Win-GRAF 使用手冊

泓格科技股份有限公司感謝您購買本公司的 Win-GRAF 系列控制系統，本系列產品結合方便整合的特性以及強大的發展性，適用於各類工業用控制系統，並期望對使用 Win-GRAF 作為開發軟體的系統整合人員、程式設計師以及系統維護人員能有所助益。

本公司出產的 Win-GRAF 可程式自動控制器 (PAC: Programming Automation Controller) 包括:

ViewPAC-2000:	VP-25W8
ViewPAC-4000:	VP-4138
WinPAC-5000:	WP-5148, WP-5238
WinPAC-8000:	WP-8148, WP-8448, WP-8848
	WP-8138, WP-8438, WP-8838
XPAC-8000:	XP-8048, XP-8348, XP-8748
XPAC-8000-CE6:	XP-8048-CE6, XP-8348-CE6, XP-8748-CE6
XPAC-8000-Atom-CE6:	XP-8148-Atom-CE6, XP-8348-Atom-CE6, XP-8748-Atom-CE6

注意事項

泓格科技股份有限公司對於因為使用本系列產品所造成的任何損害並不負任何法律上的責任，本公司並保留在任何時候修訂本書且不需通知的權利。

泓格科技股份有限公司將儘可能地提供本系列產品可靠而詳盡的資訊。然而，本公司並無義務需提供此系列產品詳盡的應用資訊，或對因不當使用本系列產品所遭受的損害負任何責任。

商標 與 著作權

本書所提所有公司商標，商標名稱及產品名稱分別屬於該商標或名稱的擁有者所有。

技術支援

請連絡當地的經銷商或 E-mail 問題至 service@icpdas.com .

目　　錄

第 1 章　軟體安裝 與 硬體設定

1.1　安裝 Win-GRAF 軟體

安裝 Win-GRAF 軟體前，請先確認您電腦中的安裝環境。

系統需求:

作業系統: Windows 7, Windows 8 (32-bit 或 64 bit)

Microsoft .Net Framework 3.5 (可在微軟官方網站下載:
http://www.microsoft.com/zh-tw/download/details.aspx?id=22)

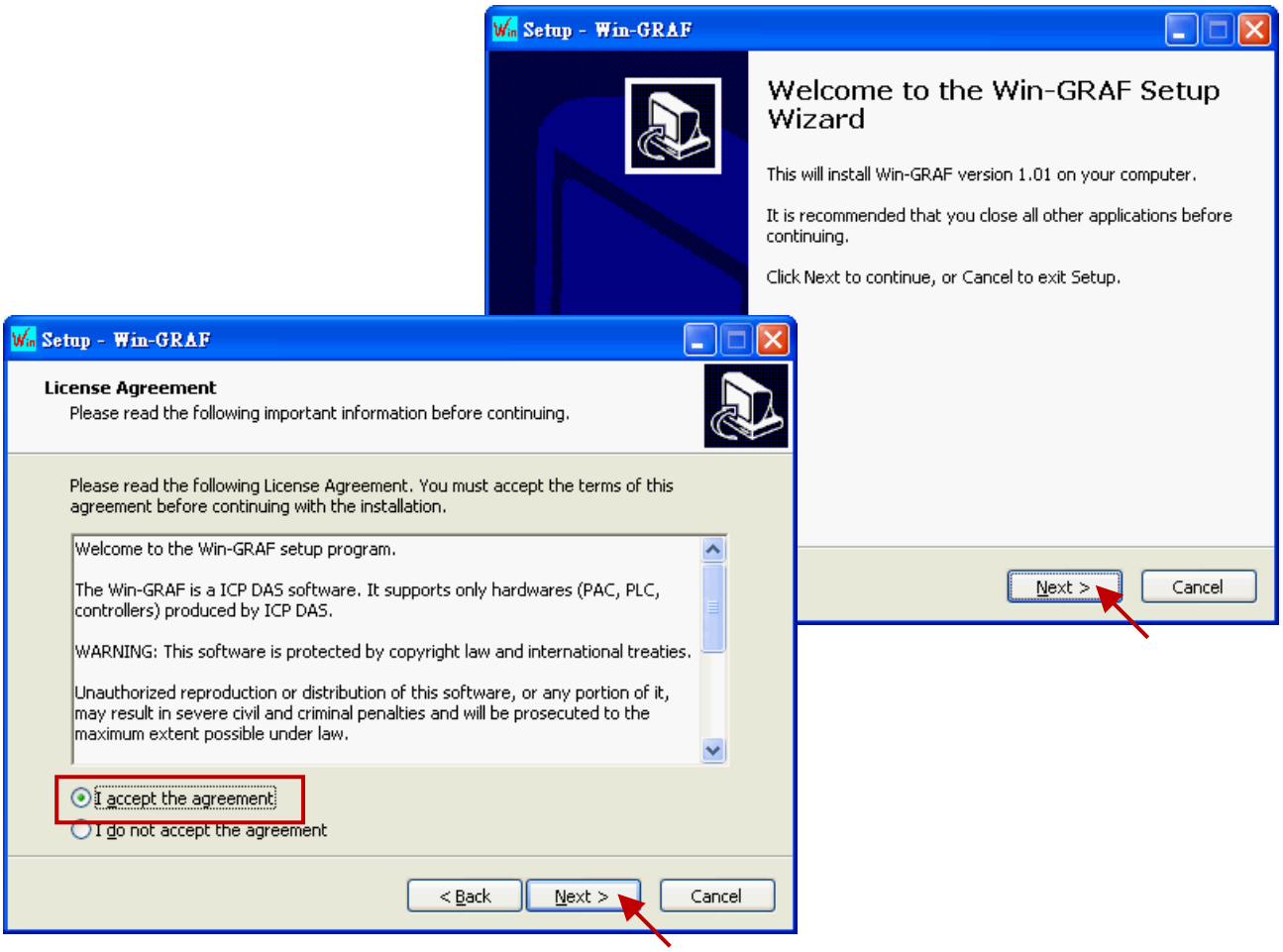
RAM: 至少 1 GB (建議 2 GB)

可用硬碟空間: 至少 200 MB

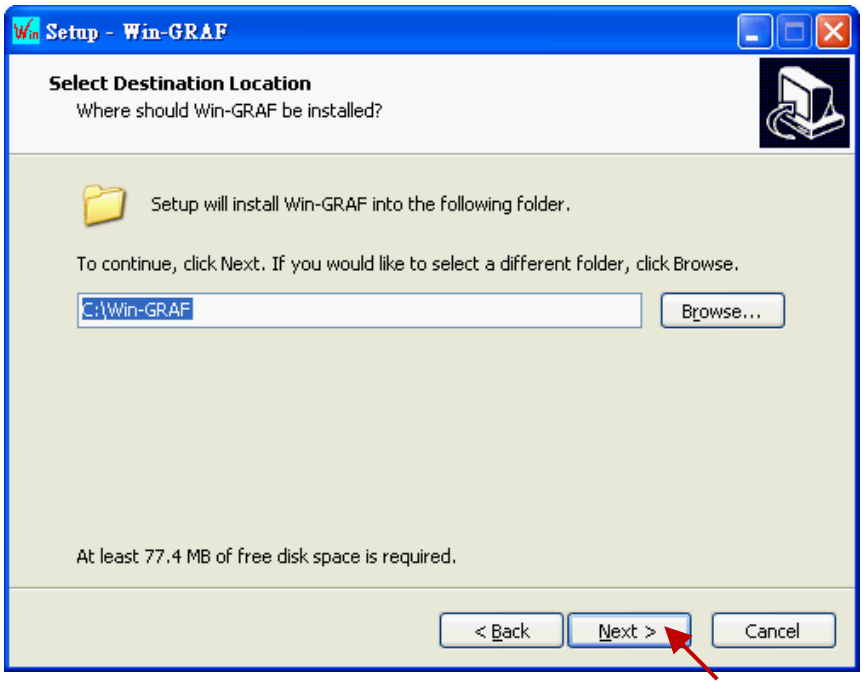
安裝步驟:

1. 於 Win-GRAF 安裝光碟中，滑鼠雙擊 “Win-GRAF-setup-ver-x.xx.exe” 檔開始安裝程序。

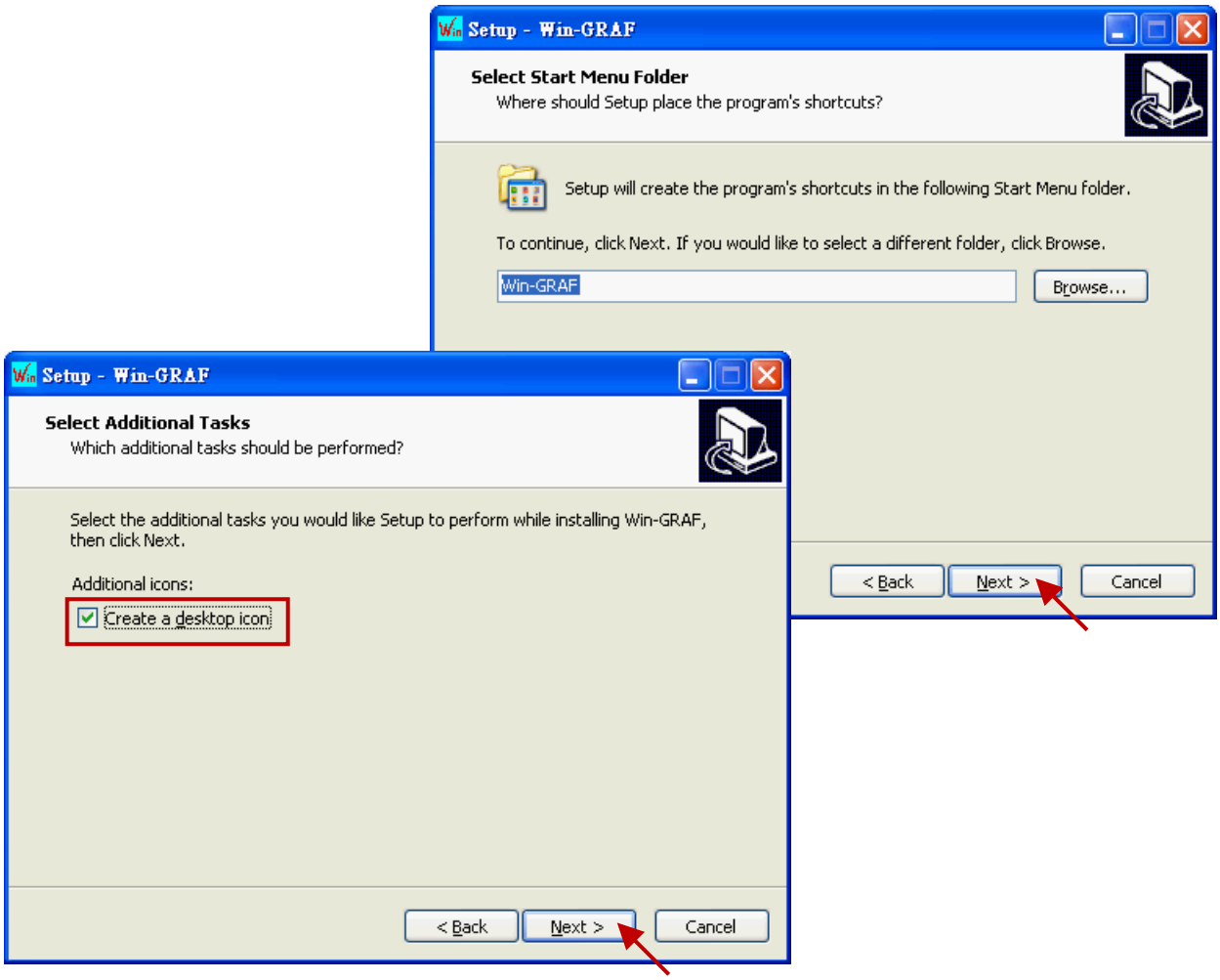
2. 點選 “Next” 進行下一步，接著選取 “I accept the agreement” 再點選 “Next” 繼續。



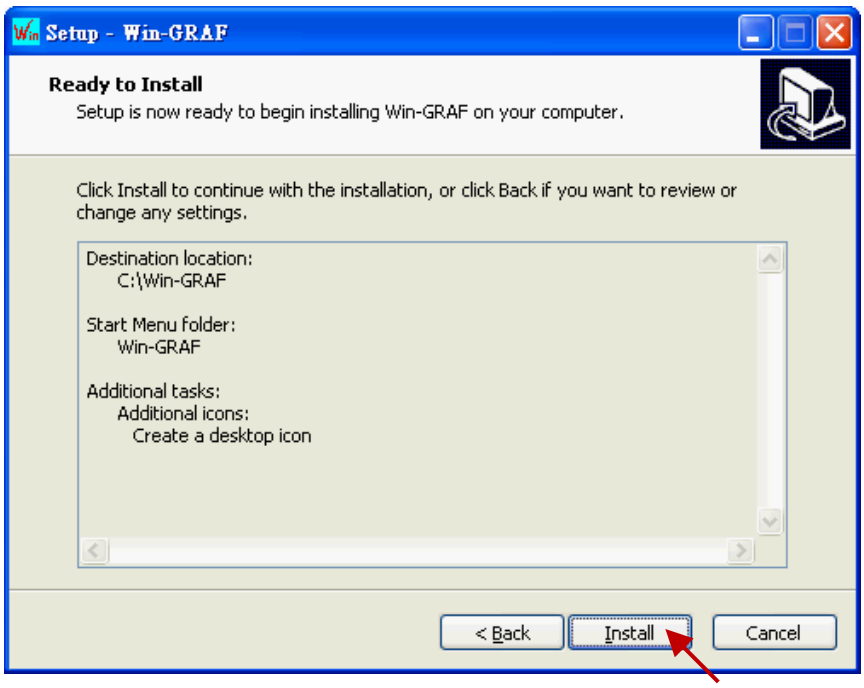
3. 建議使用預設的安裝目錄 (即，“C:\Win-GRAF”)，並點選 “Next” 進行下一步。



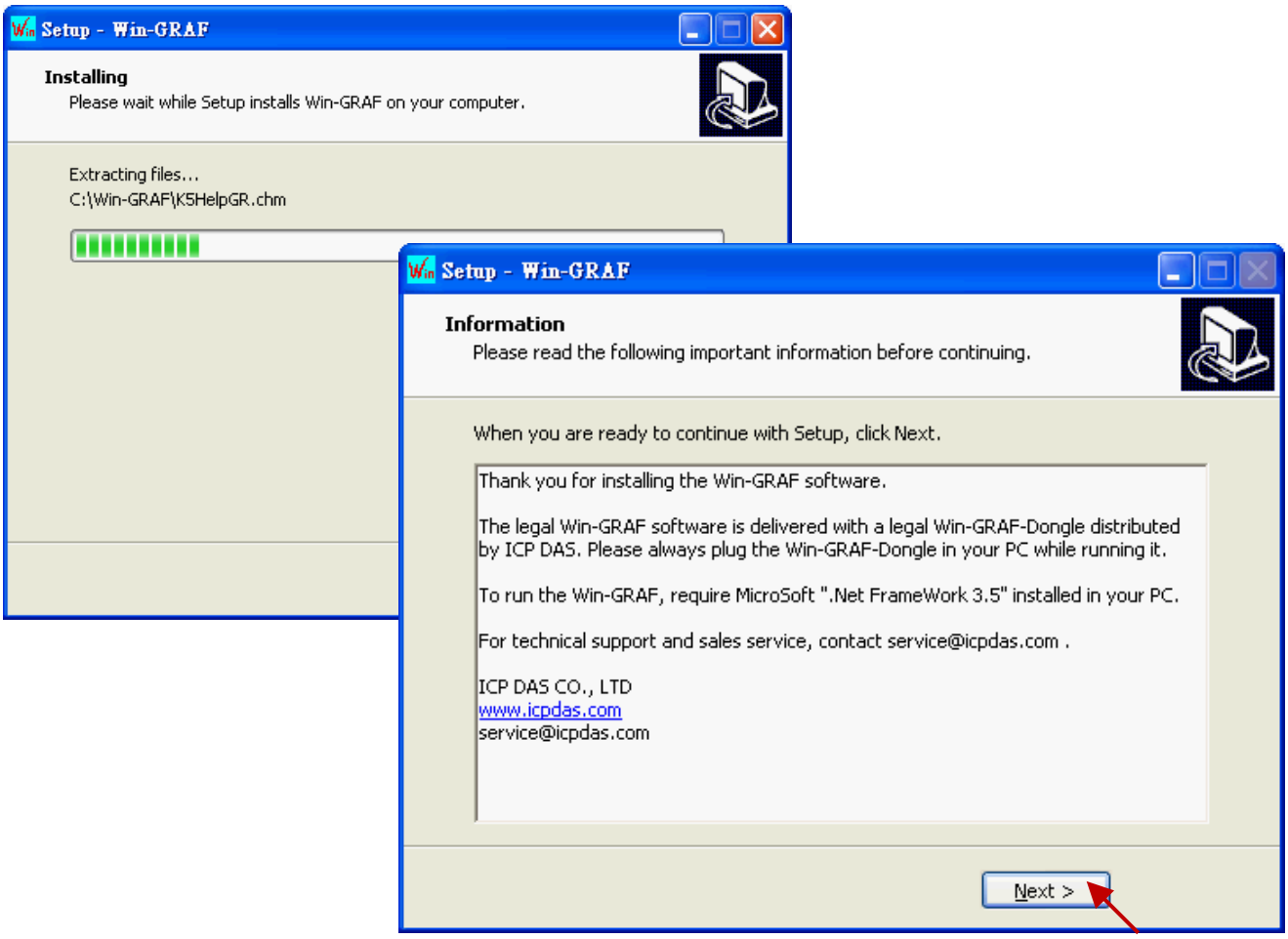
4. 點選 “Next” 於 “開始” 選單中建立 “Win-GRAF” 捷徑目錄，可勾選 “Create a desktop icon” 建立桌面捷徑，再點選 “Next” 進行下一步。



