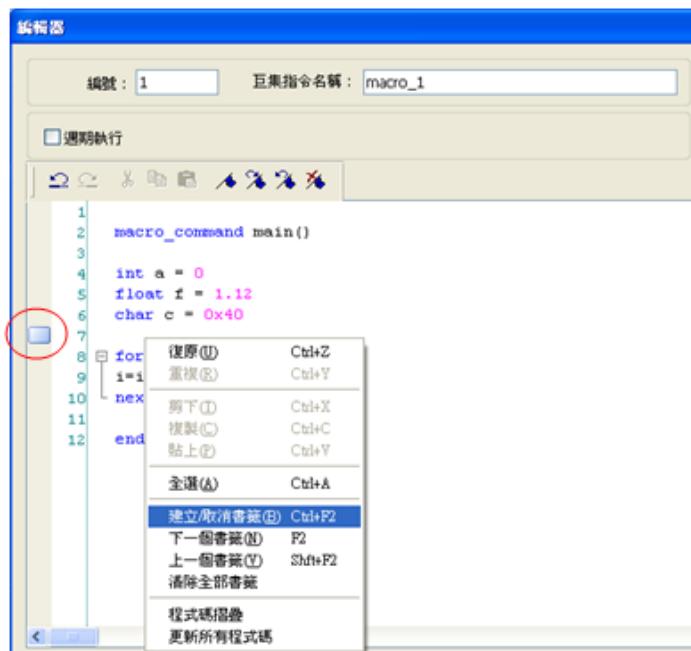
6. 在編輯區選取一段文字後可進行【剪下】、【複製】，之後可用【貼上】將選取的文字貼上。



7. 選擇 **【全選】** 可選取編輯區全部內容。

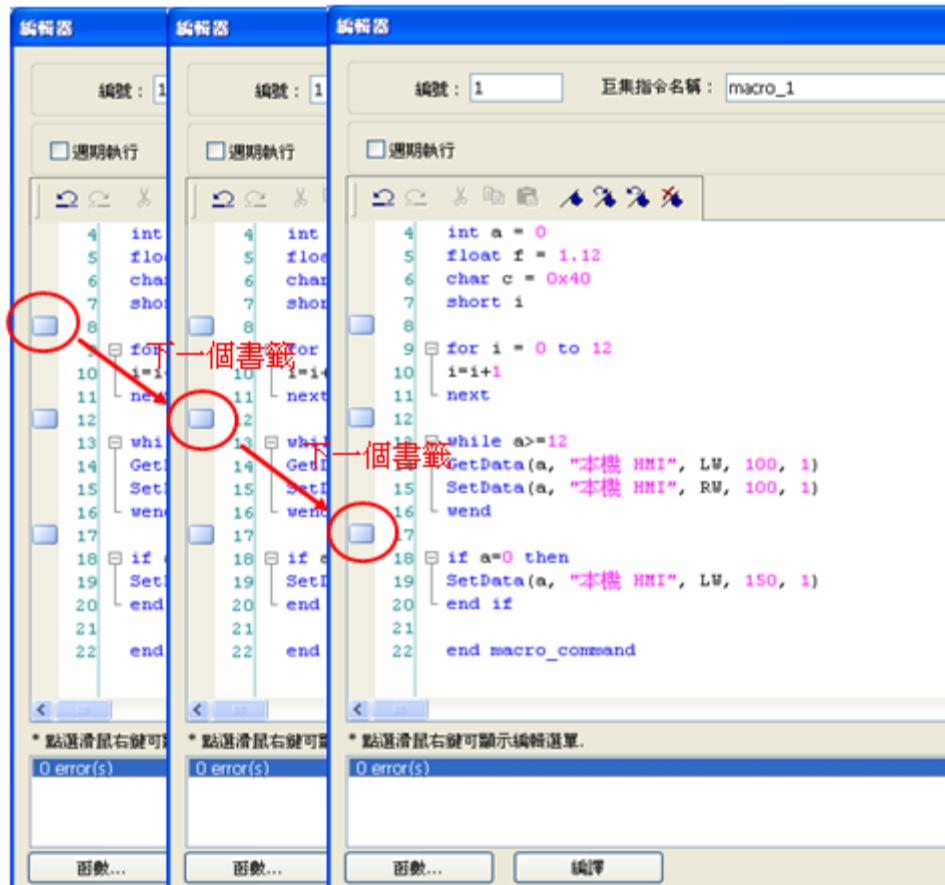


8. 當程式碼很長的時候，為方便使用者閱讀，提供了書籤功能。下面說明如何使用此功能。
- 將游標移至編輯區中想要插入書籤的位置，按右鍵，選擇 **【建立 / 取消書籤】**。編輯區左邊將會看到一個代表書籤的藍色小方塊。



- 若游標所在位置已存在書籤，選擇 **【建立 / 取消書籤】** 可將其關閉，反之，選擇 **【建立 / 取消書籤】** 可將其開啟。

- c. 右鍵選擇【下一個書籤】游標將會移至下一個書籤所在位置。選擇【上一個書籤】游標將會移至上一個書籤所在位置。



- d. 選擇【清除全部書籤】將關閉所有書籤。

9. 巨集指令編輯器提供程式碼折疊功能，方便使用者瀏覽程式碼。所謂程式碼摺疊，是指編輯器可將屬於同一區塊的程式碼隱藏起來，被隱藏起來的程式碼在編輯區裡會顯示成 。編輯區左側會顯示樹狀圖，使用者可按下 隱藏程式區塊，按下 展開程式區塊。如下圖所示：

```

Macro ID : 13
Macro ID : 13
Macro ID : 13

26 GetData(OK[0], "Local HMI", LB, 0, 8)
27 //>>>>>>>>
28 for J =0 to 3
29 //>>>>>>>> set gnCurrentMCount= 1
30 SetData(gnCurrentMCount, "Local HMI", LB, 0, 8)
31 //>>>>>>>> set gnCurrentMCount= 2
32 SetData(gnCurrentMCount, "Local HMI", LB, 0, 8)
33 //>>>>>>>> set gnCurrentMCount= 3
34 SetData(gnCurrentMCount, "Local HMI", LB, 0, 8)
35 //>>>>>>>> next J
36 if OK[K]==1 then
37   K = K + 1
38   SetData(K, "Local HMI", LB, 0, 8)
39 else
40   K = K + 1
41 end if
42 //>>>>>>>> set gnCurrentMCount= 4
43 SetData(gnCurrentMCount, "Local HMI", LB, 0, 8)
44 //>>>>>>>> set gnCurrentMCount= 5
45 SetData(gnCurrentMCount, "Local HMI", LB, 0, 8)
46 next J
47 //>>>>>>>> assert >>>>>>
48 //GetData(gnmacroID, "Local HMI", LB, 0, 8)
49 gnmacroID = gnCurrentMID

```

10. 右鍵選擇 [程式碼摺疊] 可展開所有程式碼區塊。

```

Macro ID : 13
Macro ID : 13

26 GetData(OK[0], "Local HMI", LB, 0, 8)
27 //>>>>>>>>
28 for J =0 to 3
29 //>>>>>>>> set gnCurrentMCount= 1
30 SetData(gnCurrentMCount, "Local HMI", LB, 0, 8)
31 //>>>>>>>> set gnCurrentMCount= 2
32 SetData(gnCurrentMCount, "Local HMI", LB, 0, 8)
33 //>>>>>>>> set gnCurrentMCount= 3
34 SetData(gnCurrentMCount, "Local HMI", LB, 0, 8)
35 //>>>>>>>> next J
36 if OK[K]==1 then
37   K = K + 1
38   SetData(K, "Local HMI", LB, 0, 8)
39 else
40   K = K + 1
41 end if
42 //>>>>>>>> set gnCurrentMCount= 4
43 SetData(gnCurrentMCount, "Local HMI", LB, 0, 8)
44 //>>>>>>>> set gnCurrentMCount= 5
45 SetData(gnCurrentMCount, "Local HMI", LB, 0, 8)
46 next J
47 //>>>>>>>> assert >>>>>>
48 //GetData(gnmacroID, "Local HMI", LB, 0, 8)
49 gnmacroID = gnCurrentMID

```

11. 有時候程式碼區塊可能會誤判。這種誤判起因於編輯器沒有辦法區分當前輸入的關鍵字是否存在於注釋中。例如：

編號 : 0 巨集指令名稱 : ma

週期執行

```
1 macro_command main()
2
3
4 if 1 then
5 // if 1 then
6 end if
7
8 end macro_command
```

使用者可以從右鍵選單選擇【更新所有程式碼】來更正這個錯誤。

編號 : 0 巨集指令名稱 :

週期執行

```
1 macro_command main()
2
3
4 if 1 then
5 // if 1 then
6 end if
7
8 end macro_command
```

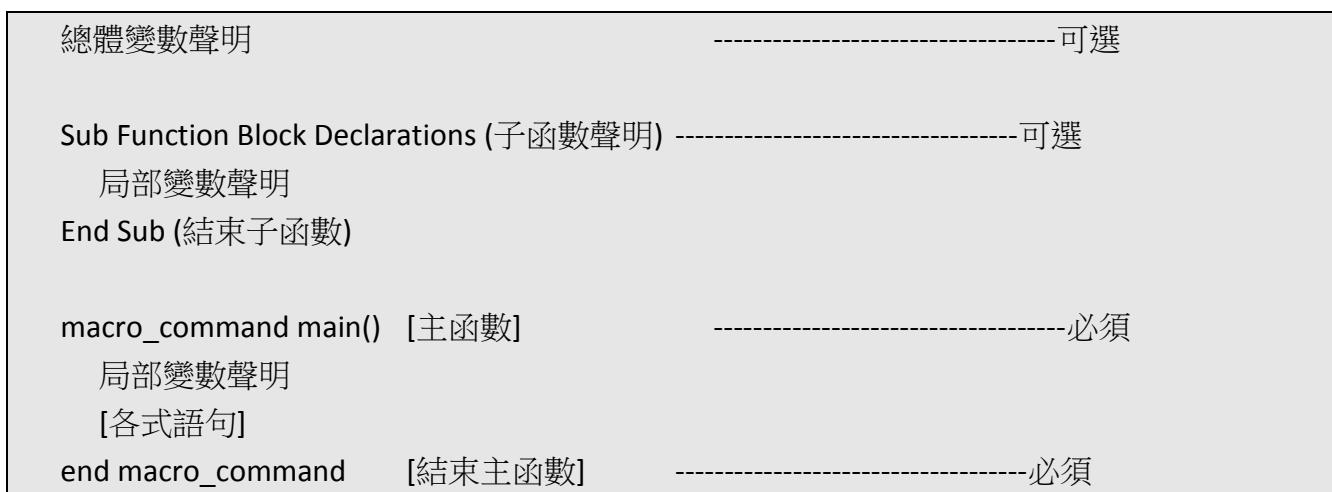
12. 包圍在特定關鍵字內的程式碼稱為一程式碼區塊。內定的程式碼區塊如下列：

- a. 子函數： sub – end sub
- b. 迴圈語句：
 - (1) for – next
 - (2) while – wend
- c. 邏輯運算語句：
 - (1) if – end if
- d. 多重判斷語句： select case – end select

18.2 巨集指令的結構

巨集指令是由各種語句組成的。這些語句包含常數、變數和各種運算符號。這些語句放置在特定的順序位置以便執行後達到一個希望的執行結果。

巨集指令的結構一般為以下格式：



一個巨集指令必須有一個且只有一個主函數，用來開始巨集指令的執行。格式為：

macro_command 函數名稱()

end macro_command

變數聲明必須放在巨集指令語句的前面，否則如果語句放置在變數聲明的前面，將會造成巨集指令無法編譯通過。

局部變數一般用在巨集指令主函數或者自定義的子函數中。它的合法性只在指定的函數中有效。

總體變數一般是定義在所有巨集指令函數的前面，且它在整個巨集指令中均具有有效性。當局部變數和總體變數被定義為相同的名稱時，只有局部變數有效。

下面就是一個簡單的巨集指令，其中就包含了變數聲明和函數呼叫。雙斜線 "://" 代表程式註解，在它後面的文字不會被執行。

```
macro_command main()
    short pressure = 10                                // 局部變數聲明
    SetData(pressure, "Allen-Bradley DF1", N7, 0, 1)   // 函數呼叫
end macro_command
```

18.3 巨集指令的語法

18.3.1 常數和變數

18.3.1.1 常數

常數是一個可以被各式語句直接使用的固定的資料。有如下格式：

常數類型	使用說明	舉例
十進位整數		345, -234, 0, 23456
十六進位數	必須以 0x 開頭	0x3b, 0xffff, 0x237
字元型	字元必須使用單引號，字串使用雙引號	'a', "data", "函數名稱"
布林型		true, false

下面即為一個簡單的常數使用的範例。

```
macro_command main()
    short A, B      // 聲明 A 和 B 為短整型變數
    A = 1234
    B = 0x12      // 1234 和 0x12 即為常數
end macro_command
```

18.3.1.2 變數

變數是一個代表著各種資料的名稱。在巨集指令中，這些資料可以隨著巨集指令語句執行的結果改變而改變。

變數的命名規則

1. 必須以英文字母開頭
2. 變數名稱長度不超過 32 個字元
3. 系統保留暫存器名稱不能作為變數名稱。

下面為 8 種不同的變數類型，前 5 種為有號數值型態，後 3 種為無號數值型態：

變數類型	描述	範圍
bool 布林型	1 bit (一個位)	0, 1
char 字元型	8 bits (一個位元組)	+127 ~ -128
short 短整型	16 bits (一個字元)	+32767 ~ -32768
int 雙整型	32 bits (雙字元)	+2147483647 ~ -2147483648
float 浮點型	32 bits (雙字元)	
unsigned char 字元型	8 bits (一個位元組)	0 到 255
unsigned short 短整型	16 bits (一個字元)	0 到 65535
unsigned int 雙整型	32 bits (雙字元)	0 到 4,294,967,295

變數聲明

變數必須在使用前聲明。所以，在巨集指令，所有的變數都必須在語句使用前都被聲明完成。聲明變數時，先定義變數的類型，後面再跟著變數名稱。

如下範例:

```
int      a
short    b, switch
float    pressure
unsigned short c
```

陣列聲明

巨集指令支援一維陣列 (下標從 0 開始)。聲明陣列變數時，先定義陣列變數的類型，變數名稱，接著就是該陣列變數的個數，變數個數必須放置在“[]”符號中。陣列變數的長度為 1 ~ 4096。一個巨集指令中最多只支援 4096 個變數。

如下範例:

```
int      a[10]
short    b[20], switch[30]
float    pressure[15]
```

陣列的下標最小為 0，最大下標為(陣列的長度-1)

如下範例:

```
char data[100] // 陣列變數的長度是 100
```

所以: 最小的陣列為 “data[0]”，最大的陣列為 “data[99]”，即 $100 - 1 = 99$ 。

變數和陣列初始化

有兩種方法可以讓變數初始化：

1. 使用語句中的賦值語句 (=)

如下範例：

```
int a  
float b[3]  
a = 10  
b[0] = 1
```

2. 聲明變數時直接賦值

```
char a = '5', b = 9
```

陣列變數的聲明是一個特殊的情況。一個完整的陣列被初始化時，可以在陣列變數聲明時，將資料放置在波形括弧“{}”裡面，各資料使用逗號分開。

如下所示：

```
float data[4] = {11, 22, 33, 44} // 這樣 data[0] = 11, data[1] = 22....
```

18.3.2 運算符號

運算符通常被用來指定資料是如何被操作和運算，如下：

運算符號	描述	舉例
=	賦值運算符號	pressure = 10

數學運算符號	描述	舉例
+	加	A = B + C
-	減	A = B - C
*	乘	A = B * C
/	除	A = B / C
%	求餘 (返回剩餘數)	A = B % 5

比較運算符號	描述	舉例
<	小於	if A < 10 then B = 5
<=	小於或者等於	if A <= 10 then B = 5
>	大於	if A > 10 then B = 5
>=	大於或者等於	if A >= 10 then B = 5
==	等於	if A == 10 then B = 5
<>	不等於	if A <> 10 then B = 5

邏輯運算符號	描述	舉例
And	與	if A < 10 and B > 5 then C = 10
Or	或	if A >= 10 or B > 5 then C = 10
Xor	異或	if A xor 256 then B = 5
Not	非	if not A then B = 5

移位元和位元運算符號通常被用來操作字元型變數、短整型變數和雙整型變數的位元。在一個語句中，這些運算符號的優先權是在從該語句的左邊到右邊依此執行的。即在語句中左邊位置的優先執行，依次從左到右執行。

移位運算符號	描述	舉例
<<	往左移動指定的位數	A = B << 8
>>	往右移動指定的位數	A = B >> 8

位運算符號	描述	舉例
&	位與運算	A = B & 0xf
 	位或運算	A = B C
^	位異或運算	A = B ^ C
~	位取反運算	A = ~B

所有運算符號的優先權

上述所有運算符號的優先權從高到低詳細如下所述：

位於圓括號裡面的運算符號最優先

數學運算符號

移位和位元運算符號

比較運算符號

邏輯運算符號

賦值運算符號

關鍵字

下面的關鍵字為巨集指令保留使用。這些均不能用來作為變數名稱、陣列名稱或者函數名稱等。

+, -, *, /, %, >=, >, <=, <, <>, ==, and, or, xor, not, <<, >>, =, &, |, ^, ~
exit, macro_command, for, to, down, step, next, return, bool, short, int, char, float, void, if, then,
else, break, continue, set, sub, end, while, wend, true, false
SQRT, CUBERT, LOG, LOG10, SIN, COS, TAN, COT, SEC, CSC, ASIN, ACOS, ATAN, BIN2BCD, BCD2BIN,
DEC2ASCII, FLOAT2ASCII, HEX2ASCII, ASCII2DEC, ASCII2FLOAT, ASCII2HEX, FILL, RAND, DELAY,
SWAPB, SWAPW, LOBYTE, HIBYTE, LOWORD, HIWORD, GETBIT, SETBITON, SETBITOFF, INVBIT,
ADDSUM, XORSUM, CRC, INPORT, OUTPORT, POW, GetError, GetData, GetDataEx, SetData,
SetDataEx, SetRTS, GetCTS, Beep, SYNC_TRIG_MACRO, ASYNC_TRIG_MACRO, TRACE,
FindDataSamplingDate, FindDataSamplingIndex, FindEventLogDate, FindEventLogIndex
StringGet, StringGetEx, StringSet, StringSetEx, StringCopy, StringMid, StringDecAsc2Bin,
StringBin2DecAsc, StringDecAsc2Float, StringFloat2DecAsc, StringHexAsc2Bin, StringBin2HexAsc,
StringLength, StringCat, StringCompare, StringCompareNoCase, StringFind, StringReverseFind,
StringFindOneOf, StringIncluding, StringExcluding, StringToUpper, StringToLower, StringToReverse,
StringTrimLeft, StringTrimRight, StringInsert

18.4 語句

18.4.1 定義語句

這個定義語句包含了變數和陣列的聲明。正式的格式如下：

類型 名稱

定義一個變數的名稱為"名稱"且類型為"類型"。

舉例：

`int A //定義了變數 A 為雙整型格式`

類型 陣列名稱[陣列長度]

定義一個陣列變數為"名稱"，大小為"陣列長度"且類型為"類型"時。

舉例：

`int B[10] 定義了一維陣列變數 B 的長度為 10，類型為雙整型`

18.4.2 賦值語句

賦值語句使用賦值運算符號將賦值運算符號右邊運算式運算的結果放置到運算符號左邊的變數中。一個運算式是由變數、常數和各種運算符號組成，執行後產生一個新的資料。

變數 = 運算式

舉例：

`A = 2 這樣變數 A 就被賦值為 2`

18.4.3 邏輯運算語句

邏輯運算語句是根據邏輯（布林）運算式的結果來執行相應的動作。它的語句如下所示：

單行格式

`if < Condition > then`

[Statements]

else

[Statements]

end if

舉例:

```
if a == 2 then  
    b = 1  
else  
    b = 2  
end if
```

區塊格式

```
If < Condition > then  
    [Statements]  
else if < Condition- n > then  
    [Statements]  
else  
    [Statements]  
end if
```

舉例:

```
if a == 2 then  
    b = 1  
else if a == 3 then  
    b = 2  
else  
    b = 3  
end if
```

語法描述:

if	必須用在該語句的開始部分
<Condition>	必要條件。 這是一個控制語句。當 <Condition> 為 0 時，即為 “FALES”，(條件為假)；當 <Condition> 為非 0 時，即為“True”(條件為真)。
then	當 <Condition> 執行為 “TRUE”(真) 時，必須放置在需要執行的語句之前。
[Statements]	在區塊形式中是可選擇的參數，在單行形式中，且沒有 else 子句時，為必要參數，該語句在 <Condition> 為真時執行。
else if	可選，一條或多條語句，在相對應的 <Condition – n> 為 true 時執行。
<Condition-n>	可選，解釋同 Condition
else	可選，在上述 Condition 和 Condition – n 都不為 true 時執行。
end if	必須。在一個 if-then 語句中使用這個來結束 if-then 語句。

18.4.4 多重判斷語句

Select-case 可用來處理多重判斷的敘述，其功能類似 if-else 語句。根據所指定變數的值，分別對應到符合該值的 case，並執行 case 下面的敘述，直到遇到 break 敘述時，才跳到結束符號 end select 處。語法結構如下：

沒有預設 case 的形式：

```
Select Case [variable]
Case [value]
[Statements]
break
end Select
```

舉例：

```
Select Case A
Case 1
    b=1
    break
end Select
```

有預設 case 的形式：

```
Select Case [variable]
Case [value]
[Statements]
break
Case else
[Statements]
break
end Select
```

舉例：

```
Select Case A
```

```
    Case 1
```

```
        b=1
```

```
        break
```

```
    Case else
```

```
        b=0
```

```
        break
```

```
end Select
```

多個不同 case 對應到相同區塊：

```
Select Case [variable]
```

```
Case [value1]
```

```
    [Statements]
```

```
Case [value2]
```

```
    [Statements]
```

```
    break
```

```
end Select
```

舉例：

```
Select Case A
```

```
    Case 1
```

```
        break
```

```
    Case 2
```

```
        b=2
```

```
        break
```

```
    Case 3
```

```
        b=3
```

```
        break
```

```
end Select
```

語法描述：

Select Case	必須用在該語句的開始部分。
[variable]	必要條件。此變數將會與每一個 case 做比較。
Case else	可選。代表預設 case。當 [variable] 的值不符合任何一個 case 時，將會執行此敘述下面的區塊。在沒有預設 case 的情況，當 [variable] 的值不符合任何一個 case 時，將不會做任何動作而直接跳出 select 控制結構。
break	可選。跳到某一個 case 下面執行時，將一句一句執行 case 語句下面的敘述直到遇到 break 命令才結束，並跳到 end select 敘述。當 case 敘述下面沒有任何 break 命令時，流程將不斷往下執行，直到遇到 end select 敘述，才結束並跳出 select 控制結構。
end Select	select-case 語句的結束標誌。

18.4.5 迴圈語句

迴圈語句依據迴圈條件來反復的執行一個任務。迴圈語句有兩種表達方式。

18.4.5.1 for-next 語句

For-next 語句通常用來執行次數固定的迴圈任務。一個變數用作為任務執行次數的計數器和結束迴圈任務執行的條件。這個變數為固定執行的次數。語法結構如下：

```
for [Counter] = <StartValue> to <EndValue> [step <StepValue>]  
    [Statements]  
next [Counter]
```

或者

```
for [Counter] = <StartValue> down <EndValue> [step <StepValue>]  
    [Statements]  
next [Counter]
```

舉例：

```
for a = 0 to 10 step 2  
    b = a  
next a
```

語法描述：

for	必須用在該語句的開始部分。
[Counter]	必要，迴圈計數器的數值變數，該變數的結果用來計數迴圈的次數。
<StartValue>	必要，Counter 的初值。
to/down	必要。用來決定步長是遞增還是遞減。 “to” 以 <StepValue> 為步長遞增 <Counter> “down” 以 <StepValue> 為步長遞減 <Counter>
<EndValue>	必要，Counter 的終值、測試點。當 <Connter> 大於該值時，巨集指令將結束這個迴圈任務。
step	可選，指定 <Step Value> 的步長，指定為 1 以外的數值。
[StepValue]	可選，Counter 的步長，只能是數值，如果沒有指定，則預設為 1。
[Statements]	可選，for 和 next 之間的語句區塊，該語句區塊將執行所指定的次數。
next	必須的。
[Counter]	可選。

18.4.5.2 while-wend 語句

While-wend 語句是用來執行不確定次數的迴圈任務。設置一個變數用來判斷結束迴圈的條件。當條件為“True”時，該語句將一直迴圈執行直到條件變為“False”。語法結構如下：

```
while <Condition>
    [Statements]
wend
```

舉例：

```
while a < 10
    a = a + 10
wend
```

語法描述：

while	必須用在該語句的開始部分。
continue	必要條件。這是一個控制語句。當為“True”時，開始執行迴圈命令，當為“False”時，結束執行迴圈命令。
return [value]	當判斷為“TRUE”時，繼續執行迴圈命令。
wend	While-wend 語句的結束標誌。

18.4.5.3 其他控制命令

break	用在 for-next 和 while-wend 語句中。當遇到此語句時，立即跳到語句的結束部分。
continue	用在 for-next 和 while-wend 語句中。當遇到此語句時，立即結束當前迴圈命令而開始執行下一個迴圈命令。
return	可用在自訂 function 的回傳值敘述。寫在主函數裡面時，用來強制跳出主函數。

18.5 子函數

使用子函數可以有效的減少迴圈命令的代碼，子函數必須在使用前被定義，且可以使用任何變數和語句類型。在主函數中，將子函數的參數放置在子函數名稱後面的圓括號中，即可調用子函數。子函數被執行後，將執行後的結果返回到主函數需要的賦值語句或者條件中。定義子函數時，不一定要有返回值，且參數部分可以為空。在主函數中調用子函數時，調用方式應符合其定義。語法結構如下：

有返回值的子函數語法：

```

sub type <函數名稱> [(parameters)]
    Local variable declarations
    [Statements]
    [return [value]]
end sub

```

舉例：

```

sub int Add(int x, int y)
    int result
    result = x +y
    return result
end sub

macro_command main()
    int a = 10, b = 20, sum
    sum = Add(a, b)
end macro_command

```

或:

```
sub int Add()
    int result, x=10, y=20
    result = x +y
    return result
end sub
```

```
macro_command main()
    int sum
    sum = Add()
end macro_command
```

沒有返回值的子函數語法：

```
sub <函數名稱> [(parameters)]
    Local variable declarations
    [Statements]
end sub
```

舉例:

```
sub Add(int x, int y)
    int result
    result = x +y
end sub
```

```
macro_command main()
    int a = 10, b = 20
    Add(a, b)
end macro_command
```

或:

```
sub Add()
    int result, x=10, y=20
    result = x +y
end sub
```

```
macro_command main()
    Add()
end macro_command
```

語法描述：

sub	必須用在該子函數的開始部分。
type	可選。用來定義子函數執行後返回的資料類型。子函數也可以不回傳任何值。
(parameters)	可選。這些參數保留了從主函數傳入的數值。這些被傳入的參數必須使用與在參數變數聲明的類型一致。 舉例： <code>sub int MyFunction(int x, int y).</code> x 和 y 必須為從主函數中傳過來的雙整型資料格式的資料。調用此子函數的語句格式大致為這樣： <code>ret = MyFunction(456, pressure)</code> ，其中 pressure 需為雙整型資料格式方符合子函數參數變數的聲明。 請注意調用語句的參數部分可以是常數也可以是變數。當執行這個子函數後，一個雙整型資料將會返回給變數 “ret”。
Local variable declaration	除了被傳遞的參數之外，子函數中使用的變數必須事先聲明。在上面的“舉例”中，X 和 Y 就是子函數可以使用的變數。總體變數也可以用在子函數中。
[Statements]	需要執行的語句。
[return [value]]	可選。用來將執行的結果返回給調用語句。這個結果可以是一個常數或者變數。返回後同時也結束了子函數的執行。子函數也可以不回傳任何值，但是當 type 部分有定義時，則必須加上此 return 敘述。
end sub	必須的。用來結束子函數。

18.6 內置函數功能

EasyBuilder 軟體巨集指令中本身提供了一些內建的函數用來從 PLC 獲取資料和傳輸資料到 PLC、資料處理和數學運算等。

18.6.1 數學運算函數

函數名稱	SQRT
語法	SQRT(<i>source, result</i>)
描述	開平方根。資料來源 “ <i>source</i> ” 可以是常數或者變數，但是存放結果的 “ <i>result</i> ” 必須為變數。資料來源必須為一個正數。
舉例	<pre>macro_command main() float source, result SQRT(15, result) source = 9.0 SQRT(source, result)// 執行後 result = 3.0 end macro_command</pre>

函數名稱	CUBERT
語法	CUBERT (<i>source, result</i>)
描述	開三次方根。資料來源 “ <i>source</i> ” 可以是常數或者變數，但是存放結果的 “ <i>result</i> ” 必須為變數。資料來源必須為一個正數。
舉例	<pre>macro_command main() float source, result CUBERT (27, result) // 執行後 result = 3.0 source = 27.0 CUBERT (source, result) // 執行後 result = 3.0 end macro_command</pre>

函數名稱	POW
語法	POW (source1, source2, result)
描述	計算 source1 的某次方 (source2)。資料來源 “source1” 和 “source2” 可以是常數或者變數，但是存放結果的 “result” 必須為變數。資料來源必須為一個正數。
舉例	<pre>macro_command main() float y, result y = 0.5 POW (25, y, result) // 執行後 result = 5 end macro_command</pre>

函數名稱	SIN
語法	SIN(source, result)
描述	三角函數的正弦計算。資料來源 “source” 可以是常數或者變數，但是存放結果的 “result” 必須為變數。
舉例	<pre>macro_command main() float source, result SIN(90, result) // result is 1 source = 30 SIN(source, result) // result is 0.5 end macro_command</pre>

函數名稱	COS
語法	COS(source, result)
描述	三角函數的餘弦計算。資料來源 “source” 可以是常數或者變數，但是存放結果的 “result” 必須為變數。
舉例	<pre>macro_command main() float source, result COS(90, result) // result is 0 source = 60 GetData(source, "Local HMI", LW, 0, 1) COS(source, result) // result is 0.5 end macro_command</pre>

語法名稱	TAN
語法	TAN(source, result)
描述	三角函數的正切計算。資料來源 “source” 可以是常數或者變數，但是存放結果的 “result” 必須為變數。
舉例	<pre>macro_command main() float source, result TAN(45, result) // result is 1 source = 60 TAN(source, result) // result is 1.732 end macro_command</pre>

函數名稱	COT
語法	COT(source, result)
描述	三角函數的餘切計算。資料來源 “source” 可以是常數或者變數，但是存放結果的 “result” 必須為變數。
舉例	<pre>macro_command main() float source, result COT(45, result) // result is 1 source = 60 COT(source, result) // result is 0.5774 end macro_command</pre>

函數名稱	SEC
語法	SEC(source, result)
描述	反三角函數中反正割計算。資料來源 “source” 可以是常數或者變數，但是存放結果的 “result” 必須為變數。
舉例	<pre>macro_command main() float source, result SEC(45, result) // result is 1.414 source = 60 SEC(source, result) // if source is 60, result is 2 end macro_command</pre>

函數名稱	CSC
語法	CSC(source, result)
描述	反三角函數中反餘割計算。資料來源 “source” 可以是常數或者變數，但是存放結果的 “result” 必須為變數。
舉例	<pre>macro_command main() float source, result CSC(45, result) // result is 1.414 source = 30 CSC(source, result) // result is 2 end macro_command</pre>

函數名稱	ASIN
語法	ASIN(source, result)
描述	反三角函數中反正弦計算。資料來源 “source” 可以是常數或者變數，但是存放結果的 “result” 必須為變數。
舉例	<pre>macro_command main() float source, result ASIN(0.8660, result) // result is 60 source = 0.5 ASIN(source, result) // result is 30 end macro_command</pre>

函數名稱	ACOS
語法	ACOS(source, result)
描述	反三角函數中反餘弦計算。資料來源 “source” 可以是常數或者變數，但是存放結果的 “result” 必須為變數。
舉例	<pre>macro_command main() float source, result ACOS(0.8660, result) // result is 30 source = 0.5 ACOS(source, result) // result is 60 end macro_command</pre>

函數名稱	ATAN
語法	ATAN(source, result)
描述	反三角函數中反正切計算。資料來源 “source” 可以是常數或者變數，但是存放結果的 “result” 必須為變數。
舉例	<pre>macro_command main() float source, result ATAN(1, result) // result is 45 source = 1.732 ATAN(source, result) // result is 60 end macro_command</pre>

函數名稱	LOG
語法	LOG (source, result)
描述	從取得 source 的自然對數，存入 result 變數。 source 可為變數或常數。 存放結果的 “result” 必須為變數。
舉例	<pre>macro_command main() float source=100, result LOG (source, result) // result約等於 4.6052 end macro_command</pre>

函數名稱	LOG10
語法	LOG10 (source, result)
描述	從取得 source 的以 10 為基底的對數，存入 result 變數。 Source 可為變數或常數。 存放結果的 “result” 必須為變數。
舉例	<pre>macro_command main() float source=100, result LOG10 (source, result) // result等於2 end macro_command</pre>

函數名稱	RAND
語法	RAND(result)
描述	產生一個隨機數 存放結果的 “result” 必須為變數。
舉例	<pre>macro_command main() short result RAND (result) // result is not a fixed value when executes macro every time end macro_command</pre>

18.6.2 資料轉換函數

函數名稱	BIN2BCD
語法	BIN2BCD(source, result)
描述	將 BIN 格式的資料 (source) 轉換為 BCD 格式的資料 (result)。資料來源 “source” 可以是常數或者變數，但是存放結果的 “result” 必須為變數。
舉例	<pre>macro_command main() short source, result BIN2BCD(1234, result) // result is 0x1234 source = 5678 BIN2BCD(source, result) // result is 0x5678 end macro_command</pre>

函數名稱	BCD2BIN
語法	BCD2BIN(source, result)
描述	將 BCD 格式的資料 (source) 轉換為 BIN 格式的資料 (result)。資料來源 “source” 可以是常數或者變數，但是存放結果的 “result” 必須為變數。
舉例	<pre>macro_command main() short source, result BCD2BIN(0x1234, result) // result is 1234 source = 0x5678 BCD2BIN(source, result) // result is 5678 end macro_command</pre>

函數名稱	DEC2ASCII
語法	DEC2ASCII(source, result[start], len)
描述	<p>將十進位的資料 (source) 轉換為 ASCII 格式的資料，並存放在一個一維陣列 (result) 中。len 表示這個轉換後的字串的長度，同時這個長度也取決於存放結果的一維陣列的資料格式。例如：如果 result 一維陣列的格式為 “char” (字元型，長度為一個位元組)，則長度為 “位元組數*len”。如果 result 一維陣列的格式為 “short” (短整型資料，2 個位元組)，則長度為 “word*len”。依此類推。</p> <p>轉換後的第一個字元放在 result[start] 中，第二個字元放在 result[start+1] 中，最後一個字元放在 result[start+(len-1)] 中。</p> <p>source 和 len 可以是常數或者變數，單數 result 必須為變數。start 必須為常數。</p>
舉例	<pre>macro_command main() short source char result1[4] short result2[4] char result3[6] source = 5678 DEC2ASCII(source, result1[0], 4) // result1[0] is '5', result1[1] is '6', result1[2] is '7', result1[3] is '8' // the length of the string (result1) is 4 bytes(= 1 * 4) DEC2ASCII(source, result2[0], 4) // result2[0] is '5', result2[1] is '6', result2[2] is '7', result2[3] is '8' // the length of the string (result2) is 8 bytes(= 2 * 4) source=-123 DEC2ASCII(source3, result3[0], 6) // result1[0] is '-', result1[1] is '0', result1[2] is '0', result1[3] is '1' // result1[4] is '2', result1[5] is '3' // the length of the string (result1) is 6 bytes(= 1 * 6) end macro_command</pre>

函數名稱	HEX2ASCII
語法	HEX2ASCII(source, result[start], len)
描述	<p>十六進位格式資料 (source) 轉換為 ASCII 格式的資料，並將結果存放在一個一維陣列 (result) 中。len 表示這個轉換後的字串的長度，同時這個長度也取決於存放結果的一維陣列的資料格式。例如：如果 result 一維陣列的格式為 “char” (字元型，長度為一個位元組)，則長度為 “位元組數*len”。如果 result 一維陣列的格式為 “short” (短整型資料，2 個位元組)，則長度為 “word*len”。依此類推。</p> <p>source 和 len 可以是常數或者變數，單數 result 必須為變數。start 必須為常數。</p>
舉例	<pre>macro_command main() short source char result[4] source = 0x5678 HEX2ASCII(source, result[0], 4) // result[0] is '5', result[1] is '6', result[2] is '7', result[3] is '8' end macro_command</pre>

函數名稱	FLOAT2ASCII
語法	FLOAT2ASCII (source, result[start], len)
描述	<p>浮點數格式資料 (source) 轉換為 ASCII 格式的資料，並將結果存放在一個一維陣列 (result) 中。len 表示這個轉換後的字串的長度，同時這個長度也取決於存放結果的一維陣列的資料格式。例如：如果 result 一維陣列的格式為 “char” (字元型，長度為一個位元組)，則長度為 “位元組數*len”。如果 result 一維陣列的格式為 “short” (短整型資料，2 個位元組)，則長度為 “word*len”。依此類推。</p> <p>source 和 len 可以是常數或者變數，單數 result 必須為變數。start 必須為常數。</p>
舉例	<pre>macro_command main() float source char result[4] source = 56.8</pre>

	<pre> FLOAT2ASCII (source, result[0], 4) // result[0] is '5', result[1] is '6', result[2] is '.', result[3] is '8' end macro_command </pre>
--	--

函數名稱	ASCII2DEC
語法	ASCII2DEC(source[start], result, len)
描述	<p>將字元型 ASCII 資料 (source) 轉換為十進位格式的資料，並存放在 result 變數中。ASCII 的長度即為 len，第一個字元的位置即為 source[start] 的資料。</p> <p>source 和 len 可以是常數或者變數，單數 result 必須為變數。start 必須為常數。</p>
舉例	<pre> macro_command main() char source[4] short result source[0] = '5' source[1] = '6' source[2] = '7' source[3] = '8' ASCII2DEC(source[0], result, 4) // result is 5678 end macro_command </pre>

函數名稱	ASCII2HEX
語法	ASCII2HEX (source[start], result, len)
描述	<p>將 ASCII 字元型資料 (source) 轉換為十六進位的資料，並存放在 result 變數中。字元的長度即為 len 的資料。第一個字元存放在 source[start] 中。</p> <p>source 和 len 可以是常數或者變數，單數 result 必須為變數。start 必須為常數。</p>
舉例	<pre> macro_command main() char source[4] short result source[0] = '5' </pre>

```
source[1] = '6'  
source[2] = '7'  
source[3] = '8'  
  
ASCII2HEX(source[0], result, 4) // result is 0x5678  
  
end macro_command
```

函數名稱	ASCII2FLOAT
語法	ASCII2FLOAT (source[start], result, len)
描述	將字元型 ASCII 資料 (source) 轉換為浮點數格式的資料，並存放在 result 變數中。ASCII 的長度即為 len，第一個字元的位置為 source[start] 的資料。 source 和 len 可以是常數或者變數，單數 result 必須為變數。Start 必須為常數。
舉例	macro_command main() char source[4] float result source[0] = '5' source[1] = '6' source[2] = '.' source[3] = '8' ASCII2FLOAT(source[0], result, 4) // result is 56.8 end macro_command

18.6.3 資料操作函數

函數名稱	FILL
語法	FILL(source[start], preset, count)
描述	依序將預設值 (preset) 放置到一維陣列 source[start] 開始的陣列中，放置的資料個數由 count 決定。 source 和 start 必須為變數，preset 可以為一個常數或者變數。
舉例	<pre>macro_command main() char result[4] char preset FILL(result[0], 0x30, 4) // result[0] is 0x30, result[1] is 0x30, , result[2] is 0x30, , result[3] is 0x30 preset = 0x31 FILL(result[0], preset, 2) // result[0] is 0x31, result[1] is 0x31 end macro_command</pre>

函數名稱	SWAPB
語法	SWAPB(source, result)
描述	將一個 16 位元字的高低位元組顛倒，並將結果存放在 result 變數中。 source 可以是常數或者是變數，單數 result 必須為變數。
舉例	<pre>macro_command main() short source, result SWAPB(0x5678, result) // result is 0x7856 source = 0x123 SWAPB(source, result) // result is 0x2301 end macro_command</pre>

函數名稱	SWAPW
語法	SWAPW(source, result)
描述	將一個 32 位元雙整型資料的高位字元和低位字元顛倒，並將結果存放在 result 變數中。 source 可以是常數或者變數，但是 result 必須為變數。
舉例	<pre>macro_command main() int source, result SWAPW(0x12345678, result) // result is 0x56781234 source = 0x12345 SWAPW(source, result) // result is 0x23450001 end macro_command</pre>

函數名稱	LOBYTE
語法	LOBYTE(source, result)
描述	獲取一個 16 位元資料的低位元組，並且放置在 result 變數中。 source 可以是常數或者變數，但是 result 必須為變數。
舉例	<pre>macro_command main() short source, result LOBYTE(0x1234, result) // result is 0x34 source = 0x123 LOBYTE(source, result) // result is 0x23 end macro_command</pre>

函數名稱	HIBYTE
語法	HIBYTE(source, result)
描述	獲取一個 16 位元資料的高位元組，並且放置在 result 變數中。 source 可以是常數或者變數，但是 result 必須為變數。
舉例	<pre>macro_command main() short source, result HIBYTE(0x1234, result) // result is 0x12 source = 0x123 HIBYTE(source, result) // result is 0x01 end macro_command</pre>

函數名稱	LOWORD
語法	LOWORD(source, result)
描述	獲取一個 32 位元資料的低位字元，並將結果放置在 result 變數中。 source 可以是常數或者變數，但是 result 必須為變數。
舉例	<pre>macro_command main() int source, result LOWORD(0x12345678, result) // result is 0x5678 source = 0x12345 LOWORD(source, result) // result is 0x2345 end macro_command</pre>

函數名稱	HIWORD
語法	HIWORD(source, result)
描述	獲取一個 32 位元資料的高位字元，並將結果放置在 result 變數中。 source 可以是常數或者變數，但是 result 必須為變數。
舉例	<pre>macro_command main() int source, result HIWORD(0x12345678, result) // result is 0x1234 source = 0x12345 HIWORD(source, result) // result is 0x0001 end macro_command</pre>

18.6.4 位元狀態轉換

函數名稱	GETBIT
語法	GETBIT(source, result, bit_pos)
描述	獲取資料或者變數 (source) 指定的位元的狀態，並將結果放置在 result 變數中。result 的資料將為 1 或者 0。 source 和 bit_pos 可以是常數或者變數，但是 result 必須為變數。
舉例	<pre>macro_command main() int source, result short bit_pos GETBIT(9, result, 3) // result is 1 source = 4 bit_pos = 2 GETBIT(source, result, bit_pos) // result is 1 end macro_command</pre>

函數名稱	SETBITON
語法	SETBITON(source, result, bit_pos)
描述	將資料或者變數 (source) 指定的位元位址設置為 1，並將改變後的資料存放在 result 變數中。 source 和 bit-pos 可以是常數或者變數，但是 result 必須為變數。
舉例	<pre>macro_command main() int source, result short bit_pos SETBITON(1, result, 3) // result is 9 source = 0 bit_pos = 2 SETBITON (source, result, bit_pos) // result is 4 end macro_command</pre>

函數名稱	SETBITOFF
語法	SETBITOFF(source, result, bit_pos)
描述	將資料或者變數 (source) 指定的位元位址設置為 0，並將改變後的資料存放在 result 變數中。 source 和 bit-pos 可以是常數或者變數，但是 result 必須為變數。
舉例	<pre>macro_command main() int source, result short bit_pos SETBITOFF(9, result, 3) // result is 1 source = 4 bit_pos = 2 SETBITOFF(source, result, bit_pos) // result is 0 end macro_command</pre>

函數名稱	INVBIT
語法	INVBIT(source, result, bit_pos)
描述	將資料或者變數 (source) 指定的位元位址狀態相反，並將改變後的資料存放在 result 變數中。 source 和 bit-pos 可以是常數或者變數，但是 result 必須為變數。
舉例	<pre>macro_command main() int source, result short bit_pos INVBIT(4, result, 1) // result = 6 source = 6 bit_pos = 1 INVBIT(source, result, bit_pos) // result = 4 end macro_command</pre>

18.6.5 通訊有關的函數

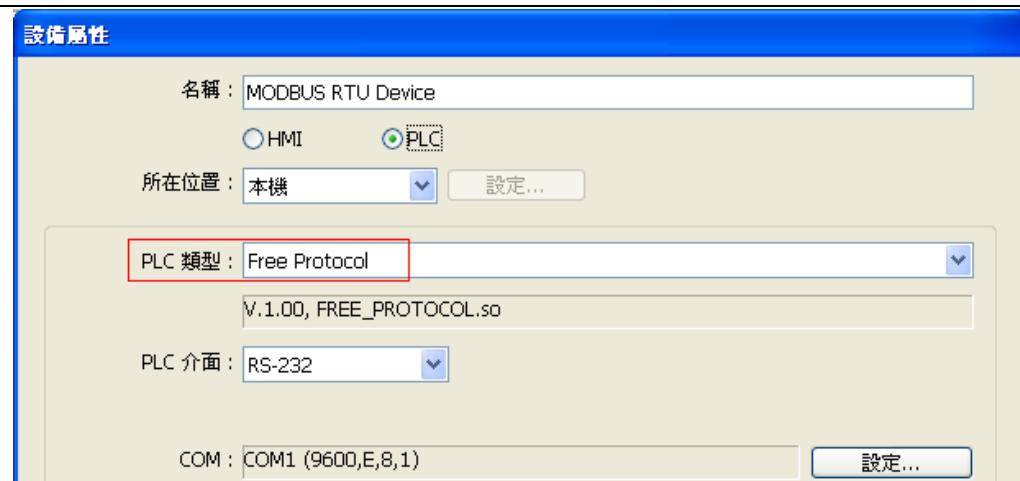
函數名稱	DELAY
語法	DELAY(time)
描述	讓巨集指令暫停執行，持續的時間至少是指定的這個時間。時間的單位為毫秒。 time 可以是常數或者變數。
舉例	<pre>macro_command main() int time = 500 DELAY(100) // delay 100 ms DELAY(time) // delay 500 ms end macro_command</pre>

函數名稱	ADDSUM
語法	ADDSUM(source[start], result, data_count)
描述	將 source[start] 到 source[start+data-count-1] 的所有一維陣列的資料累加起來，以獲得 checksum (校驗和)，並將結果存放在 result 變數中。 result 必須為變數，data_count 是進行累加的資料的個數，可以是常數或者是變數。
舉例	<pre>macro_command main() char data[5] short checksum data[0] = 0x1 data[1] = 0x2 data[2] = 0x3 data[3] = 0x4 data[4] = 0x5 ADDSUM(data[0], checksum, 5) // checksum is 0xf end macro_command</pre>

函數名稱	XORSUM
語法	XORSUM(source[start], result, data_count)
描述	將 source[start] 到 source[start+data-count-1] 的所有一維陣列的資料進行異或運算，以獲得 checksum (校驗和)，並將結果存放在 result 變數中。 result 必須為變數，data_count 是進行異或計算的資料的個數，可以是常數或者是變數。
舉例	<pre>macro_command main() char data[5] = {0x1, 0x2, 0x3, 0x4, 0x5} short checksum XORSUM(data[0], checksum, 5) // checksum is 0x1 end macro_command</pre>

函數名稱	CRC
語法	CRC(source[start], result, data_count)
描述	將 source[start] 到 source[start+data-count-1] 的所有一維陣列的資料進 16-bit CRC 計算，以獲得 checksum (校驗和)，並將結果存放在 result 變數中。result 必須為變數，data_count 是進行計算的資料的個數，可以是常數或者是變數。
舉例	<pre>macro_command main() char data[5] = {0x1, 0x2, 0x3, 0x4, 0x5} short 16bit_CRC CRC(data[0], 16bit_CRC, 5) // 16bit_CRC is 0xbb2a end macro_command</pre>

函數名稱	OUTPORT																					
語法	OUTPORT(source[start], device_函數名稱, data_count)																					
描述	將放置在從 source[start] 到 source[start+count-1] 的所有資料通過串列埠或者乙太網口傳送給 PLC 或者控制器中。 device_函數名稱是在“設備列表”中定義的“PLC 名稱”，而這個 device 必須選擇為“Free Protocol”這個 PLC 類型。 Data_count 是發送資料的個數，可以是常數或變數。																					
舉例	要使用 OUTPORT 函數，必須要在 PLC 類型中選擇“Free Protocol”，如下圖所示。  <table border="1"> <thead> <tr> <th>編號</th> <th>名稱</th> <th>位置</th> <th>設備類型</th> <th>介面類型</th> <th>通訊協議</th> <th>站號</th> </tr> </thead> <tbody> <tr> <td>本機 HMI</td> <td>Local HMI</td> <td>本機</td> <td>MT6056i (320 x ...)</td> <td>停用</td> <td>N/A</td> <td>N/A</td> </tr> <tr style="background-color: #ffffcc;"> <td>本機 雖服器</td> <td>MODEUS RTU Device</td> <td>本機</td> <td>Free Protocol</td> <td>COM1(9600,E,8,1)</td> <td>RS232</td> <td>0</td> </tr> </tbody> </table>	編號	名稱	位置	設備類型	介面類型	通訊協議	站號	本機 HMI	Local HMI	本機	MT6056i (320 x ...)	停用	N/A	N/A	本機 雖服器	MODEUS RTU Device	本機	Free Protocol	COM1(9600,E,8,1)	RS232	0
編號	名稱	位置	設備類型	介面類型	通訊協議	站號																
本機 HMI	Local HMI	本機	MT6056i (320 x ...)	停用	N/A	N/A																
本機 雖服器	MODEUS RTU Device	本機	Free Protocol	COM1(9600,E,8,1)	RS232	0																



這裡的 device_函數名稱即為 “MODBUD RTU Device”。埠的屬性也是依據在這個 “系統參數” 中的設定，譬如，在此設定為 (9600 , E , 8 , 1...)

下面是一個範例程式，使用自由協定，以 MODBUS RTU 的協定格式，將單個暫存器設置為 ON。

```
macro_command main()

char command[32]
short address, checksum

FILL(command[0], 0, 32) // 初始化命令

command[0] = 0x1 // 站號
command[1] = 0x5 // 功能碼：寫單個位

address = 0
HIBYTE(address, command[2])
LOBYTE(address, command[3])

command[4] = 0xff // 使該bit設置為ON
command[5] = 0

CRC(command[0], checksum, 6)

LOBYTE(checksum, command[6])
HIBYTE(checksum, command[7])
```

	<pre>// 將命令通過串列埠送出去 OUTPORT(command[0], "MODBUS RTU Device", 8) end macro_command</pre>
--	--

函數名稱	IMPORT
語法	IMPORT(read_data[start], device_函數名稱, read_count, return_value)
描述	<p>從串列埠或者乙太網口讀數據到人機界面上。這些資料保存在 read-data[start]~read-data[start+read-count-1] 這個一維陣列中。同樣的，device_函數名稱見上面的說明，在此不再詳述。</p> <p>read-count 是設定的需要讀取的命令的位元組長度，它可以是一個常數或變數。如果這個函數能夠成功的從 PLC 或者控制器中讀取到資料，則 return_value 的值為 1，否則就為 0。</p>
舉例	<p>下面就是一個使用 IMPORT 函數讀取一個 MODBUS 設備保持暫存器資料的範例。</p> <pre>// 讀取保持暫存器資料 macro_command main() char command[32], response[32] short address, checksum short read_no, return_value, read_data[2] FILL(command[0], 0, 32) // 命令初始化 FILL(response[0], 0, 32) command[0] = 0x1 // 站號 command[1] = 0x3 // 功能碼：讀取保持暫存器 address = 0 HIBYTE(address, command[2]) LOBYTE(address, command[3]) read_no = 2 // read 2 words (4x_1 and 4x_2) HIBYTE(read_no, command[4]) LOBYTE(read_no, command[5]) CRC(command[0], checksum, 6)</pre>

```

LOBYTE(checksum, command[6])
HIBYTE(checksum, command[7])

// 使用 OUTPORT 函數將命令送出去
OUTPORT(command[0], "MODBUS RTU Device", 8)

// 使用 IMPORT 函數讀取返回的命令
IMPORT(response[0], "MODBUS RTU Device", 9, return_value)

if return_value > 0 then
read_data[0] = response[4] + (response[3] << 8) // data in 4x_1
read_data[1] = response[6] + (response[5] << 8) // data in 4x_2

SetData(read_data[0], "Local HMI", LW, 100, 2)
end if

end macro_command

```

函數名稱	IMPORT2
語法	IMPORT2(response[start], device_name, receive_len, wait_time)
描述	從串列埠或者乙太網口讀數據到人機界面上。這些資料保存在 response 這個一維陣列中。device_name 的說明與 OUTPORT 相同，在此不再詳述。receive_len 存放所接收到的資料位元組長度，必須為變數。receive_len 的最大長度將不會超過 response 陣列宣告的大小。wait_time 表示等待時間（單位是 millisecond），可以是一個常數或變數。當資料讀取完畢後，若在指定的等待時間內未再收到任何資料，此函數便結束執行並回傳結果。
舉例	<pre> macro_command main() short wResponse[6], receive_len, wait_time=20 IMPORT2(wResponse[0], "Free Protocol", receive_len, wait_time) // wait_time 單位 : millisecond if receive_len > 0 then SetData(wResponse[0], "Local HMI", LW, 0, 6) // 把回應的資料寫入LW0 </pre>

	<pre>end if end macro_command</pre>
--	--------------------------------------

函數名稱	GetData
語法	<pre>GetData(read_data[start], device_函數名稱, device_type, address_offset, data_count) or GetData(read_data, device_函數名稱, device_type, address_offset, 1)</pre>
描述	<p>獲取 PLC 的資料。資料是存儲在 <code>read_data[start]~read_data[start+data_count-1]</code> 這些一維陣列變數中。</p> <p><code>data_count</code> 是設定的讀取數據的個數。一般來說，<code>read_data</code> 是一個一維陣列，但是如果 <code>data_count</code> 是 1，<code>read_data</code> 可以是一個一維陣列，也可以是一個普通的變數。下面是兩種從 PLC 中讀取一個字的方法。</p> <pre>macro_command main() short read_data_1[2], read_data_2 GetData(read_data_1[0], "FATEK FB Series", RT, 5, 1) GetData(read_data_2, "FATEK FB Series", RT, 5, 1) end macro_command</pre>

此處的 `device_函數名稱`，即為在“系統參數”中建立 PLC 類型時，設定的“PLC 名稱”。在此，PLC 名稱被設定為“FATEK FB Series”，如下圖所示。



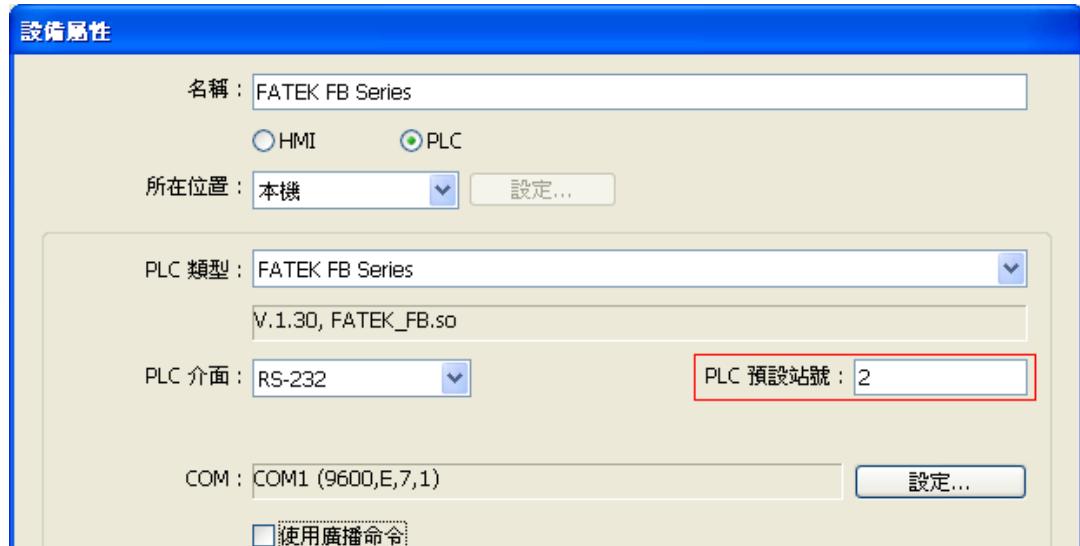
`device_type` 是設備類型和 PLC 中資料的編碼方式。例如：如果 `device_type` 是 `LW_BIN`，那麼讀取的設備類型為 `LW`，資料編碼方式為 `BIN`。如果使用 `BIN` 編碼方式，“`_BIN`”可以忽略。

如果 `device_type` 是 `LW_BCD`，表示設備類型 `LW`，資料的編碼方式為 `BCD` 格式。

`address_offset` 是 PLC 中的地址偏移量。

例如，`GetData(read_data_1[0], "FATEK FB Series", RT, 5, 1)` 代表讀取的設備位址偏移量為 5。

如果 `address_offset` 使用格式為 “N#AAAAA”，N 表示 PLC 的站號，AAAAA 表示位址偏移量。此情況一般使用在同一個串列埠上連接有多台 PLC 或者控制器的情況下。例如：`GetData(read_data_1[0], "FATEK FB Series", RT, 2#5, 1)` 表示讀取站號為 2 的 PLC 的資料。如果 `GetData()` 使用“系統參數 / 設備列表”中設定的默認的站號，在此可以不填這個站號。



從 PLC 中讀取的資料個數，根據 `read_data` 變數的類型和 `data_count` 的值來決定的。如下表所示：

<code>read_data</code> 的類型	<code>data_count</code> 的值	讀取 16 位元數據的個數
char (8-bit)	1	1
char (8-bit)	2	1
bo (8-bit)	1	1
bool (8-bit)	2	1
short (16-bit)	1	1
short (16-bit)	2	2
int (32-bit)	1	2
int (32-bit)	2	4
float (32-bit)	1	2
float (32-bit)	2	4

當 `GetData()` 函數讀取 32 位元的資料類型 (int 或者 float 型) 時，此函數會

	自動的轉換這個資料。例如： <pre>macro_command main() float f GetData(f, "MODBUS", 6x, 2, 1) // f 中將會是浮點型的數據 end macro_command</pre>
舉例	<pre>macro_command main() bool a bool b[30] short c short d[50] int e int f[10] double g[10] // 讀取 LB2 的狀態到變數 a 中 GetData(a, "Local HMI", LB, 2, 1) // 讀取 LB0~LB29共 30 個狀態，到變數 b[0] ~ b[29] 中 GetData(b[0], "Local HMI", LB, 0, 30) // 讀取 LW2 的資料到變數 c 中 GetData(c, "Local HMI", LW, 2, 1) // 讀取 LW0 ~ LW49 共 50 個字到變數 d[0] 到 d[49] 中 GetData(d[0], "Local HMI", LW, 0, 50) // 讀取兩個字 LW6 ~ LW7 到變數 e 中 // 注意此時變數 e 的類型為 int GetData(e, "Local HMI", LW, 6, 1) // 讀取 LW0 ~ LW19 共 20 個字到變數 f[0] ~ f[9] 中 (共 10 個 int 型變數)，陣列 f[10] 的變數類型定義為 int。 // 注意一個 int 資料將佔據 2 個字元 GetData(f[0], "Local HMI", LW, 0, 10) // 讀取 LW2 ~ LW3 共 2 個字到變數 f 中 GetData(f, "Local HMI", LW, 2, 1)</pre>

	end macro_command
--	-------------------

函數名稱	GetDataEx
語法	<pre>GetDataEx (read_data[start], device_函數名稱, device_type, address_offset, data_count) or GetDataEx (read_data, device_函數名稱, device_type, address_offset, 1)</pre>
描述	獲取 PLC 的資料，不等待 PLC 回應，逕自往下執行。 read_data、device_函數名稱、device_type、address_offset 和 data_count 的說明和 GetData 相同。
舉例	<pre>macro_command main() bool a bool b[30] short c short d[50] int e int f[10] double g[10] // 讀取 LB2 的狀態到變數 a 中 GetDataEx (a, "Local HMI", LB, 2, 1) // 讀取 LB0 ~ LB29 共 30 個狀態，到變數 b[0] ~ b[29] 中 GetDataEx (b[0], "Local HMI", LB, 0, 30) // 讀取 LW2 的資料到變數 c 中 GetDataEx (c, "Local HMI", LW, 2, 1) // 讀取 LW0 ~ LW49 共 50 個字到變數 d[0] 到 d[49]中 GetDataEx (d[0], "Local HMI", LW, 0, 50) // 讀取兩個字 LW6 ~ LW7 到變數 e 中 // 注意此時變數 e 的類型為 int GetDataEx (e, "Local HMI", LW, 6, 1)</pre>

```

// 讀取 LW0 ~ LW19 共 20 個字到變數 f[0] ~ f[9] 中，陣列 f[10] 的
// 變數類型定義為 int。
// 注意一個 int 資料將佔據 2 個字元
GetDataEx (f[0], "Local HMI", LW, 0, 10)

// 讀取 LW2 ~ LW3 共 2 個字到變數 f 中
GetDataEx (f, "Local HMI", LW, 2, 1)

end macro_command

```

函數名稱	SetData
語法	<pre>SetData(send_data[start], device_函數名稱, device_type, address_offset, data_count) or SetData(send_data, device_函數名稱, device_type, address_offset, 1)</pre>
描述	<p>將數據寫到 PLC 中。資料保存在 send_data[start]~send_data[start+data_count-1] 中。</p> <p>data_count 是寫入到 PLC 中資料的個數。一般來說，send_data 是一個陣列。但是如果 data_count 是 1，send_data 可以是一個陣列也可以是一個普通的變數。下面是寫一個資料到 PLC 中的方法。</p> <pre>macro_command main() short send_data_1[2] = { 5, 6}, send_data_2 = 5 SetData(send_data_1[0], "FATEK FB Series", RT, 5, 1) SetData(send_data_2, "FATEK FB Series", RT, 5, 1) end macro_command</pre> <p>device_函數名稱詳見上面的說明，在此不在說明。</p> <p>device_type 是設備類型和 PLC 中資料的編碼方式。例如：如果 device_type 是 LW_BIN，那麼讀取的設備類型為 LW，資料編碼方式為 BIN。如果使用 BIN 編碼方式，“_BIN”可以忽略。</p> <p>如果 device_type 是 LW_BCD，表示設備類型 LW，資料的編碼方式為 BCD 格式。</p> <p>address_offset 是 PLC 中的地址偏移量。</p> <p>例如，SetData (read_data_1[0], "FATEK FB Series", RT, 5, 1) 代表讀取的設備位址偏移量為 5。</p>

如果 `address_offset` 使用格式為 “N#AAAAA”，N 表示 PLC 的站號，AAAAA 表示位址偏移量。此情況一般使用在同一個串列埠上連接有多台 PLC 或者控制器的情況下。例如：`SetData(send_data_1[0], "FATEK FB Series", RT, 2#5, 1)` 表示設定站號為 2 的 PLC 的資料。如果 `SetData()` 使用 “系統參數 / 設備列表” 中設定的默認的站號，在此可以不填這個站號。

設定到 PLC 的資料個數，根據 `sead_data` 變數的類型和 `data_count` 的值來決定的。如下表所示：

<code>sead_data</code> 的類型	<code>data_count</code> 的值	設定 16 位元數據的個數
char (8-bit)	1	1
char (8-bit)	2	1
bool (8-bit)	1	1
bool (8-bit)	2	1
short (16-bit)	1	1
short (16-bit)	2	2
int (32-bit)	1	2
int (32-bit)	2	4
float (32-bit)	1	2
float (32-bit)	2	4

當 `Setdata()` 函數寫入 32 位元的資料類型 (int 或者 float 型) 到 PLC 時，此函數會自動的轉換這個資料。例如：

```
macro_command main()
float f = 2.6
SetData(f, "MODBUS", 6x, 2, 1) // 在此將會設定一個浮點數到 PLC 中
end macro_command
```

舉例

```
macro_command main()
int i
bool a = true
bool b[30]
short c = false
short d[50]
int e = 5
int f[10]
```

```
for i = 0 to 29
b[i] = true
next i

for i = 0 to 49
d[i] = i * 2
next i

for i = 0 to 9
f [i] = i * 3
next i

// 變數 a 的數值設定到 LB2 中
SetData(a, "Local HMI", LB, 2, 1)

// 設定 LB0 ~ LB29 共 30 個位的狀態
SetData(b[0], "Local HMI", LB, 0, 30)

// 將變數 c 的值設定到 LW2 中
SetData(c, "Local HMI", LW, 2, 1)

// 設定 LW0 ~ LW49 共 50 個數據
SetData(d[0], "Local HMI", LW, 0, 50)

// 將變數 e 的值寫入到 LW6 ~ LW7 兩個暫存器中，注意變數 e 的類型為
int。
SetData(e, "Local HMI", LW, 6, 1)

// 設定 LW0 ~ LW19 共 20 個字的數據
// 10 個雙整型數據等於 20 個 16 位元整型數 (1個字元)，因一個 int 資
料將佔據 2 個字元
SetData(f[0], "Local HMI", LW, 0, 10)

end macro_command
```

函數名稱	SetDataEx
語法	<pre>SetDataEx (send_data[start], device_函數名稱, device_type, address_offset, data_count) or SetDataEx (send_data, device_函數名稱, device_type, address_offset, 1)</pre>
描述	將數據寫到 PLC 中，不等待 PLC 回應，逕自往下執行。 send_data、device_函數名稱、device_type、address_offset 和 data_count 的說明和 SetData 相同。
舉例	<pre>macro_command main() int i bool a = true bool b[30] short c = false short d[50] int e = 5 int f[10] for i = 0 to 29 b[i] = true next i for i = 0 to 49 d[i] = i * 2 next i for i = 0 to 9 f [i] = i * 3 next i // 將變數 a 的數值設定到 LB2 中 SetDataEx(a, “Local HMI”, LB, 2, 1) // 設定 LB0 ~ LB29 共 30 個位的狀態 SetDataEx (b[0], “Local HMI”, LB, 0, 30) // 將變數 c 的值設定到 LW2 中 SetDataEx (c, “Local HMI”, LW, 2, 1)</pre>

```
// 設定 LW0 ~ LW49 共 50 個數據
SetDataEx (d[0], "Local HMI", LW, 0, 50)

// 將變數 e 的值寫入到 LW6 ~ LW7 兩個暫存器中，注意變數 e 的類型為
int。
SetDataEx (e, "Local HMI", LW, 6, 1)

// 設定 LW0 ~ LW19 共 20 個字的數據
// 10 個雙整型數據等於 20 個 16 位元整型數 (1個字元)，因一個 int 資
料將佔據 2 個字元
SetDataEx (f[0], "Local HMI", LW, 0, 10)

end macro_command
```

函數名稱	GetError
語法	GetError(err)
描述	取得錯誤碼。
舉例	<pre>macro_command main() short err char byData[10] GetDataEx(byData[0], "MODBUS RTU", 4x, 1, 10) // 讀取 10 byte資料 // 當錯誤碼 (err) 為 0，表示 GetDataEx 成功執行 GetErr(err) // 讀取錯誤碼，存入變數err end macro_command</pre>

函數名稱	PURGE
語法	PURGE (com_port)
描述	com_port 表示序列埠編號，支援 COM1 ~ COM3。此參數可以為變數或常數。可利用這個指令來清空 COM port 的輸出入緩衝區。
舉例	<pre>macro_command main() int com_port=3 PURGE (com_port) PURGE (1) end macro_command</pre>

函數名稱	SetRTS
語法	SetRTS(com_port, source)
描述	<p>拉高或拉低 RS-232 之 RTS 訊號。</p> <p>com_port 表示序列埠編號，支援 COM1。此參數可以為變數或常數。source 參數也可以為變數或常數。</p> <p>當傳入的 source 參數值大於 0 時，將拉高 RTS 電位，當 source 參數值等於 0 時，將拉低 RTS 電位。</p>
舉例	<pre>macro_command main() char com_port=1 char value=1 SetRTS(com_port, value) // value > 0, 拉高 COM1 之 RTS 電位 SetRTS(1, 0) // 拉低 COM1 之 RTS 電位 end macro_command</pre>

函數名稱	GetCTS
語法	GetCTS(com_port, result)
描述	<p>偵測 RS-232 之 CTS 訊號。</p> <p>com_port 表示序列埠編號，支援 COM1。此參數可以為變數或常數。result 參數必須為變數。</p> <p>此函數可偵測 CTS 電位值，當 CTS 在高電位時，result 將寫入 1，否則寫入 0。</p>
舉例	<pre>macro_command main() char com_port=1 char result GetCTS(com_port, result) // 偵測 COM1 之 CTS 電位值 GetCTS (1, result) // 偵測 COM1 之 CTS 電位值 end macro_command</pre>

18.6.6 字串處理函數

函數名稱	StringGet																								
語法	StringGet(read_data[start], device_函數名稱, device_type, address_offset, data_count)																								
描述	<p>獲取 PLC 的資料。字串的資料型態為字元陣列，是存儲在 <code>read_data[start]~read_data[start+data_count-1]</code> 這些一維陣列變數中。</p> <p><code>read_data</code> 必須為一維字元陣列。</p> <p><code>data_count</code> 是設定的讀取字元的個數，可以是常數也可以是變數。</p> <p>此處的 <code>device_函數名稱</code>，即為在“系統參數”中建立 PLC 類型時，設定的“PLC 名稱”。在此，PLC 名稱被設定為“FATEK FB Series”，如下圖所示。</p>  <table border="1" data-bbox="382 1035 1383 1208"> <thead> <tr> <th>編號</th> <th>名稱</th> <th>位置</th> <th>介面類型</th> <th>通訊協議</th> <th>站號</th> </tr> </thead> <tbody> <tr> <td>本機 HMI</td> <td>Local HMI</td> <td>本機</td> <td>停用</td> <td>N/A</td> <td>N/A</td> </tr> <tr> <td>本機 伺服器</td> <td>Free Protocol</td> <td>本機</td> <td>COM1(960...)</td> <td>RS232</td> <td>0</td> </tr> <tr> <td>遠端 PLC 1</td> <td>FATEK FB Series</td> <td>遠端(IP:210.68.117.224, 連接...</td> <td>COM1(960...)</td> <td>RS232</td> <td>1</td> </tr> </tbody> </table>	編號	名稱	位置	介面類型	通訊協議	站號	本機 HMI	Local HMI	本機	停用	N/A	N/A	本機 伺服器	Free Protocol	本機	COM1(960...)	RS232	0	遠端 PLC 1	FATEK FB Series	遠端(IP:210.68.117.224, 連接...	COM1(960...)	RS232	1
編號	名稱	位置	介面類型	通訊協議	站號																				
本機 HMI	Local HMI	本機	停用	N/A	N/A																				
本機 伺服器	Free Protocol	本機	COM1(960...)	RS232	0																				
遠端 PLC 1	FATEK FB Series	遠端(IP:210.68.117.224, 連接...	COM1(960...)	RS232	1																				

`device_type` 是設備類型和 PLC 中資料的編碼方式。例如：如果 `device_type` 是 `LW_BIN`，那麼讀取的設備類型為 `LW`，資料編碼方式為 `BIN`。如果使用 `BIN` 編碼方式，“`_BIN`”可以忽略。

如果 `device_type` 是 `LW_BCD`，表示設備類型 `LW`，資料的編碼方式為 `BCD` 格式。

`address_offset` 是 PLC 中的地址偏移量。

例如，`StringGet(read_data_1[0], "FATEK FB Series", RT, 5, 1)` 代表讀取的設備位址偏移量為 5。

如果 `address_offset` 使用格式為 “`N#AAAAAA`”，`N` 表示 PLC 的站號，`AAAAAA` 表示位址偏移量。此情況一般使用在同一個串列埠上連接有多台 PLC 或者控制器的情況下。例如：`StringGet(read_data_1[0], "FATEK FB Series", RT, 2#5, 1)` 表示讀取站號為 2 的 PLC 的資料。如果 `StringGet()` 使用“系統參數 / 設備列表”中設定的默認的站號，在此可以不填這個站號。



從 PLC 中讀取的資料個數，由 `data_count` 的值來決定，因 `read_data` 變數僅接受 `char` 陣列型態。如下表所示：

<code>read_data</code> 的類型	<code>data_count</code> 的值	讀取 16 位元數據的個數
char (8-bit)	1	1
char (8-bit)	2	1

因為一個 WORD (16 位元) 等於 2 個 ASCII 字元的長度，當設備類型長度為 WORD 時，根據上表，讀取 2 個 ASCII 字元實際上是讀 1 個 WORD 的數據。

舉例

```

macro_command main()
char str1[20]

// 讀取 LW0 ~ LW9 共 10 個 WORD 到變數 str1[0] 到 str1[19] 中
// 因 1 WORD 可存放 2 個 ASCII 字元，欲讀取 20 個 ASCII 字元
// 實際上共讀取了 10 個WORD
StringGet(str1[0], "Local HMI", LW, 0, 20)

end macro_command

```

函數名稱	StringGetEx
語法	StringGetEx (read_data[start], device_函數名稱, device_type, address_offset, data_count)
描述	<p>獲取 PLC 的資料，不等待 PLC 回應，逕自往下執行。</p> <p>read_data、device_函數名稱、device_type、address_offset 和 data_count 的說明和 GetData 相同。</p>
舉例	<pre>macro_command main() char str1[20] short test=0 // 當 MODBUS 設備未回應，test = 1 將照常執行 StringGetEx(str1[0], "MODBUS RTU", 4x, 0, 20) test = 1 // 當 MODBUS 設備未回應，test = 2 將不會被執行，直到得到回應 StringGet(str1[0], "MODBUS RTU", 4x, 0, 20) test = 2 end macro_command</pre>

函數名稱	StringSet
語法	StringSet (send_data[start], device_函數名稱, device_type, address_offset, data_count)
描述	<p>將數據寫到 PLC 中。字串資料保存在 send_data[start]~send_data[start+data_count-1] 中，send_data 必須為一維字元陣列型態。</p> <p>data_count 是寫入到 PLC 中字元資料的個數，可以是常數也可以是變數。</p> <p>device_函數名稱詳見上面的說明，在此不在說明。</p> <p>device_type 是設備類型和 PLC 中資料的編碼方式。例如：如果 device_type 是 LW_BIN，那麼讀取的設備類型為 LW，資料編碼方式為 BIN。如果使用 BIN 編碼方式，“_BIN”可以忽略。</p> <p>如果 device_type 是 LW_BCD，表示設備類型 LW，資料的編碼方式為 BCD 格式。</p> <p>address_offset 是 PLC 中的地址偏移量。</p> <p>例如，StringSet(read_data_1[0], “FATEK FB Series”, RT, 5, 1) 代表讀取的設</p>

備位址偏移量為 5。

如果 `address_offset` 使用格式為 “N#AAAAA”，N 表示 PLC 的站號，AAAAAA 表示位址偏移量。此情況一般使用在同一個串列埠上連接有多台 PLC 或者控制器的情況下。例如：`StringSet(send_data_1[0], "FATEK FB Series", RT, 2#5, 1)` 表示設定站號為 2 的 PLC 的資料。如果 `StringSet()` 使用“系統參數 / 設備列表”中設定的默認的站號，在此可以不填這個站號。

設定到 PLC 的資料個數，根據 `data_count` 的值來決定，因 `send_data` 變數僅接受 `char` 陣列型態。如下表所示：

<code>send_data</code> 的類型	<code>data_count</code> 的值	設定 16 位元數據的個數
<code>char (8-bit)</code>	1	1
<code>char (8-bit)</code>	2	1

因為一個 WORD (16 位元) 等於 2 個 ASCII 字元的長度，當設備類型長度為 WORD 時，根據上表，寫入 2 個 ASCII 字元實際上是寫 1 個 WORD 的數據。巨集指令會以先寫 low byte 再寫 hight byte 的順序，依序將 ASCII 字元寫入。

使用字元顯示物件顯示數據時，`data_count` 必須填入 2 的倍數才能正確顯示。例如：

```
macro_command main()
char src1[10]="abcde"
StringSet(src1[0], "Local HMI", LW, 0, 5)
end macro_command
```

字元顯示物件顯示如下：



abcd

當 `data_count` 為一個大於或等於字串長度的偶數時，可以完整顯示整個字串：

```
macro_command main()
char src1[10]="abcde"
StringSet(src1[0], "Local HMI", LW, 0, 6)
end macro_command
```



abcde

舉例

```
macro_command main()
char str1[10] = "abcde"
```

	<pre>// 字串 str1 寫入 LW0 ~ LW2 共三個 WORD // 即使 data_count 為 10，寫到第 3 個 WORD 時發現字串已結束， // 便不再寫入後面的數據 StringSet(str1[0], "Local HMI", LW, 0, 10) end macro_command</pre>
--	--

函數名稱	StringSetEx
語法	StringSetEx(send_data[start], device_函數名稱, device_type, address_offset, data_count)
描述	將數據寫到 PLC 中，不等待 PLC 回應，逕自往下執行。 send_data、device_函數名稱、device_type、address_offset 和 data_count 的說明和 StringSet 相同。
舉例	<pre>macro_command main() char str1[20] = "abcde" short test = 0 // 當 MODBUS 設備未回應，test = 1 將照常執行 StringSetEx(str1[0], "MODBUS RTU", 4x, 0, 20) test = 1 // 當 MODBUS 設備未回應，test = 2 將不會被執行，直到得到回應 StringSet(str1[0], "MODBUS RTU", 4x, 0, 20) test = 2 end macro_command</pre>

函數名稱	StringCopy
語法	<pre>success = StringCopy ("source", destination[start]) 或 success = StringCopy (source[start], destination[start])</pre>
描述	<p>利用此函數進行字串的複製。此函數可以將傳入的靜態字串 (以雙引號""括起來的字串)，或是存放在字元陣列中的字串複製到目的 buffer。</p> <p>來源字串 source 可以為靜態字串 (如 "source")或是一維字元陣列變數 (如 source[start])。</p> <p>destination[start] 必須為一維字元陣列變數。</p> <p>複製完畢會回傳一 bool 型態的值給 success 欄位。當複製成功，success 等於 true，否則等於 false。當來源字串的長度大於目的 buffer 的大小時，將不做任何處理，並回傳 false 到 success 欄位。</p> <p>success 欄位可寫可不寫。</p>
舉例	<pre>macro_command main() char src1[5] = "abcde" char dest1[5] bool success1 success1 = StringCopy(src1[0], dest1[0]) // success1=true , dest1為"abcde" char dest2[5] bool success2 success2 = StringCopy("12345", dest2[0]) // success2 = true , dest2為 "12345" char src3[10] = "abcdefghijkl" char dest3[5] bool success3 success3 = StringCopy(src3[0], dest3[0]) // success3 = false , dest3內容不變 char src4[10] = "abcdefghijkl" char dest4[5] bool success4 success4 = StringCopy(src4[5], dest4[0]) // success4=true , dest4為"fghij" end macro_command</pre>

函數名稱	StringDecAsc2Bin
語法	<pre>success = StringDecAsc2Bin(source[start], destination) 或 success = StringDecAsc2Bin("source", destination)</pre>
描述	<p>此函數將十進位字串轉換成整數。</p> <p>來源字串 <code>source</code> 可以為靜態字串 (如 “<code>source</code>”)或是一維字元陣列變數 (如 <code>source[start]</code>)。</p> <p><code>destination</code> 必須為一變數，用以存放轉換後的整數值。</p> <p>執行完畢會回傳一 <code>bool</code> 型態的值給 <code>success</code> 欄位。當轉換成功，<code>success</code> 等於 <code>true</code>，否則等於 <code>false</code>。</p> <p>來源字串必需為十進位字串，若其包含 ‘0’ ~ ‘9’ 以外的字元，函數將會回傳 <code>false</code>。</p> <p><code>success</code> 欄位可寫可不寫。</p>
舉例	<pre>macro_command main() char src1[5] = "12345" int result1 bool success1 success1 = StringDecAsc2Bin(src1[0], result1) // success1 = true, result1 為12345 char result2 bool success2 success2 = StringDecAsc2Bin("32768", result2) // success2 = true, 但結果超出 result2 所能表達的範圍 char src3[2] = "4b" char result3 bool success3 success3 = StringDecAsc2Bin (src3[0], result3) // success3 = false, 因 src3 包含 '0' ~ '9' 以外的字元 end macro_command</pre>

函數名稱	StringBin2DecAsc
語法	success = StringBin2DecAsc (source, destination[start])
描述	<p>此函數將整數轉換成十進位字串。</p> <p>來源 source 可以為常數或變數。</p> <p>destination[start] 必須為一維字元陣列變數，用以存放轉換後的十進位字串。</p> <p>執行完畢會回傳一 bool 型態的值給 success 欄位。當轉換成功，success 等於 true，否則等於 false。</p> <p>若轉換後的十進位字串長度大於目標陣列的大小，函數將會回傳 false。</p> <p>success 欄位可寫可不寫。</p>
舉例	<pre>macro_command main() int src1 = 2147483647 char dest1[20] bool success1 success1 = StringBin2DecAsc(src1, dest1[0]) // success1 = true, dest1為 "2147483647" short src2 = 0x3c char dest2[20] bool success2 success2 = StringBin2DecAsc(src2, dest2[0]) // success2 = true, dest2為 "60" int src3 = 2147483647 char dest3[5] bool success3 success3 = StringBin2DecAsc(src3, dest3[0]) // success3 = false, dest3 內容不變 end macro_command</pre>

函數名稱	StringDecAsc2Float
語法	<pre>success = StringDecAsc2Float (source[start], destination) 或 success = StringDecAsc2Float ("source", destination)</pre>
描述	<p>此函數將十進位字串轉換成浮點數。</p> <p>來源字串 <code>source</code> 可以為靜態字串 (如 “<code>source</code>”) 或是一維字元陣列變數 (如 <code>source[start]</code>)。</p> <p><code>destination</code> 必須為一變數，用以存放轉換後的浮點數值。</p> <p>執行完畢會回傳一 <code>bool</code> 型態的值給 <code>success</code> 欄位。當轉換成功，<code>success</code> 等於 <code>true</code>，否則等於 <code>false</code>。</p> <p>來源字串必需為十進位字串，若其包含 ‘0’ ~ ‘9’ 或 ‘.’ 以外的字元，函數將會回傳 <code>false</code>。</p> <p><code>success</code> 欄位可寫可不寫。</p>
舉例	<pre>macro_command main() char src1[10] = "12.345" float result1 bool success1 success1 = StringDecAsc2Float(src1[0], result1) // success1 = true, result1 為 12.345 float result2 bool success2 success2 = StringDecAsc2Float("1.234567890", result2) // success2 = true, 但結果超出 result2 所能表達的範圍，可能喪失精確度 char src3[2] = "4b" float result3 bool success3 success3 = StringDecAsc2Float(src3[0], result3) // success3 = false, 因 src3 包含 ‘0’ ~ ‘9’ 或 ‘.’ 以外的字元 end macro_command</pre>

函數名稱	StringFloat2DecAsc
語法	success = StringFloat2DecAsc(source, destination[start])
描述	<p>此函數將浮點數轉換成十進位字串。</p> <p>來源 <code>source</code> 可以為常數或變數。</p> <p><code>destination[start]</code> 必須為一維字元陣列變數，用以存放轉換後的十進位字串。</p> <p>執行完畢會回傳一 <code>bool</code> 型態的值給 <code>success</code> 欄位。當轉換成功，<code>success</code> 等於 <code>true</code>，否則等於 <code>false</code>。</p> <p>若轉換後的十進位字串長度大於目標陣列的大小，函數將會回傳 <code>false</code>。</p> <p><code>success</code> 欄位可寫可不寫。</p>
舉例	<pre>macro_command main() float src1 = 1.2345 char dest1[20] bool success1 success1 = StringFloat2DecAsc(src1, dest1[0]) // success1 = true, dest1為 "1.2345" float src2 = 1.23456789 char dest2 [20] bool success2 success2 = StringFloat2DecAsc(src2, dest2 [0]) // success2 = true, 但可能喪失精確度 float src3 = 1.2345 char dest3[5] bool success3 success3 = StringFloat2DecAsc(src3, dest3 [0]) // success3 = false, dest3 內容不變 end macro_command</pre>

函數名稱	StringHexAsc2Bin
語法	success = StringHexAsc2Bin (source[start], destination) 或 success = StringHexAsc2Bin ("source", destination)
描述	<p>此函數將十六進位字串轉換成整數。</p> <p>來源字串 source 可以為靜態字串 (如 "source") 或是一維字元陣列變數 (如 source[start])。</p> <p>destination 必須為一變數，用以存放轉換後的整數值。</p> <p>執行完畢會回傳一 bool 型態的值給 success 欄位。當轉換成功，success 等於 true，否則等於 false。</p> <p>來源字串必需為十六進位字串，若其包含 '0' ~ '9' 或 'a' ~ 'f' 或 'A' ~ 'F' 以外的字元，函數將會回傳 false。</p> <p>success 欄位可寫可不寫。</p>
舉例	<pre>macro_command main() char src1[5] = "0x3c" int result1 bool success1 success1 = StringHexAsc2Bin(src1[0], result1) // success1 = true, result1 為 3c short result2 bool success2 success2 = StringDecAsc2Bin("1a2b3c4d", result2) // success2 = true, 但結果超出 result2 所能表達的範圍, result2 = 3c4d char src3[2] = "4g" char result3 bool success3 success3 = StringDecAsc2Bin (src3[0], result3) // success3=false, 因 src3 包含 '0' ~ '9' 或 'a' ~ 'f' 或 'A' ~ 'F' 以外的字元 end macro_command</pre>

函數名稱	StringBin2HexAsc
語法	<code>success = StringBin2HexAsc (source, destination[start])</code>
描述	<p>此函數將整數轉換成十六進位字串。</p> <p>來源 <code>source</code> 可以為常數或變數。</p> <p><code>destination[start]</code> 必須為一維字元陣列變數，用以存放轉換後的十六進位字串。</p> <p>執行完畢會回傳一 <code>bool</code> 型態的值給 <code>success</code> 欄位。當轉換成功，<code>success</code> 等於 <code>true</code>，否則等於 <code>false</code>。</p> <p>若轉換後的十六進位字串長度大於目標陣列的大小，函數將會回傳 <code>false</code>。</p> <p><code>success</code> 欄位可寫可不寫。</p>
舉例	<pre>macro_command main() int src1 = 20 char dest1[20] bool success1 success1 = StringBin2HexAsc(src1, dest1[0]) // success1 = true, dest1 為 "14" short src2 = 0x3c char dest2[20] bool success2 success2 = StringBin2HexAsc(src2, dest2[0]) // success2 = true, dest2 為 "3c" int src3 = 0x1a2b3c4d char dest3[6] bool success3 success3 = StringBin2HexAsc(src3, dest3[0]) // success3 = false, dest3 內容不變 end macro_command</pre>

函數名稱	StringMid
語法	<pre>success = StringMid (source[start], count, destination[start]) 或 success = StringMid ("string", start, count, destination[start])</pre>
描述	<p>利用此函數可將一個字串中的某一段子字串提取出來。</p> <p>來源字串 <code>source</code> 可以為靜態字串 (如 “<code>source</code>”) 或是一維字元陣列變數 (如 <code>source[start]</code>)。當來源字串為字元陣列變數時，由陣列下標決定子字串起始位置。當來源字串為靜態字串時，由第二個參數 <code>start</code> 決定子字串起始位置。<code>count</code> 決定要提取的子字串長度。</p> <p><code>destination[start]</code> 必須為一維字元陣列變數，用以存放提取出來的子字串。執行完畢會回傳一 <code>bool</code> 型態的值給 <code>success</code> 欄位。當執行成功，<code>success</code> 等於 <code>true</code>，否則等於 <code>false</code>。</p> <p>若提取出來的子字串長度大於目標陣列的大小，函數將會回傳 <code>false</code>。</p> <p><code>success</code> 欄位可寫可不寫。</p>
舉例	<pre>macro_command main() char src1[20] = "abcdefghijklmnopqrst" char dest1[20] bool success1 success1 = StringMid(src1[5], 6, dest1[0]) // success1 = true, dest1 為 "fghijk" char src2[20] = "abcdefghijklmnopqrst" char dest2[5] bool success2 success2 = StringMid(src2[5], 6, dest2[0]) // success2 = false, dest2 內容不變 char dest3[20] = "12345678901234567890" bool success3 success3 = StringMid("abcdefghijklmnopqrst", 5, 5, dest3[15]) // success3 = true, dest3 = "123456789012345fghij" end macro_command</pre>

函數名稱	StringLength
語法	length = StringLength (source[start]) 或 length = StringLength ("source")
描述	取得字串的長度。 來源字串 source 可以為靜態字串 (如 "source") 或是一維字元陣列變數(如 source[start])。 函數回傳值代表來源字串的長度。
舉例	<pre>macro_command main() char src1[20] = "abcde" int length1 length1= StringLength(src1[0]) // length1 = 5 char src2[20] = { 'a', 'b' , 'c' , 'd' , 'e' } int length2 length2= StringLength(src2[0]) // length2 = 20 char src3[20] = "abcdefghij" int length3 length3 = StringLength(src3 [2]) // length3 = 8 end macro_command</pre>

函數名稱	StringCat
語法	<pre>success = StringCat (source[start], destination[start]) 或 success = StringCat ("source", destination[start])</pre>
描述	<p>利用此函數將來源字串銜接於目標字串之後。此函數執行成功後，目標字串將等於兩字串銜接後的結果。</p> <p>來源字串 <code>source</code> 可以為靜態字串 (如 “<code>source</code>”) 或是一維字元陣列變數 (如 <code>source[start]</code>)。</p> <p><code>destination[start]</code> 必須為一維字元陣列變數。</p> <p>執行完畢會回傳一 <code>bool</code> 型態的值給 <code>success</code> 欄位。當執行成功，<code>success</code> 等於 <code>true</code>，否則等於 <code>false</code>。當兩字串銜接後的長度超過目標陣列的長度時，目標字串將保留銜接以前的內容，不做任何改變，並回傳 <code>false</code>。</p>
舉例	<pre>macro_command main() char src1[20] = "abcdefghijkl" char dest1[20] = "1234567890" bool success1 success1= StringCat(src1[0], dest1[0]) // success1 = true , dest1 = "123456790abcdefghijkl" char dest2 [10] = "1234567890" bool success2 success2= StringCat("abcde", dest2 [0]) // success2 = false , dest2 內容不變 char src3[20] = "abcdefghijkl" char dest3[20] bool success3 success3 = StringCat(src3[0], dest3[15]) // success3 = false , dest3內容不變 end macro_command</pre>

函數名稱	StringCompare
語法	<pre>ret = StringCompare (str1[start], str2[start]) ret = StringCompare ("string1", str2[start]) ret = StringCompare (str1[start], "string2") ret = StringCompare ("string1", "string2")</pre>
描述	<p>比較兩字串的內容是否相等。此函數將大小寫視為不同。</p> <p>兩個傳入的字串皆可以為靜態字串 (如 “source”)或是一維字元陣列變數 (如 source[start])。</p> <p>執行完畢會回傳一 bool 型態的值給 ret 欄位。若兩字串相等，ret 為 true，否則為 false。</p>
舉例	<pre>macro_command main() char a1[20] = "abcde" char b1[20] = "ABCDE" bool ret1 ret1 = StringCompare(a1[0], b1[0]) // ret1 = false char a2[20] = "abcde" char b2[20] = "abcde" bool ret2 ret2 = StringCompare(a2[0], b2[0]) // ret2 = true char a3 [20] = "abcde" char b3[20] = "abcdefg" bool ret3 ret3 = StringCompare(a3[0], b3[0]) // ret3 = false end macro_command</pre>

函數名稱	StringCompareNoCase
語法	<pre>ret = StringCompareNoCase(str1[start], str2[start]) ret = StringCompareNoCase("string1", str2[start]) ret = StringCompareNoCase(str1[start], "string2") ret = StringCompareNoCase("string1", "string2")</pre>
描述	<p>比較兩字串的內容是否相等。此函數將大小寫視為相同。</p> <p>兩個傳入的字串皆可以為靜態字串 (如 “source”) 或是一維字元陣列變數(如 source[start])。</p> <p>執行完畢會回傳一 bool 型態的值給 ret 欄位。若兩字串相等，ret 為 true，否則為 false。</p>
舉例	<pre>macro_command main() char a1[20] = "abcde" char b1[20] = "ABCDE" bool ret1 ret1 = StringCompareNoCase(a1[0], b1[0]) // ret1 = true char a2[20] = "abcde" char b2[20] = "abcde" bool ret2 ret2 = StringCompareNoCase(a2[0], b2[0]) // ret2 = true char a3 [20] = "abcde" char b3[20] = "abcdefg" bool ret3 ret3 = StringCompareNoCase(a3[0], b3[0]) // ret3 = false end macro_command</pre>

函數名稱	StringFind
語法	<pre>position = StringFind (source[start], target[start]) position = StringFind ("source", target[start]) position = StringFind (source[start], "target") position = StringFind ("source", "target")</pre>
描述	<p>尋找某一字串 (target) 在另一個字串 (source) 裡第一次出現的位置。 兩個傳入的字串皆可以為靜態字串 (如 “source”) 或是一維字元陣列變數(如 source[start])。 執行完畢會回傳 target 字串在 source 字串裡出現的位置。Source 字串由 0 開始遞增為字元編索引值。若 source 字串存在一個子字串，其所包含的字元與排列順序跟 target 字串完全相等，則函數會回傳此子字串開頭的索引值，若沒有找到，則回傳 -1。</p>
舉例	<pre>macro_command main() char src1[20] = "abcde" char target1[20] = "cd" bool pos1 pos1 = StringFind(src1[0], target1[0]) // pos1 = 2 char target2[20] = "ce" bool pos2 pos2 = StringFind("abcde", target2[0]) // pos2 = -1 char src3[20] = "abcde" bool pos3 pos3 = StringFind(src3[3], "cd") // pos3 = -1 end macro_command</pre>

函數名稱	StringReverseFind
語法	<pre>position = StringReverseFind (source[start], target[start]) position = StringReverseFind ("source", target[start]) position = StringReverseFind (source[start], "target") position = StringReverseFind ("source", "target")</pre>
描述	<p>尋找某一字串 (target) 在另一個字串 (source) 裡最後一次出現的位置。 兩個傳入的字串皆可以為靜態字串 (如 "source") 或是一維字元陣列變數(如 source[start])。 執行完畢會回傳 target 字串在 source 字串裡最後一次出現的位置。source 字串由 0 開始遞增為字元編索引值。若 source 字串存在一個子字串，其所 包含的字元與排列順序跟 target 字串完全相等，則函數會回傳此子字串開頭 的索引值，若沒有找到，則回傳 -1。若 source 字串中存在多個與 target 字 串相等的子字串，則回傳最後一個出現的子字串的位置。</p>
舉例	<pre>macro_command main() char src1[20] = "abcdeabcde" char target1[20] = "cd" bool pos1 pos1 = StringReverseFind(src1[0], target1[0]) // pos1 = 7 char target2[20] = "ce" bool pos2 pos2 = StringReverseFind("abcdeabcde" , target2[0]) // pos2 = -1 char src3[20] = "abcdeabcde" bool pos3 pos3 = StringReverseFind(src3[6], "ab") // pos3 = -1 end macro_command</pre>

函數名稱	StringFindOneOf
語法	<pre>position = StringFindOneOf (source[start], target[start]) position = StringFindOneOf ("source", target[start]) position = StringFindOneOf (source[start], "target") position = StringFindOneOf ("source", "target")</pre>
描述	<p>尋找 target 字串中任一個字元在 source 字串中第一次出現的位置。</p> <p>兩個傳入的字串皆可以為靜態字串 (如 “source”) 或是一維字元陣列變數(如 source[start])。</p> <p>執行完畢會回傳 target 字串中任一個字元在 source 字串裡第一次出現的位置，即 source 字串中為該字元編的索引值 (由 0 開始)。若沒有找到，則回傳 -1。</p>
舉例	<pre>macro_command main() char src1[20] = "abcdeabcde" char target1[20] = "sdf" bool pos1 pos1 = StringFindOneOf(src1[0], target1[0]) // pos1 = 3 char src2[20] = "abcdeabcde" bool pos2 pos2 = StringFindOneOf(src2[1], "agi") // pos2 = 4 char target3 [20] = "bus" bool pos3 pos3 = StringFindOneOf("abcdeabcde", target3[1]) // pos3 = -1 end macro_command</pre>

函數名稱	StringIncluding
語法	<pre>success = StringIncluding (source[start], set[start], destination[start]) success = StringIncluding ("source", set[start], destination[start]) success = StringIncluding (source[start], "set", destination[start]) success = StringIncluding ("source", "set", destination[start])</pre>
描述	<p>提取 <code>source</code> 字串中某個以索引值 0 的字元 (第一個字元) 開頭的子字串，而且此子字串的每個字元都能在 <code>set</code> 字串中找到相同的字元。此函數將會從 <code>source</code> 字串的第一個字元開始尋找，直到找到不存在於 <code>set</code> 字串中的字元為止。</p> <p><code>source</code> 字串與 <code>set</code> 字串皆可以為靜態字串 (如 “<code>source</code>”) 或是一維字元陣列變數 (如 <code>source[start]</code>)。</p> <p>執行完畢會回傳一 <code>bool</code> 型態的值給 <code>success</code> 欄位。當執行成功，<code>success</code> 等於 <code>true</code>，否則等於 <code>false</code>。當提取出來的字串長度大於目標陣列的大小，將會回傳 <code>false</code>。</p>
舉例	<pre>macro_command main() char src1[20] = "cabbageabc" char set1[20] = "abc" char dest1[20] bool success1 success1 = SDtringIncluding(src1[0], set1[0], dest1[0]) // success1 = true, dest1 = "cabba" char src2[20] = "gecabba" char dest2[20] bool success2 success2 = StringIncluding(src2[0], "abc", dest2[0]) // success2 = true, dest2 = "" char set3[20] = "abc" char dest3[4] bool success3 success3 = StringIncluding("cabbage", set3[0], dest3[0]) // success3 = false, dest3內容不變 end macro_command</pre>

函數名稱	StringExcluding
語法	<pre>success = StringExcluding (source[start], set[start], destination[start]) success = StringExcluding ("source", set[start], destination[start]) success = StringExcluding (source[start], "set", destination[start]) success = StringExcluding ("source", "set", destination[start])</pre>
描述	<p>提取 <code>source</code> 字串中某個以索引值 0 的字元 (第一個字元) 開頭的子字串，而且此子字串的每個字元必不存在於 <code>set</code> 字串中。此函數將會從 <code>source</code> 字串的第一個字元開始尋找，直到找到存在於 <code>set</code> 字串中的字元為止。</p> <p><code>source</code> 字串與 <code>set</code> 字串皆可以為靜態字串 (如 “<code>source</code>”) 或是一維字元陣列變數 (如 <code>source[start]</code>)。</p> <p>執行完畢會回傳一 <code>bool</code> 型態的值給 <code>success</code> 欄位。當執行成功，<code>success</code> 等於 <code>true</code>，否則等於 <code>false</code>。當提取出來的字串長度大於目標陣列的大小，將會回傳 <code>false</code>。</p>
舉例	<pre>macro_command main() char src1[20] = "cabbageabc" char set1[20] = "ge" char dest1[20] bool success1 success1 = StringExcluding(src1[0], set1[0], dest1[0]) // success1 = true, dest1 = "cabba" char src2[20] = "cabbage" char dest2[20] bool success2 success2 = StringExcluding(src2[0], "abc", dest2[0]) // success2 = true, dest2 = "" char set3[20] = "ge" char dest3[4] bool success3 success3 = StringExcluding("cabbage", set3[0], dest3[0]) // success3 = false, dest3內容不變 end macro_command</pre>

函數名稱	StringToUpper
------	---------------

語法	<pre>success = StringToUpper (source[start], destination[start]) success = StringToUpper ("source", destination[start])</pre>
描述	<p>將 <code>source</code> 字串的字元全部轉換成大寫，並寫入 <code>destination</code> 字串。</p> <p><code>source</code> 字串可以為靜態字串 (如 “<code>source</code>”) 或是一維字元陣列變數 (如 <code>source[start]</code>)。</p> <p>執行完畢會回傳一 <code>bool</code> 型態的值給 <code>success</code> 欄位。當執行成功，<code>success</code> 等於 <code>true</code>，否則等於 <code>false</code>。<code>source</code> 字串長度大於目標陣列的大小，將會回傳 <code>false</code>。</p>
舉例	<pre>macro_command main() char src1[20] = "aBcDe" char dest1[20] bool success1 success1 = StringToUpper(src1[0], dest1[0]) // success1 = true, dest1 = "ABCDE" char dest2[4] bool success2 success2 = StringToUpper("aBcDe", dest2[0]) // success2 = false, dest2內容不變 end macro_command</pre>

函數名稱	<code>StringToLower</code>
語法	<pre>success = StringToLower (source[start], destination[start]) success = StringToLower ("source", destination[start])</pre>
描述	<p>將 <code>source</code> 字串的字元全部轉換成小寫，並寫入 <code>destination</code> 字串。</p> <p><code>source</code> 字串可以為靜態字串 (如 “<code>source</code>”) 或是一維字元陣列變數 (如 <code>source[start]</code>)。</p> <p>執行完畢會回傳一 <code>bool</code> 型態的值給 <code>success</code> 欄位。當執行成功，<code>success</code> 等於 <code>true</code>，否則等於 <code>false</code>。<code>source</code> 字串長度大於目標陣列的大小，將會回傳 <code>false</code>。</p>
舉例	<pre>macro_command main() char src1[20] = "aBcDe" char dest1[20] bool success1 success1 = StringToLower(src1[0], dest1[0]) // success1 = true, dest1 = "abcde"</pre>

	<pre>char dest2[4] bool success2 success2 = StringToUpper("aBcDe", dest2[0]) // success2 = false, dest2 內容不變 end macro_command</pre>
--	---

函數名稱	StringToReverse
語法	<pre>success = StringToReverse (source[start], destination[start]) success = StringToReverse ("source", destination[start])</pre>
描述	<p>將 source 字串反轉，並寫入 destination 字串。</p> <p>source 字串可以為靜態字串（如 “source”）或是一維字元陣列變數（如 source[start]）。</p> <p>執行完畢會回傳一 bool 型態的值給 success 欄位。當執行成功，success 等於 true，否則等於 false。source 字串長度大於目標陣列的大小，將會回傳 false。</p>
舉例	<pre>macro_command main() char src1[20] = "abcde" char dest1[20] bool success1 success1 = StringToUpper(src1[0], dest1[0]) // success1 = true, dest1 = "edcba" char dest2[4] bool success2 success2 = StringToUpper("abcde", dest2[0]) // success2 = false, dest2 內容不變 end macro_command</pre>

函數名稱	StringTrimLeft
語法	<pre>success = StringTrimLeft (source[start], set[start], destination[start]) success = StringTrimLeft ("source", set[start], destination[start]) success = StringTrimLeft (source[start], "set", destination[start]) success = StringTrimLeft ("source", "set", destination[start])</pre>
描述	<p>從 <code>source</code> 字串的第一個字元開始往後尋找，若找到與 <code>set</code> 字串相同的字元便裁剪掉該字元，直到遇到不存在於 <code>set</code> 字串中的字元為止。</p> <p><code>source</code> 字串與 <code>set</code> 字串皆可以為靜態字串 (如 “<code>source</code>”) 或是一維字元陣列變數 (如 <code>source[start]</code>)。</p> <p>執行完畢會回傳一 <code>bool</code> 型態的值給 <code>success</code> 欄位。當執行成功，<code>success</code> 等於 <code>true</code>，否則等於 <code>false</code>。當裁剪完後的字串長度大於目標陣列的大小，將會回傳 <code>false</code>。</p>
舉例	<pre>macro_command main() char src1[20] = "# *a*#bc" char set1[20] = "# *" char dest1[20] bool success1 success1 = StringTrimLeft (src1[0], set1[0], dest1[0]) // success1 = true, dest1 = "a*#bc" char set2[20] = {'#', ' ', '*'} char dest2[4] success2 = StringTrimLeft ("# *a*#bc", set2[0], dest2[0]) // success2 = false, dest2 內容不變 char src3[20] = "abc *#" char dest3[20] bool success3 success3 = StringTrimLeft (src3[0], "# *", dest3[0]) // success3 = true, dest3 = "abc *#" end macro_command</pre>

函數名稱	StringTrimRight
語法	<pre>success = StringTrimRight (source[start], set[start], destination[start]) success = StringTrimRight ("source", set[start], destination[start]) success = StringTrimRight (source[start], "set", destination[start]) success = StringTrimRight ("source", "set", destination[start])</pre>
描述	<p>從 <code>source</code> 字串的最後一個字元開始往前尋找，若找到與 <code>set</code> 字串相同的字元便裁剪掉該字元，直到遇到不存在於 <code>set</code> 字串中的字元為止。</p> <p><code>source</code> 字串與 <code>set</code> 字串皆可以為靜態字串 (如 “<code>source</code>”) 或是一維字元陣列變數 (如 <code>source[start]</code>)。</p> <p>執行完畢會回傳一 <code>bool</code> 型態的值給 <code>success</code> 欄位。當執行成功，<code>success</code> 等於 <code>true</code>，否則等於 <code>false</code>。當裁剪完後的字串長度大於目標陣列的大小，將會回傳 <code>false</code>。</p>
舉例	<pre>macro_command main() char src1[20] = "# *a*#bc# * " char set1[20] = "# *" char dest1[20] bool success1 success1 = StringTrimRight(src1[0], set1[0], dest1[0]) // success1 = true, dest1="# *a*#bc" char set2[20] = {'#', ' ', '*'} char dest2[20] success2 = StringTrimRight("# *a*#bc", set2[0], dest2[0]) // success2 = true, dest2 = "# *a*#bc" char src3[20] = "ab**c *#" char dest3[4] bool success3 success3 = StringTrimRight(src3[0], "# *", dest3[0]) // success3 = false, dest3 內容不變 end macro_command</pre>

函數名稱	StringInsert
語法	<pre>success = StringInsert (pos, insert[start], destination[start]) success = StringInsert (pos, "insert", destination[start]) success = StringInsert (pos, insert[start], length, destination[start]) success = StringInsert (pos, "insert", length, destination[start])</pre>
描述	<p>將 <code>insert</code> 字串插入到目標字串中的特定位置，插入位置由 <code>pos</code> 所指定。</p> <p><code>Insert</code> 字串可以為靜態字串（如 “<code>insert</code>”）或是一維字元陣列變數（如 <code>insert[start]</code>）。</p> <p>使用者亦可以在 <code>length</code> 欄位指定 <code>insert</code> 字串的長度。</p> <p>執行完畢會回傳一 <code>bool</code> 型態的值給 <code>success</code> 欄位。當執行成功，<code>success</code> 等於 <code>true</code>，否則等於 <code>false</code>。當插入完成後的字串長度大於目標陣列的大小，將會回傳 <code>false</code>。</p>
舉例	<pre>macro_command main() char str1[20] = "but the question is" char str2[10] = ", that is" char dest[40] = "to be or not to be" bool success success = StringInsert(18, str1[3], 13, dest[0]) // success = true, dest = "to be or not to be the question" success = StringInsert(18, str2[0], dest[0]) // success=true, dest="to be or not to be, that is the question" success = StringInsert(0, "Hamlet:", dest[0]) // success = false, dest內容不變 end macro_command</pre>

18.6.7 其他函數

函數名稱	Beep
語法	Beep ()
描述	發出系統警笛音。 此函數可發出 800 赫茲，30 毫秒的系統警笛音。
舉例	<pre>macro_command main() Beep() end macro_command</pre>

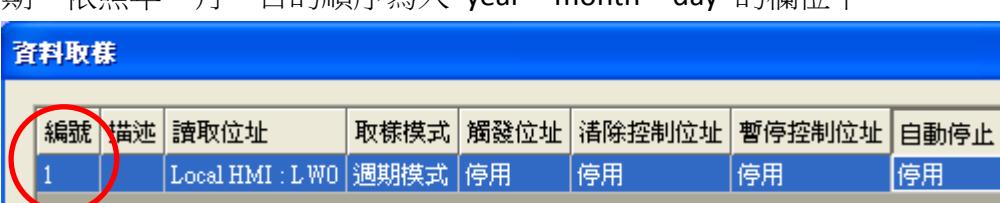
函數名稱	Buzzer
語法	Buzzer ()
描述	開啟 / 關閉 蜂鳴器
舉例	<pre>char on = 1, off = 0 Buzzer(on) // turn on the buzzer DELAY(1000) // delay 1 second Buzzer(off) // turn off the buzzer DELAY(500) // delay 500ms Buzzer(1) // turn on the buzzer DELAY(1000) // delay 1 second Buzzer(0) // turn off the buzzer</pre>

函數名稱	SYNC_TRIG_MACRO
語法	SYNC_TRIG_MACRO(macro_id)
描述	<p>一個執行中的巨集指令可以使用此函數利用同步的方式觸發執行其他的巨集指令，使用 macro_id 指定被觸發的巨集指令 【編號】。</p> <p>使用此函數的巨集指令將暫停執行，直到被觸發的巨集指令執行完成才會繼續執行後續的命令。</p> <p>macro_id 可以是常數或使用變數來表示。</p>
舉例	<pre>macro_command main() char ON = 1, OFF = 0 SetData(ON, "Local HMI", LB, 0, 1) SYNC_TRIG_MACRO(5) // call a macro (its ID is 5) SetData(OFF, "Local HMI", LB, 0, 1) end macro_command</pre>

函數名稱	ASYNC_TRIG_MACRO
語法	ASYNC_TRIG_MACRO(macro_id)
描述	<p>一個執行中的巨集指令可以使用此函數，利用非同步的方式觸發執行其他巨集指令，使用 <code>macro_id</code> 指定被觸發的巨集指令 【編號】。</p> <p>巨集指偷在使用此函數後將立即執行後續的命令，不需等待被觸發的巨集指令執行完成。</p> <p><code>macro_id</code> 可以是常數或使用變數來表示。</p>
舉例	<pre>macro_command main() char ON = 1, OFF = 0 SetData(ON, "Local HMI", LB, 0, 1) ASYNC_TRIG_MACRO(5) // call a macro (its ID is 5) SetData(OFF, "Local HMI", LB, 0, 1) end macro_command</pre>

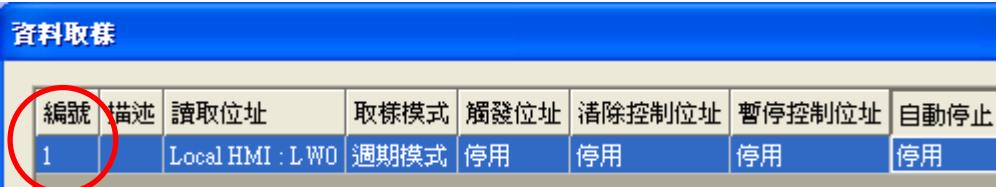
函數名稱	TRACE
語法	TRACE(format, argument)
描述	<p>一個執行中的 巨集指令可以使用此函數，監視變數內容的變化，並列印字串，以協助除錯。使用者應開啟 EasyDiagnoser 觀看此函數的輸出結果。</p> <p>當 <code>TRACE</code> 函數抓取一個%開頭的特殊字元，將同時從 <code>argument</code> 抓取一個參數做格式化後輸出。</p> <p><i>Format</i> 代表列印格式，支援 % 開頭的特殊字元。特殊字元格式如下，其中方括號內的欄位為可選，粗體字欄位為必需：</p> <p>%[flags] [width] [.precision] type</p> <p>每個欄位的意義如下所述：</p> <p><i>flags</i> (可選) :</p> <ul style="list-style-type: none"> - + <p><i>width</i> (可選) :</p> <ul style="list-style-type: none"> 十進位正整數，指定應預留的字元寬度，不足部份補空白字元。 <p><i>precision</i> (可選) :</p> <ul style="list-style-type: none"> 十進位正整數，指定精確度，以及輸出字元數。 <p><i>type</i> :</p>

	<p>C 或 c : 以字元方式輸出</p> <p>d : 以 signed 十進位整數輸出</p> <p>i : 以 signed 十進位整數輸出</p> <p>o : 以 unsigned 八進位整數輸出</p> <p>u : 以 unsigned 十進位整數輸出</p> <p>X 或 x : 以 unsigned 十六進位整數輸出</p> <p>E 或 e : 以科學表示法輸出。格式為[-] d.dddd e [sign]ddd。其中欄位 d 是十進位數字，欄位 dddd 是一至多個十進位數字，欄位 ddd 必須是三個十進位數字。sign 是+或-。</p> <p>f : 以單倍精確度浮點數輸出。格式為[-] dddd.dddd。其中欄位 dddd 是一至多個十進位數字。</p>
	<p><i>Format</i> 字串最長支援 256 個字元，多出的字元將被忽略。</p> <p><i>Argument</i> 部份可寫可不寫。但一個特殊字元應搭配一個變數。</p>
舉例	<pre>macro_command main() char c1 = 'a' short s1 = 32767 float f1 = 1.234567 TRACE("The results are") // 輸出：The results are TRACE("c1 = %c, s1 = %d, f1 = %f", c1, s1, f1) // 輸出：c1 = a, s1 = 32767, f1 = 1.234567 end macro_command</pre>

函數名稱	FindDataSamplingDate
語法	<pre>return_value = FindDataSamplingDate (data_log_number, index, year, month, day)</pre> <p style="text-align: center;">or</p> <pre>FindDataSamplingDate (data_log_number, index, year, month, day)</pre>
描述	<p>利用此函數查詢資料取樣檔的日期。根據所輸入的資料取樣編號 (data_log_number) 與資料取樣檔案索引 (index) 可查詢該資料取樣的日期，依照年、月、日的順序寫入 year、month、day 的欄位中。</p>  <p>資料取樣編號</p> <p>資料取樣檔的儲存方式為：[保存位置] \ [取樣資料夾名稱] \ yyyyymmdd.dtl。而資料取樣檔案索引 (index) 指的是取樣資料夾下面所有資料取樣檔案依檔名排序後的順位值 (從 0 開始)。Index 值越小者，日期越新。例如假設某取樣資料夾下面有四個資料取樣檔：</p> <p>20101210.dtl 20101230.dtl 20110110.dtl 20110111.dtl</p> <p>則 index 依序為：</p> <p>20101210.dtl -> index 為 3 20101230.dtl -> index 為 2 20110110.dtl -> index 為 1 20110111.dtl -> index 為 0</p> <p>當成功搜尋到所欲查詢的資料取樣檔時，將回傳 1 至 return_value，否則將回傳 0。</p> <p>data_log_number 與 index 可以為常數或是變數，但 year、month、day 與 return_value 必須為變數。</p> <p>return_value 為可選。</p>
舉例	<pre>macro_command main() short data_log_number = 1, index = 2, year, month, day short success // 若存在一資料取樣檔 20101230.dtl，其資料取樣編號為 1，檔案索引為 2</pre>

```
// 則success == 1, year == 2010, month == 12, day == 30
success = FindDataSamplingDate(data_log_number, index, year,
month, day)

end macro_command
```

函數名稱	FindDataSamplingIndex																
語法	<pre>return_value = FindDataSamplingIndex (data_log_number, year, month, day, index)</pre> <p>or</p> <pre>FindDataSamplingIndex (data_log_number, year, month, day, index)</pre>																
描述	<p>利用此函數查詢資料取樣檔的檔案索引值。根據所輸入的資料取樣編號 (data_log_number) 與日期可查詢該資料取樣檔的檔案索引，並將其寫入 index。year、month、day 三個欄位依序代表年、月、日，其輸入格式為 YYYY (年)、MM (月)、DD (日)。</p> <p>資料取樣</p>  <table border="1"> <thead> <tr> <th>編號</th> <th>描述</th> <th>讀取位址</th> <th>取樣模式</th> <th>觸發位址</th> <th>清除控制位址</th> <th>暫停控制位址</th> <th>自動停止</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Local HMI : LWO</td> <td>週期模式</td> <td>停用</td> <td>停用</td> <td>停用</td> <td>停用</td> <td>停用</td> </tr> </tbody> </table> <p>資料取樣編號</p> <p>資料取樣檔的儲存方式為：[保存位置] \ [取樣資料夾名稱] \ yyyyymmdd.dtl。而資料取樣檔案索引 (index) 指的是取樣資料夾下面所有資料取樣檔案依檔名排序後的順位值 (從 0 開始)。Index 值越小者，日期越新。例如假設某取樣資料夾下面有四個資料取樣檔：</p> <p>20101210.dtl 20101230.dtl 20110110.dtl 20110111.dtl</p> <p>則 index 依序為：</p> <p>20101210.dtl -> index 為 3 20101230.dtl -> index 為 2 20110110.dtl -> index 為 1 20110111.dtl -> index 為 0</p> <p>當成功搜尋到所欲查詢的資料取樣檔時，將回傳 1 至 return_value，否則將回傳 0。</p> <p>data_log_number 與 year、month、day 可以為常數或是變數，但 index 與 return_value 必須為變數。</p> <p>return_value 為可選。</p>	編號	描述	讀取位址	取樣模式	觸發位址	清除控制位址	暫停控制位址	自動停止	1	Local HMI : LWO	週期模式	停用	停用	停用	停用	停用
編號	描述	讀取位址	取樣模式	觸發位址	清除控制位址	暫停控制位址	自動停止										
1	Local HMI : LWO	週期模式	停用	停用	停用	停用	停用										

舉例	<pre> macro_command main() short data_log_number = 1, year = 2010, month = 12, day = 10, index short success // 若存在一資料取樣檔 20101210.dtl，其資料取樣編號為 1，檔案索引為 2 // 則success == 1, index == 2 success = FindDataSamplingIndex (data_log_number, year, month, day, index) end macro_command </pre>
----	--

函數名稱	FindEventLogDate								
語法	<pre> return_value = FindEventLogDate (index, year, month, day) or FindEventLogDate (index, year, month, day) </pre>								
描述	<p>利用此函數查詢事件登錄檔的日期。根據所輸入的事件登錄檔案索引 (index) 可查詢該事件登錄的日期，依照年、月、日的順序寫入 year、month、day 的欄位中。</p> <p>事件登錄檔案索引 (index) 指的是指定的保存位置 (HMI、SD 卡或 USB) 下事件登錄檔案依檔名排序後的順位值 (從 0 開始)。Index 值越小者，日期越新。例如假設有四個事件登錄檔：</p> <p>EL_20101210.evt EL_20101230.evt EL_20110110.evt EL_20110111.evt</p> <p>則 index 依序為：</p> <table> <tbody> <tr> <td>EL_20101210.evt</td> <td>-> index 為 3</td> </tr> <tr> <td>EL_20101230.evt</td> <td>-> index 為 2</td> </tr> <tr> <td>EL_20110110.evt</td> <td>-> index 為 1</td> </tr> <tr> <td>EL_20110111.dtl</td> <td>-> index 為 0</td> </tr> </tbody> </table> <p>當成功搜尋到所欲查詢的事件登錄檔時，將回傳 1 至 return_value，否則將回傳 0。</p> <p>Index 可以為常數或是變數，但 year、month、day 與 return_value 必須為變數。</p> <p>return_value 為可選。</p>	EL_20101210.evt	-> index 為 3	EL_20101230.evt	-> index 為 2	EL_20110110.evt	-> index 為 1	EL_20110111.dtl	-> index 為 0
EL_20101210.evt	-> index 為 3								
EL_20101230.evt	-> index 為 2								
EL_20110110.evt	-> index 為 1								
EL_20110111.dtl	-> index 為 0								

舉例	<pre> macro_command main() short index = 1, year, month, day short success // 若存在一事件登錄檔 EL_20101230.evt，檔案索引為 1 // 則 success == 1, year == 2010, month == 12, day == 30 success = FindEventLogDate (index, year, month, day) end macro_command </pre>
----	---

函數名稱	FindEventLogIndex
語法	<pre> return_value = FindEventLogIndex (year, month, day, index) or FindEventLogIndex (year, month, day, index) </pre>
描述	<p>利用此函數查詢事件登錄檔的檔案索引值。根據所輸入的日期可查詢該事件登錄檔的檔案索引，並將其寫入 <code>index</code>。<code>year</code>、<code>month</code>、<code>day</code> 三個欄位依序代表年、月、日，其輸入格式為 YYYY (年)、MM (月)、DD (日)。</p> <p>事件登錄檔案索引 (<code>index</code>) 指的是指定的保存位置 (HMI、SD 卡或 USB) 下事件登錄檔案依檔名排序後的順位值 (從 0 開始)。<code>Index</code> 值越小者，日期越新。例如假設有四個事件登錄檔：</p> <p>EL_20101210.evt EL_20101230.evt EL_20110110.evt EL_20110111.evt</p> <p>則 <code>index</code> 依序為：</p> <p>EL_20101210.evt → <code>index</code> 為 3 EL_20101230.evt → <code>index</code> 為 2 EL_20110110.evt → <code>index</code> 為 1 EL_20110111.evt → <code>index</code> 為 0</p> <p>當成功搜尋到所欲查詢的事件登錄檔時，將回傳 1 至 <code>return_value</code>，否則將回傳 0。</p> <p><code>year</code>、<code>month</code>、<code>day</code> 可以為常數或是變數，但 <code>index</code> 與 <code>return_value</code> 必須為變數。</p> <p><code>return_value</code> 為可選。</p>
舉例	<pre> macro_command main() short year = 2010, month = 12, day = 10, index short success </pre>

```
// 若存在一事件登錄檔 EL_20101210.evt，檔案索引為 2  
// 則success == 1, index == 2  
success = FindEventLogIndex (year, month, day, index)  
  
end macro_command
```

18.7 怎樣建立和執行巨集指令

18.7.1 怎樣建立一個巨集指令

按照以下步驟以建立一個巨集指令。

步驟 1：

單擊 EasyBuilder 軟體工具欄上的巨集指令圖示  打開巨集指令管理對話方塊。如下圖所示。



在巨集指令管理對話方塊中，已經編譯成功的巨集指令會出現在 **[已編譯成功]** 列表上，未完成編譯的會出現在 **[未完成編譯]** 列表中。下面是巨集指令管理對話方塊中各按鍵的功能描述。

[新增]

新增一個巨集指令，並開啟新建的巨集指令編輯區。

[刪除]

刪除選擇的巨集指令

[編輯]

打開巨集指令編輯區，並開啟選擇的巨集指令。

[複製]

複製選擇的巨集指令。

[貼上]

將剛剛選擇需要複製的巨集指令，貼至“已編譯完成”列表區，並產生一個新的巨集指令名稱。

[確認]

確認此次所編輯的所有巨集指令後離開此巨集指令管理對話方塊，必須按此鍵才會儲存此次編輯內容。

[取消]

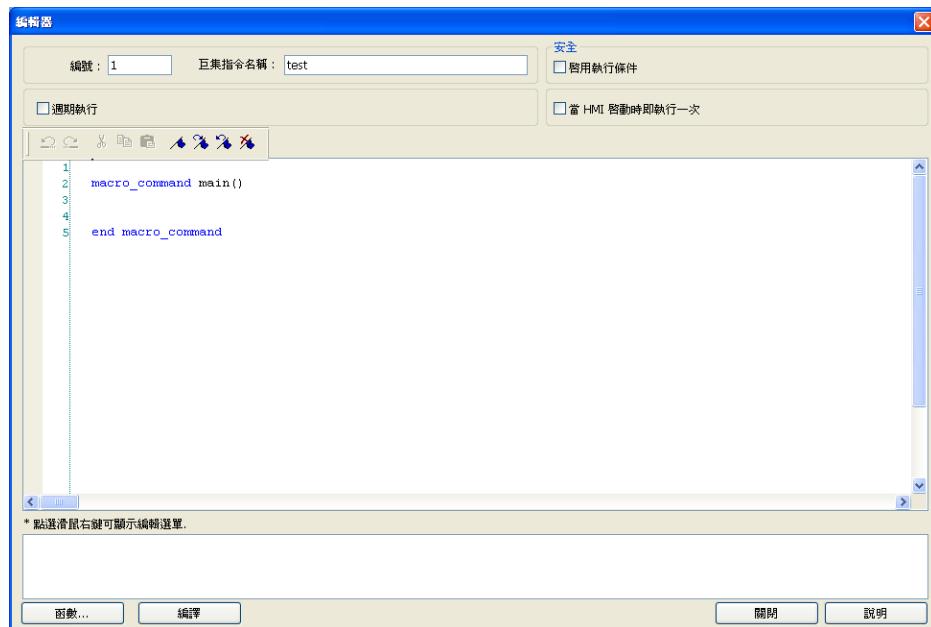
取消此次所編輯的所有巨集指令，離開巨集指令管理對話方塊。

[巨集指令庫]

進入巨集函數庫管理對話方塊。

步驟 2：

按下 **[新增]** 按鈕，打開一個新增的巨集指令編輯區。每一個巨集指令都有一個唯一的編號，定義在 **[編號]** 這個位置。在 **[巨集指令名稱]** 這個欄目中也必須輸入巨集指令的名稱，否則編譯將無法通過。



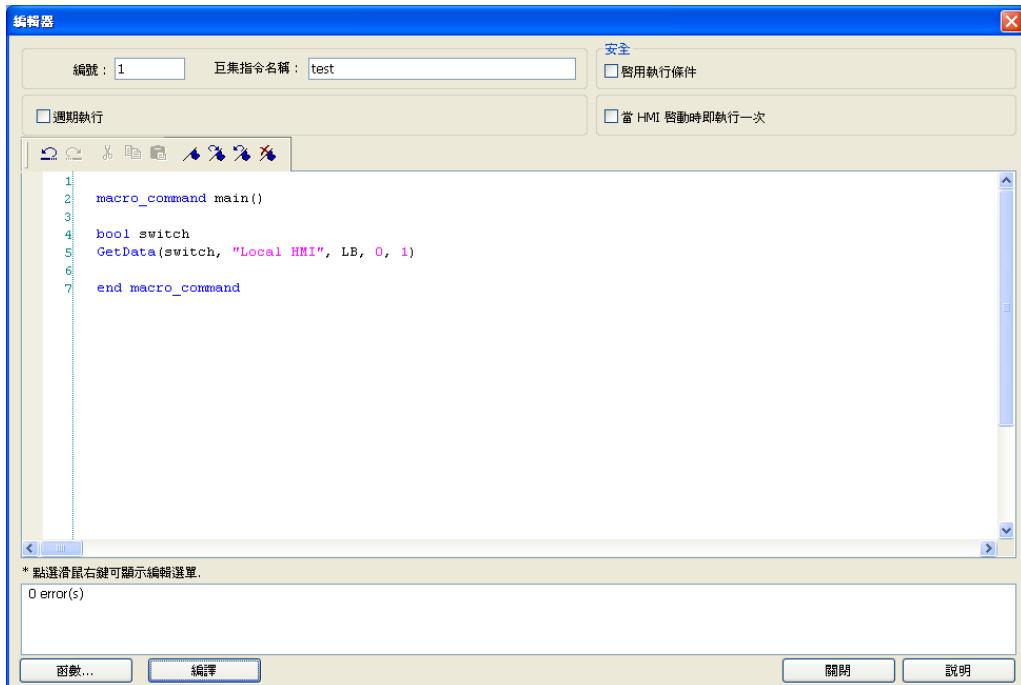
步驟 3：

設計屬於您的巨集指令程式。如果要使用內建的函數，譬如 Setdata() 或者 Getdata 等函數，單擊 **【函數】** 按鍵開啟函數列表對話方塊，選擇需要的函數，並設定必要的參數。



步驟 4：

編輯完成一個新建的巨集指令程式後，單擊 **【編譯】** 按鍵，對該巨集指令進行編譯工作。



如果沒有錯誤，單擊 **【關閉】** 按鍵，這樣在 **【已編譯成功】** 區會發現新增了一個 “**test**” 這個名稱的巨集指令。



18.7.2 執行巨集指令

執行巨集指令有多種不同的方法，下面分別說明。

a. 使用 **[PLC 控制]** 物件

1. 打開 **[PLC 控制]** 物件，並設定屬性為 **[執行巨集指令]**；
2. 選擇需要執行的巨集指令名稱。選擇一個位元作為觸發巨集指令並設定觸發巨集指令的條件。在條件滿足時，該巨集指令將會被重複執行。為了每次只讓巨集指令執行一次，設計時需在巨集指令將該觸發位元復位。
3. 使用一個 **[位元狀態設定]** 物件或者 **[位元狀態切換開關]** 物件作為這個位元的控制開關。

b. 使用 **[位元狀態設定]** 物件或者 **[位元狀態切換開關]** 物件

1. 在 **[位元狀態設定]** 物件或者 **[位元狀態切換開關]** 物件的一般屬性頁中，勾選 **[使用巨集指令]**；
2. 選擇要執行的巨集指令。當這個物件被執行時，選擇的巨集指令就會被執行一次。

c. 使用 **[功能鍵]**

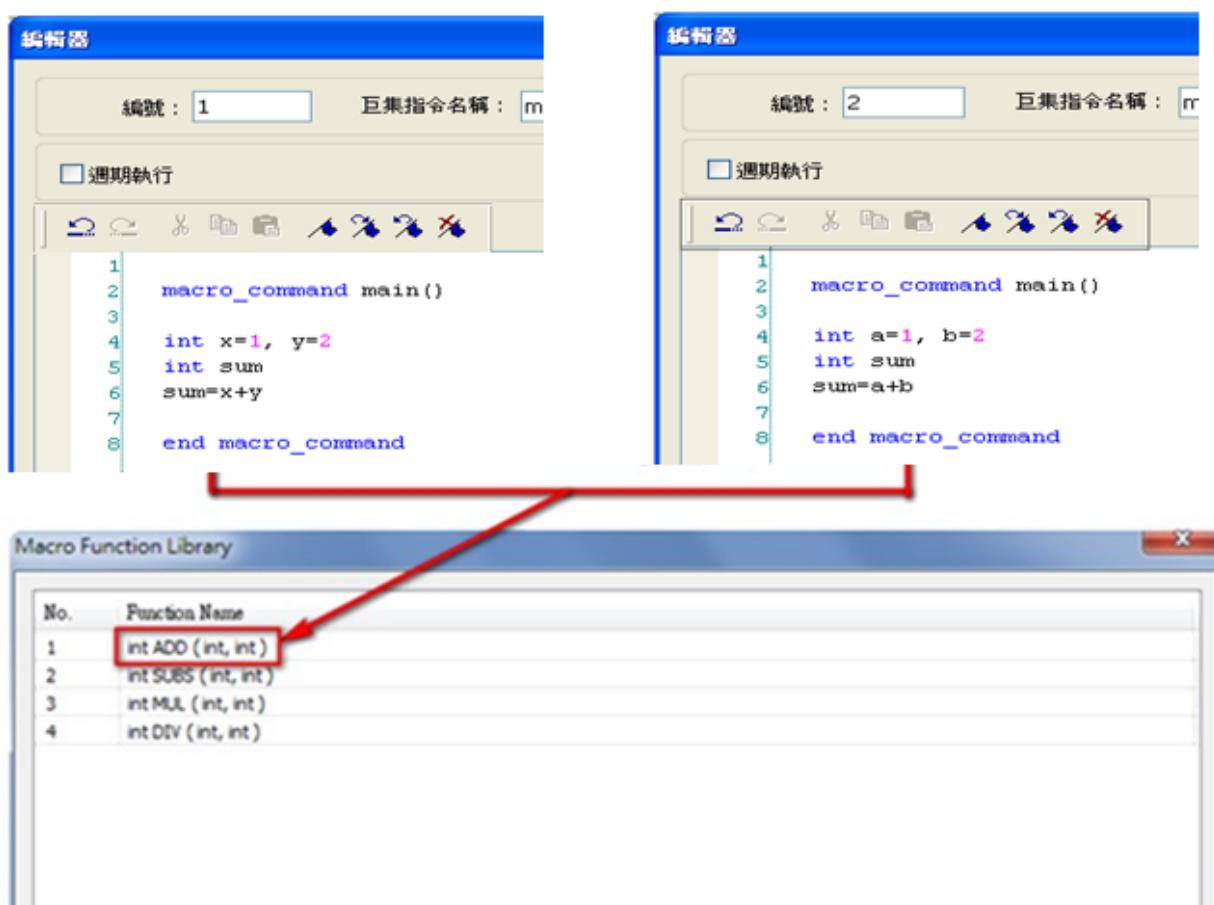
1. 在 **[功能鍵]** 的一般屬性頁對話方塊中，勾選 **[觸發巨集指令]**；
2. 選擇需要執行的巨集指令。每按一下這個功能鍵時，選擇的巨集指令就會被執行一次。

d. 使用 **[巨集指令編輯器]** 設定條件

1. **[週期執行]**：可以設定每間隔幾秒自動執行巨集指令一次。
2. **[當 HMI 啟動時即執行一次]**：當 HMI 重新上電或重新啟動時會執行此巨集指令一次。

18.8 使用者自定義函數功能

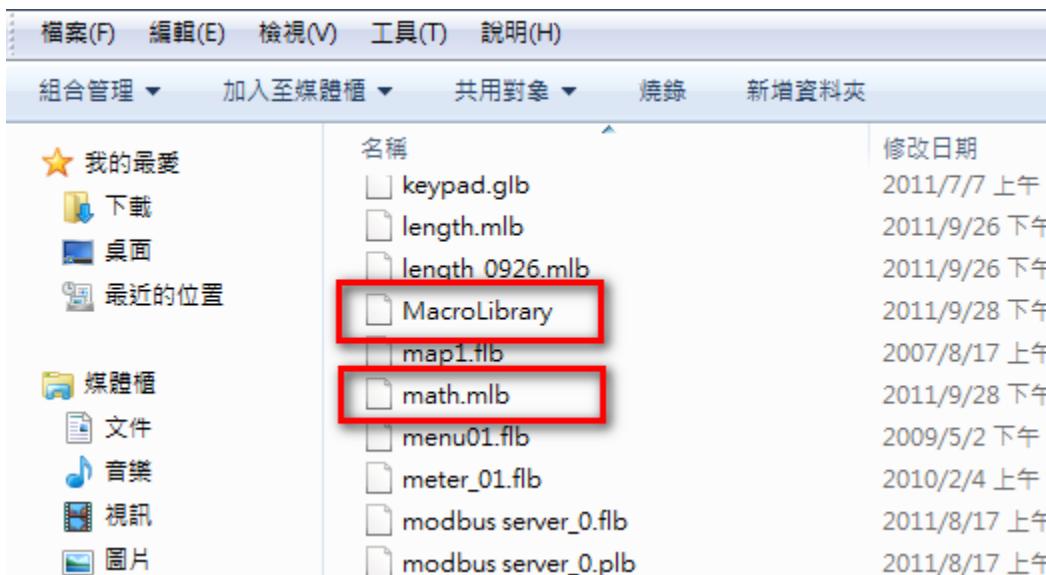
在使用巨集編輯器時，為了減少定義函數的時間，使用者可從內建的函數庫搜尋所需的函數。然而，當使用者編輯巨集時，若有某些特定的函數常常使用卻無法從內建函數庫搜尋到時，就可以自行定義所需的函數並儲存起來。當下次需要再定義相同的函數時，可由【巨集指令庫】呼叫出已儲存的函數，方便函數編輯。另外，【巨集指令庫】也大幅提升了使用者自訂函數之可攜性。建立函數前可先查看現有內建函數或線上函數庫是否有現成函數可使用。



18.8.1 匯入函數庫檔案

HMI 編輯軟體開啟一個工程檔時，會自動去讀入一個預設函數庫檔案，並載入函數資訊。當開啟一專案有呼叫到使用者定義函數時必須先載入相關的 mlb 檔案。

1. 預設函數庫檔名：MacroLibrary (沒有副檔名)
2. 函數庫路徑：HMI 編輯軟體的安裝目錄的 \library 資料夾下面。
3. \library 資料夾下面可以看到兩種函數庫檔案：
 - 沒有副檔名： MacroLibrary，為預設函數庫， HMI 編輯軟體指定讀入此檔案。
 - 有副檔名 (*.mlb)：例如 math.mlb，使用者匯入/匯出時會去讀 / 寫的檔案，具可攜性，欲使用時再從資料夾呼叫出檔案即可。
4. EasyBuilder 開啟時，只會去載入預設函數庫中的函數，使用者若需要用到 *.mlb 檔內的函數時，必須自行將其匯入。

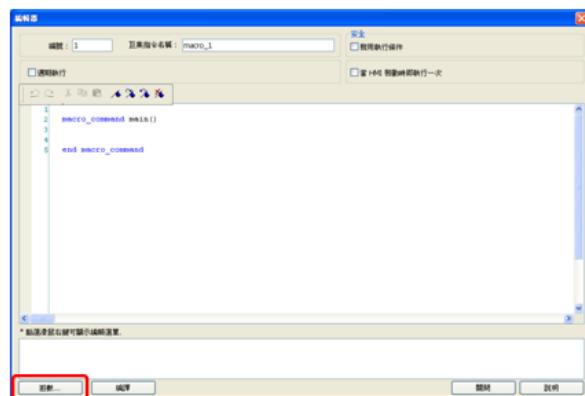


18.8.2 如何使用巨集函數庫

1. 在巨集編輯器中直接呼叫函數。



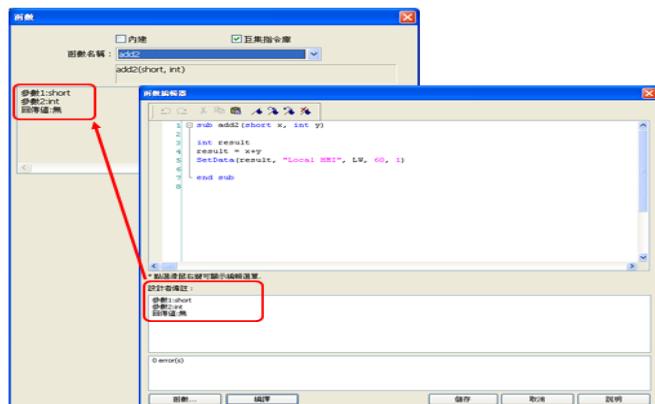
2. 按下巨集編輯器左下角的【函數】按鈕開啟函數對話框。



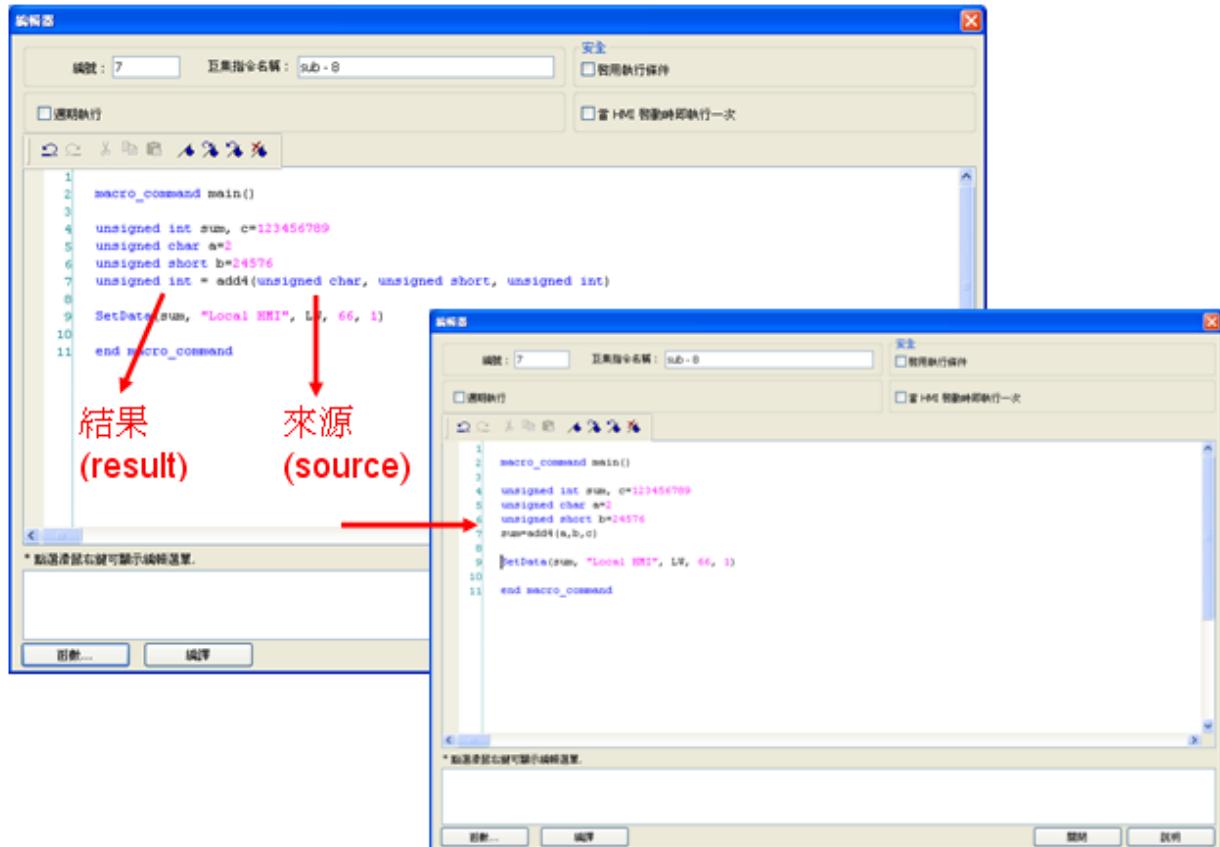
3. 須至少勾選一項【巨集指令庫】或【內建】，並選擇欲使用的函數。



4. 顯示函數說明文字，即是使用者在函數編輯器中編輯的說明文字。



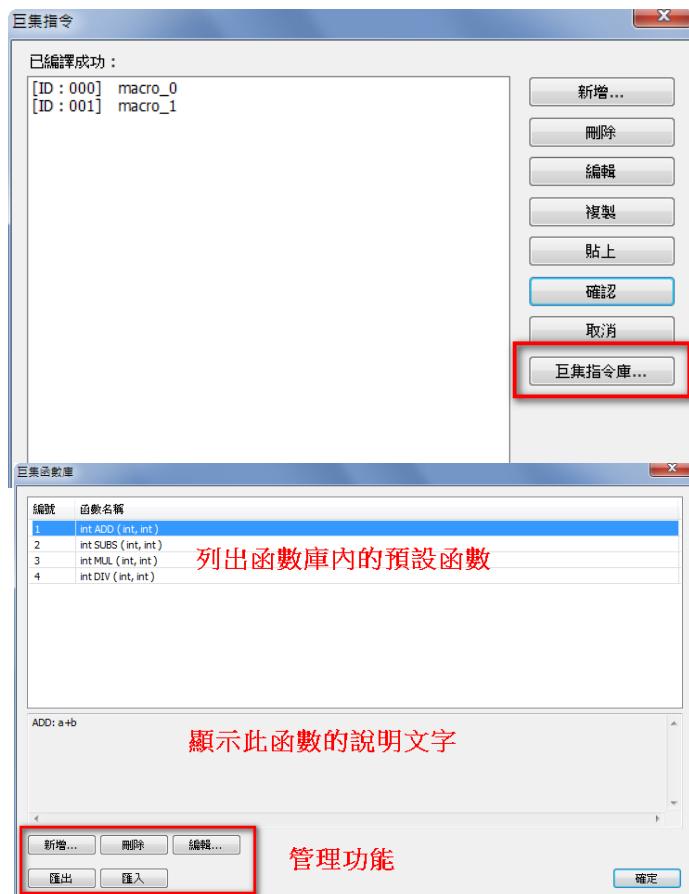
5. 選取欲使用的函數，將函數庫中的【變數類型】修改為指定的變數名稱。



使用者根據以上的步驟，即可完成使用自訂函數的功能，有效節省了重複定義相同的函數。

18.8.3 函數庫管理介面

- 開啟巨集管理對話框，按下右下角【巨集指令庫】按鈕，進入函數庫管理對話框介面。



- 函數列表中每一行的格式如下：

```
return_type function_name (parameter_type1, ..., parameter_typeN)
```

return_type 表示回傳值型態，若無回傳值則此欄位省略；function_name 表示函數名稱。
parameter_typeN 表示第 N 個參數型態，若此函數不接受任何參數，此欄位省略。

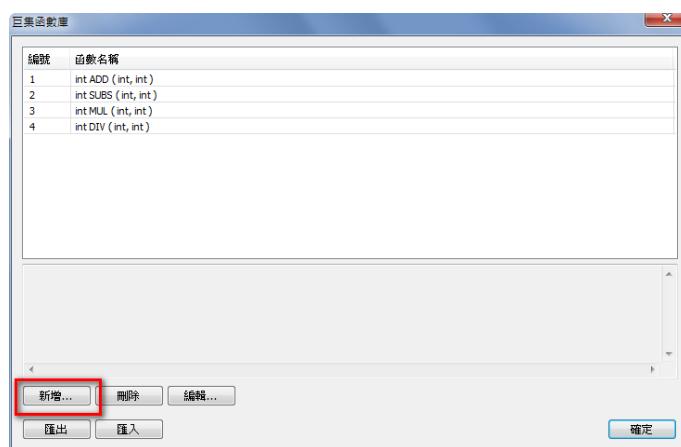
```

1  sub int ADD(int a, int b)
2    int ret
3    ret = a+b
4    return ret
5  end sub
6

```

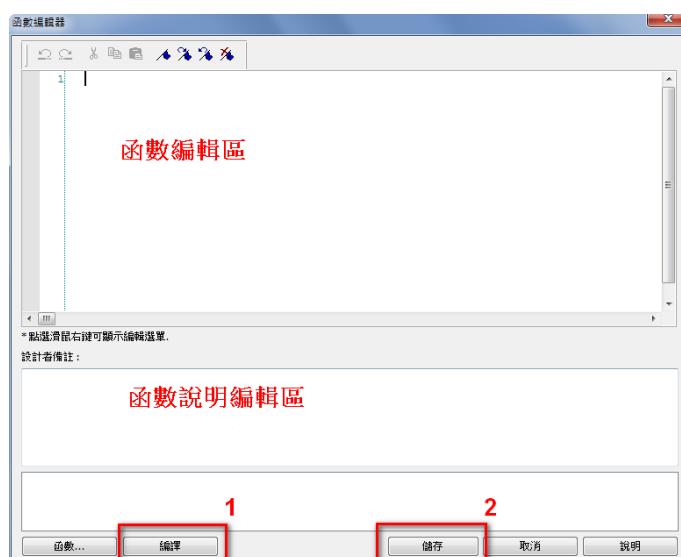
18.8.3.1 新建函數

1. 按下【新增】進入函數編輯器。

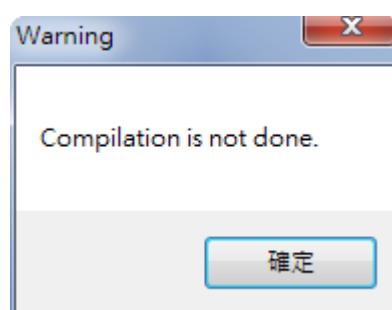


2. 在函數編輯區編輯函數。

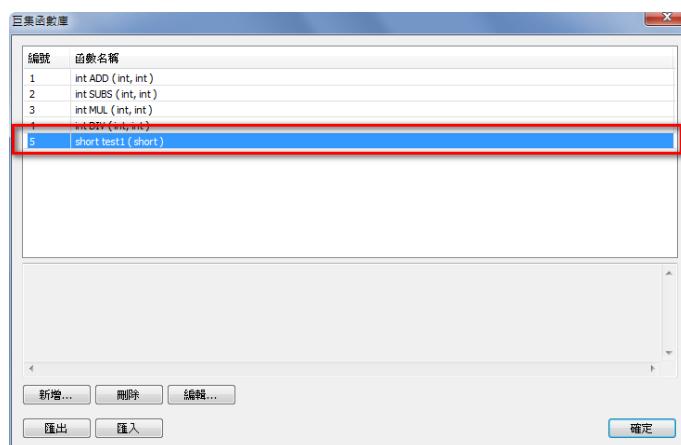
3. 使用者可以在函數說明編輯區中編輯說明文字，說明此函數的規格、使用方法、作者聲明等等。



4. 函數編輯完成後，必須通過編譯，才可以按下【儲存】寫入函數庫。否則會出現右圖的彈出視窗，警告使用者此函數未完成編譯。



5. 成功加入函數庫。

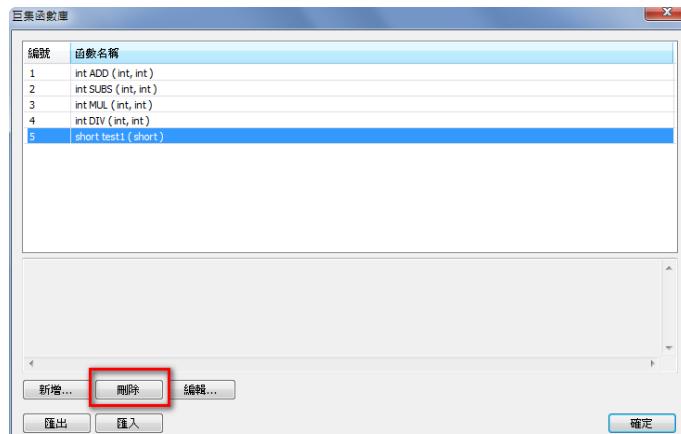




1. 函數中可宣告的資料型態總數為 4096 個位元組。
2. 函數名稱必須為英數字元，且不可為數字開頭。

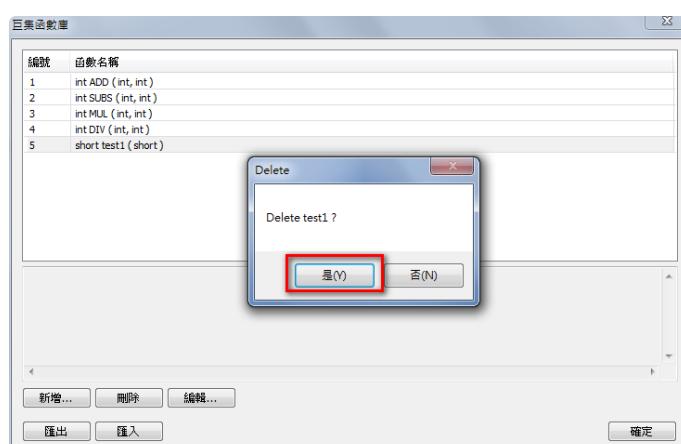
18.8.3.2 刪除函數

- 在函數列表中選定要刪除的函數，按下【刪除】按鈕，即可刪除該函數。



- 按下【是】確認刪除，按下【否】取消刪除。

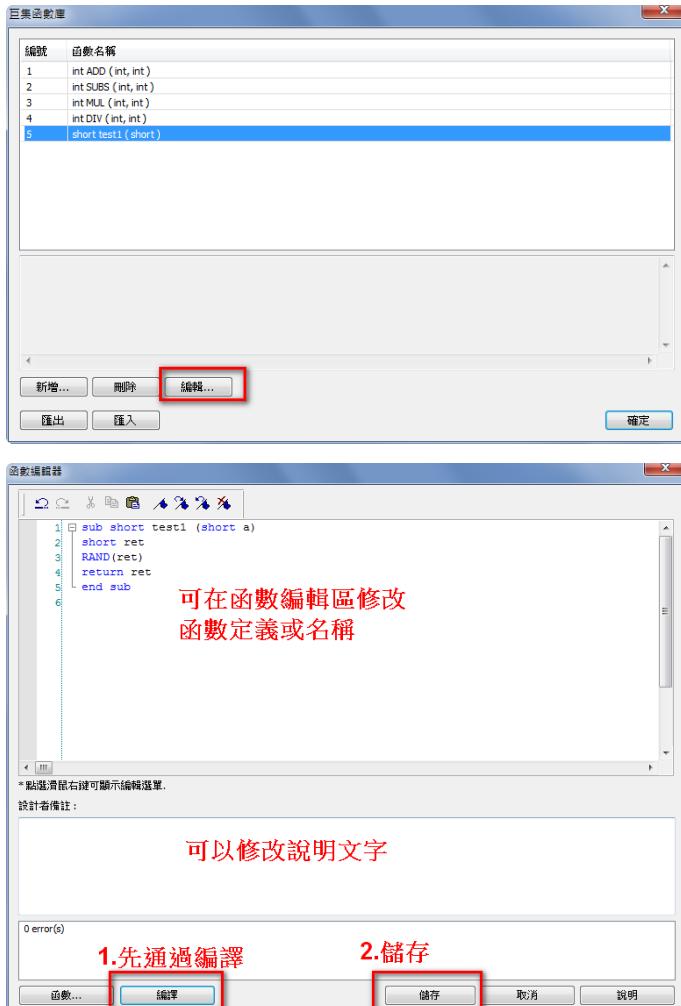
- 按下【是】刪除 MAX_SHORT 此函數。



18.8.3.3 修改函數

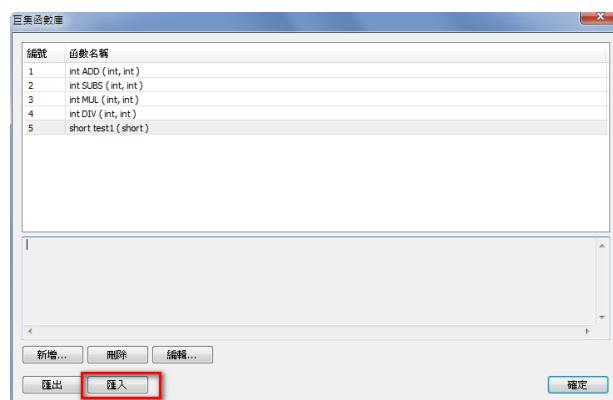
- 使用者可以修改函數庫中的函數。
- 選定要修改的函數，按下【編輯】按鈕，進入函數編輯器。

- 也可直接在該函數上雙點滑鼠左鍵，進入函數編輯器。
- 修改完成後，一樣要先通過編譯，才可以儲存離開。



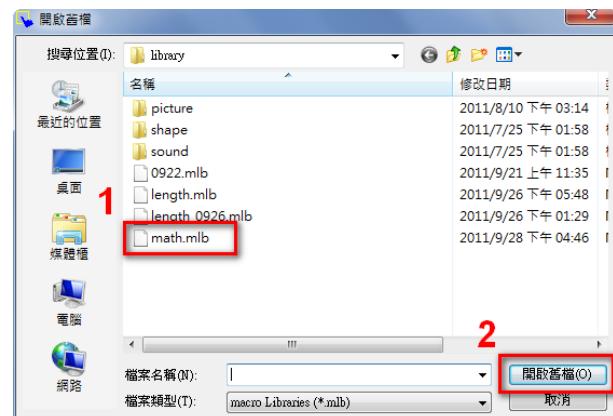
18.8.3.4 匯入函數

- 使用者可以從外部的 *.mlb 檔案將函數匯入。



- 假設欲加入函數庫 math.mlb，此函數庫內含有一函數 test1。

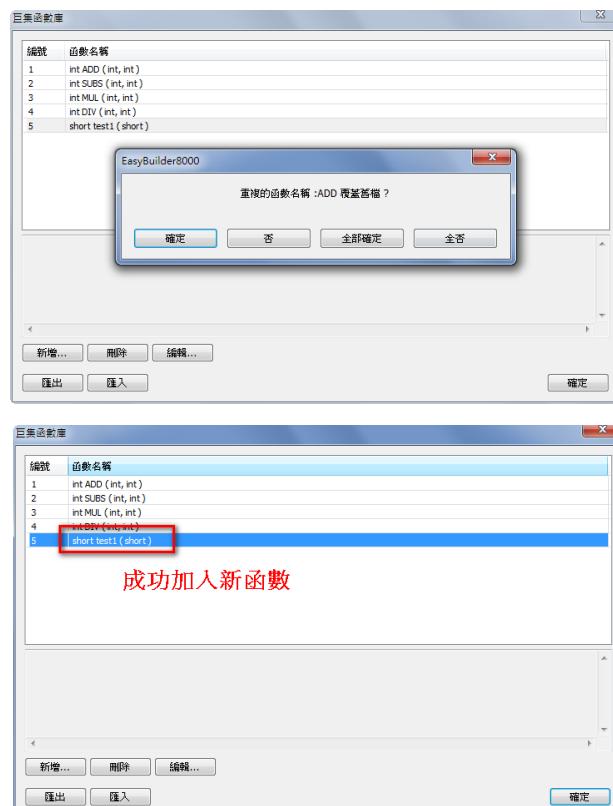
- 按下[開啟舊檔]按鈕。



- 匯入函數時，若檔案庫中已存在相同名稱的函數，會出現下面的彈出視窗。

- [確定]**：用外部匯入的函數覆蓋函數庫中現有的同名函數。
- [否]**：放棄外部匯入此同名函數。
- [全部確定]**：全部用外部匯入的同名函數覆蓋所有同名函數。
- [全否]**：全部放棄覆蓋匯入同名的函數。

- 完成匯入後，匯入的函數都已寫入到預設函數庫中，因此使用者可以將 math.mlb 檔案刪除，而 test1 仍會存在函數庫中，即使重新開啟 HMI 編輯軟體亦然。



18.8.3.5 匯出函數

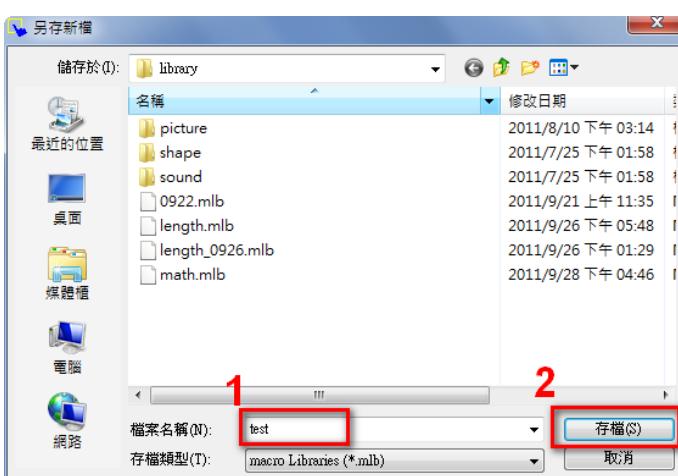
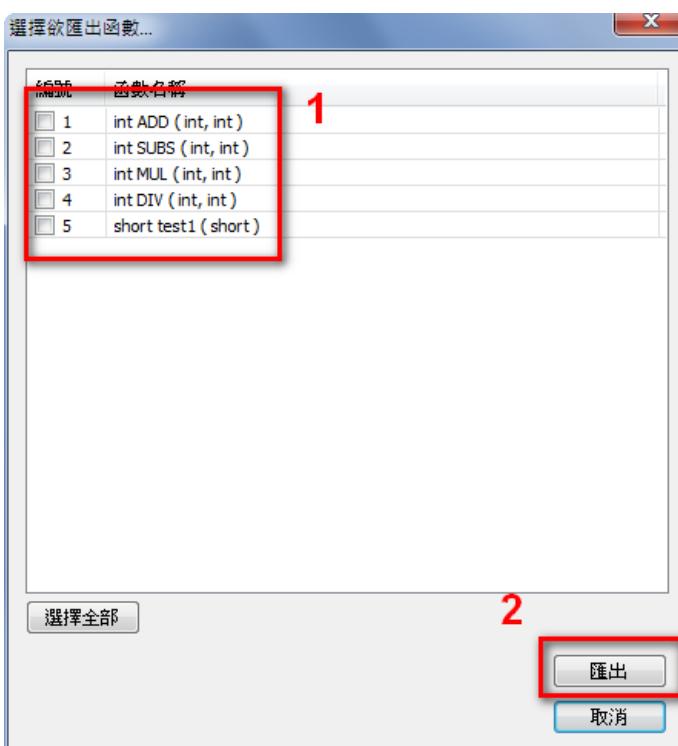
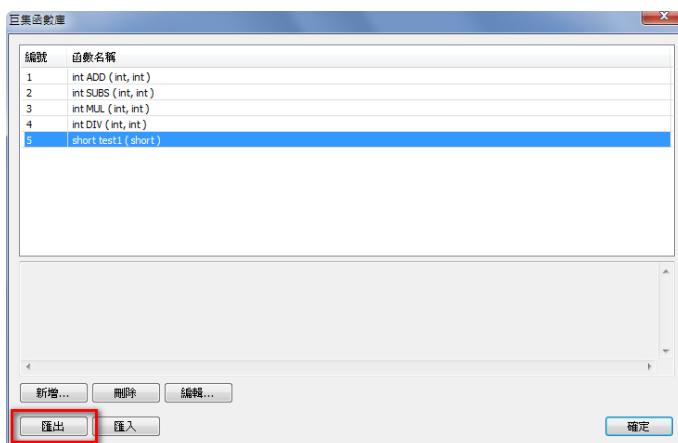
1. 使用者可以將函數庫中的函數匯出到 *.mlb 檔。

2. 按下【匯出】按鈕。

3. 選擇要匯出的函數，然後按【匯出】。

4. 匯出的目錄下出現一個 math.mlb 的函數庫檔案，其中包含 ADD、SUBS、MUL、DIV 這四支函數。

5. 匯出的 *.mlb 檔可以攜帶到別的電腦上，只要使用者開啟 HMI 編輯軟體並完成匯入動作後，即可使用此檔案內所提供的所有函數。



18.9 使用巨集指令時的注意事項

1. 儲存局部變數的空間是 4KB，所以各種不同變數類型的最大陣列大小為如下：

char	a[4096]
bool	b[4096]
short	c[2048]
int	d[1024]
float	e[1024]

2. 一個 EasyBuilder 工程中最多包含 255 個巨集指令。
3. 巨集指令有可能造成 HMI 當機，可能的原因為：
 - 巨集指令中執行了一個閉環命令
 - 陣列的大小超過了巨集指令的變數容量
4. PLC 的通訊速度可能影響巨集指令的執行速度。相對的，使用過多的巨集指令，可能會造成與 PLC 的通訊速度變慢。