5. 點選 “Install” 開始安裝 Win-GRAF 軟體。

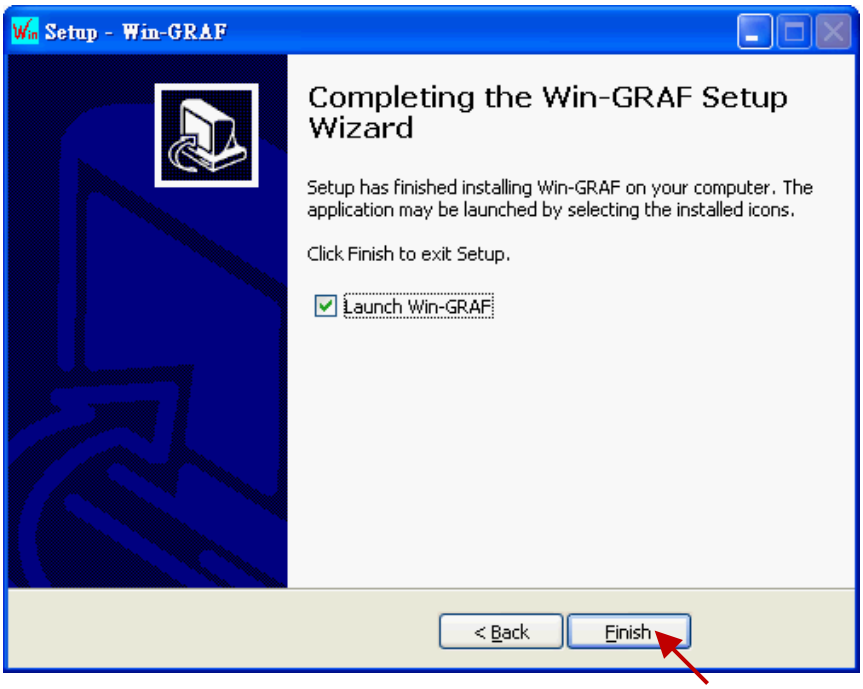


6. 等待安裝完成，您會見到以下訊息視窗，此處說明了:
   a. 泓格科技合法授權的 Win-GRAF 軟體配備有一個 USB 保護鎖 (Win-GRAF Dongle)，開啟該軟體前，請確認 USB 保護鎖已裝置在您的電腦中。
   b. 您的電腦必須安裝 “.NET FrameWork 3.5” 才能運行 Win-GRAF 軟體。



7. 您已完成 Win-GRAF 安裝，點選 “Finish” 離開此視窗。

   (勾選 “Launch Win-GRAF” 將會自動開啟此軟體，請確認您的 PC 已裝置了 Win-GRAF Dongle。)



1.2 開啟 Win-GRAF 軟體

開啟軟體前，請確認您的 PC 已裝置了 USB 保護鎖 (Win-GRAF Dongle)。

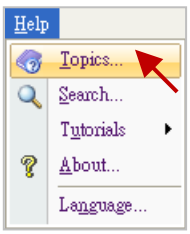
您可在 “開始” 選單中，點選 “Win-GRAF”目錄再點選 “Win-GRAF” 來開啟軟體。

Win-GRAF 目錄說明:

Libraries: 可供使用者建立自訂的函式 或 修改內建的函式。

Handbook: 原廠提供關於軟體介面、編程環境、編程語言…等詳細說明。

(或 Win-GRAF 的功能表 [Help] > [Topics])



History: 記錄 Win-GRAF 軟體更新內容的文字檔。

Manual: ICP DAS 提供的 Win-GRAF 相關手冊。

(或 Win-GRAF 的功能表 [Help] > [Tutorials]，存放於 C:\Win-GRAF\Tutorials)

ReadMe: 關於 Win-GRAF 版權相關資訊。

1.2.1 軟體操作模式

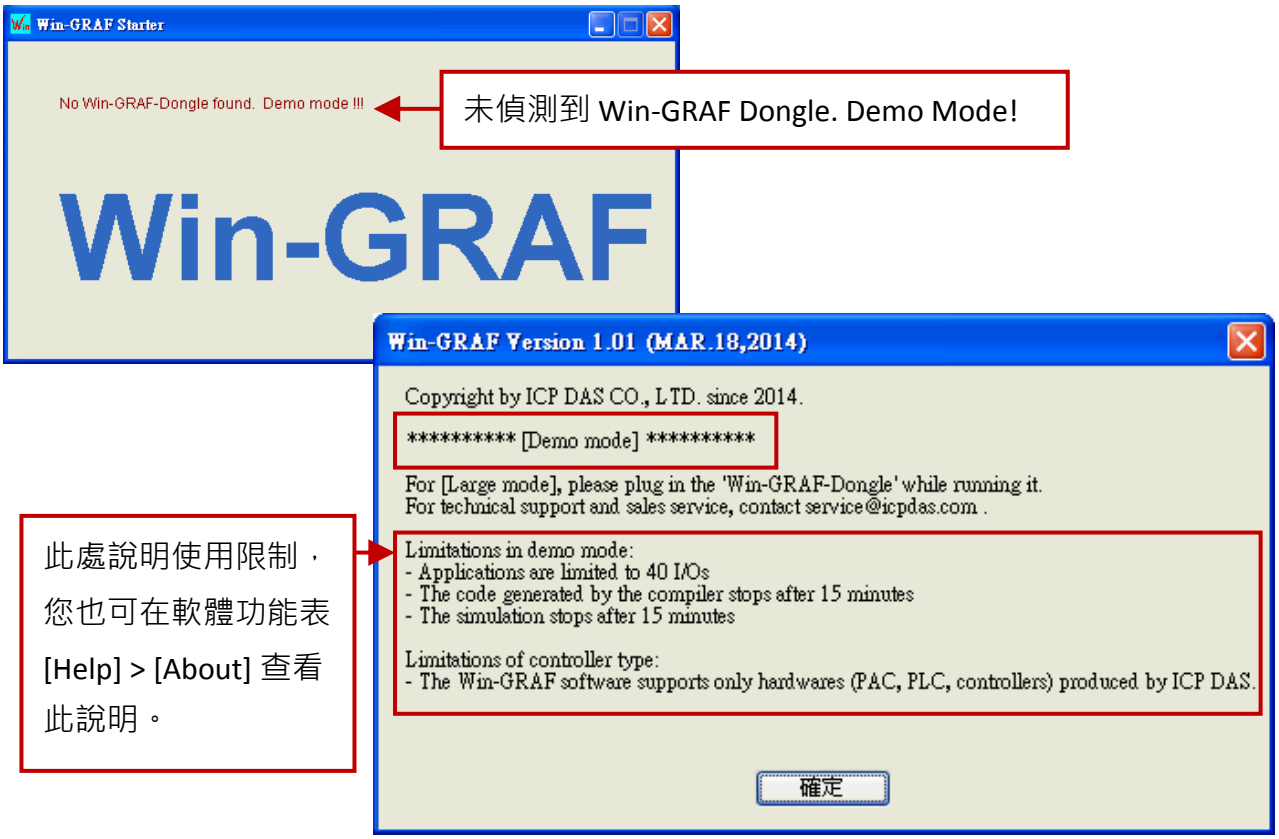
Win-GRAF 提供了兩種操作模式:

"Demo Mode": PC 未裝置 USB 保護鎖，表示 Win-GRAF 編譯出來的專案 (Project) 只能在 PAC 內運行 15 分鐘，時間超過後需再重新下載專案到 PAC 中，且只能使用 40 個 I/O 點。

"Large Mode": PC 有裝置 USB 保護鎖，表示 PAC 內的專案可一直 Run 沒有時間限制。

Demo Mode

若您的 PC 未裝置 Win-GRAF Dongle，則軟體開啟時會先顯示以下畫面:



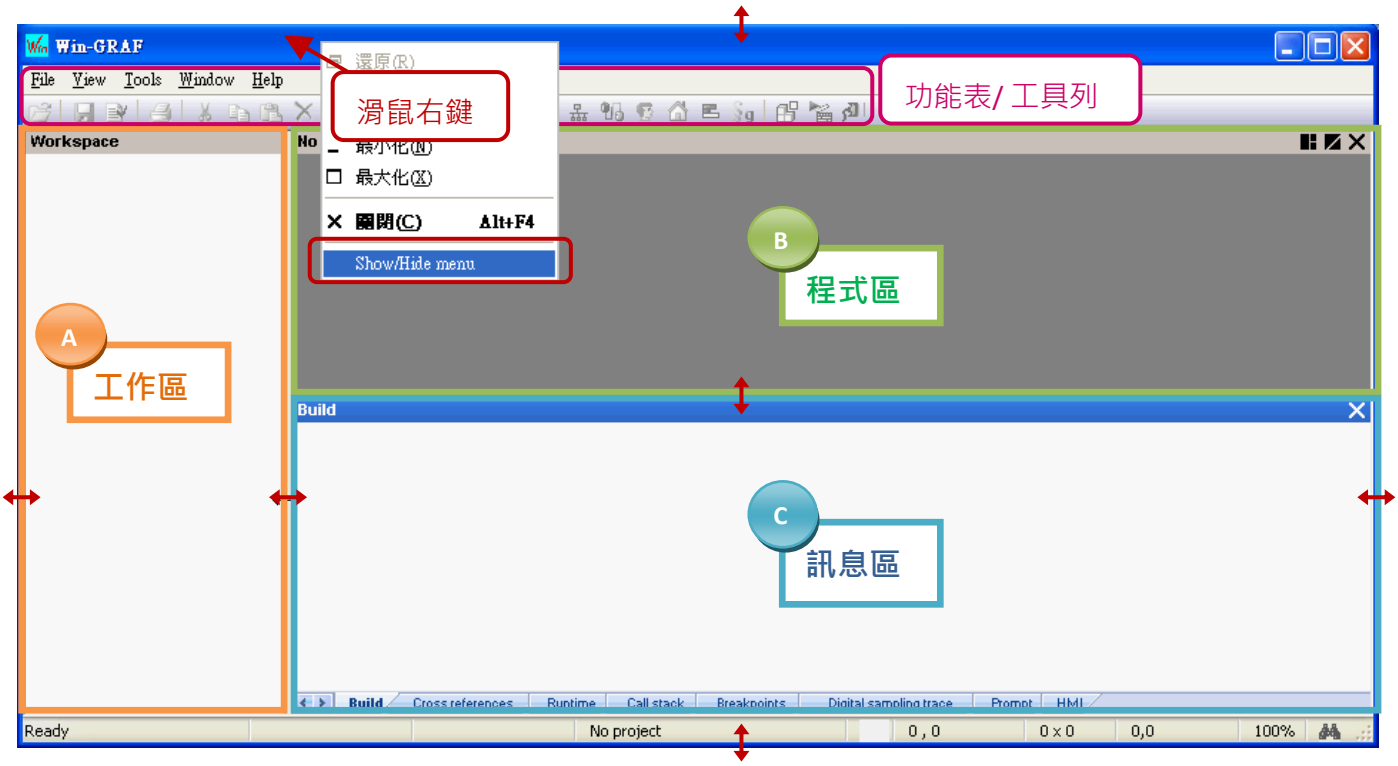
註: 若於 [Demo Mode] 下裝置 Win-GRAF Dongle，需重新開啟 Win-GRAF 軟體來進入 [Large Mode]。

Large Mode

若您的 PC 已裝置 Win-GRAF Dongle，則軟體開啟時會先顯示以下畫面:

1.2.2 軟體介面說明

開啟 Win-GRAF 後，畫面如下:



註: 滑鼠右鍵點選視窗最上方可顯示/隱藏功能表。

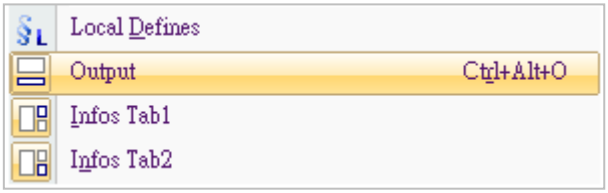
A. 工作區: 用來建立/開啟 Win-GRAF 程式的清單。
B. 程式區: 用來顯示/編輯程式，可再分割為多個功能區塊 (參考 2.2.1 節)。
C. 訊息區: 用來顯示編譯訊息，並提供多個診斷工具。

使用小技巧:

1. 可使用滑鼠拖曳的方式，任意調整視窗的大小。
2. 可在工作區上按 “F1” 鍵，開啟使用說明。

回復功能視窗

若操作中不慎關閉了程式區的變數視窗 或是 訊息區，可點選 **功能表 "View"** 再選取下列選項來回復視窗。



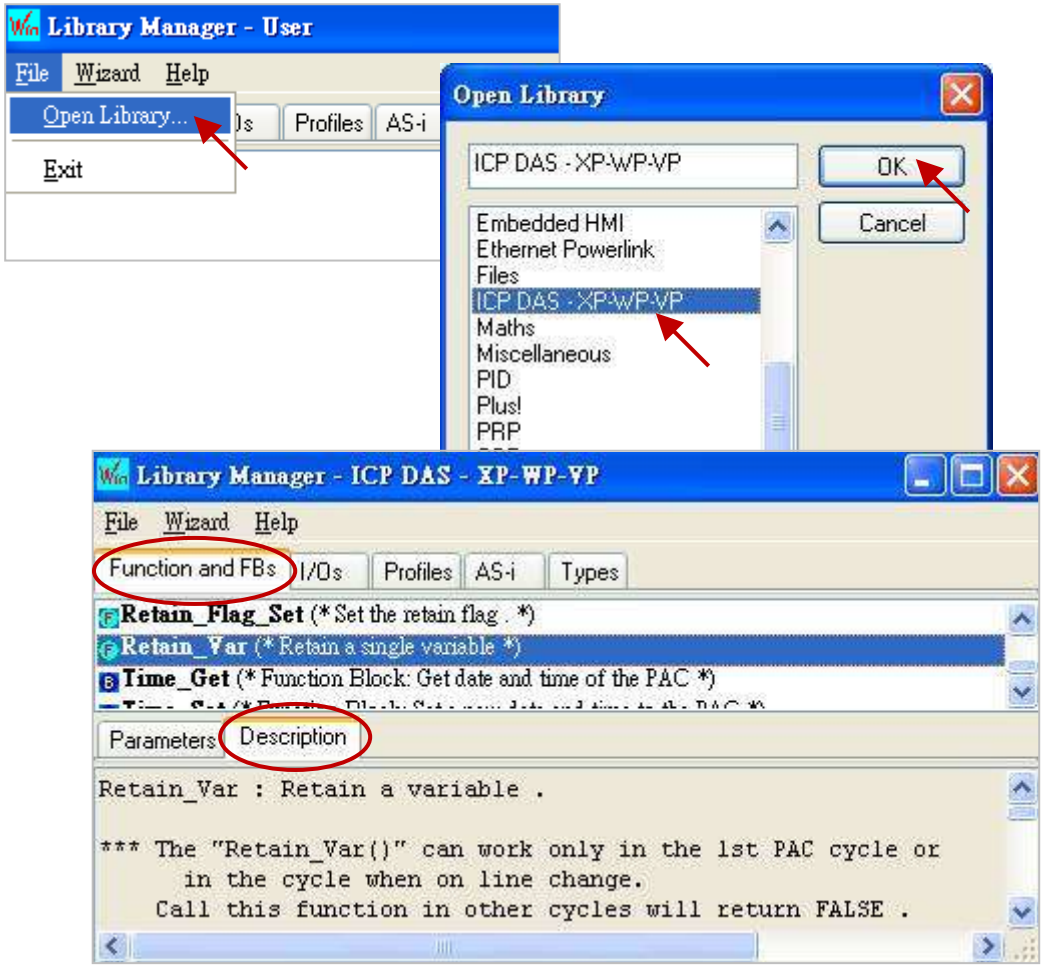
Output: 表示訊息區的視窗。
Infos Tab1: 表示程式區 - 變數視窗 (參考 2.2.1 節)。
Infos Tab2: 表示程式區 – 功能方塊視窗 (參考 2.2.1 節)。

1.2.3 程式庫管理員 (Library Manager) 功能

Win-GRAF Workbench 提供了程式庫管理員 (Library Manager) 功能，可用來查詢各函式 (Function)、功能方塊 (Function Block) 與 I/O 卡 (I/O Board) 的功能說明。

1. 於 PC 的 “開始” 選單中，點選 “Win-GRAF” 目錄再點選 "Libraries" > “OEM”。

2. 於 “Library Manager” 視窗中，點選功能表 “File” > “Open Library” 並選擇 “ICP DAS – XP-WP-VP” 再點選 “OK”。
3. 可在 “Function and FBs” 頁籤，點選下方的 “Description” 來查看各函式 (Function) 或 功能方塊 (Function Block) 的說明 ; 點選 “I/Os” 頁籤再點選下方的 “Description” 則可查看 I/O 卡 (I/O Board) 的功能說明。

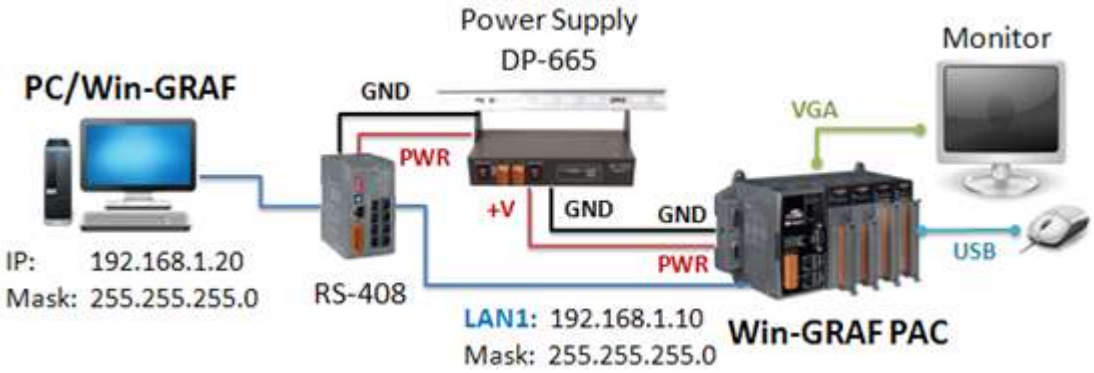


1.3 設定 Win-GRAF PAC 的 IP 位址

Win-GRAF Workbench 需知道 PAC 的 IP 位址才能與之連線，以下將說明 PAC 端的 IP 設定。

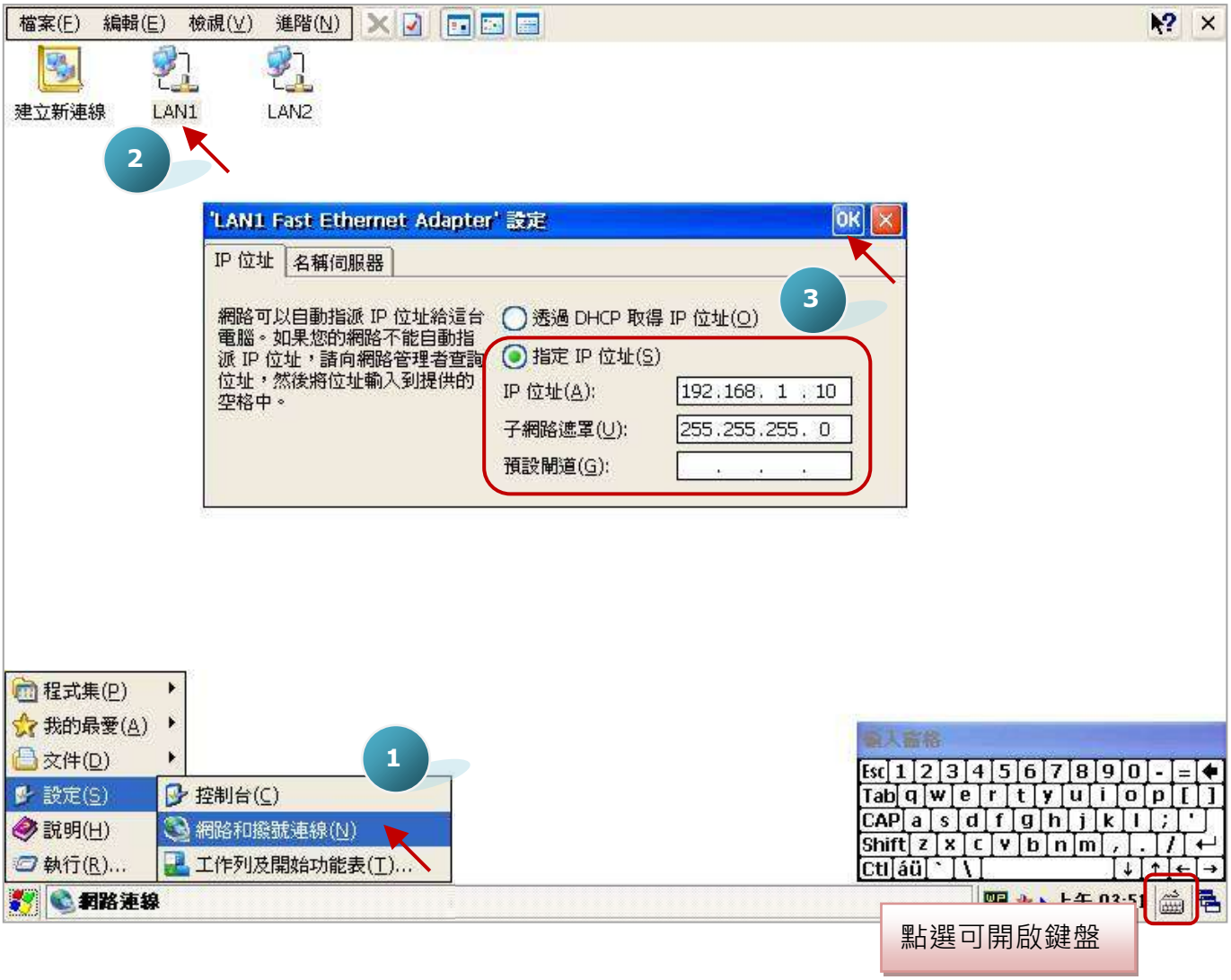
以 XPAC (XP-8xx8, XP-8xx8-CE6, XP-8xx8-Atom-CE6) 與 WinPAC (WP-8xx8, WP-5xx8) 為例:

硬體連接圖



PAC 端

使用連接在 PAC 上的 USB Mouse 點選螢幕左下角的 “開始” > “設定” > “網路和撥號連線”，再雙擊 LAN1 (或 LAN2)，並輸入適當的 IP 位址。

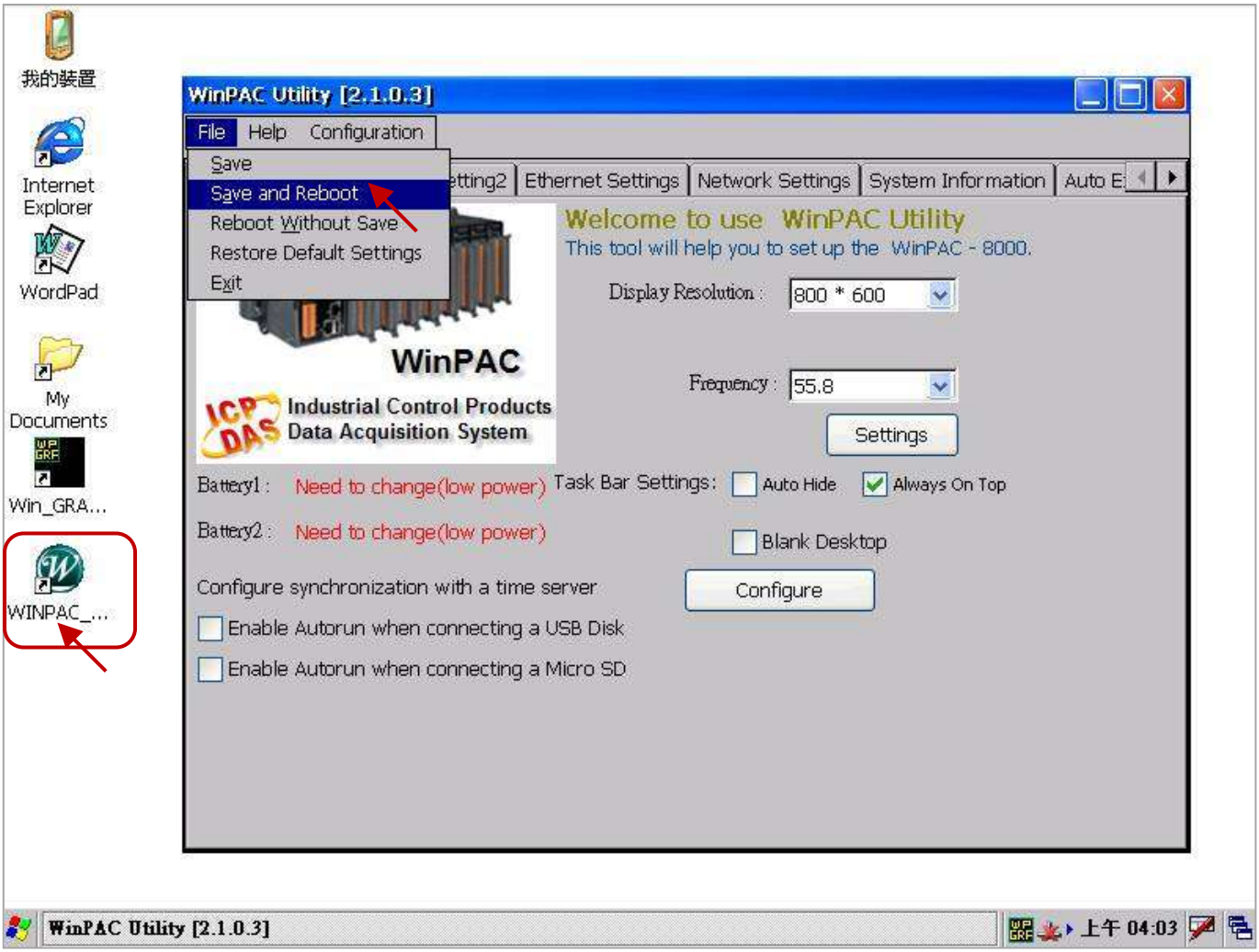


於 PAC 桌面上 (或 \System Disk\tools\ 路徑下)，開啟 “WinPAC_Utility.exe” (或 “XPAC_Utility.exe”)，再點選 “File” > “Save and Reboot” 重新啟動 PAC。

注意:

PC/Win-GRAF 的 IP 與 PAC IP 需在同一個 IP 網段內，才連的上。

例如: PC 的 IP 設定為 192.168.1.20 (Mask: 255.255.255.0)。



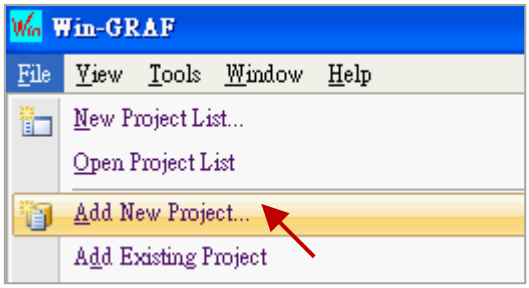
第 2 章　編寫一個簡單的 Win-GRAF 範例

本章將介紹一個簡單的樣版專案，可用來讀取或寫入 Win-GRAF PAC 中的系統時間。請依照以下步驟，一步步地實現此範例。

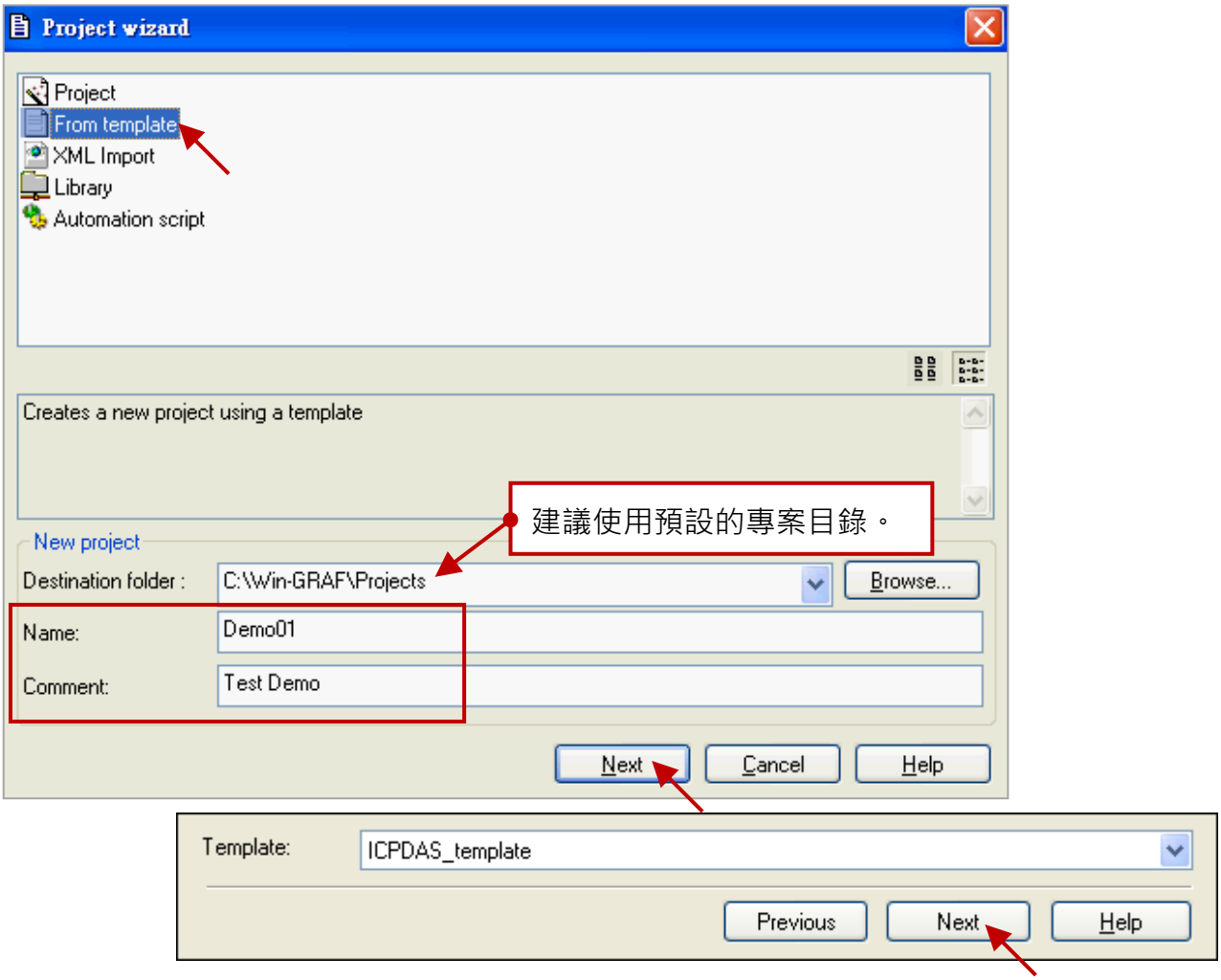
2.1　建立 Win-GRAF 專案

2.1.1　建立樣版專案 (Demo01)

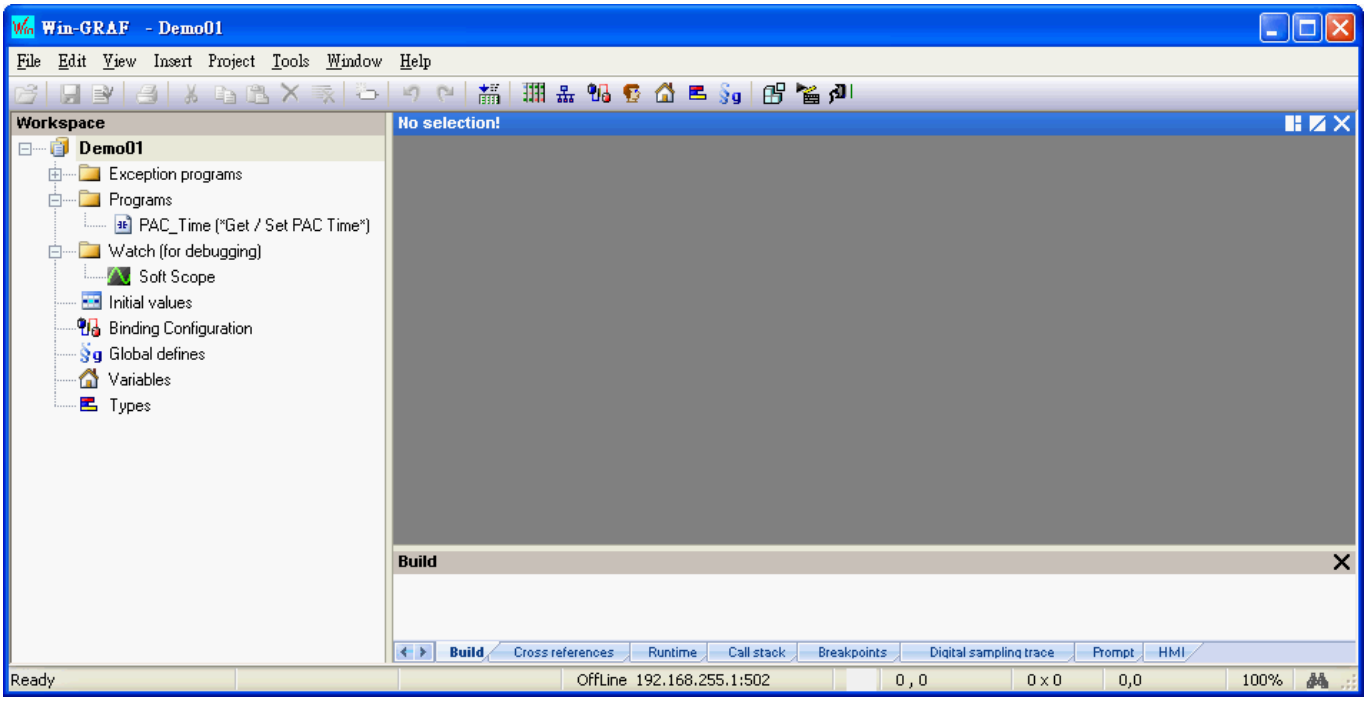
1. 開啟 Win-GRAF 軟體後 (可參考 1.2 節)，點選 "File" 功能表再選擇 "Add New Project..." 選項。