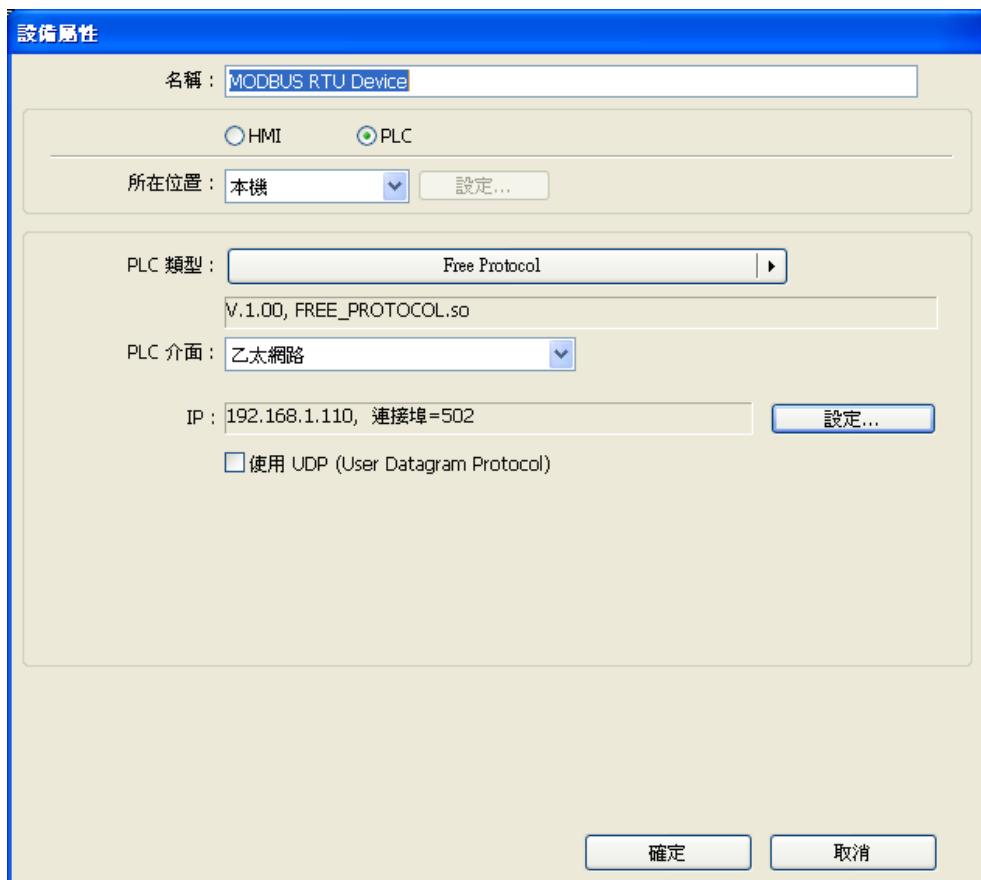
18.10 使用自由協定去控制一個設備

當 EasyBuilder 沒有內建與某一個設備通訊的驅動程式時，使用者也可以使用巨集指令中的 **OUTPORT** 和 **IMPORT** 函數來實現與該設備的通訊。使用 **OUTPORT** 和 **IMPORT** 函數發送和接收的資料，必須遵行該設備的通訊協定。下面的範例程式說明了如何使用這兩個函數來控制一個 MODBUS RTU 設備。

首先，在系統參數/設備列表中建立一個新的設備。這個新建的 **[PLC 類型]** 設置為 “**Free Protocol**”，**[PLC 名稱]** 設置為 “**MODBUS RTU Device**”，如下圖所示。



這裡的設備連接使用 RS232，如果連接一個 MODBUS TCP / IP 設備，這個介面類型必須設定為“乙太網路”，同時必須設定正確的 IP 位址和連接埠號，如下圖所示。



假設 HMI 人機界面將要讀取設備中的 4x_1 和 4x_2 兩個資料暫存器。首先，使用 **OUTPORT** 函數發送讀命令給這個設備，**OUTPORT** 函數的寫法為：

`OUTPORT(command[start], device_函數名稱, cmd_count)`

因為 “**MODBUS RTU Device**” 是一個 MODBUS RTU 設備，讀數據的命令必須遵行 MODBUS RTU 協定的命令規則。所以必須使用 0x03 這個命令去讀取 4x_1 和 4x_2 這兩個資料暫存器的資料。下圖說明瞭讀命令的格式。(省略了設備的站號和最後的兩個 CRC 位元組)

Request

Function code	1 Byte	0x03
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 125 (0x7D)

Response

Function code	1 Byte	0x03
Byte count	1 Byte	2 x N*
Register value	N* x 2 Bytes	

*N = Quantity of Registers

Error

Error code	1 Byte	0x83
Exception code	1 Byte	01 or 02 or 03 or 04

根據這個 MODBUS RTU 協定，發送命令的內容如下所示：（總共 8 個位元組）

command[0]: station number	(BYTE 0) 站號
command[1]: function code	(BYTE 1) 功能碼
command[2]: high byte of starting address	(BYTE 2) 起始位址高位元
command[3]: low byte of starting address	(BYTE 3) 起始位址低位元
command[4]: high byte of quantity of registers	(BYTE 4) 資料暫存器的高位元
command[5]: low byte of quantity of registers	(BYTE 5) 資料暫存器的低位元
command[6]: low byte of 16-bit CRC	(BYTE 6) CRC的低位元組
command[7]: high byte of 16-bit CRC	(BYTE 7) CRC的高位元組

所以讀數據的命令巨集指令程式設計如下：

```
char command[32]
short address, checksum

FILL(command[0], 0, 32) // initialize command[0]~command[31] to 0

command[0] = 0x1      // station number
command[1] = 0x3      // read holding registers (function code is 0x3)

address = 0           // starting address (4x_1) is 0
HIBYTE(address, command[2])
LOBYTE(address, command[3])

read_no = 2           // the total words of reading is 2 words
HIBYTE(read_no, command[4])
LOBYTE(read_no, command[5])

CRC(command[0], checksum, 6) // calculate 16-bit CRC

LOBYTE(checksum, command[6])
HIBYTE(checksum, command[7])
```

最後，使用 **OUTPORT** 函數將這個讀命令發送給這個 MODBUS RTU 設備。

```
OUTPORT(command[0], "MODBUS RTU Device", 8) // 發送讀指令
```

發送完這個命令後，使用 **IMPORT** 函數讀取該 MODBUS RTU 設備返回的命令。根據 MODBUS RTU 協定，這個回復的命令內容為如下 (總共 9 個位元組)：

command[0]: station number	(BYTE 0) 站號
command[1]: function code	(BYTE 1) 功能碼
command[2]: byte count	(BYTE 2) 位元組長度
command[3]: high byte of 4x_1	(BYTE 3) 第一個資料的高位元組
command[4]: low byte of 4x_1	(BYTE 4) 第一個資料的低位元組
command[5]: high byte of 4x_2	(BYTE 5) 第二個資料的高位元組
command[6]: high byte of 4x_2	(BYTE 6) 第二個資料的低位元組
command[7]: low byte of 16-bit CRC	(BYTE 7) CRC的低位元組
command[8]: high byte of 16-bit CRC	(BYTE 8) CRC的高位元組

此時，**IMPORT** 函數的語句如下：

```
IMPORT(response[0], "MODBUS RTU Device", 9, return_value) // 讀取回復的命令
```

函數中，實際讀取到的位元組長度存放在變數 `return_value` (變數類型為位元組) 中。如果 `return_value` 的資料為 0，則表示使用 **IMPORT** 讀取命令失敗。

根據 MODBUS RTU 協定，如果命令人回復的正確，則 `response[1]` 必須為 0x03。當讀取到正確的命令後，計算出 `4x_1` 和 `4x_2` 這兩個暫存器的值，並將這兩個資料送到人機界面的 `LW100` 和 `LW101` 暫存器中。

```
if (return_value >0 and response[1] == 0x3) then
    read_data[0] = response[4] + (response[3] << 8) // 計算4x_1 的資料
    read_data[1] = response[6] + (response[5] << 8) // 計算4x_2 的資料

    SetData(read_data[0], "Local HMI", LW, 100, 2) // 將資料存至HMI上
end if
```

完整的巨集指令程式如下：

```
// Read Holding Registers
macro_command main()

char command[32], response[32]
```

```
short address, checksum
short read_no, return_value, read_data[2], i

FILL(command[0], 0, 32)// initialize command[0]~command[31] to 0
FILL(response[0], 0, 32)

command[0] = 0x1// station number
command[1] = 0x3// read holding registers (function code is 0x3)

address = 0
address = 0// starting address (4x_1) is 0
HIBYTE(address, command[2])
LOBYTE(address, command[3])

read_no = 2// the total words of reading is 2 words
HIBYTE(read_no, command[4])
LOBYTE(read_no, command[5])

CRC(command[0], checksum, 6)// calculate 16-bit CRC

LOBYTE(checksum, command[6])
HIBYTE(checksum, command[7])

OUTPORT(command[0], "MODBUS RTU Device", 8 )// send request
INPORT(response[0], "MODBUS RTU Device", 9, return_value)// read response

if (return_value > 0 and response[1] == 0x3) then
    read_data[0] = response[4] + (response[3] << 8)// 4x_1
    read_data[1] = response[6] + (response[5] << 8)// 4x_2

    SetData(read_data[0], "Local HMI", LW, 100, 2)
end if

end macro_command
```

下面的舉例說明如何使用自由協定設定 MODBUS RTU 設備中 0x_1 的狀態。這個是使用 MODBUS RTU 協議中的“寫單個暫存器”的功能碼“0x05”來實現的。

Request

Function code	1 Byte	0x05
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

Response

Function code	1 Byte	0x05
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

Error

Error code	1 Byte	0x85
Exception code	1 Byte	01 or 02 or 03 or 04

完整的巨集指令程式如下：

```
// Write Single Coil (ON)
macro_command main()

char command[32], response[32]
short address, checksum
short i, return_value

FILL(command[0], 0, 32)// initialize command[0]~ command[31] to 0
FILL(response[0], 0, 32)

command[0] = 0x1// station number
command[1] = 0x5// function code : write single coil

address = 0
HIBYTE(address, command[2])
LOBYTE(address, command[3])

command[4] = 0xff// force 0x_1 on
command[5] = 0

CRC(command[0], checksum, 6)

LOBYTE(checksum, command[6])
HIBYTE(checksum, command[7])
```

```
OUTPORT(command[0], "MODBUS RTU Device", 8)// send request  
IMPORT(response[0], "MODBUS RTU Device", 8, return_value)// read response  
  
end macro_command
```

18.11 編譯錯誤提示資訊

1. 錯誤資訊格式：

error C# : error 描述

(# 是錯誤資訊編號)

舉例：error C37 : undeclared identifier : i

當編譯後提示有錯誤資訊時，這個錯誤的描述內容可以參考錯誤資訊編號。

2. Error 描述

(C1) 語法 error：“識別字”

出現這個資訊時，有許多種可能；

舉例：

```
macro_command main()
char i, 123xyz // 不支援這個變數類型
end macro_command
```

(C2) ‘identifier’ used without having been initialized (使用的識別字沒有初始化)

巨集指令必須定義聲明的陣列變數的大小

舉例：

```
macro_command main()
char i
int g[i] // i 必須為一個數值常數
end macro_command
```

(C3) redefinition error : ‘identifier’ (識別字被重複定義)

函數名稱和變數名稱在有效範圍內，必須是唯一的。

舉例：

```
macro_command main()
int g[10], g // 重複定義錯誤
end macro_command
```

(C4) function 函數名稱 error : 'identifier' (函數名稱定義錯誤)

保留的關鍵字和常數，不能被定義為函數名稱

舉例：

```
sub int if() // 函數名稱定義錯誤
```

(C5) parentheses have not come in pairs (圓括號沒有成對的出現)

語句中缺少了 "(" or ")"

舉例：

```
macro_command main ) // 缺少了 "("
```

(C6) illegal expression without matching 'if' (if 語句中沒有合法的運算式)

也就是在 if 語句中缺少運算式

(C7) illegal expression (no 'then') without matching 'if' (if 語句中缺少了 then)

也就是 if 和 then 沒有成對

(C8) illegal expression (no 'end if') (if 語句中缺少了 end if)

缺少了 "end if"

(C9) illegal 'end if' without matching 'if' (end if 語句前缺少了 if)

End if 語句前缺少了 if 語句

(C10) illegal 'else' (不合法的 else 語句)

這個 "if" 語句的格式為：

```
if [邏輯運算式] then
```

```
[ else [if [邏輯運算式] then] ]
```

```
end if
```

任何與以上格式不符合的語句，在編譯時就會錯誤。

(C17) illegal expression (no 'for') without matching 'next' (沒有與 next 相配的 for 語句)

“for” 語句錯誤：在 “next” 前，缺少了 “for” 語句

(C18) illegal variable type (not integer or char) (不合法的變數類型)

變數類型定義錯誤，此處應為整數型態或字元型態變數

(C19) variable type error

缺少賦值語句

(C20) must be keyword 'to' or 'down' (缺少了關鍵字 “to” 或者 “down”)

缺少了關鍵字 “to” 或者 “down”

(C21) illegal expression (no 'next') (非法的運算式，缺少了 “next”)

“for” 語句的格式為：

for [變數] = [初始值] to [結束值] [step]

next [變數]

任何與上述格式不符合的語句，編譯時會錯誤。

(C22) ‘wend’ statement contains no ‘while’

迴圈缺少 “while” 關鍵字，“wend” 前面應有 “while” 關鍵字

(C23) illegal expression without matching ‘wend’

缺少 “wend” 關鍵字

“while” 語句的格式為：

while [邏輯運算式]

wend

任何不符合上述語法的，在編譯時會錯誤。

(C24) 語法 error : ‘break’

不合法的 “break” 語句。break 語句只能在 for 迴圈、while 迴圈選擇結構中使用。

(C25) 語法 error : ‘continue’

不合法的 “continue” 語句。continue 語句只會在 “for” 或者 “while” 語句中出現。

(C26) 語法 error

運算式不正確

(C27) 語法 error

運算式中缺少了一個運算符號可能會造成這個編譯錯誤資訊。

舉例：

```
macro_command main( )
int a, b
for a = 0 to 2
b = 4 + xyz // 不合法之處：xyz 變數沒有被定義
next a
end macro_command
```

(C28) must be ‘macro_command’

此處應該為 “macro_command”

(C29) must be key word ‘sub’

子函數的定義格式為：

```
sub [data type] function_函數名稱 (...)

.....
end sub
```

舉例::

```
sub int pow (int exp)

.....
end sub
```

任何不符合上述語法結構的，在編譯時會錯誤。

(C30) number of parameters is incorrect

參數個數不對。

(C31) parameter type is incorrect

參數資料類型不相配。調用函數時，參數必須在資料類型、個數上一一對應才能通過編譯，否則編譯時將出現此項錯誤訊息。

(C32) variable is incorrect

變數類型不正確。當變數被當成參數傳遞給一個函數時，變數的資料類型應與函數所宣告的類型相同，否則將出現此項錯誤訊息。

(C33) function 函數名稱 : undeclared function

沒有定義的函數名稱

(C34) expected constant expression

不合法的陣列下標表達形式

(C35) invalid array declaration

不合法的陣列定義

(C36) array index error

不合法的陣列下標

(C37) undeclared identifier : i 'identifier'

使用沒有定義的變數。只能使用已經定義的變數和函數，否則編譯時將出現此項錯誤訊息。

(C38) un-supported PLC data address

通訊函數 GetData(...)、SetData(...)的參數中有包含 PLC 位址類型資訊，當 PLC 位址類型不是此種 PLC 支援的位址類型時，編譯時將出現此項錯誤訊息。

(C39) 'idenifier' must be integer, char or constant

陣列的格式為：

聲明： array_函數名稱[constant] (constant is the size of the array)

使用： array_函數名稱[integer, character or constant]

任何不符合上述規則的陣列運算式，編譯時將會錯誤

(C40) execution 語法 should not exist before variable declaration or constant definition

變數定義語句的前面不能有執行語句

舉例：

```
macro_command main( )  
int a, b  
for a = 0 To 2  
    b = 4 + a  
int h, k // 定義變數語句在此處是錯誤的，在一個函數內定義變數語句的前面  
        不能有執行語句，例如 b = 4 + a  
  
next a  
end macro_command
```

(C41) float variables cannot be contained in shift calculation

移位運算中，運算元不能為浮點數。

(C42) function must return a value

函數應有返回值

(C43) function should not return a value

函數不應有返回值

(C44) float variables cannot be contained in calculation

運算中不能有 float 型資料

(C45) PLC address error

PLC 位址錯誤

(C46) array size overflow (max. 4k)

一維陣列的大小超過 4k

(C47) macro command entry function is not only one

巨集指令程式入口只能有一個

(C48) macro command entry function must be only one

巨集指令入口函數不是唯一。巨集指令的入口函數只能有一個，形式為：

```
macro_command function_函數名稱()  
end macro_command
```

(C49) an extended addressee's station number must be between 0 and 255

在巨集指令中，擴展地址內的站號大小只能從 0 到 255

舉例：

```
SetData(bits[0] , "PLC 1", LB , 300#123, 100)  
// illegal : 300#123 意思是站號為 300, 但是最大值是 255
```

(C50) an invalid PLC 函數名稱

在巨集指令中，PLC 的名稱並未定義在系統參數的設備列表中

(C51) macro command do not control a remote device

巨集指令只能控制本機連接的設備

舉例：

```
SetData(bits[0] , "PLC 1", LB , 300#123, 100)
```

“PLC1” 連接在遠端的 HMI 上，所以它不能被執行。

18.12 巨集指令範例程式

1. for 迴圈，各種運算式 (算術，移位元，邏輯，關係運算式)

```
macro_command main()
    int a[10], b[10], i

    b[0] = (400 + 400 << 2) / 401
    b[1] = 22 * 2 - 30 % 7
    b[2] = 111 >> 2
    b[3] = 403 > 9 + 3 >= 9 + 3 < 4 + 3 <= 8 + 8 == 8
    b[4] = not 8 + 1 and 2 + 1 or 0 + 1 xor 2
    b[5] = 405 and 3 and not 0
    b[6] = 8 & 4 + 4 & 4 + 8 | 4 + 8 ^ 4
    b[7] = 6 - (~4)
    b[8] = 0x11
    b[9] = 409

    for i = 0 to 4 step 1
        if (a[0] == 400) then
            GetData(a[0],"Device 1", 4x, 0,9)
            GetData(b[0],"Device 1", 4x, 11,10)
        end If
    next i
end macro_command
```

2. while, if, break 語句

```
macro_command main()
    int b[10], i
    i = 5
    while i == 5 - 20 % 3
        GetData(b[1], "Device 1", 4x, 11, 1)

        if b[1] == 100 then
            break
        end if
    wend
end macro_command
```

3. 總體變數和子函數調用

```
char g
sub int fun(int j, int k)
    int y

    SetData(j, "Local HMI", LB, 14, 1)
    GetData(y, "Local HMI", LB, 15, 1)
    g = y

    return y
end Sub
```

```
macro_command main()
    int a, b, i

    a = 2
    b = 3
    i = fun(a, b)
    SetData(i, "Local HMI", LB, 16, 1)
end macro_command
```

4. if 結構語句

```
macro_command main()
    int k[10], j

    for j = 0 to 10
        k[j] = j
    next j

    if k[0] == 0 then
        SetData(k[1], "Device 1", 4x, 0, 1)
    end if

    if k[0] == 0 then
        SetData(k[1], "Device 1", 4x, 0, 1)
    else
        SetData(k[2], "Device 1", 4x, 0, 1)
    end if

    if k[0] == 0 then
        SetData(k[1], "Device 1", 4x, 1, 1)
    else if k[2] == 1 then
        SetData(k[3], "Device 1", 4x, 2, 1)
    end If

    if k[0] == 0 then
        SetData(k[1], "Device 1", 4x, 3, 1)
    else if k[2] == 2 then
        SetData(k[3], "Device 1", 4x, 4, 1)
    else
        SetData(k[4], "Device 1", 4x, 5, 1)
    end If
end macro_command
```

5. while 和 wend 結構語句

```
macro_command main()
    char i = 0
    int a[13], b[14], c = 4848

    b[0] = 13

    while b[0]
        a[i] = 20 + i * 10

        if a[i] == 120 then
            c=200
            break
        end if

        i = i + 1
    wend

    SetData(c, "Device 1", 4x, 2, 1)
end macro_command
```

6. break 和 continue 語句結構

```
macro_command main()
    char i = 0
    int a[13], b[14], c = 4848
    b[0] = 13

    while b[0]
        a[i] = 20 + i * 10

        if a[i] == 120 then
            c = 200
            i = i + 1
            continue
        end if
        i = i + 1

        if c == 200 then
            SetData(c, "Device 1", 4x, 2, 1)
            break
        end if
    wend
end macro_command
```

7. 陣列結構

```
macro_command main()
    int a[25], b[25], i

    b[0] = 13

    for i = 0 to b[0] step 1
        a[i] = 20 + i * 10
    next i

    SetData(a[0], "Device 1", 4x, 0, 13)
end macro_command
```

18.13 巨集指令 TRACE 函數

巨集指令的 TRACE 函數，搭配使用 EasyDiagnoser，可用來檢視所使用變數目前的內容。
下面示範如何在巨集指令中利用 TRACE 命令。

首先，請在工程檔案中新增 macro_1，並在 macro_1 的內容中加入 **TRACE("LW = %d", a)**，“%d”表示使用 10 進制顯示 LW 目前的數值。macro_1 的內容如下：

```
macro_command main()

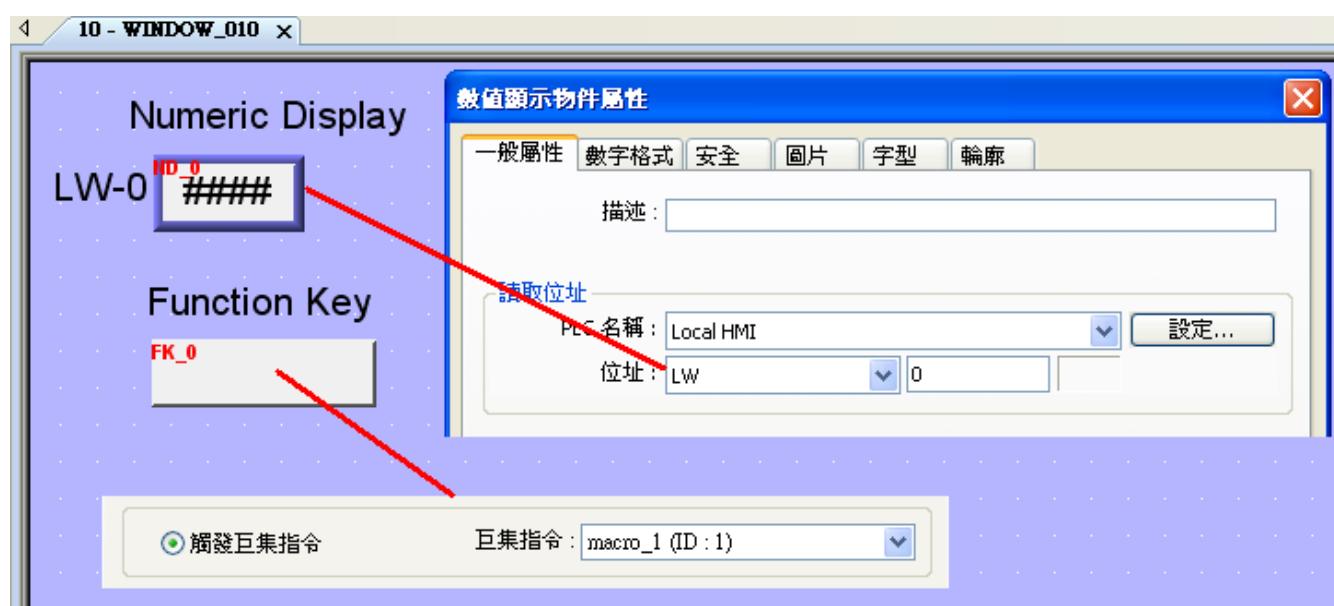
short a
GetData(a, "Local HMI", LW, 0, 1)
a= a + 1
SetData(a, "Local HMI", LW, 0, 1)
TRACE("LW0 = %d", a)

end macro_command
```

(TRACE 命令更詳細的用法請參考下面的說明。)



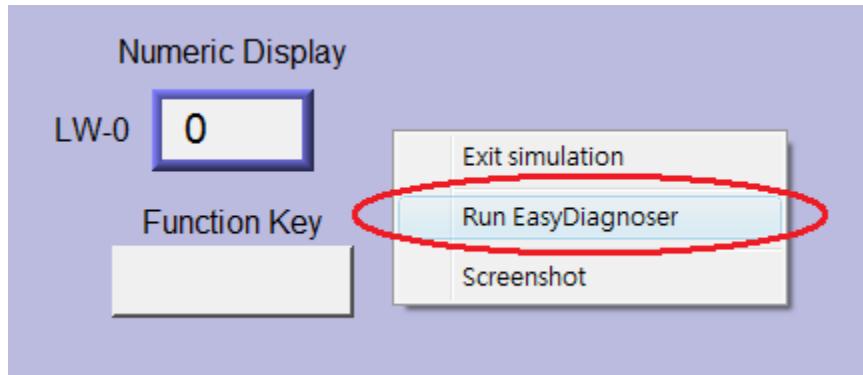
接著在工程檔案的第 10 頁分別加上 [數值顯示] 與 [功能鍵] 物件，物件的設定內容請參考下圖，[功能鍵] 物件用來執行 macro_1。



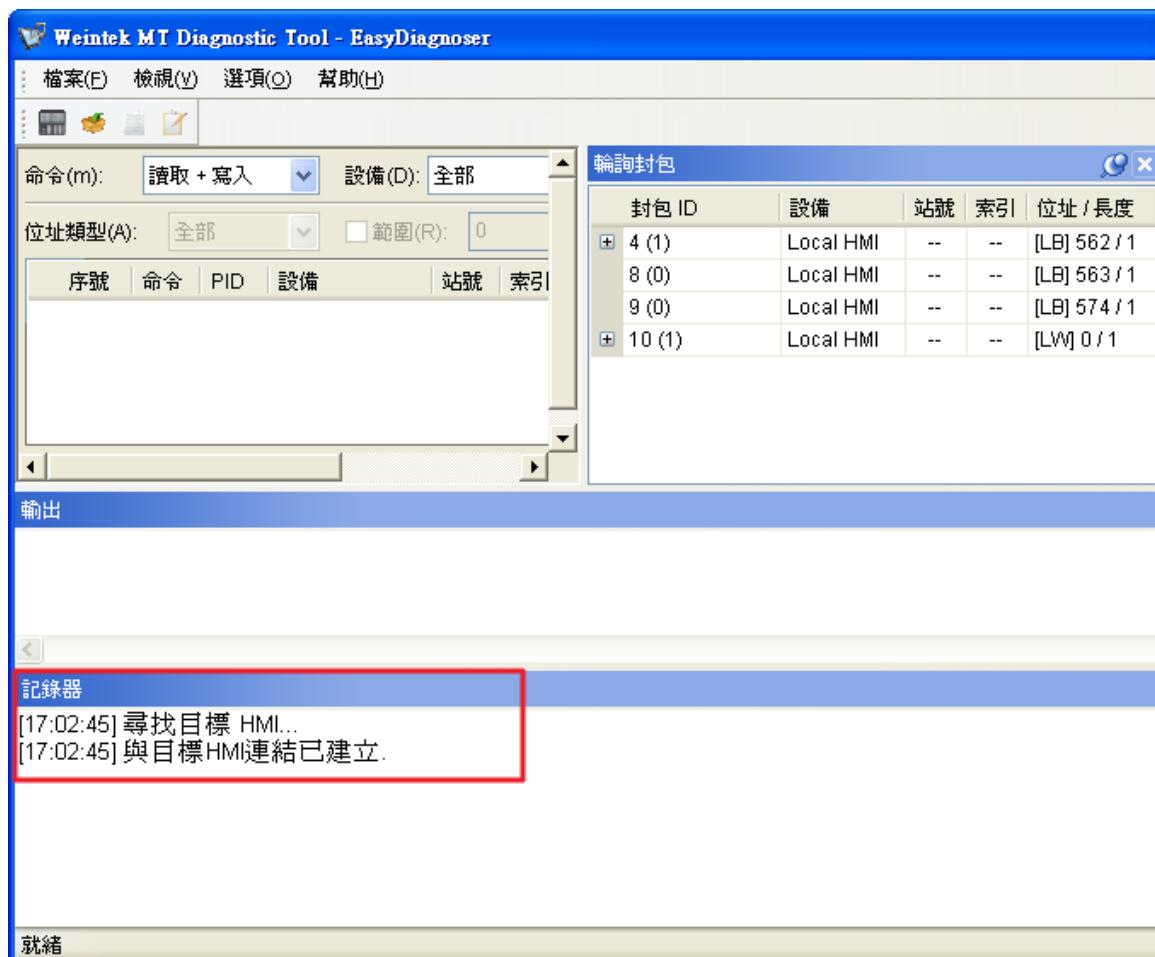
最後編譯已完成的工程檔案並執行離線或線上模擬。



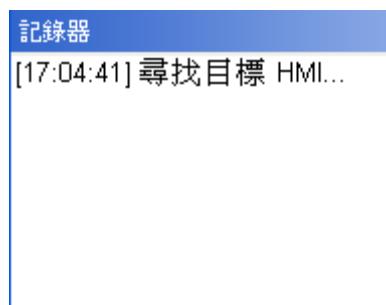
在 PC 上進行模擬功能時，點擊滑鼠右鍵後選擇選單上的 "Run EasyDiagnoser"。



此時即會出現 EasyDiagnoser 的畫面，[Logger] 窗口用來顯示 EasyDiagnoser 是否可以連接上需要監視的 HMI，[Output] 窗口用來顯示 TRACE 的執行結果，下圖表示 EasyDiagnoser 已成 功連接上 HMI。



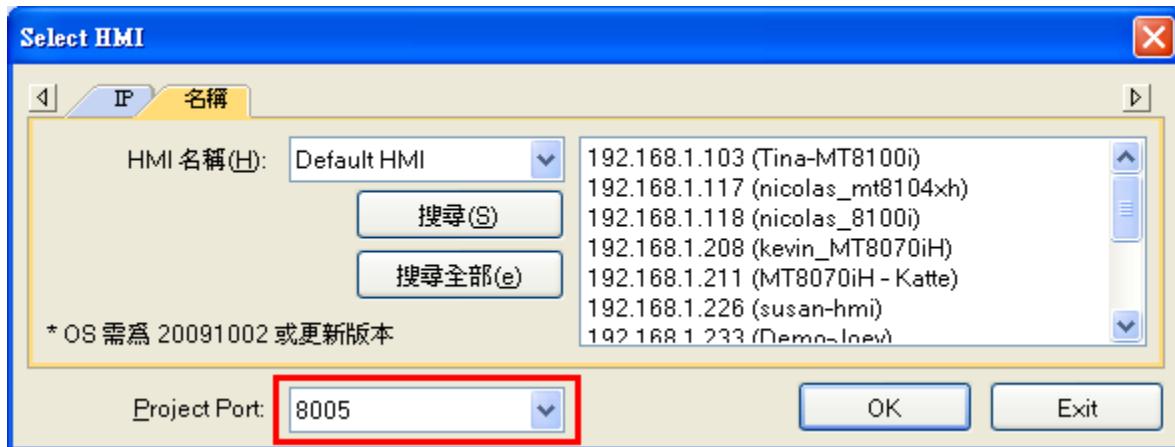
若未成功連接上 HMI，[Logger] 的窗口會顯示下面的內容：



未連線成功的可能原因是 PC 未成功執行模擬功能；另一個原因是在 PC 執行模擬功能的工程檔案所使用的 Port No. 不正確（可能已被系統佔用），此時請更改工程檔案的 Port No.（請參考下圖），再次編譯重新執行模擬功能即可。



開啟 EasyDiagnoser 時，應設定與工程檔案相同的連接埠。



從所設定的 Port No. 算起連續 3 個 port 將保留給 HMI 做通訊用。例如，上圖中設定 Port No. 為 8005，則 8005、8006、8007 三個 port 將被保留。因此在 PC 上模擬時，應確定這些被保留的 port 未被其他程式所佔用。

1. TRACE 命令語法如下：

函數名稱	TRACE				
語法	TRACE(format, argument)				
描述	<p>一個執行中的巨集指令可以使用此函數，監視變數內容的變化，並列印字串，以協助除錯。使用者應開啟 EasyDiagnoser 觀看此函數的輸出結果。</p> <p>當 TRACE 函數抓取一個 % 開頭的特殊字元，將同時從 <i>argument</i> 抓取一個參數做格式化後輸出。</p> <p><i>format</i> 代表列印格式，支援 % 開頭的特殊字元。特殊字元格式如下，其中方括號內的欄位為可選，粗體字欄位為必需：</p> $\%[\text{flags}] [\text{width}] [.precision] \text{type}$ <p>每個欄位的意義如下所述：</p> <p><i>flags</i> (可選) :</p> <ul style="list-style-type: none"> - + <p><i>width</i> (可選) :</p> <p>十進位正整數，指定應預留的字元寬度，不足部份補空白字元。</p> <p><i>precision</i> (可選) :</p> <p>十進位正整數，指定精確度，以及輸出字元數。</p> <p><i>type</i> :</p> <table border="0"> <tr> <td>C 或 c</td> <td>: 以字元方式輸出</td> </tr> <tr> <td>d</td> <td>: 以 signed 十進位整數輸出</td> </tr> </table>	C 或 c	: 以字元方式輸出	d	: 以 signed 十進位整數輸出
C 或 c	: 以字元方式輸出				
d	: 以 signed 十進位整數輸出				

	<p>i : 以 signed 十進位整數輸出 o : 以 unsigned 八進位整數輸出 u : 以 unsigned 十進位整數輸出 X 或 x : 以 unsigned 十六進位整數輸出 E 或 e : 以科學表示法輸出 f : 以單倍精確度浮點數輸出</p> <p><i>format</i> 字串最長支援 256 個字元，多出的字元將被忽略。 <i>argument</i> 部份可寫可不寫。但一個特殊字元應搭配一個變數。</p>
舉例	<pre>macro_command main() char c1 = 'a' short s1 = 32767 float f1 = 1.234567 TRACE("The results are") // 輸出 : The results are TRACE("c1 = %c, s1 = %d, f1 = %f", c1, s1, f1) // 輸出 : c1 = a, s1 = 32767, f1 = 1.234567 end macro_command</pre>

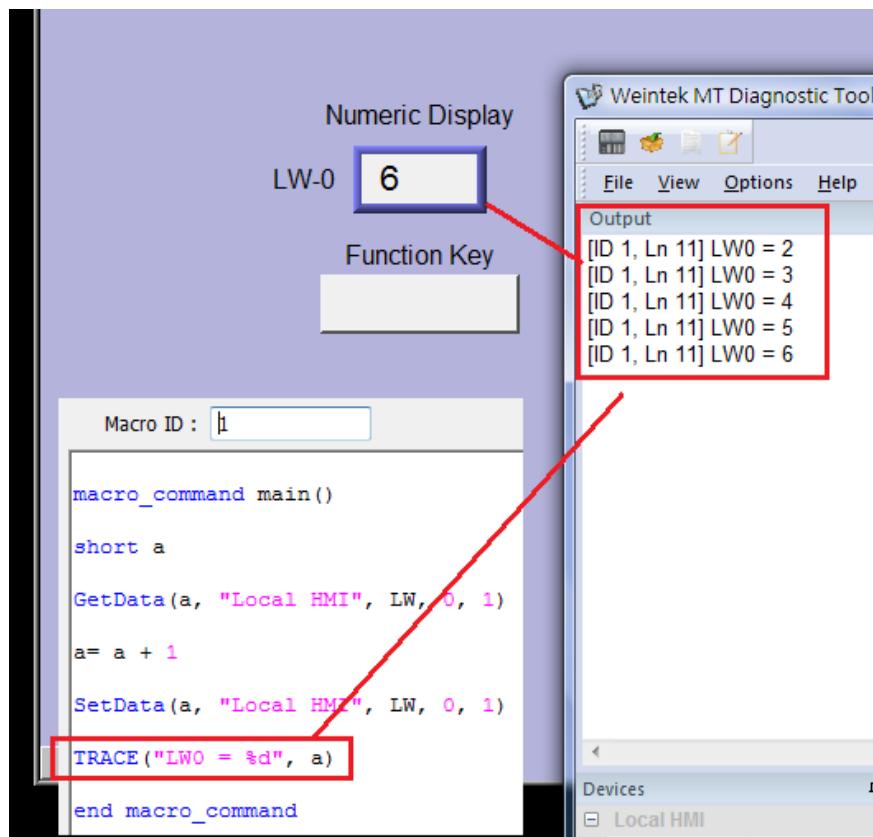
2. 新增 LB9059 – disable MACRO TRACE function (when ON)。

當設定為 ON 時，TRACE 將不會把數據輸出到 EasyDiagnoser。

- 也可以直接利用 Utility Manager 執行 EasyDiagnoser.exe，Utility Manager 將會顯示網路上目前存在的 HMI，此時只要選擇要監看通訊狀態的 HMI 即可。請注意 Project Port 的部份應設定與工程檔案所使用的 Port No. 相同。

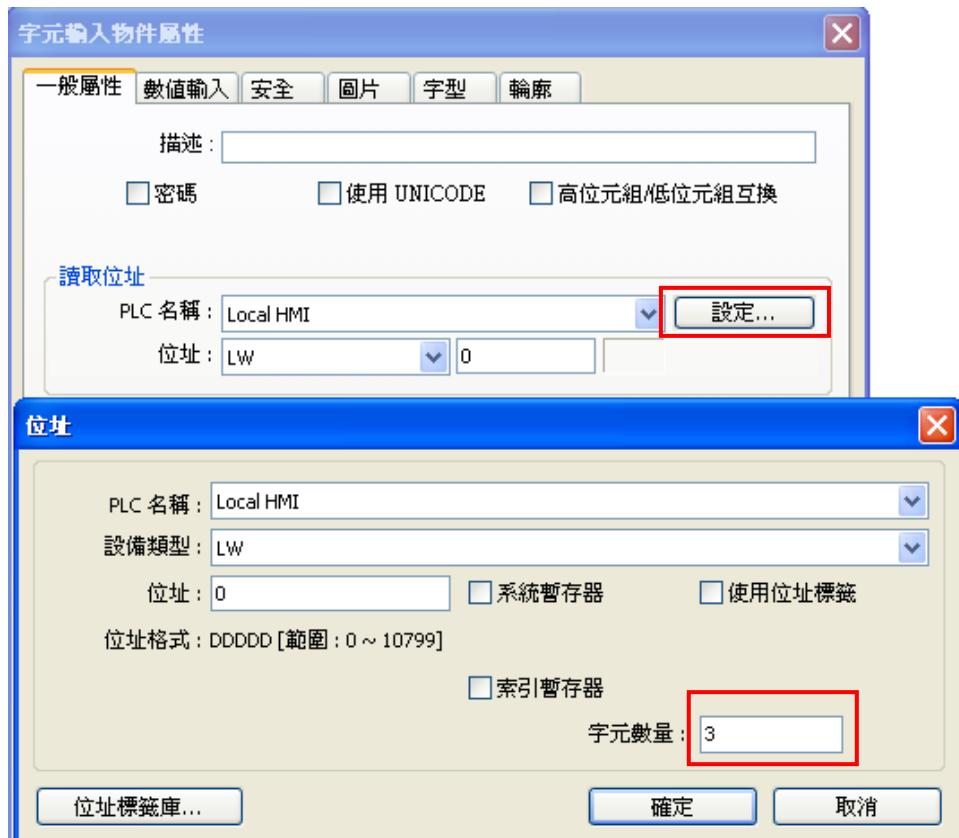


4. 將工程檔下載到 HMI 實際操作。當 EasyDiagnoser 無法連接上需要監視的 HMI 時，一般可能的原因是 HMI 未上電。或是 Port No. 不正確，可能發生 EasyDiagnoser 不斷連上 HMI 又斷線的情況。請確定 EasyDiagnoser 應設定與工程檔案相同的 Port No.，更改方式如前面所述。
5. 當 EasyDiagnoser 成功與 HMI 連結後，只要執行 macro_1，即可發現 [Output] 窗口顯示目前 TRACE 的執行結果。



18.14 字串處理函式使用方法

巨集指令提供字串處理函式，令使用者可以很方便的對字串進行各種操作。所謂字串，是由一連串的 ASCII 字元組成，每一個 ASCII 字元佔用 1 位元組 (byte)，在 16 位元暫存器中是以低位元組優先方式儲存。舉例來說，我們利用一個文字輸入元件在 LW0 ~ LW2 的位置，寫入一條字串 "abcdef"，設定如下：



在此文字輸入元件輸入 "abcdef"：

abcdef

此字串在 LW0 ~ LW2 中的存放方式如下圖所示 (LB 代表低位元組，HB 代表高位元組)：

	HB	LB
LW0	'B'	'A'
LW1	'D'	'C'
LW2	'F'	'E'
LW3		
LW4		
LW5		

由於文字輸入元件顯示的資料長度單位為 word，而每個 ASCII 字元長度為一個 byte，所以每次最少會顯示兩個字元，此部份在文字輸入元件的章節有詳細的說明。因此上面的例子中，設定文字輸入元件的字元數量為 3，表示最多可輸入 / 顯示 6 個 ASCII 字元。

目前提供的字串處理函式，其功能整理如下表：

函數名稱	說明
StringGet	獲取 PLC 的字串資料。
StringGetEx	獲取 PLC 的字串資料，不等待 PLC 回應，逕自往下執行。
StringSet	將字串數據寫到 PLC 中。
StringSetEx	將字串數據寫到 PLC 中，不等待 PLC 回應，逕自往下執行。
StringCopy	利用此函數進行字串的複製。
StringMid	利用此函數可將一個字串中的某一段子字串提取出來。
StringDecAsc2Bin	此函數將十進位字串轉換成整數。
StringBin2DecAsc	此函數將整數轉換成十進位字串。
StringDecAsc2Float	此函數將十進位字串轉換成浮點數。
StringFloat2DecAsc	此函數將浮點數轉換成十進位字串。
StringHexAsc2Bin	此函數將十六進位字串轉換成整數。
StringBin2HexAsc	此函數將整數轉換成十六進位字串。
StringLength	取得字串的長度。
StringCat	連接兩字串。
StringCompare	比較兩字串的內容是否相等。大小寫視為不同。
StringCompareNoCase	比較兩字串的內容是否相等。大小寫視為相同。
StringFind	在某個字串中尋找另一個字串。
StringReverseFind	在某個字串中尋找另一個字串，從後面開始找。
StringFindOneOf	尋找某字串中任一個字元在另一個字串中第一次出現的位置。
StringIncluding	從某字串第一個字元開始提取包含特定字元的一段字串。
StringExcluding	從某字串第一個字元開始提取不包含特定字元的一段字串。
StringToUpper	字元全部轉換成大寫。
StringToLower	字元全部轉換成小寫。
StringToReverse	字串反轉。
StringTrimLeft	裁剪字串開頭的特定字元。
StringTrimRight	裁剪字串尾端的特定字元。
StringInsert	插入字串到另一字串中的特定位置。

上表中所有字串處理函數的詳細規格與使用範例請參考內置函數功能的章節。下面將舉例說明如何從新增一個巨集指令的方法開始，一直到執行模擬為止，帶您一步步建立一個可執行的工程檔，以實際體會字串處理函數強大的功能。

1. 說明如何從暫存器讀入與寫入一個字串。

請新增一個巨集指令：

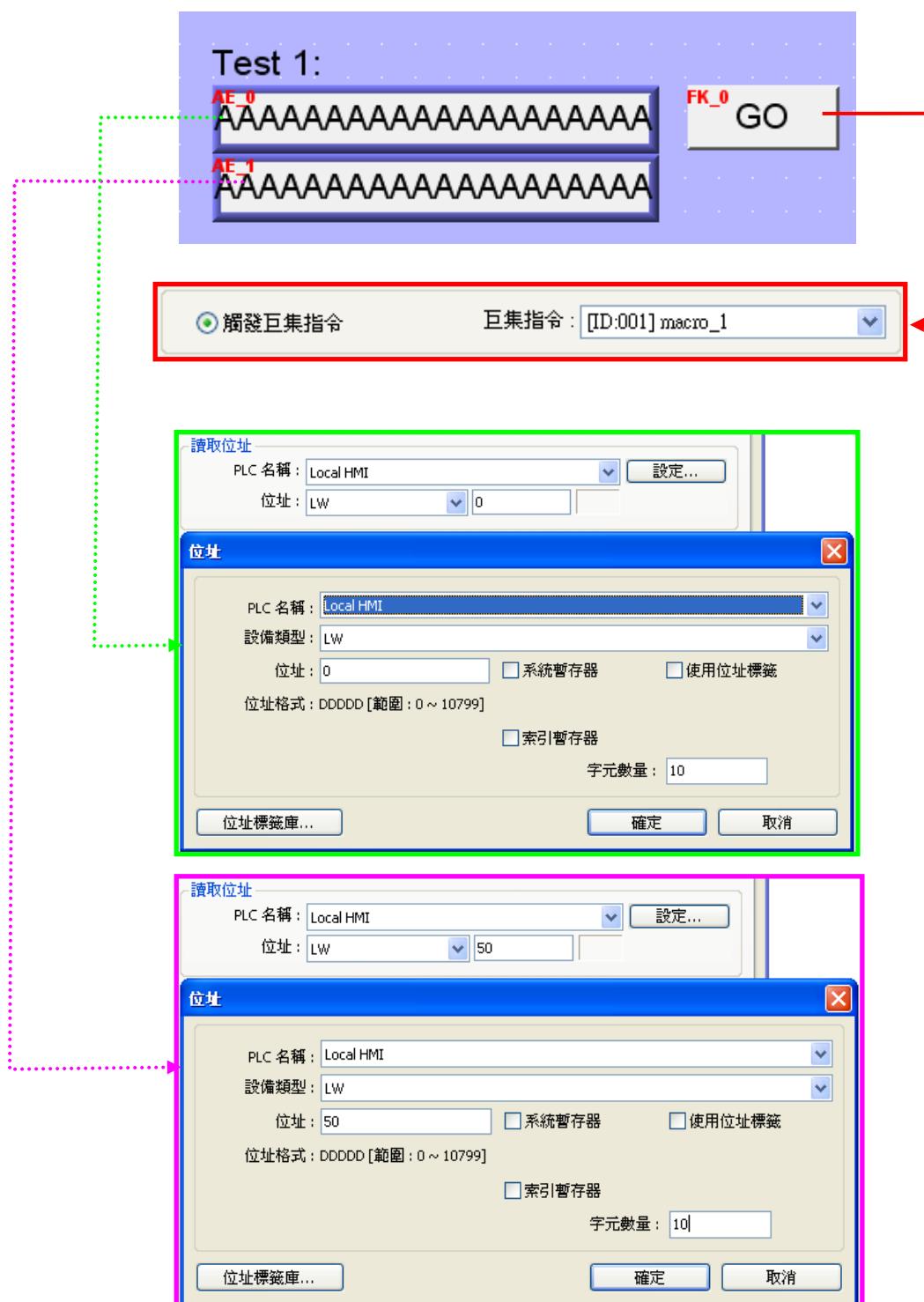


在巨集指令編輯器編輯以下內容：

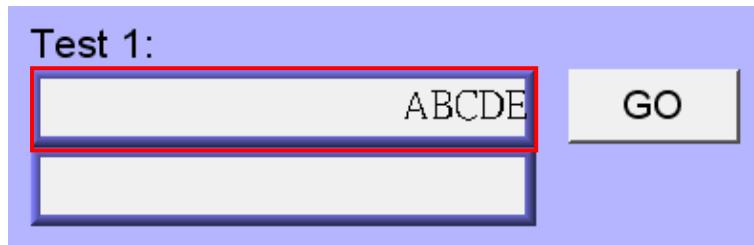


此巨集指令的目的是利用 `StringGet` 函數從暫存器中讀入一條字串放入字元陣列 `str` 中，並利用 `StringSet` 輸出 `str` 的內容。

接著在工程檔案的第 10 頁分別加上 ASCII Input 與 Function Key 物件，物件的設定內容請參考下圖，Function Key 物件用來執行 `macro_1`。



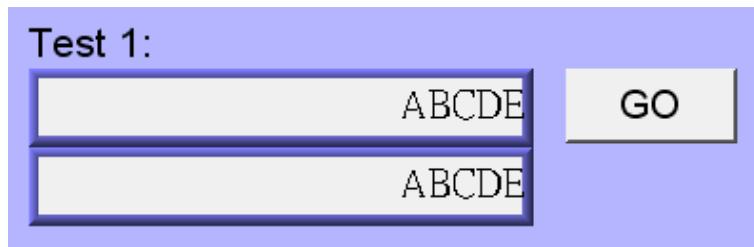
最後編譯 已完成的工程檔案並執行離線 (Off-line) 或線上 (on-line) 模擬，並在模擬畫面按照以下步驟操作：



第一步：輸入字串



第二步：按下 "GO" 按鈕



第三步：顯示結果

2. 字串之初始化。

在巨集指令編輯器編輯以下內容：

```

編輯器

編號 : 1 巨集指令名稱 : macro_1
 週期執行

1 macro_command main()
2
3
4 char str1[20] = "abcde"
5 char str2[20] = {'a', 'b', 'c', 'd', 'e'}
6
7 StringGet(str[0], "Local HMI", LW, 0, 20)
8 StringGet(str[0], "Local HMI", LW, 50, 20)
9
10 end macro_command

```

用雙引號 ("") 刮起來的內容視為一個字串。str1 是以字串方式初始化，而 str2 是以字元陣列方式初始化。利用 ASCII Input 元件分別顯示其內容可以發現兩者之間的不同：



以字串方式初始化時巨集編譯器會在字串結尾後面加上結束符號 '\0' 代表字串結束。利用 **StringSet** 將字串寫入暫存器時遇到結束符號便會停止繼續寫入陣列後面的內容。即使 **data count** 被設為大於字串長度的數值，顯示字串時只會顯示結束符號之前的內容。

以字元陣列方式初始化時陣列內容並不會被視為一個字串，因此也不會加上結束符號 '\0'。寫入暫存器的字元數會依據 **data count** 所設定的數值決定。

3. 一個簡單的帳號密碼登入頁面。

在巨集指令編輯器編輯以下內容：

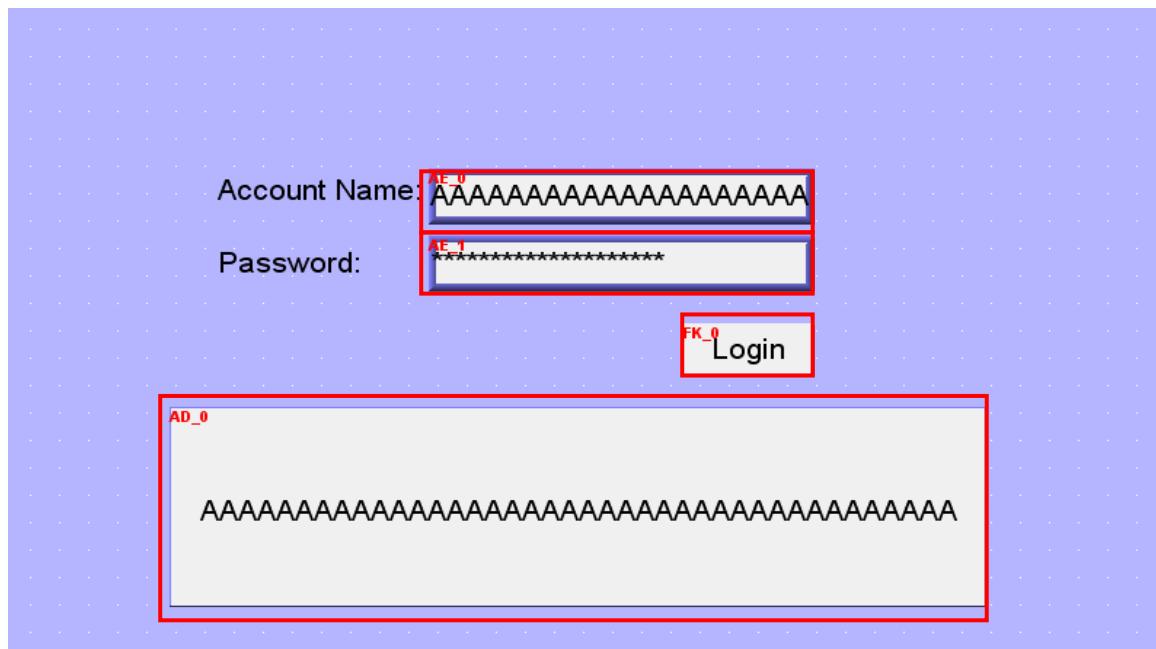
編輯器

編號： 1 巨集

```
macro_command main()
1
2
3 char name[20] = "admin"
4 char password[20] = "123456"
5 char name_input[20], password_input[20]
6 char message_success[40] = "Success! Access Accepted."
7 char message_fail[40] = "Fail! Access Denied."
8 char message_clear[40]
9 bool name_match = false, password_match = false
10
11 StringGet(name_input[0], "Local HMI", LW, 0, 20)
12 StringGet(password_input[0], "Local HMI", LW, 50, 20)
13
14 name_match = StringCompare(name_input[0], name[0])
15 password_match = StringCompare(password_input[0], password[0])
16
17 FILL(message_clear[0], 0x20, 40) // FILL with white space
18 StringSet(message_clear[0], "Local HMI", LW, 100, 40)
19 if(name_match == true and password_match == true) then
20     StringSet(message_success[0], "Local HMI", LW, 100, 40)
21 else
22     StringSet(message_fail[0], "Local HMI", LW, 100, 40)
23 end if
24 end macro_command
```

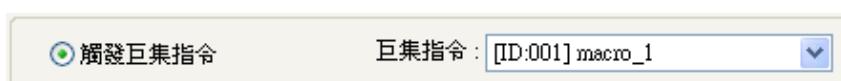
此巨集將利用 **[StringGet]** 取得使用者輸入的帳號密碼的字串，分別放入 `name_input` 與 `password_input` 兩個陣列中。接著利用 **[StringCompare]** 比對帳號密碼，若帳號相符，則 `name_match` 設為 `true`；若密碼相符，則 `password_match` 設為 `true`。最後檢查 `name_match` 與 `password_match` 的值，若同時為 `true`，表示登入成功，並印出 “**Success! Access Accepted.**” 字串。若其中之一不相符，則印出 “**Fail! Access Denied.**” 字串。

接著在工程檔案的第 10 頁分別加上 ASCII Input  與 Function Key  物件，物件的設定內容請參考下圖，Function Key 物件用來執行 `macro_1`。

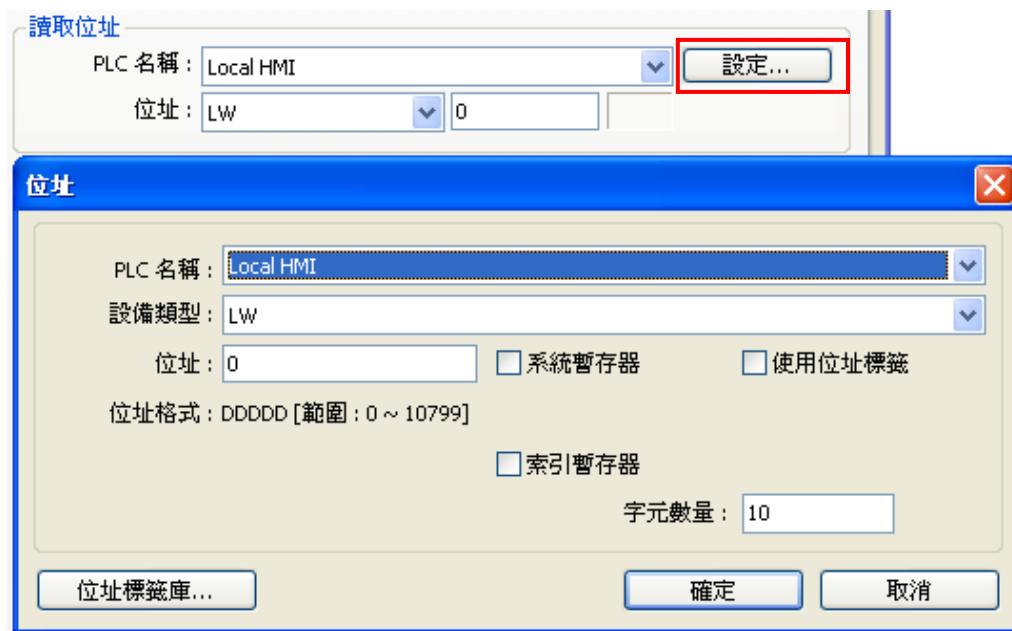


元件設定：

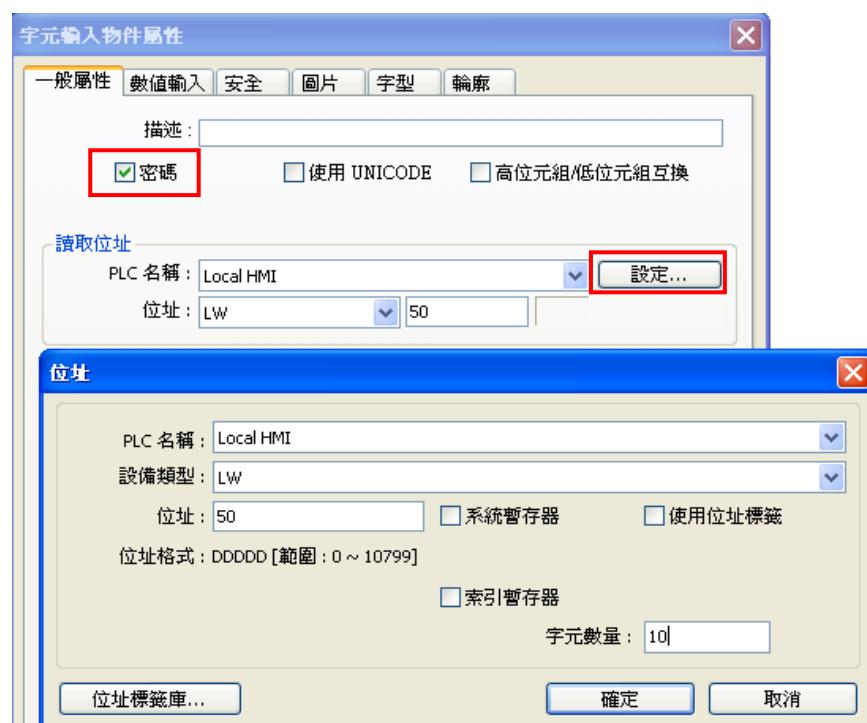
元件 1：Function Key 物件 



元件 2 : ASCII Input 物件

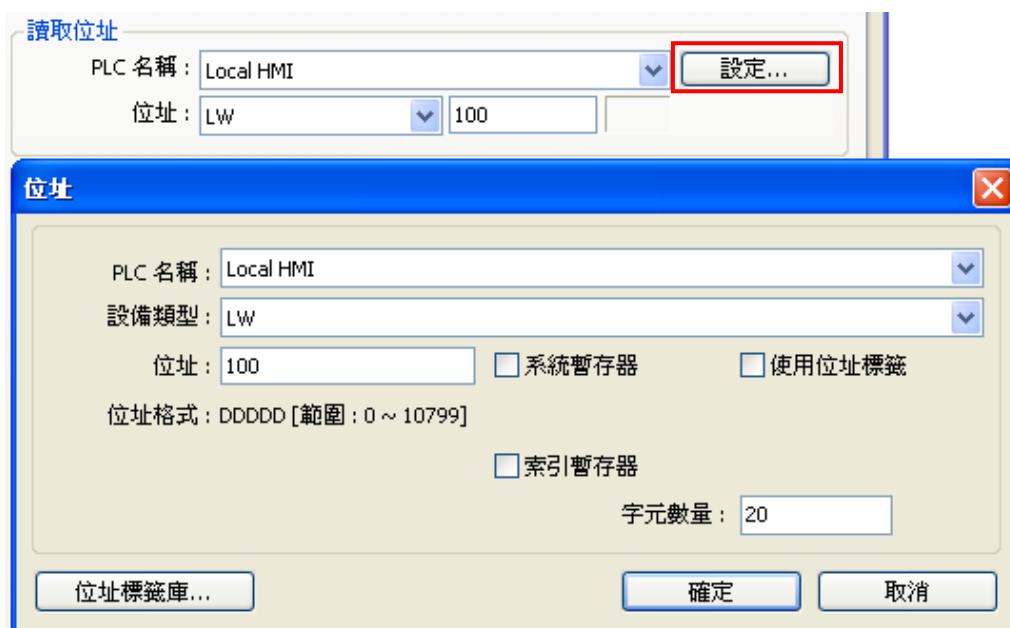


元件 3 : ASCII Input 物件

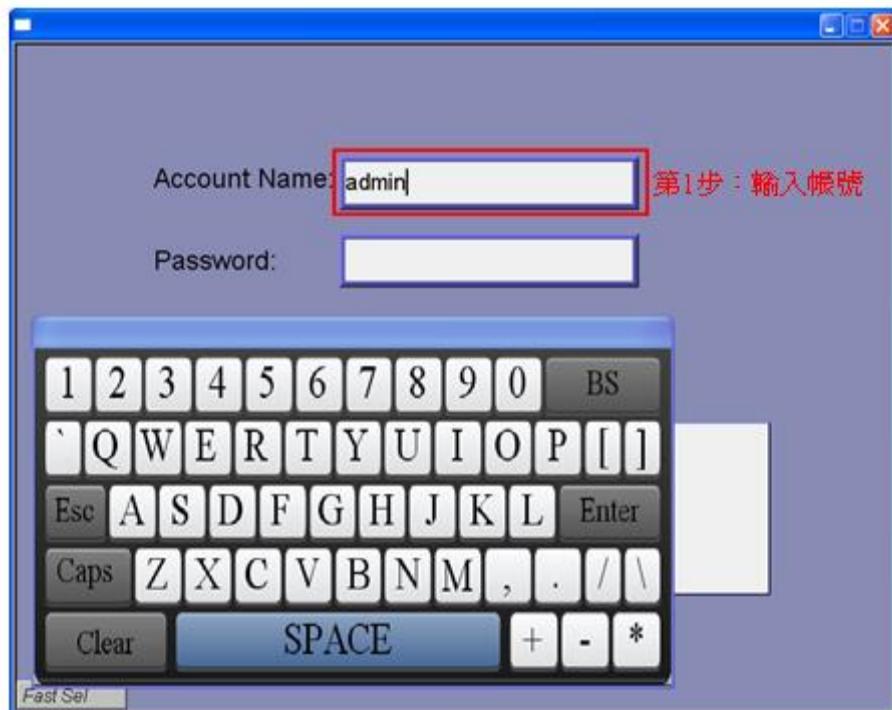


元件 4：ASCII Display 物件

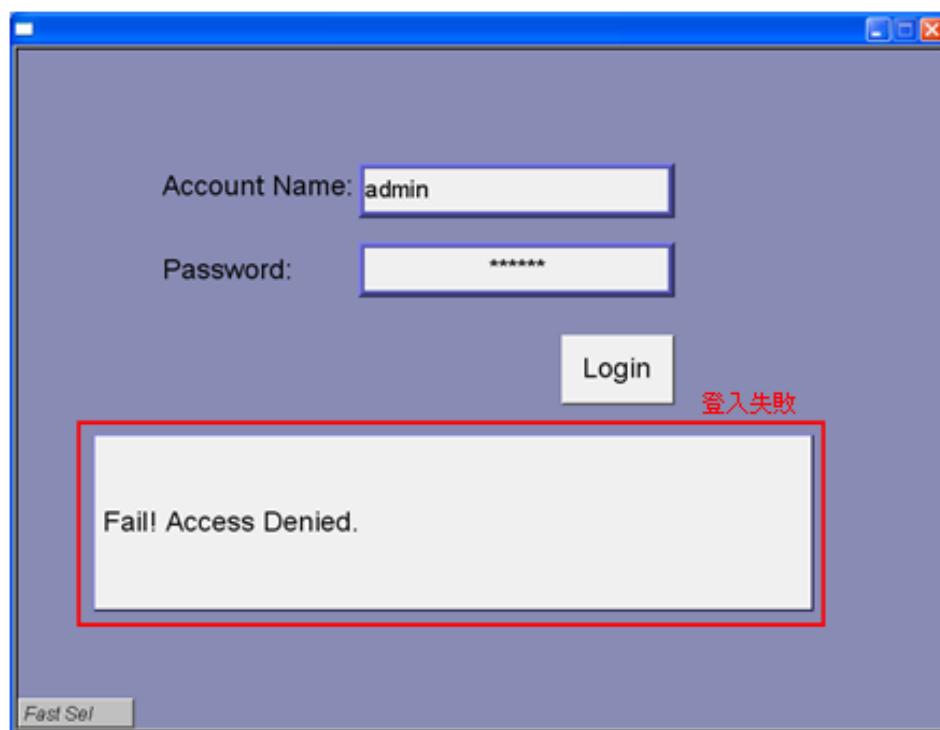
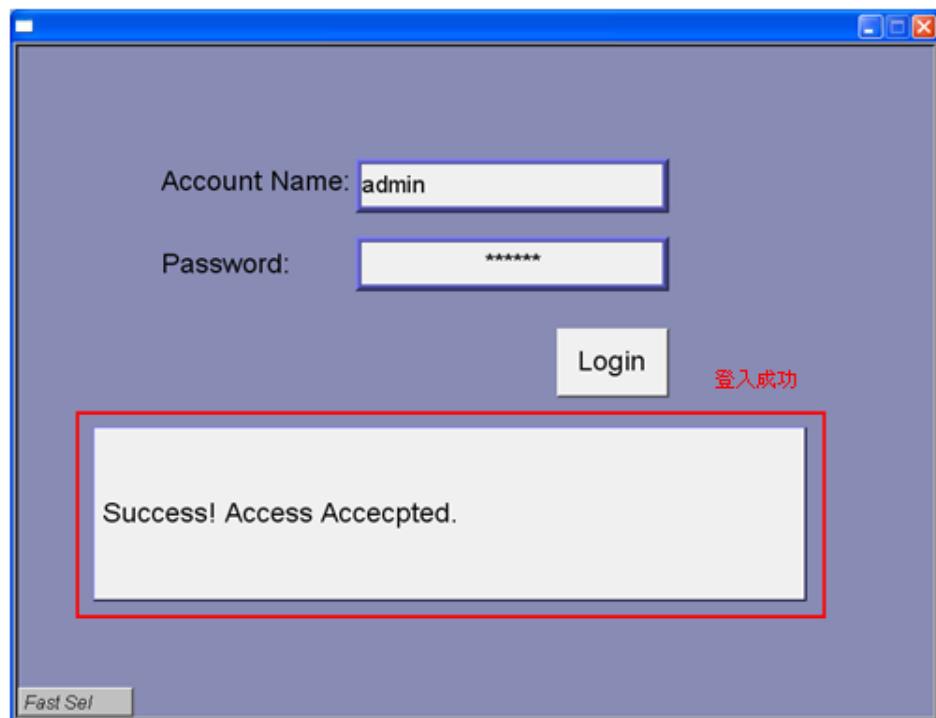
ABC



最後編譯 已完成的工程檔案並執行離線 (Off-line) 或線上(on-line) 模擬，並在模擬畫面按照以下步驟操作：



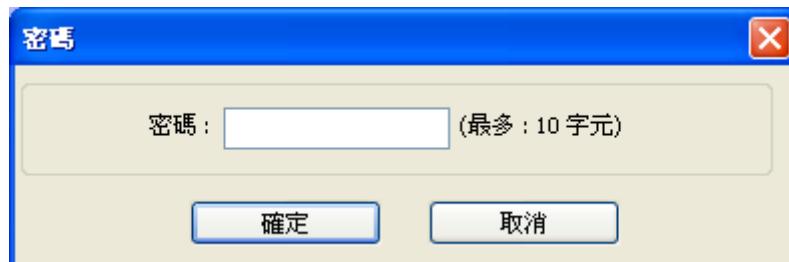




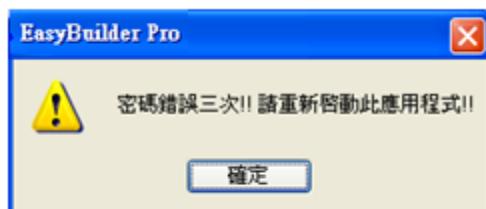
18.15 巨集指令密碼保護



當設定巨集指令 **【密碼保護】** 功能後，用戶欲開啟巨集指令編輯器窗口時，需先輸入正確的密碼。



當輸入不正確的密碼三次，需重新啟動 EasyBuilder，才能重新輸入密碼。

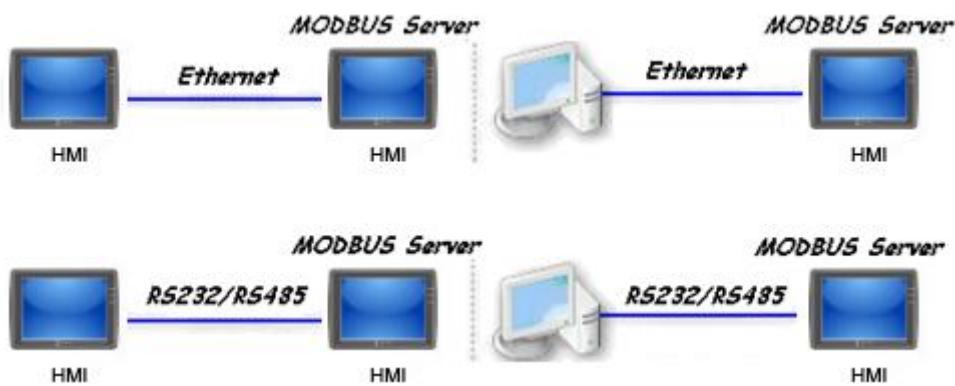


- 當開啟巨集指令密碼保護功能後，反編譯 XOB 檔案將無法回復巨集指令的內容。

第十九章 如何將 HMI 設定成 MODBUS 裝置

19.1 將 HMI 設定成 MODBUS 裝置

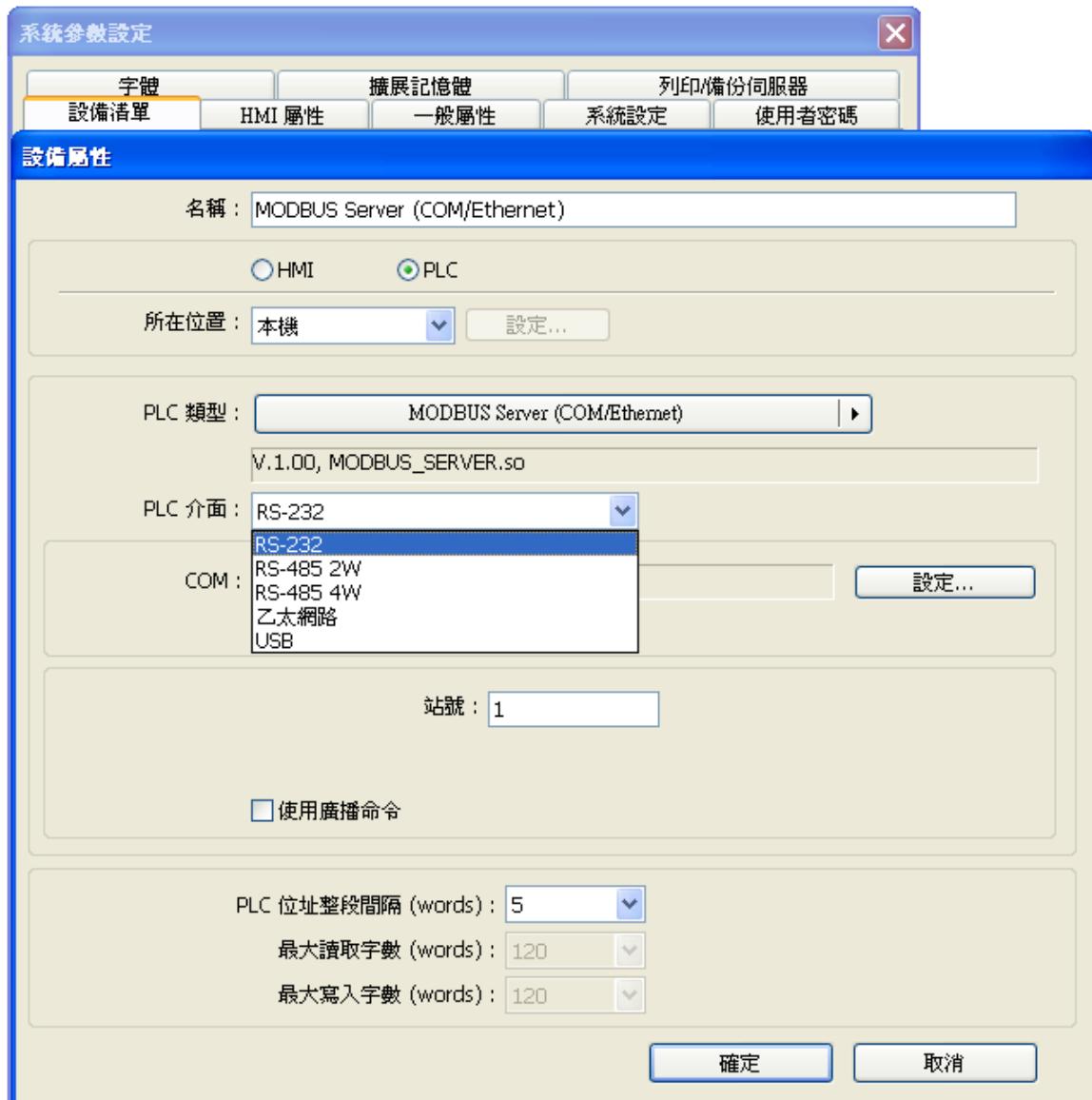
將 HMI 設定成 MODBUS 裝置後，透過 MODBUS 協議即可讀寫 HMI 上的數據。



上圖顯示 HMI 被設定成 MODBUS 裝置 (又稱為 MODBUS Server)，HMI、PC 或其它裝置只需使用 MODBUS 協議，透過 Ethernet 或 RS-232 / RS-485 介面，即可讀寫 HMI 上的數據。以下將說明如何將 HMI 設定成 MODBUS 裝置，並說明讀寫 HMI 上數據的方式。

19.1.1 建立一個 MODBUS Server 裝置

要將 HMI 設定為 MODBUS 裝置，首先需在 **【設備清單】** 中增加一個新的設備，此時 PLC 類型需挑選 MODBUS Server，**【PLC 介面】** 可以挑選 RS-232、RS-485 2W、RS-485 4W 或乙太網路。



當 PLC 介面選擇 **[RS-232]** 或 **[RS-485]** 時，需選擇使用的 **[COM]** (COM 1 ~ COM 3)，並設定正確的通訊參數。如下圖，此時 MODBUS Server 的 **[站號]** 設定為 1。

點選 **[設定]**，可以設定 **[限制 LW 最大讀取/寫入位址]**。當工程檔案的物件使用 LW 暫存器時，超過此範圍的位址將不會被 Modbus 客戶端讀/寫。



當 PLC 介面選擇 **[乙太網路]** 時，需設定 **[連接埠號]**。



因 MODBUS Server 與 HMI 須使用相同的 **[連接埠]**，若要更改 MODBUS Server 的連接埠，需在 **[HMI 屬性]** 頁籤中修改。



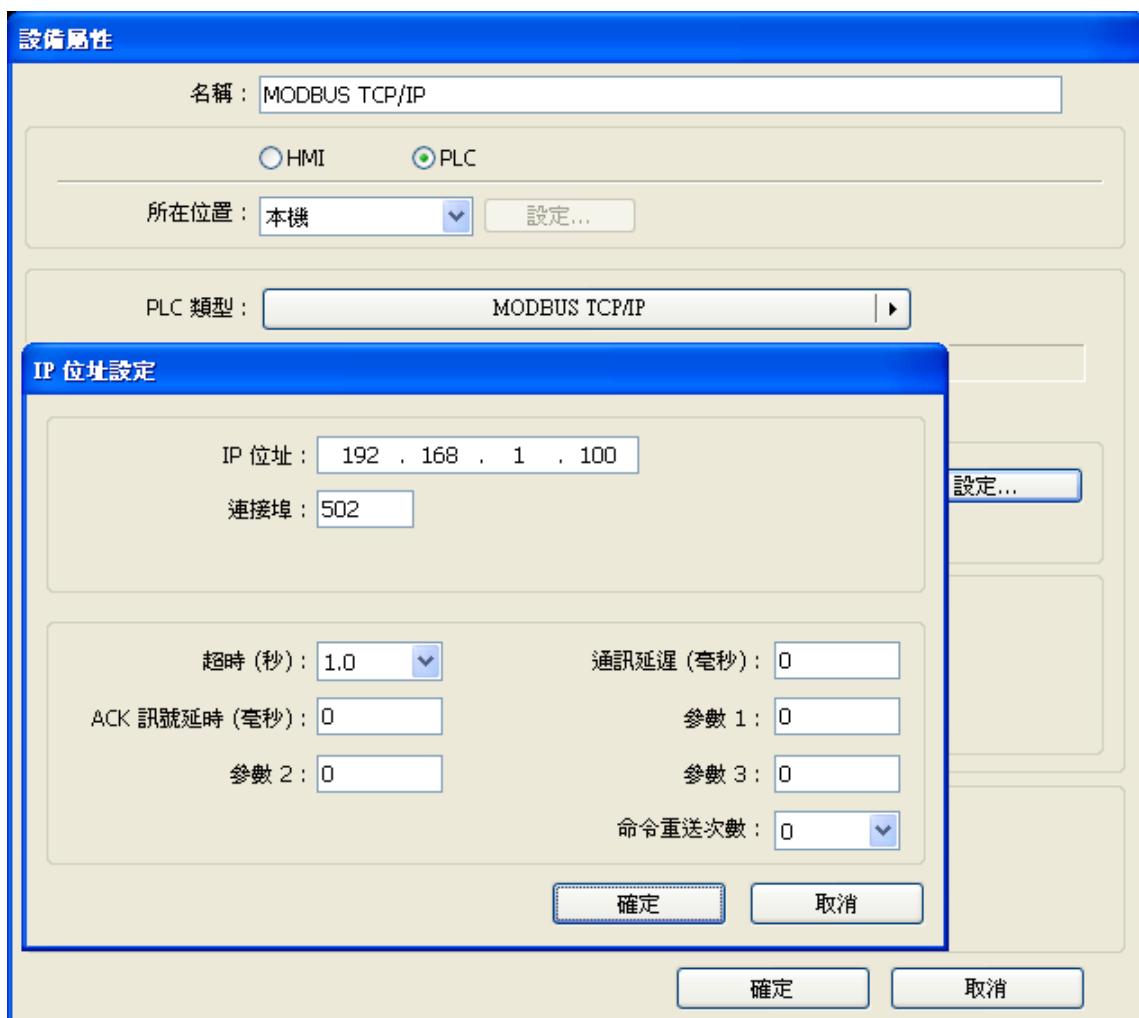
在按下確定鍵後，即可在 **【設備清單】** 中發現一個新的裝置：MODBUS Server，此時即完成 MODBUS 裝置的設定，在完成 mtp 檔案的編譯並將獲得的 xob 檔案下載到 HMI 後，即可透過 MODBUS 協議讀寫 HMI 上的數據。



19.1.2 讀寫一個 MODBUS Server 裝置

兩台 HMI 可以透過設定成 MODBUS Client (主機端) 和 Server (從機端) 相互通訊。

首先在 Client 端的設備清單中，需增加一個新的設備。若 Client 端使用【乙太網路】介面，則 **[PLC 類型]** 需挑選 MODBUS TCP/IP，並正確設定 **[IP 位址]** (即 server 端所在位置的 IP)、**[連接埠]** 與 **[站號]**。



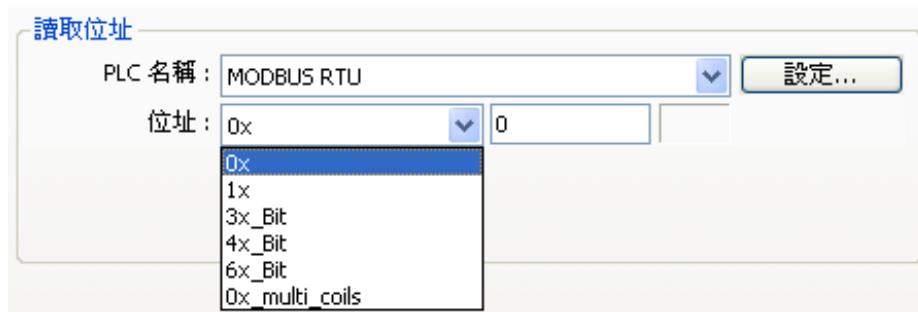
若 Client 端要使用 **[RS-232]** 或 **[RS-485]** 介面。則 **[PLC 類型]** 需挑選 MODBUS RTU，並正確設定各項通訊參數。



完成各項設定並按下確定鍵後，即可在**【設備清單】**中發現一個新的設備“MODBUS RTU”。

系統參數設定						
字體	擴展記憶體	列印/備份伺服器				
設備清單	HMI 屬性	一般屬性	系統設定	使用者密碼		
設備清單：						
編號	名稱	位置	設備類型	介面類型	通訊協議	站號
本機 HMI	Local HMI	本機	MT6070iH/MT8070...	停用	N/A	N/A
本機 PLC 1	MODBUS RTU	本機	MODBUS RTU	COM 1 (9600,E,8,1)	RS485 2W	1

開啟各個物件的設定頁，在 **[PLC 名稱]** 選擇 MODBUS RTU 後，即可設定 MODBUS 裝置的各項讀寫位址。



此時因被讀寫的裝置 (Server 端) 為 HMI，所以實際讀寫的位置的對應關係如下：

讀寫 0x/1x (1 ~ 12096)	對應到 讀寫 LB (0 ~ 12095)
讀寫 3x/4x/5x (1 ~ 9999)	對應到 讀寫 LW (0 ~ 9998)
讀寫 3x/4x/5x (10000 ~ 65535)	對應到 讀寫 RW (0 ~ 55535)