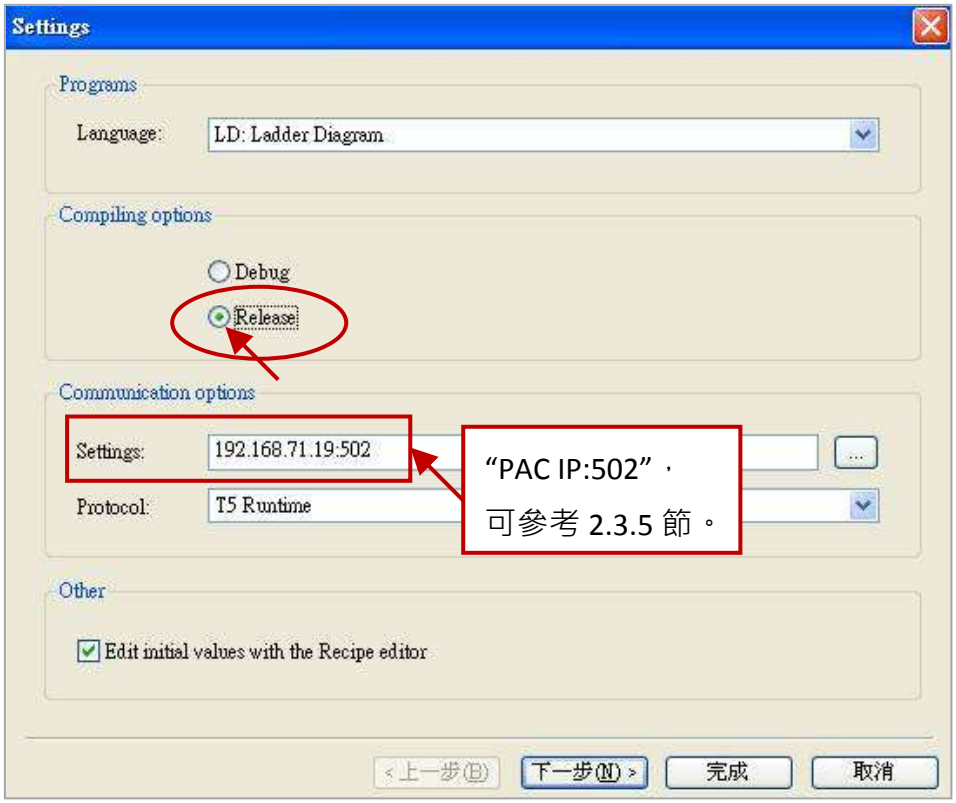
2. 點選 “From template” 來建立一個套用樣版的專案，於 "Name" 欄位中輸入專案名稱 (例如: Demo01)，並在 “Comment” 欄位中輸入此專案的註解，點選 "Next" 後，會出現 Win-GRAF 提供的預設樣版 "ICPDAS_template"，請直接點選 "Next" 進行下一步。



3. 您已建立一個 “Demo01” 樣版專案。



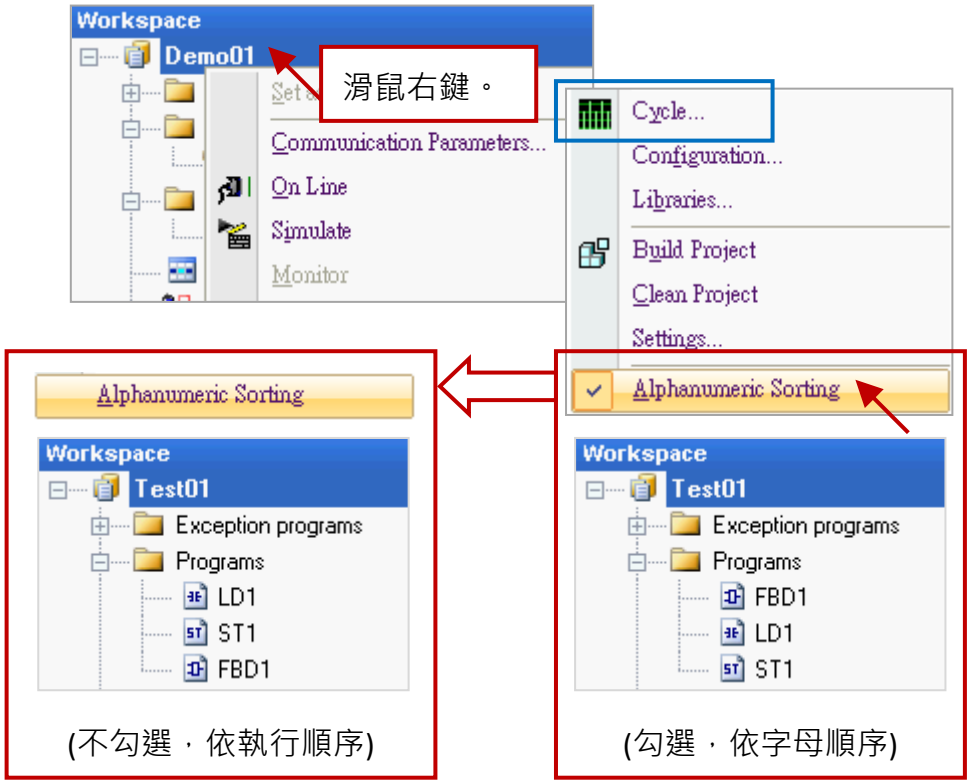
注意: 本範例選用 “From template” 方式來建立新專案，若您在步驟 2 選擇了 "Project" 方式，“Compiling options“ 請選擇 "Release"，其餘項目可在後續章節中進行設定，請點選 "下一步" 再點選 "完成"。



2.1.2 重要專案設定

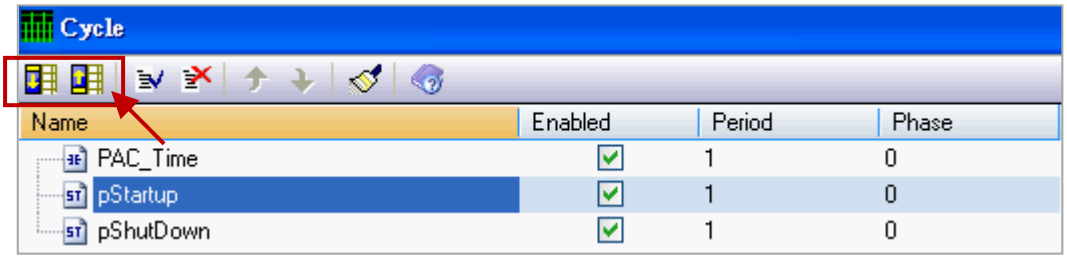
建立專案後，有兩個需先完成的重要設定。

1. 在 "工作區" 中，滑鼠右鍵點選專案名稱 (例如: Demo01)，並取消勾選 "Alphanumeric Sorting" (即最後一個選項)。若不勾選，表示依執行順序來排列程式。若勾選，表示依字母順序來排列程式 (例如: FBD1, LD1, ST1)。

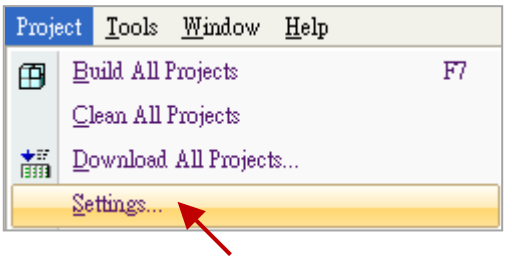


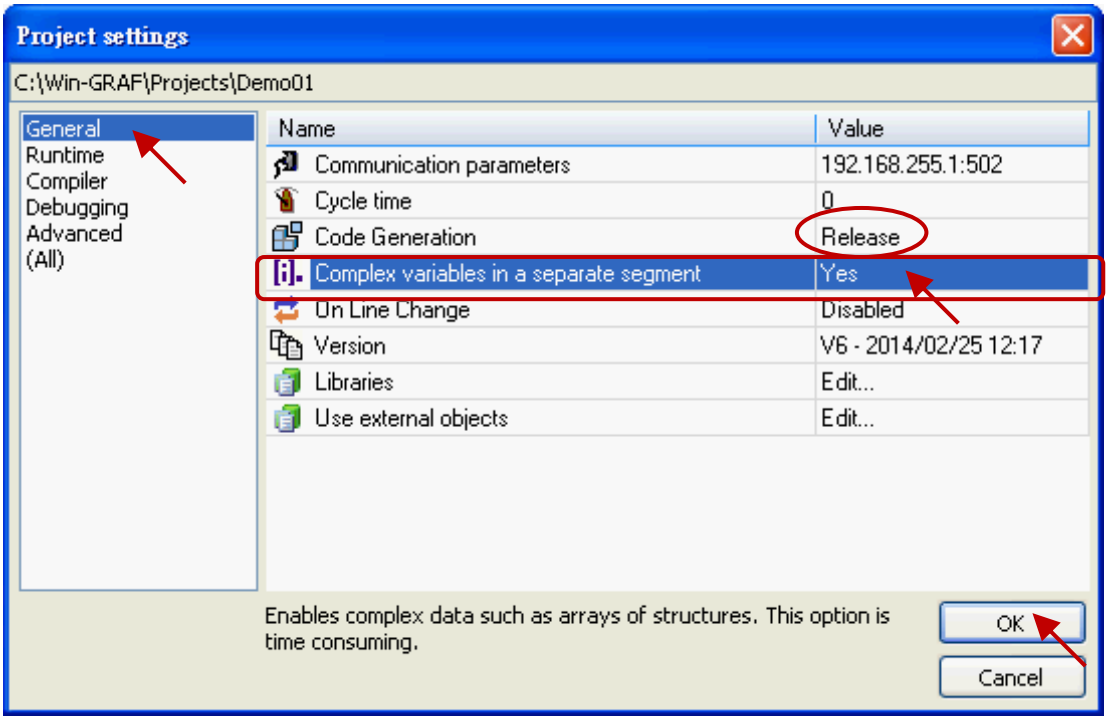
修改程式的執行順序:

注意: 若您需修改程式的執行順序，(如上圖) 滑鼠右鍵點選專案名稱 (例如: Demo01)，再點選 "Cycle" 選項來開啟設定視窗，再點選 "Move Up" 或 "Move Down" 按鈕來挪動順序。



2. 若選用 "Project" 方式 (見 2.1.1 節 - 步驟 2，本例選用 “From template” 方式) 來建立新專案，請點選功能表 "Project" > "Settings..." 開啟專案設定視窗 (如下圖)，再點選 "General" 選項，並將 "Complex variables in a separate segment" 設定為 "Yes"，以允許使用像是陣列的資料型態，再點選 "OK" 離開設定視窗。



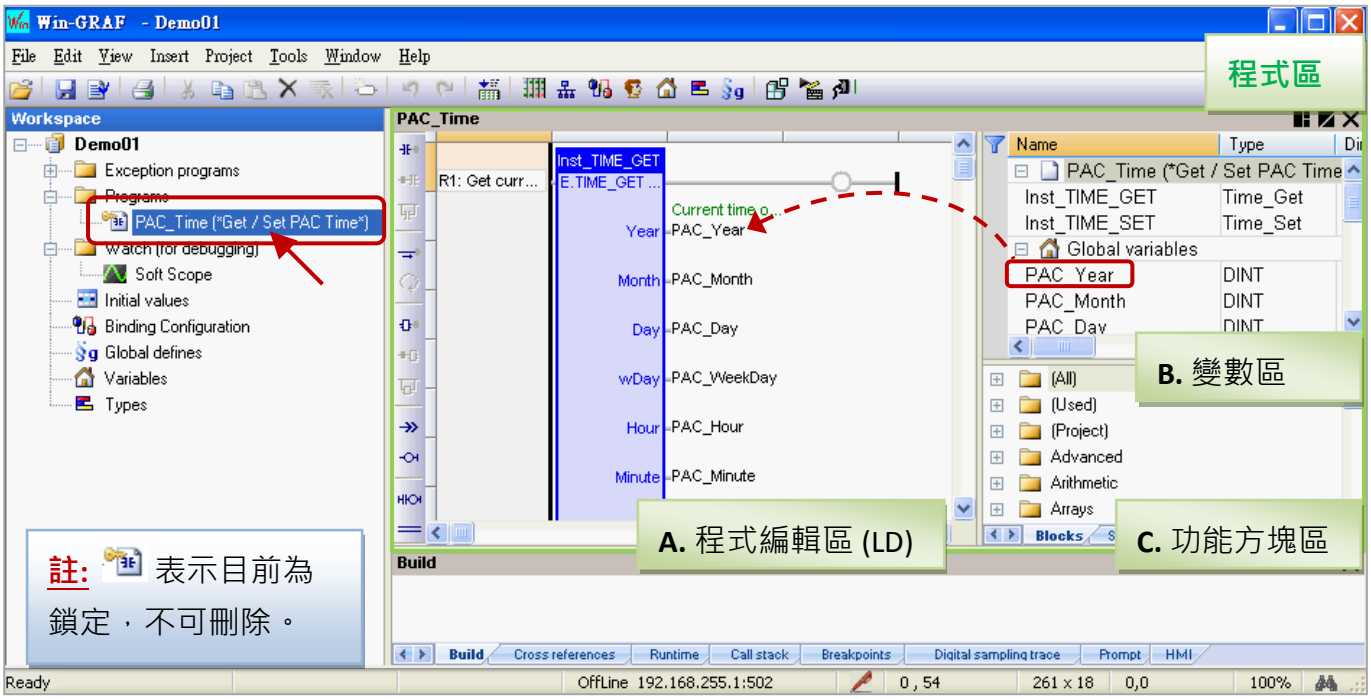


注意: "Code Generation" 需設定在 "Release"。

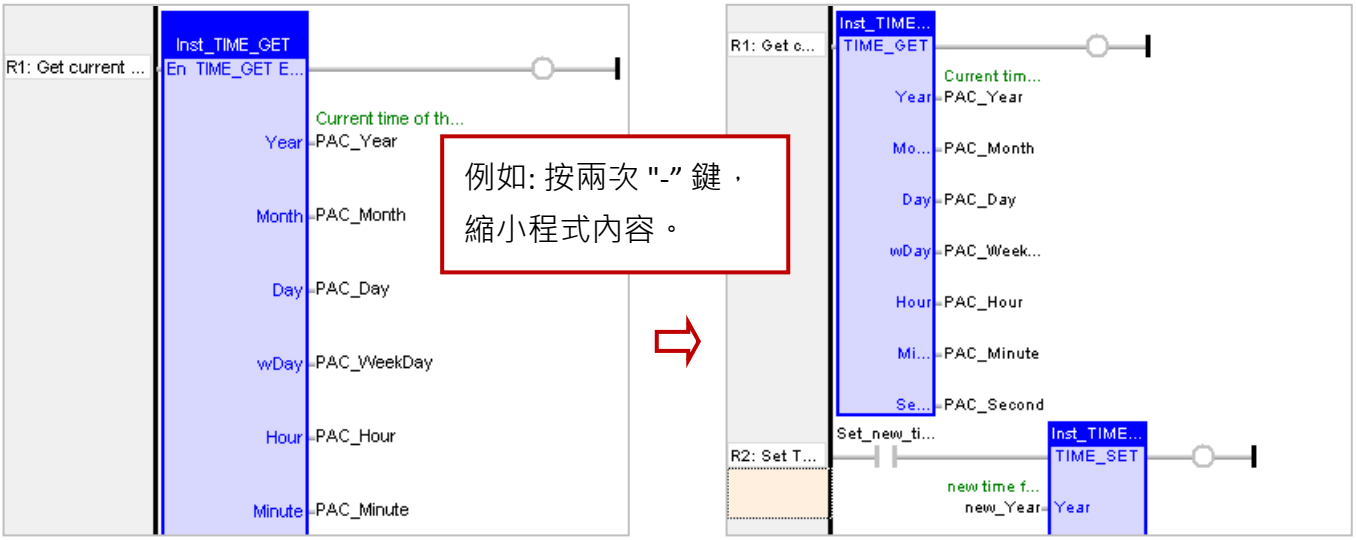
2.2 專案介紹

2.2.1 Demo01 - LD 程式

此程式用來讀取或寫入 Win-GRAF PAC 中的系統時間。於工作區中，滑鼠雙擊 LD 程式名稱 (即，“PAC_Time") 來開啟相關的視窗。如畫面中，程式區 劃分為 3 個區塊:



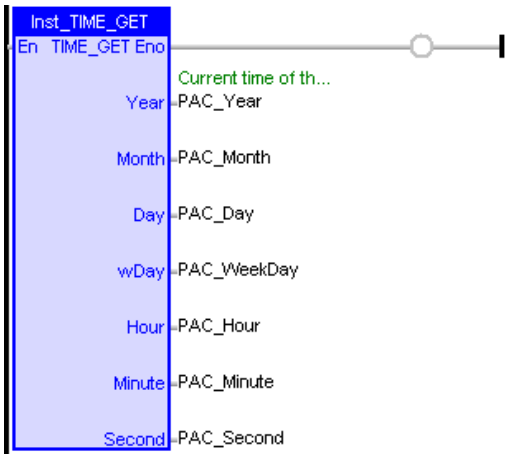
使用小技巧: 滑鼠點選一下程式編輯區，再按 "+" 或 “-” 鍵，可放大或縮小程式內容。



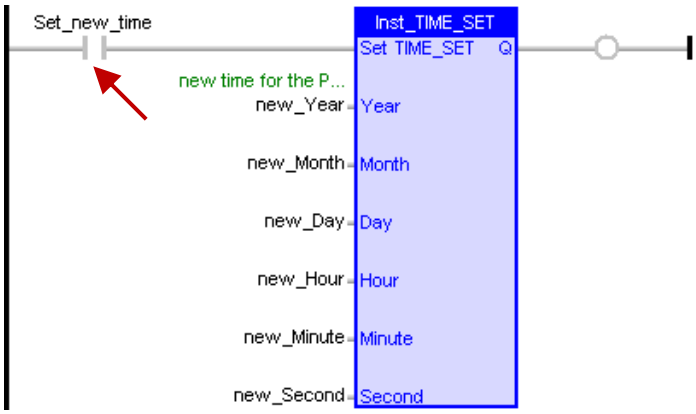
A. 程式編輯區 (LD):

此區塊用來編輯/顯示 LD 程式，您可點選左邊的元件按鈕來新增程式，也可用滑鼠拖曳方式將定義好的變數指定給功能方塊。

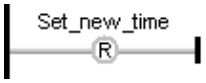
R1: 讀取 PAC 目前時間



R2: 將 “Set_new_time” 設為 “True”，可寫入新的時間

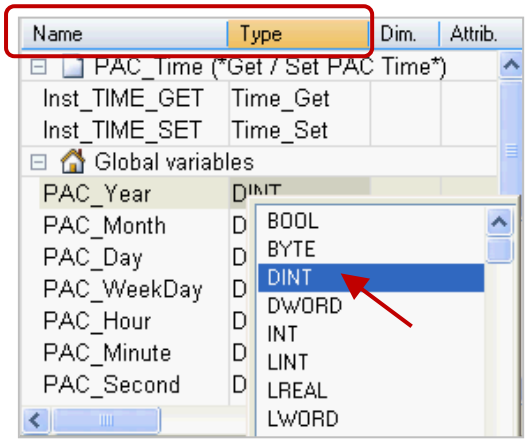


R3: 重置為 “False”狀態

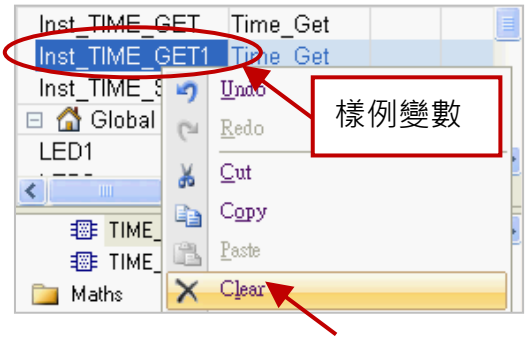


B. 變數區:

此區塊顯示了程式中使用的 功能方塊 與 變數。滑鼠雙擊任一 “Name” 或 “Type” 項目，可修改其名稱 或 資料型態，再按 “Enter” 鍵完成修改。關於變數的定義方式，請參考 2.3.1 節。

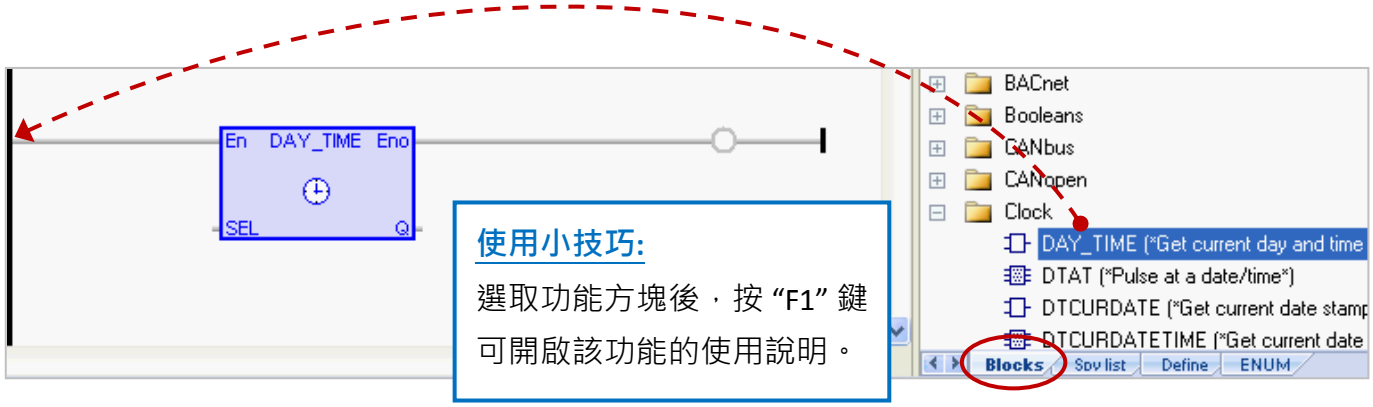


註: 程式內每增加一個功能方塊都會自動建立一個 “Inst_xxx..” 樣例變數，這樣才能正確使用功能方塊。但若刪除程式的功能方塊時，為了安全並不會自動刪除變數區的樣例變數，因此使用者可用滑鼠右鍵點選名稱再選擇 “Clear” 來刪除不需使用的樣例變數。



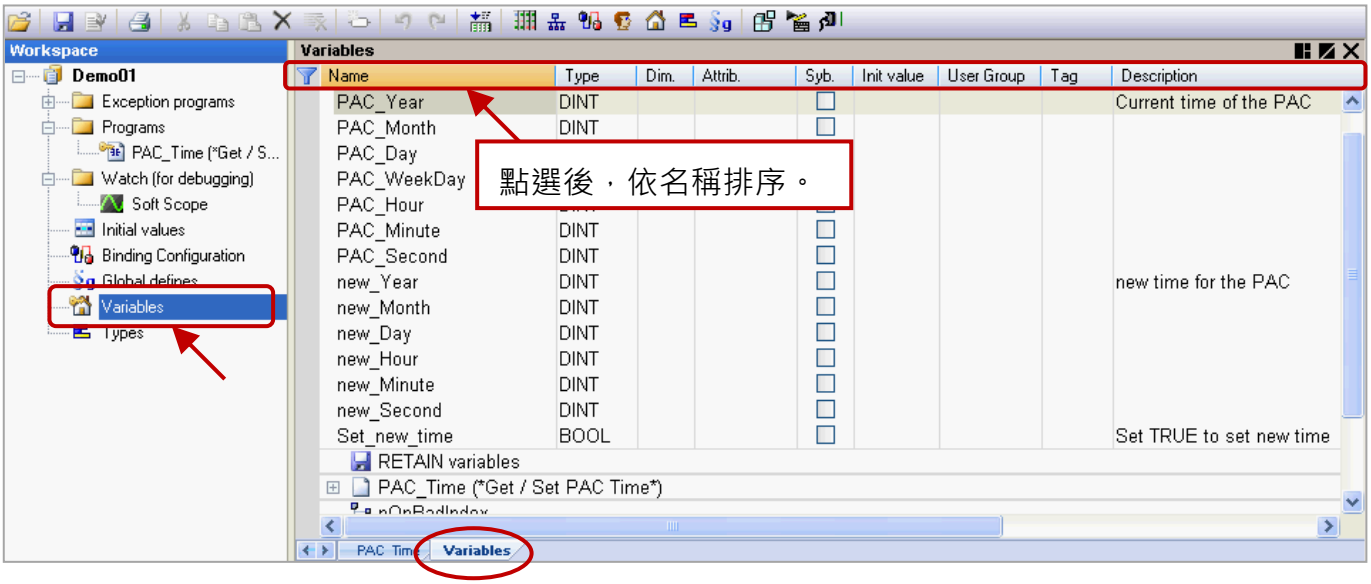
C. 功能方塊區:

此處 (“Blocks” 頁籤) 提供了許多類型的功能方塊，您可依需求將功能方塊拖曳到程式的編輯區。

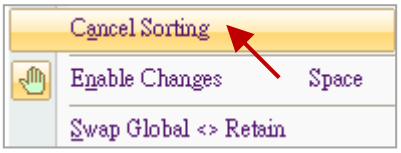


2.2.2 Demo01 - 變數說明

工作區中，滑鼠雙擊 “Variables” 項目來開啟變數視窗，此處列出了所有 “Demo01” 專案中需使用、已定義好的變數。



使用小技巧: 在變數視窗中，可點選任一標題欄位 (例如: “Name”) 來進行排序。
若要恢復原有排序，以滑鼠右鍵點選視窗內任一處並選擇 “Cancel Sorting”。



欄位說明: (可按 “F1” 鍵，查看詳細說明)

可用滑鼠雙擊任一個欄位項目，來進行設定/修改。

Name:	變數名稱，無大小寫的區別，都視為相同可使用的字元。 名稱只能是 A ~ Z (或 a ~ z)、0 ~ 9 與 “_” (底線字元)，而且第一個字必需是英文字元。
Type:	資料型態。(數值範圍，請參考 附錄 A)
Dim.:	可供陣列使用 (例如: 輸入 “10” 來表示 Counter [0] ~ Counter [9])。
Attrib.:	若滑鼠雙擊此欄位，會設定為 “Read Olny” 表示該變數只能讀取，不可變更值。
Syb.:	若勾選，會將變數名稱也下載至 PAC 中。
Init value:	可輸入變數的初始值。
User Group:	可將變數分成群組 (例如: 群組 1, 2)，以方便查看、搜尋變數。
Tag:	可輸入易於辨識的名稱 (Nick name)。
Description:	可輸入變數的用途說明。

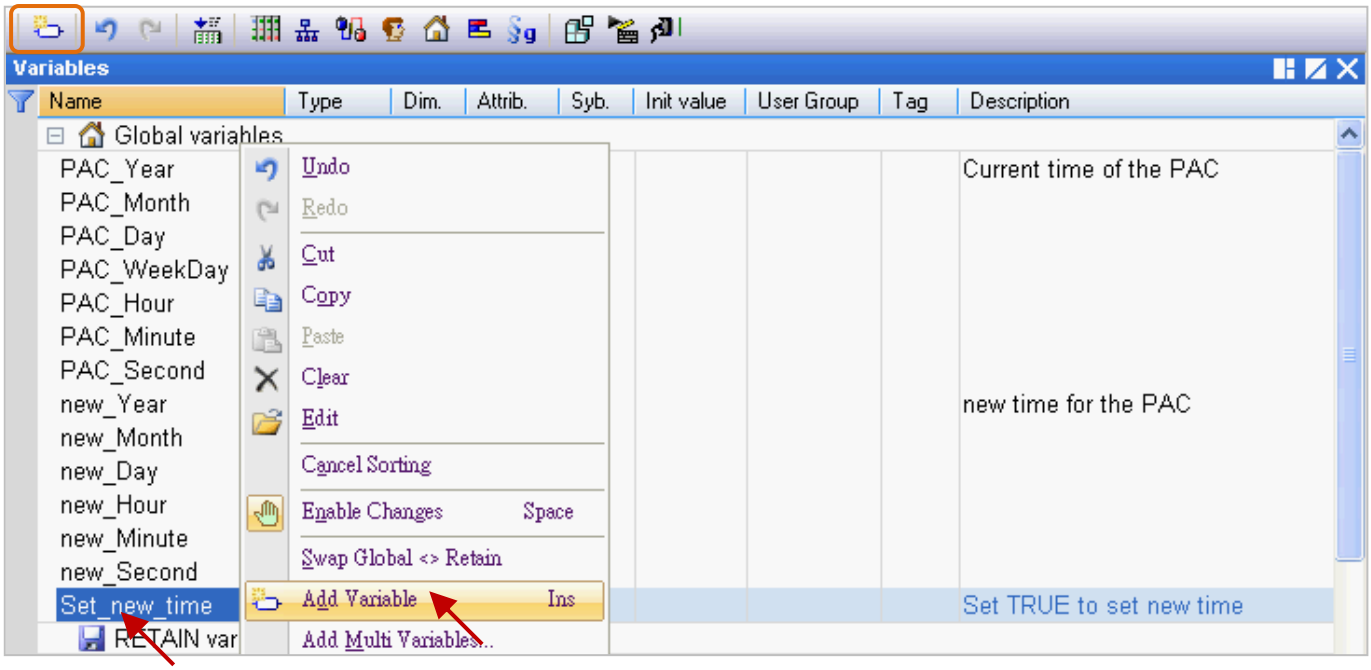
2.3 小試身手

在前述章節中，分別介紹了 “Demo01” 專案中的 LD 程式 (2.2.1 節) 與 變數說明 (2.2.2 節)，本章節將在此專案中練習如何宣告變數與新增一個提供閃爍功能的 LD 程式。

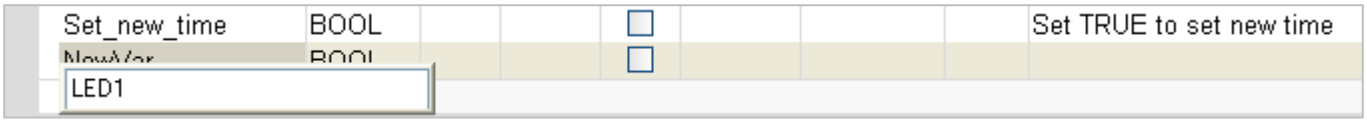
2.3.1 宣告專案變數

首先，我們將宣告 2 個布林變數 (即，LED1 與 LED2) 可供程式使用。

1. 在 “Variables” 視窗中，滑鼠右鍵點選任一個 “Name” 項目，再選擇 “Add Variable” 或 按鍵盤 "Ins" 或 點選工具按鈕  來新增一個變數。



2. 滑鼠雙擊新增的 “NewVar” 項目將名稱修改為 “LED1”，再按 “Enter” 完成設定。
   此例，資料型態為 "BOOL"。



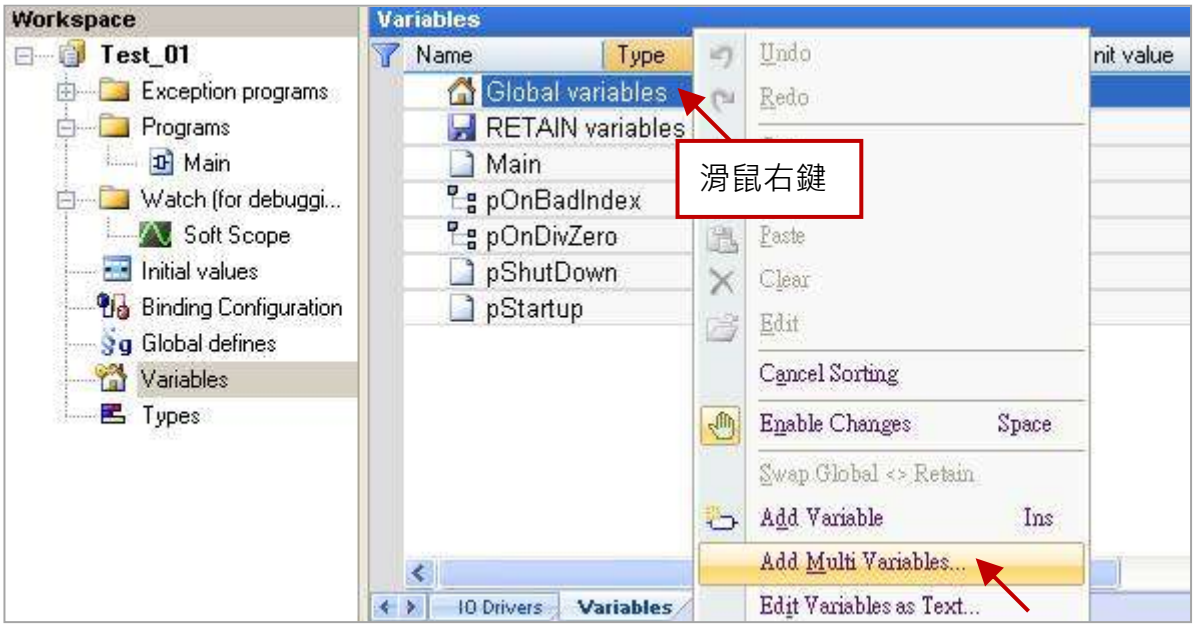
   注意: 所有的修改都必須按 “Enter” 才能完成設定。

3. 如同上述步驟，請再新增 “LED2” 布林變數。

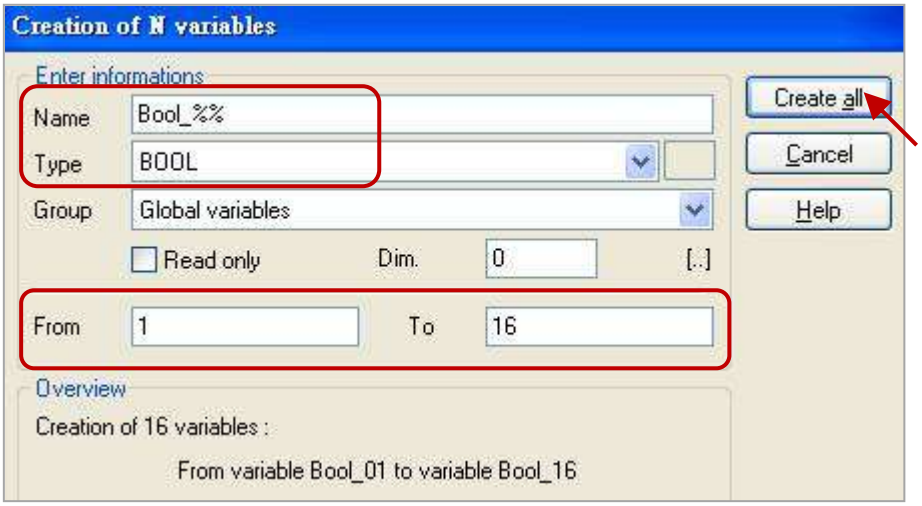
使用小技巧: 對於有連續順序的變數，在步驟 2 可改以輸入 “LED” 再以複製 (Ctrl+C) 並貼上 (Ctrl+V) 的方式 自動產生編號 (即，“LED1”、“LED2”)，最後再刪除 “LED” 即可。