19.2 線上更改 MODBUS Server 的站號

EasyBuilder 提供下列系統暫存器，讓使用者可以線上更改 MODBUS Server 所使用的站號。

- [LW-9541] MODBUS/ASCII server 站號 (COM 1)
- [LW-9542] MODBUS/ASCII server 站號 (COM 2)
- [LW-9543] MODBUS/ASCII server 站號 (COM 3)
- [LW-9544] MODBUS/ASCII server 站號 (乙太網路)

19.3 關於 MODBUS 各位址的說明

EasyBuilder 中 MODBUS 協定的設備類型為 0x, 1x, 3x, 4x, 5x, 6x, 還有 3x_bit, 4x_bit 等，下面將分別說明這些設備類型在 MODBUS 協定中支援哪些功能碼。

0x：是個可讀可寫的設備類型，相當於操作 PLC 的輸出點。該設備類型讀位元狀態的時候，發出的功能碼為 01H，寫位元狀態的時候發出的功能碼為 05H。寫多個位元暫存器時，發出的功能碼為 0FH。

1x：是個唯讀的設備類型，相當於讀 PLC 的輸入點。讀位元狀態的時候發出的功能碼為 02H。

3x：是個唯讀的設備類型，相當於讀 PLC 的唯讀資料暫存器。讀數據的時候，發出的功能碼為 04H。

4x：是個可讀可寫的設備類型，相當於操作 PLC 的資料暫存器。當讀數據的時候，發出的功能碼是 03H，當寫資料的時候發出的功能碼是 10H。

5x：該設備類型與 4x 的設備類型屬性是一樣的。即發出讀寫的功能碼完全一樣。不同之處在於，當為雙字元時，若 32_bit unsigned 格式的資料，使用 5x 和 4x 兩種設備類型分別讀取資料時，高字元和低字元的位置是顛倒的。若使用 4x 設備類型讀到的資料是 0x1234，那麼使用 5x 設備類型讀取的資料是 0x3412。

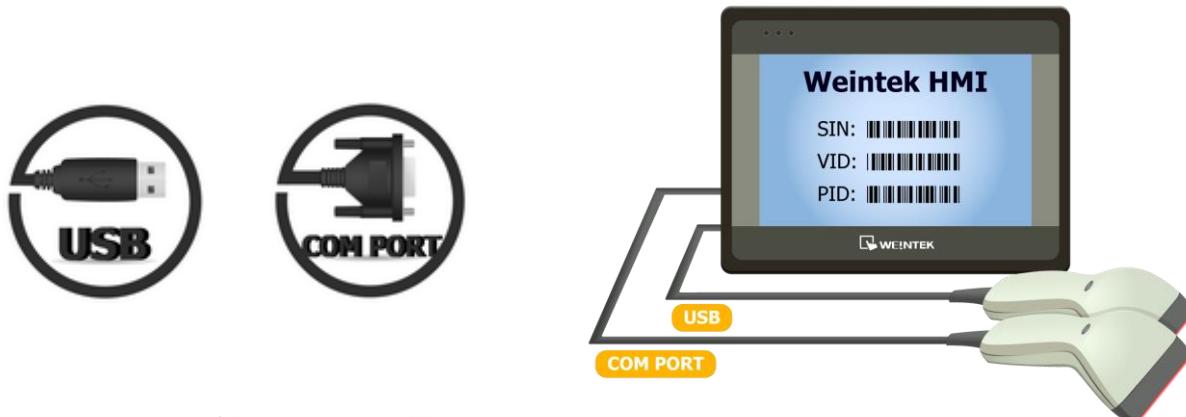
6x：是一個可讀可寫的設備類型，讀數據的時候發出的功能碼也是 03H，與 4x 不同之處在於寫資料的時候，發出的功能碼為 06H，即寫單個暫存器的資料。

3x_bit: 該設備類型支援的功能碼與 3x 設備類型完全一致，不同之處是 3x 是讀數據，而 3x_bit 是讀數據中的某一個 bit 的狀態。

4x_bit: 該設備類型支援的功能碼與 4x 設備類型完全一致，不同之處是 4x 是讀數據，而 4x_bit 是讀數據中的某一個 bit 的狀態。

第二十章 如何使用條碼掃描器裝置

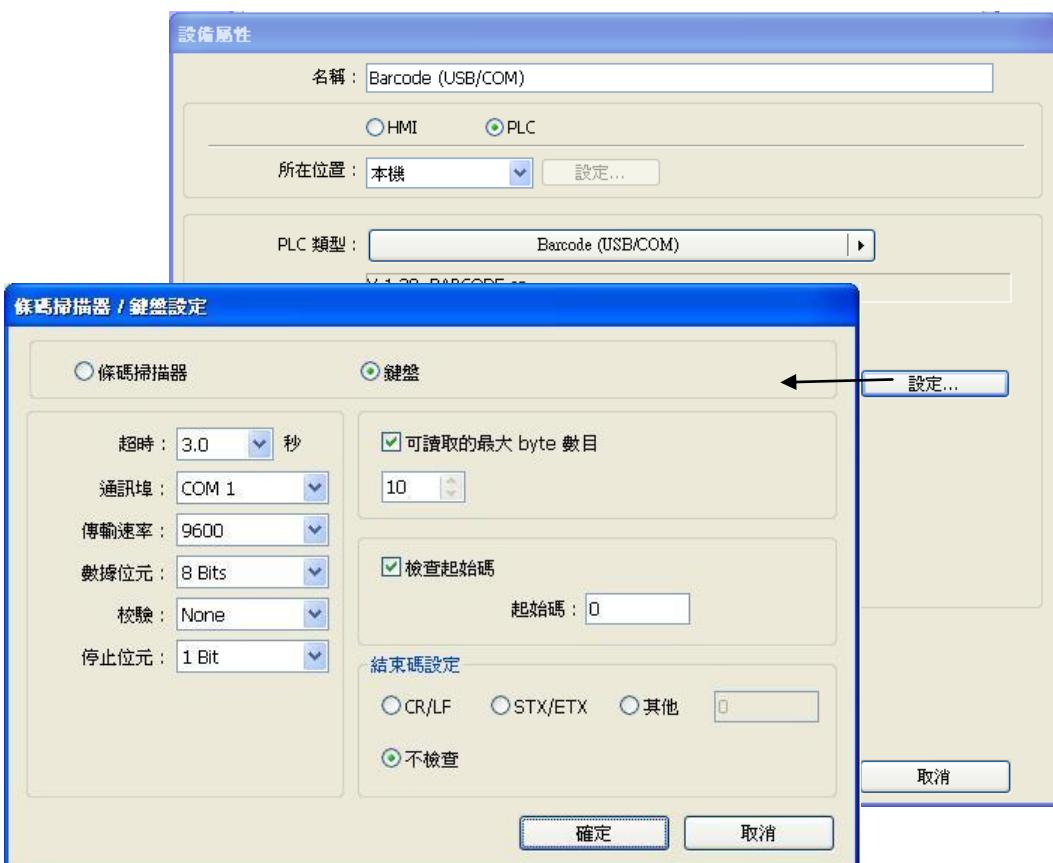
支援的條碼掃描器裝置介面：



20.1 如何使用條碼掃描器裝置

HMI 可支援連接條碼掃描器 (USB/COM) 或鍵盤裝置，首先需在【編輯功能表】»【系統參數設定】»【設備清單】頁籤中增加一個 Barcode/Keyboard (USB/COM) 驅動，如下圖所示：

按下【設定】鍵後即可設定裝置的各項屬性。





[超時]

當勾選 [鍵盤] 時，可設定鍵盤資料輸入之時間範圍，系統將於開始輸入資料時才計時。

[通訊埠]、[傳輸速率]、[數據位元]、[校驗]、[停止位元]

HMI 支援的條碼掃描器介面分為 USB 和 COM 埠，當選用 COM 介面時，須正確設定條碼掃描器的通訊參數；若選用 USB 介面，則無須設定通訊參數。

[可讀取的最大 byte 數目]

若勾選，則可以限制條碼掃描器讀取的 byte 數目，以避免裝置讀取過多的數據。此項設定值範圍為 10 ~ 512。

注意：若實際讀取的 byte 數目超過設定值，將無法讀取。

[檢查起始碼]

若勾選，則條碼掃描器所讀取到的第一個數據必須與起始碼相同，系統才會將讀取的數據視為是合法的輸入，否則將會忽略讀取的數據。

起始碼並不會被存放在條碼掃描器所對應的位址中。例如起始碼為 255 (0xff)，且讀取到的數據為

0xff 0x34 0x39 0x31 0x32 0x30 0x30 0x34 0x37

則實際存放在條碼掃描器對應的位址中的數據為

0x34 0x39 0x31 0x32 0x30 0x30 0x34 0x37

[結束碼設定]

結束碼用來標示數據的結尾，當讀取到結束碼時，表示讀取到一筆完整的數據。

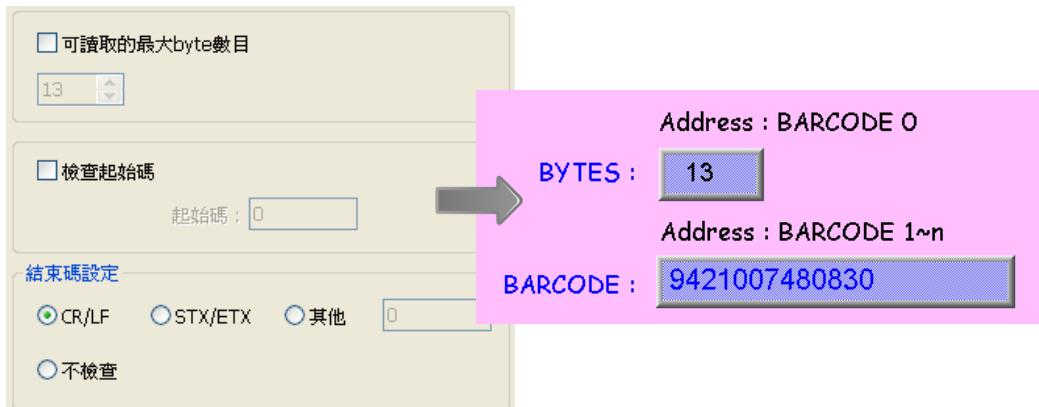
- [CR/LF]** 0x0a 或 0x0d 皆為結束碼
- [STX/ETX]** 0x02 或 0x03 皆為結束碼
- [其他]** 由使用者自訂數據的結束碼
- [不檢查]** 若選擇此項設定，HMI 會將全部讀取到的數據存放至條碼掃描器對應的位址中

完成以上各項設定後，即可在 **【設備清單】** 中完成一個新的條碼掃描器裝置。

此時在物件的設定屬性頁中的 **【PLC 名稱】** 即可選擇條碼掃描器，並可使用相關的位址類型。

位址 類型	位址名稱	用途
位元	FLAG	FLAG 0: 指示數據是否讀取完成。在讀取到數據時，系統會自動將 FLAG 先設定為 OFF，待讀取成功後再設定為 ON。
	RESET	RESET 0: 當設為 ON 時，可清除 BARCODE 和 RESULT 內的數據。
	CONNECT_STATUS	CONNECT_STATUS 0: 指示是否接上 USB 介面的條碼掃瞄器設備，當狀態為 ON 時表示已接上。
字元	BARCODE	BARCODE 0: 記錄目前讀取到的 byte 數目。 BARCODE 1 ~ n: 存放裝置讀取的數據。
	RESULT	RESULT 0: 指示 BARCODE 的讀取結果。各項數據的表示意義如下： 0x00: 等待讀取 BARCODE。 0x01: 讀取 BARCODE 成功。 0x02: BARCODE 格式錯誤。 0x03: 在啟用 [可讀取的最大 byte 數目] 時，所讀取的數據長度超過所設定的大小。 0x04: 在啟用 [檢查起始碼] 時，所讀取的數據不符合設定值。 0x05: 在啟用 [結束碼] 時，所讀取的數據不符合設定值。

假設目前條碼掃描器的設定如下圖，且讀取到的條碼為 9421007480830，圖中的數值顯示物件 (BYTES) 的位址為 BARCODE 0，字元顯示物件 (BARCODE) 的位址為 BARCODE 1 ~ n。



此時條碼掃描器裝置對應的位址所存放的數據如下：

條碼掃描器對應位址	數據
BARCODE 0	13 bytes (十進制) 但實際存入位址中的數據為 14 bytes = 7 words 也就是當讀取 byte 數目為奇數時，系統會自動加上一個 byte 的數據 (0x00)
BARCODE 1	3439 (HEX)
BARCODE 2	3132 (HEX)
BARCODE 3	3030 (HEX)
BARCODE 4	3437 (HEX)
BARCODE 5	3038 (HEX)
BARCODE 6	3338 (HEX)
BARCODE 7	0030 (HEX)



- 每台 HMI 只支援連接一台 USB 介面的條碼掃描器裝置。當工程檔案的裝置列表中包含 USB 條碼掃描器裝置時，系統暫存器 LB-9064 [啟用 USB 條碼掃描器裝置 (鍵盤功能關閉) (當狀態為 ON)] 將自動被設定為 ON。若此時需恢復 USB 鍵盤的功能並暫停使用 USB 條碼掃描器，可以將 LB-9064 設定為 OFF。

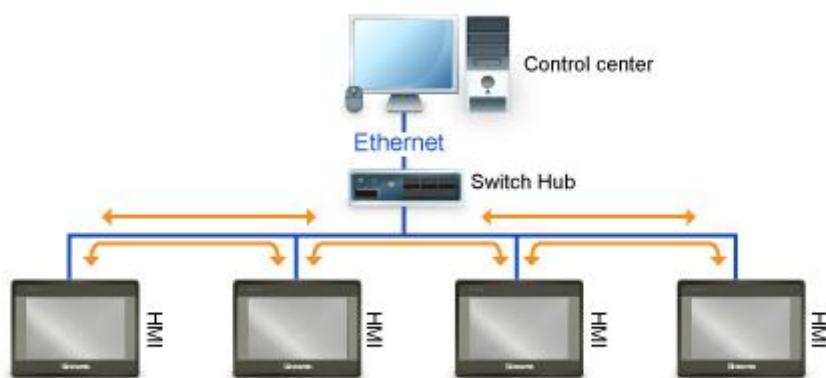


下載範例程式前，請先確定已連上網路線。

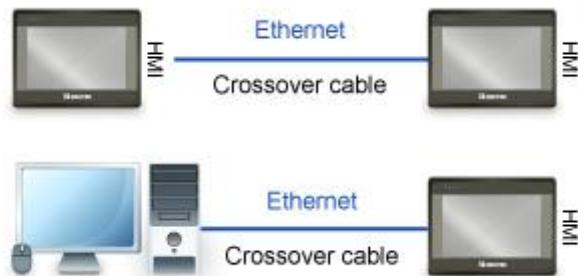
第二十一章 乙太網路通訊與多台人機連線

乙太網路連線的方式分為兩種：

1. 使用 RJ45 平行網路線 + 集線器。



2. 使用 RJ45 跳接網路線，不需使用集線器，但只限使用在一對一連線的情況下 (HMI 對 HMI，或 PC 對 HMI)。



透過乙太網路連線，系統提供了下列三種數據交換的方式：

1. HMI 與 HMI 間的通訊。
2. PC 與 HMI 間的通訊。
3. 控制連接在其他 HMI 上的 PLC。

21.1 HMI 與 HMI 間的通訊

人機之間通訊可在【系統參數設定】中新增一個遠端 HMI 裝置即可。

以兩台 HMI 的通訊為例 (HMI A 與 HMI B)，假設 HMI A 欲使用位元狀態設定物件控制 HMI B 的 [LB-0] 位址的內容：



1. 設定各台 HMI 的 IP 位址，假設 HMI A: 192.168.1.1，HMI B: 192.168.1.2。

2. HMI A 工程檔案：

【系統參數設定】»【設備清單】，新增一台遠端 HMI，即為 HMI B (IP: 192.168.1.2)。



3. HMI A 工程檔案：

設定一個位元狀態設定物件，在【PLC 名稱】中選擇“HMI B”，即可控制遠端 HMI 的位址。



- 一台 HMI 最多可同時處理來自 64 個不同 HMI 的訪問要求。

21.2 PC 與 HMI 間的通訊



透過連線模擬功能，PC 可以藉由乙太網路擷取 HMI 上的數據，並保存在 PC 上。

假設 PC 欲通訊的設備為兩台 HMI (HMI A 與 HMI B)，則 PC 端所使用工程檔案的設定步驟如下：

1. 設定各台 HMI 的 IP 位址，假設 HMI A: 192.168.1.1，HMI B: 192.168.1.2。

2. PC 端工程檔案：

[系統參數設定] » [設備清單]，新增兩台遠端 HMI，分別為 HMI A (IP:192.168.1.1)，與 HMI B (IP:192.168.1.2)。



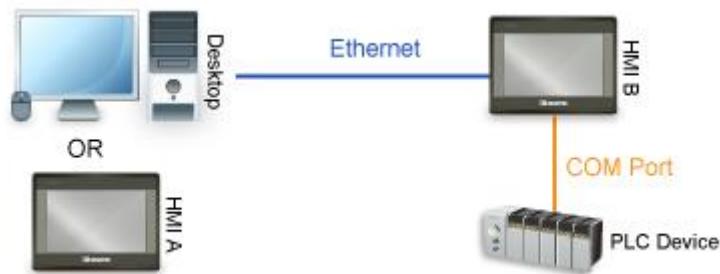
3. PC 端工程檔案：

設定一個位元狀態設定物件，在 [PLC 名稱] 中選擇 “HMI A”，即可控制遠端 HMI A 的位址。同樣可用於 HMI B。



- 一台 PC 最多可同時控制 64 台遠端 HMI。
- 如上面的例子，HMI 也允許操作 PC 上的數據，此時只需將 PC 視為另一台 HMI 即可，也就是必須在 HMI A / HMI B 使用的工程檔案中新增一台遠端 HMI，並將此遠端 HMI 的 IP 位址指向 PC。

21.3 控制連接在其他 HMI 上的 PLC



透過乙太網路連線，PC 或 HMI 可以操作連接在其他 HMI 上的遠端 PLC。

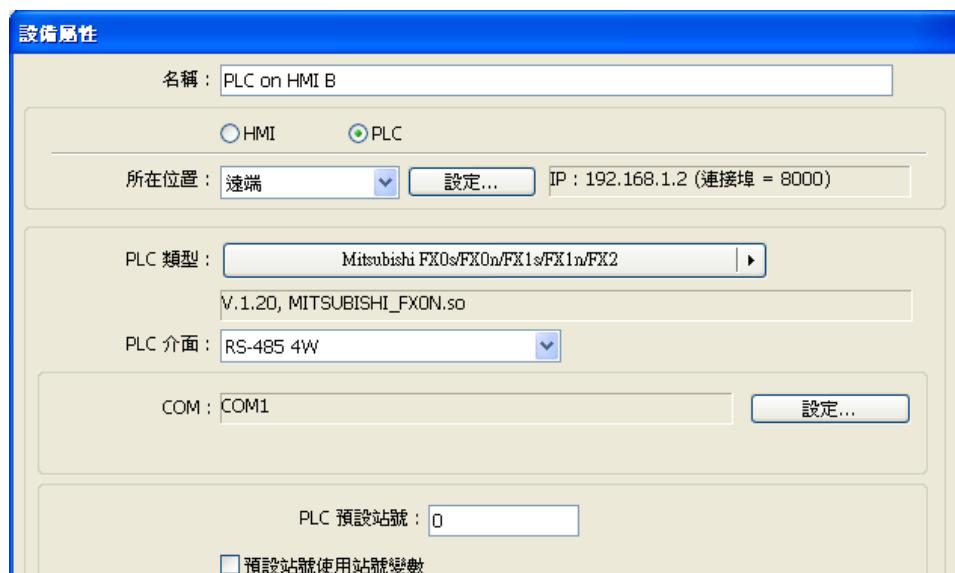
如上圖，假設現在有一台 PLC 連接到 HMI B 的 COM 1，當 PC 或 HMI A 欲讀取此台 PLC 上的數據，則 PC 端或 HMI A 上所使用工程檔案設定步驟如下：

1. 設定 HMI B 的 IP 位址，假設 HMI B: 192.168.1.2。

2. PC 或 HMI A 工程

檔案：

[系統參數設定] » [設備清單]，新增一台遠端 PLC，將名稱設為“PLC on HMI B”並正確設定 PLC 的相關通訊參數。因此台 PLC 是連接在遠端 HMI B 上，所以將遠端 IP 位址指向 HMI B (IP: 192.168.1.2)。



3. PC 或 HMI A 工程檔案：

設定一個位元狀態設定物件，在 **[PLC 名稱]** 中選擇“PLC on HMI B”，即可控制遠端 HMI B 上的 PLC。



第二十二章 位址暫存器

EasyBuilder8000 工程軟體保留了一些位元位址和字元位址的暫存器供給系統使用，這些系統保留暫存器分別有不同的功用，而我們將系統保留暫存器位址分類如下。

位址標籤庫

(使用者定義) (系統暫存器)

編號	位址標籤名稱	PLC名稱	位址...	位址	讀/寫	描述
1	LB-9000：重新開機時狀態為 ON	本機 HMI	位元	LB-9000	讀/寫	
2	LB-9001：重新開機時狀態為 ON	本機 HMI	位元	LB-9001	讀/寫	
3	LB-9002：重新開機時狀態為 ON	本機 HMI	位元	LB-9002	讀/寫	
4	LB-9003：重新開機時狀態為 ON	本機 HMI	位元	LB-9003	讀/寫	
5	LB-9004：重新開機時狀態為 ON	本機 HMI	位元	LB-9004	讀/寫	
6	LB-9005：重新開機時狀態為 ON	本機 HMI	位元	LB-9005	讀/寫	
7	LB-9006：重新開機時狀態為 ON	本機 HMI	位元	LB-9006	讀/寫	
8	LB-9007：重新開機時狀態為 ON	本機 HMI	位元	LB-9007	讀/寫	
9	LB-9008：重新開機時狀態為 ON	本機 HMI	位元	LB-9008	讀/寫	
10	LB-9009：重新開機時狀態為 ON	本機 HMI	位元	LB-9009	讀/寫	
11	LB-9010：資料下載指示	本機 HMI	位元	LB-9010	唯讀	
12	LB-9011：資料上傳指示	本機 HMI	位元	LB-9011	唯讀	
13	LB-9012：資料下載/上傳指示	本機 HMI	位元	LB-9012	唯讀	
14	LB-9013：快選視窗控制 隱藏 (ON)...	本機 HMI	位元	LB-9013	讀/寫	
15	LB-9014：快選按鍵控制 隱藏 (ON)...	本機 HMI	位元	LB-9014	讀/寫	
16	LB-9015：快選視窗/按鍵控制 隱...	本機 HMI	位元	LB-9015	讀/寫	
17	LB-9016：當遠端 HMI 連接至 HMI ...	本機 HMI	位元	LB-9016	讀/寫	
18	LB-9017：取消 PLC 控制物件切換...	本機 HMI	位元	LB-9017	讀/寫	
19	LB-9018：滑鼠游標控制 隱藏 (ON)...	本機 HMI	位元	LB-9018	讀/寫	
20	LB-9019：取消/開啟聲音輸出功能 ...	本機 HMI	位元	LB-9019	讀/寫	
21	LB-9020：系統設定列控制 顯示 (O...	本機 HMI	位元	LB-9020	讀/寫	
22	LB-9021：重置目前的事件記錄 段...	本機 HMI	位元	LB-9021	讀/寫	

* 使用者可匯入 MT500 標籤來表示所指定的位址。

新增... 刪除 刪除全部 設定...
輸出... 匯入...
輸出為 CSV... 匯入自 CSV... 輸出為 EXCEL... 匯入自 EXCEL... 關閉

22.1 本機 HMI 記憶體位址範圍

22.1.1 位元位址

暫存器	設備類型	範圍	格式
本機位元位址	LB	0 ~ 12095	DDDDDD
本機字元位址取位元位址	LW_Bit	0 ~ 1089915	DDDDDDdd DDDDD: 位址 dd: 位元位址 (00 ~ 15)
配方暫存器的位元位址索引偏移量	RBI	0 ~ 65535f	DDDDDH DDDDD: 位址 h: 位元位址 (0 ~ f) 透過 LW-9000 來當作索引暫存器，並對應到 RW_Bit
配方暫存器 RW 的位元位址	RW_Bit	0 ~ 524287f	DDDDDH DDDDD: 位址 h: 位元位址 (0 ~ f)
配方暫存器 RW_A 的位元位址	RW_A_Bit	0 ~ 65535f	DDDDDH DDDDD: 位址 h: 位元位址 (0 ~ f)

22.1.2 字元位址

暫存器	設備類型	範圍	格式
本機字元位址	LW	0 ~ 11200	DDDDDD
配方暫存器 RW	RW	0 ~ 524287	DDDDDDDD
配方暫存器 RW_A	RW_A	0 ~ 65535	DDDDDD
配方暫存器的字元位址索引偏移量	RWI	0 ~ 65535	DDDDDD 透過 LW-9000 來當作索引暫存器，並對應到 RW
擴展記憶體暫存器	EM0 ~ EM9	0 ~ 1073741823	DDDDDDDDDDDDDDDDDD

22.2 系統暫存器

22.2.1 HMI 時間

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-11958	時間設定錯誤 (當狀態為 ON) *注 3	讀	讀	讀
LW-9010	(16bit-BCD)：本地時間 (秒)	讀/寫	讀/控制	讀/控制
LW-9011	(16bit-BCD)：本地時間 (分)	讀/寫	讀/控制	讀/控制
LW-9012	(16bit-BCD)：本地時間 (時)	讀/寫	讀/控制	讀/控制
LW-9013	(16bit-BCD)：本地時間 (日)	讀/寫	讀/控制	讀/控制
LW-9014	(16bit-BCD)：本地時間 (月)	讀/寫	讀/控制	讀/控制
LW-9015	(16bit-BCD)：本地時間 (年)	讀/寫	讀/控制	讀/控制
LW-9016	(16bit-BCD)：本地時間 (星期)	讀	讀	讀
LW-9017	(16bit)：本地時間 (秒)	讀/寫	讀/控制	讀/控制
LW-9018	(16bit)：本地時間 (分)	讀/寫	讀/控制	讀/控制
LW-9019	(16bit)：本地時間 (時)	讀/寫	讀/控制	讀/控制
LW-9020	(16bit)：本地時間 (日)	讀/寫	讀/控制	讀/控制
LW-9021	(16bit)：本地時間 (月)	讀/寫	讀/控制	讀/控制
LW-9022	(16bit)：本地時間 (年) *注 1	讀/寫	讀/控制	讀/控制
LW-9023	(16bit)：本地時間 (星期) *注 2	讀	讀	讀
LW-9030	(32bit)：系統時間 (單位：0.1 秒)	讀	讀	讀
LW-9048	(16bit)：時間 (0 : AM, 1 : PM)	讀/寫	讀/控制	讀/控制
LW-9049	(16bit)：本地時間 (12 小時制)	讀/寫	讀/控制	讀/控制



- 數值範圍為 2000 ~ 2037。
- 數值範圍為 0 ~ 6，即為星期日 ~ 星期六。
- 當透過 LW-9010 ~ LW-9023 更新 HMI 時間時，系統將比對 RTC 內的時間，以確認時間是否修改成功。若不成功，系統會將時間回復為設定前的時間，並將 LB-11958 設定為 ON。若使用 LW-9010 ~ LW-9023 於 PC 模擬時修改時間，將無法作用。

22.2.2 HMI 操作

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9018	滑鼠游標控制 [隱藏 (ON)/顯示 (OFF)]	讀/寫	讀/控制	讀/控制
LB-9019	聲音輸出功能 [取消 (ON)/開啟 (OFF)]	讀/寫	讀/控制	讀/控制
LB-9020	系統設定列控制 [顯示 (ON)/隱藏 (OFF)]	讀/寫	讀/控制	讀/控制
LB-9033	HMI 上傳功能 [取消 (ON)/開啟 (OFF)] (只支援 i 系列) *注 1	讀/寫	讀/控制	讀
LB-9040	背光燈調亮 (設定為 ON) *注 2	寫	控制	控制
LB-9041	背光燈調暗 (設定為 ON) *注 2	寫	控制	控制
LB-9047	重新啟動 HMI (設定為 ON, 並當 LB-9048 狀態為 ON 時)	寫	控制	控制
LB-9048	重啟機制保護	讀/寫	讀/控制	讀/控制
LB-9062	開啟硬體設定對話窗 (設定為 ON)	寫	控制	控制
LB-9063	當插上 USB 碟時, 彈出下載視窗控制 [隱藏 (ON)/顯示 (OFF)] (只支援 i 系列)	讀/寫	讀/控制	讀/控制
LB-9064	啟用 USB 條碼掃瞄器裝置 (鍵盤功能關閉) (當狀態為 ON) *注 4	讀/寫	讀/控制	讀
LB-12024	取消報警聲 (設定為 ON)	讀/寫	讀/控制	讀/控制
LB-12051	蜂鳴器狀態 (當狀態為 ON 時表示啟動)	讀/寫	讀/控制	讀/控制
LW-9008	(32bit-float) : 電池電壓 (只支援 i 系列) *注 3	讀	讀	讀
LW-9025	(16bit) : CPU 使用率	讀	讀	讀
LW-9026	(16bit) : OS 版本 (年)	讀	讀	讀
LW-9027	(16bit) : OS 版本 (月)	讀	讀	讀
LW-9028	(16bit) : OS 版本 (日)	讀	讀	讀
LW-9040	(16bit) : 背光燈亮度值 *注 2	讀	讀	讀
LW-9080	(16bit) : 背光節能時間 (單位：分鐘)	讀/寫	讀/控制	讀/控制
LW-9081	(16bit) : 螢幕保護時間 (單位：分鐘)	讀/寫	讀/控制	讀/控制
LW-9350	(16bit) : 本機尚未處理的命令數目	讀	讀	讀
LW-10884	(16 words) : HMI 名稱	讀/寫	讀/控制	讀/控制



1. 當變更設定時，需重啟 HMI 以更新設定值。

2. LW-9040 可搭配 LB-9040 ~ LB-9041 來調整背光亮度，亮度值為 0 ~ 31。
3. 建議當 LW-9008 的電池電壓值少於 2.80V 時，可更換電池。
4. LB-9064 啟用 USB 條碼掃瞄器裝置（鍵盤功能關閉）範例如下：



下載範例程式前，請先確定已連上網路線。

22.2.3 觸碰位置

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LW-9041	(16bit) : 觸控狀態 (位元 0 on = 正在觸碰螢幕)	讀	讀	讀
LW-9042	(16bit) : 觸碰時, X 的位置	讀	讀	讀
LW-9043	(16bit) : 觸碰時, Y 的位置	讀	讀	讀
LW-9044	(16bit) : 離開時, X 的位置	讀	讀	讀
LW-9045	(16bit) : 離開時, Y 的位置	讀	讀	讀

想知道如何透過觸碰位置相關的暫存器來達到隱密性的換頁動作嗎？



下載範例程式前，請先確定已連上網路線。

22.2.4 本地 HMI 網路資訊

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-12041	刷新 HMI IP 資訊 (LW-9129~9132) (設定為 ON)	讀/寫	讀/控制	讀/控制
LW-9125	(16bit) : HMI 乙太網路所使用的閘道 0 (只在 HMI 上有效)	讀/寫	讀/控制	讀/控制
LW-9126	(16bit) : HMI 乙太網路所使用的閘道 1 (只在 HMI 上有效)	讀/寫	讀/控制	讀/控制
LW-9127	(16bit) : HMI 乙太網路所使用的閘道 2 (只在 HMI 上有效)	讀/寫	讀/控制	讀/控制
LW-9128	(16bit) : HMI 乙太網路所使用的閘道 3 (只在 HMI 上有效)	讀/寫	讀/控制	讀/控制
LW-9129	(16bit) : HMI 乙太網路所使用的 IP 0 (只在 HMI 上有效)	讀/寫	讀/控制	讀/控制
LW-9130	(16bit) : HMI 乙太網路所使用的 IP 1 (只在 HMI 上有效)	讀/寫	讀/控制	讀/控制
LW-9131	(16bit) : HMI 乙太網路所使用的 IP 2 (只在 HMI 上有效)	讀/寫	讀/控制	讀/控制
LW-9132	(16bit) : HMI 乙太網路所使用的 IP 3 (只在 HMI 上有效)	讀/寫	讀/控制	讀/控制
LW-9133	(16bit) : 乙太網路所使用的連接埠	讀	讀	讀
LW-9135	(16bit) : 實體位址 (MAC) 0	讀	讀	讀
LW-9136	(16bit) : 實體位址 (MAC) 1	讀	讀	讀
LW-9137	(16bit) : 實體位址 (MAC) 2	讀	讀	讀
LW-9138	(16bit) : 實體位址 (MAC) 3	讀	讀	讀
LW-9139	(16bit) : 實體位址 (MAC) 4	讀	讀	讀
LW-9140	(16bit) : 實體位址 (MAC) 5	讀	讀	讀
LW-9141	(16bit) : HMI 站號	讀/寫	讀/控制	讀/控制
LW-10750	(16bit) : HMI 乙太網路所使用的遮罩 0 (只在 HMI 上有效)	讀/寫	讀/控制	讀/控制
LW-10751	(16bit) : HMI 乙太網路所使用的遮罩 1 (只在 HMI 上有效)	讀/寫	讀/控制	讀/控制
LW-10752	(16bit) : HMI 乙太網路所使用的遮罩 2 (只在 HMI 上有效)	讀/寫	讀/控制	讀/控制
LW-10753	(16bit) : HMI 乙太網路所使用的遮罩 3 (只在 HMI 上有效)	讀/寫	讀/控制	讀/控制
LW-10812	(16bit) : 自動分配 IP 位址 (DHCP => 0 : off, 1 : on)	讀/寫	讀/控制	讀/控制

22.2.5 工程檔案資訊

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LW-9100	(16bit) : 工程檔案的名稱 (16 字元)	讀	讀	讀
LW-9116	(32bit) : 工程檔案的大小 (單位 : byte)	讀	讀	讀
LW-9118	(32bit) : 工程檔案的大小 (單位 : K bytes)	讀	讀	讀
LW-9120	(32bit) : EasyBuilder8000 版本	讀	讀	讀
LW-9122	(16bit) : 工程檔案編譯時間 (年)	讀	讀	讀
LW-9123	(16bit) : 工程檔案編譯時間 (月)	讀	讀	讀
LW-9124	(16bit) : 工程檔案編譯時間 (日)	讀	讀	讀

22.2.6 儲存空間管理

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9035	HMI 剩餘空間不足警示 (當狀態為 ON)	讀	讀	讀
LB-9036	SD 卡剩餘空間不足警示 (當狀態為 ON)	讀	讀	讀
LB-9037	USB 碟 1 剩餘空間不足警示 (當狀態為 ON)	讀	讀	讀
LB-9038	USB 碟 2 剩餘空間不足警示 (當狀態為 ON)	讀	讀	讀
LB-12048	USB 碟 1 狀態 (當狀態為 ON 時表示存在)	讀	讀	讀
LB-12049	USB 碟 2 狀態 (當狀態為 ON 時表示存在)	讀	讀	讀
LB-12050	SD 卡狀態 (當狀態為 ON 時表示存在)	讀	讀	讀
LW-9070	(16bit) : 剩餘空間警示下限 (Mega bytes)	讀	讀	讀
LW-9071	(16bit) : 系統保留的剩餘空間 (Mega bytes)	讀	讀	讀
LW-9072	(32bit) : HMI 目前的剩餘空間 (K bytes)	讀	讀	讀
LW-9074	(32bit) : SD 卡目前的剩餘空間 (K bytes)	讀	讀	讀
LW-9076	(32bit) : USB 碟 1 目前的剩餘空間 (K bytes)	讀	讀	讀
LW-9078	(32bit) : USB 碟 2 目前的剩餘空間 (K bytes)	讀	讀	讀

想知道如何使用 LW-9072 ~ LW-9078 來搭配備份物件的應用嗎？



下載範例程式前，請先確定已連上網路線。

22.2.7 配方及擴展記憶體

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9028	重置配方資料 (設定為 ON)	寫	控制	控制
LB-9029	儲存配方資料到 HMI (設定為 ON)	寫	控制	控制
LB-9460	EM0 的儲存裝置 (SD 卡) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9461	EM1 的儲存裝置 (SD 卡) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9462	EM2 的儲存裝置 (SD 卡) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9463	EM3 的儲存裝置 (SD 卡) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9464	EM4 的儲存裝置 (SD 卡) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9465	EM5 的儲存裝置 (SD 卡) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9466	EM6 的儲存裝置 (SD 卡) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9467	EM7 的儲存裝置 (SD 卡) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9468	EM8 的儲存裝置 (SD 卡) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9469	EM9 的儲存裝置 (SD 卡) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9470	EM0 的儲存裝置 (USB 碟 1) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9471	EM1 的儲存裝置 (USB 碟 1) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9472	EM2 的儲存裝置 (USB 碟 1) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9473	EM3 的儲存裝置 (USB 碟 1) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9474	EM4 的儲存裝置 (USB 碟 1) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9475	EM5 的儲存裝置 (USB 碟 1) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9476	EM6 的儲存裝置 (USB 碟 1) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9477	EM7 的儲存裝置 (USB 碟 1) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9478	EM8 的儲存裝置 (USB 碟 1) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9479	EM9 的儲存裝置 (USB 碟 1) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9480	EM0 的儲存裝置 (USB 碟 2) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9481	EM1 的儲存裝置 (USB 碟 2) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9482	EM2 的儲存裝置 (USB 碟 2) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9483	EM3 的儲存裝置 (USB 碟 2) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9484	EM4 的儲存裝置 (USB 碟 2) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9485	EM5 的儲存裝置 (USB 碟 2) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9486	EM6 的儲存裝置 (USB 碟 2) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9487	EM7 的儲存裝置 (USB 碟 2) 不存在 (當狀態為 ON)	讀	讀	讀
LB-9488	EM8 的儲存裝置 (USB 碟 2) 不存在 (當狀態為 ON)	讀	讀	讀

LB-9489	EM9 的儲存裝置 (USB 碟 2) 不存在 (當狀態為 ON)	讀	讀	讀
---------	-----------------------------------	---	---	---

22.2.8 資料取樣

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9025	刪除 HMI 記憶體裡日期最早的資料取樣檔案 (設定為 ON)	寫	控制	控制
LB-9026	刪除 HMI 記憶體裡全部資料取樣檔案 (設定為 ON)	寫	控制	控制
LB-9027	更新 HMI 記憶體裡資料取樣統計資訊 (設定為 ON)	寫	控制	控制
LB-9034	儲存事件記錄與取樣數據至 HMI, USB 碟, SD 卡 (設定為 ON)	寫	控制	控制
LB-11949	刪除 SD 卡裡日期最早的資料取樣檔案 (設定為 ON)	寫	控制	控制
LB-11950	刪除 SD 卡裡全部資料取樣檔案 (設定為 ON)	寫	控制	控制
LB-11951	更新 SD 卡裡資料取樣統計資訊 (設定為 ON)	寫	控制	控制
LB-11952	刪除 USB 碟 1 裡日期最早的資料取樣檔案 (設定為 ON)	寫	控制	控制
LB-11953	刪除 USB 碟 1 裡全部資料取樣檔案 (設定為 ON)	寫	控制	控制
LB-11954	更新 USB 碟 1 裡資料取樣統計資訊 (設定為 ON)	寫	控制	控制
LB-11955	刪除 USB 碟 2 裡日期最早的資料取樣檔案 (設定為 ON)	寫	控制	控制
LB-11956	刪除 USB 碟 2 裡全部資料取樣檔案 (設定為 ON)	寫	控制	控制
LB-11957	更新 USB 碟 2 裡資料取樣統計資訊 (設定為 ON)	寫	控制	控制
LW-9063	(16bit) : HMI 記憶體裡存在的資料取樣檔案數目	讀	讀	讀
LW-9064	(32bit) : HMI 記憶體裡存在的資料取樣檔案大小	讀	讀	讀
LW-10489	(16bit) : SD 卡裡存在的資料取樣檔案數目	讀	讀	讀
LW-10490	(32bit) : SD 卡裡存在的資料取樣檔案大小	讀	讀	讀
LW-10492	(16bit) : USB 碟 1 裡存在的資料取樣檔案數目	讀	讀	讀
LW-10493	(32bit) : USB 碟 1 裡存在的資料取樣檔案大小	讀	讀	讀
LW-10495	(16bit) : USB 碟 2 裡存在的資料取樣檔案數目	讀	讀	讀
LW-10496	(32bit) : USB 碟 2 裡存在的資料取樣檔案大小	讀	讀	讀



- 刪除或更新資料取樣的相關暫存器，於 PC 模擬時皆無作用。

22.2.9 事件登錄

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9021	重置目前的事件記錄 (OFF->ON)	寫	控制	控制
LB-9022	刪除 HMI 記憶體裡日期最早事件記錄檔案 (設定為 ON)	寫	控制	控制
LB-9023	刪除 HMI 記憶體裡全部事件記錄檔案 (設定為 ON)	寫	控制	控制
LB-9024	更新 HMI 記憶體裡事件記錄統計資訊 (設定為 ON)	寫	控制	控制
LB-9034	儲存事件記錄與取樣數據至 HMI, USB 碟, SD 卡 (設定為 ON)	寫	控制	控制
LB-9042	確認全部事件 (設定為 ON)	寫	控制	控制
LB-9043	存在未確認的事件 (當狀態為 ON)	讀	讀	讀
LB-11940	刪除 SD 卡裡日期最早事件記錄檔案 (設定為 ON)	寫	控制	控制
LB-11941	刪除 SD 卡裡全部事件記錄檔案 (設定為 ON)	寫	控制	控制
LB-11942	更新 SD 卡裡事件記錄統計資訊 (設定為 ON)	寫	控制	控制
LB-11943	刪除 USB 碟 1 裡日期最早事件記錄檔案 (設定為 ON)	寫	控制	控制
LB-11944	刪除 USB 碟 1 裡全部事件記錄檔案 (設定為 ON)	寫	控制	控制
LB-11945	更新 USB 碟 1 裡事件記錄統計資訊 (設定為 ON)	寫	控制	控制
LB-11946	刪除 USB 碟 2 裡日期最早事件記錄檔案 (設定為 ON)	寫	控制	控制
LB-11947	刪除 USB 碟 2 裡全部事件記錄檔案 (設定為 ON)	寫	控制	控制
LB-11948	更新 USB 碟 2 裡事件記錄統計資訊 (設定為 ON)	寫	控制	控制
LW-9060	(16bit) : HMI 記憶體裡存在的事件記錄檔案數目	讀	讀	讀
LW-9061	(32bit) : HMI 記憶體裡存在的事件記錄檔案大小	讀	讀	讀
LW-9450	(16bit) : 事件登錄的時間標籤 - 秒 *注 1	讀/寫	讀/控制	讀/控制
LW-9451	(16bit) : 事件登錄的時間標籤 - 分 *注 1	讀/寫	讀/控制	讀/控制
LW-9452	(16bit) : 事件登錄的時間標籤 - 時 *注 1	讀/寫	讀/控制	讀/控制
LW-9453	(16bit) : 事件登錄的時間標籤 - 日 *注 1	讀/寫	讀/控制	讀/控制
LW-9454	(16bit) : 事件登錄的時間標籤 - 月 *注 1	讀/寫	讀/控制	讀/控制
LW-9455	(16bit) : 事件登錄的時間標籤 - 年 *注 1	讀/寫	讀/控制	讀/控制
LW-10480	(16bit) : SD 卡裡存在的事件記錄檔案數目	讀	讀	讀
LW-10481	(32bit) : SD 卡裡存在的事件記錄檔案大小	讀	讀	讀
LW-10483	(16bit) : USB 碟 1 裡存在的事件記錄檔案數目	讀	讀	讀

LW-10484	(32bit) : USB 碟 1 裡存在的事件記錄檔案大小	讀	讀	讀
LW-10486	(16bit) : USB 碟 2 裡存在的事件記錄檔案數目	讀	讀	讀
LW-10487	(32bit) : USB 碟 2 裡存在的事件記錄檔案大小	讀	讀	讀



1. 若欲使用 LW-9450 ~ LW-9455 作為事件登錄的時間來源標籤，請先在 **【系統參數設定】>>【一般屬性】** 頁籤設定相關屬性。
2. 刪除或更新事件記錄的相關暫存器，於 PC 模擬時皆無作用。

使用 LW-9450 ~ LW-9455 作為事件登錄的時間來源標籤範例如下：



下載範例程式前，請先確定已連上網路線。

22.2.10 站號變數

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LW-10000	(16bit) : var0 - 站號變數 (語法 : var0#位址)	讀/寫	讀/控制	讀/控制
LW-10001	(16bit) : var1 - 站號變數 (語法 : var1#位址)	讀/寫	讀/控制	讀/控制
LW-10002	(16bit) : var2 - 站號變數 (語法 : var2#位址)	讀/寫	讀/控制	讀/控制
LW-10003	(16bit) : var3 - 站號變數 (語法 : var3#位址)	讀/寫	讀/控制	讀/控制
LW-10004	(16bit) : var4 - 站號變數 (語法 : var4#位址)	讀/寫	讀/控制	讀/控制
LW-10005	(16bit) : var5 - 站號變數 (語法 : var5#位址)	讀/寫	讀/控制	讀/控制
LW-10006	(16bit) : var6 - 站號變數 (語法 : var6#位址)	讀/寫	讀/控制	讀/控制
LW-10007	(16bit) : var7 - 站號變數 (語法 : var7#位址)	讀/寫	讀/控制	讀/控制
LW-10008	(16bit) : var8 - 站號變數 (語法 : var8#位址)	讀/寫	讀/控制	讀/控制
LW-10009	(16bit) : var9 - 站號變數 (語法 : var9#位址)	讀/寫	讀/控制	讀/控制
LW-10010	(16bit) : var10 - 站號變數 (語法 : var10#位址)	讀/寫	讀/控制	讀/控制
LW-10011	(16bit) : var11 - 站號變數 (語法 : var11#位址)	讀/寫	讀/控制	讀/控制
LW-10012	(16bit) : var12 - 站號變數 (語法 : var12#位址)	讀/寫	讀/控制	讀/控制
LW-10013	(16bit) : var13 - 站號變數 (語法 : var13#位址)	讀/寫	讀/控制	讀/控制
LW-10014	(16bit) : var14 - 站號變數 (語法 : var14#位址)	讀/寫	讀/控制	讀/控制
LW-10015	(16bit) : var15 - 站號變數 (語法 : var15#位址)	讀/寫	讀/控制	讀/控制



下載範例程式前，請先確定已連上網路線。

22.2.11 索引暫存器

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LW-9200	(16bit) : 位址索引暫存器 0	讀/寫	讀/控制	讀/控制
LW-9201	(16bit) : 位址索引暫存器 1	讀/寫	讀/控制	讀/控制
LW-9202	(16bit) : 位址索引暫存器 2	讀/寫	讀/控制	讀/控制
LW-9203	(16bit) : 位址索引暫存器 3	讀/寫	讀/控制	讀/控制
LW-9204	(16bit) : 位址索引暫存器 4	讀/寫	讀/控制	讀/控制
LW-9205	(16bit) : 位址索引暫存器 5	讀/寫	讀/控制	讀/控制
LW-9206	(16bit) : 位址索引暫存器 6	讀/寫	讀/控制	讀/控制
LW-9207	(16bit) : 位址索引暫存器 7	讀/寫	讀/控制	讀/控制
LW-9208	(16bit) : 位址索引暫存器 8	讀/寫	讀/控制	讀/控制
LW-9209	(16bit) : 位址索引暫存器 9	讀/寫	讀/控制	讀/控制
LW-9210	(16bit) : 位址索引暫存器 10	讀/寫	讀/控制	讀/控制
LW-9211	(16bit) : 位址索引暫存器 11	讀/寫	讀/控制	讀/控制
LW-9212	(16bit) : 位址索引暫存器 12	讀/寫	讀/控制	讀/控制
LW-9213	(16bit) : 位址索引暫存器 13	讀/寫	讀/控制	讀/控制
LW-9214	(16bit) : 位址索引暫存器 14	讀/寫	讀/控制	讀/控制
LW-9215	(16bit) : 位址索引暫存器 15	讀/寫	讀/控制	讀/控制
LW-9230	(32bit) : 位址索引暫存器 16	讀/寫	讀/控制	讀/控制
LW-9232	(32bit) : 位址索引暫存器 17	讀/寫	讀/控制	讀/控制
LW-9234	(32bit) : 位址索引暫存器 18	讀/寫	讀/控制	讀/控制
LW-9236	(32bit) : 位址索引暫存器 19	讀/寫	讀/控制	讀/控制
LW-9238	(32bit) : 位址索引暫存器 20	讀/寫	讀/控制	讀/控制
LW-9240	(32bit) : 位址索引暫存器 21	讀/寫	讀/控制	讀/控制
LW-9242	(32bit) : 位址索引暫存器 22	讀/寫	讀/控制	讀/控制
LW-9244	(32bit) : 位址索引暫存器 23	讀/寫	讀/控制	讀/控制
LW-9246	(32bit) : 位址索引暫存器 24	讀/寫	讀/控制	讀/控制
LW-9248	(32bit) : 位址索引暫存器 25	讀/寫	讀/控制	讀/控制
LW-9250	(32bit) : 位址索引暫存器 26	讀/寫	讀/控制	讀/控制
LW-9252	(32bit) : 位址索引暫存器 27	讀/寫	讀/控制	讀/控制
LW-9254	(32bit) : 位址索引暫存器 28	讀/寫	讀/控制	讀/控制
LW-9256	(32bit) : 位址索引暫存器 29	讀/寫	讀/控制	讀/控制

LW-9258	(32bit) : 位址索引暫存器 30	讀/寫	讀/控制	讀/控制
LW-9260	(32bit) : 位址索引暫存器 31	讀/寫	讀/控制	讀/控制



下載範例程式前，請先確定已連上網路線。

22.2.12 MODBUS Server 通訊

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9055	MODBUS server (COM 1) 接收到合法的命令 (當狀態為 ON)	讀	讀	讀
LB-9056	MODBUS server (COM 2) 接收到合法的命令 (當狀態為 ON)	讀	讀	讀
LB-9057	MODBUS server (COM 3) 接收到合法的命令 (當狀態為 ON)	讀	讀	讀
LB-9058	MODBUS server (乙太網路) 接收到合法的命令 (當狀態為 ON)	讀	讀	讀
LB-12052	MODBUS server 狀態 (當狀態為 ON 時關閉 server 功能)	讀/寫	讀/控制	讀/控制
LW-9270	(16bit) : 請求的功能碼 - MODBUS server (COM 1)	讀	讀	讀
LW-9271	(16bit) : 請求的開始位址 - MODBUS server (COM 1)	讀	讀	讀
LW-9272	(16bit) : 請求的位址數目 - MODBUS server (COM 1)	讀	讀	讀
LW-9275	(16bit) : 請求的功能碼 - MODBUS server (COM 2)	讀	讀	讀
LW-9276	(16bit) : 請求的開始位址 - MODBUS server (COM 2)	讀	讀	讀
LW-9277	(16bit) : 請求的位址數目 - MODBUS server (COM 2)	讀	讀	讀
LW-9280	(16bit) : 請求的功能碼 - MODBUS server (COM 3)	讀	讀	讀
LW-9281	(16bit) : 請求的開始位址 - MODBUS server (COM 3)	讀	讀	讀
LW-9282	(16bit) : 請求的位址數目 - MODBUS server (COM 3)	讀	讀	讀
LW-9285	(16bit) : 請求的功能碼 - MODBUS server (乙太網路)	讀	讀	讀
LW-9286	(16bit) : 請求的開始位址 - MODBUS server (乙太網路)	讀	讀	讀
LW-9287	(16bit) : 請求的位址數目 - MODBUS server (乙太網路)	讀	讀	讀
LW-9288	(16bit) : 最後通訊錯誤碼 - MODBUS server (乙太網路)	讀	讀	讀

LW-9541	(16bit) : MODBUS/ASCII server 站號 (COM 1)	讀/寫	讀/控制	讀/控制
LW-9542	(16bit) : MODBUS/ASCII server 站號 (COM 2)	讀/寫	讀/控制	讀/控制
LW-9543	(16bit) : MODBUS/ASCII server 站號 (COM 3)	讀/寫	讀/控制	讀/控制
LW-9544	(16bit) : MODBUS/ASCII server 站號 (乙太網路)	讀/寫	讀/控制	讀/控制
LW-9570	(32bit) : 已接收的數據 (bytes) (COM 1 MODBUS server)	讀	讀	讀
LW-9572	(32bit) : 已接收的數據 (bytes) (COM 2 MODBUS server)	讀	讀	讀
LW-9574	(32bit) : 已接收的數據 (bytes) (COM 3 MODBUS server)	讀	讀	讀
LW-9576	(32bit) : 已接收的數據 (bytes) (乙太網路 MODBUS server)	讀	讀	讀

22.2.13 通訊參數設定

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9030	更新 COM 1 通訊參數 (設定為 ON)	讀/寫	讀/控制	讀/控制
LB-9031	更新 COM 2 通訊參數 (設定為 ON)	讀/寫	讀/控制	讀/控制
LB-9032	更新 COM 3 通訊參數 (設定為 ON)	讀/寫	讀/控制	讀/控制
LB-9065	停用/啟用 COM 1 廣播站號	讀/寫	讀/控制	讀/控制
LB-9066	停用/啟用 COM 2 廣播站號	讀/寫	讀/控制	讀/控制
LB-9067	停用/啟用 COM 3 廣播站號	讀/寫	讀/控制	讀/控制
LW-9550	(16bit) : COM 1 模式 (0:RS232,1:RS485 2W,2:RS485 4W)	讀/寫	讀/控制	讀/控制
LW-9551	(16bit) : COM 1 串列傳輸速率 (7:1200,8:2400,0:4800,1:9600,10:14400,2:19200,11:28800,3:38400,4:57600,..)	讀/寫	讀/控制	讀/控制
LW-9552	(16bit) : COM 1 數據位元 (7 : 7 bits, 8 : 8 bits)	讀/寫	讀/控制	讀/控制
LW-9553	(16bit) : COM 1 校驗 (0:none, 1:even, 2:odd, 3:mark, 4:space)	讀/寫	讀/控制	讀/控制
LW-9554	(16bit) : COM 1 停止位元 (1 : 1 bit, 2 : 2 bits)	讀/寫	讀/控制	讀/控制
LW-9555	(16bit) : COM 2 模式 (0:RS232,1:RS485 2W,2:RS485 4W)	讀/寫	讀/控制	讀/控制
LW-9556	(16bit) : COM 2 串列傳輸速率	讀/寫	讀/控制	讀/控制

	(7:1200,8:2400,0:4800,1:9600,10:14400,2:19200,11:2 8800,3:38400,4:57600,...)			
LW-9557	(16bit) : COM 2 數據位元 (7 : 7 bits, 8 : 8 bits)	讀/寫	讀/控制	讀/控制
LW-9558	(16bit) : COM 2 校驗 (0:none, 1:even, 2:odd, 3:mark, 4:space)	讀/寫	讀/控制	讀/控制
LW-9559	(16bit) : COM 2 停止位元 (1 : 1 bit, 2 : 2 bits)	讀/寫	讀/控制	讀/控制
LW-9560	(16bit) : COM 3 模式 (0:RS232, 1:RS485 2W)	讀/寫	讀/控制	讀/控制
LW-9561	(16bit) : COM 3 串列傳輸速率 (7:1200,8:2400,0:4800,1:9600,10:14400,2:19200,11:2 8800,3:38400,4:57600,...)	讀/寫	讀/控制	讀/控制
LW-9562	(16bit) : COM 3 數據位元 (7 : 7 bits, 8 : 8 bits)	讀/寫	讀/控制	讀/控制
LW-9563	(16bit) : COM 3 校驗 (0:none, 1:even, 2:odd, 3:mark, 4:space)	讀/寫	讀/控制	讀/控制
LW-9564	(16bit) : COM 3 停止位元 (1 : 1 bit, 2 : 2 bits)	讀/寫	讀/控制	讀/控制
LW-9565	(16bit) : COM 1 廣播站號	讀/寫	讀/控制	讀/控制
LW-9566	(16bit) : COM 2 廣播站號	讀/寫	讀/控制	讀/控制
LW-9567	(16bit) : COM 3 廣播站號	讀/寫	讀/控制	讀/控制
LW-10500	(16bit) : PLC 1 超時 (單位 : 100ms)	讀/寫	讀/控制	讀/控制
LW-10501	(16bit) : PLC 1 通訊延時 (單位 : ms)	讀/寫	讀/控制	讀/控制
LW-10502	(16bit) : PLC 1 ACK 訊號延時 (單位 : ms)	讀/寫	讀/控制	讀/控制
LW-10503	(16bit) : PLC 1 參數 1	讀/寫	讀/控制	讀/控制
LW-10504	(16bit) : PLC 1 參數 2	讀/寫	讀/控制	讀/控制
LW-10505	(16bit) : PLC 2 超時 (單位 : 100ms)	讀/寫	讀/控制	讀/控制
LW-10506	(16bit) : PLC 2 通訊延時 (單位 : ms)	讀/寫	讀/控制	讀/控制
LW-10507	(16bit) : PLC 2 ACK 訊號延時 (單位 : ms)	讀/寫	讀/控制	讀/控制
LW-10508	(16bit) : PLC 2 參數 1	讀/寫	讀/控制	讀/控制
LW-10509	(16bit) : PLC 2 參數 2	讀/寫	讀/控制	讀/控制
LW-10510	(16bit) : PLC 3 超時 (單位 : 100ms)	讀/寫	讀/控制	讀/控制
LW-10511	(16bit) : PLC 3 通訊延時 (單位 : ms)	讀/寫	讀/控制	讀/控制
LW-10512	(16bit) : PLC 3 ACK 訊號延時 (單位 : ms)	讀/寫	讀/控制	讀/控制
LW-10513	(16bit) : PLC 3 參數 1	讀/寫	讀/控制	讀/控制
LW-10514	(16bit) : PLC 3 參數 2	讀/寫	讀/控制	讀/控制
LW-10515	(16bit) : PLC 4 超時 (單位 : 100ms)	讀/寫	讀/控制	讀/控制
LW-10516	(16bit) : PLC 4 通訊延時 (單位 : ms)	讀/寫	讀/控制	讀/控制
LW-10517	(16bit) : PLC 4 ACK 訊號延時 (單位 : ms) (SIEMENS S7/400 連接類型)	讀/寫	讀/控制	讀/控制
LW-10518	(16bit) : PLC 4 參數 1 (SIEMENS S7/400 機座)	讀/寫	讀/控制	讀/控制

LW-10519	(16bit) : PLC 4 參數 2 (SIEMENS S7/400 CPU 插槽)	讀/寫	讀/控制	讀/控制
LW-10520	(16bit) : PLC 5 超時 (單位 : 100ms)	讀/寫	讀/控制	讀/控制
LW-10521	(16bit) : PLC 5 通訊延時 (單位 : ms)	讀/寫	讀/控制	讀/控制
LW-10522	(16bit) : PLC 5 ACK 訊號延時 (單位 : ms) (SIEMENS S7/400 連接類型)	讀/寫	讀/控制	讀/控制
LW-10523	(16bit) : PLC 5 參數 1 (SIEMENS S7/400 機座)	讀/寫	讀/控制	讀/控制
LW-10524	(16bit) : PLC 5 參數 2 (SIEMENS S7/400 CPU 插槽)	讀/寫	讀/控制	讀/控制
LW-10525	(16bit) : PLC 6 超時 (單位 : 100ms)	讀/寫	讀/控制	讀/控制
LW-10526	(16bit) : PLC 6 通訊延時 (單位 : ms)	讀/寫	讀/控制	讀/控制
LW-10527	(16bit) : PLC 6 ACK 訊號延時 (單位 : ms) (SIEMENS S7/400 連接類型)	讀/寫	讀/控制	讀/控制
LW-10528	(16bit) : PLC 6 參數 1 (SIEMENS S7/400 機座)	讀/寫	讀/控制	讀/控制
LW-10529	(16bit) : PLC 6 參數 2 (SIEMENS S7/400 CPU 插槽)	讀/寫	讀/控制	讀/控制
LW-10530	(16bit) : PLC 7 超時 (單位 : 100ms)	讀/寫	讀/控制	讀/控制
LW-10531	(16bit) : PLC 7 通訊延時 (單位 : ms)	讀/寫	讀/控制	讀/控制
LW-10532	(16bit) : PLC 7 ACK 訊號延時 (單位 : ms) (SIEMENS S7/400 連接類型)	讀/寫	讀/控制	讀/控制
LW-10533	(16bit) : PLC 7 參數 1 (SIEMENS S7/400 機座)	讀/寫	讀/控制	讀/控制
LW-10534	(16bit) : PLC 7 參數 2 (SIEMENS S7/400 CPU 插槽)	讀/寫	讀/控制	讀/控制
LW-10535	(16bit) : PLC 8 超時 (單位 : 100ms)	讀/寫	讀/控制	讀/控制
LW-10536	(16bit) : PLC 8 通訊延時 (單位 : ms)	讀/寫	讀/控制	讀/控制
LW-10537	(16bit) : PLC 8 ACK 訊號延時 (單位 : ms) (SIEMENS S7/400 連接類型)	讀/寫	讀/控制	讀/控制
LW-10538	(16bit) : PLC 8 參數 1 (SIEMENS S7/400 機座)	讀/寫	讀/控制	讀/控制
LW-10539	(16bit) : PLC 8 參數 2 (SIEMENS S7/400 CPU 插槽)	讀/寫	讀/控制	讀/控制
LW-10655	(16bit) : PLC 32 超時 (單位 : 100ms)	讀/寫	讀/控制	讀/控制
LW-10656	(16bit) : PLC 32 通訊延時 (單位 : ms)	讀/寫	讀/控制	讀/控制
LW-10657	(16bit) : PLC 32 ACK 訊號延時 (單位 : ms)	讀/寫	讀/控制	讀/控制
LW-10658	(16bit) : PLC 32 參數 1	讀/寫	讀/控制	讀/控制
LW-10659	(16bit) : PLC 32 參數 2	讀/寫	讀/控制	讀/控制

22.2.14 與 PLC (COM) 的通訊狀態與控制

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9150	自動連結 PLC 1 (COM 1) (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9151	自動連結 PLC 2 (COM 2) (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9152	自動連結 PLC 3 (COM 3) (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9200	與 PLC 1 的通訊狀態 (站號 0, COM 1), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9201	與 PLC 1 的通訊狀態 (站號 1, COM 1), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9202	與 PLC 1 的通訊狀態 (站號 2, COM 1), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9203	與 PLC 1 的通訊狀態 (站號 3, COM 1), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9204	與 PLC 1 的通訊狀態 (站號 4, COM 1), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9205	與 PLC 1 的通訊狀態 (站號 5, COM 1), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9206	與 PLC 1 的通訊狀態 (站號 6, COM 1), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9207	與 PLC 1 的通訊狀態 (站號 7, COM 1), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9455	與 PLC 1 的通訊狀態 (站號 255, COM 1), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9500	與 PLC 2 的通訊狀態 (站號 0, COM 2), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9501	與 PLC 2 的通訊狀態 (站號 1, COM 2), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9502	與 PLC 2 的通訊狀態 (站號 2, COM 2), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9503	與 PLC 2 的通訊狀態 (站號 3, COM 2), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9504	與 PLC 2 的通訊狀態 (站號 4, COM 2), 設 ON 重連一次	讀/寫	讀/控制	讀/控制

LB-9505	與 PLC 2 的通訊狀態 (站號 5, COM 2), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9506	與 PLC 2 的通訊狀態 (站號 6, COM 2), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9507	與 PLC 2 的通訊狀態 (站號 7, COM 2), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9755	與 PLC 2 的通訊狀態 (站號 255, COM 2), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9800	與 PLC 3 的通訊狀態 (站號 0, COM 3), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9801	與 PLC 3 的通訊狀態 (站號 1, COM 3), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9802	與 PLC 3 的通訊狀態 (站號 2, COM 3), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9803	與 PLC 3 的通訊狀態 (站號 3, COM 3), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9804	與 PLC 3 的通訊狀態 (站號 4, COM 3), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9805	與 PLC 3 的通訊狀態 (站號 5, COM 3), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9806	與 PLC 3 的通訊狀態 (站號 6, COM 3), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9807	與 PLC 3 的通訊狀態 (站號 7, COM 3), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-10055	與 PLC 3 的通訊狀態 (站號 255, COM 3), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-12030	COM 1 開啟狀態指示 (OFF : 正常, ON : 開啟失敗) * 注 1	讀	讀	讀
LB-12031	COM 2 開啟狀態指示 (OFF : 正常, ON : 開啟失敗)	讀	讀	讀
LB-12032	COM 3 開啟狀態指示 (OFF : 正常, ON : 開啟失敗)	讀	讀	讀
LB-12033	COM 4 開啟狀態指示 (OFF : 正常, ON : 開啟失敗)	讀	讀	讀
LB-12034	COM 5 開啟狀態指示 (OFF : 正常, ON : 開啟失敗)	讀	讀	讀
LB-12035	COM 6 開啟狀態指示 (OFF : 正常, ON : 開啟失敗)	讀	讀	讀
LB-12036	COM 7 開啟狀態指示 (OFF : 正常, ON : 開啟失敗)	讀	讀	讀
LB-12037	COM 8 開啟狀態指示 (OFF : 正常, ON : 開啟失敗)	讀	讀	讀
LB-12038	COM 9 開啟狀態指示 (OFF : 正常, ON : 開啟失敗)	讀	讀	讀

LW-9351	(16bit) : PLC 1 (COM 1) 尚未處理的命令數目	讀	讀	讀
LW-9352	(16bit) : PLC 2 (COM 2) 尚未處理的命令數目	讀	讀	讀
LW-9353	(16bit) : PLC 3 (COM 3) 尚未處理的命令數目	讀	讀	讀



1. COM 的開啟狀態指示可使用於 PC 模擬時，查看 COM 是否被其他程式占用。

22.2.15 與 PLC (乙太網路) 的通訊狀態與控制

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9153	自動連結 PLC 4 (乙太網路) (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9154	自動連結 PLC 5 (乙太網路) (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9155	自動連結 PLC 6 (乙太網路) (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9156	自動連結 PLC 7 (乙太網路) (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9157	自動連結 PLC 8 (乙太網路) (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9158	自動連結 PLC 9 (乙太網路) (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9189	自動連結 PLC 40 (乙太網路) (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-10070	當線上更改 PLC 4 (乙太網路) 的 IP 或系統參數時，設 ON 重新連結 PLC	讀/寫	讀/控制	讀/控制
LB-10071	當線上更改 PLC 5 (乙太網路) 的 IP 或系統參數時，設 ON 重新連結 PLC	讀/寫	讀/控制	讀/控制
LB-10072	當線上更改 PLC 6 (乙太網路) 的 IP 或系統參數時，設 ON 重新連結 PLC	讀/寫	讀/控制	讀/控制
LB-10073	當線上更改 PLC 7 (乙太網路) 的 IP 或系統參數時，設 ON 重新連結 PLC	讀/寫	讀/控制	讀/控制
LB-10074	當線上更改 PLC 8 (乙太網路) 的 IP 或系統參數時，設 ON 重新連結 PLC	讀/寫	讀/控制	讀/控制
LB-10075	當線上更改 PLC 9 (乙太網路) 的 IP 或系統參數時，設 ON 重新連結 PLC	讀/寫	讀/控制	讀/控制
LB-10099	當線上更改 PLC 33 (乙太網路) 的 IP 或系統參數時，設 ON 重新連結 PLC	讀/寫	讀/控制	讀/控制
LB-10100	與 PLC 4 的通訊狀態 (乙太網路)，設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-10400	與 PLC 5 的通訊狀態 (乙太網路)，設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-10700	與 PLC 6 的通訊狀態 (乙太網路)，設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-11000	與 PLC 7 的通訊狀態 (乙太網路)，設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-11300	與 PLC 8 的通訊狀態 (乙太網路)，設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-11600	與 PLC 9 的通訊狀態 (乙太網路)，設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-11900	與 PLC 10 的通訊狀態 (乙太網路)，設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-11901	與 PLC 11 的通訊狀態 (乙太網路)，設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-11902	與 PLC 12 的通訊狀態 (乙太網路)，設 ON 重連一次	讀/寫	讀/控制	讀/控制

LB-11903	與 PLC 13 的通訊狀態 (乙太網路), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-11904	與 PLC 14 的通訊狀態 (乙太網路), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-11905	與 PLC 15 的通訊狀態 (乙太網路), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-11906	與 PLC 16 的通訊狀態 (乙太網路), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-11939	與 PLC 49 的通訊狀態 (乙太網路), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LW-9354	(16bit) : PLC 4 (乙太網路) 尚未處理的命令數目	讀	讀	讀
LW-9355	(16bit) : PLC 5 (乙太網路) 尚未處理的命令數目	讀	讀	讀
LW-9356	(16bit) : PLC 6 (乙太網路) 尚未處理的命令數目	讀	讀	讀
LW-9357	(16bit) : PLC 7 (乙太網路) 尚未處理的命令數目	讀	讀	讀
LW-9389	(16bit) : PLC 39 (乙太網路) 尚未處理的命令數目	讀	讀	讀
LW-9600	(16bit) : PLC 4 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9601	(16bit) : PLC 4 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9602	(16bit) : PLC 4 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9603	(16bit) : PLC 4 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9604	(16bit) : PLC 4 的連接埠	讀/寫	讀/控制	讀/控制
LW-9605	(16bit) : PLC 5 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9606	(16bit) : PLC 5 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9607	(16bit) : PLC 5 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9608	(16bit) : PLC 5 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9609	(16bit) : PLC 5 的連接埠	讀/寫	讀/控制	讀/控制
LW-9610	(16bit) : PLC 6 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9611	(16bit) : PLC 6 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9612	(16bit) : PLC 6 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9613	(16bit) : PLC 6 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9614	(16bit) : PLC 6 的連接埠	讀/寫	讀/控制	讀/控制
LW-9615	(16bit) : PLC 7 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9616	(16bit) : PLC 7 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9617	(16bit) : PLC 7 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9618	(16bit) : PLC 7 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9619	(16bit) : PLC 7 的連接埠	讀/寫	讀/控制	讀/控制
LW-9620	(16bit) : PLC 8 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9621	(16bit) : PLC 8 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制

LW-9622	(16bit) : PLC 8 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9623	(16bit) : PLC 8 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9624	(16bit) : PLC 8 的連接埠	讀/寫	讀/控制	讀/控制
LW-9625	(16bit) : PLC 9 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9626	(16bit) : PLC 9 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9627	(16bit) : PLC 9 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9628	(16bit) : PLC 9 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9629	(16bit) : PLC 9 的連接埠	讀/寫	讀/控制	讀/控制
LW-9765	(16bit) : PLC 37 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9766	(16bit) : PLC 37 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9767	(16bit) : PLC 37 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9768	(16bit) : PLC 37 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9769	(16bit) : PLC 37 的連接埠	讀/寫	讀/控制	讀/控制

22.2.16 與 PLC (USB) 的通訊狀態與控制

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9190	自動連結 PLC (USB 1) (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9191	與 PLC 的通訊狀態 (USB 1), 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LW-9390	(16bit) : PLC (USB) 尚未處理的命令數目	讀	讀	讀

22.2.17 與遠端 HMI 的通訊狀態與控制

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9068	自動連結遠端 HMI 1 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9069	自動連結遠端 HMI 2 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9070	自動連結遠端 HMI 3 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9071	自動連結遠端 HMI 4 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9072	自動連結遠端 HMI 5 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9073	自動連結遠端 HMI 6 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9074	自動連結遠端 HMI 7 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9075	自動連結遠端 HMI 8 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9099	自動連結遠端 HMI 32 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9100	與遠端 HMI 1 的通訊狀態, 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9101	與遠端 HMI 2 的通訊狀態, 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9102	與遠端 HMI 3 的通訊狀態, 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9103	與遠端 HMI 4 的通訊狀態, 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9104	與遠端 HMI 5 的通訊狀態, 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9105	與遠端 HMI 6 的通訊狀態, 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9106	與遠端 HMI 7 的通訊狀態, 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9107	與遠端 HMI 8 的通訊狀態, 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9148	與遠端 HMI 49 的通訊狀態, 設 ON 重連一次	讀/寫	讀/控制	讀/控制
LB-9149	當線上更改遠端 HMI 的 IP 時, 設 ON 重新連結遠端 HMI	讀/寫	讀/控制	讀/控制
LW-9800	(16bit) : 遠端 HMI 1 的 IPO (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制

LW-9801	(16bit) : 遠端 HMI 1 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9802	(16bit) : 遠端 HMI 1 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9803	(16bit) : 遠端 HMI 1 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9804	(16bit) : 遠端 HMI 1 的連接埠	讀/寫	讀/控制	讀/控制
LW-9805	(16bit) : 遠端 HMI 2 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9806	(16bit) : 遠端 HMI 2 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9807	(16bit) : 遠端 HMI 2 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9808	(16bit) : 遠端 HMI 2 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9809	(16bit) : 遠端 HMI 2 的連接埠	讀/寫	讀/控制	讀/控制
LW-9810	(16bit) : 遠端 HMI 3 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9811	(16bit) : 遠端 HMI 3 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9812	(16bit) : 遠端 HMI 3 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9813	(16bit) : 遠端 HMI 3 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9814	(16bit) : 遠端 HMI 3 的連接埠	讀/寫	讀/控制	讀/控制
LW-9815	(16bit) : 遠端 HMI 4 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9816	(16bit) : 遠端 HMI 4 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9817	(16bit) : 遠端 HMI 4 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9818	(16bit) : 遠端 HMI 4 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9819	(16bit) : 遠端 HMI 4 的連接埠	讀/寫	讀/控制	讀/控制
LW-9820	(16bit) : 遠端 HMI 5 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9821	(16bit) : 遠端 HMI 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制

LW-9822	(16bit) : 遠端 HMI 5 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9823	(16bit) : 遠端 HMI 5 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9824	(16bit) : 遠端 HMI 5 的連接埠	讀/寫	讀/控制	讀/控制
LW-9825	(16bit) : 遠端 HMI 6 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9826	(16bit) : 遠端 HMI 6 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9827	(16bit) : 遠端 HMI 6 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9828	(16bit) : 遠端 HMI 6 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9829	(16bit) : 遠端 HMI 6 的連接埠	讀/寫	讀/控制	讀/控制
LW-9830	(16bit) : 遠端 HMI 7 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9831	(16bit) : 遠端 HMI 7 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9832	(16bit) : 遠端 HMI 7 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9833	(16bit) : 遠端 HMI 7 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9834	(16bit) : 遠端 HMI 7 的連接埠	讀/寫	讀/控制	讀/控制
LW-9835	(16bit) : 遠端 HMI 8 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9836	(16bit) : 遠端 HMI 8 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9837	(16bit) : 遠端 HMI 8 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9838	(16bit) : 遠端 HMI 8 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9839	(16bit) : 遠端 HMI 8 的連接埠	讀/寫	讀/控制	讀/控制
LW-9895	(16bit) : 遠端 HMI 20 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9896	(16bit) : 遠端 HMI 20 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9897	(16bit) : 遠端 HMI 20 的 IP2 (IP 位址 =	讀/寫	讀/控制	讀/控制

	IP0:IP1:IP2:IP3)			
LW-9898	(16bit) : 遠端 HMI 20 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9899	(16bit) : 遠端 HMI 20 的連接埠	讀/寫	讀/控制	讀/控制
LW-9905	(16bit) : 遠端 HMI 21 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9906	(16bit) : 遠端 HMI 21 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9907	(16bit) : 遠端 HMI 21 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9908	(16bit) : 遠端 HMI 21 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9909	(16bit) : 遠端 HMI 21 的連接埠	讀/寫	讀/控制	讀/控制
LW-9910	(16bit) : 遠端 HMI 22 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9911	(16bit) : 遠端 HMI 22 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9912	(16bit) : 遠端 HMI 22 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9913	(16bit) : 遠端 HMI 22 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9914	(16bit) : 遠端 HMI 22 的連接埠	讀/寫	讀/控制	讀/控制
LW-9915	(16bit) : 遠端 HMI 23 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9916	(16bit) : 遠端 HMI 23 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9917	(16bit) : 遠端 HMI 23 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9918	(16bit) : 遠端 HMI 23 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9919	(16bit) : 遠端 HMI 23 的連接埠	讀/寫	讀/控制	讀/控制
LW-9920	(16bit) : 遠端 HMI 24 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9921	(16bit) : 遠端 HMI 24 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9922	(16bit) : 遠端 HMI 24 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制

LW-9923	(16bit) : 遠端 HMI 24 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9924	(16bit) : 遠端 HMI 24 的連接埠	讀/寫	讀/控制	讀/控制
LW-9925	(16bit) : 遠端 HMI 25 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9926	(16bit) : 遠端 HMI 25 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9927	(16bit) : 遠端 HMI 25 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9928	(16bit) : 遠端 HMI 25 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9929	(16bit) : 遠端 HMI 25 的連接埠	讀/寫	讀/控制	讀/控制
LW-9930	(16bit) : 遠端 HMI 26 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9931	(16bit) : 遠端 HMI 26 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9932	(16bit) : 遠端 HMI 26 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9933	(16bit) : 遠端 HMI 26 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9934	(16bit) : 遠端 HMI 26 的連接埠	讀/寫	讀/控制	讀/控制
LW-9935	(16bit) : 遠端 HMI 27 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9936	(16bit) : 遠端 HMI 27 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9937	(16bit) : 遠端 HMI 27 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9938	(16bit) : 遠端 HMI 27 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9939	(16bit) : 遠端 HMI 27 的連接埠	讀/寫	讀/控制	讀/控制
LW-9940	(16bit) : 遠端 HMI 28 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9941	(16bit) : 遠端 HMI 28 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9942	(16bit) : 遠端 HMI 28 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9943	(16bit) : 遠端 HMI 28 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制

	IP0:IP1:IP2:IP3)			
LW-9944	(16bit) : 遠端 HMI 28 的連接埠	讀/寫	讀/控制	讀/控制
LW-9945	(16bit) : 遠端 HMI 29 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9946	(16bit) : 遠端 HMI 29 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9947	(16bit) : 遠端 HMI 29 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9948	(16bit) : 遠端 HMI 29 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9949	(16bit) : 遠端 HMI 29 的連接埠	讀/寫	讀/控制	讀/控制
LW-9950	(16bit) : 遠端 HMI 30 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9951	(16bit) : 遠端 HMI 30 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9952	(16bit) : 遠端 HMI 30 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9953	(16bit) : 遠端 HMI 30 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9954	(16bit) : 遠端 HMI 30 的連接埠	讀/寫	讀/控制	讀/控制
LW-9955	(16bit) : 遠端 HMI 31 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9956	(16bit) : 遠端 HMI 31 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9957	(16bit) : 遠端 HMI 31 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9958	(16bit) : 遠端 HMI 31 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9959	(16bit) : 遠端 HMI 31 的連接埠	讀/寫	讀/控制	讀/控制
LW-9960	(16bit) : 遠端 HMI 32 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9961	(16bit) : 遠端 HMI 32 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9962	(16bit) : 遠端 HMI 32 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9963	(16bit) : 遠端 HMI 32 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制

LW-9964	(16bit) : 遠端 HMI 32 的連接埠	讀/寫	讀/控制	讀/控制
LW-9995	(16bit) : 遠端 HMI 39 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9996	(16bit) : 遠端 HMI 39 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9997	(16bit) : 遠端 HMI 39 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9998	(16bit) : 遠端 HMI 39 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9999	(16bit) : 遠端 HMI 39 的連接埠	讀/寫	讀/控制	讀/控制

22.2.18 與遠端 PLC 的通訊狀態與控制

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LW-10050	(16bit) : 連接遠端 PLC 1 的 HMI 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10051	(16bit) : 連接遠端 PLC 1 的 HMI 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10052	(16bit) : 連接遠端 PLC 1 的 HMI 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10053	(16bit) : 連接遠端 PLC 1 的 HMI 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10054	(16bit) : 連接遠端 PLC 1 的 HMI 的連接埠	讀/寫	讀/控制	讀/控制
LW-10055	(16bit) : 連接遠端 PLC 2 的 HMI 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10056	(16bit) : 連接遠端 PLC 2 的 HMI 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10057	(16bit) : 連接遠端 PLC 2 的 HMI 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10058	(16bit) : 連接遠端 PLC 2 的 HMI 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10059	(16bit) : 連接遠端 PLC 2 的 HMI 的連接埠	讀/寫	讀/控制	讀/控制
LW-10060	(16bit) : 連接遠端 PLC 3 的 HMI 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10061	(16bit) : 連接遠端 PLC 3 的 HMI 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10062	(16bit) : 連接遠端 PLC 3 的 HMI 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10063	(16bit) : 連接遠端 PLC 3 的 HMI 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10064	(16bit) : 連接遠端 PLC 3 的 HMI 的連接埠	讀/寫	讀/控制	讀/控制
LW-10065	(16bit) : 連接遠端 PLC 4 的 HMI 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10066	(16bit) : 連接遠端 PLC 4 的 HMI 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制