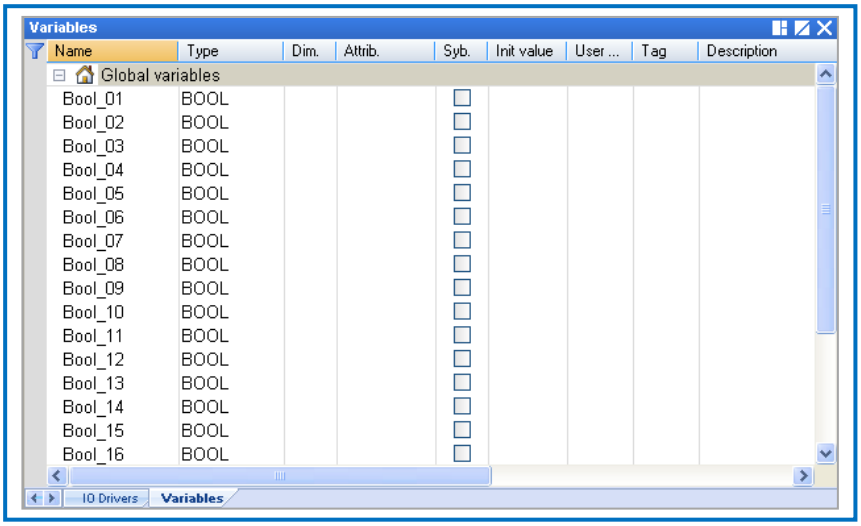
使用小技巧 2:

1. 若需新增多個變數時 (例如: “Boo_01 ~ Boo_16”)，滑鼠右鍵點選 “Global variables” 再選擇 “Add Multi Variables”。



2. 如圖，輸入以下設定值 (Name: “Bool_%%”；Type: “BOOL”；From: 1；To: 16) 來建立名稱為 “Bool_01” ~ “Boo_16” 的布林變數，再點選 “Create all” 完成設定。

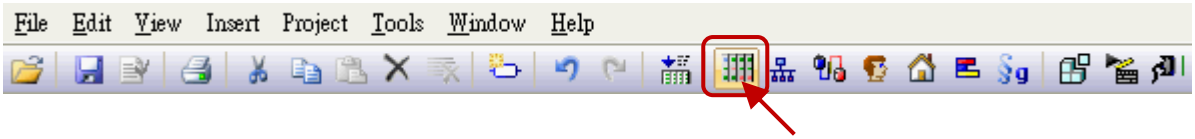




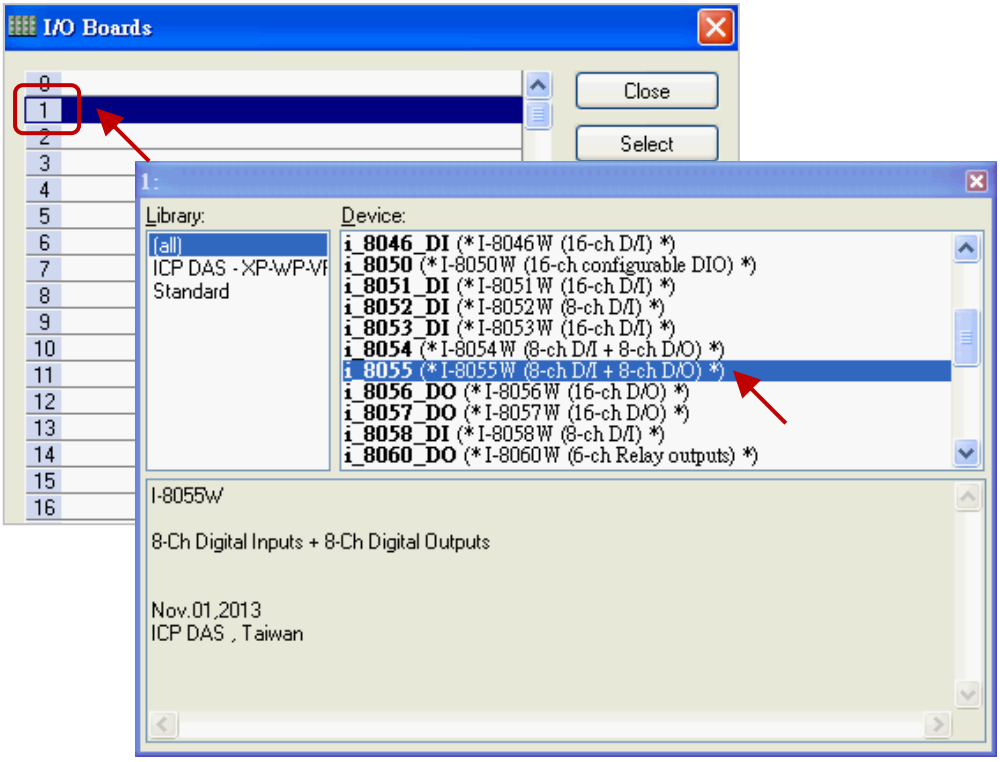
2.3.2 宣告 I/O 變數

此範例中，會在 PAC 的 Slot 1 插上一片 I-8055W 模組來顯示閃爍功能。因此，在 “Demo01” 專案中，需新增 I/O 連結來對應到 PAC 中的 I/O 模組。

1. 點選工具列的 "Open I/Os" 按鈕來新增 I/O 連結。

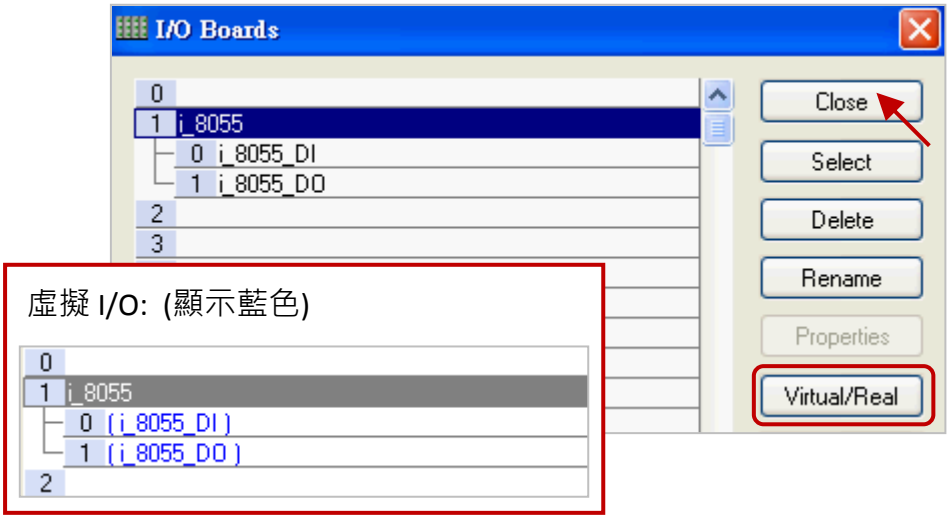


2. 滑鼠雙擊 "Slot 1" 再雙擊 "i_8055" 以完成 I/O 選取。

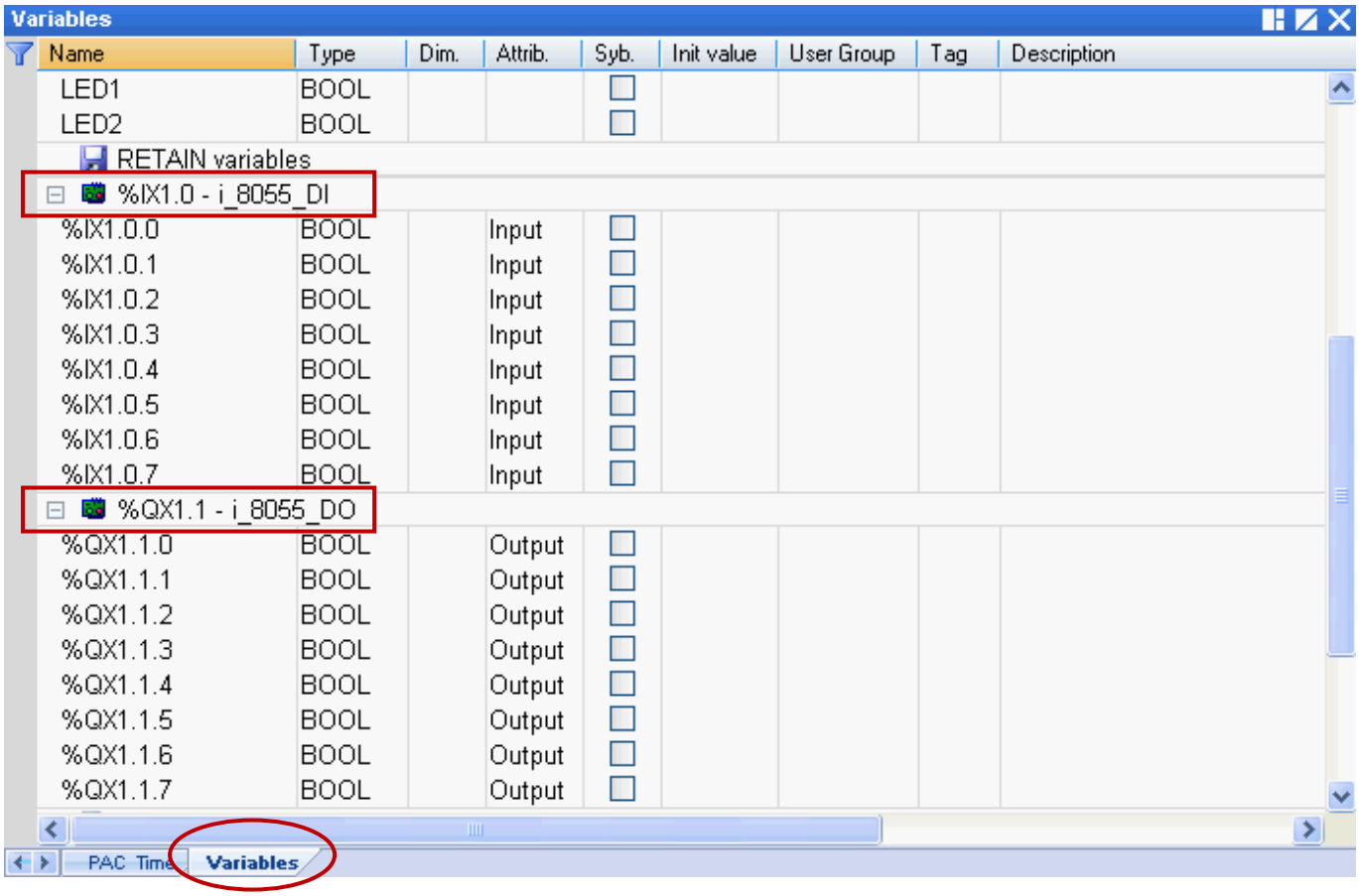


3. 點選 “Close” 離開 “I/O Boards” 視窗。

   註: 可點選 “Virtual/Real” 來切換為 虛擬 I/O (測試用) 或 真實 I/O，此處使用真實 I/O。



剛剛在 “I/O Boards” 視窗連上 “i_8055” 後，會自動在 “Variables” 視窗中新增 8 個 Input 與 Output 變數供程式使用。



%IX1.0 – i_8055_DI

I : 表示 “Input”

X : 表示 “Boolean”

1 : 表示 “Slot 1”

%QX1.1 – i_8055_DO

Q : 表示 “Output”

X : 表示 “Boolean”

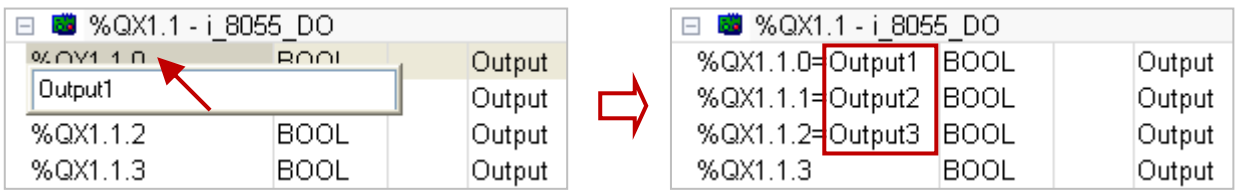
1 : 表示 “Slot 1”

%ID 或 %QD

D : 表示 “Integer/Real”

此範例中需使用 3 個 Output 變數，您可修改變數名稱以方便識別。

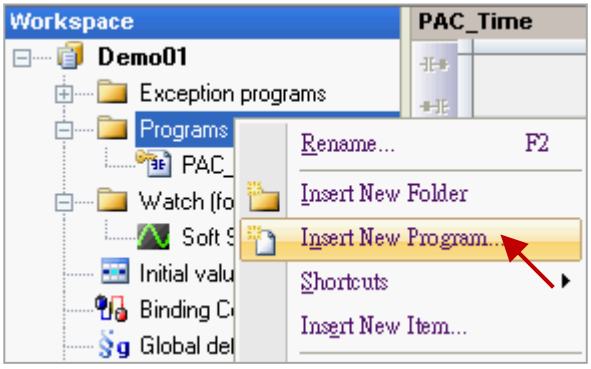
滑鼠雙擊欲使用的項目，輸入名稱後按 “Enter” 鍵完成設定。



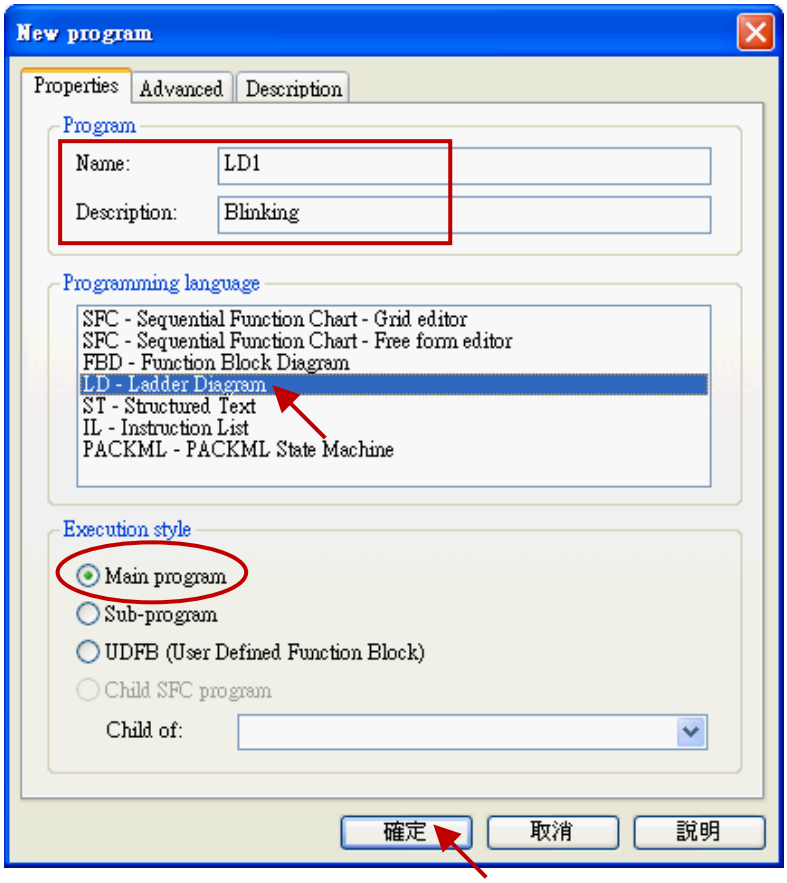
2.3.3 建立 LD 程式

在 “Demo01” 專案中，我們將新增一個名稱為 “LD1” 的程式，用來產生閃爍功能。

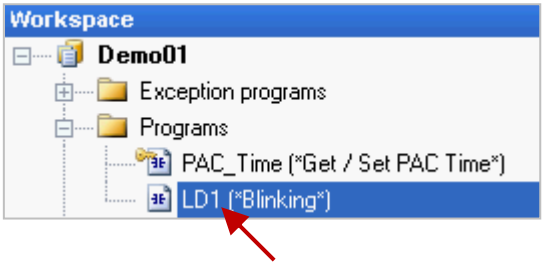
1. 工作區中，滑鼠右鍵點選 “Programs” 再選擇 “Insert New Program…”。