LW-10067	(16bit) : 連接遠端 PLC 4 的 HMI 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10068	(16bit) : 連接遠端 PLC 4 的 HMI 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10069	(16bit) : 連接遠端 PLC 4 的 HMI 的連接埠	讀/寫	讀/控制	讀/控制
LW-10205	(16bit) : 連接遠端 PLC 32 的 HMI 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10206	(16bit) : 連接遠端 PLC 32 的 HMI 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10207	(16bit) : 連接遠端 PLC 32 的 HMI 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10208	(16bit) : 連接遠端 PLC 32 的 HMI 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10209	(16bit) : 連接遠端 PLC 32 的 HMI 的連接埠	讀/寫	讀/控制	讀/控制
LW-10300	(16bit) : 遠端 PLC 1 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10301	(16bit) : 遠端 PLC 1 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10302	(16bit) : 遠端 PLC 1 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10303	(16bit) : 遠端 PLC 1 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10304	(16bit) : 遠端 PLC 1 的連接埠	讀/寫	讀/控制	讀/控制
LW-10305	(16bit) : 遠端 PLC 2 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10306	(16bit) : 遠端 PLC 2 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10307	(16bit) : 遠端 PLC 2 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10308	(16bit) : 遠端 PLC 2 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10309	(16bit) : 遠端 PLC 2 的連接埠	讀/寫	讀/控制	讀/控制
LW-10310	(16bit) : 遠端 PLC 3 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10311	(16bit) : 遠端 PLC 3 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10312	(16bit) : 遠端 PLC 3 的 IP2 (IP 位址 =	讀/寫	讀/控制	讀/控制

	IP0:IP1:IP2:IP3)			
LW-10313	(16bit) : 遠端 PLC 3 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10314	(16bit) : 遠端 PLC 3 的連接埠	讀/寫	讀/控制	讀/控制
LW-10315	(16bit) : 遠端 PLC 4 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10316	(16bit) : 遠端 PLC 4 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10317	(16bit) : 遠端 PLC 4 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10318	(16bit) : 遠端 PLC 4 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10319	(16bit) : 遠端 PLC 4 的連接埠	讀/寫	讀/控制	讀/控制
LW-10455	(16bit) : 遠端 PLC 32 的 IP0 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10456	(16bit) : 遠端 PLC 32 的 IP1 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10457	(16bit) : 遠端 PLC 32 的 IP2 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10458	(16bit) : 遠端 PLC 32 的 IP3 (IP 位址 = IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-10459	(16bit) : 遠端 PLC 32 的連接埠	讀/寫	讀/控制	讀/控制

22.2.19 本地/遠端操作限制

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9044	禁止遠端控制 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9053	禁止遠端讀取密碼操作 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9054	禁止遠端寫入密碼操作 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9196	本地 HMI 只支援檢視功能 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9197	只允許遠端 HMI 使用檢視功能 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9198	禁止本地 HMI 觸發巨集 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9199	禁止遠端 HMI 觸發巨集 (當狀態為 ON)	讀/寫	讀/控制	讀/控制

22.2.20 通訊錯誤碼

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LW-9400	(16bit) : 與 PLC 1 通訊錯誤時產生的錯誤訊息	讀	讀	讀
LW-9401	(16bit) : 與 PLC 2 通訊錯誤時產生的錯誤訊息	讀	讀	讀
LW-9402	(16bit) : 與 PLC 3 通訊錯誤時產生的錯誤訊息	讀	讀	讀
LW-9403	(16bit) : 與 PLC 4 通訊錯誤時產生的錯誤訊息	讀	讀	讀
LW-9404	(16bit) : 與 PLC 5 通訊錯誤時產生的錯誤訊息	讀	讀	讀
LW-9405	(16bit) : 與 PLC 6 通訊錯誤時產生的錯誤訊息	讀	讀	讀
LW-9406	(16bit) : 與 PLC 7 通訊錯誤時產生的錯誤訊息	讀	讀	讀
LW-9407	(16bit) : 與 PLC 8 通訊錯誤時產生的錯誤訊息	讀	讀	讀
LW-9449	(16bit) : 與 PLC 50 通訊錯誤時產生的錯誤訊息	讀	讀	讀
LW-9490	(16bit) : 與 PLC (USB) 通訊錯誤時產生的錯誤訊息	讀	讀	讀

22.2.21 驅動程式 ID

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LW-9300	(16bit) : 連接在本機的 PLC 1 所使用的驅動程式編號	讀	讀	讀
LW-9301	(16bit) : 連接在本機的 PLC 2 所使用的驅動程式編號	讀	讀	讀
LW-9302	(16bit) : 連接在本機的 PLC 3 所使用的驅動程式編號	讀	讀	讀
LW-9303	(16bit) : 連接在本機的 PLC 4 所使用的驅動程式編號	讀	讀	讀
LW-9331	(16bit) : 連接在本機的 PLC 32 所使用的驅動程式編號	讀	讀	讀

22.2.22 DLT645 控制器

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LW-10700	(4 words) : DLT_645 使用者 (COM 1)	讀/寫	讀/控制	讀/控制
LW-10704	(4 words) : DLT_645 密碼 (COM 1)	讀/寫	讀/控制	讀/控制
LW-10708	(6 words) : DLT_645 位址 (COM 1)	讀/寫	讀/控制	讀/控制
LW-10715	(4 words) : DLT_645 使用者 (COM 2)	讀/寫	讀/控制	讀/控制
LW-10719	(4 words) : DLT_645 密碼 (COM 2)	讀/寫	讀/控制	讀/控制
LW-10723	(6 words) : DLT_645 位址 (COM 2)	讀/寫	讀/控制	讀/控制
LW-10730	(4 words) : DLT_645 使用者 (COM 3)	讀/寫	讀/控制	讀/控制
LW-10734	(4 words) : DLT_645 密碼 (COM 3)	讀/寫	讀/控制	讀/控制
LW-10738	(6 words) : DLT_645 位址 (COM 3)	讀/寫	讀/控制	讀/控制

22.2.23 [PLC No Response] 視窗控制

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9192	禁止彈出 PLC (USB 1) 的 "PLC No Response" 視窗 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-9195	禁止彈出 PLC (USB 2) 的 "PLC No Response" 視窗 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-11960	禁止彈出 PLC 1 的 "PLC No Response" 視窗 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-11961	禁止彈出 PLC 2 的 "PLC No Response" 視窗 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-11962	禁止彈出 PLC 3 的 "PLC No Response" 視窗 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-11963	禁止彈出 PLC 4 的 "PLC No Response" 視窗 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-11964	禁止彈出 PLC 5 的 "PLC No Response" 視窗 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-11965	禁止彈出 PLC 6 的 "PLC No Response" 視窗 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-11966	禁止彈出 PLC 7 的 "PLC No Response" 視窗 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-11967	禁止彈出 PLC 8 的 "PLC No Response" 視窗 (當狀態為 ON)	讀/寫	讀/控制	讀/控制
LB-12023	禁止彈出 PLC 64 的 "PLC No Response" 視窗 (當狀態為 ON)	讀/寫	讀/控制	讀/控制

22.2.24 [快選] 視窗控制

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9013	快選視窗控制 [隱藏 (ON)/顯示 (OFF)]	讀/寫	讀/控制	讀/控制
LB-9014	快選按鍵控制 [隱藏 (ON)/顯示 (OFF)]	讀/寫	讀/控制	讀/控制
LB-9015	快選視窗/按鍵控制 [隱藏 (ON)/顯示 (OFF)]	讀/寫	讀/控制	讀/控制

22.2.25 EasyAccess

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9051	與 EasyAccess 伺服器斷線 (設 OFF)/連線 (設 ON)	讀/寫	讀/控制	讀/控制
LB-9052	與 EasyAccess 伺服器連線狀態 (當連線中狀態為 ON)	讀	讀	讀

關於 EasyAccess 的更多詳情，請參閱網址 <http://www.ihmi.net/>。



下載範例程式前，請先確定已連上網路線。

22.2.26 遠端列印/備份伺服器

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-10069	當線上更改遠端列印/備份伺服器的 IP 時，設 ON 重新連結遠端列印/備份伺服器	讀/寫	讀/控制	讀/控制
LB-12040	遠端列印/備份伺服器斷線警示 (當狀態為 ON)	讀	讀	讀
LW-9770	(16bit)：遠端列印/備份伺服器的 IP0 (IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9771	(16bit)：遠端列印/備份伺服器的 IP1 (IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9772	(16bit)：遠端列印/備份伺服器的 IP2 (IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9773	(16bit)：遠端列印/備份伺服器的 IP3 (IP0:IP1:IP2:IP3)	讀/寫	讀/控制	讀/控制
LW-9774	(6 words)：登入遠端列印/備份伺服器所需的使用者名稱 *注 1	讀/寫	讀/控制	讀/控制
LW-9780	(6 words)：登入遠端列印/備份伺服器所需的密碼 *注 1	讀/寫	讀/控制	讀/控制



- 若欲使用 LW-9774 及 LW-9780 更改設定，必須重新啟動 HMI 此變更才有效。



下載範例程式前，請先確定已連上網路線。

22.2.27 穿透通訊設定

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LW-9901	(16bit) : 穿透通訊數據來源串列埠口 (1~3 : COM 1~COM 3)	讀/寫	讀/控制	讀/控制
LW-9902	(16bit) : 穿透通訊數據目標串列埠口 (1~3 : COM 1~COM 3)	讀/寫	讀/控制	讀/控制
LW-9903	(16bit) : 穿透通訊控制 (0 : 正常, 1 : 暫停, 2 : 執行穿透功能時, 停止 HMI 與 PLC 間的通訊)	讀/寫	讀/控制	讀/控制
LW-9904	(16bit) : 穿透伺服器連接埠 (2000~2100)	讀/寫	讀/控制	讀/控制



下載範例程式前，請先確定已連上網路線。

22.2.28 VNC 控制

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-12090	VNC client 連接至 HMI (當狀態為 ON) *註 1	讀	讀	讀
LB-12091	當 VNC client 連接至 HMI 時取消自動登出功能 (當狀態為 ON) *註 1	讀/寫	讀/控制	讀/控制
LB-12092	VNC 功能 [開啟 (ON)/取消 (OFF)]	讀/寫	讀/控制	讀/控制
LW-9530	(8 words) : VNC 伺服器密碼	讀/寫	讀/控制	讀/控制



- i 系列 HMI 請搭配使用版本 20120621 或更新版本的 OS 才有效。

22.2.29 HMI 和工程檔案識別碼

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9046	工程檔案識別碼與 HMI 識別碼不同 (當狀態為 ON)	讀	讀	讀
LW-9046	(32bit) : HMI 識別碼 (只支援 i 系列) *注 1	讀/寫	讀/控制	讀



- 若欲使用 LW-9046 更改 HMI 識別碼設定，必須重新啟動 HMI 此變更才有效。



下載範例程式前，請先確定已連上網路線。

22.2.30 使用者名稱和密碼

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9050	使用者登出	寫	控制	控制
LB-9060	密碼輸入錯誤指示	讀	讀	讀
LB-9061	更新密碼 (設定為 ON)	寫	控制	控制
LW-9082	(16bit)：自動登出時間 (單位：分鐘, 0：取消此功能)	讀/寫	讀/控制	讀/控制
LW-9219	(16bit)：使用者編號 (1 ~ 12)	讀/寫	讀/控制	讀/控制
LW-9220	(32bit)：密碼	讀/寫	讀/控制	讀/控制
LW-9222	(16bit)：目前使用者可使用的物件類別 (bit 0:A, bit 1:B, bit 2:C, ...)	讀	讀	讀
LW-9500	(32bit)：使用者 1 的密碼	讀/寫	讀/控制	讀/控制
LW-9502	(32bit)：使用者 2 的密碼	讀/寫	讀/控制	讀/控制
LW-9504	(32bit)：使用者 3 的密碼	讀/寫	讀/控制	讀/控制
LW-9506	(32bit)：使用者 4 的密碼	讀/寫	讀/控制	讀/控制
LW-9508	(32bit)：使用者 5 的密碼	讀/寫	讀/控制	讀/控制
LW-9510	(32bit)：使用者 6 的密碼	讀/寫	讀/控制	讀/控制
LW-9512	(32bit)：使用者 7 的密碼	讀/寫	讀/控制	讀/控制
LW-9514	(32bit)：使用者 8 的密碼	讀/寫	讀/控制	讀/控制
LW-9516	(32bit)：使用者 9 的密碼	讀/寫	讀/控制	讀/控制
LW-9518	(32bit)：使用者 10 的密碼	讀/寫	讀/控制	讀/控制
LW-9520	(32bit)：使用者 11 的密碼	讀/寫	讀/控制	讀/控制
LW-9522	(32bit)：使用者 12 的密碼	讀/寫	讀/控制	讀/控制



下載範例程式前，請先確定已連上網路線。

22.2.31 巨集

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9059	關閉巨集指令 TRACE 功能 (當狀態為 ON) *注 1	讀/寫	讀/控制	讀/控制
LW-10900	(16bit) : 巨集指令 0 狀態 (0:就緒, 3:執行中, 5:等待 PLC 回覆, 9:等待被同步, 17:延遲)	讀	讀	讀
LW-10901	(16bit) : 巨集指令 1 狀態 (0: 就緒, 3:執行中, 5:等待 PLC 回覆, 9:等待被同步, 17:延遲)	讀	讀	讀
LW-10902	(16bit) : 巨集指令 2 狀態 (0: 就緒, 3:執行中, 5:等待 PLC 回覆, 9:等待被同步, 17:延遲)	讀	讀	讀
LW-10903	(16bit) : 巨集指令 3 狀態 (0: 就緒, 3:執行中, 5:等待 PLC 回覆, 9:等待被同步, 17:延遲)	讀	讀	讀
LW-10904	(16bit) : 巨集指令 4 狀態 (0: 就緒, 3:執行中, 5:等待 PLC 回覆, 9:等待被同步, 17:延遲)	讀	讀	讀
LW-10905	(16bit) : 巨集指令 5 狀態 (0: 就緒, 3:執行中, 5:等待 PLC 回覆, 9:等待被同步, 17:延遲)	讀	讀	讀
LW-10906	(16bit) : 巨集指令 6 狀態 (0: 就緒, 3:執行中, 5:等待 PLC 回覆, 9:等待被同步, 17:延遲)	讀	讀	讀
LW-10907	(16bit) : 巨集指令 7 狀態 (0: 就緒, 3:執行中, 5:等待 PLC 回覆, 9:等待被同步, 17:延遲)	讀	讀	讀
LW-10908	(16bit) : 巨集指令 8 狀態 (0: 就緒, 3:執行中, 5:等待 PLC 回覆, 9:等待被同步, 17:延遲)	讀	讀	讀
LW-10909	(16bit) : 巨集指令 9 狀態 (0: 就緒, 3:執行中, 5:等待 PLC 回覆, 9:等待被同步, 17:延遲)	讀	讀	讀
LW-11154	(16bit) : 巨集指令 254 狀態 (0: 就緒, 3:執行中, 5:等待 PLC 回覆, 9:等待被同步, 17:延遲)	讀	讀	讀



1. LB-9059 關閉巨集指令 TRACE 功能範例如下:



下載範例程式前，請先確定已連上網路線。

22.2.32 輸入物件功能

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LW-9002	(32bit-float)：數值輸入物件允許輸入的上限值	讀	讀	讀
LW-9004	(32bit-float)：數值輸入物件允許輸入的下限值	讀	讀	讀
LW-9052	(32bit-float)：數值輸入物件的前一次輸入值	讀	讀	讀
LW-9150	(32 words)：顯示目前鍵盤所輸入的資料 (ASCII)	讀	讀	讀
LW-9540	(16bit)：鍵盤大小寫切換	讀/寫	讀/控制	讀/控制

22.2.33 其它功能

位址	敘述	讀寫/控制		
		本地 HMI	巨集 讀/控制	遠端 HMI 讀/控制
LB-9000~ LB-9009	重新開機時狀態為 ON	讀/寫	讀/控制	讀/控制
LB-9010	資料下載指示	讀	讀	讀
LB-9011	資料上傳指示	讀	讀	讀
LB-9012	資料下載/上傳指示	讀	讀	讀
LB-9016	遠端 HMI 連接至本機 HMI (當狀態為 ON)	讀	讀	讀
LB-9017	取消 PLC 控制物件[切換視窗]的[寫回]功能	讀/寫	讀/控制	讀/控制
LB-9039	檔案備份動作狀態 (備份中狀態為 ON)	讀	讀	讀
LB-9045	memory-map 通訊失敗 (當狀態為 ON)	讀	讀	讀
LB-9049	看門狗功能 [開啟 (ON)/取消 (OFF)] (只支援 i 系列)* 注 1	讀/寫	讀/控制	讀/控制
LW-9006	(16bit)：連接到本機的遠端 HMI 數目	讀	讀	讀
LW-9024	(16bit) : memory link 系統暫存器	讀/寫	讀/控制	讀/控制
LW-9032	(8 words)：備份歷史記錄到 SD 卡, USB 碟的資料夾 名稱 *注 3	讀/寫	讀/控制	讀/控制
LW-9050	(16bit)：目前顯示的基本視窗編號	讀	讀	讀
LW-9134	(16bit)：目前所使用的語言 *注 2	讀/寫	讀/控制	讀/控制
LW-9900	(16bit) : HMI 工作模式 (0 : 正常模式, 1~3 : 測試模式 (使用 COM 1~COM 3)	讀/寫	讀/控制	讀/控制



1. 若啟用 **LB-9049** 看門狗功能，當 **HMI** 發生不可預期的通訊錯誤(失敗)時，系統將在 10 秒後自動重啟 **HMI**。
2. 當物件的文字內容要求表現出多國語言的效果時，除了需使用文字標籤外，也需搭配系統保留暫存器 **LW-9134** 的使用。**LW-9134** 的有效可設定值範圍為 0 ~ 23，此字元位址數值的映射方式將與下載至 **HMI** 的語言種類有關。當編譯下載的檔案沒有勾選全部語言時，**LW-9134** 使用方式將有所改變。

例如：使用者在文字標籤庫建立了 5 種語言，分別是語言 1 (繁體中文)，語言 2 (簡體中文)，語言 3 (英文)，語言 4 (法文)，語言 5 (日文)。若使用者只下載語言 1、語言 3、語言 5，此時 **LW-9134** 裡的數值對應的語言種類為 0 → 語言 1 (繁體中文)，1 → 語言 3 (英文)，2 → 語言 5 (日文)。

想知道如何透過項目選單物件搭配 **LW-9134** 來切換語言嗎？



下載範例程式前，請先確定已連上網路線。

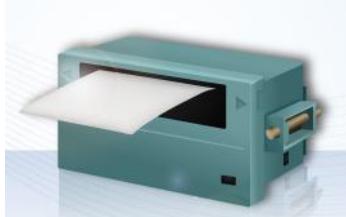
3. 系統將以 **HMI** 名稱作為預設的備份資料夾名稱。

第二十三章 HMI 支援的印表機類型

23.1 支援的印表機類型

HMI 可支援的印表機驅動大致分為以下幾種：

■ 北京迅普 SP-M, D, E, F



此驅動使用 EPSON ESC 串口微型印表機協議並使用串列埠連接。請配合印表機調整通訊參數。[每行點數] 必須正確設定，且不得超過印表機預設值：

100 pixels : 1610 型號之印表機；

220 pixels : 2407, 4004 型號之印表機。

SP-E1610SK (紙張寬度 45mm),

SP-E400-4S (紙張寬度 57.5mm)

建議中國地區以外的用戶使用此種 SP 印表機型。北京迅普：<http://www.siupo.com>

■ EPSON ESC/P2 系列



使用串列埠連接，請配合印表機調整通訊參數。此驅動使用 EPSON ESC/P2 印表機通訊協議。

點陣式印表機: LQ-300, LQ-300+,
LQ-300K+ (RS232), LQ-300+II (RS232)

噴墨式印表機: Stylus Photo 750

雷射印表機: EPL-5800

■ HP PCL 系列 (USB)



使用 USB 埠連接，可支援 HP 印表機的 PCL5 協議或 PostScript3 的印表機語言。因 PCL 印表機語言的向下支援特性，支援 PCL5 或以上的印表機皆可以支援 PCL5 協議。

■ Axiohm A630



法國愛克勝微型印表機，使用串列埠連接，請配合印表機調整通訊參數。

■ SPRT



使用串列埠連接，請配合印表機調整通訊參數。[每行點數] 必須正確設定，且不得超過印表機預設值“100”。支援型號：

SP-DN40SH: 點陣式印表機

SP-RMDIII40SH: 熱感式印表機

■ EPSON TM-L90

使用串列埠連接，請配合印表機調整通訊參數。[每行點數] 必須正確設定，且不得超過印表機預設值 “576” 。

■ EPSON TM-T70

使用串列埠連接，請配合印表機調整通訊參數。[每行點數] 必須正確設定，且不得超過印表機預設值 “576” 。 可選擇紙張切割模式，分為 [不切] 或 [半切] 。

■ BRIGHTEK WH-A9

支援型號：

A92R10-00E72A: 尾碼為 72 的是 16 位印表機，A 表示寬電壓 5~9V，此款與 A6 16 點陣相同

■ BRIGHTEK WH-E19

支援型號：

使用串列埠連接，請配合印表機調整通訊參數。

■ BRIGHTEK WH-E22 (煥煌)

支援型號：

E22R10-00E725: 與 A7 16 點陣相同，A7 型號為 A72R90-31E72A

E221R90-00E11740GA: 此印表機為 485 介面，需要使用 232 轉 485 模組

■ BRIGHTEK WH-C1/C2

使用串列埠連接，請配合印表機調整通訊參數。可選擇紙張切割模式，分為 [不切]、[半切] 或 [全切] 。

■ 遠端列印伺服器



HMI 可透過乙太網路連接在 PC 上的印表機，並使用 EasyPrinter 來執行列印動作。由於 EasyPrinter 在 MS Windows 系統下運行，因此支援市面上大部分的印表機。

23.2 如何新增一台印表機設備並觸發列印

23.2.1 新增印表機類型

從【系統參數設定】»【HMI 屬性】頁籤選擇欲連接的印表機型號，並正確設定相關參數。



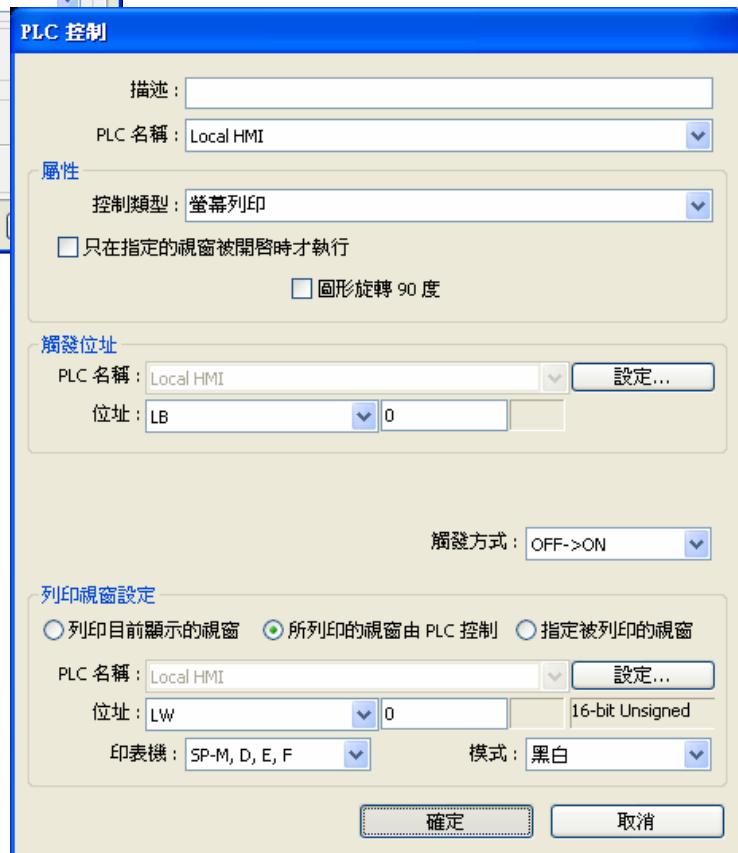
23.2.2 觸發列印功能

使用者可透過  “功能鍵” 物件來直接觸發列印功能。



印功能。

或者, 使用  “PLC 控制” 物件的 [螢幕列印], 藉由預先定義的位元位址來觸發列



第二十四章 Recipe Editor

24.1 概要

Recipe Editor 可用來建立 HMI 所使用的配方資料檔案，也可開啟及編輯現有的配方資料檔案。可從 Project Manager 點擊【配方資料/擴展記憶體編輯器】，開啟工具進行編輯。



24.2 Recipe/Extended Memory Editor 設定

如何新增 *.rcp / *.emi 檔案

設定存取範圍 » 新增資料格式

【選擇資料格式】

定義完成的資料格式可儲存，並於下次需要時載入。範本將存成“dataEX.fmt”檔案並存放在 EasyBuilder 8000 安裝目錄

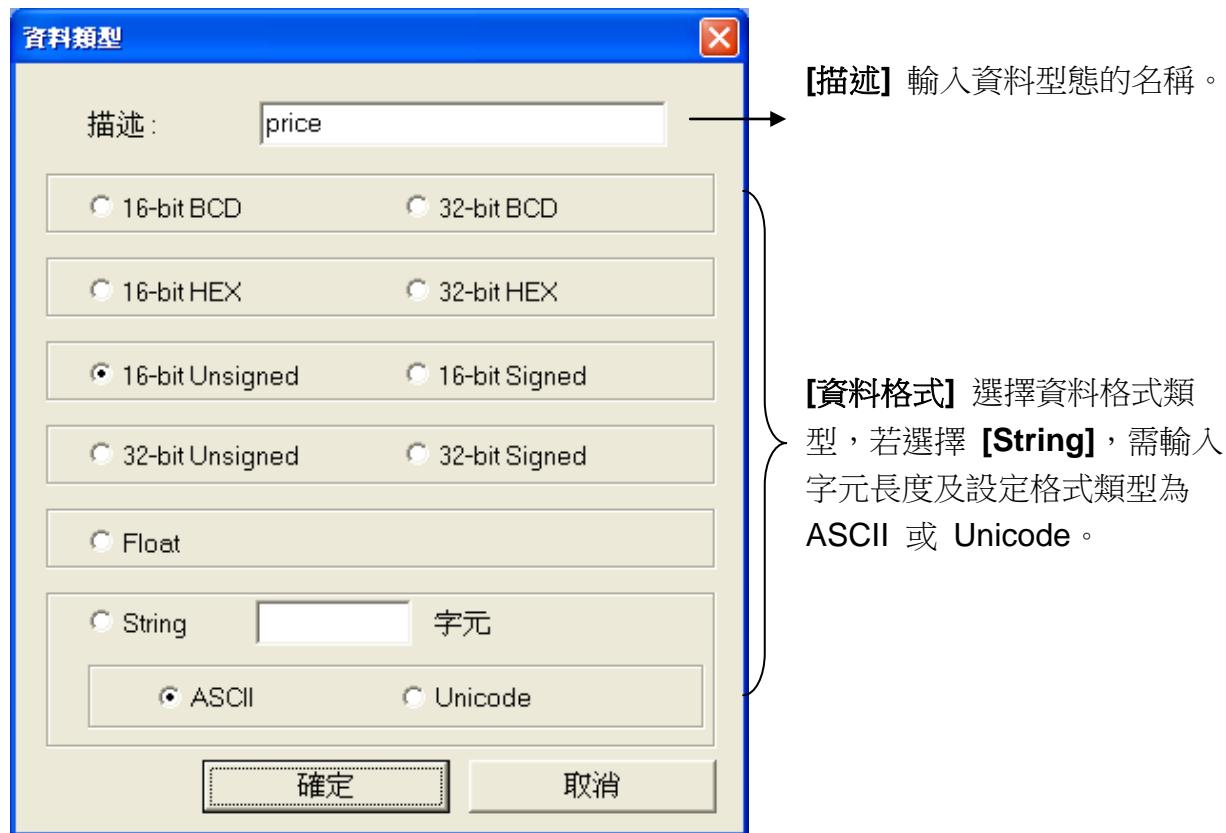
【存取範圍】 填入起始位址和結束位址，以字元為單位。

【資料格式】 可在資料格式區域中編輯新的資料格式。



Example 1

步驟 1 當點擊【新增】後，彈出資料類型編輯視窗如下：



步驟 2 資料格式定義完成後，點選 **【確定】** 即可編輯配方資料。



配方/擴展記憶體編輯器 - [New document*]

檔案(F) 編輯(E) 檢視(V) 視窗(W) 幫助(H)

ID ADDRESS recipe no. product price barcode

ID	ADDRESS	recipe no.	product	price	barcode
0	0	0	bread	13	1547965231
1	23	1	coke	20	1437286591
2	46	2	coffee	245	3265479210
3	69	3	chair	599	6454789321
4	92	4	desk	750	1464545441
5	115	5	tea	15	1234534444
6	138	6	egg	8	4568974164
7	161	7	coke	28	1246634344
8	184	8	coke	29	4644245645
9	207	9	lemon	35	4645244575
10	230	10	tea bag	68	5676454567
11	253	11	book	245	4564676454
12	276	0		0	
13	299	0		0	
14	322	0		0	
		~		~	

Ready NUM

步驟 3

此範例的資料格式長度為 23 個字元，因此可將每 23 個字元長度視為一組配方來使用。如下，

第一組的 “recipe no.” 為 address 0，”product” 位址為 address 1 ~ 10，”price” 為 address 11 ~ 12，”barcode” 為 address 13 ~ 22；

第二組的 “recipe no.” 為 address 23，”product” 位址為 address 24 ~ 33，”price” 為 address 34 ~ 35，”barcode” 為 address 36 ~ 45...依此類推。



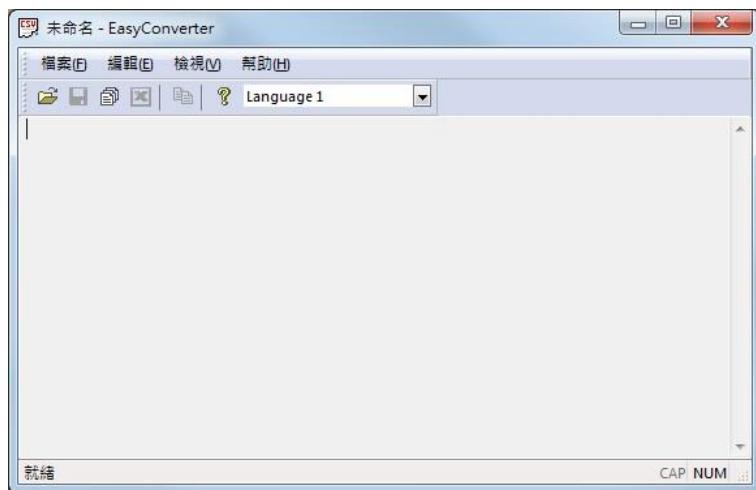
- 編輯完成的配方資料可儲存成 *.rcp、*.emi 或 *.csv 檔案格式。儲存完成的 *.rcp 檔案可藉由 Project Manager 或外部裝置 (USB 碟或 SD 卡) 下載到 HMI，而 *.emi 則可直接存放至外部裝置並插入至 HMI 讀取，即為擴展位址 EM。

第二十五章 EasyConverter

25.1 概要

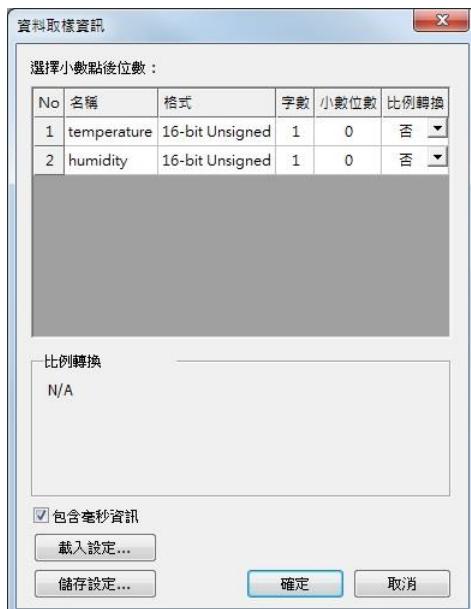
EasyConverter 程式可讀取由 HMI 摄取的資料取樣記錄、事件記錄檔案，並轉換成 Excel 格式。

- 從 Project Manager 點擊 EasyConverter。
- 從 EasyBuilder8000 工具選單下點擊 [事件記錄/資料取樣記錄轉換程式]。

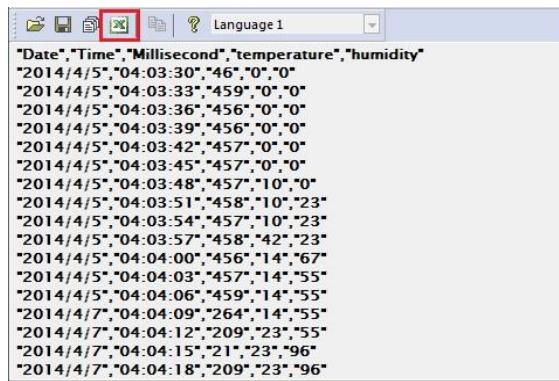


25.2 將資料取樣記錄檔案輸出至 Excel

1. 當開啟資料取樣記錄檔案，此時將彈出設定視窗如下，請依照需求作相關設定。

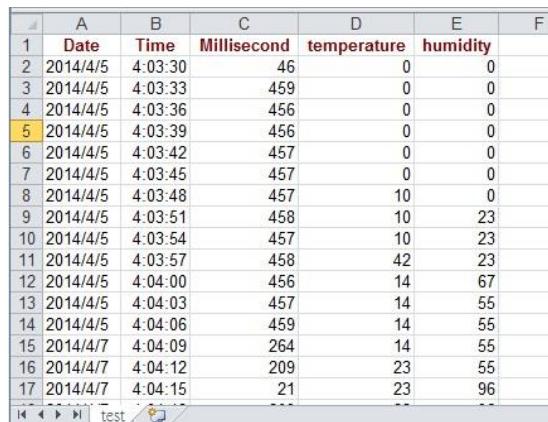


2. 按下【確定】後，資料取樣記錄將顯示如下，再按下【匯出至 Excel】即可轉換成 Excel 格式。



```
"Date","Time","Millisecond","temperature","humidity"
"2014/4/5","04:03:30","46","0","0"
"2014/4/5","04:03:33","459","0","0"
"2014/4/5","04:03:36","456","0","0"
"2014/4/5","04:03:39","456","0","0"
"2014/4/5","04:03:42","457","0","0"
"2014/4/5","04:03:45","457","0","0"
"2014/4/5","04:03:48","457","10","0"
"2014/4/5","04:03:51","458","10","23"
"2014/4/5","04:03:54","457","10","23"
"2014/4/5","04:03:57","458","42","23"
"2014/4/5","04:04:00","456","14","67"
"2014/4/5","04:04:03","457","14","55"
"2014/4/5","04:04:06","459","14","55"
"2014/4/7","04:04:09","264","14","55"
"2014/4/7","04:04:12","209","23","55"
"2014/4/7","04:04:15","21","23","96"
"2014/4/7","04:04:18","209","23","96"
```

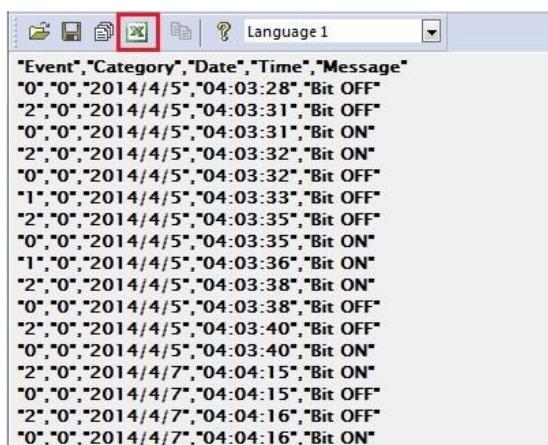
3. Excel 檔案顯示如下。



	A	B	C	D	E	F
1	Date	Time	Millisecond	temperature	humidity	
2	2014/4/5	4:03:30	46	0	0	
3	2014/4/5	4:03:33	459	0	0	
4	2014/4/5	4:03:36	456	0	0	
5	2014/4/5	4:03:39	456	0	0	
6	2014/4/5	4:03:42	457	0	0	
7	2014/4/5	4:03:45	457	0	0	
8	2014/4/5	4:03:48	457	10	0	
9	2014/4/5	4:03:51	458	10	23	
10	2014/4/5	4:03:54	457	10	23	
11	2014/4/5	4:03:57	458	42	23	
12	2014/4/5	4:04:00	456	14	67	
13	2014/4/5	4:04:03	457	14	55	
14	2014/4/5	4:04:06	459	14	55	
15	2014/4/7	4:04:09	264	14	55	
16	2014/4/7	4:04:12	209	23	55	
17	2014/4/7	4:04:15	21	23	96	

25.3 將事件記錄檔案輸出至 Excel

1. 當開啟事件記錄檔案，按下【確定】後，事件記錄將顯示如下，再按下【匯出至 Excel】即可轉換成 Excel 格式。



```
"Event","Category","Date","Time","Message"
"0","0","2014/4/5","04:03:28","Bit OFF"
"2","0","2014/4/5","04:03:31","Bit OFF"
"0","0","2014/4/5","04:03:31","Bit ON"
"2","0","2014/4/5","04:03:32","Bit ON"
"0","0","2014/4/5","04:03:32","Bit OFF"
"1","0","2014/4/5","04:03:33","Bit OFF"
"2","0","2014/4/5","04:03:35","Bit OFF"
"0","0","2014/4/5","04:03:35","Bit ON"
"1","0","2014/4/5","04:03:36","Bit ON"
"2","0","2014/4/5","04:03:38","Bit ON"
"0","0","2014/4/5","04:03:38","Bit OFF"
"2","0","2014/4/5","04:03:40","Bit OFF"
"0","0","2014/4/5","04:03:40","Bit ON"
"2","0","2014/4/7","04:04:15","Bit ON"
"0","0","2014/4/7","04:04:15","Bit OFF"
"2","0","2014/4/7","04:04:16","Bit OFF"
"0","0","2014/4/7","04:04:16","Bit ON"
```

2. Excel 檔案顯示如下。

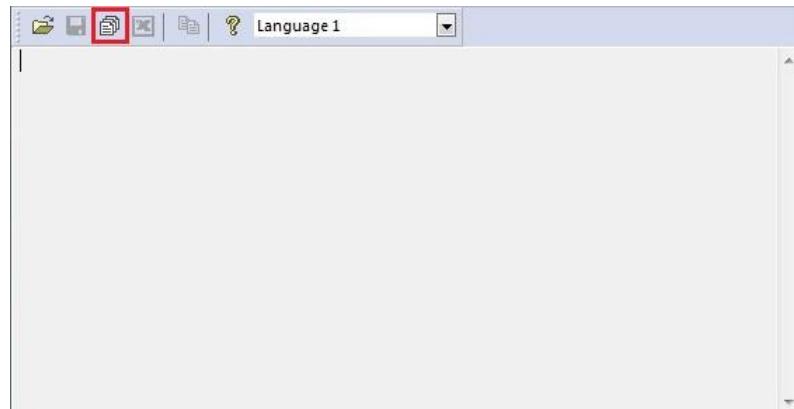
A	B	C	D	E	F
1	Event	Category	Date	Time	Message
2	0	0	2014/4/5	4:03:28	Bit OFF
3	2	0	2014/4/5	4:03:31	Bit OFF
4	0	0	2014/4/5	4:03:31	Bit ON
5	2	0	2014/4/5	4:03:32	Bit ON
6	0	0	2014/4/5	4:03:32	Bit OFF
7	1	0	2014/4/5	4:03:33	Bit OFF
8	2	0	2014/4/5	4:03:35	Bit OFF
9	0	0	2014/4/5	4:03:35	Bit ON
10	1	0	2014/4/5	4:03:36	Bit ON
11	2	0	2014/4/5	4:03:38	Bit ON
12	0	0	2014/4/5	4:03:38	Bit OFF
13	2	0	2014/4/5	4:03:40	Bit OFF
14	0	0	2014/4/5	4:03:40	Bit ON
15	2	0	2014/4/7	4:04:15	Bit ON



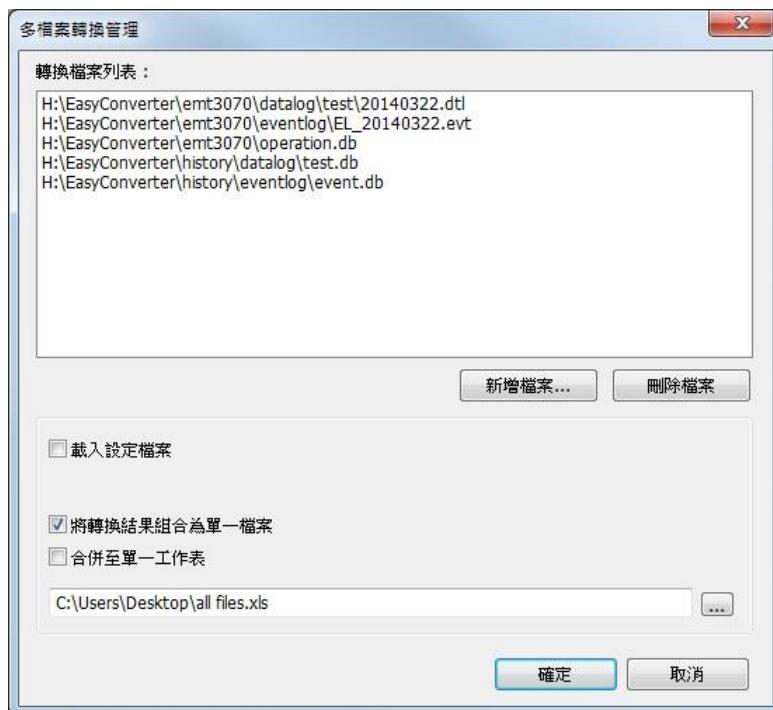
- 在表格的第一行可以發現 "Event" 欄位，**0** -> 表示事件觸發時；**1** -> 表示事件確認時；**2** -> 表示事件恢復正常時。

25.4 多檔案轉換

1. 點選 [多檔案轉換] 圖示可呼叫多檔案轉換管理視窗。



2. 點選 [新增檔案] 可加入欲轉換的檔案名稱，若未勾選 [將轉換結果組合為單一檔案] 而按下 [確定]，檔案將個別被輸出至不同的 Excel 檔案。



3. 若勾選 [將轉換結果組合為單一檔案]，則所有的檔案將被輸出至單一個 Excel 檔案，且每個檔案分一頁籤，Excel 檔案顯示如下。

	A	B	C	D	E	F	G
1	Date	Time	Millisecond	temperature	humidity		
2	2014/3/22	6:36:52	260	2	1		
3	2014/3/22	6:36:55	250	6	3		
4	2014/3/22	6:36:58	250	10	6		
5	2014/3/22	6:37:01	300	13	8		
6	2014/3/22	6:37:04	280	17	10		
7	2014/3/22	6:37:07	250	21	13		
8							
9							
10							
11							
12							
13							
14							
15							

20140322 / EL_20140322 / operation / test / event

25.5 比例轉換功能

比例轉換功能使用方式如下：

新數值 = [(數值 + A) × B] + C，使用者可以在 A、B 和 C 設定數值。

A -> 數值下限；**B** -> [(比例最大值) – (比例最小值) / (數值上限) – (數值下限)]；**C** -> 比例最小值。

範例：

有一電壓資料，其格式是 16-bit unsigned，電壓數值介於 0 ~ 4096，若要將其電壓數值轉換成伏特，介於 -5V ~ +5V 之間。

新數值 = [(數值 + 0) × 0.0024] + (-5)：



比例轉換前

Date	Time	Millisecond	sample
2014/06/30	23:02:50	80	0
2014/06/30	23:02:54	30	0
2014/06/30	23:02:57	990	55
2014/06/30	23:03:02	70	55
2014/06/30	23:03:06	20	89
2014/06/30	23:03:10	20	159
2014/06/30	23:03:14	30	530
2014/06/30	23:03:18	20	898
2014/06/30	23:03:22	40	1024
2014/06/30	23:03:26	0	2055
2014/06/30	23:03:30	30	2055

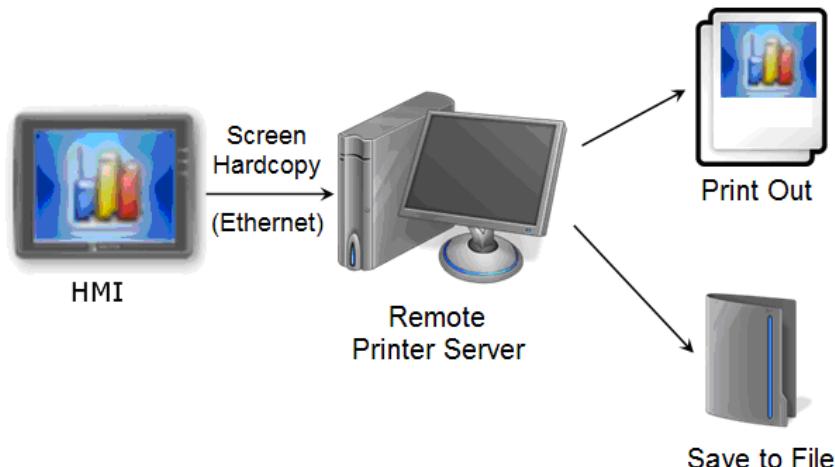
比例轉換後

Date	Time	Millisecond	sample
2014/06/30	23:02:50	80	-5.000
2014/06/30	23:02:54	30	-5.000
2014/06/30	23:02:57	990	-4.868
2014/06/30	23:03:02	70	-4.868
2014/06/30	23:03:06	20	-4.786
2014/06/30	23:03:10	20	-4.618
2014/06/30	23:03:14	30	-3.728
2014/06/30	23:03:18	20	-2.845
2014/06/30	23:03:22	40	-2.542
2014/06/30	23:03:26	0	-0.068
2014/06/30	23:03:30	30	-0.068

以上的資料設定可以儲存成範本，並於下次使用時可以直接載入設定。範本設定的附檔名為 *.lgs。

第二十六章 EasyPrinter

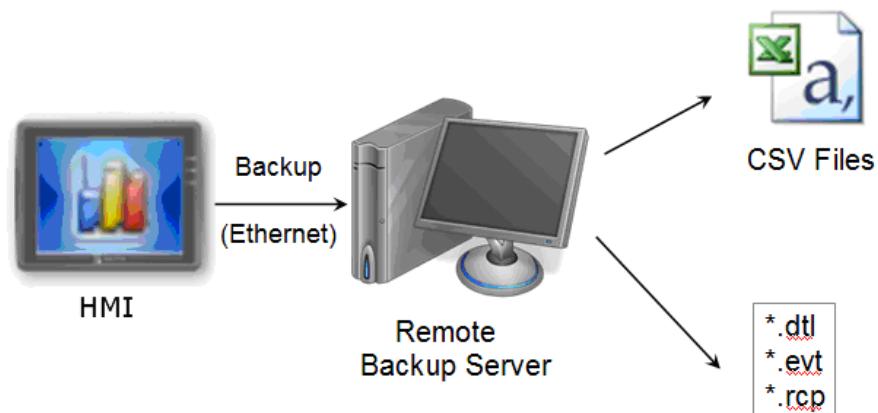
EasyPrinter 是屬於 Win 32 的應用程式，因此只能在 MS Windows 2000 / XP / Vista / 7 等系統下運行。此功能讓人機可以透過乙太網路，輸出螢幕列印於遠端電腦，請見以下說明：



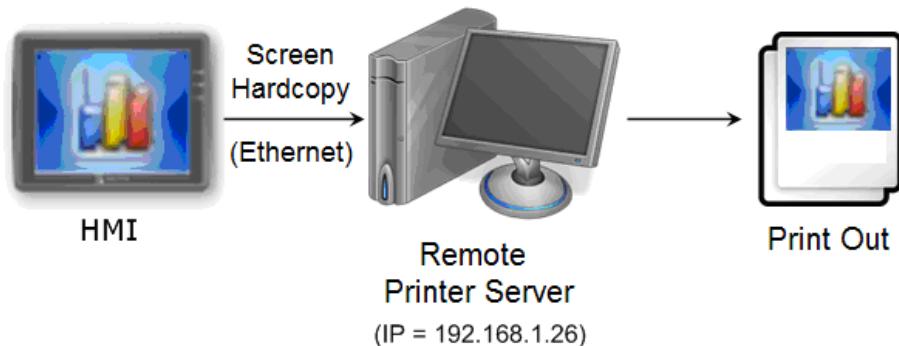
以下為使用 EasyPrinter 的優點：

- EasyPrinter 提供兩種螢幕列印輸出模式：[輸出至] 及 [儲存至]。使用者可使用其中一種或兩個都使用。(請參考 26.1.1 章節)
- 由於 Easy Printer 在 MS Windows 系統下運行，因此支援市面上大部分的印表機。
- 此功能下多台人機可以共用一台實體印表機，使用者不需為每台人機各準備一台印表機。

另外，EasyPrinter 可以當做是一台備份伺服器。使用者可使用人機上的備份元件，透過乙太網路，將取樣資料與事件記錄等歷史檔案拷貝至遠端 PC。請見下方說明：



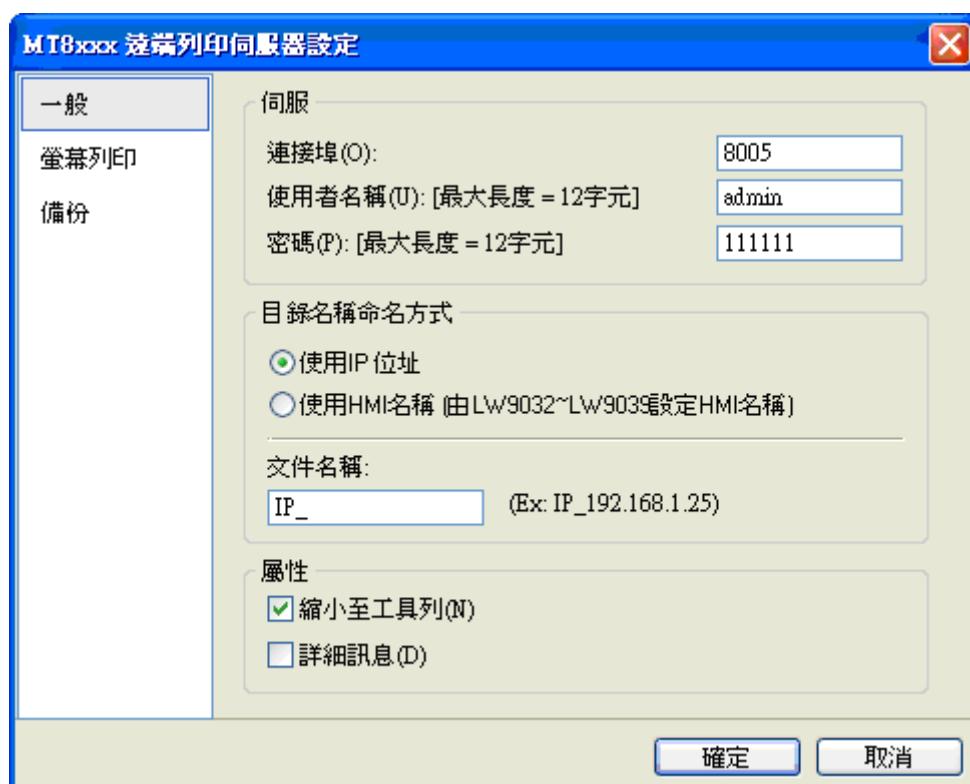
26.1 使用 EasyPrinter 為列印伺服器



使用者可用 **【功能鍵】** 物件來操作螢幕列印。這些螢幕列印會透過乙太網路被傳送至遠端印表機伺服器，然後被列印出來。

26.1.1 EasyPrinter 設定程序

在 EasyPrinter 設定頁下點選 **【選項】»【設定】** 會出現下面的對話窗：



1. 點選左列 **【一般】**。
2. 在 **【伺服】**，設定 **【連接埠】** 為 “8005”，**【使用者名稱】** 為 “admin”，**【密碼】** 為“111111”。(以上皆為預設值)。
3. 在 **【目錄名稱命名方式】**，點選 **【使用 IP 位址】** 並在 **【文件名稱】** 填入 “IP_”。

4. 在 [屬性]，選擇 [縮小至工具列]。

接著設定輸出位置：

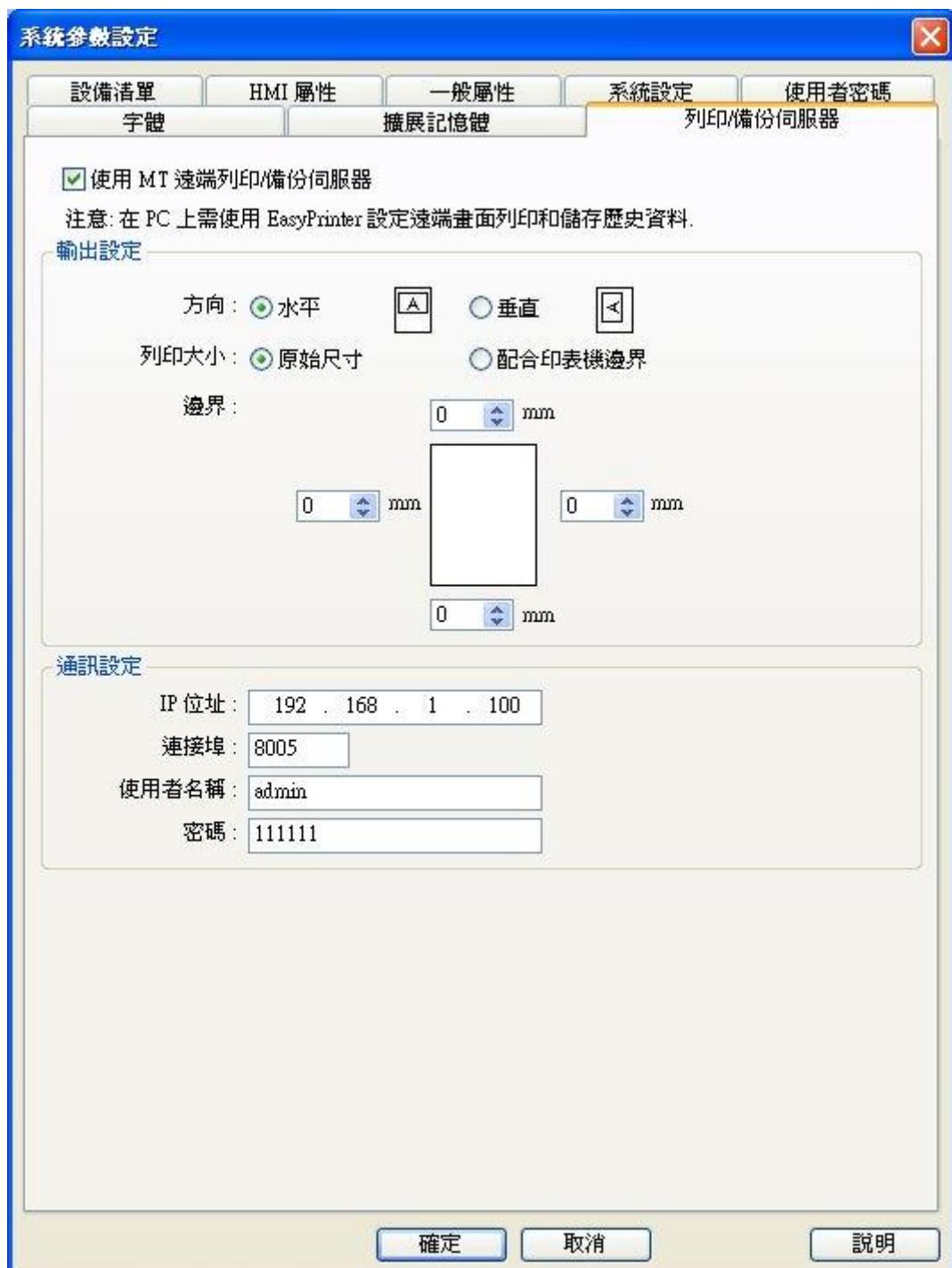


1. 點選左列 [螢幕列印]。
2. 在 [輸出]，點選 [輸出至] 並選擇一台印表機做為螢幕列印的輸出裝置。(注意：使用者只能選擇自己的系統中存在的印表機，上列印表機僅為參考。)
3. 按下 [確定] 確認使用以上設定。
4. 在 EasyPrinter 設定頁下點選 [檔案] » [允許輸出]，EasyPrinter 會將這些列印指令輸出。

26.1.2 EasyBuilder 設定程序

在 EasyBuilder 設定 EasyPrinter 的設定程序：

1. 開啟 EasyBuilder，並開啟新專案或已使用之專案。
2. 在 [編輯] » [系統參數設定] » [列印/備份伺服器] 中，勾選 [使用遠端列印/備份伺服器]，如下：



3. 在 [輸出設定]，指定適當的邊界，(在此例中上下左右邊界皆設為 15mm)。
4. 在 [通訊設定]，輸入列印伺服器 [IP 位址]。同 EasyPrinter 的設定，指定 [連接埠] 號“8005”，[使用者名稱] 為 “admin”，[密碼] 為“111111”。
5. 按下 [確定]。
6. 接著在功能表 [物件] » [開關] 選擇 **F** [功能鍵] 並在物件設定頁中點選 [畫面列印] 並將 [印表機] 設定至 [MT 遠端列印/備份伺服器]。

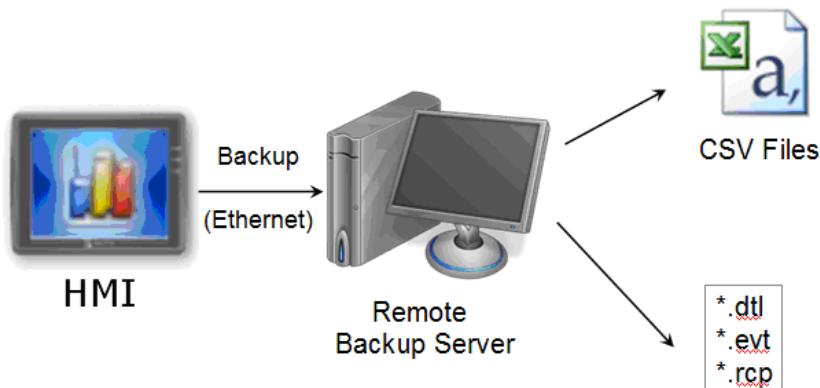


7. 將 **[功能鍵]** 物件置於 **[公共視窗]** (四號視窗)，使用者便可以隨時開始螢幕列印
8. **[編譯]** 及 **[下載]** 工程檔至人機，按下已設定的 **[功能鍵]** 物件，開始列印。



- 使用者亦可透過 **[PLC 控制]** 物件來達成螢幕列印
- 警報資料無法透過 EasyPrinter 列印。
- EasyPrinter 只能透過乙太網路與人機通訊，請確認所使用人機網路是否設定正確。

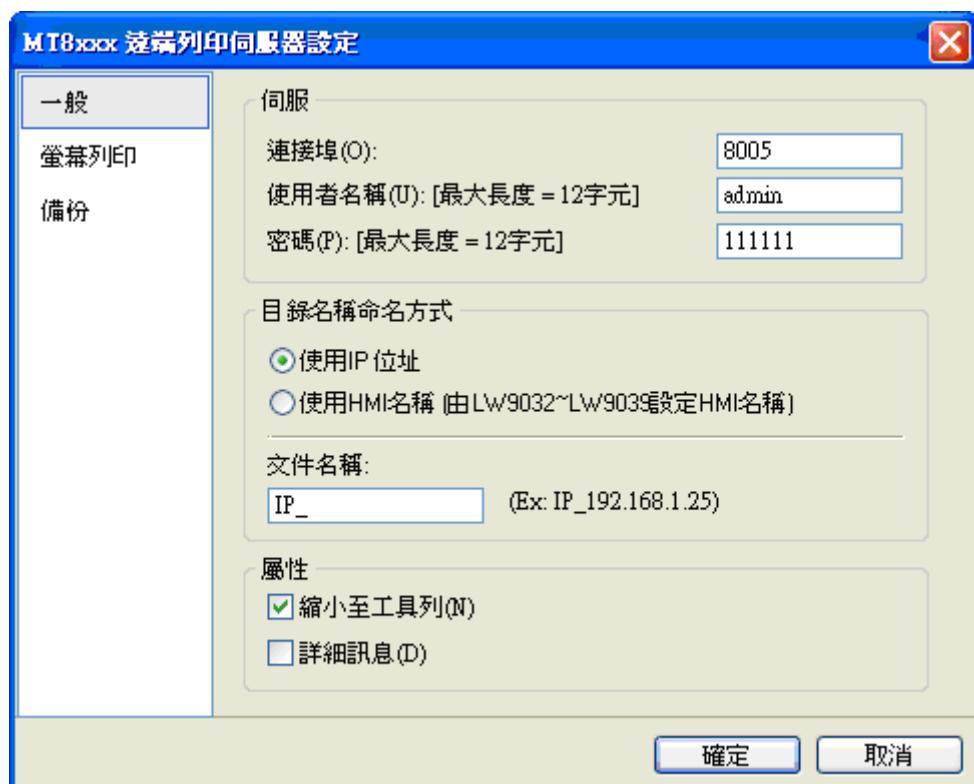
26.2 使用 EasyPrinter 為備份伺服器



使用者可使用 [備份] 物件，將歷史資料，例如取樣資料與事件記錄等上傳至遠端備份伺服器。

26.2.1 EasyPrinter 備份設定程序

在 EasyPrinter 設定頁下點選 [選項] » [設定] 會出現下面的對話窗：



1. 在左列選擇 [一般]
2. 在 [伺服器] 中，指定 [連接埠] 為 “8005”，[使用者名稱] 為 “admin”，[密碼] 為 “111111”。(以上皆為預設值)。
3. 在 [目錄名稱命名方式]，點選[使用 IP 位址] 並指定“IP_” 為[文件名稱]。

4. 在 [屬性]，選擇[縮小至工具列]。

接著設定備份位置：

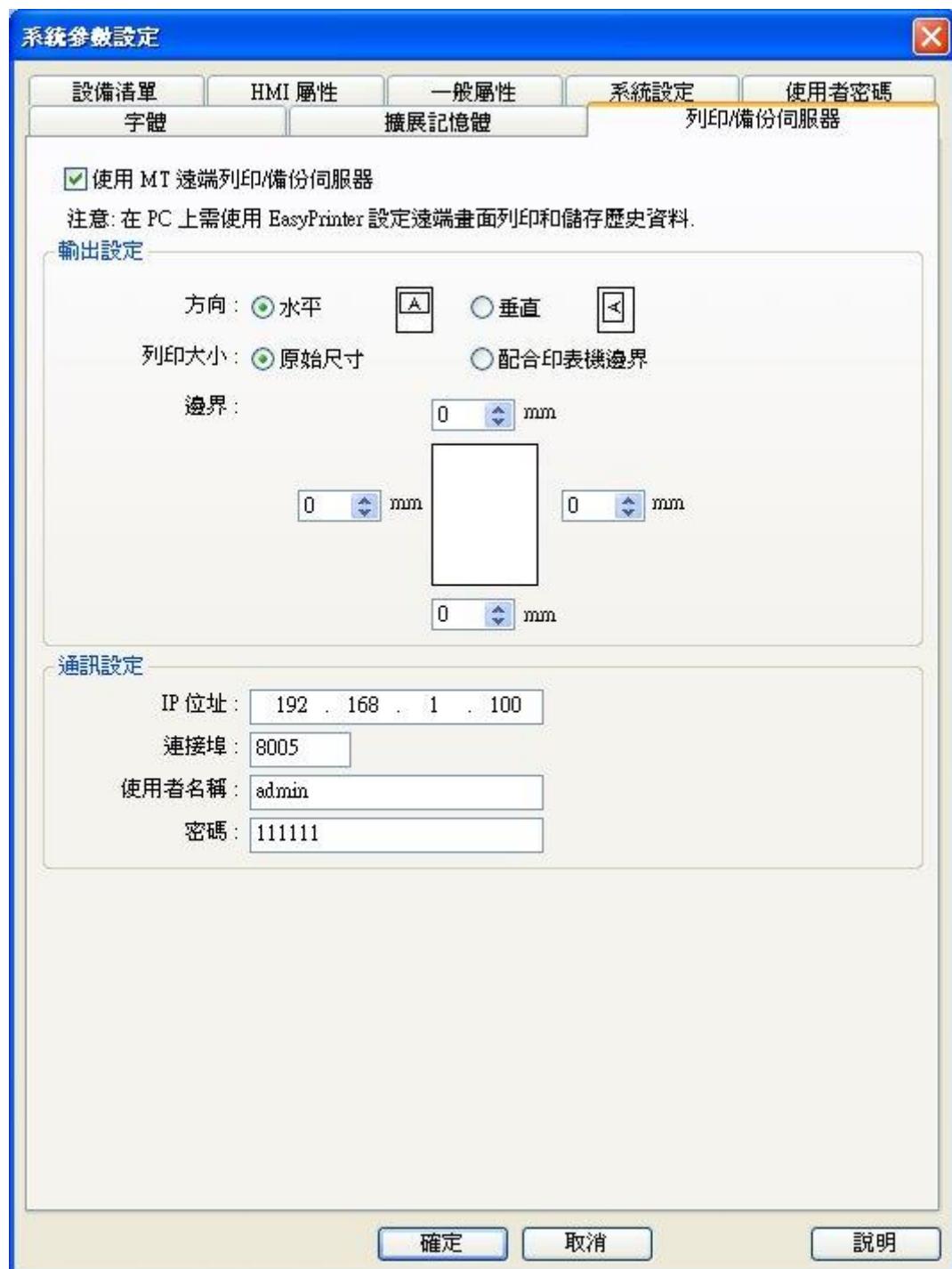


1. 點選左列 [備份]。
2. 在 [輸出]，點擊 來瀏覽及選擇歷史資料的儲存路徑。
3. 按下 [確定] 確認使用以上設定。
4. 在 EasyPrinter 設定頁下點選 [檔案] » [允許輸出]，EasyPrinter 會將備份資料儲存在方才所選的路徑。

26.2.2 EasyBuilder 備份設定程序

接著在 EasyBuilder 工程檔中加入相關設定：

1. 開啟 EasyBuilder，並開啟新專案或已使用之專案。
2. 在 [編輯] » [系統參數設定] » [列印/備份伺服器]中，勾選 [使用遠端列印/備份伺服器]，如下：



3. 在**【通訊設定】**，輸入列印伺服器**[IP 位址]** 同 EasyPrinter 的設定，指定**[連接埠]**號“8005”，**[使用者名稱]** 為 “admin”，**[密碼]** 為“111111”。(以上皆為預設值)。
4. 按下 **【確定】**。

接下來添加備份功能到視窗：

- 接著在功能表 **[物件]** 選擇 **[備份]** 會出現下面的對話窗：



- 在 **[來源]**，選擇 **[事件記錄]** (或照需求選擇 **[RW]**、**[RW_A]**)。
- 在 **[備份位置]**，選擇 **[遠端列印/備份伺服器]**
- 在 **[範圍]**，選擇 **[今天]** 和 **[全部]** (或照實際需求改變)。
- 在 **[觸發]**，選擇 **[手動]**
- 按下 **[確定]**
- 將 **[備份]** 物件置於視窗中，例如 **[4:共用視窗]**，使用者便可隨時執行備份。
- [編譯]** 及 **[下載]** 工程檔至人機，按下前面設定的 **[備份]** 物件，開始備份歷史資料。



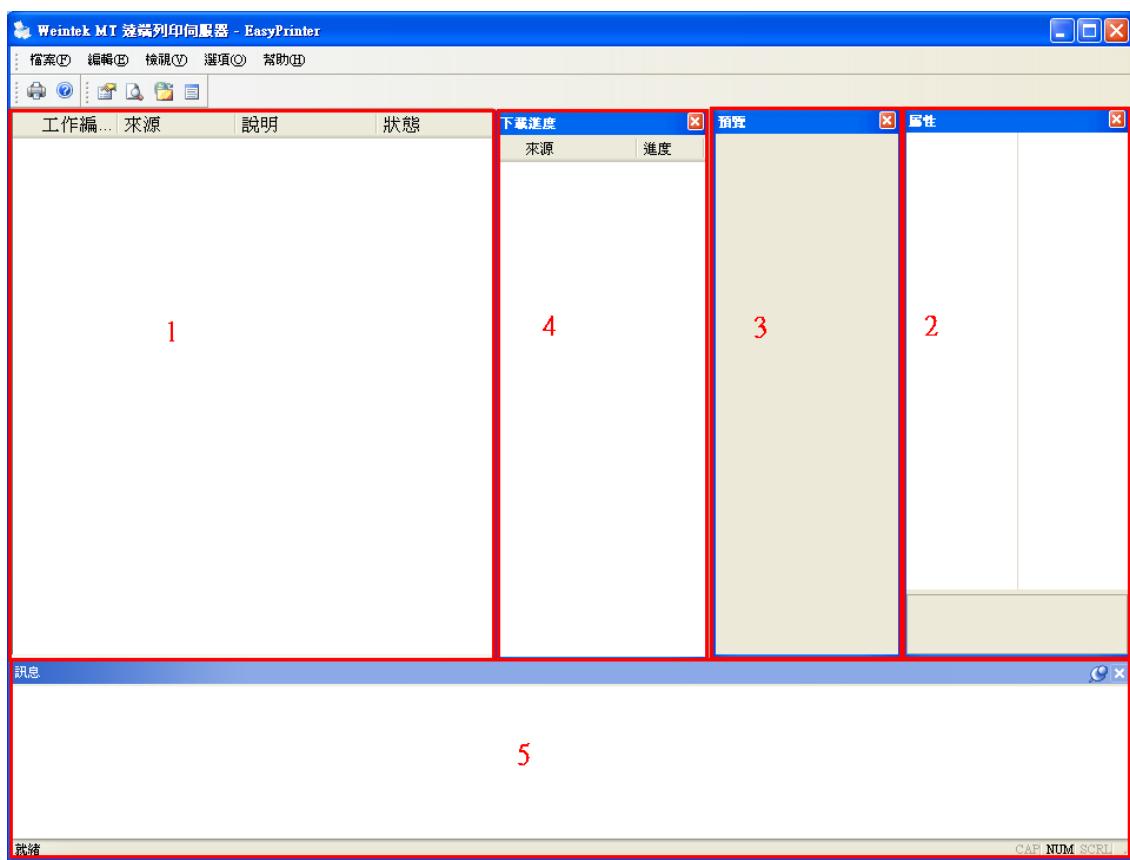
- **[備份]** 物件亦可用位元位址觸發。
- 使用者可以放置一個 **[排程]** 物件，在一週的最後一天轉為 ON，用以觸發 **[備份]** 物件自動備份所有歷史資料。

26.3 EasyPrinter 操作說明

以下介紹 EasyPrinter 視窗介面和操作說明：

26.3.1 視窗介面

EasyPrinter 視窗可切分為五個區域，如下圖：



各區域名稱和功用如下：

區域	名稱	說明
1	工作列表	此視窗顯示所有工作，包括螢幕列印與備份工作。
2	屬性視窗	此視窗顯示工作列表中所選工作之資訊。
3	預覽視窗	此視窗顯示工作列表中所選螢幕列印之預覽影像。
4	下載進度視窗	此視窗顯示新進工作之下載進度。
5	訊息視窗	此視窗顯示工作執行中的時間與訊息，例如密碼錯誤等等。

26.3.2 操作說明

以下說明未提及之 EasyPrinter 功能：

- **[檔案] » [允許輸出]**

已選：EasyPrinter 一一執行工作

未選：EasyPrinter 將工作保留於記憶體中



- EasyPrinter 只能將最多至 128MB 的工作資料保留於記憶體中，若記憶體已滿，所有新進工作會被拒絕。使用者可以選擇[允許輸出]直接執行，或是刪除部分工作來清出記憶空間給新進工作。

- **[編輯] » [編輯]**

編輯 [螢幕列印]。使用者可以在此自由設定方向、列印大小、邊界。



- **[編輯] » [刪除]**

永久刪除所選工作。

- **[編輯] » [選擇全部]**

選擇 [工作列表] 上的所有工作。



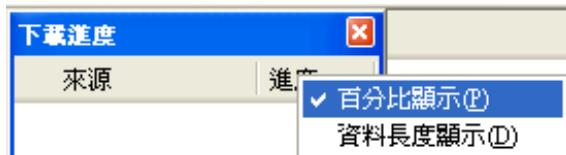
- 備份工作不可編輯。

- 只有在選擇工作後才能[編輯]。

- 選擇至少一樣工作方可[刪除]。

- [檢視] » [屬性視窗] 顯示或隱藏屬性視窗。
- [檢視] » [預覽視窗] 顯示或隱藏預覽視窗。
- [檢視] » [下載進度]

使用者可以選擇下載進度的顯示方式，可點擊 **[進度]** 如下圖：

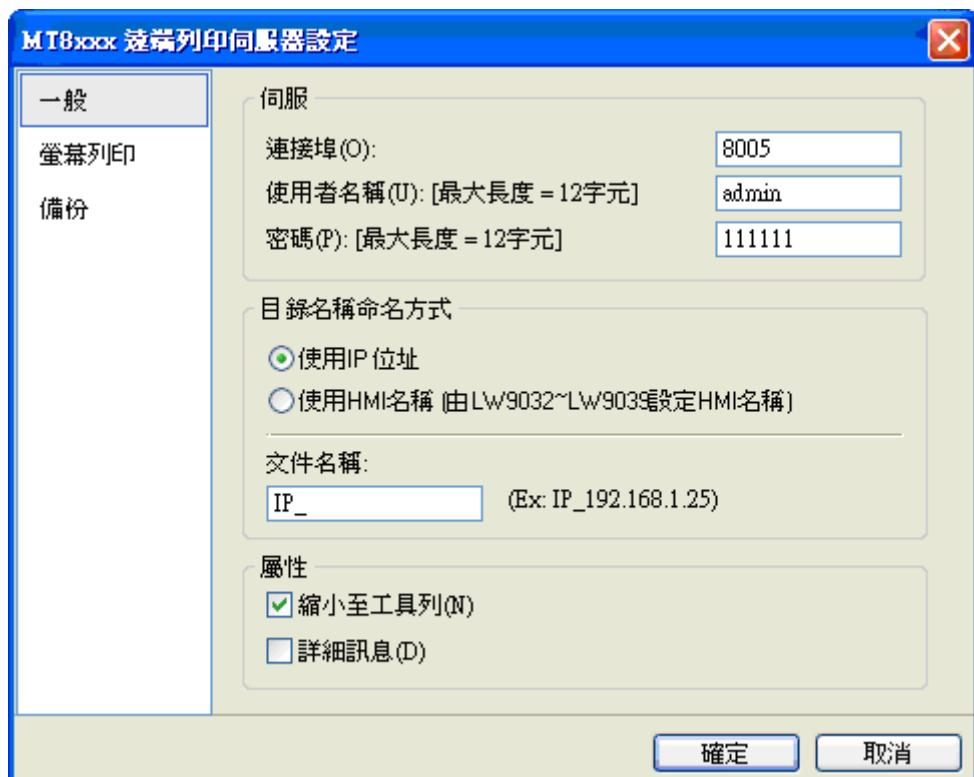


- [檢視] » [訊息視窗]

EasyPrinter 可以保留 **10,000** 筆訊息視窗中的訊息，當新訊息產生時，最舊的訊息會被刪除。

以下詳細說明 **[選項] » [設定]** 的各項設定和意義：

1. 一般：



• 伺服

[連接埠]

設定乙太網路接口讓人機連結，範圍 1~65535，8005 是預設值。

[使用者名稱] 及 [密碼]

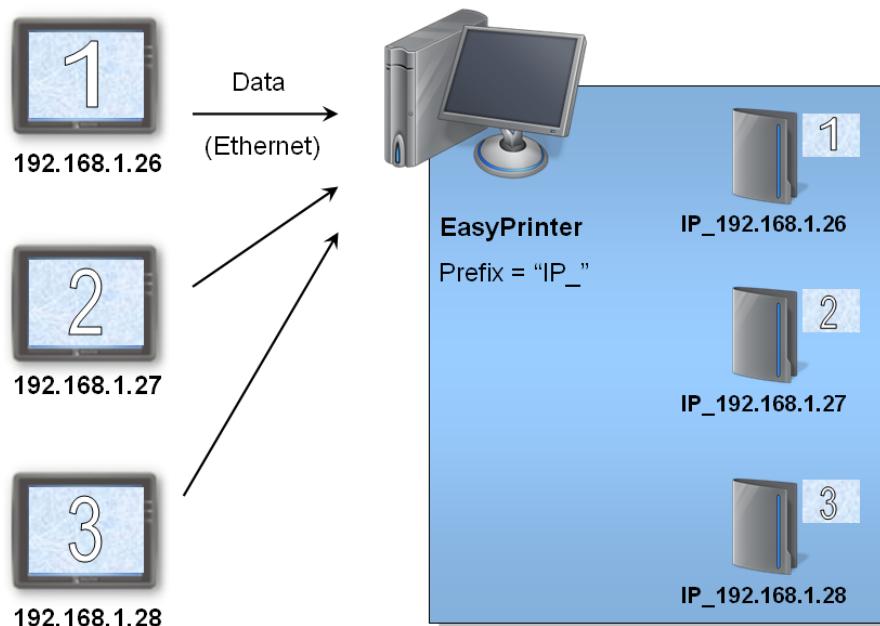
設定使用者名稱及密碼，讓有授權的人機方能使用 EasyPrinter 功能。

- 目錄名稱命名方式

EasyPrinter 使用不同的資料夾儲存來自不同人機的各種檔案(螢幕列印點陣圖檔、備份檔等)。有兩種方式可以命名這些資料夾：

[使用 IP 位址]

在使用所設 IP 位址的人機送出指令後，EasyPrinter 會為資料夾命名為文件名稱+IP 位址。請見下圖：



[使用 HMI 名稱]

EasyPrinter 使用送出指令的 HMI 之名稱來為資料夾命名：[文件名稱]+[HMI 名稱]

- 屬性

[縮小至工具列]

勾選此項，EasyPrinter 會被縮小並置於工具列中，使用者只要雙擊工具列上的圖像，即可打開 EasyPrinter。

[詳細訊息]

勾選此項，訊息視窗中會顯示更詳細的事件訊息。

2. 螢幕列印：



EasyPrinter 提供兩種螢幕列印結果輸出模式：

- **輸出**

[輸出至](印表機)

選擇此項告知 EasyPrinter 將螢幕列印結果透過特定印表機列印出來。

[儲存至](檔案)

選擇此項告知 EasyPrinter 將螢幕列印結果轉成點陣圖檔，並儲存於指定路徑。使用者可以在以下路徑找到點陣圖檔：

[使用者指定路徑] \ [HMI 資料夾] \ yymmdd_hhmm.bmp

舉例來說，當一個螢幕列印指示發生於 17:35:00 2009 年 1 月 12 日，點陣圖檔將被命名為"090112_1735.bmp"。如果同一分鐘有另一個點陣圖檔產生，新圖檔將被命名為"090112_1735_01.bmp"，以此類推。

3. 備份：



- **輸出**

EasyPrinter 將備份檔案儲存於特定路徑下。

事件記錄歷史資料路徑：

[使用者指定路徑] \ [HMI 資料夾] \ [事件記錄] \ EL_yyyymmdd.evt

資料取樣歷史資料路徑：

[使用者指定路徑] \ [HMI 資料夾] \ [取樣資料紀錄] \ [資料取樣物件的檔案名稱] \ yyyymmdd.dtl

配方數據備份路徑：

[使用者指定路徑] \ [HMI 資料夾] \ [配方數據] \ recipe.rcp 或 recipe_a.rcp

- **轉換批次檔**

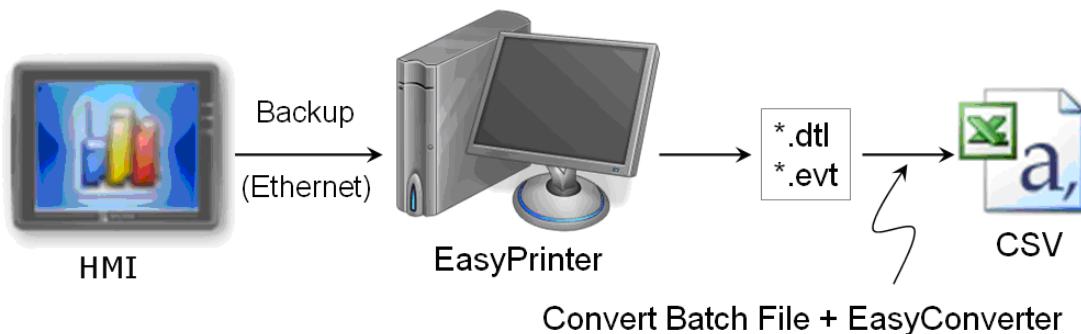
勾選 **【開啟】** 指定自動將上傳之歷史檔案轉檔成 csv 或 excel 檔之轉換批次檔。請參考下一小節。



- 使用者可以用暫存器 LW9032 至 LW9039 來指定 HMI 名稱。
- 若尚未設定 HMI 名稱，EasyPrinter 會使用 IP 位址來命名檔案夾。

26.4 轉換批次檔

EasyPrinter 提供一個轉檔裝置，可將上傳之資料取樣與事件記錄等兩種歷史檔案自動存成 CSV 檔。使用者若需要此功能，須先準備一個轉換批次檔，告知 EasyPrinter 如何轉換歷史檔案。



如上所示，此轉檔功能實際是由 EasyConverter 執行，EasyPrinter 只是遵照轉換批次檔的標準來用正確的參數啟動 EasyConverter 達成轉檔指示。