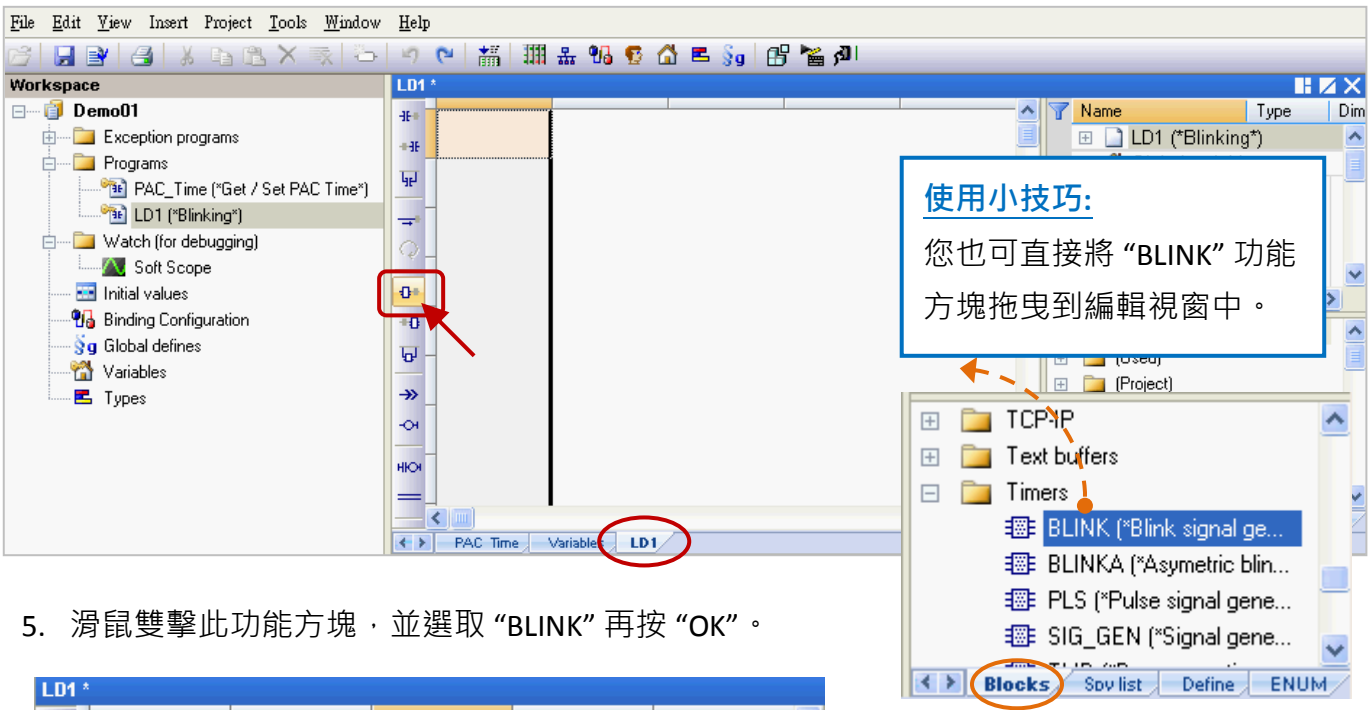
2. 於 “Name” 欄位中填入程式名稱，並在 “Desciption” 欄位加入簡單的描述。接著，選取 “LD – Ladder Diagram” 編程語言，再點選 “確定”。



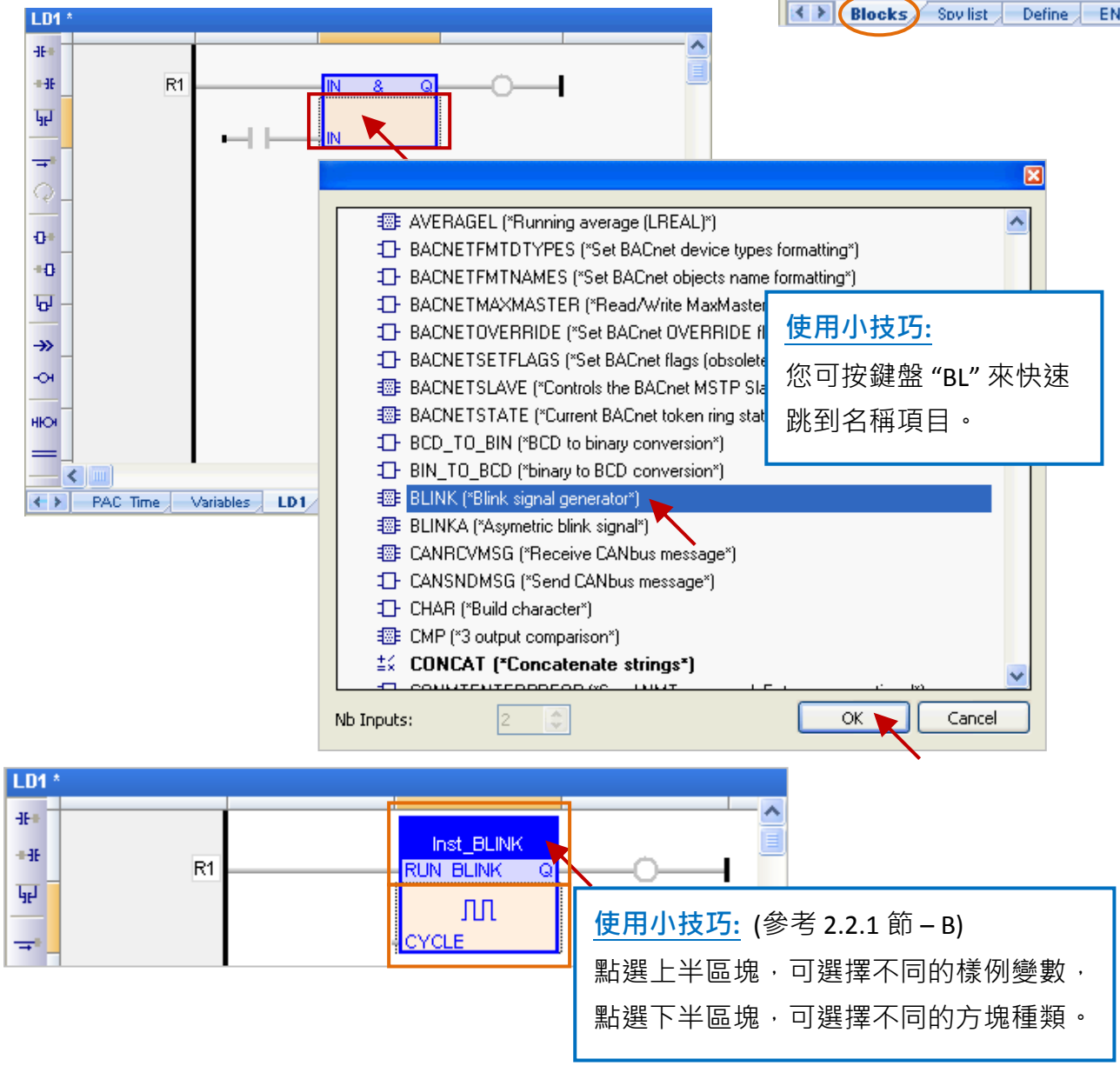
3. 滑鼠雙擊 “LD1” 程式來開啟編輯視窗。



4. 點選 “LD1” 視窗左側的 “Insert FB..” 按鈕來加入一個功能方塊 (Function Block)。



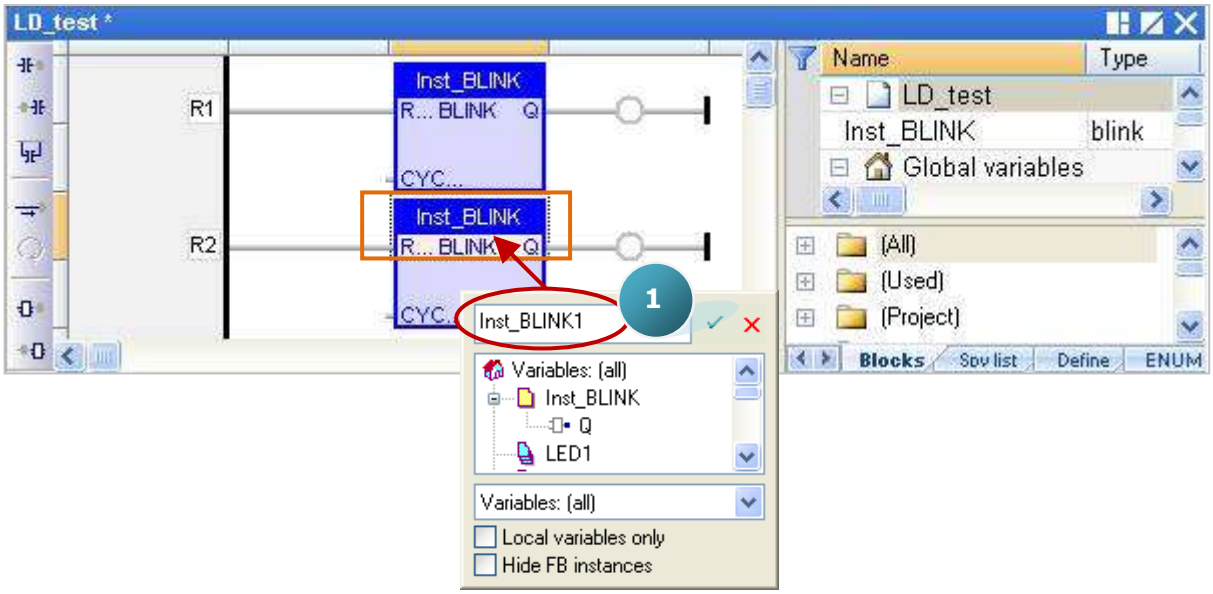
5. 滑鼠雙擊此功能方塊，並選取 “BLINK” 再按 “OK”。



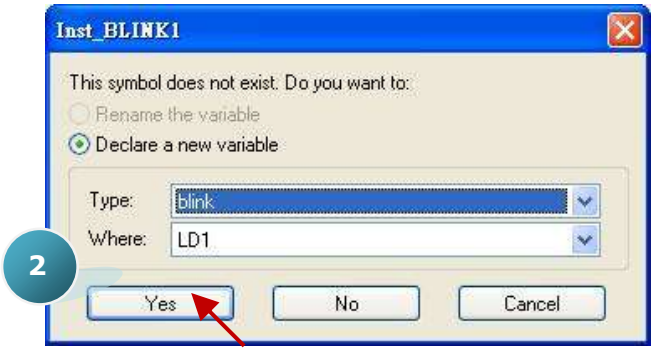
重要注意事項:

使用者在編輯程式時，可能會**以複製並貼上的方式**來新增功能方塊，但此方式會造成功能方塊的樣例變數名稱重複，而導致功能異常。因此，**請務必再建立一個新的 樣例變數 名稱。**

1. 滑鼠雙擊該功能方塊，並輸入新的名稱 (例如: “Inst_BLINK1”)，再點選 ✓ 完成設定。



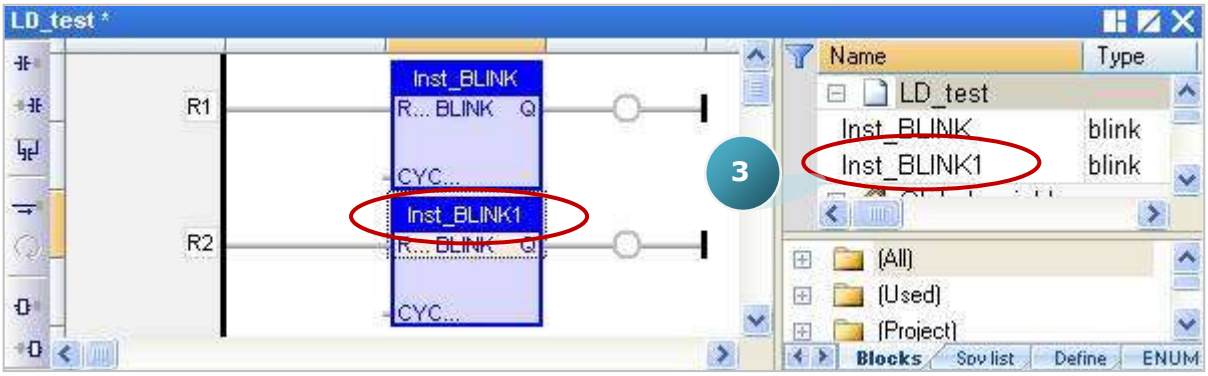
2. 於 “Inst_BLINK1” 變數視窗中，點選 “Yes” 來建立此樣例變數。



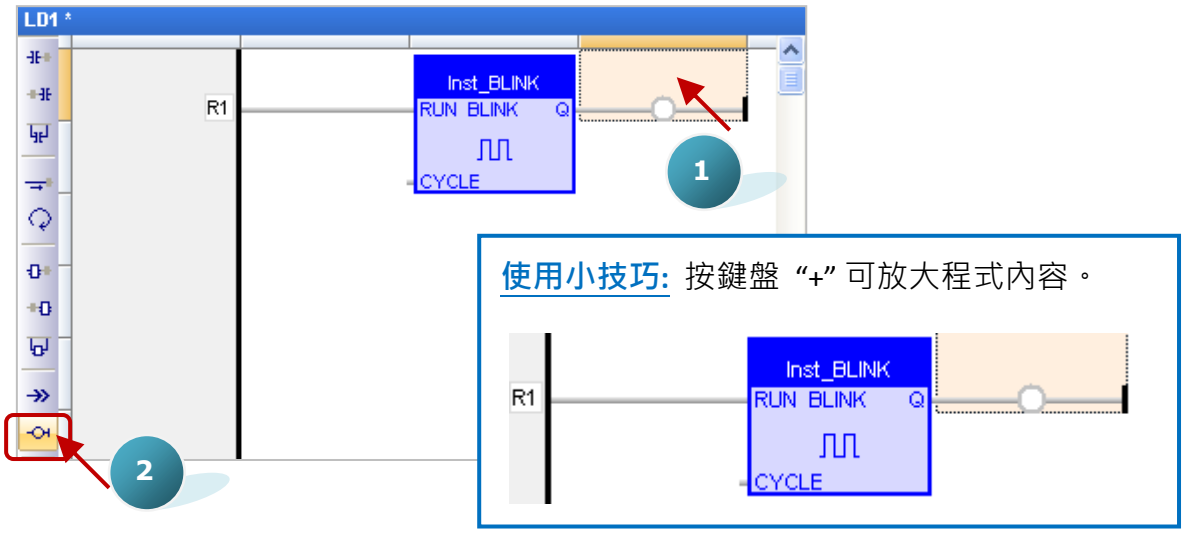
使用小技巧:

您也可依照相同的方式，滑鼠雙擊功能方塊旁的 “Coil” 來 新建/指定一個變數 (參考下方步驟 7)。

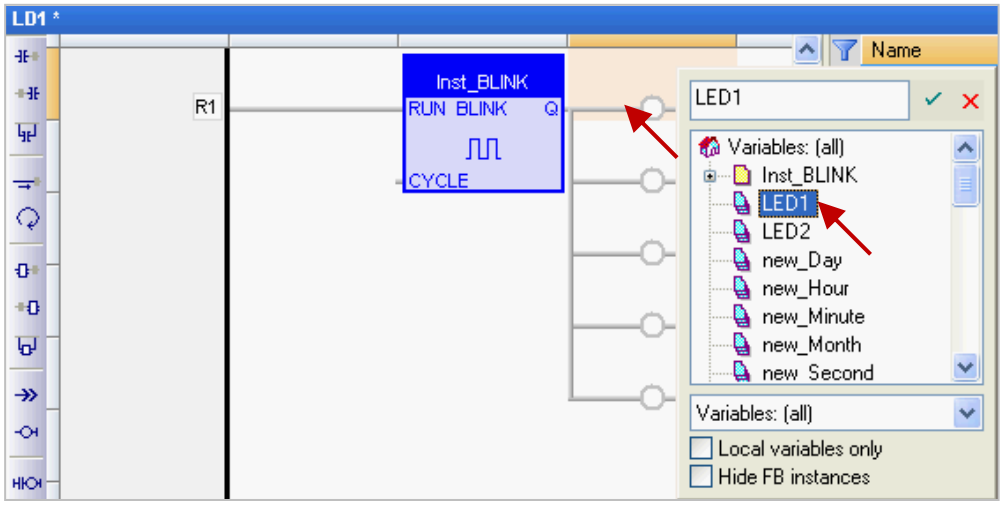
3. 變數區已建立了一個名稱為 “Inst_BLINK1” 的樣例變數。



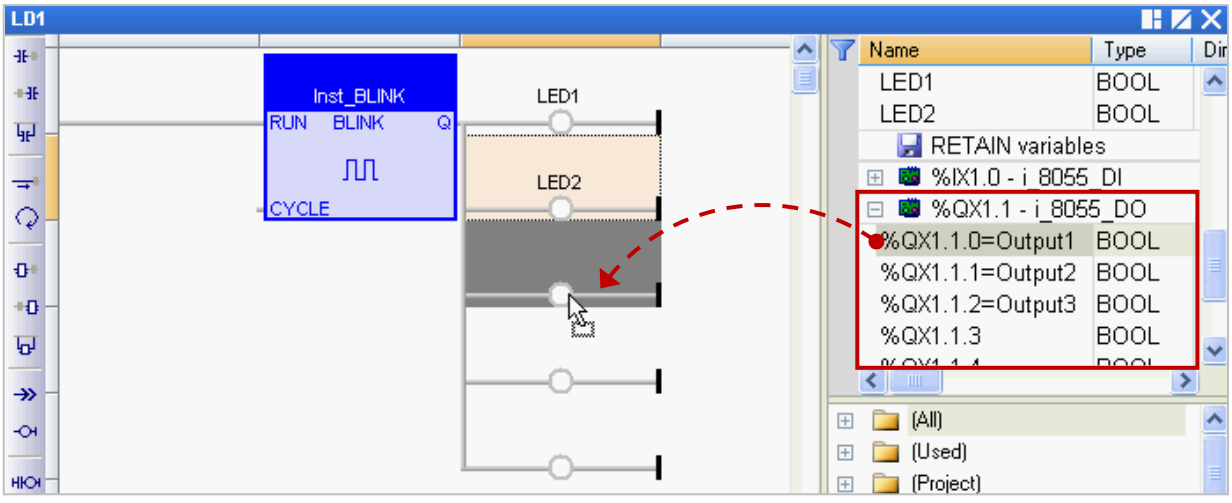
6. 點選 “BLINK” 功能方塊右方的 “Coil”，再連續點選 “Insert Coil” 按鈕來新增 4 個 “Coil”。