- EasyConverter 是另一個 Win32 應用程式，可將歷史資料轉換成 csv 或 Excel 的 xls 等檔案。使用者可在 EasyBuilder 下載路徑中找到這個程式。
- 使用者若需使用此功能，須先確定 EasyPrinter 及 EasyConverter 皆被放置於相同路徑下。

26.4.1 轉換批次檔預設值

以下為預設轉換批次檔：

程式碼 1 預設轉換批次檔 (convert2csv.def)

```

1: "dtl", "EasyConverter /c $(路徑名稱)"
2: "evt", "EasyConverter /c $(路徑名稱)"
  
```

檔案文字會以兩行呈現，每行含有兩個參數，用逗號分隔，形成對應特定類型檔案(資料取樣與事件記錄)的處理標準。第一個參數顯示該檔案類型的副檔名，第二個參數顯示操作模式所需執行的命令。“\$(路徑名稱)”是關鍵字，告訴 EasyPrinter 需用轉檔的備份檔案名稱來取代之。例如，資料取樣檔案名稱為 20090112.dtl，已被上傳及儲存，EasyPrinter 會輸出以下指令於命令視窗。

```
1: EasyConverter /c 20090112.dtl
```

如此一個名稱為 20090112.csv 的檔案即被建立。

因此，轉換批次檔的預設標準如下：

1. 轉換所有資料取樣歷史檔案(*.dtl)為 CSV 檔。
2. 轉換所有事件記錄歷史檔案(*.evt)為 CSV 檔。



- 事實上，第二個參數中的“\$(路徑名稱)”代表檔案的完整路徑名稱，在前面的例子中，EasyPrinter 以下面名稱取代：
[使用者指定路徑]\[HMI 資料夾]\[資料記錄]\[資料取樣物件檔案名稱]\20090112.dtl
- EasyPrinter 以一行檔案文字為單位來解讀轉換批次檔，也就是說，一行文字形成一個標準。
- 任兩個參數皆須以逗號分隔。
- 任一參數皆須以雙引號標示。
- 同一參數中切勿放逗號。
- 詳細介紹請見《第二十五章 EasyConverter》。

26.4.2 特定標準

- 可針對特定 HMI 上傳的檔案使用特別的操作，例如程式碼 2。
- 可用 HMI 的名稱來辨別該 HMI，例如程式碼 3。
- 或是針對不同的資料取樣歷史記錄需要不同的操作方式，例如程式碼 4。

(只適用在 [資料取樣] 檔案，且名稱為"Voltage"時。)

第三個參數("*)表示接受來自任何 HMI 的資料取樣當中符合標準者。

使用者可以轉換第三個參數為"192.168.1.26"，"192.168.1.*"，HMI 名稱等等，用以減少目標人機範圍。

程式碼 2 針對 HMI IP = 192.168.1.26 的特別定義標準

```
1: "dtl", "EasyConverter /c $(路徑名稱)", "192.168.1.26"
```

程式碼 3 針對 HMI 名稱為 Weintek_01 的特別定義標準

```
1: "dtl", "EasyConverter /c $(路徑名稱)", "Weintek_01"
```

程式碼 4 針對資料取樣物件檔案名稱 = Voltage 的特別定義標準

```
1: "dtl", "EasyConverter /s Voltage.lgs $(路徑名稱)", "*",  
"Voltage"
```

26.4.3 轉換批次檔格式

以下列出標準格式與各參數之說明：

檔案類型	指令(行)	HMI IP/名稱	條件 1	條件 2
------	-------	-----------	------	------

- 檔案類型
這個參數特定出此標準所針對的上傳檔案類型之副檔名(e.g. “dtl” 為資料取樣歷史檔案， “evt”為事件記錄歷史檔案)
- 指令(行)
當上傳檔案符合標準時，EasyPrinter 送至命令視窗的確切指令。
- HMI IP/名稱
此參數指定這個標準所針對的 HMI 。
- 條件 1
如果檔案類型是“dtl”， 此參數將這個標準所針對之**[資料取樣]**物件的檔案夾名稱指定出來。對其他檔案類型無效。
- 條件 2
未使用 (保留為以後使用)
-

26.4.4 執行

EasyPrinter 於每個檔案上傳後，由下往上驗證各標準。一旦符合標準，就會停止驗證，並開始處理下一個檔案。因此，使用者可將較廣泛的標準放在下方，而將較明確的標準置於上方。例如，以下為批次檔內容：

```
"dtl", "EasyConverter /c $(路徑名稱)"  
"evt", "EasyConverter /c $(路徑名稱)"  
"dtl", "EasyConverter /c $(路徑名稱)", "192.168.1.26"  
"dtl", "EasyConverter /c $(路徑名稱)", "my_HMI_01"  
"dtl", "EasyConverter /c $(路徑名稱)", "my_HMI_02"  
"dtl", "EasyConverter /s Voltage.lgs $(路徑名稱)", "*", "Voltage"
```

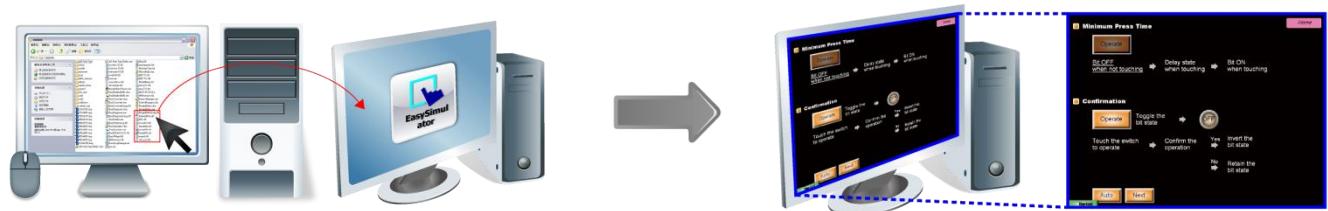
其正確的順序為 (由最後一行往上執行)：

```
"dtl", "EasyConverter /s Voltage.lgs $(路徑名稱)", "*", "Voltage"  
"dtl", "EasyConverter /c $(路徑名稱)", "my_HMI_02"  
"dtl", "EasyConverter /c $(路徑名稱)", "my_HMI_01"  
"dtl", "EasyConverter /c $(路徑名稱)", "192.168.1.26"
```

```
"dtl", "EasyConverter /c $(路徑名稱)"  
"evt", "EasyConverter /c $(路徑名稱)"
```

第二十七章 EasySimulator

EasySimulator 功能允許使用者可以不用安裝整個 EasyBuilder 軟體，即可執行連線/離線模擬，但使用者須先備好所需的相關檔案。



27.1 準備相關檔案

1. [driver] → [win32]
2. 320x234.bmp
3. 480x234.bmp
4. 480x272.bmp
5. 640x480.bmp
6. 800x480.bmp
7. 800x600.bmp
8. 1024x768.bmp
9. 234x320.bmp
10. 272x480.bmp
11. 480x800.bmp
12. 600x800.bmp
13. com.exe
14. EasySimulator.exe
15. gui.exe
16. xob_pos.def

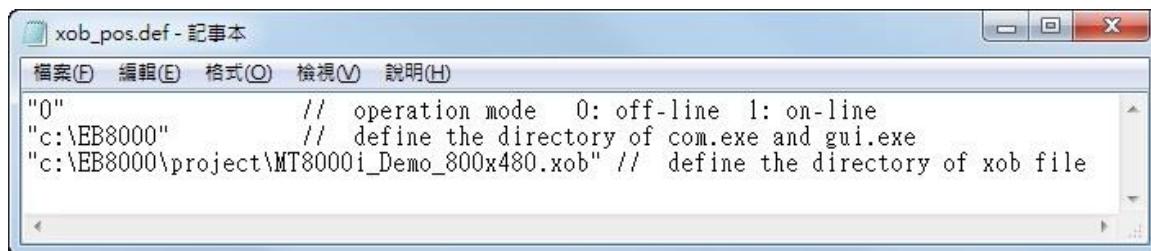


- 以上相關檔案皆可在 EasyBuilder 安裝目錄資料夾裡找到，所以使用者必須在某部電腦上安裝 EasyBuilder 軟體後，再將這些檔案複製到目標電腦上。

27.2 設定 xob_pos.def 內容

步驟 1

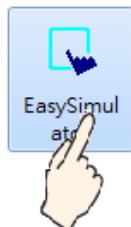
使用文字編輯工具開啟 **xob_pos.def** (例如：記事本)，並正確設定相對內容。



行數	敘述
1	“0” 執行離線模擬；“1” 執行連線模擬。
2	指出相關檔案的存放路徑。 (例如： com.exe , gui.exe , EasySimulator.exe ...等等)。
3	指出 xob 檔案的存放路徑。

步驟 2

雙擊 **EasySimulator.exe** 即可開始執行模擬。

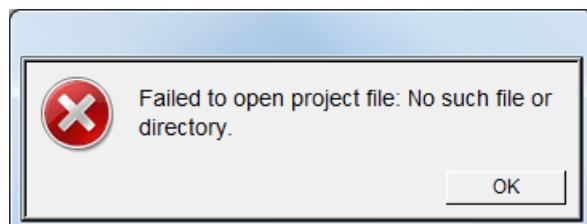


步驟 3

連線 / 紛線模擬結果將顯示在螢幕上。



- 若執行 **EasySimulator.exe** 後沒有反應，請再次確認相關檔案路徑是否定義正確。
- 若彈出下列視窗，表示 **xob** 檔路徑錯誤，請再次確認是否定義正確。



第二十八章 使用串列埠實現一機多屏功能（主從模式）

使用串列埠實現一機多屏是指：HMI 透過串列埠連接遠端的 HMI，並讀取連接在遠端 HMI 上 PLC 的數據，參考下圖。



上圖顯示 PLC 連接在 HMI 1 上，HMI 1 與 HMI 2 使用串列埠直接連接，HMI 2 可以透過 HMI 1 讀取 PLC 上的數據。

下面將以上圖的例子，說明如何使用 EasyBuilder 規劃 HMI 1 與 HMI 2 所使用的工程檔案，實現一機多屏的功能。

28.1 如何設定主機所使用工程檔案的內容

下圖為 HMI 1 所使用工程檔案【系統參數】中【設備清單】的內容。

設備清單：							
編號	名稱	位置	設備類型	介面類型	通訊協議	站	備註
本機 HMI	Local HMI	本機	MT6070IH/MT8070...	停用	N/A	N/A	
本機 PLC 1	FATEK FB Series	本機	FATEK FB Series	COM 1 (9600,E,7,1)	RS232	1	
本機 同服器	Master-Slave Se...	本機	Master-Slave Se...	COM 3 (115200,E,8,1)	RS232	N/A	

步驟 1

因為 HMI 1 的 COM 1 連接 PLC，所以設備清單中需存在【本機 PLC1】，並設定正確的 PLC 通訊參數。假設此時所連接的 PLC 為 FATEK FB Series。

步驟 2

因 HMI1 的 COM 3 用來接收來自 HMI2 的命令，所以必須建立【Master-Slave Server】類型的裝置，用來設定 COM 3 的屬性。

由上圖可以發現 COM 3 的通訊參數為 [115200,E,8,1]，並使用 RS-232 介面。此項參數並不限定需與 PLC 的通訊參數相同，但限制數據位元必須為 8。另外，盡可能設定為較快的通訊速度，這樣 HMI 2 可以較有效率讀取到 PLC 的數據。

28.2 如何設定從機所使用工程檔案的內容

設備清單：				
編號	名稱	位置	設備類型	介面類型
本機 HMI	Local HMI	本機	MT8121T (800 ...)	停用
*遠端 PLC 1	FATEK FB Series	COM 1 (主從模式)	FATEK FB Series	COM1(115200,E,8,1)

上圖為 HMI 工程檔案 **【系統參數】/【設備清單】** 的設定內容。因為 HMI 2 所讀取的 PLC 連接在 HMI 1 上，所以 HMI 2 將 PLC 視為遠端 PLC，因此在設備清單中需存在 **[*遠端 PLC 1]**，此時所連接的 PLC 為 FATEK FB Series。下文說明如何建立 **[*遠端 PLC 1]**。

步驟 1

在設備清單中建立一個新的裝置，**[PLC 類型]** 請選擇 **[FATEK FB Series]**，**[PLC 預設站號]** 需與 PLC 所使用的站號相同。



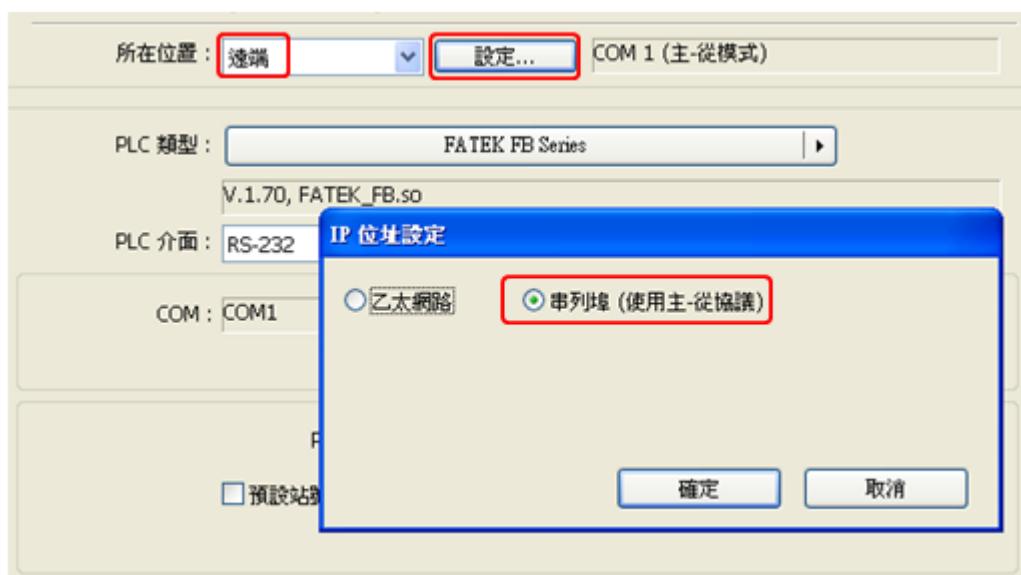
步驟 2

設定正確的通訊參數。此時 HMI 2 的 COM 1 是與 HMI 1 的 COM 3 相互連接，並不是與 PLC 直接連接，因此必須忽略 PLC 的通訊參數，而應讓 HMI 2 的 COM 1 與 HMI 1 的 COM 3 所使用的介面與通訊參數相同。因為 HMI 1 的 COM 3 使用 RS-232 介面，通訊參數為 **[115200,E,8,1]**，所以 HMI 2 的 COM 1 也需依此參數設定，參考右圖。



步驟 3

因為 HMI 2 視 PLC 為遠端 PLC，所以需選擇【所在位置】為【遠端】。並選擇使用【串列埠】的方式連接遠端 HMI（即 HMI 1）。



設備清單：

編號	名稱	位置	設備類型	介面類型
本機 HMI	Local HMI	本機	MT8121T (800...)	停用
*遠端 PLC 1	FATEK FB Series	COM 1 (主從模式)	FATEK FB Series	COM1(115200,E,8,1)

步驟 4

完成上述的各項步驟後，在【設備清單】中可以發現新增一項設備：【*遠端 PLC 1】。此設備名稱包含「*」符號，用來表示即使名稱中包含“遠端”，但實際上仍由本機的串列埠發送命令與接收回覆，所以與 PLC 的連接狀態只需檢視本機的系統保留位址即可；也就是【*遠端 PLC 1】、【*遠端 PLC 2】、【*遠端 PLC 3】與【本機 PLC 1】、【本機 PLC 2】、【本機 PLC 3】使用相同的系統保留位址。這些系統保留位址包含：

位址	敘述
LB-9150	狀態為 ON 時，若與連接在 COM 1 的 PLC 斷線，系統將自動連線。 狀態為 OFF 時，忽略與此 PLC 的斷線狀態。
LB-9151	狀態為 ON 時，若與連接在 COM 2 的 PLC 斷線，系統將自動連線。 狀態為 OFF 時，忽略與此 PLC 的斷線狀態。
LB-9152	狀態為 ON 時，若與連接在 COM 3 的 PLC 斷線，系統將自動連線。 狀態為 OFF 時，忽略與此 PLC 的斷線狀態。

LB-9200~ LB-9455	這些暫存器用來指示與連接在 COM 1 的 PLC 間的連線狀態。 LB-9200 指示與站號為 0 的 PLC 的連線狀態，LB-9201 指示與站號為 1 的 PLC 的連線狀態，依此類推。 狀態為 ON 表示目前連線正常。 狀態為 OFF 表示目前與 PLC 為斷線狀態，此時可以將此狀態重設為 ON，系統將嘗試與 PLC 再連線一次。
LB-9500~ LB-9755	這些暫存器用來指示與連接在 COM 2 的 PLC 間的連線狀態。 LB-9500 指示與站號為 0 的 PLC 的連線狀態，LB-9501 指示與站號為 1 的 PLC 的連線狀態，依此類推。 狀態為 ON 表示目前連線正常。 狀態為 OFF 表示目前與 PLC 為斷線狀態，此時可以將此狀態重設為 ON，系統將嘗試與 PLC 再連線一次。
LB-9800~ LB-10055	這些暫存器用來指示與連接在 COM 3 的 PLC 間的連線狀態。 LB-9800 指示與站號為 0 的 PLC 的連線狀態，LB-9801 指示與站號為 1 的 PLC 的連線狀態，依此類推。 狀態為 ON 表示目前連線正常。 狀態為 OFF 表示目前與 PLC 為斷線狀態，此時可以將此狀態重設為 ON，系統將嘗試與 PLC 再連線一次。

28.3 如何連結從機的 MT500 工程

目的是讓 MT500 可以使用 EasyBuilder 的主從通訊協議以令 MT8000 系列的 Local Data 和 MT500 所連接的 PLC Data 可以資訊交換。

■ EasyBuilder 設定方式

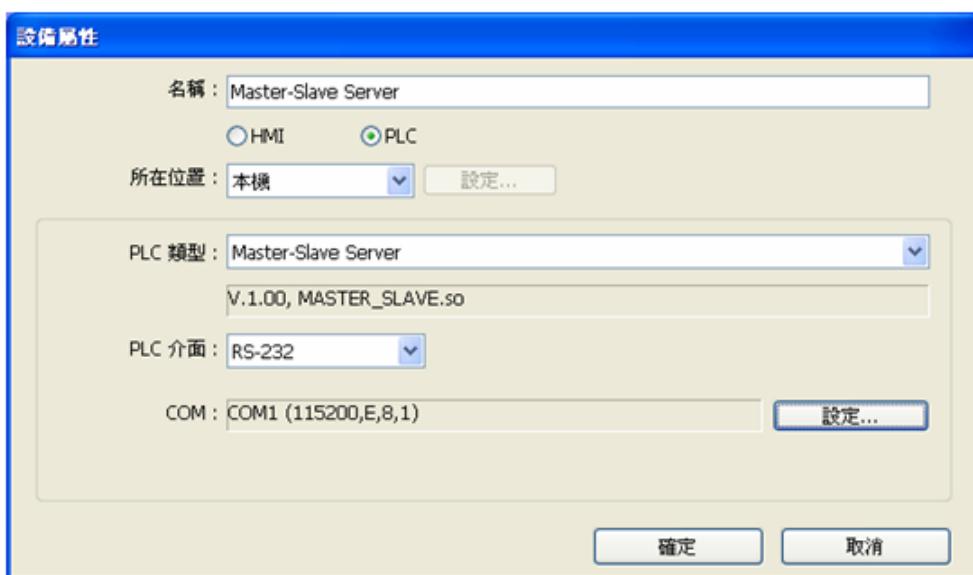
步驟 1

選取 [Master-Slave Server] 驅動並點擊 [設定]。若有連接 PLC，則依照原本設定方式即可。



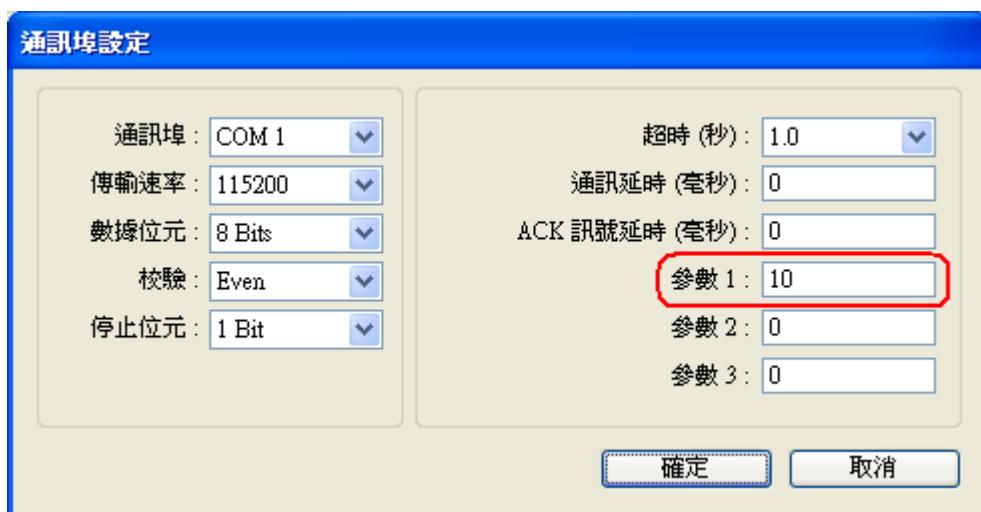
步驟 2

選取 RS-232 並點擊 [設定]。



步驟 3

參數 1 要填入 MT500 PLC ID No. (請參考 EB500 設定)。



■ EB500 設定方式

步驟 1

在 EB500 的系統參數設定內設定多台人機: Slave，人機間連結速度: 115200。

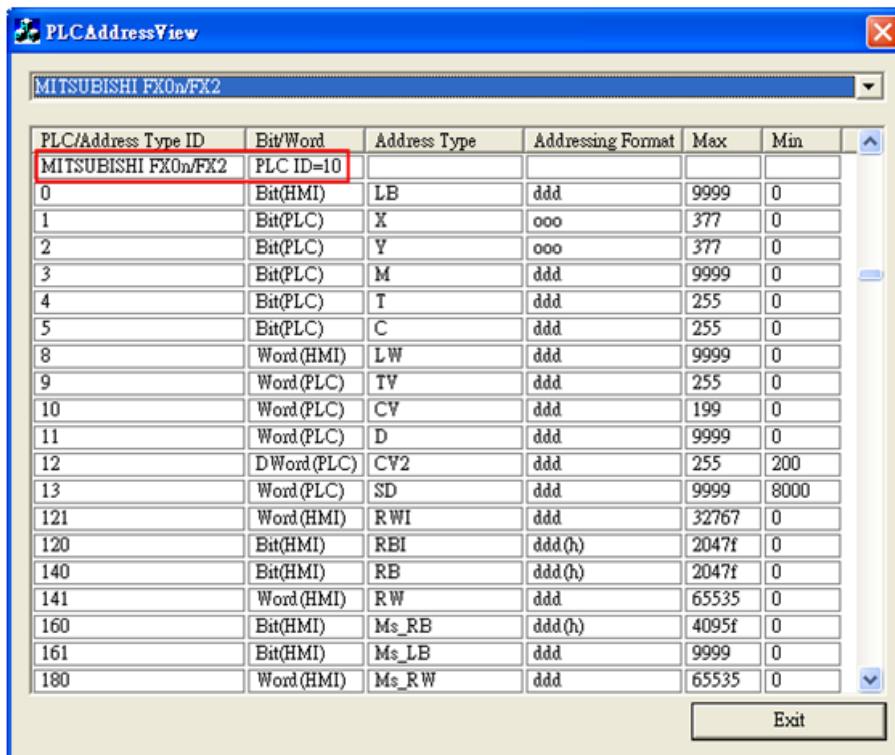




- Baud rate 設定在 EB500 及 EasyBuilder 要相同。

步驟 2

雙擊 PLC Address View.exe 來查詢 PLC 的 ID No. 並填入至 EasyBuilder 的參數 1。



步驟 3

HMI 之間使用串列埠 RS-232 連接後即可通訊。

可使用之裝置位址：

設備位址	MT500	EasyBuilder	範圍
位元	Ms_RB	RW_Bit	dddd: 0~4095 (h): 0~f
位元	Ms_LB	LB	ddd: 0~9999
字元	Ms_RW	RW	ddddd: 0~65535
字元	Ms_LW	LW	ddd: 0~9999

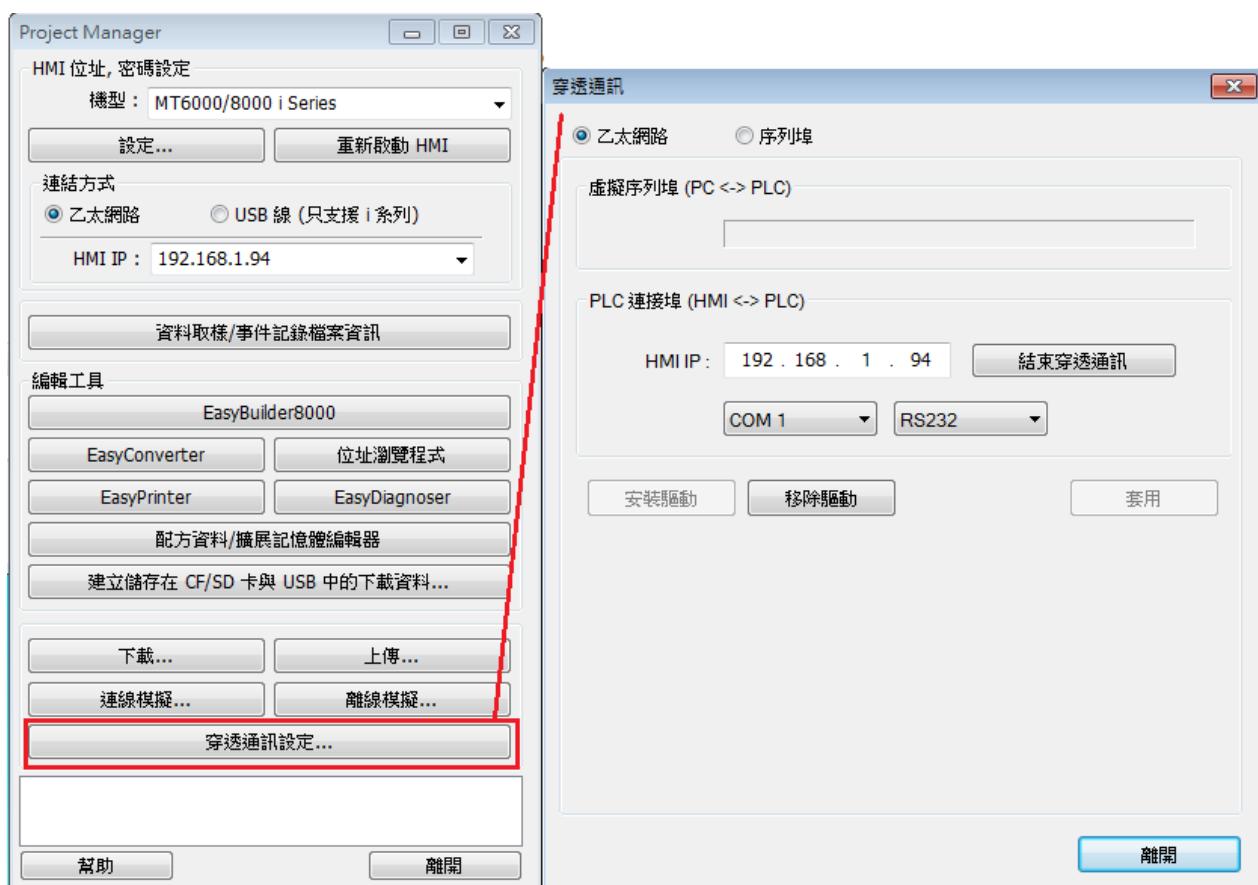


- 因為 MT500 參數設定一定會有某台 PLC 被選定，因此即使只要讀寫 EB8000 的 Local Data，仍需要將 EB500 參數設定所選中的 PLC ID 填入到 EasyBuilder 的參數 1。
- 以下方式，主從模式不適用。當 MT500 選用 S7-200、S7-300 驅動時，因 MT500 將高低位元互換，而導致 MT500 讀取 MT8000 的 Local Data 時，發生錯誤。

第二十九章 穿透通訊功能

EasyBuilder 所提供的穿透通訊功能允許 PC 上的應用程式透過 HMI 直接控制 PLC，此時 HMI 所扮演的角色類似轉接器的功能。

穿透通訊功能包含 【序列埠】 模式與 【乙太網路】 模式，點擊 Project Manager 的 【穿透通訊設定】 按鈕，即可檢視這兩種模式的設定內容。



29.1 乙太網路模式

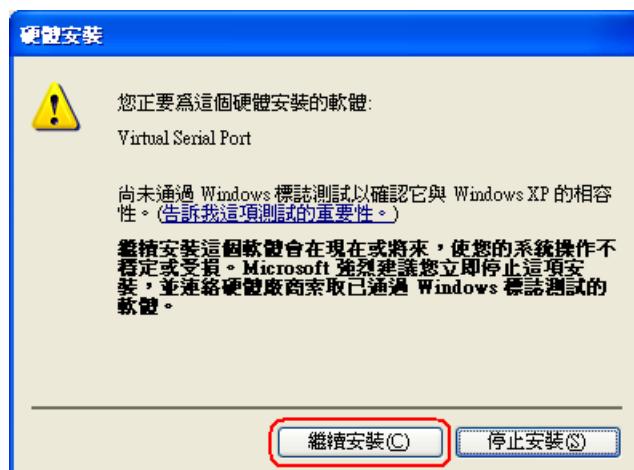
29.1.1 如何安裝虛擬序列埠驅動程式

在使用 [乙太網路] 穿透通訊功能前，要先安裝虛擬序列埠驅動程式，安裝步驟如下：

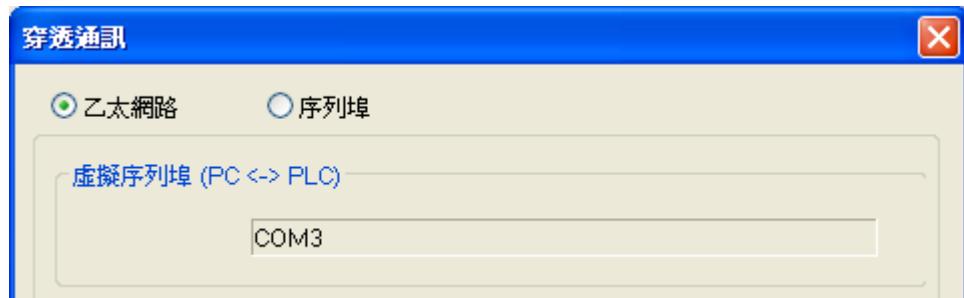
首先開啟 Project Manager 檢視目前驅動程式的安裝狀態，若畫面顯示 **【請安裝虛擬序列埠驅動程式】**，請點擊 **【安裝驅動】** 按鈕，參考下圖。



在安裝驅動程式的過程中若出現下列的畫面，請選擇 **【繼續安裝】**。



在完成安裝程序後，原先顯示【請安裝虛擬序列埠驅動程式】的位置將顯示目前所使用的虛擬序列埠號，下圖顯示目前所使用的虛擬序列埠為 COM 3。



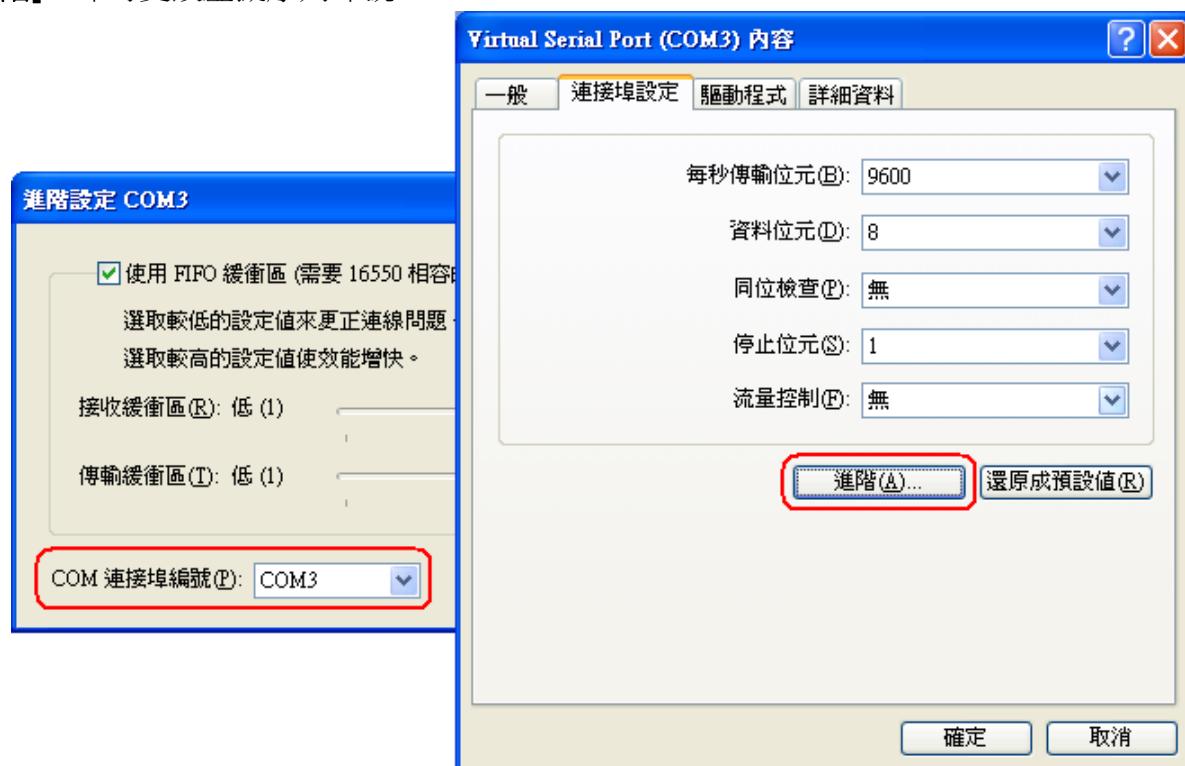
- 此驅動不支援於 Windows 7 64 位元作業系統。

29.1.2 如何更改虛擬串列埠

在【裝置管理員】的內容中可發現已安裝完成 **[Virtual Serial Port]**。



若要更改虛擬序列埠號，只需進入 **[Virtual Serial Port]** 的內容，並選擇 **【連接埠設定】** 下的 **【進階】**，即可更改虛擬序列埠號。



29.1.3 如何使用乙太網路穿透通訊功能

在安裝完成虛擬序列埠驅動程式後，只需依照下面步驟即可使用網路穿透通訊功能。

步驟 1

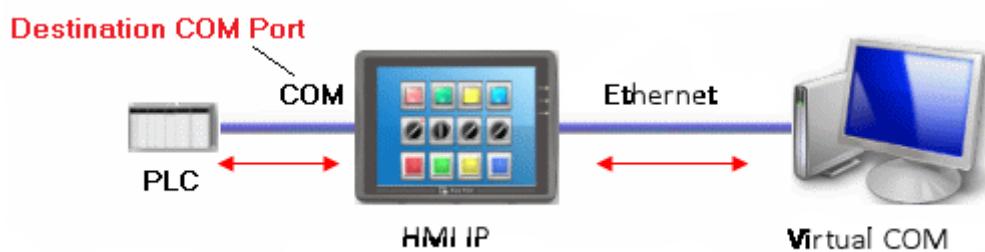
請設定連接 PLC 的 HMI IP 位址。

步驟 2

指定 HMI 通訊埠、HMI 連接 PLC 的序列埠與屬性。

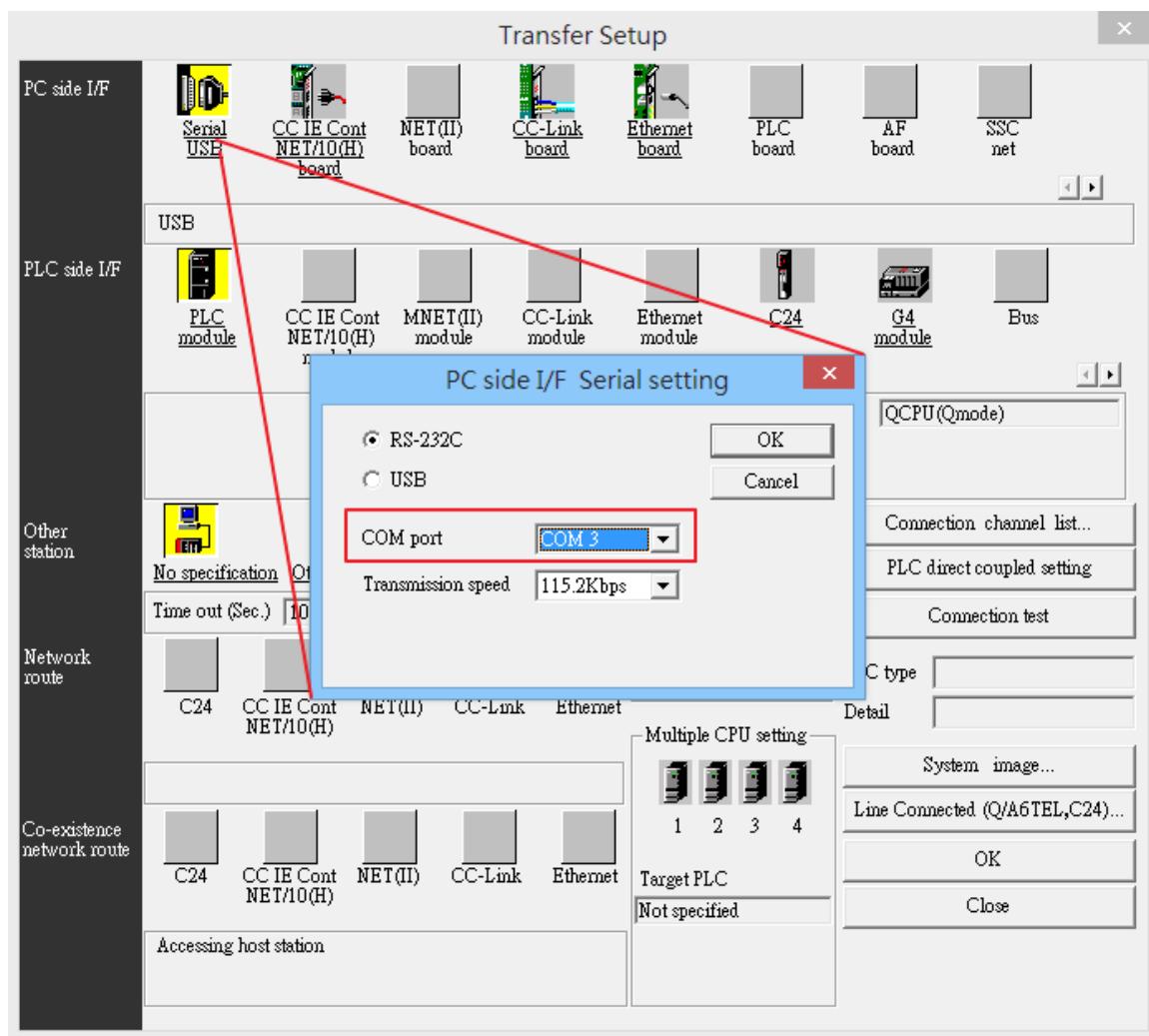
步驟 3

完成所有設定後需按下 **【套用】** 按鈕，所有屬性才會生效。

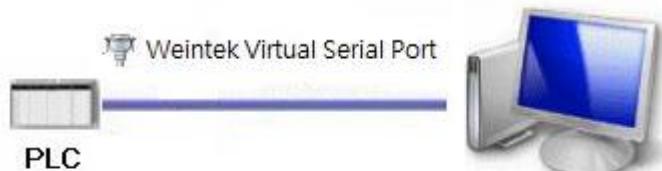


步驟 4

在執行 PC 上的應用程式時，所使用的序列埠號需指向虛擬序列埠號。以 Mitsubishi 的應用程式為例，若此時的虛擬序列埠為 COM 3，則在 **[PC side I/F Serial setting]** 視窗中的 **[COM port]** 需選擇 COM 3，請參考下圖。

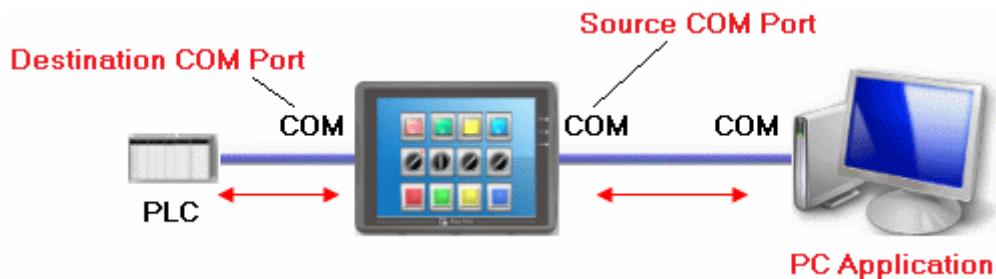


完成上述各項設定後，使用者在執行 PC 上的 PLC 應用程式時，HMI 會自動切換為穿透通訊模式，此時可以將應用程式視為直接使用虛擬序列埠控制 PLC。在關閉應用程式時，HMI 也會自動關閉穿透通訊模式。



- 當使用乙太網路穿透模式時，目前有些驅動可同時支援 HMI 與 PLC 間的通訊，但並非全部驅動。另外系統提供暫存器 [LW-9903: 穿透通訊控制 (0：正常, 1：暫停, 2：執行穿透功能時，停止 HMI 與 PLC 間的通訊)]，可控制乙太網路模式下的功能。

29.2 序列埠模式



【數據來源串口】是指 HMI 與 PC 的連接埠。

【數據目標串口】是指 HMI 與 PLC 的連接埠。

在使用 【序列埠】 穿透通訊功能前，需先正確設定這兩個串口的屬性。

29.2.1 序列埠設定

EasyBuilder 提供兩種方式可啟動 【序列埠】 穿透通訊功能。

(1): 使用 Project Manager

(2): 使用系統暫存器

LW-9901 設定數據來源串口 (1 ~ 3: COM 1 ~ COM 3)

LW-9902 設定數據目標串口 (1 ~ 3: COM 1 ~ COM 3)

使用 Project Manager 開啟序列埠穿透通訊功能

下面說明序列埠穿透通訊功能設定頁的各項功能。



[HMI IP]

需指定 HMI 的 IP 位址。

[讀取 HMI 通訊參數設定]

此項功能用來讀取 HMI 上數據來源串口與數據目標串口的各項設定值。在按下 [讀取 HMI 通訊參數設定] 按鈕後，所有通訊參數將被更新。

[數據來源 COM 埠]、[數據目標 COM 埠]

用來顯示與設定數據來源串口與數據目標串口的各項通訊參數。當點選 [開始穿透通訊]，將使用 [數據來源 COM 埠]、[數據目標 COM 埠] 中所設定的內容，執行穿透通訊功能。

通常 [數據來源 COM 埠] 與 [數據目標 COM 埠] 中的 [傳輸速率]、[數據位元]、[校驗]、[停止位元] 需設為相同。[數據來源 COM 埠] 因為是連接到 PC，通訊模式通常選擇為“RS-232”即可；[數據目標 COM 埠] 則是因為接到 PLC，所以通訊模式需依照 PLC 的設定來決定，可以選擇“RS-232”、“RS-485 2W”或“RS-485 4W”。



- 當使用完序列埠模式的穿透通訊功能，需點選【結束穿透通訊】來關閉穿透通訊功能，此時 HMI 才會重新開啟和 PLC 的通訊。

29.2.2 HMI 工作模式

共有三種模式可顯示目前 HMI 的工作狀態。

設定	敘述
未知	在未讀取 HMI 的設定值前，所顯示的 HMI 工作模式為”未知”。
正常模式	在讀取 HMI 的設定值後，工作模式如為 ”正常模式”，表示 HMI 處在正常通訊狀態，不接受來自數據來源串口的任何數據。
穿透模式	若工作模式為 ”穿透模式”，表示 HMI 目前處在穿透模式狀態，此時 PC 上的應用程式可以透過數據來源串口直接控制連接在數據目標串口上的 PLC。

29.3 使用系統暫存器啟動穿透通訊功能

另一種啟動 HMI 穿透通訊功能的方式為直接更改系統暫存器 LW-9901 (數據來源串口) 與 LW-9902 (數據目標串口) 中的數據內容。當 LW-9901 與 LW-9902 中的數據符合下列條件時，HMI 將自動啟動穿透通訊功能：

- LW-9901 與 LW-9902 中的數據需為 1 ~ 3 (1 ~ 3 分別表示 COM 1 ~ COM 3)。
- LW-9901 與 LW-9902 中的數據不可相同。

如有需要更改各串口的通訊參數，只需更改各參數相對應的系統暫存器中的數據，並對 [LB-9030 : 更新 COM 1 通訊參數]、[LB-9031 : 更新 COM 2 通訊參數]、[LB-9032 : 更新 COM 3 通訊參數] 送出 ON 的訊號即可。

位址	敘述
LB-9030	更新 COM 1 通訊參數 (設定為 ON)
LB-9031	更新 COM 2 通訊參數 (設定為 ON)
LB-9032	更新 COM 3 通訊參數 (設定為 ON)
LW-9550	(16bit) : COM 1 模式 (0:RS232,1:RS485 2W,2:RS485 4W)
LW-9551	(16bit) : COM 1 串列傳輸速率

	(7:1200,8:2400,0:4800,1:9600,2:19200,3:38400,4:57600,..)
LW-9552	(16bit) : COM 1 數據位元 (7 : 7 bits, 8 : 8 bits)
LW-9553	(16bit) : COM 1 校驗 (0:none, 1:even, 2:odd, 3:mark, 4:space)
LW-9554	(16bit) : COM 1 停止位元 (1 : 1 bit, 2 : 2 bits)
LW-9555	(16bit) : COM 2 模式 (0:RS232,1:RS485 2W,2:RS485 4W)
LW-9556	(16bit) : COM 2 串列傳輸速率 (7:1200,8:2400,0:4800,1:9600,2:19200,3:38400,4:57600,..)
LW-9557	(16bit) : COM 2 數據位元 (7 : 7 bits, 8 : 8 bits)
LW-9558	(16bit) : COM 2 校驗 (0:none, 1:even, 2:odd, 3:mark, 4:space)
LW-9559	(16bit) : COM 2 停止位元 (1 : 1 bit, 2 : 2 bits)
LW-9560	(16bit) : COM 3 模式 (0:RS232, 1:RS485 2W)
LW-9561	(16bit) : COM 3 串列傳輸速率 (7:1200,8:2400,0:4800,1:9600,2:19200,3:38400,4:57600,..)
LW-9562	(16bit) : COM 3 數據位元 (7 : 7 bits, 8 : 8 bits)
LW-9563	(16bit) : COM 3 校驗 (0:none, 1:even, 2:odd, 3:mark, 4:space)
LW-9564	(16bit) : COM 3 停止位元 (1 : 1 bit, 2 : 2 bits)



- 若要關閉 HMI 的穿透通訊功能，只需將 LW-9901 與 LW-9902 中的數據更改為 0。

第三十章 工程檔案保護功能

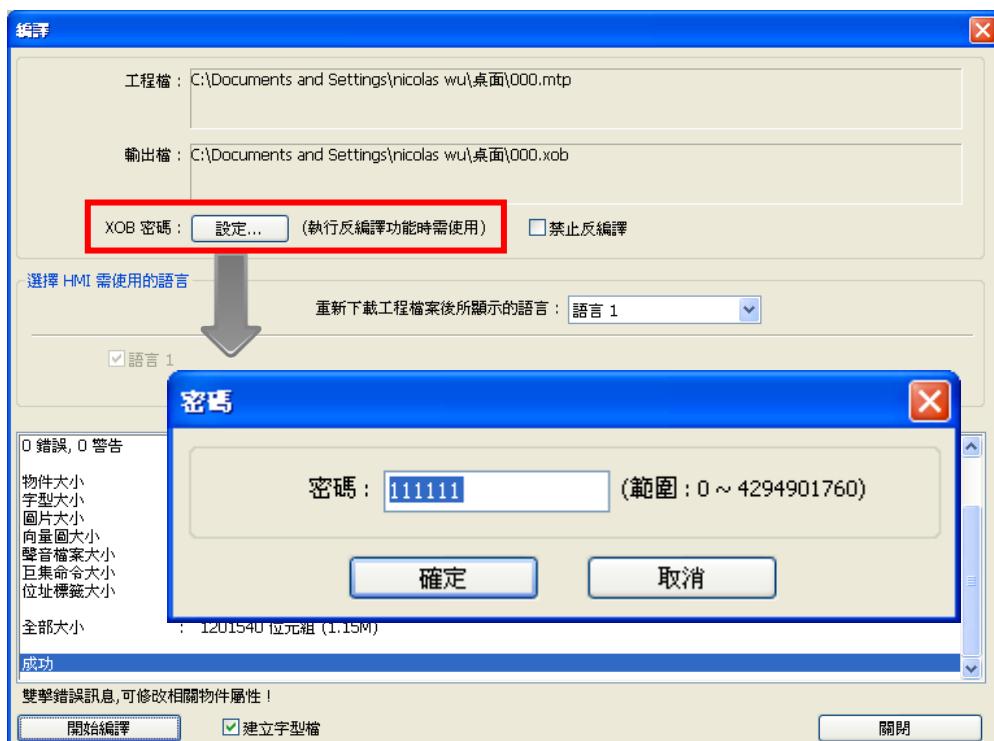
由於程式設計是來自使用者智慧的組合，因此 EasyBuilder 提供許多工程檔案的保護功能，來保障用戶的設計成果。



- 以下的保護功能若使用者不慎忘記密碼時，因已由用戶自行加密，所以原廠也無法解開，請妥善保管密碼。

30.1 XOB 密碼

在編輯完成工程檔案 (mtp) 後，可透過系統提供的編譯功能，將 mtp 工程檔案編譯為下載至 HMI 所需的 xob 檔案。用戶可在編譯視窗設定 **[XOB 密碼]** (範圍: 0 ~ 4294967295)，若之後要將此 xob 反編譯成 mtp 工程檔案，必需輸入此密碼才能完成反編譯。若反編譯時密碼輸入錯誤三次，請重新啟動 EasyBuilder。



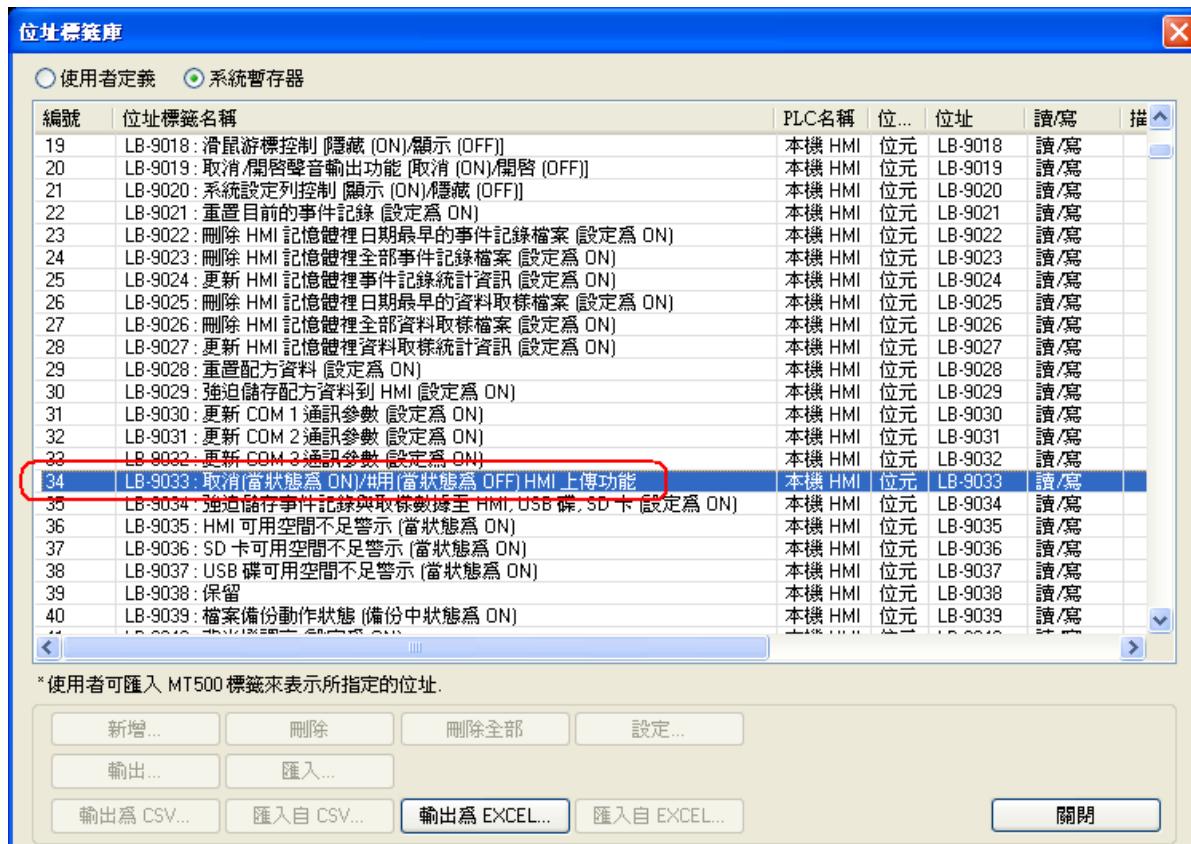
30.2 禁止反編譯

在編輯完成工程檔案 (mtp) 後，可透過系統提供的編譯功能，將 mtp 工程檔案編譯為下載至 HMI 所需的 xob 檔案。用戶可在編譯視窗設定 **【禁止反編譯】**，系統將忽略 **[XOB 密碼]** 的設定，且此 xob 檔案將無法反編譯至 mtp 工程檔案。



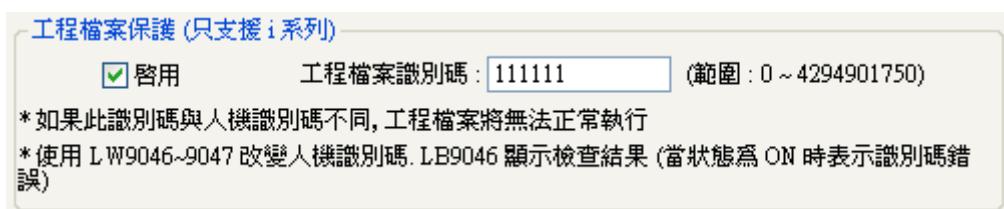
30.3 禁止 XOB 檔案上傳功能

EasyBuilder 提供系統暫存器 LB-9033，當此位址被設定為 ON 時，系統將取消 XOB 檔案上傳功能。當變更設定時，必須重新啟動 HMI 以更新設定值。若嘗試著去上傳一個被設定此功能的 xob 檔案，上傳後得到的 xob 檔案大小將為 0 位元組，亦無法反編譯。



30.4 工程檔案識別碼

用戶的工程檔案可以被限制只能在特定的 HMI 上執行，下圖為【系統參數設定】»【一般屬性】頁籤中【工程檔案保護】的設定畫面。



當啟用【工程檔案保護】功能時，可設定【工程檔案識別碼】(範圍: 0 ~ 4294901750)，且需搭配系統暫存器 LW-9046 ~ LW-9047 (32-bit) 設定 HMI 的【人機識別碼】，其數據無法被讀取或遠端寫入。而編譯後所得到的 .xob 檔案只能在【人機識別碼】與【工程檔案識別碼】相同的 HMI 上執行。若【人機識別碼】與【工程檔案識別碼】不相同時，LB-9046 的狀態將被設定為 ON。當變更【人機識別碼】設定時，必須重新啟動 HMI 以更新設定值。



- 若【人機識別碼】與【工程檔案識別碼】不相同時，HMI 與 PLC 之間將無法通訊。



下載範例程式前，請先確定已連上網路線。

30.5 MTP 密碼

在編輯完成工程檔案 (mtp) 後，可以設定保護 mtp 工程檔案的密碼，請至 **【系統參數設定】» [使用者密碼]** 頁籤設定此功能 (範圍: 1 ~ 4294967295)。啟用此功能後，每次欲開啟該 mtp 工程檔案時，須先輸入正確的密碼後才能開啟並編輯。



- 在使用 **【窗口複製】** 功能時，若欲複製的來源工程檔案設有 MTP 密碼保護時，須先輸入正確的密碼後，系統才能執行窗口複製功能。

第三十一章 Memory Map 通訊協定

31.1 簡介

Memory Map 通信協定類似於 IBM 3764R 通信協定，使用於對應記憶體資料的變化量較少的場合，且專為兩台設備交換資料的通信協定。Memory Map 通信協定的特徵是兩台設備必須一方為主方 (master)，另一方為從方 (slave)。在一般情況下，主方和從方並沒有建立通信，只有當某一方所指定的記憶體資料變化時，通信才建立。而雙方資料一致後，通信中斷。所以通信的目的是保持兩台設備 (主方和從方) 之間相對應的一塊相同大小記憶體資料的一致性。

其中主方和從方中對應的記憶體需和 MW(MB) 記憶體具有相同的性質和大小。人機中的 MW(MB) 大小皆為 10,000 字。

MB 和 MW 都是指向相同的暫存器區塊，例如：即 MB0~MBf 對應到 MW0 的各位元，MB10~MB1f 對應到 MW1，如下表：

設備名稱	格式	範圍
MB	DDDDh	DDDD:0~4095, h: 0~F(hex)
MW	DDDD	DDDD:0~9999

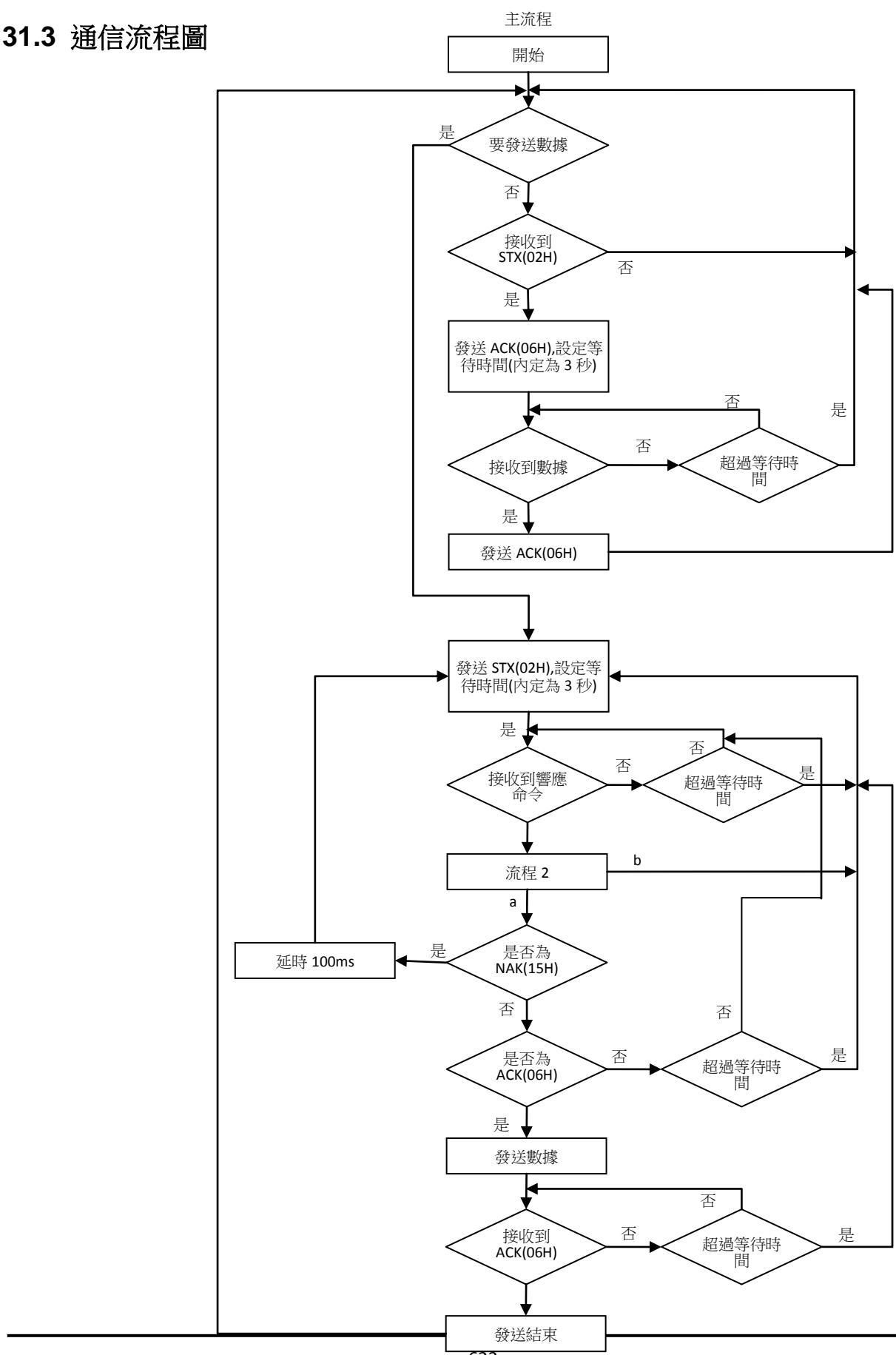
31.2 接腳設定

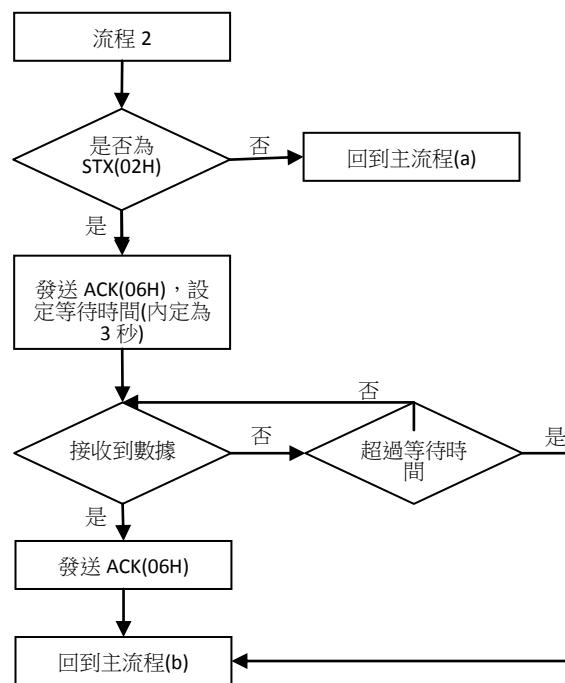
使用 Memory Map 通信協定時，主方和從方必須使用相同的通信參數。其接線方式如下：

介面	RS232	
設備	Master	Slave
對應腳位	TX(#)	RX(#)
	RX(#)	TX(#)
	GND(#)	GND(#)
介面	RS485(4W)	
設備	Master	Slave
對應腳位	TX+(#)	RX+(#)
	TX-(#)	RX-(#)
	RX+(#)	TX+(#)
	RX-(#)	TX-(#)
	GND(#)	GND(#)

注意：#表示由具體 PLC 或控制器決定。

31.3 通信流程圖





- 流程 2 對從方有效，對主方無效。
- STX 為通信請求信號，ACK 為回應請求信號，NAK 為忙信號。

31.4 通信資料格式

通信資料格式可分為兩種，MB 指令和 MW 指令。

對 MB 操作的指令格式，請參見下表：

MB 指令		
偏移量 (位元組)	格式	描述
0	0x02	對 MB 操作的標誌
1 2	0x## 0x##	位址(低位元組) 位元位址(高位元組) 如果是 MB-18，則 $1*16+2=18$ ，為 0x12，0x00
3	0x00 (或 0x01)	表示所指定 MB 位址的資料內容(因為是 Bit 類型，只能是 0 或 1)
4、5	0x10、0x03	結束標誌
6	0x##	校驗和；從偏移量 0 到 5 之位元組進行 XOR 運算。

對 MW 操作的指令格式，請參見下表：

MW 指令		
偏移量 (位元組)	格式	描述
0	0x01	對 MW 操作的標誌
1 2	0x## 0x##	位址(低位元組) 位元位址(高位元組) 如果位址資料中包含一個 0x10，則在 0x10 後再插入一個 0x10，位址表示多出一個位元組，命令格式相應的向後推移一個位元組，例如位址為 0x10，0x04，則變為 0x10，0x10，0x04
3	0x##	傳送的位元組數(由於對字操作，位元組數一定為偶數)，如果位元組數為 0x10，則在 0x10 後再插入一個 0x10 命令格式相應的向後推移一個位元組
4~4+n-1	0x##(L),0x##(H) 0x##(L),0x##(H) ...	為 1,2 位元組所對應位址為起始位址的資料，其中 n 為資料的位元組數，如果資料中有 0x10，則在 0x10 後再插入一個 0x10，而“傳送位元組數”不變，n 則為 n+1，以次類推。
4+n， 4+n+1	0x10、 0x03	結束標誌
4+n+2	0x##	校驗和；對前面所有位元組 xor。

31.4.1 通信範例

Example 1

假設主方將 MW-3 的內容設為 0x0a，因為資料更動，主方立刻會和從方建立通信，而從方接收到資料後把它對應的 MW-3 的內容更新為 0x0a。過程為：

1. 主方發送 STX(0x02h)。
2. 從方接收到主方發送的 STX(0x02h)後，發送返回命令 ACK(0x06h)。
3. 主方接收到從方的返回命令 ACK(0x06h)。
4. 主方發送資料 0x01，0x03，0x00，0x02，0x0a，0x00，0x10，0x03，0x19，如下表所示：

偏移量(位元組)	格式	描述
0	0x01	對 MW 操作的標誌
1	0x03	位址(低位元組)
2	0x00	位元位址(高位元組)
3	0x02	傳送的位元組數(MW3 為兩個位元組)
4，5	0x0a，0x00	MW-3 的內容為 0x0a，0x00
6，7	0x10，0x03	結束標誌
8	0x19	校驗和， $0x01 \oplus 0x03 \oplus 0x00 \oplus 0x02 \oplus 0x0a \oplus 0x00 \oplus 0x10 \oplus 0x03 = 0x19$

5. 從方收到主方發送的資料後，發送返回命令 ACK(0x06h)。
6. 主方接收到從方的返回命令 ACK(0x06h)。

通信完成，主方把更改的 MW 的位址和內容傳送給了從方，從方再更改 MW 的資料，使得主方和從方對應節點位址內容保持一致。

Example 2

如果位址和資料中包括 0x10 的情況，請注意觀察資料格式的變化。

我們假設從方把 MW-10 的內容設為 0x10，根據這個協定，從方立刻會和主方建立通信，從而使得主方接收到資料後把它對應的 MW-10 的內容置為 0x10。過程為：

1. 從方發送 STX(0x02h)。
2. 主方接收到從方發送的 STX(0x02h)後，發送返回命令 ACK(0x06h)。
3. 從方接收到主方的返回命令 ACK(0x06h)。
4. 從方發送資料 0x01，0x10，0x10，0x00，0x02，0x10，0x10，0x00，0x10，0x03，0x10 如下表所示：

偏移量 (位元組)	格式	描述
0	0x01	對 MW 操作的標誌
1	0x10	位址(低位元組)
2	0x10	插入一個 0x10 位元組
3	0x00	位元位址(高位元組)
4	0x02	傳送的位元組數(MW-10 為兩個位元組)
5	0x10	MW-10 的低位元組內容為 0x10
6	0x10	插入一個 0x10 位元組
7	0x00	高位元組內容為 0x00
8、9	0x10、 0x03	結束標誌
10	0x10	校驗和， $0x01^0x10^0x10^0x00^0x02^0x10^0x10^0x00^0x10^0x03=0x10$

5. 主方收到從方發送的資料後，發送返回命令 ACK(0x06h)。
6. 從方接收到主方的返回命令 ACK(0x06h)。

從方把更改的 MW 的位址和內容傳送給了主方，主方再更改 MW 的資料，使得從方和主方對應節點位址內容保持一致。

31.5 實作範例

以下將示範如何將兩台人機使用 Memory Map 通訊協定連接。

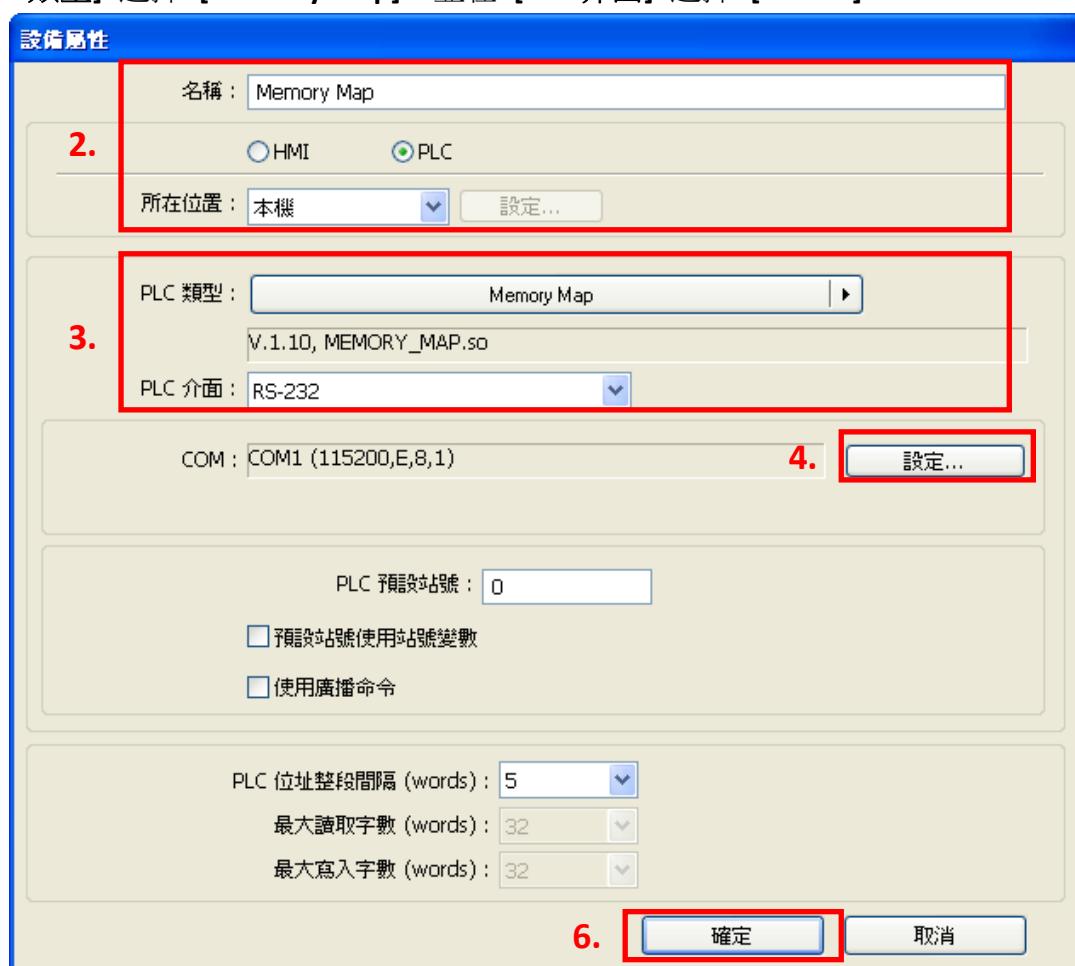


- 若是兩台的型號不同，請分別建立不同的工程檔，或者是在完成第一台人機的設定後，直接更改 [編輯] » [系統參數設定] » [HMI 屬性] 為第二台人機的型號，重新編譯工程檔再下載至第二台人機。

31.5.1 新增 Memory Map 設備

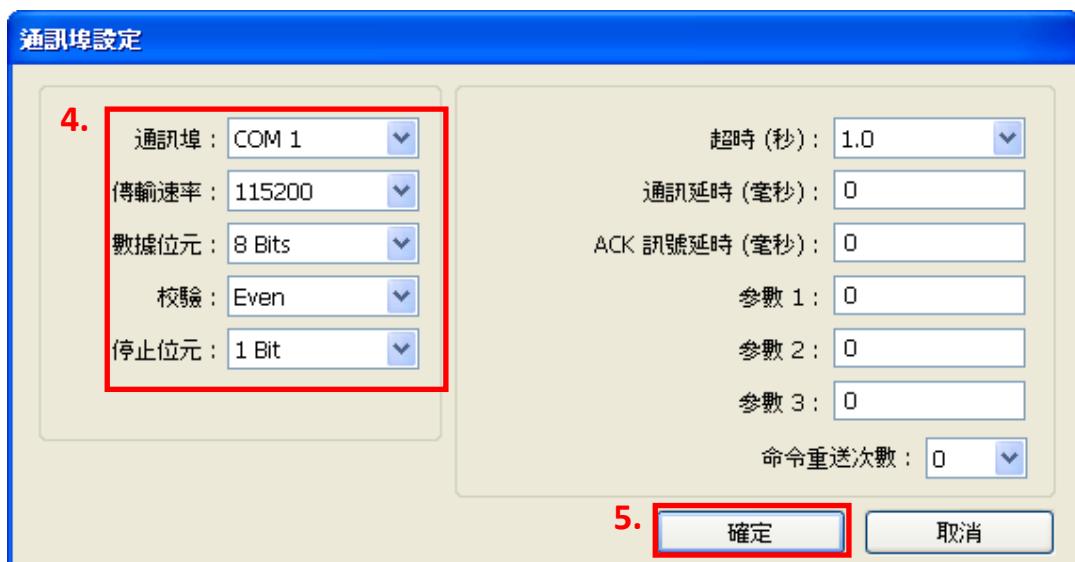
開啟 EasyBuilder，選擇 [開新檔案]，設定人機型號後，照以下步驟：

- 點選功能表列 [編輯]，點擊 [系統參數設定]，接著選擇 [設備清單] 頁籤，點選 [新增...] 加入新的設備。
- [名稱] 填入 “Memory Map”，並點選 [PLC]，設定 [所在位置] 為 [本機]。
- [PLC 類型] 選擇 [Memory Map]，並在 [PLC 介面] 選擇 [RS-232]。



4. 點擊 **[設定]**，設定如下：

- A. 通訊埠：COM 1
- B. 傳輸速率：115200
- C. 數據位元：8 Bits
- D. 校驗：Even
- E. 停止位元：1 Bit



5. 完成通訊埠設定後按 **[確定]**。

6. 點擊 **[確定]**。



- MT500 系列有分 MemoryMap_Master 和 MemoryMap_Slave 請參考相關手冊。
eMT 3000 系列和 MT8000 系列選擇 Memory Map 即可。
- **[數據位元]** 必須為 8 Bits。
- 兩台人機的所有其他設置必須一致。

31.5.2 物件設定

接著在視窗 10 上增加 2 個物件，「位元狀態切換開關」和「多狀態切換開關」：

新增位元狀態切換開關，如下圖：

1. 讀取和寫入位址的 [PLC 名稱] 選擇 [Memory Map]。
2. 位址選擇 MB-0。
3. [開關類型] 選擇 [切換開關]。(可選擇適合的圖片或標籤以供辨識)



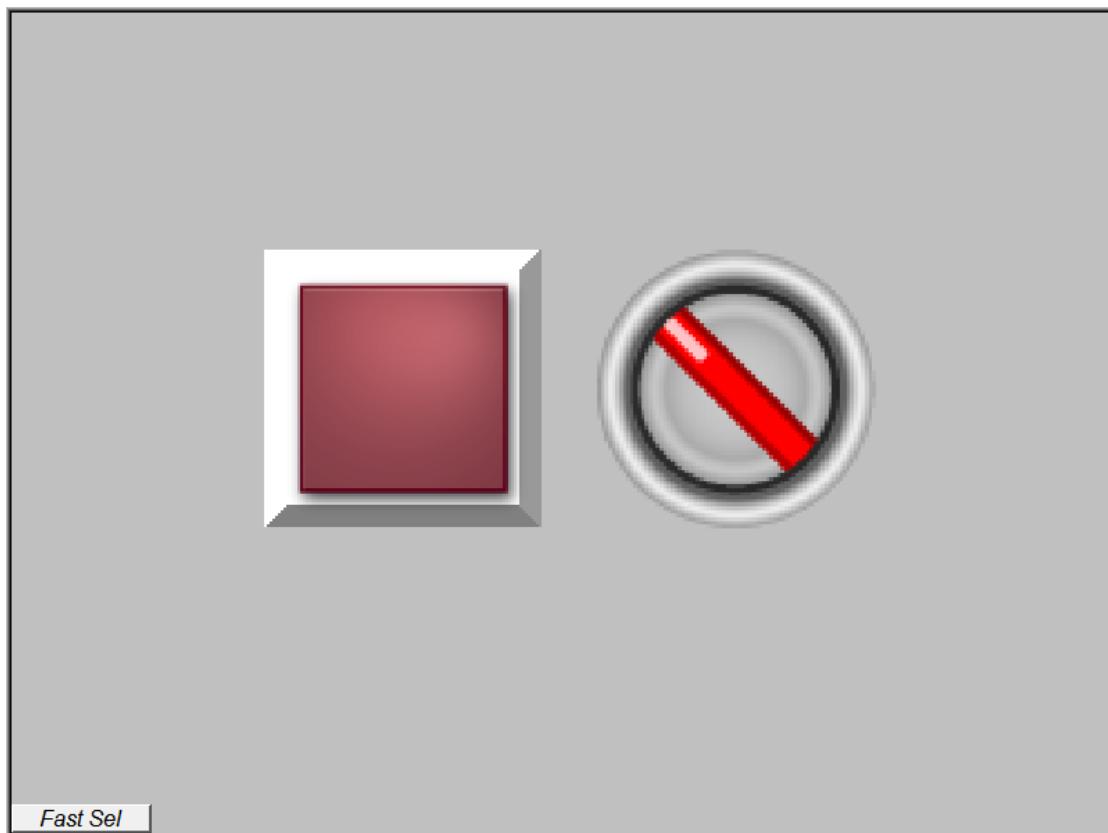
新增多狀態切換開關，如下圖：

1. 讀取和寫入位址的 [PLC 名稱] 選擇 [Memory Map]。
2. 位址選擇 MW-1。
3. 【迴圈】選擇 [啟用]。(可選擇適合的圖片或標籤以供辨識)



31.5.3 執行結果

將工程檔案編譯並下載至人機，再將工程檔案下載至第二台人機。
完成之畫面可參考下圖：

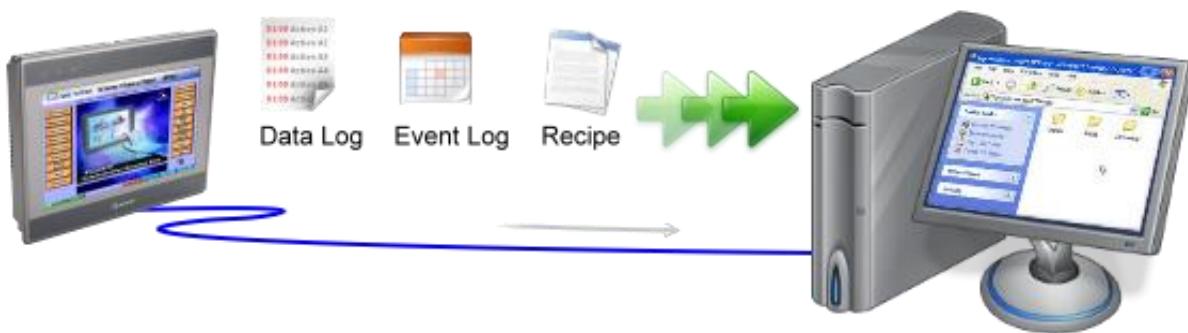


按一下任意一個按鈕，對應另一台觸控螢幕的該按鈕也將跟著動作，它們的狀態將始終保持一致。

一台人機和任一台控制器之間的通信其方式與上述類似，其根本原理是 2 台設備的相同暫存器的資料要保持一致。

第三十二章 FTP 伺服器之運用

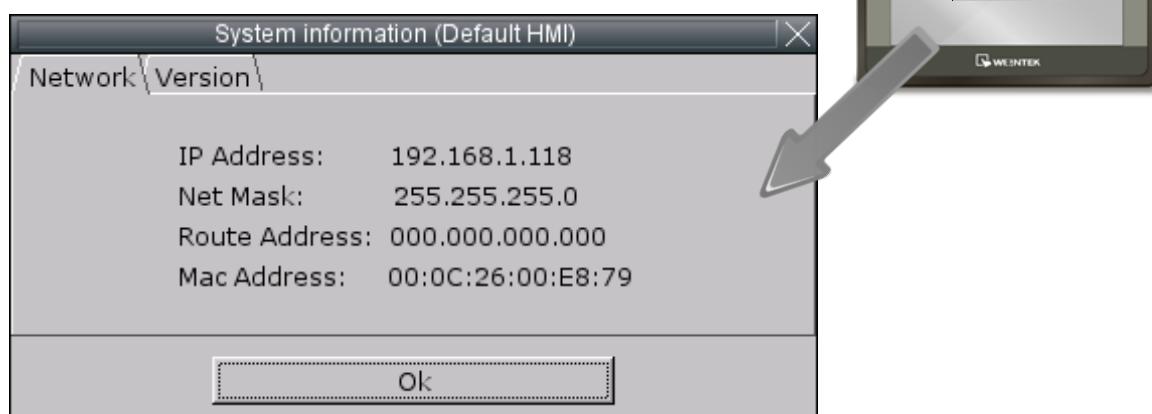
除了使用 SD 卡、USB 碟或 EasyPrinter 伺服器將歷史資料由 HMI 備份至 PC 之外，現在也能利用 FTP 伺服器達成這個目標。當工程檔案下載到 HMI 後，可透過 FTP 伺服器進行歷史資料備份、配方資料備份或更新配方資料的動作，但是無法刪除存在 FTP 伺服器內的檔案。



32.1 登入 FTP 伺服器

步驟 1

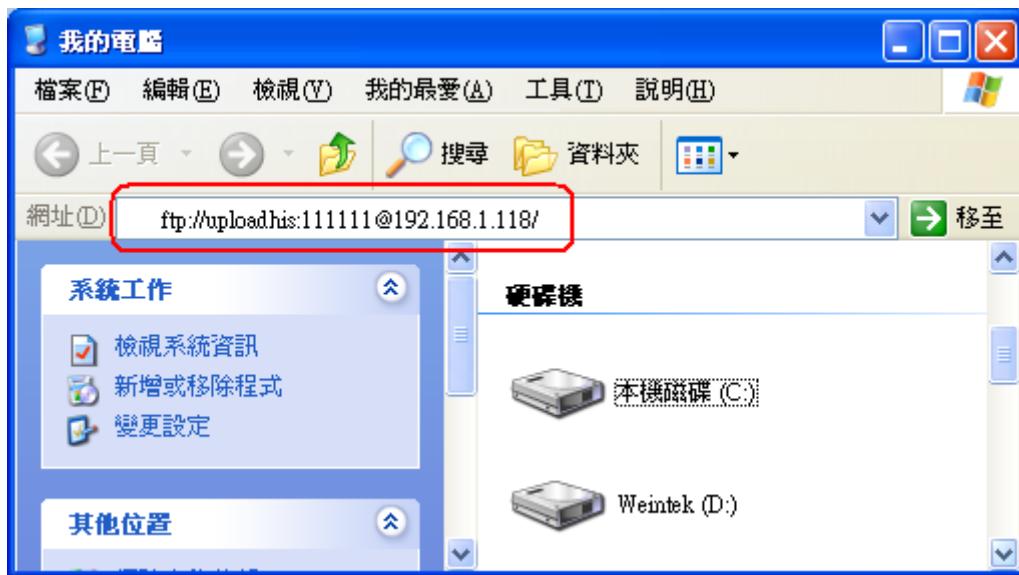
在登入 FTP 伺服器之前請先確認 HMI 的 IP 位址。



步驟 2

在 [我的電腦] 網址列輸入 HMI 的 IP 位址 <ftp://192.168.1.118> (範例) 然後登入帳號 uploadhis，密碼為 HMI 的 history upload password (若沒有更改過密碼，預設密碼為 111111)。

或是直接輸入 <ftp://uploadhis:111111@192.168.1.118/>



步驟 3

輸入 IP 位址後，網址列會顯示為 <ftp://192.168.1.118/>，並且可以看到 datalog、eventlog 及 recipe 的資料夾。



32.2 備份歷史資料及更新配方資料

◆ 備份資料取樣記錄

點選 **datalog** 資料夾後，可看到 **datalog** 在 EasyBuilder 設定的資料夾檔名，點選檔名後即可看到 **datalog** 的檔案，可使用複製及貼上的功能，將資料取樣的記錄保存在 PC 上。



◆ 備份警報及事件記錄

點選 **eventlog** 資料夾後，即可看到事件的記錄檔案。可使用複製及貼上的功能將事件記錄保存在 PC 上。



◆ 備份及更新配方資料

點選 **recipe** 資料夾後，即可看到配方資料的檔案。可使用複製及貼上的功能將配方資料保存在 PC 上。





- 因為配方資料每分鐘會自動儲存一次，若要更新 recipe.rcp 或 recipe_a.rcp，請務必於 1 分鐘內將 HMI 重啟，否則新配方資料將被原有舊配方覆蓋過去。
- 使用者也可使用系統暫存器 LB-9047 (重新啟動 HMI) & LB-9048 (重啟機制保護) 將 HMI 重啟。若是使用 LB-9047 及 LB-9048，需先將 LB-9048 設 ON 後，再將 LB-9047 設 ON，即可重啟 HMI。

第三十三章 EasyDiagnoser

33.1 簡介與設定方式

簡介

EasyDiagnoser 是用來偵測人機與 PLC 之間通信是否正常的工具。

設定方式

步驟一

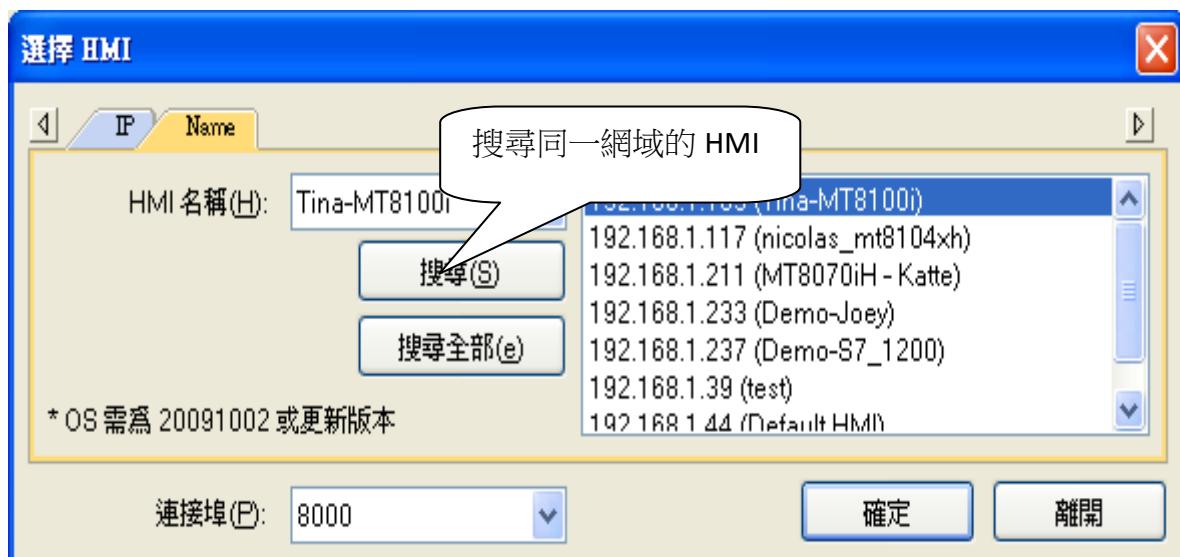
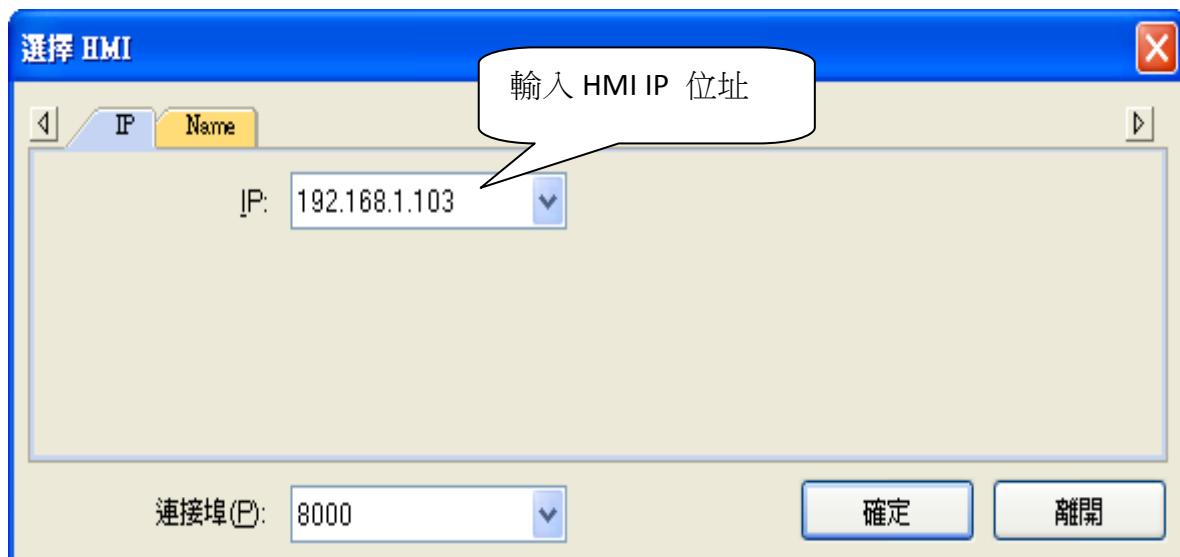
開啟 Project Manager 並且點選 EasyDiagnoser。



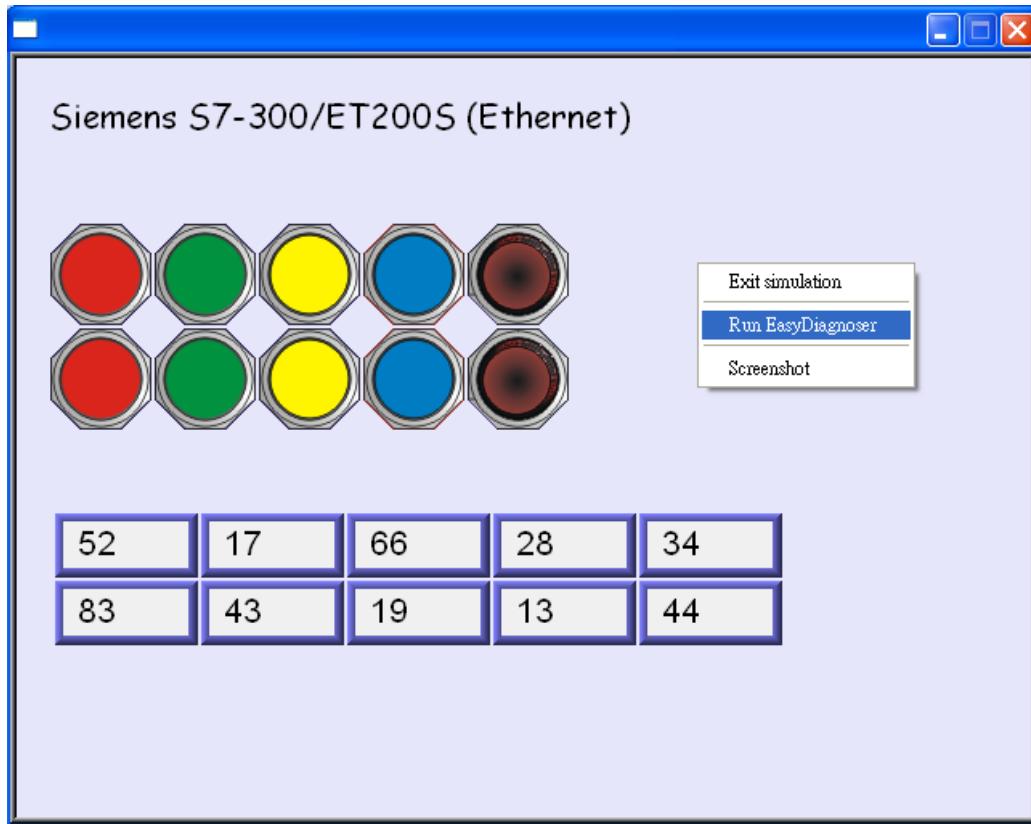
步驟二

設定欲進行通信之人機 IP 位址

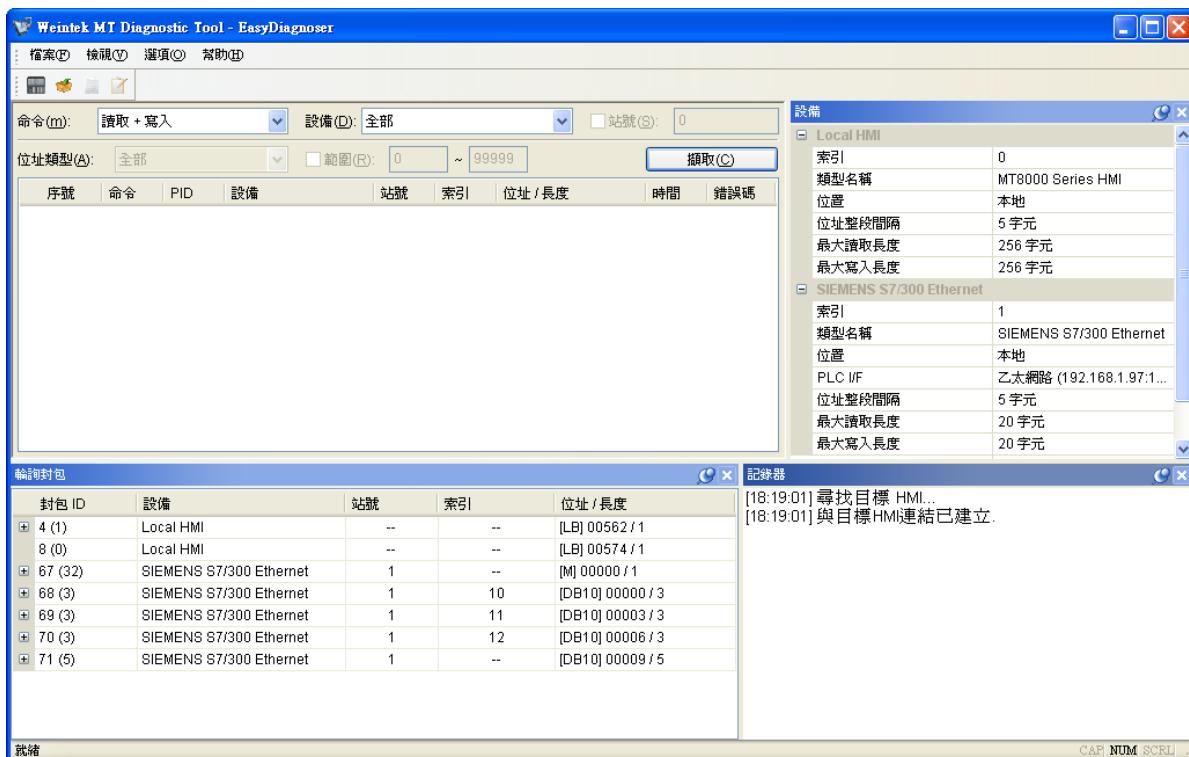
可選擇自行輸入 IP 位址或使用 **[Search all]** 功能，並輸入 **[Project Port]**。



另外在 EasyBuilder 執行 On Line Simulation 時，按下右鍵可選擇 [Run EasyDiagnoser] 進入 EasyDiagnoser。



完成以上設定之後，按下 [OK]，EasyDiagnoser 操作畫面如下圖：



33.2 EasyDiagnoser 設定

項目	敘述
檔案	<p>另存新檔 可將擷取下來的通訊資料，儲存成 xls，並可於 Excel 開啟。</p>  <p>離開 離開目前檔案。</p>
檢視	<p>設備列表① Ctrl+Alt+D 封包列表② Ctrl+Alt+P 訊息視窗③ Ctrl+Alt+L 輸出視窗④ Ctrl+Alt+O</p> <p>點選【設備列表】可顯示設備列表視窗 點選【封包列表】可顯示封包列表視窗 點選【訊息視窗】可顯示訊息視窗視窗 點選【輸出視窗】可顯示輸出視窗視窗</p>
選項	<p>工具列 顯示【設備列表】、【封包列表】、【訊息視窗】、【輸出視窗】之工具列</p>   <p>狀態列 在 EasyDiagnoser 視窗最底下，顯示 CAP，NUM 或 SCRL 的資訊</p>  <p>更新封包列表 顯示目前人機頁面的封包</p> <p>顯示物件 ID 顯示人機上物件的 ID</p>



● 通訊記錄區

在通訊記錄區，使用者可以觀察人機和 PLC 之間的通訊。

Weintek MT Diagnostic Tool - EasyDiagnoser									
檔案(F) 檢視(V) 選項(O) 幫助(H)     									
命令(m):		讀取 + 寫入		設備(D):		全部		<input type="checkbox"/> 站號(S): 0	
位址類型(A):		全部		<input type="checkbox"/> 範圍(R): 0 ~ 99999		擷取(C)			
序號	命令	PID	設備	站號	索引	位址 / 長度	時間	錯誤碼	
118	R	8	Local HMI	--	--	[LB] 00574 / 1	20	0	
117	R	68	SIEMENS S7/300 Et...	1	10	[DB10] 00000 / 3	20	0	
116	R	69	SIEMENS S7/300 Et...	1	11	[DB10] 00003 / 3	30	0	
115	R	4	Local HMI	--	--	[LB] 00562 / 1	20	0	
114	R	8	Local HMI	--	--	[LB] 00574 / 1	20	0	
113	R	70	SIEMENS S7/300 Et...	1	12	[DB10] 00006 / 3	20	0	
112	R	71	SIEMENS S7/300 Et...	1	255	[DB10] 00009 / 5	30	0	
111	R	4	Local HMI	--	--	[LB] 00562 / 1	20	0	
110	R	8	Local HMI	--	--	[LB] 00574 / 1	20	0	
109	R	67	SIEMENS S7/300 Et...	1	255	[M] 00000 / 1	20	0	
108	R	68	SIEMENS S7/300 Et...	1	10	[DB10] 00000 / 3	40	0	

項目	敘述
命令	<p>a. 讀取 + 寫入 顯示讀和寫的命令在通訊記錄區</p>
	<p>b. 讀取 只顯示讀的命令在通訊記錄區</p>
	<p>c. 寫入 只顯示寫的命令在通訊記錄區</p>
設備	<p>a. 全部 顯示本地 HMI 和 PLC 的資訊 如果設定 [命令 : 讀取 + 寫入], 在通訊記錄區會顯示本地 HMI 和 PLC 的讀和寫的資訊 如果 [命令 : 讀取], 在通訊記錄區會顯示本地 HMI 和 PLC 的讀的資訊 如果 [命令 : 寫入], 在通訊記錄區會顯示本地 HMI 和 PLC 的寫的資訊</p> <p>b. Local HMI 顯示本地 HMI 的資訊 如果設 [命令 : 讀取 + 寫入], 在通訊記錄區會顯示本地 HMI 的讀和寫的資訊 如果 [命令 : 讀取], 在通訊記錄區會顯示本地 HMI 的讀的資訊 如果 [命令 : 寫入], 在通訊記錄區會顯示本地 HMI 的寫的資訊</p> <p>c. PLC 顯示 PLC 的資訊 如果設 [命令 : 讀取 + 寫入], 在通訊記錄區會顯示 PLC 的讀和寫的資訊 如果 [命令 : 讀取], 在通訊記錄區會顯示 PLC 的讀的資訊 如果 [命令 : 寫入], 在通訊記錄區會顯示 PLC 的寫的資訊</p>
站號	選擇想顯示的 PLC 之站號 (當 [設備] 選擇 All 時無法使用使功能)
位址類型	使用者可以選擇全部或是其中的設備位址類型顯示在螢幕上 (當 [設備] 選擇 All 時無法使用使功能)
範圍	設定要擷取的位址範圍 (當 [設備] 選擇 All 時無法使用使功能)
擷取	點選 [擷取] 鈕開始或停止擷取通訊信息
錯誤碼	請參考本 33.3 節

- 輪詢封包

輪詢封包				
封包 ID	設備	站號	索引	位址 / 長度
+ 4 (1)	Local HMI	--	--	[LB] 00562 / 1
8 (0)	Local HMI	--	--	[LB] 00574 / 1
+ 67 (32)	SIEMENS S7/300 Ethernet	1	--	[M] 00000 / 1
+ 68 (3)	SIEMENS S7/300 Ethernet	1	10	[DB10] 00000 / 3
+ 69 (3)	SIEMENS S7/300 Ethernet	1	11	[DB10] 00003 / 3
+ 70 (3)	SIEMENS S7/300 Ethernet	1	12	[DB10] 00006 / 3
+ 71 (5)	SIEMENS S7/300 Ethernet	1	--	[DB10] 00009 / 5

項目	敘述
封包 ID	封包的 ID 編號 可由通訊記錄區看出那一個封包 ID 的物件有問題
設備	顯示人機和 PLC 型號
站號	顯示 PLC 站號.
索引	顯示物件所使用的 Index register 編號
位址 / 長度	顯示設備類型位址及封包內的 word 長度

輪詢封包				
物件	視窗	ID	位址	
+ 4 (1)	--	--	[LB] 00562 / 1	
66 (32)	--	--	[I] 00000 / 1	
▶ 位元狀態切換開關	10	30	[I] 00000	
位元狀態切換開關	10	30	[I] 00000	
位元狀態切換開關	10	29	[I] 00000	
位元狀態切換開關	10	29	[I] 00000	
位元狀態切換開關	10	28	[I] 00000	
位元狀態切換開關	10	28	[I] 00000	
位元狀態切換開關	10	27	[I] 00000	

項目	敘述
物件	封包 ID 內的物件
視窗	物件在程式中的所在視窗
ID	物件的 ID 號碼
位址	物件位址

注意：

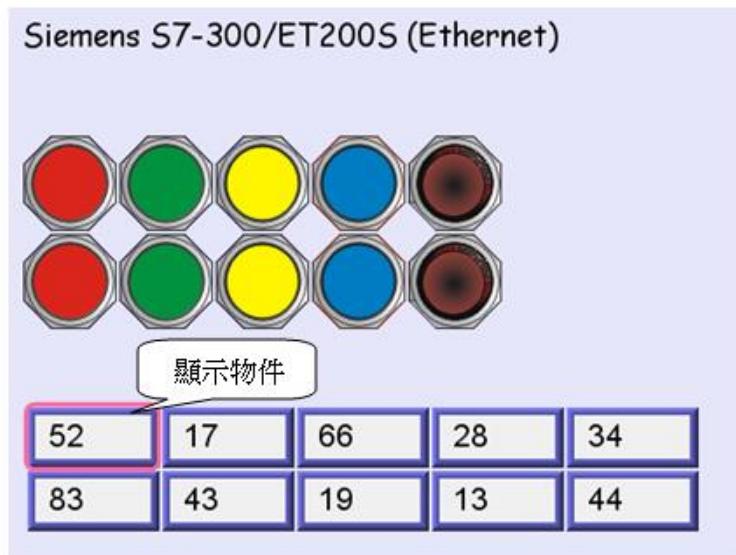
- a. 點選封包 ID 後，第 3 欄會顯示設備的站號：

封包 ID	設備	站號	索引	位址 / 長度
+ 131 (32)	SIEMENS S7/200	3	--	[MW_Bit] 00000
+ 130 (32)	SIEMENS S7/200	3	--	[M] 00000 / 1
+ 129 (32)	SIEMENS S7/200	3	--	[Q] 00000 / 1
+ 128 (32)	SIEMENS S7/200	3	--	[I] 00000 / 1
+ 127 (16)	SIEMENS S7/200	3	--	[WW] 00000 / 16
--- Numeric Input		10	0	[WW] 00000
--- Numeric Input		10	65	[WW] 00001

- b. 雙擊封包 ID 之後點選物件，可顯示物件所在的位置。

例如，選擇數值輸入 同時【視窗】顯示 10，表示此物件在程式中的第 10 視窗中，同時此物件在人機上會被粉紅色的框框標示出來，如下圖：

物件	視窗	ID	位址
+ 4 (1)	--	--	[LB] 00562 / 1
--- 66 (32)	--	--	0 00000 / 1
▶ 位元狀態切換開關	10	30	0 00000
位元狀態切換開關	10	30	0 00000
位元狀態切換開關	10	29	0 00000
位元狀態切換開關	10	29	0 00000
位元狀態切換開關	10	28	0 00000
位元狀態切換開關	10	28	0 00000
位元狀態切換開關	10	27	0 00000



● 設備

設備視窗顯示 HMI 及 PLC 的相關訊息

設備	
索引	0
類型名稱	MT8000 Series HMI
位置	本地
位址整段間隔	5字元
最大讀取長度	256字元
最大寫入長度	256字元
索引	1
類型名稱	SIEMENS S7/300 Ethernet
位置	本地
PLC I/F	乙太網路 (192.168.1.97:102)
位址整段間隔	5字元
最大讀取長度	20字元
最大寫入長度	20字元

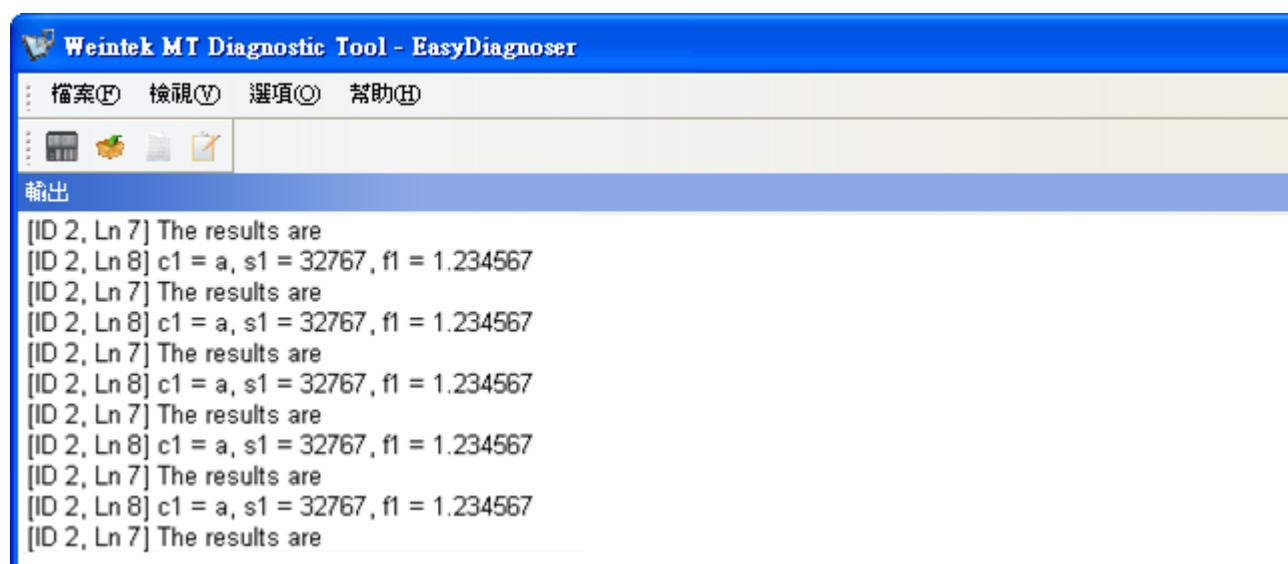
● 輸出 (Macro debug)

搭配使用 Macro 所提供的 Trace 函數,能夠偵測 Macro 執行的狀態,請參考使用手冊第 18 章有詳細的說明。

以下圖為例， [ID 2,Ln 7] 及 [ID 2,Ln 8]

ID 2 表示 Macro 的名稱

Ln 7 及 Ln 8 表示顯示在 Macro 中的第 7 行及第 8 行資料



Weintek MT Diagnostic Tool - EasyDiagnoser

檔案(F) 檢視(V) 選項(O) 幫助(H)

輸出

```
[ID 2, Ln 7] The results are
[ID 2, Ln 8] c1 = a, s1 = 32767, f1 = 1.234567
[ID 2, Ln 7] The results are
[ID 2, Ln 8] c1 = a, s1 = 32767, f1 = 1.234567
[ID 2, Ln 7] The results are
[ID 2, Ln 8] c1 = a, s1 = 32767, f1 = 1.234567
[ID 2, Ln 7] The results are
[ID 2, Ln 8] c1 = a, s1 = 32767, f1 = 1.234567
[ID 2, Ln 7] The results are
[ID 2, Ln 8] c1 = a, s1 = 32767, f1 = 1.234567
[ID 2, Ln 7] The results are
[ID 2, Ln 8] c1 = a, s1 = 32767, f1 = 1.234567
```

33.3 錯誤代碼

在通訊記錄區可從錯誤代碼找出錯誤原因。請參考下列錯誤代碼。

0 : Normal

1 : Time out

2 : Fail Error

12 : Ignore

當錯誤發生時，錯誤的訊息會標示成紅色，如下圖：

錯誤代碼是 1 是由於 PLC 與人機斷了通訊，

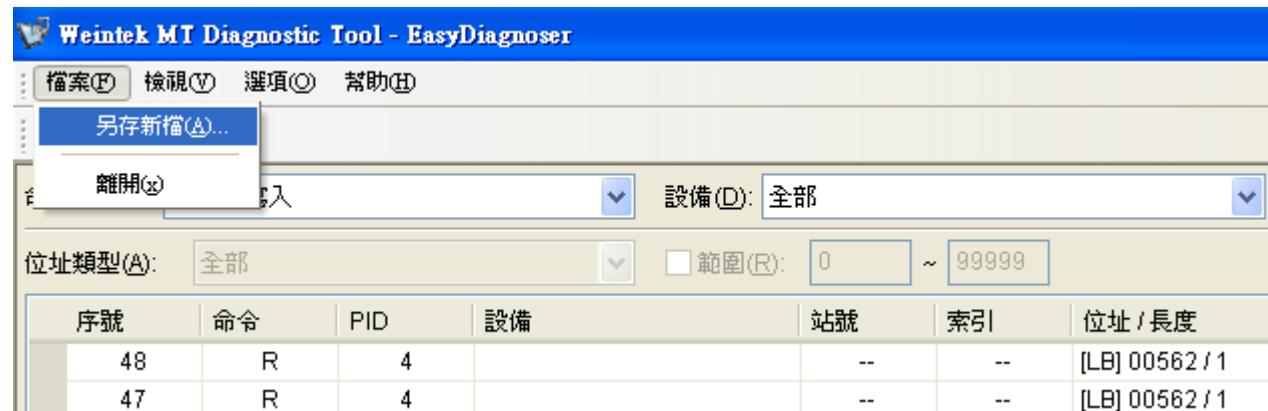
如果錯誤代碼是 12，表示出現 “**PLC No Response**” 訊息視窗。

The screenshot shows the 'Weintek MT Diagnostic Tool - EasyDiagnoser' application window. The menu bar includes '檔案(F)', '檢視(V)', '選項(O)', and '幫助(H)'. Below the menu is a toolbar with icons for file operations. The main area contains a table with the following columns: 序號 (Row ID), 命令 (Command), PID, 設備 (Device), 站號 (Station ID), 索引 (Index), 位址/長度 (Address/Length), 時間 (Time), and 錯誤碼 (Error Code). The table lists 14 rows of data, with rows 199, 198, 197, 196, 195, 194, 193, 192, 191, 190, and 189 highlighted in red, indicating errors. The '命令' column shows mostly 'R' (Read) commands, except for row 191 which is 'M' (Memory). The '站號' column is mostly '1', except for rows 197, 196, 195, 194, 193, 192, and 191 which have values 255, 255, 10, 11, 12, 11, and 255 respectively. The '錯誤碼' column shows values 12, 12, 12, 12, 12, 12, 12, 1, 0, 0, 0, 0, 0, and 0.

序號	命令	PID	設備	站號	索引	位址/長度	時間	錯誤碼
199	R	68		1	11	[DB10] 00003/3	310	12
198	R	69		1	12	[DB10] 00006/3	310	12
197	R	70		1	255	[DB10] 00009/5	310	12
196	R	66		1	255	[M] 00000/1	300	12
195	R	67		1	10	[DB10] 00000/3	310	12
194	R	68		1	11	[DB10] 00003/3	2220	12
193	R	69		1	12	[DB10] 00006/3	2110	12
192	R	70		1	255	[DB10] 00009/5	2030	1
191	R	66		1	255	[M] 00000/1	30	0
190	R	67		1	10	[DB10] 00000/3	40	0
189	R	68		1	11	[DB10] 00003/3	30	0

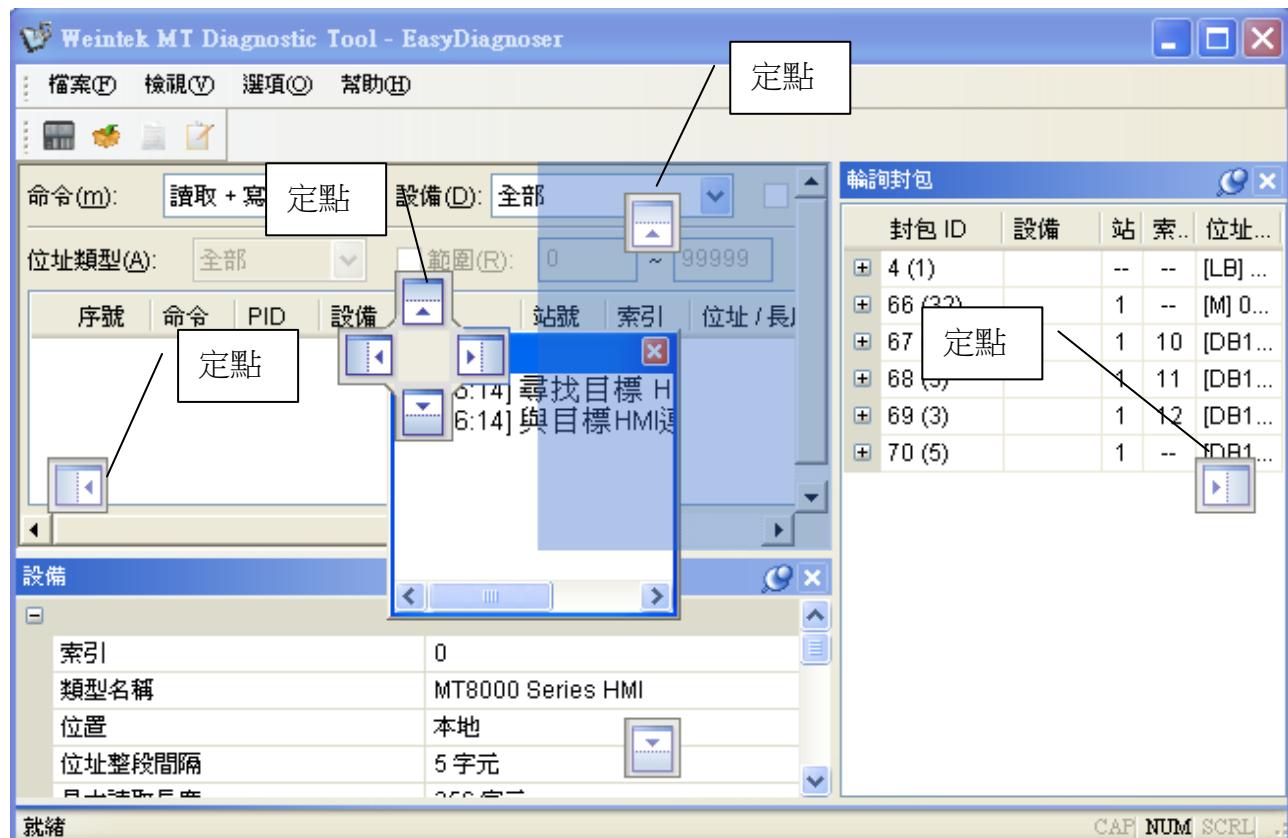
33.4 另存新檔

可將所擷取的檔案另存成 *.xls 格式,並且可以使用 Excel 開啟檔案瀏覽。



33.5 視窗調整

使用者可以使用拖曳功能及顯示在編輯畫面上的多個定點圖示來放置視窗到您喜愛的位置。



注意:

EasyDiagnoser 不支援使用 Siemens S7-1200 (Ethernet)、Rockwell EtherNet / IP (CompactLogix) – Free Tag Names 和 Rockwell EtherNet / IP (ControlLogix) – Free Tag Names 等使用 tag 的 PLC。

第三十四章 Rockwell EtherNet/IP Free Tag Names

使用 Rockwell EtherNet/IP-Tag (CompactLogix/ControlLogix)的驅動時可以將使用者於 RSLogix5000 編輯的 tag 匯出成 csv 檔案，再開啟 EasyBuilder 設定驅動後直接匯入 csv 檔案，以取得 tag 的資訊，但是 User-Defined, Predefined 和 Module-Defined 的結構並不會被匯出。

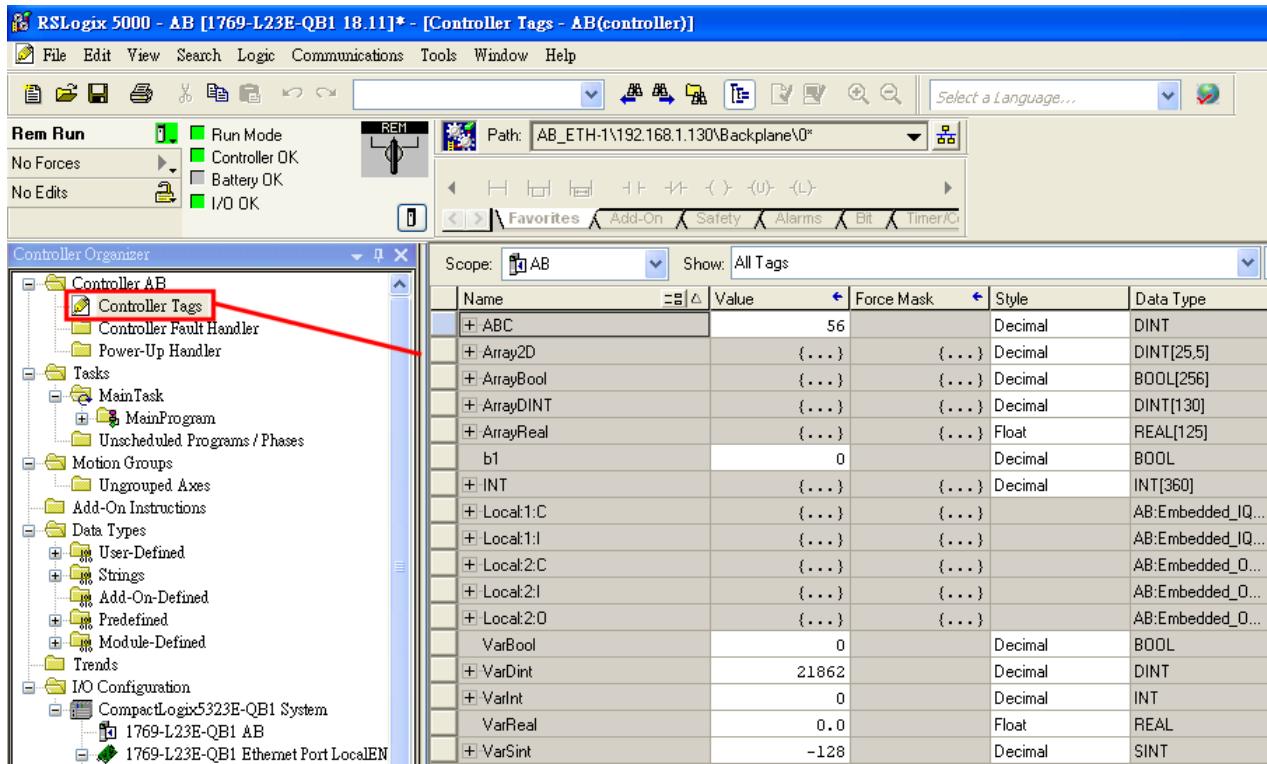
	A	B	C	D	E	F	
	TYPE	SCOPE	NAME	DESCRIPTION	DATATYPE	SPECIFIER	ATTRIBUTES
7	TAG		Local:1:C		AB:Embedded_IQ16F:C:0		
9	TAG		Local:1:I		AB:Embedded_IQ16F:I:0		
10	TAG		Local:2:C		AB:Embedded_OB16:C:0		
11	TAG		Local:2:I		AB:Embedded_OB16:I:0		
12	TAG		Local:2:O		AB:Embedded_OB16:O:0		
13	TAG		Array2D		DINT[25,5]		(RADIX := Decimal, Cons)
14	TAG		ArrayBool		BOOL[256]		(RADIX := Decimal, Cons)
15	TAG		ArrayDINT		DINT[130]		(RADIX := Decimal, Cons)
16	TAG		ArrayReal		REAL[125]		(RADIX := Float, Constant)
17	TAG		B001		INT[15]		(RADIX := Decimal, PLC)
18	TAG		b003		INT[255]		(RADIX := Decimal, PLC)
19	TAG		b001		B001		(RADIX := Decimal, Cons)

此時可利用 EasyBuilder 的 Structure Editor 工具來輸入和編輯 User-Defined, Predefined 和 Module-Defined 。

34.1 汇入使用者自訂 AB Tag CSV 檔至 EasyBuilder

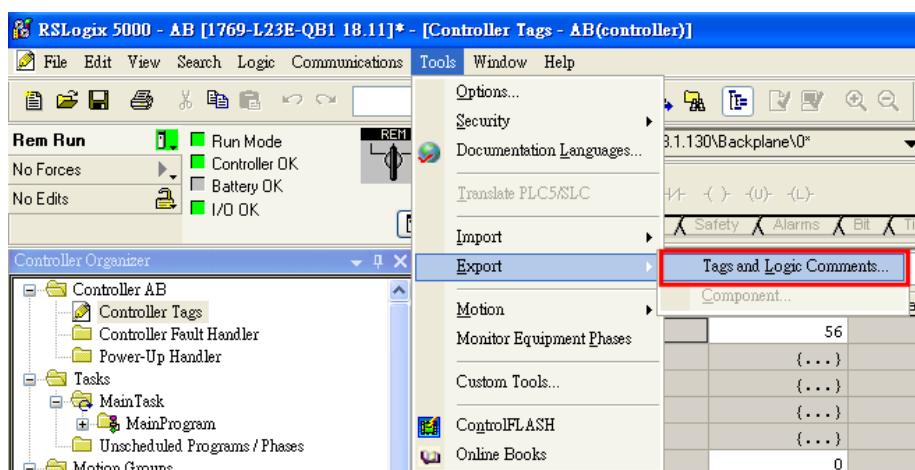
步驟 1

在 RSLogix5000 建立 Tags。



步驟 2

匯出 Tags 成 csv 檔。



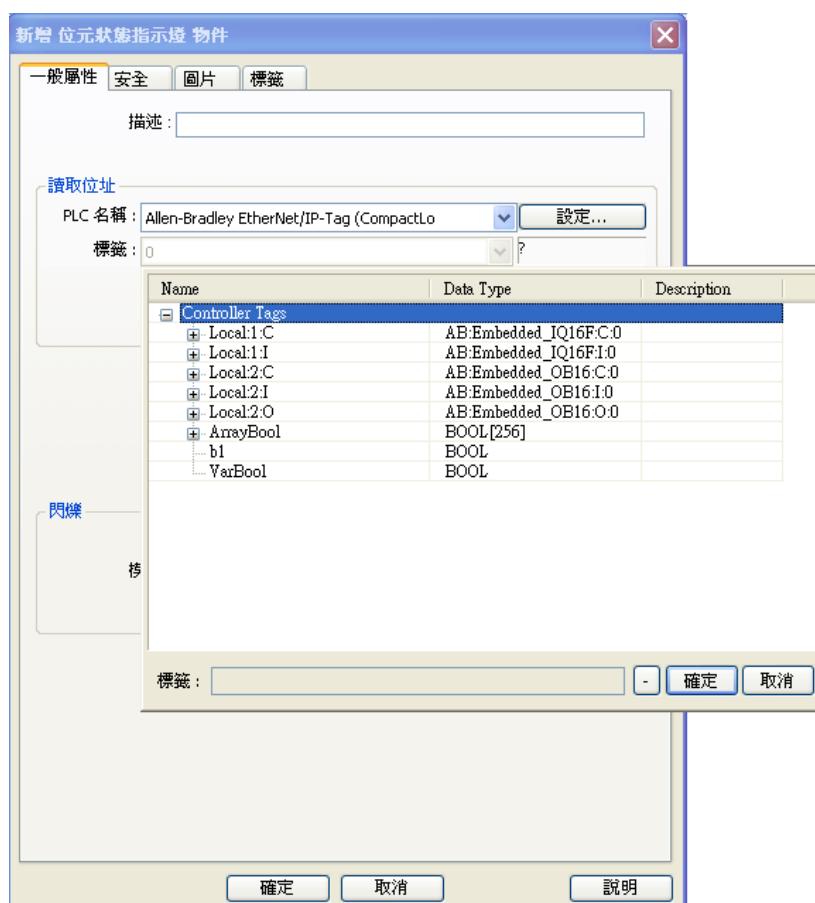
步驟 3

在 EasyBuilder 建立 Rockwell EtherNet/IP-Tag (CompactLogix/ControlLogix) 驅動。輸入 PLC 的 IP 位址並點選【匯入標籤】。



步驟 4

在物件視窗中選擇 PLC，並選擇 controller tag 即可。

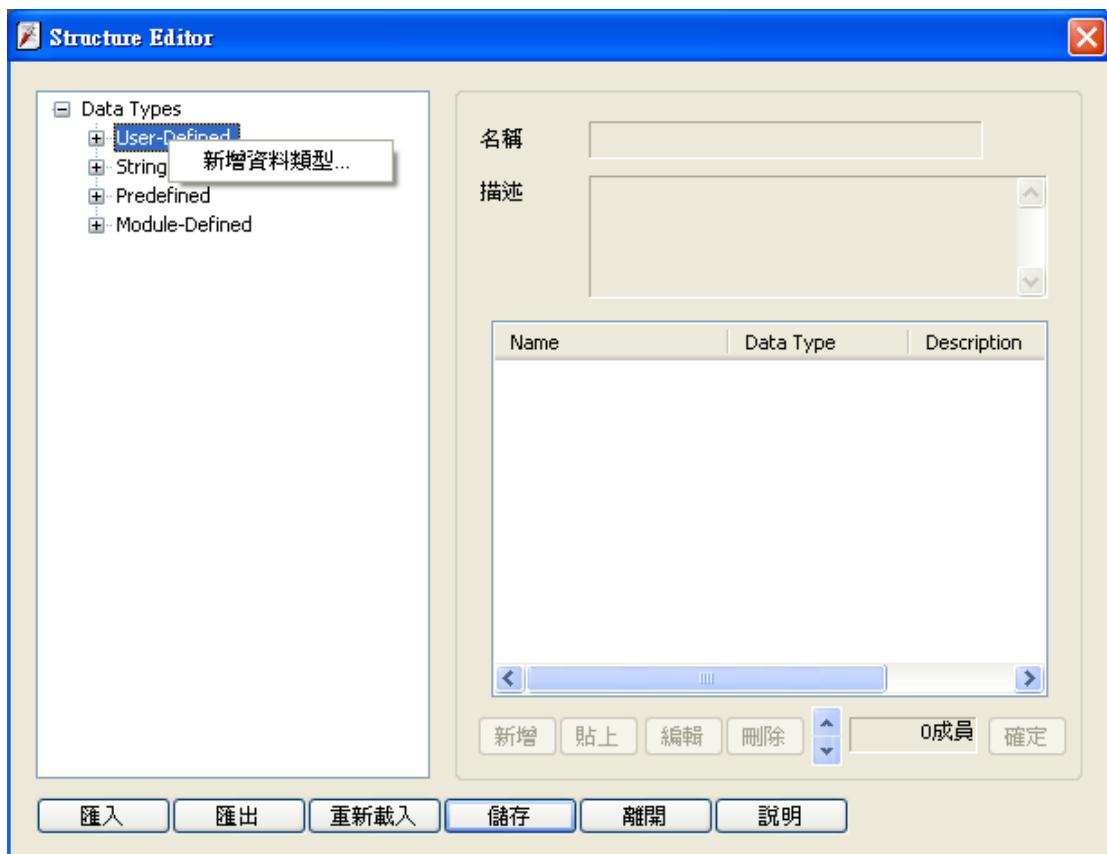


34.2 新增資料型態

Structure Editor 工具放置於 EasyBuilder 安裝後的資料夾內。點擊 Structure Editor.exe 後會出現如下圖的編輯視窗。

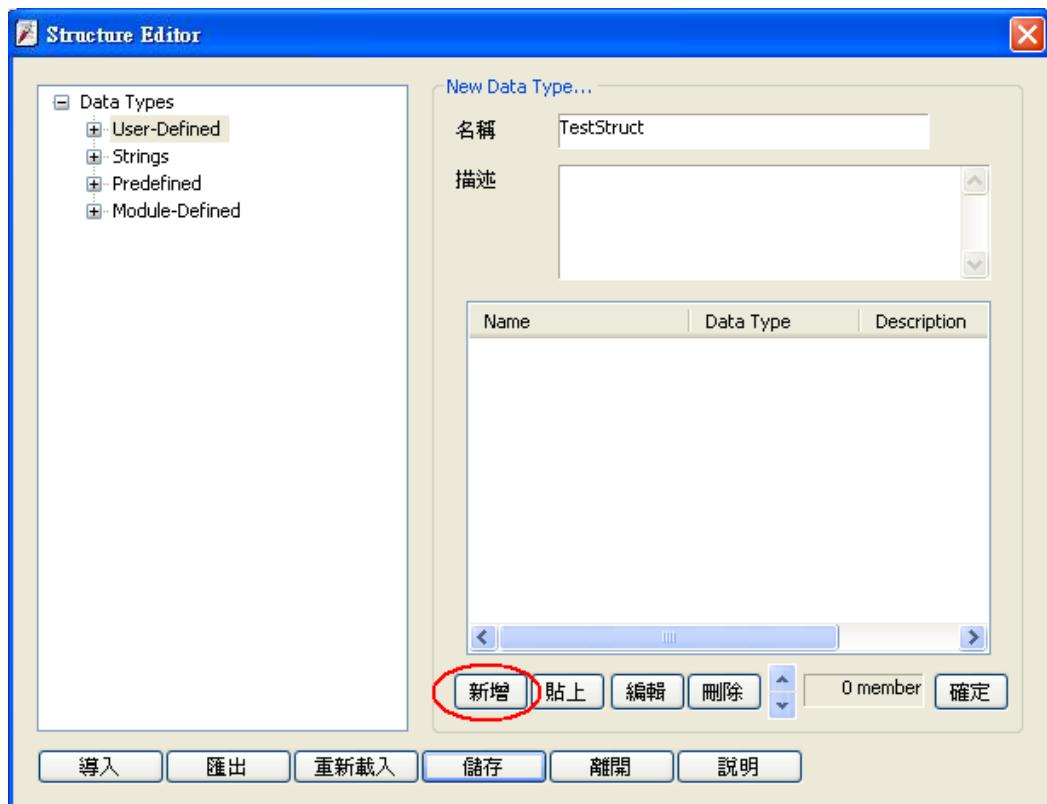
步驟 1

在所屬的類型上點擊滑鼠右鍵（通常為 User-Defined），點擊選單的【新增資料類型】即可開始編輯。



步驟 2

輸入類型名稱後，【描述】可略過。新增資料成員，點擊【新增】按鈕。



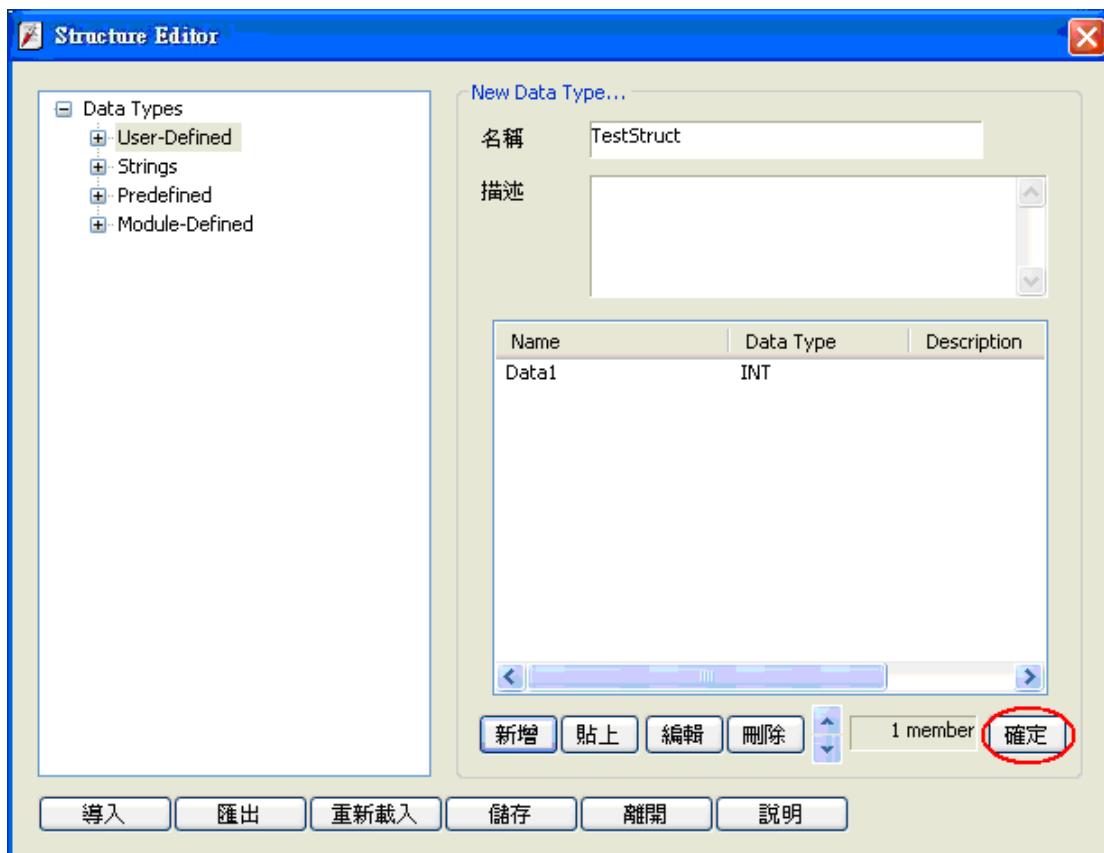
步驟 3

輸入資料的名稱與型態後按下【確定】離開。



步驟 4

增加完所有的資料成員後按下**確定**鍵，此時左邊的類別表即會出現剛建立的型別。

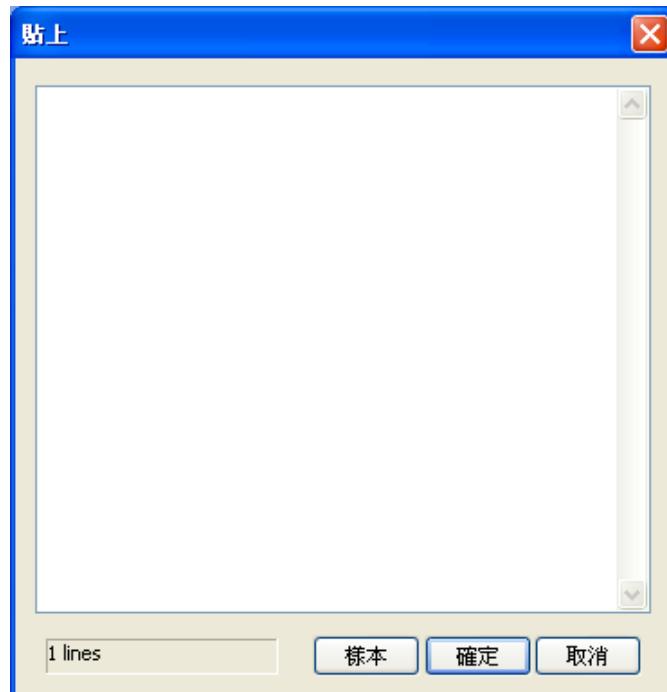


修改資料型態名稱與描述後需按下**確定**才進行更動。

34.3 貼上功能

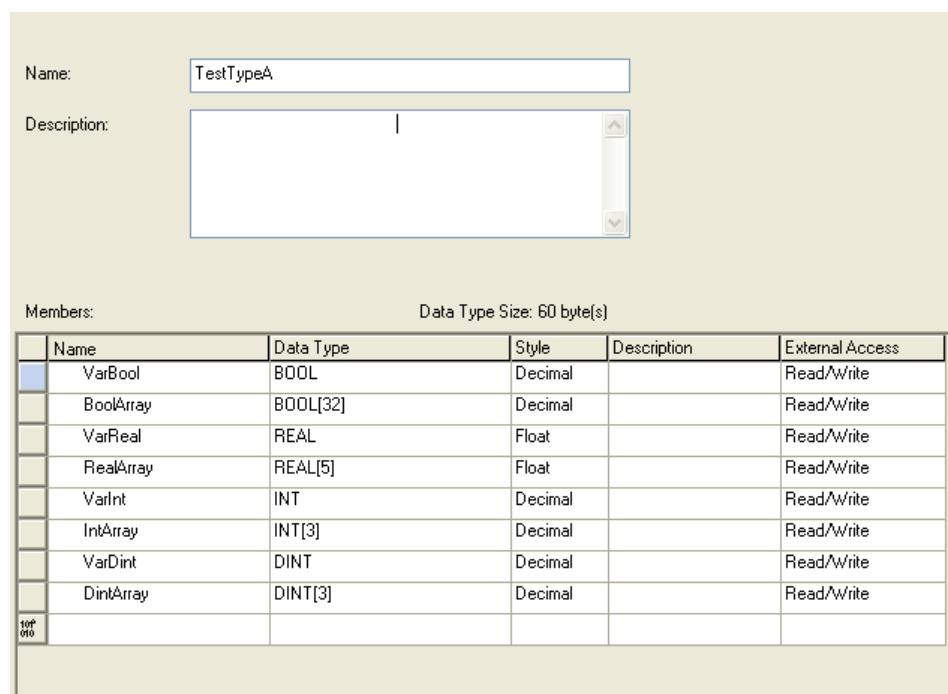
步驟 1

在新增資料成員時，使用此功能可一次新增多筆資料；方式為在主畫面按下【貼上】按鈕：



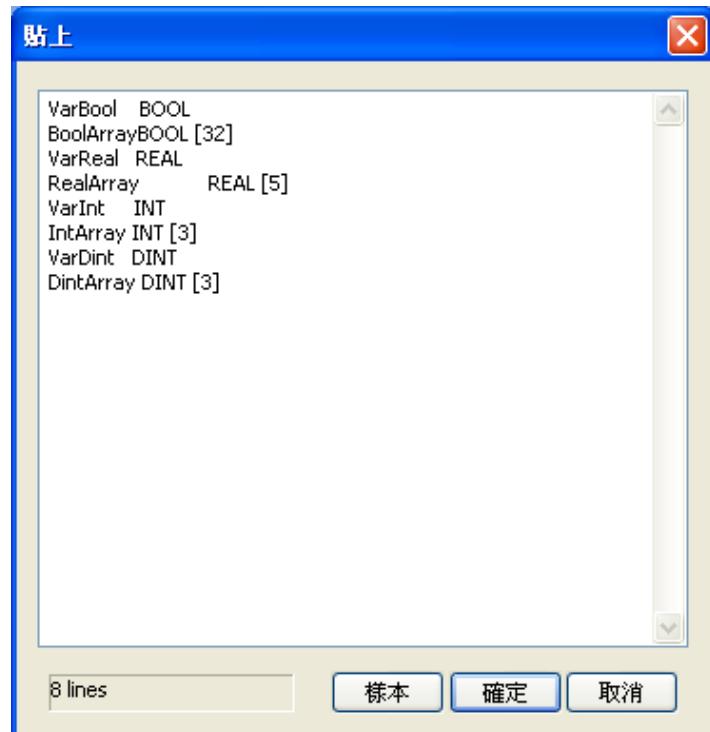
步驟 2

編輯方式每行先輸入資料名稱後接一空格或 tab，然後輸入資料型態，可按【樣本】按鈕參考；建議從 RSLogix 5000 中直接複製貼上以避免輸入錯誤。



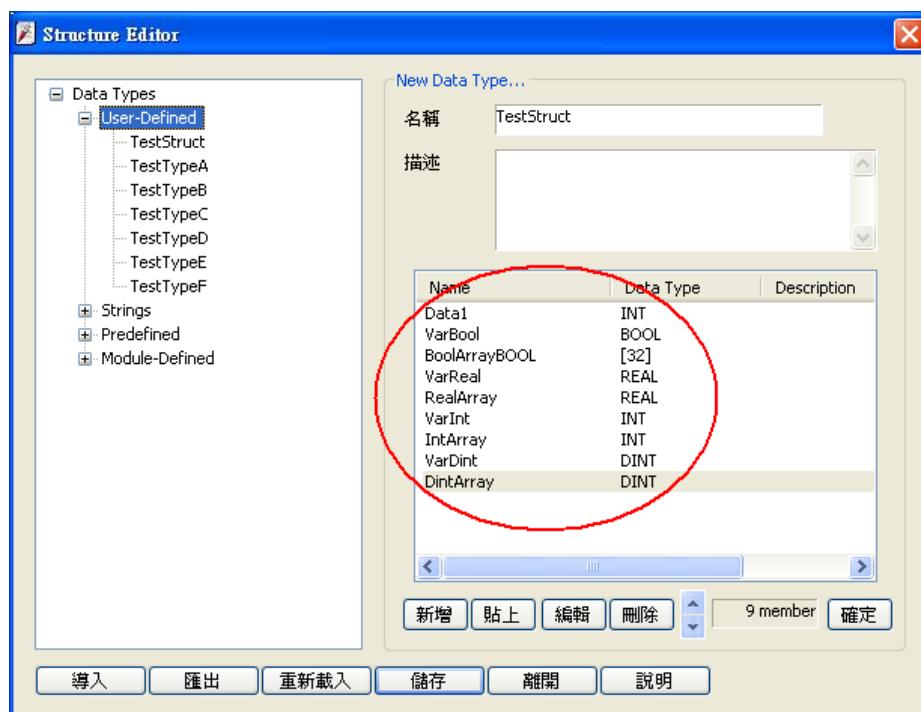
步驟 3

上圖為一在 RSLogix 中自訂的型態，使用滑鼠將 Name 與 Data Type 選取起來，可從第一項選取壓住滑鼠移至底部直到捲軸自動滑到底部後放開，如此便可全部選取；接下來按下 **ctrl + v** 進行複製，然後貼入編輯畫面上。



步驟 4

此時按下**[確定]**完成操作回到主畫面便可看到已成功新增多筆資料。



34.4 其它功能

- 修改成員資料：

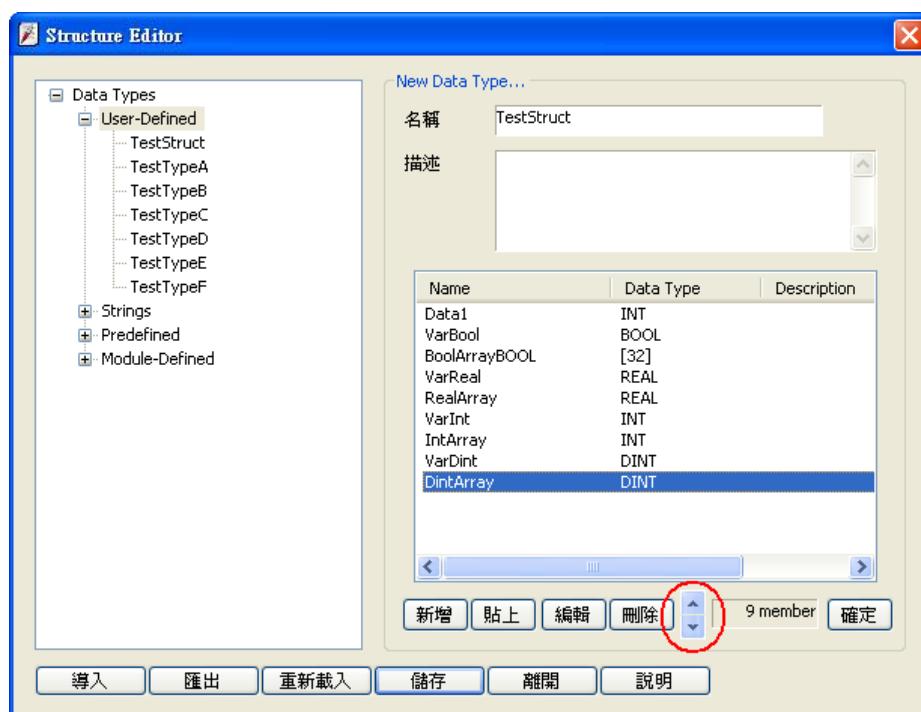
直接雙擊主畫面要修改的資料成員，或點擊該成員後按下**[編輯]**按鈕。

- 刪除資料成員：

選取要刪除的資料按下**[刪除]**按鈕；若要刪除所有資料成員則壓住鍵盤上的 Delete 鍵並點擊主畫面上的**[刪除]**按鈕。

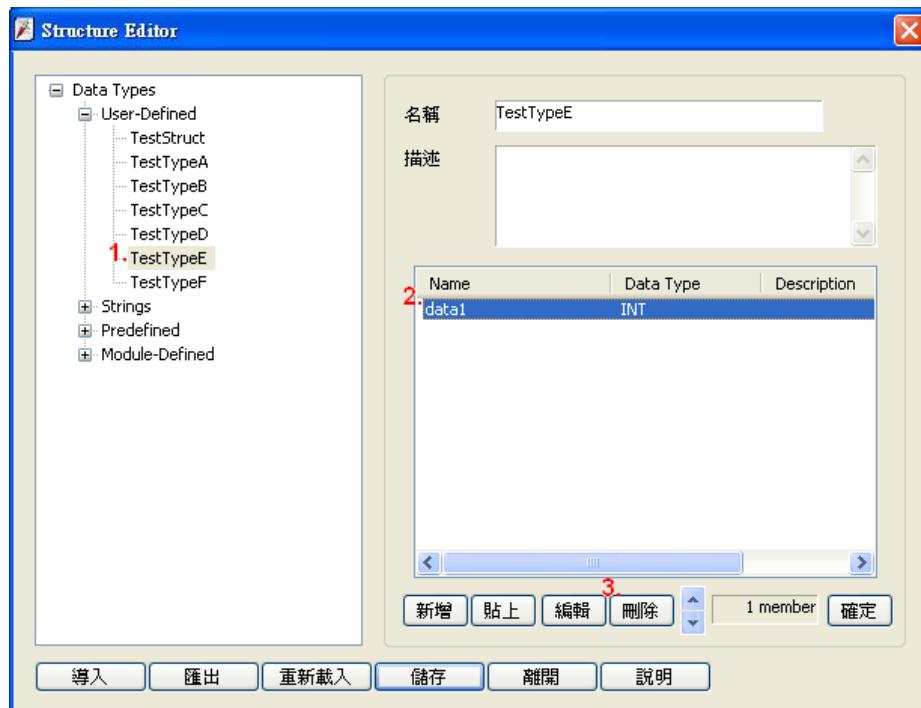
- 調整資料成員順序：

選取單筆資料成員後，可利用主畫面上的上下按鈕調整順序，以增加在 EasyBuilder 選取的方便性。



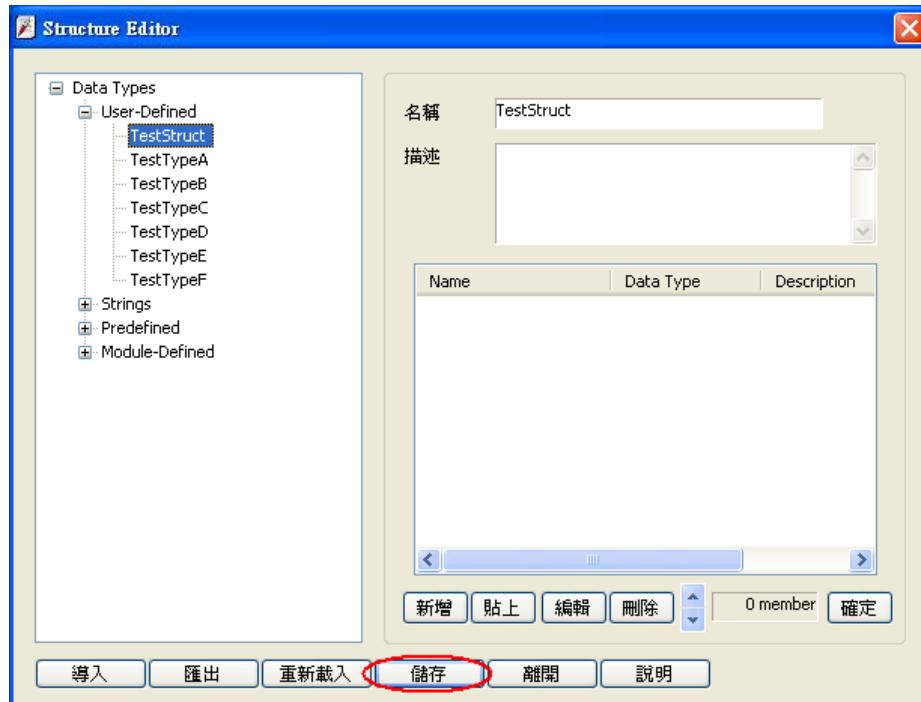
- 刪除資料型態：

在主畫面左邊的型態表選取要刪除的型態，再於右邊的項目中選擇要刪除的項目，按下鍵盤上的 Delete 按鍵，即可刪除該型態。



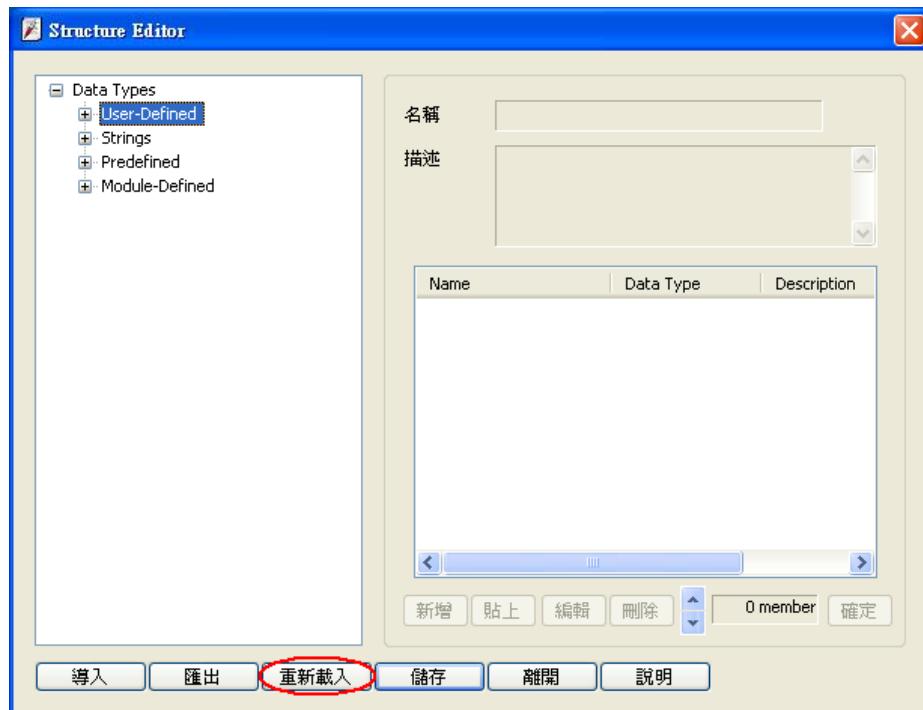
- 儲存修改結果：

修改完成後必須按下主畫面的【儲存】按鈕，此時重啟 EasyBuilder 即可看到變更的結果。



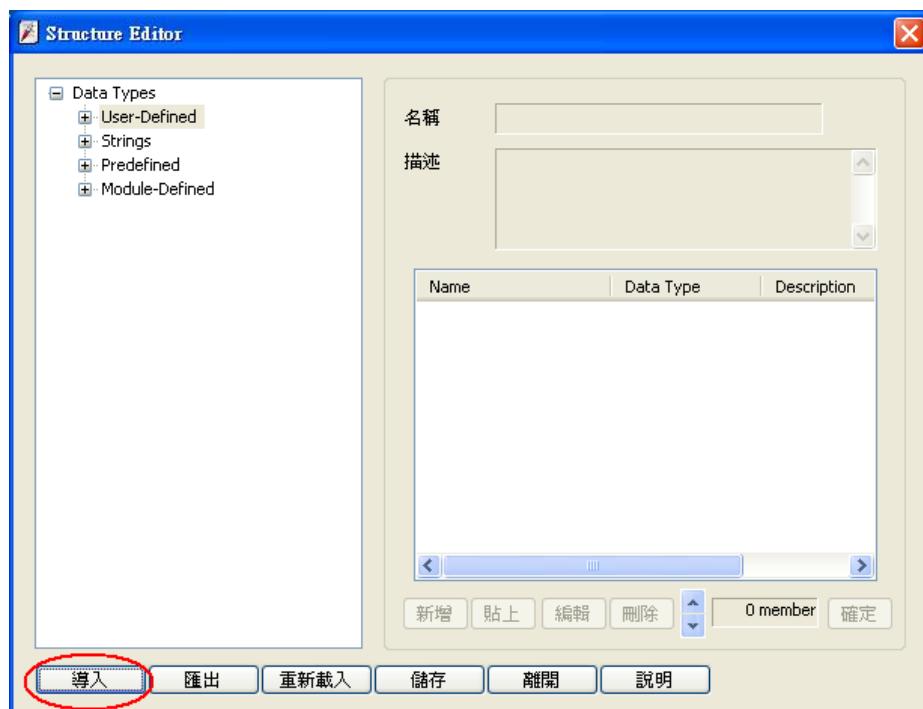
- 重新編輯：

若要放棄所有修改重新編輯，按下主畫面的**【重新載入】**按鈕。



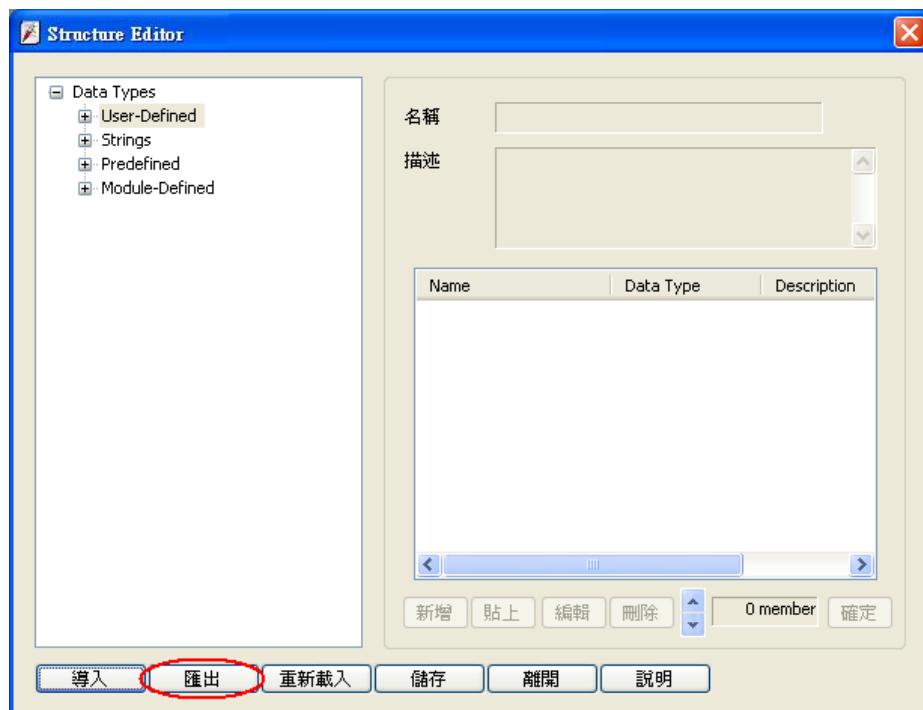
- 導入：

可使用導入功能開啟 TDF 檔案。



- **匯出：**

使用者可將編輯好的資料使用匯出功能匯出至 XXX.TDF 檔案，匯出的 TDF 檔可用在備份或是在其他 PC 上使用。

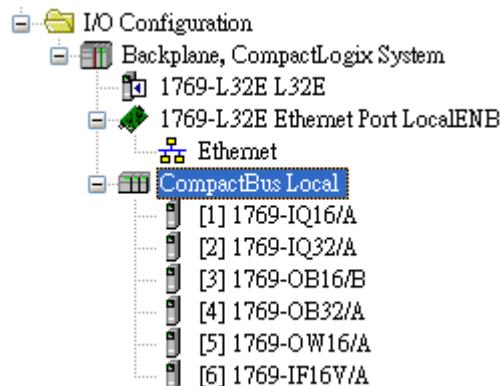


34.5 模組預設結構

模組預設結構 Module-Defined。

這裡示範如何建立一個模組的預設結構。

在 RSLogix5000 的 **I/O Configuration** 裡設定了 I/O 的模組：

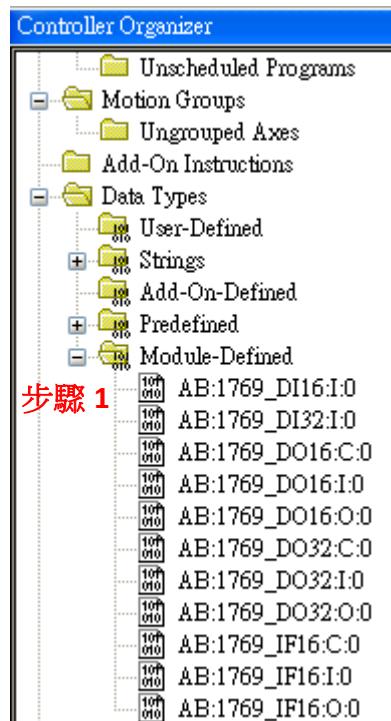


這些模組的 Tag 在輸出到 csv 檔案時並不會把結構列出來，所以我們必須幫它建立。

	A	B	C	D	E	F	G	H
7	TYPE	SCOPE	NAME	DESCRIPTION	DATATYPE	SPECIFIER	ATTRIBUTES	
8	TAG		Local:1:I		AB:1769_DI16:I:0			
9	TAG		Local:2:I		AB:1769_DI32:I:0			
10	TAG		Local:3:C		AB:1769_DO16:C:0			
11	TAG		Local:3:I		AB:1769_DO16:I:0			
12	TAG		Local:3:O		AB:1769_DO16:O:0			
13	TAG		Local:4:C		AB:1769_DO32:C:0			
14	TAG		Local:4:I		AB:1769_DO32:I:0			
15	TAG		Local:4:O		AB:1769_DO32:O:0			
16	TAG		Local:5:C		AB:1769_DO16:C:0			
17	TAG		Local:5:I		AB:1769_DO16:I:0			
18	TAG		Local:5:O		AB:1769_DO16:O:0			
19	TAG		Local:6:C		AB:1769_IF16:C:0			
20	TAG		Local:6:I		AB:1769_IF16:I:0			
21	TAG		Local:6:O		AB:1769_IF16:O:0			
22								

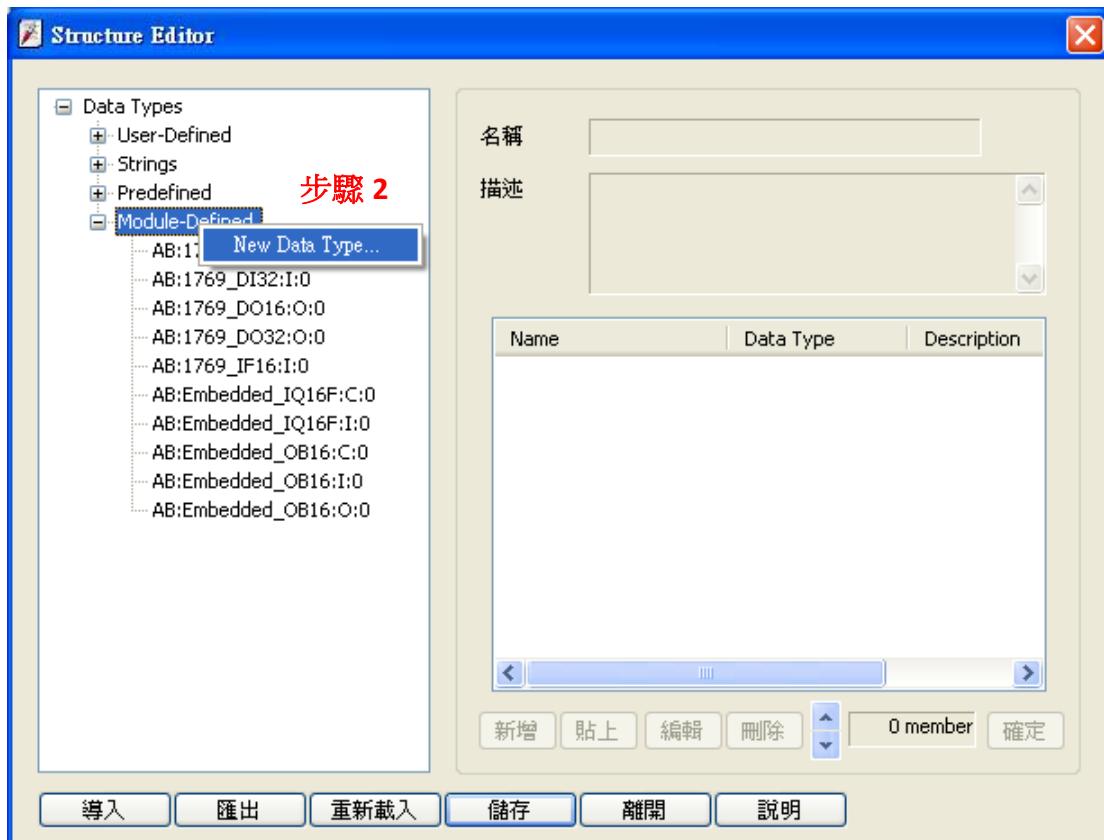
步驟 1.

在 RSLogix5000 的[Controller Organizer]»[Data Types]»[Module-Defined] 對模組的 DataType 雙擊滑鼠左鍵，就會彈出對話框顯示模組的 Data Type 的 Members。複製 Members 裡的 Name 和 Data Type。



步驟 2.

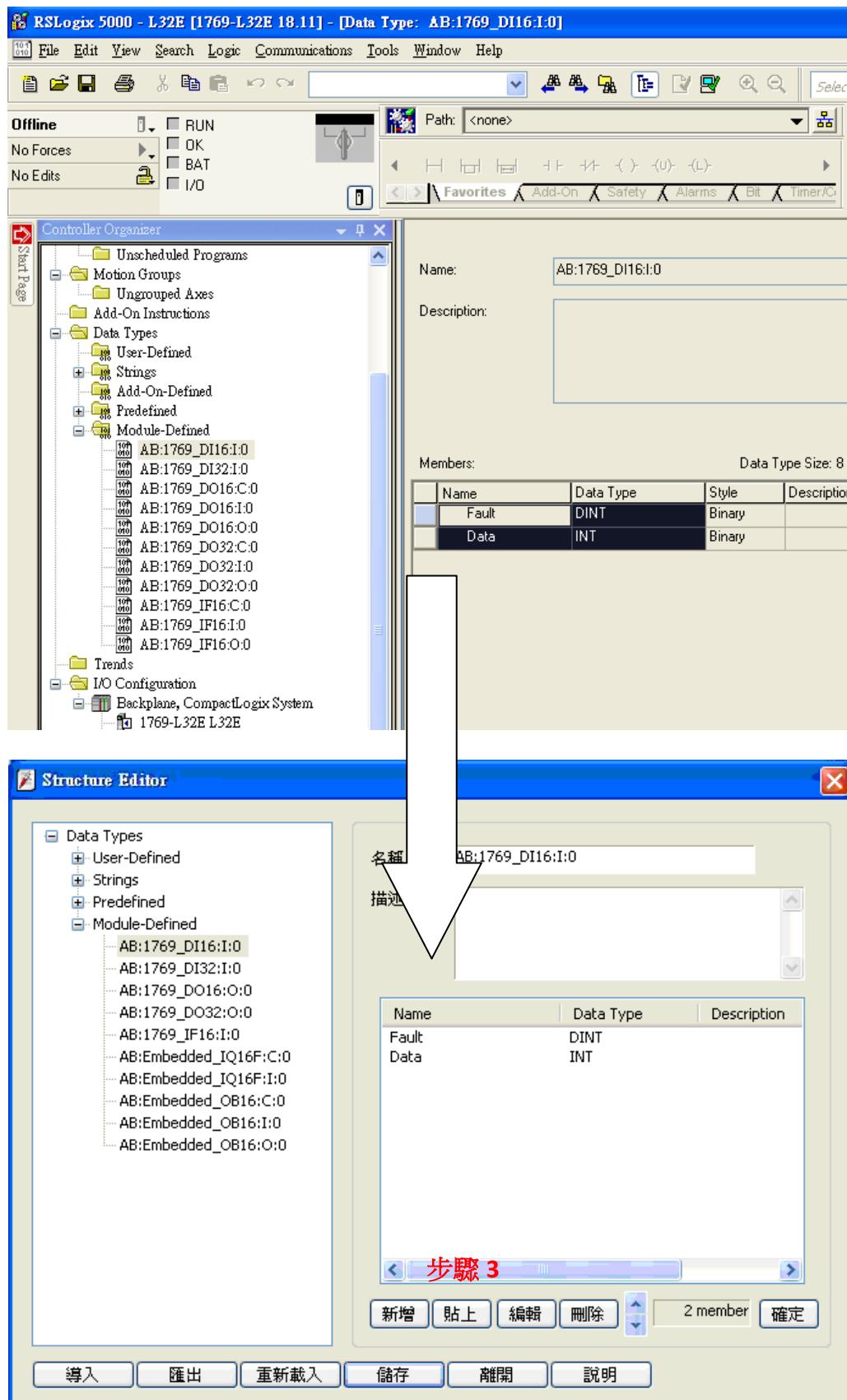
在 EasyBuilder 的 Structure Editor.exe 對著 **[Module-Defined]** 按下滑鼠右鍵，點選 **[New Data Type]**



在 New Data Type 的 Name 寫入 Module-Defined Name

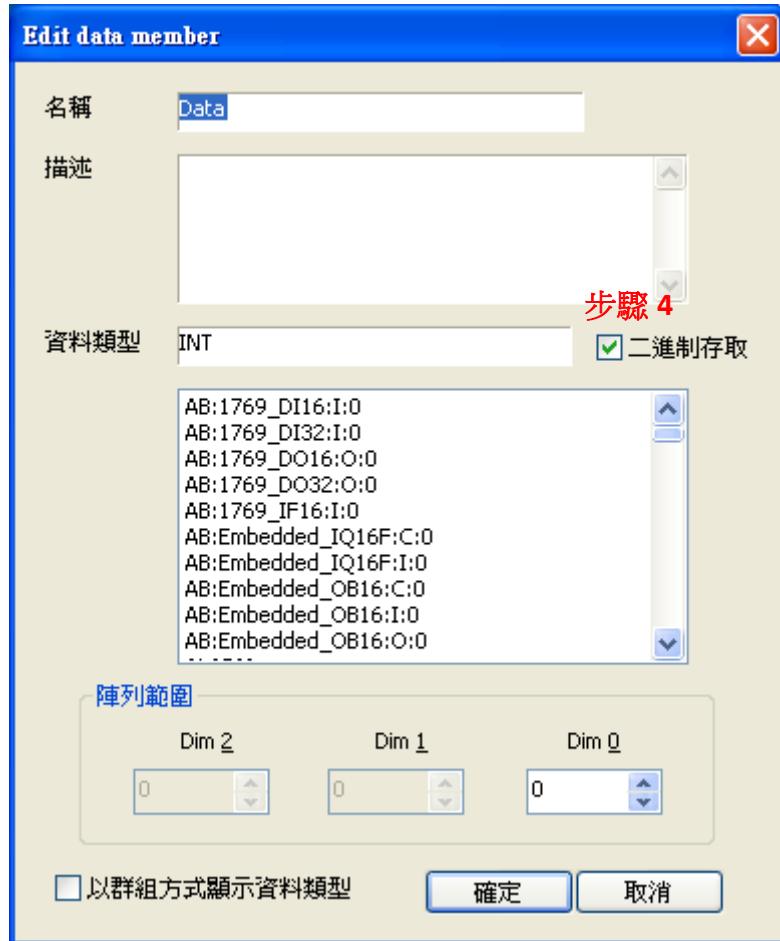
步驟 3.

按下 [Paste] 按鈕，在對話框中按 Ctrl+V 把 Name 和 Data Type 貼上。



步驟 4.

點選 Data 再按下 【編輯】按鈕，因為模組的資料可以用 bit 來操作，所以這裡要勾選【二進制存取】，按下【確定】回到 Structure Editor。



按下【確定】完成設定。

第三十五章 EasyWatch

35.1 EasyWatch 概述

EasyWatch 可以透過 PC 監看或設定 HMI 和 PLC 內的位址數值，同時也可以進行巨集指令的呼叫，更方便使用者進行除錯及遠端監控使用。

以下範例說明，使用者如何透過 EasyWatch 查看設定值及資料的正確性。在 EasyBuilder 新增【數值輸入】物件，位址設定成 LW-10，再到 EasyWatch 新增相同位址，當執行監視時，狀態為已連結，且顯示正確數值，表示已連線，即可開始監看。



當系統暫存器 **[LB-9044 (禁止遠端控制)]** 或 **【系統參數設定】»【系統設定】»【禁止遠端 HMI 連接】** 被設定時，將無法使用 EasyWatch 功能監控。

35.2 基本功能介紹

35.2.1 基本功能

項目	敘述
檔案	新增 開啟一個新的 EasyWatch 檔案。
	開啟 開啟已編輯的 EasyWatch 檔案。
	儲存檔案 儲存 EasyWatch 檔案設定。
	另存新檔 可將 EasyWatch 檔案設定，儲存成 ewt 格式。
	離開 關閉 EasyWatch。
編輯	剪下 剪下選取的物件至剪貼簿中。
	複製 複製選取的物件至剪貼簿中。
	貼上 貼上剪貼簿中的物件。
物件	新增物件 可新增監視物件或巨集物件。
	刪除物件 選擇欲刪除的物件，系統會彈出訊息，確認是否刪除。
	修改物件 選擇欲修改的物件，即可修改內容。
	HMI 管理器 對 HMI 進行新增、修改、移除的管理。
	執行 選擇欲執行的物件，即可執行此物件。
	停止 選擇執行中的物件，即可停止此物件。
說明	說明主題 提供基本功能的操作方法，供使用者參考。
	關於 EasyWatch

	顯示此版本資訊。
--	----------

35.2.2 快速工具



開啟新增：開啟一個新的 EasyWatch 檔案。



開啟舊檔：開啟已編輯的 EasyWatch 檔案。



儲存檔案：儲存 EasyWatch 檔案設定。



剪下：剪下選取的物件至剪貼簿中。



複製：複製選取的物件至剪貼簿中。



貼上：貼上剪貼簿中的物件。



執行：選擇欲執行的物件，即可執行物件。



停止：選擇執行中的物件，即可停止物件。



刪除物件：選擇欲刪除的物件，即可刪除物件。



監視物件：新增監視物件。



巨集物件：新增巨集物件。



HMI 管理器：對 HMI 進行新增、修改、移除的管理。



說明主題：提供基本功能的操作方法，供使用者參考。

35.3 監視物件設定

35.3.1 新增監視物件

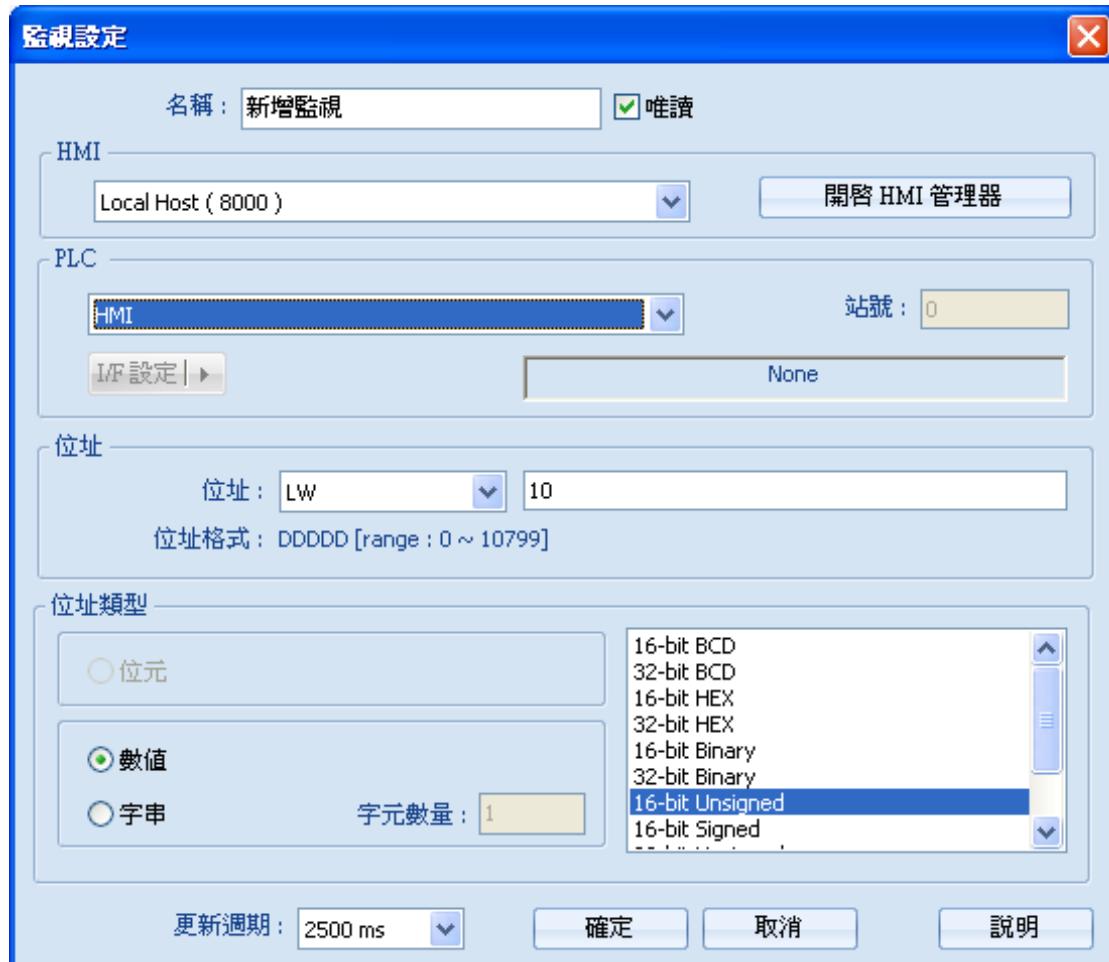
系統提供兩種方式新增物件：

1. 在基本工具列選擇 **【物件】** » **【新增物件】** » **【新增監視物件】**。



2. 在快速工具列選擇 **【新增監視物件】** 圖示。

35.3.2 監視物件設定



[Name]: 對物件命名，名稱不可重複。

[Read-only]: 若物件設為唯讀時，將不能設定該位址的數值。

[HMI]: 選擇欲監視的 HMI。

[PLC]: 設定欲預監位址所屬 PLC 的類型、站號及連線方式。

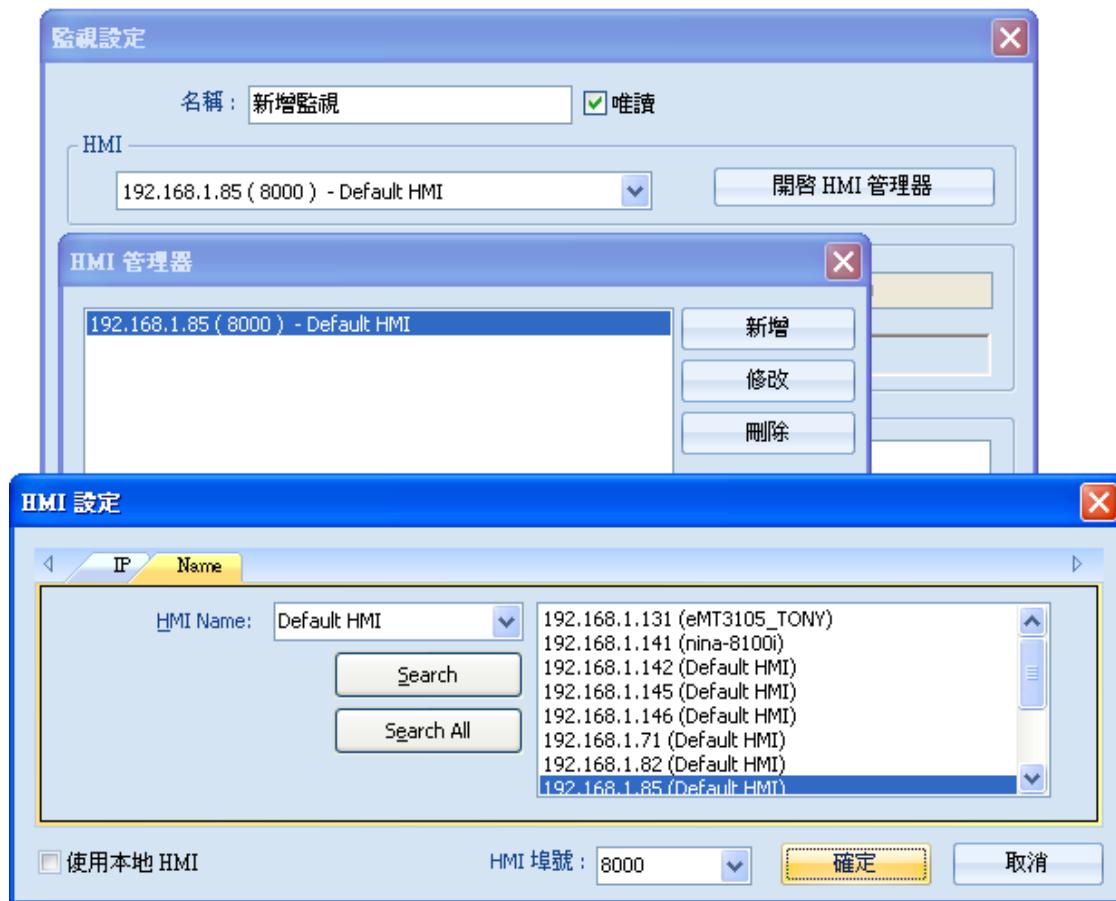
[Address]: 設定欲監視位址的類型及位址。

[Address Type]: 將依照位址類型列出可以選擇的顯示方式，執行時會依照顯示方式來解析並顯示該位址。

[Update Period]: 設定監視物件的更新週期，若同時運行過多的物件將導致誤差與延遲。

35.3.3 新增監視設定

- 選擇 **HMI**：先選擇欲操作的 HMI，若 HMI 不存在，可透過下列步驟來新增：
點選 **開啟 HMI 管理器** » **新增**，可透過網路搜尋 HMI，選擇 **確定** 即可完成新增動作。

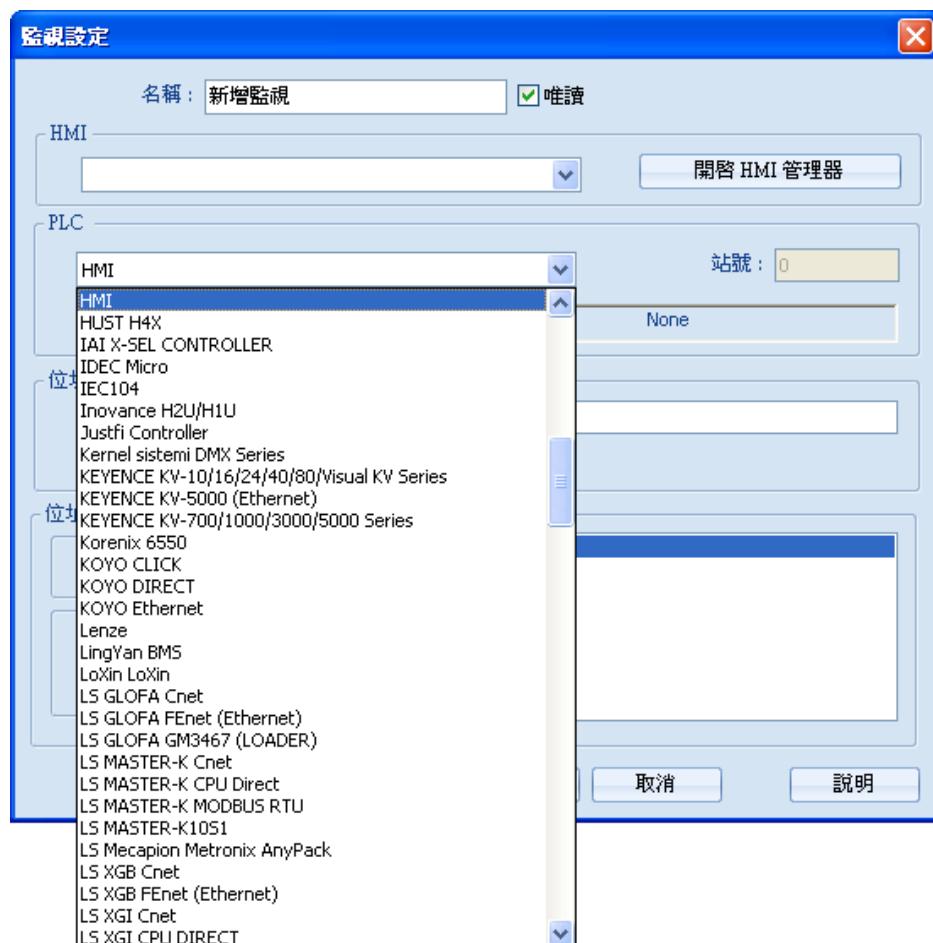


也可勾選 **[使用本地 HMI]**，將以 PC 上所模擬的程式為監視設備。



2. 選擇 PLC：可選擇直接操作 HMI 或 PLC。

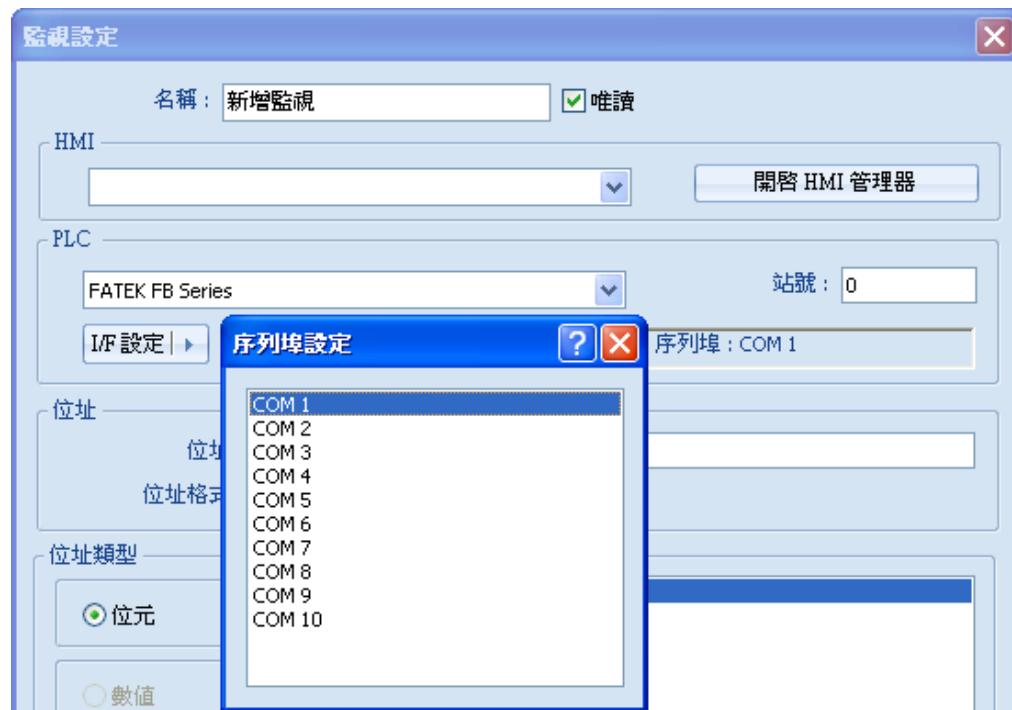
若選擇 HMI，即可直接對本機 HMI 操作。



若選擇 PLC 時，PLC 連結方式 (I/F 設定) 可使用 【序列埠】，或是【乙太網路】。



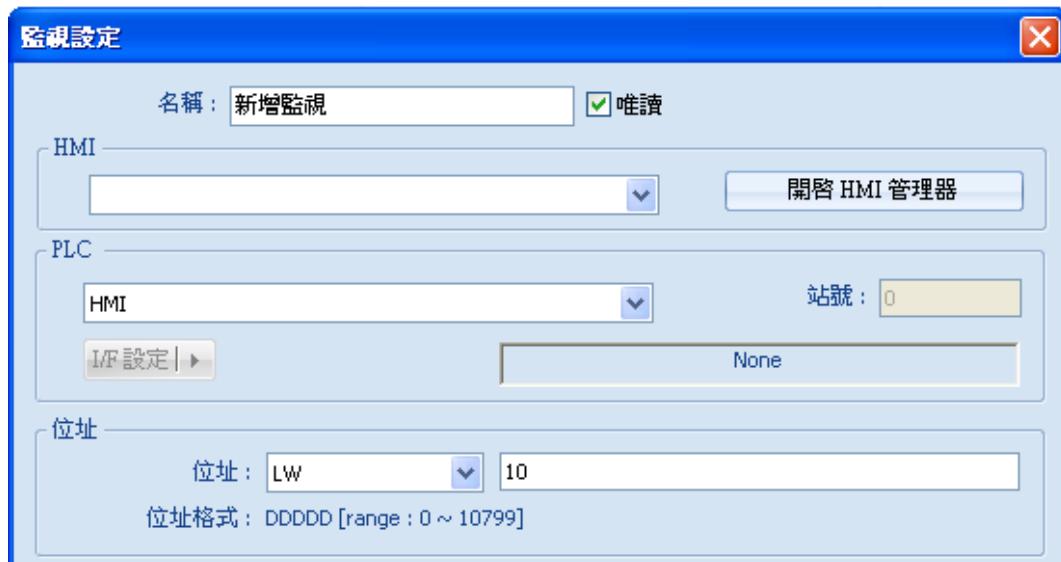
[序列埠]：點擊 **I/F 設定** 會跳出序列埠選項可供選擇。



[乙太網路]：點擊 **I/F 設定** 會跳出 **IP 位址設定** 欄位。



3. 設定位址：設定欲監控的位址及類型。



4. 位址類型：當選用字元類型時，可設定該位址為數值或字串。

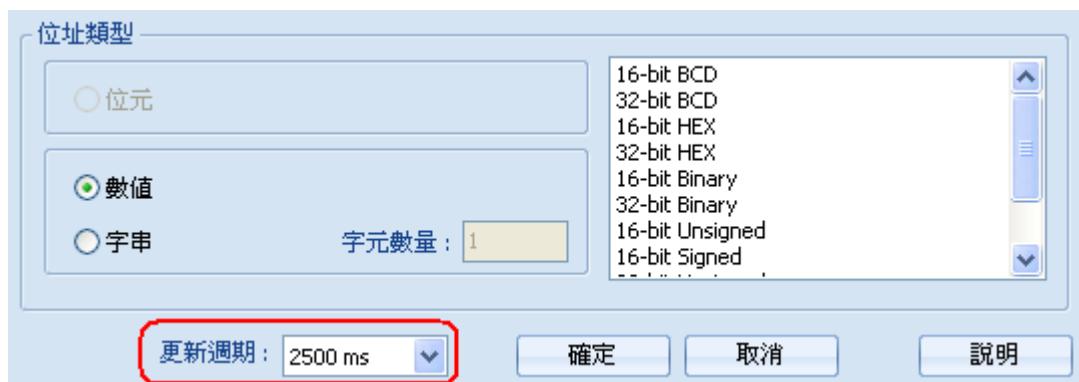
[數值]：可選擇欲監視的位址的資料格式。



[字串]：可設定 ANSI、UNICODE、高/低反向三種格式的資料。並可由 **[字元數量]** 設定欲讀取的字元數量。



5. 更新週期：設定監視物件的更新週期。可設定範圍從 500ms 至 5000ms。



35.4 巨集物件設定

35.4.1 新增巨集物件

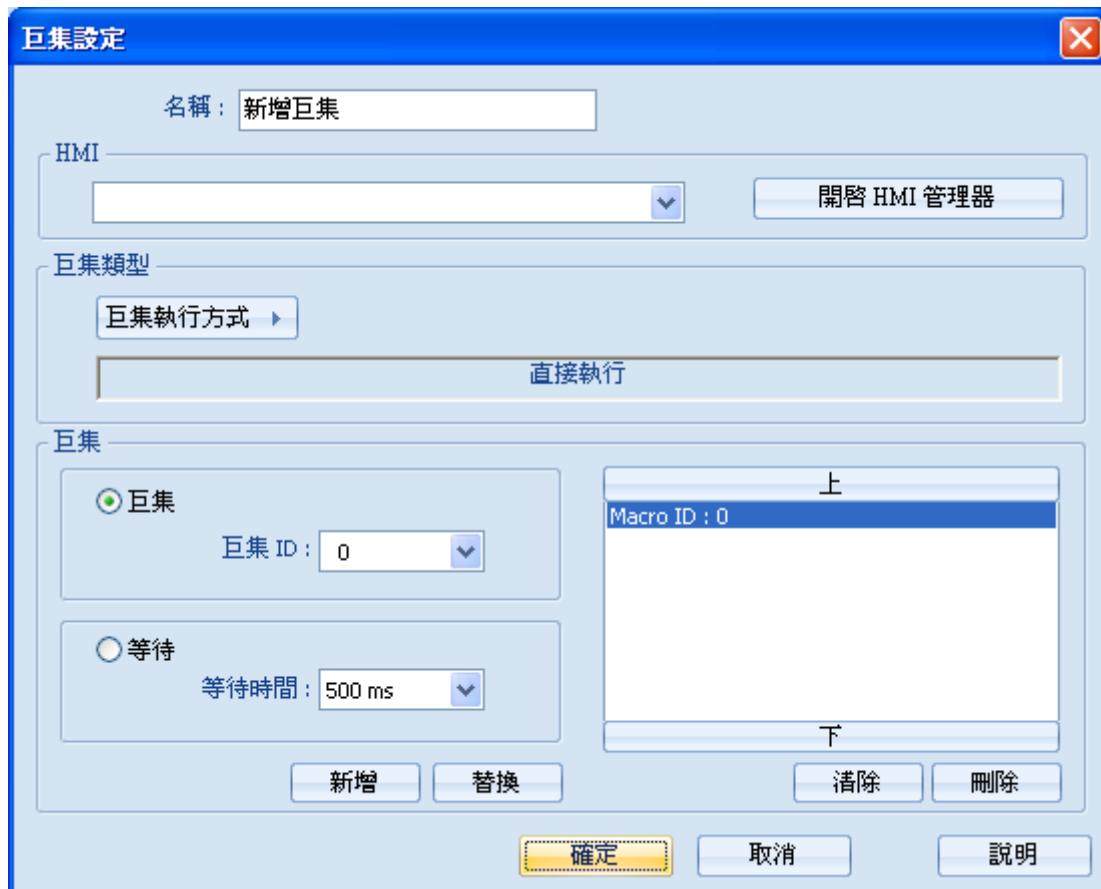
系統提供兩種方式新增物件：

1. 在基本工具列選擇【物件】»【新增物件】»【新增巨集物件】。



2. 在快速工具列選擇新增巨集物件圖示。

35.4.2 巨集物件設定



[名稱]：對物件命名，名稱不可重複。

[HMI]：選擇欲監視的 HMI 設備。

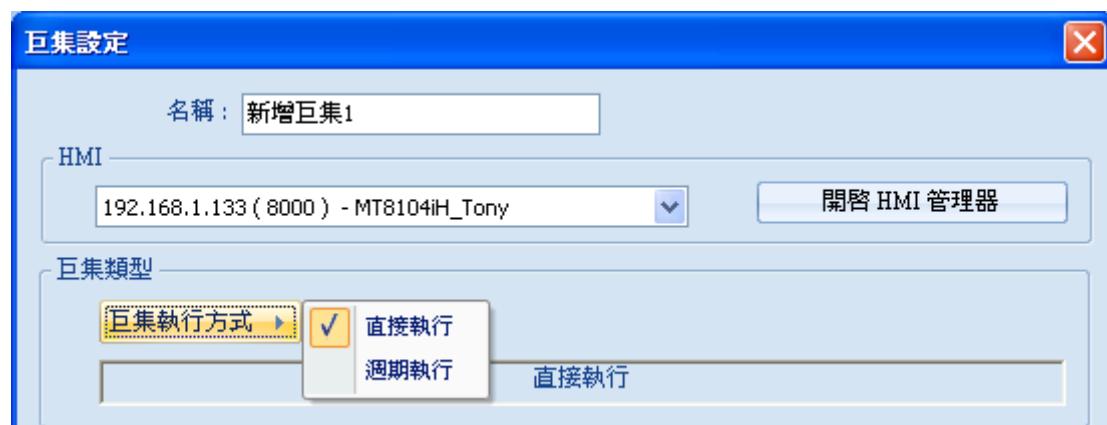
[巨集類型]：巨集執行的方式可分為直接呼叫和週期呼叫。

[巨集]：每個巨集物件可執行數個巨集指令，且巨集與巨集執行之間可設定間隔時間。

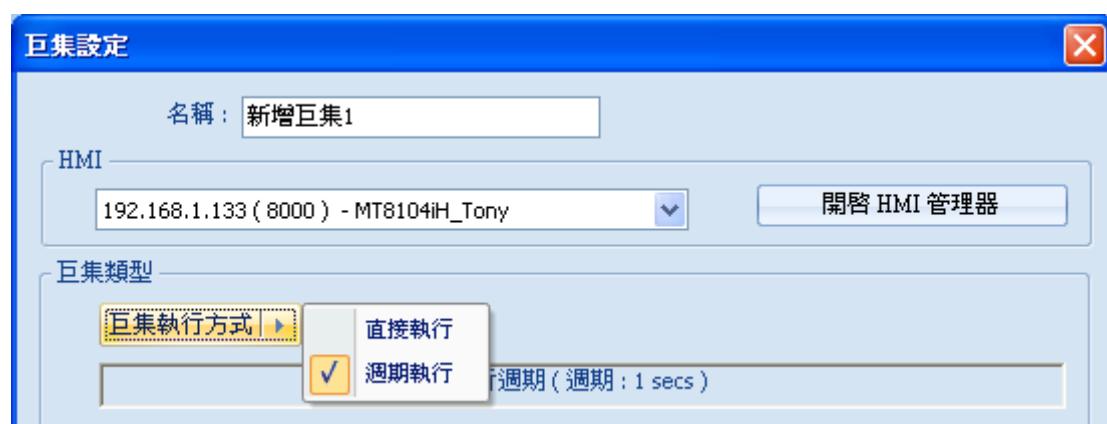
35.4.3 新增巨集設定

1. **HMI**：可參照“[章節 35.3.3 新增監視設定](#)”。
2. **巨集類型**：可選擇直接執行或週期執行。

[直接執行]：巨集指令會直接執行，並執行一次。



[週期執行]：可設定巨集指令執行的週期時間。



假設在 [執行週期] 設定為 5 秒，當執行完所有巨集指令，將於 5 秒後重新執行此巨集物件。



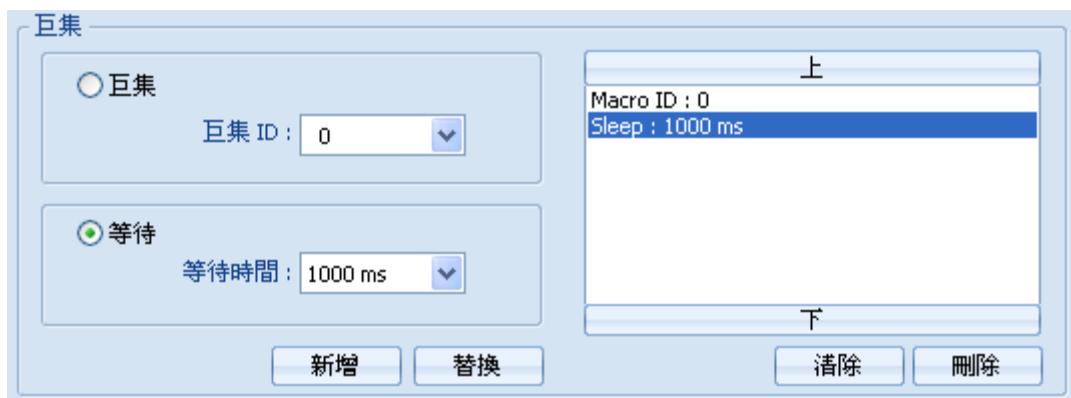
3. 巨集

包含執行【巨集】與【等待】時間。

【巨集】：選擇要執行的巨集 ID，點擊【新增】即可新增到巨集列表中。



【等待】：選擇等待時間，在執行完一個巨集指令後，等待所設定的時間，再執行下一個巨集指令。點選【新增】或【替換】可新增或取代巨集列表中所選取的巨集。



35.5 HMI 設定

35.5.1 HMI 設定

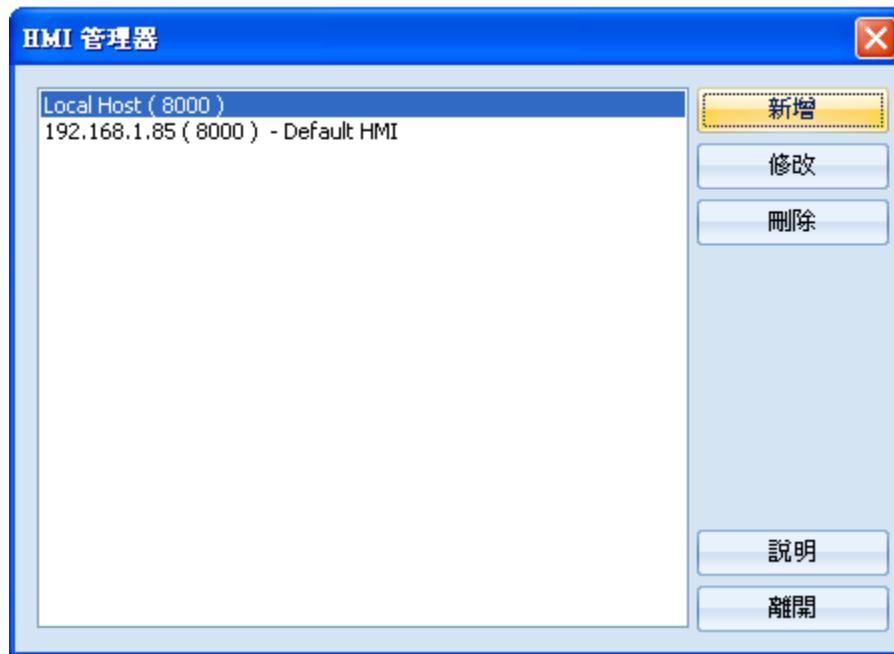
系統提供兩種方式開啟 HMI 設定：

1. 在基本工具列選擇 **[物件] » [HMI 管理器]**



2. 在快速工具列選擇 HMI 管理器圖示。

35.5.2 HMI 管理器設定

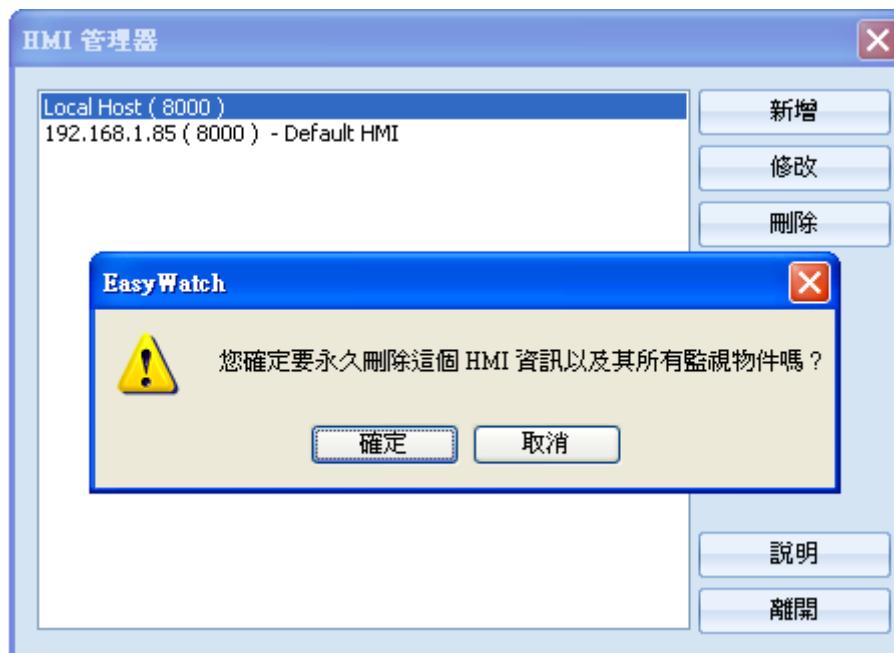


EasyWatch 可支援同時監控不同 HMI 上的位址，方便使用者管理數個 HMI。

[新增]：可參照“[章節 35.3.3 新增監視設定](#)”。

[修改]：選擇欲修改 HMI 項目即可修改。

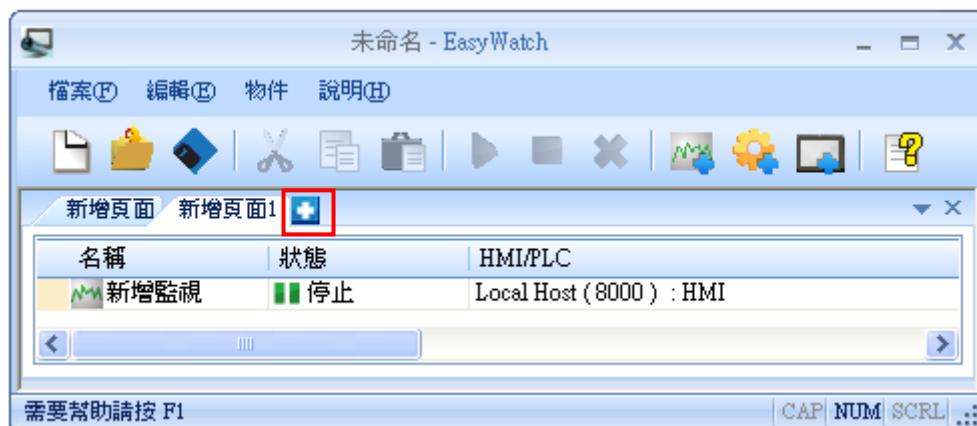
[刪除]：選擇欲移除 HMI 項目，確認後即可移除。



35.6 物件顯示列表

35.6.1 頁面設定

新增頁面：點選“+”來新增頁面。



刪除頁面：點選頁面右上角“X”並經確認後即可刪除該頁面。



重新命名：在頁面的名稱上雙擊滑鼠左鍵後，即可重新命名。



35.6.2 物件顯示欄位



[名稱]：顯示物件名稱，並且透過圖形顯示，可以方便使用者識別物件的種類。

[狀態]：顯示目前物件的執行狀態，分別有 [連結中]、[已連結] 或是 [停止]，同時也可顯示錯誤訊息。若該 HMI 不在線上或是連接埠號輸入錯誤，會顯示“無法發現 HMI”的訊息；若為監視物件且位址設定有誤，則顯示“位址錯誤”的訊息。

[HMI/PLC]：顯示目前物件所操作 HMI/PLC 的相關資訊。

[位址] / [位址類型]：若為監視物件，將顯示其位址相關設定資料。

[更新週期]： 設定監視物件的更新週期。

[數值]：若為監視物件，且狀態為已連結時，將顯示目前 HMI 上該位址的數值。當監視物件不具唯讀屬性時，也可透過修改該欄位來設定監視位址的內容。若為巨集物件，並且型態為 [直接執行]，在數值欄位上會顯示按鈕，點擊後可直接執行該巨集指令。

物件顯示欄位順序可依使用者喜好自行排列選擇。

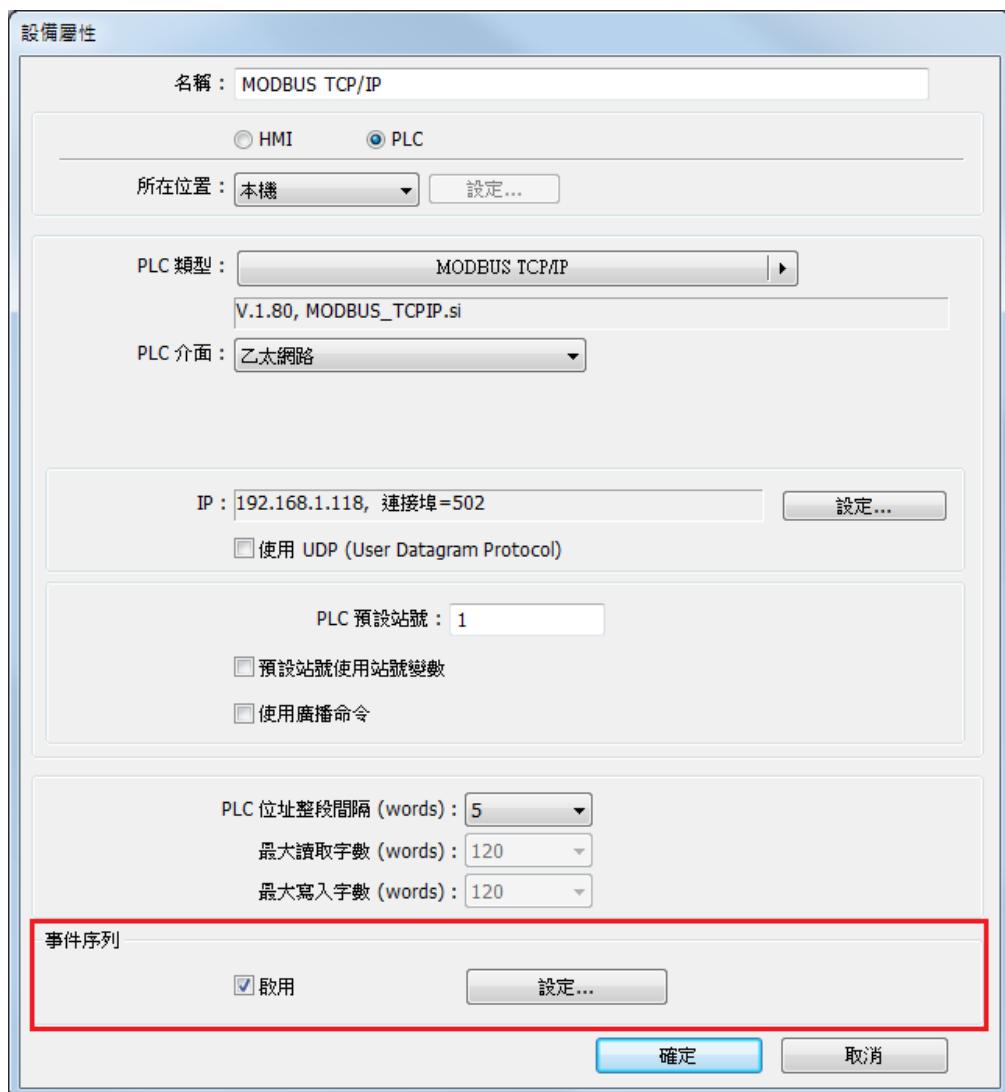


第三十六章 事件序列記錄

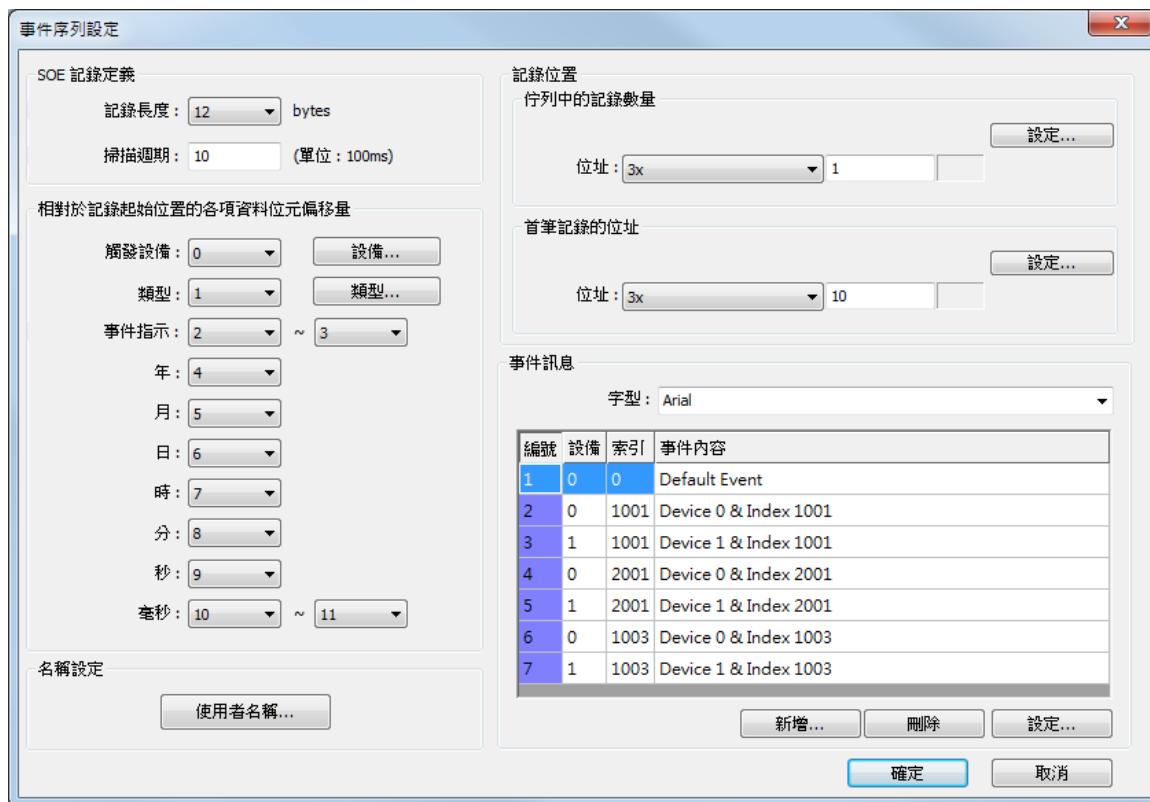
36.1 介紹

事件序列記錄 (SOE) 為 Sequence Of Events 的縮寫，主要用於當事件發生時，可記錄多個信號變化的精確時間，以便區分多個信號變化的先後順序。當事件發生時，PLC 會將發生的資料暫存於記錄位址，並在佇列位址寫入信息的數量，HMI 會依掃描週期去檢查佇列位址，若數值不為 0 即會去讀取記錄位址的資料，內容包含事件發生時的精確時間、事件類型等等。

在 EasyBuilder 的 [系統參數設定] > [設備清單] > [設備屬性] 可啟用事件序列功能，事件序列資料格式可由使用者自行定義。啟用後便可設定事件序列顯示物件，方便使用者檢視事件序列記錄。



36.2 事件序列設定



設定	敘述
SOE 記錄定義	
記錄長度	設定一筆事件序列資料的總長度，範圍：10 ~ 128 bytes。
掃描週期	設定系統輪詢事件序列資料的頻率。以 100 毫秒為單位，範圍：10 ~ 32767。
相對於記錄起始位置的各項資料位元偏移量	
由於各廠牌 PLC 的事件序列資料格式定義皆不相同，因此使用者需手動定義各欄位的資料格式之開始與結束位址，以便系統解讀每一筆事件序列的資料。目前資料格式定義如下：	
觸發設備	設定設備索引與相對應的名稱，索引範圍: 0 ~ 255。
類型	設定事件類型索引與相對應的名稱，索引範圍: 0 ~ 255。
事件指示	事件的索引值。
年	事件發生的日期 (西元年)。
月	事件發生的日期 (月份)。
日	事件發生的日期 (日期)。
時	事件發生的時間 (時)。
分	事件發生的時間 (分)。

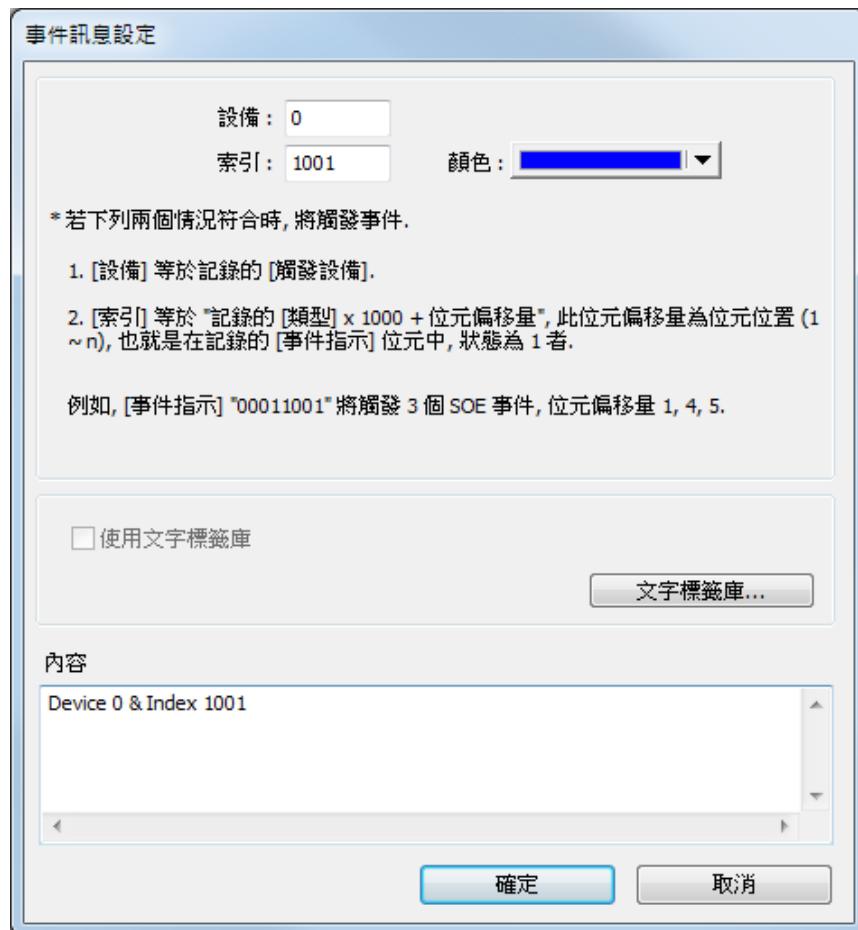
秒	事件發生的時間 (秒)。
毫秒	事件發生的時間 (毫秒)。
名稱設定	
使用者名稱	設定登入的使用者名稱，範圍：1 ~ 12。
記錄位置	
佇列中的記錄數量	事件發生時，事件序列的數量將被寫入此暫存器位址。例如，首筆記錄的位址設為 LW-0，且資料長度為 12 bytes (6 words)，若發生兩筆事件，則 LW-0 至 LW-5 即為第一筆資料，LW-6 至 LW-11 則為第二筆資料，依此類推。 此位址數值輸入完畢後，系統將自動歸零。
首筆記錄的位址	設定事件訊息的讀取位址，即為事件序列資料格式的起始位址。
事件訊息	
字型	設定事件顯示時的字型。
事件訊息登錄	<p>設定設備與事件訊息索引之相對應的訊息內容。</p> <p>當事件發生時，系統將比對事件訊息登錄內 [設備] 與 記錄位址 [觸發設備] 欄位的值，以及事件訊息登錄內 [索引] 與 記錄位址 [類型] $\times 1000 +$ [事件指示之位元偏移量] 的值。</p> <p>當兩者條件皆相符時，才能顯示相對應的事件訊息內容。若發生的事件之 [設備] 或 [索引] 有其一未被登錄，將顯示事件訊息索引值為 0 的內容。</p> <p>例如，當事件序列資料的 [類型] 值為 1，[事件指示] 值為 5 (二進位：101)，則將觸發事件訊息登錄內索引值為 1001 與 1003 的事件訊息。</p> $1 \times 1000 + \text{Bit } 1 \Rightarrow \text{事件索引 } 1001$ $1 \times 1000 + \text{Bit } 3 \Rightarrow \text{事件索引 } 1003$

事件序列功能於 EasyBuilder8000 V4.65.04 之前的版本與 V4.65.05 之後的版本有所不同，主要的變更在於計算事件索引值的方式，詳情變動請參考下表：

EB8000 V4.65.04 之前	<p>[事件類型] × 16 + [事件索引之位元偏移量]</p> <p>例如，當事件序列資料的 [事件類型] 值為 1，[事件索引] 值為 5 (二進位：101)，則將觸發事件登錄內索引值為 17 與 19 的事件訊息。</p> <p>$1 \times 16 + \text{Bit } 1 \Rightarrow \text{事件索引} 17$</p> <p>$1 \times 16 + \text{Bit } 3 \Rightarrow \text{事件索引} 19$</p>
EB8000 V4.65.05 之後	<p>[類型] × 1000 + [事件指示之位元偏移量]</p> <p>例如，當事件序列資料的 [類型] 值為 1，[事件指示] 值為 5 (二進位：101)，則將觸發事件訊息登錄內索引值為 1001 與 1003 的事件訊息。</p> <p>$1 \times 1000 + \text{Bit } 1 \Rightarrow \text{事件索引} 1001$</p> <p>$1 \times 1000 + \text{Bit } 3 \Rightarrow \text{事件索引} 1003$</p>

關於事件訊息的設定請參考下頁。

事件訊息設定視窗：

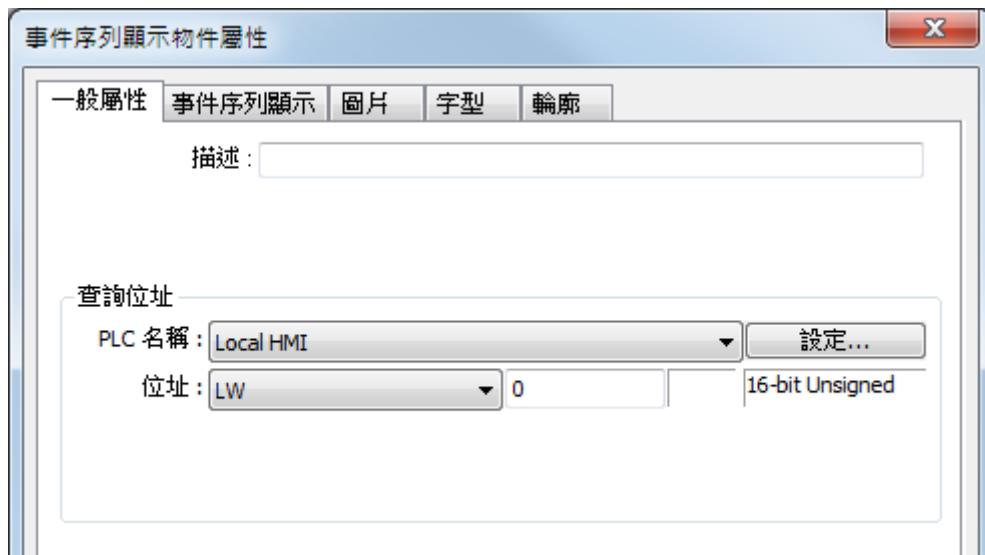


設定	敘述
設備	設定設備類型的索引值，索引範圍: 0 ~ 255。
索引	設定事件訊息的索引值。
顏色	設定事件顯示時的顏色。
使用文字標籤庫	若勾選，訊息文字的內容將來自文字標籤庫。
文字標籤庫	呼叫文字標籤庫來建立、更改或刪除文字標籤。
內容	輸入欲顯示的訊息內容。

36.3 事件序列顯示物件

透過事件序列顯示物件可以讓使用者瀏覽事件訊息。在設備屬性中啟用事件序列後，系統才會開放事件序列顯示物件的設定。

事件序列顯示物件的【一般屬性】頁籤設定如下：



一般屬性頁籤可設定事件序列的查詢位址。查詢位址將以連續 19 個字元作為相關控制與數據比對的來源。

名稱	位址	敘述
模式	查詢位址 n	(請參考下列表格)
狀態	查詢位址 n + 1	(請參考下列表格)
開始日期	查詢位址 n + 2 ~ 4	表示：年.月.日
開始時間	查詢位址 n + 5 ~ 7	表示：時.分.秒
開始毫秒	查詢位址 n + 8	
類型	查詢位址 n + 9	範圍：0 ~ 9
設備	查詢位址 n + 10	範圍：0 ~ 99
使用者名稱	查詢位址 n + 11	範圍：1 ~ 12
結束日期	查詢位址 n + 12 ~ 14	表示：年.月.日
結束時間	查詢位址 n + 15 ~ 17	表示：時.分.秒
結束毫秒	查詢位址 n + 18	

[模式] 內的數值各代表如下：

數值	模式
1	以開始日期 (查詢位址 n + 2 ~ 4) 內的數值為查詢依據。當 [狀態] 設為 4 時，需搭配結束日期 (查詢位址 n + 12 ~ 14) 作為一個查詢範圍。
2	以開始時間 (查詢位址 n + 5 ~ 7) 內的數值為查詢依據。當 [狀態] 設為 4 時，需搭配結束時間 (查詢位址 n + 15 ~ 17) 作為一個查詢範圍。
4	以開始毫秒 (查詢位址 n + 8) 內的數值為查詢依據。當 [狀態] 設為 4 時，需搭配結束毫秒 (查詢位址 n + 18) 作為一個查詢範圍。
8	以類型 (查詢位址 n + 9) 內的數值為查詢依據。
16	以設備 (查詢位址 n + 10) 內的數值為查詢依據。
32	以使用者名稱 (查詢位址 n + 11) 內的數值為查詢依據。

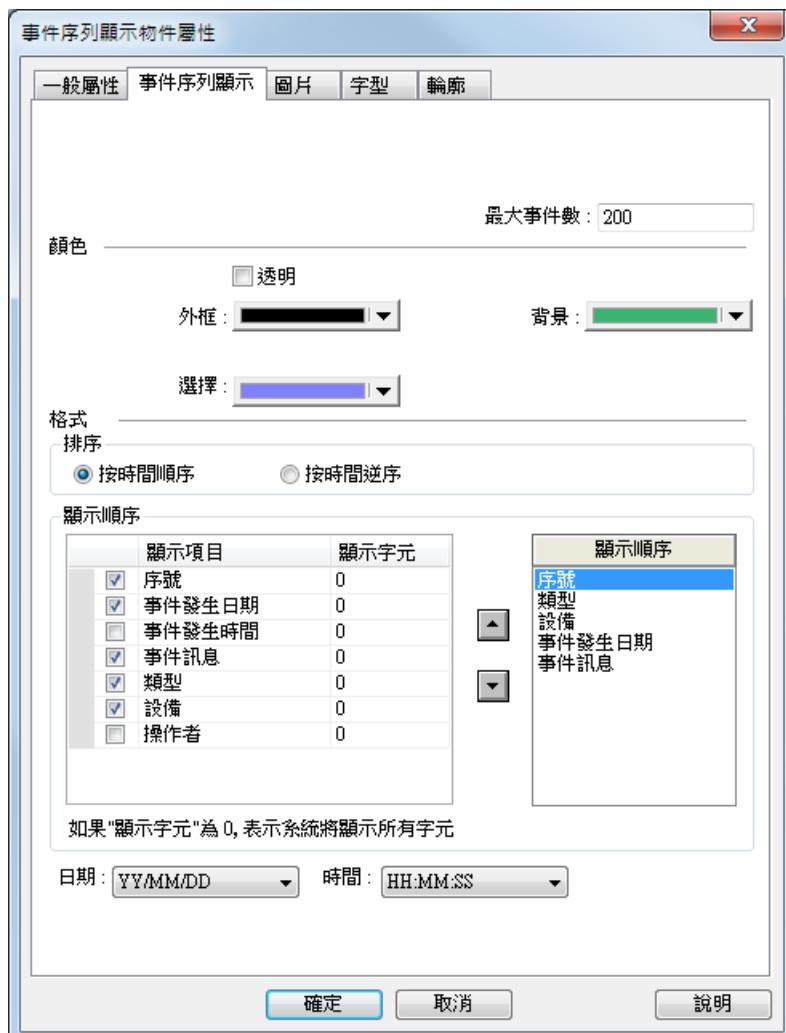
[狀態] 內的數值各代表如下：

數值	模式
0	顯示全部訊息。
1	只顯示等於特定模式下的訊息。
2	只顯示大於等於特定模式下的訊息。
3	只顯示小於等於特定模式下的訊息。
4	只顯示範圍內特定模式下的訊息。



- [類型]、[設備]、[使用者名稱] 僅支援查詢狀態為 0 (顯示全部) 與 1 (等於)。即無論 [狀態] 的設定為何，[類型]、[設備]、[使用者名稱] 都只會尋找與其數值相等的事件記錄。
- 當 [模式] 或 [狀態] 的數值改變時，才會重新觸發查詢功能。

事件序列顯示物件的 [事件序列顯示] 頁籤設定如下：



設定	敘述
最大事件數	設定欲顯示的事件數量，若事件觸發數量超出設定範圍，舊事件將依序被新事件覆蓋。
透明	若勾選，將不使用外框與背景的顏色。
外框	設定外框的顏色。
背景	設定背景的顏色。
選擇	設定事件被選取時，標示的框線顏色。
按時間順序	事件將由舊到新依序顯示，最近觸發的事件顯示於底部。
按時間逆序	事件將由新到舊依序顯示，最近觸發的事件顯示於頂部。
顯示順序	設定當事件觸發時，欲顯示的項目與字元數量。若字元設為 0，將顯示項目的所有內容。
日期	設定欲顯示日期資訊的格式。
時間	設定欲顯示時間資訊的格式。

第三十七章 MODBUS TCP/IP 閘道功能

37.1 MODBUS TCP/IP 閘道簡介

以往若要使用 SCADA (Supervisory Control and Data Acquisition) 軟體去存取與 HMI 連接的 PLC 資料時，需透過資料傳輸先將 PLC 資料傳送至 HMI 的本地位址，再於 PC 上使用 MODBUS TCP/IP 協議去讀取 HMI 的本地位址將 PLC 資料取回。

現在用戶可以透過 EasyBuilder 提供的 MODBUS TCP/IP 閘道功能，將 MODBUS 與 PLC 的位址預先設定對應後，即可以直接利用 MODBUS TCP/IP 通訊協議存取 PLC 上的資料。

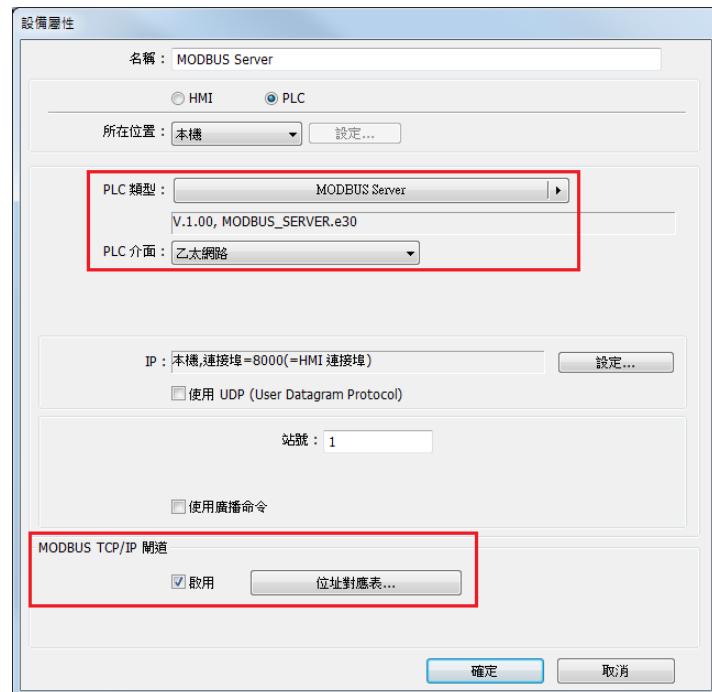


37.2 位址對應設定

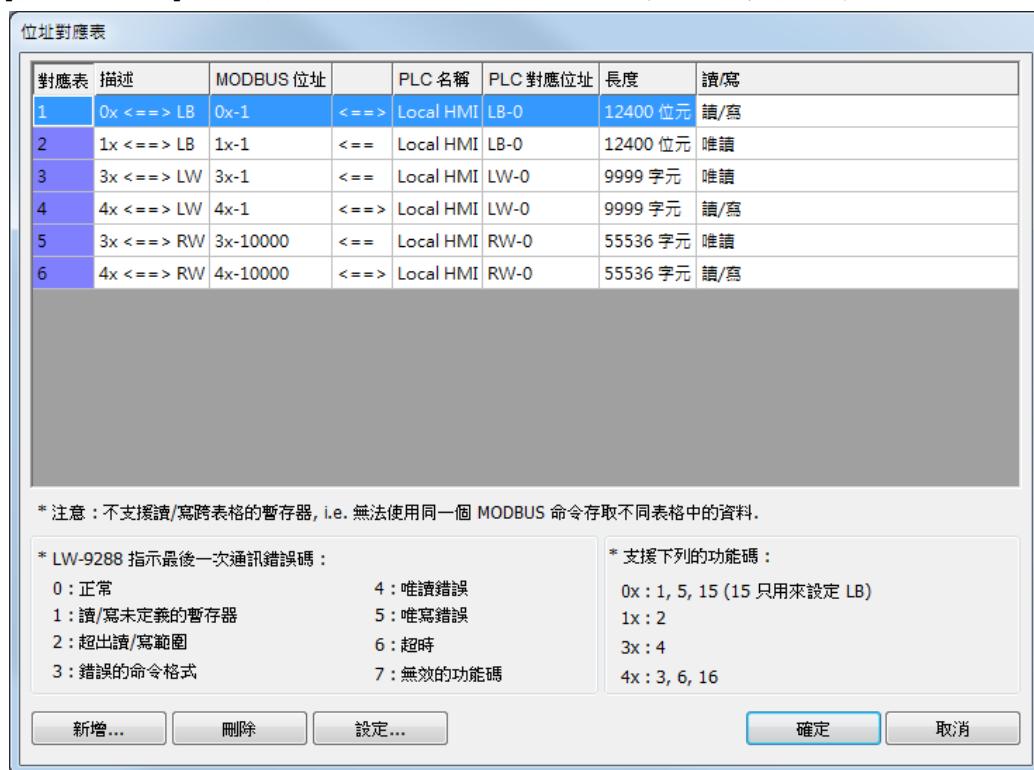
37.2.1 如何建立一個位址對應表

新增一個位址對應表，請依照下列步驟：

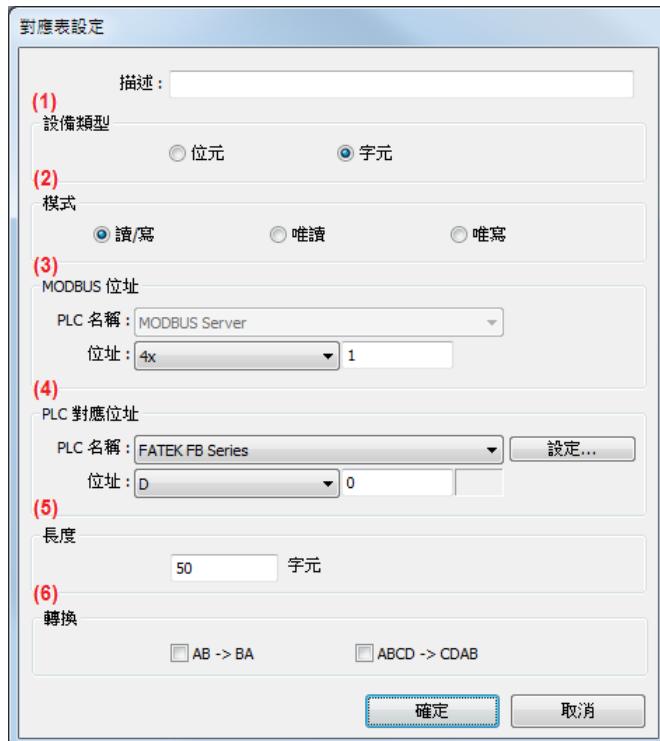
1. 於 [系統參數設定 \ 設備清單] 新增欲監控的 PLC 設備。(以 FATEK FB Series 為例)
2. 新增一個 MODBUS Server (乙太網路)，並啟用 [MODBUS TCP/IP 閘道]，如下圖：



3. 點選【位址對應表】按鈕後，會顯示預設的對應表，用戶可以依需求修改並新增其他對應表。



4. 假設，SCADA 需存取 FATEK FB Series PLC 的 D0 暫存器開始的連續 50 個位址，則設定如下：



- (1) 設定欲對應的暫存器類型，此範例為 [字元]。
- (2) 設定欲對應的暫存器之存取模式，此範例為 [讀/寫]。
- (3) 設定欲對應的 MODBUS 起始位址，此範例為 [4x1]。
- (4) 設定欲對應的 PLC 起始位址，此範例為 [D0]。
- (5) 設定欲對應位址的範圍大小，此範例為 [50]。
- (6) 選擇是否要高/低位元組或高/低字元組轉換。

對應表	描述	MODBUS 位址		PLC 名稱	PLC 對應位址	長度	讀/寫
1	Access D0 ~ D49	4x-1	<==>	FATEK FB Series	D-0	50 字元	讀/寫

上圖的設定內容說明 MODBUS Server 4x1 ~ 4x50 位址對應到 FATEK FB Series PLC 的 D0 ~ D49 位址。

5. 完成以上設定後，SCADA 只需利用 MODBUS TCP/IP 協議，發送讀/寫 4x1 ~ 4x50 位址的命令，即可以直接存取 FATEK FB Series PLC 的 D0 ~ D49 位址。

37.2.2 位址對應設定須知

- [MODBUS TCP/IP 閘道] 功能不支援使用 UDP。
- 只支援使用 MODBUS Server (乙太網路) 介面。
- 系統提供暫存器 LW-9288，可用來指示此功能資料傳送是否正常。

錯誤碼各表示：

數值	定義
0	正常
1	讀取或寫入未定義在位址對應表中的暫存器
2	讀取或寫入的位址範圍超出單一位址對應表所定義的數據長度 (或是讀取/寫入跨表格的暫存器)
3	命令格式未遵循 MODBUS TCP/IP 協議
4	修改只允許讀取的暫存器
5	讀取只允許寫入的暫存器
6	在設定的時間內無法得到 PLC 的正確回應
7	使用了 MODBUS Server 不支援的功能碼

- 各個對應表間定義的暫存器之位址範圍不可重複。
- 啟用 [MODBUS TCP/IP 閘道] 功能後，EasyBuilder 將取消 MODBUS Server 與 HMI 位址間原有的對應關係，包含：
 - (1) 0x, 1x 對應到 LB
 - (2) 3x, 4x 對應到 LW, RW

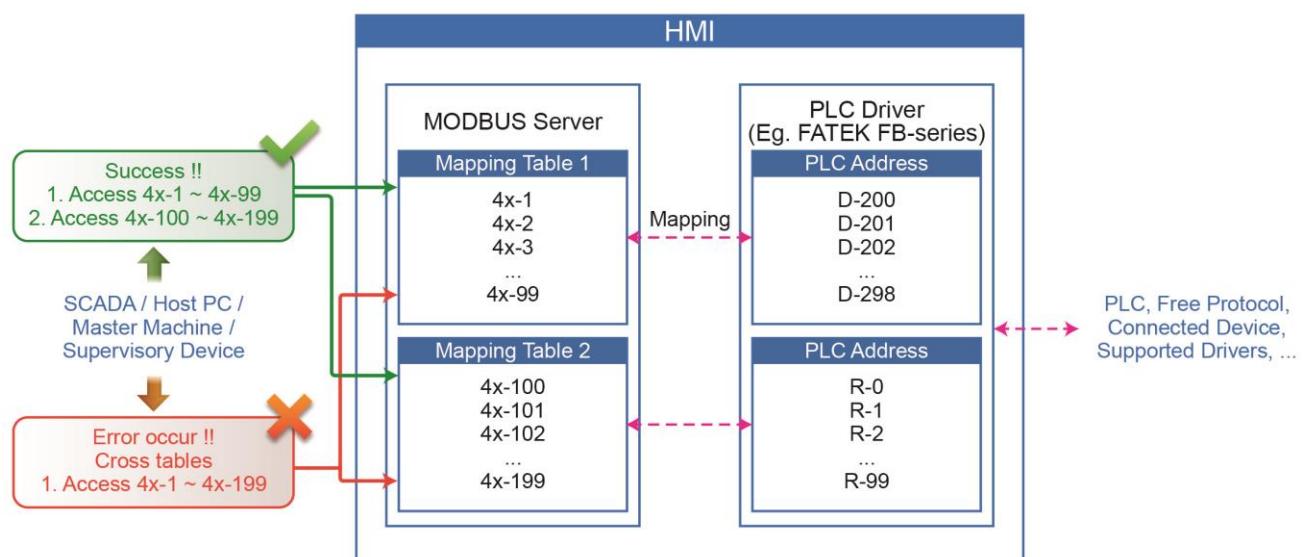
因此如需透過 0x, 1x, 3x, 4x 的命令來存取 LB 或 LW 的資料，仍需先將位址對應關係重新設定於【位址對應表】中，可參考下列設定內容。

對應表	描述	MODBUS 位址		PLC 名稱	PLC 對應位址	長度	讀/寫
1	0x <==> LB	0x-1	<==>	Local HMI	LB-0	12400 位元	讀/寫
2	1x <==> LB	1x-1	<==>	Local HMI	LB-0	12400 位元	唯讀
3	3x <==> LW	3x-1	<==>	Local HMI	LW-0	9999 字元	唯讀
4	4x <==> LW	4x-1	<==>	Local HMI	LW-0	9999 字元	讀/寫
5	3x <==> RW	3x-10000	<==>	Local HMI	RW-0	55536 字元	唯讀
6	4x <==> RW	4x-10000	<==>	Local HMI	RW-0	55536 字元	讀/寫

- SCADA 一次只能讀取/寫入一個對應表內的暫存器，即無法使用同一個 MODBUS 命令存取不同表格中的暫存器。

對應表	描述	MODBUS 位址		PLC 名稱	PLC 對應位址	長度	讀/寫
1	Access D200 ~ D298	4x-1	<==>	FATEK FB Series	D-200	99 字元	讀/寫
2	Access R0 ~ R99	4x-100	<==>	FATEK FB Series	R-0	100 字元	讀/寫

以上圖為例，於【對應表 1】設定 MODBUS 4x1 對應到 D200 位址，長度為 99；於【對應表 2】設定 MODBUS 4x100 對應到 R0 位址，長度為 100，若此時 SCADA 發出一道命令要一次讀取 4x1 ~ 4x199 長度為 199 的位址，因已經跨表格存取，此命令將不被 HMI 接受，應將命令分為兩道分別存取 4x1 ~ 4x99 和 4x100 ~ 4x199。如下圖：

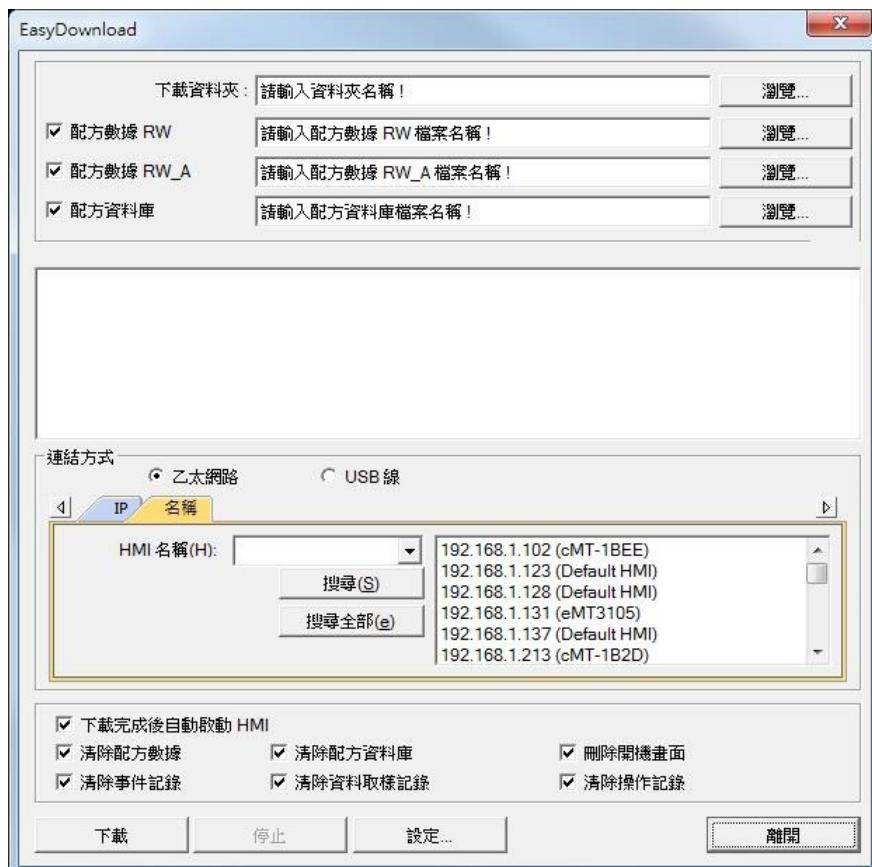


第三十八章 EasyDownload

38.1 概要

EasyDownload 程式可以透過乙太網路或 USB 線來下載於 EasyBuilder Pro 編輯軟體建立的工程檔案下載資料。需先透過 EasyBuilder Pro 編輯軟體的【工具】選單 » [建立使用在 USB 碟與 SD 卡所需的下載資料] 建立工程檔案下載資料。

38.2 設定



設定	敘述
下載資料夾	選擇欲下載的工程檔案下載資料。
配方數據 RW	若勾選，可瀏覽欲下載的配方數據 (.rcp) 檔案。
配方數據 RW_A	若勾選，可瀏覽欲下載的配方數據 (.rcp) 檔案。
配方資料庫	若勾選，可瀏覽欲下載的配方資料庫 (.db) 檔案。
乙太網路	使用乙太網路傳送檔案至 HMI。

USB 線	使用 USB 線傳送檔案至 HMI。 請先確認 USB 驅動是否正確安裝至 PC。
IP	輸入下載目標 HMI 的 IP 位址。
名稱	輸入下載目標 HMI 的名稱。
搜尋	搜尋指定的 HMI 名稱。
搜尋全部	搜尋在同一區網上的所有 HMI 名稱。
下載完成後自動啟動 HMI	若勾選，下載程序完成後會自動重新啟動 HMI。
清除配方數據 清除配方資料庫 刪除開機畫面 清除事件記錄 清除資料取樣記錄 清除操作記錄	若勾選，下載程序進行前會先清除 HMI 上所選取的資料。
下載	點選後將開始下載程序。
設定	輸入 HMI 系統設定裡所設定之下載密碼。 



- 以 MT8000 系列為範例，建立完成的下載資料夾格式如下所示，請選擇第一層資料夾名稱作為下載路徑。

第一層資料夾	第二層資料夾	第三層資料夾
MT8000	001	
	002	
	Pub	driver
		font

- 第一層資料夾的名稱將依型號不同而有所變化。