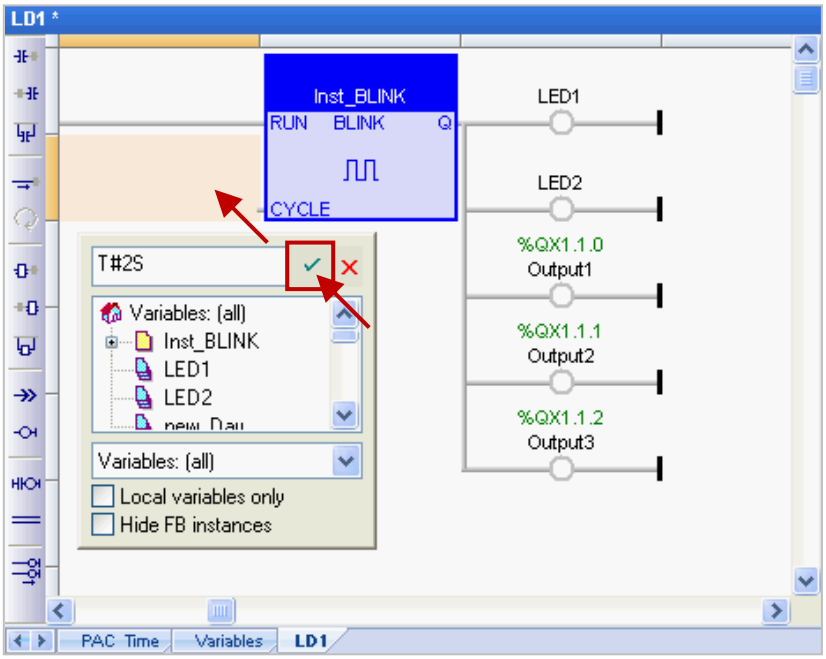
7. 滑鼠雙擊第一個 “Coil” 再雙擊 “LED1” 來指定變數。依照同樣的方式，將第二個 “Coil” 指定為 “LED2” 變數。



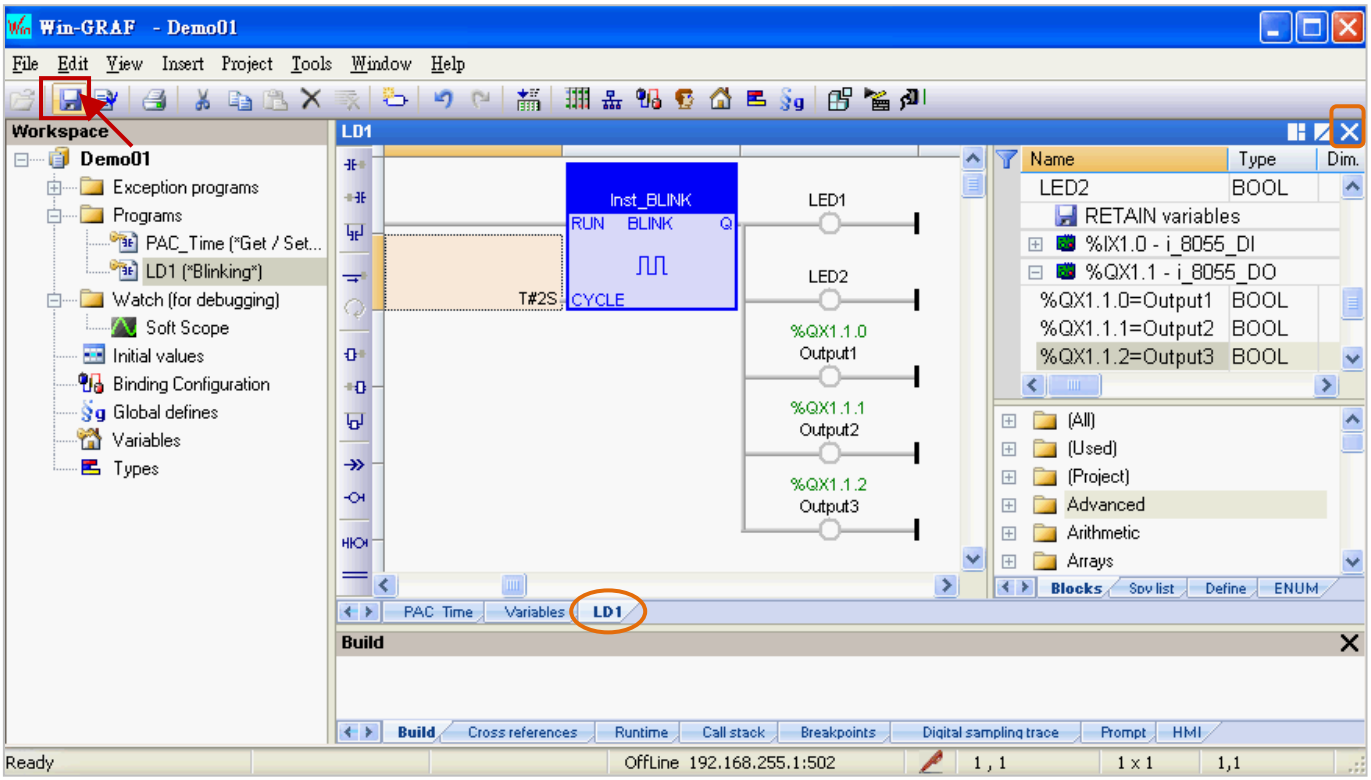
8. 接著，可用滑鼠拖曳方式來指定 “Output1”、“Output2”、“Output3” 變數。



9. 滑鼠雙擊 “CYCLE” 的左側，並輸入 “T#2S” (表示每 2 秒閃爍一次)，再點選 完成設定。



10. 最後，點選 “Save” 按鈕，儲存 “LD1” 程式。

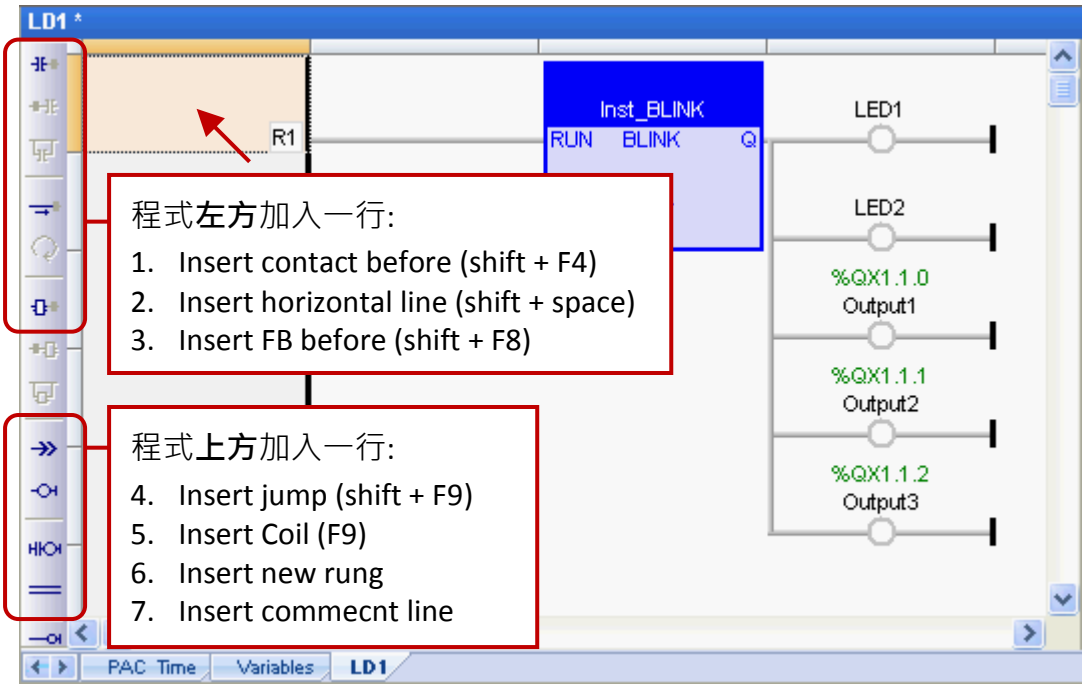


註: 表示該程式目前為開啟狀態 (鎖定，不可刪除)，可點選程式區右上角的 “X” 來關閉程式視窗 (解除鎖定，)。

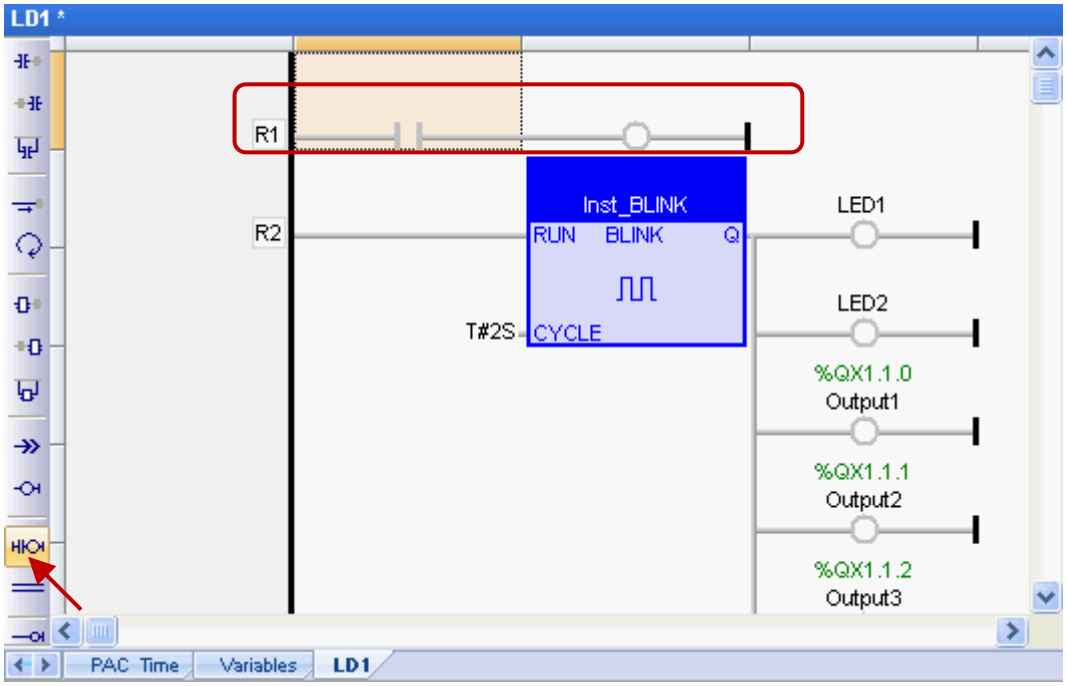
程式編輯完成後，若您想在程式的最上方新增一行程式:

1. 滑鼠點選程式編輯區的左上角，此時元件區會顯示可點選的項目，點選元件按鈕 (如圖，4 ~ 7) 可在第一行加入程式。

   註: 若點選元件按鈕 (如圖，1 ~ 3) 則會在程式的左方加入一行程式。



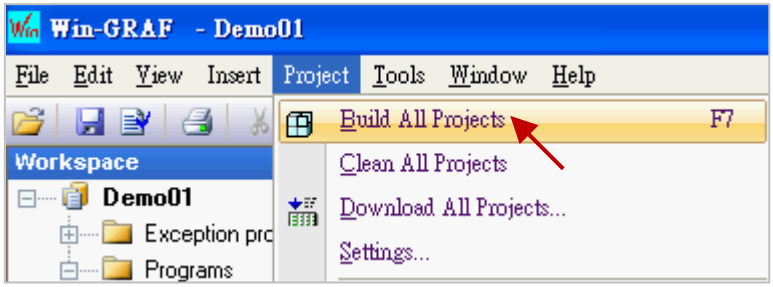
2. 此例，點選 “Insert new rung” 可在第一行加入程式。



2.3.4 編譯程式

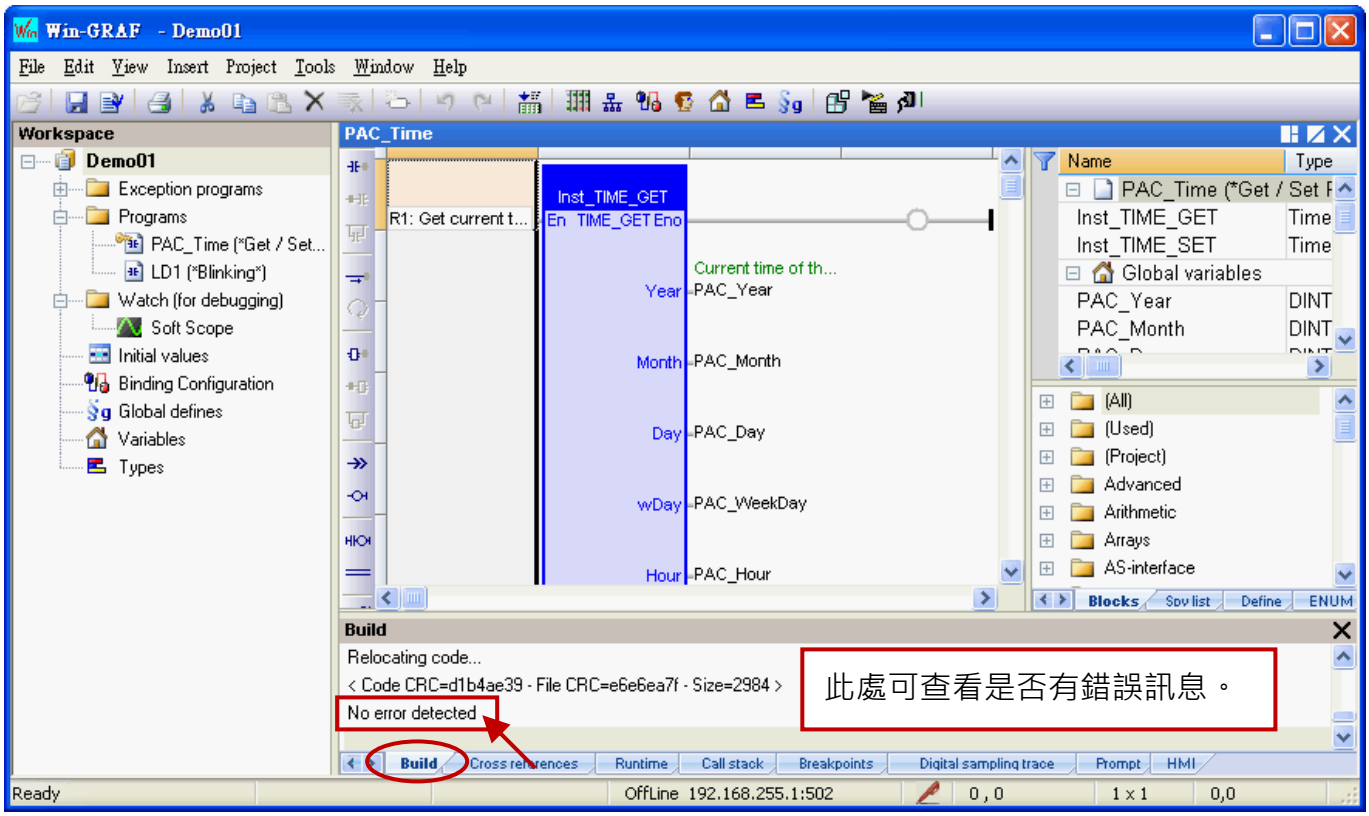
在上一節中，我們已新增並儲存了 LD 程式，為了讓 Win-GRAF 程式可在 PAC 上正常運作，接下來要進行程式編譯。

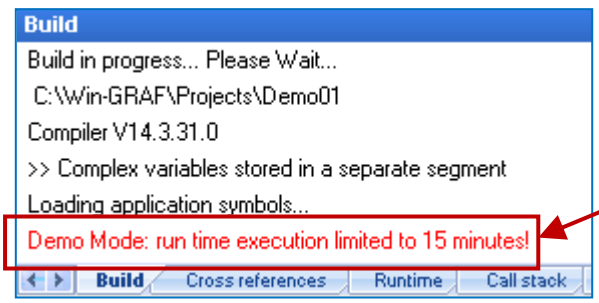
1. 點選功能表 “Project” 再選擇 “Build All Projects” 開始編譯程式。



2. 若訊息區中出現 “No error detected” 表示編譯成功。

   注意: 若編譯完成後，又再修改並儲存程式，請先點選 (上圖) “Clean All Projects” 清除先前的編譯結果並再執行編譯一次。



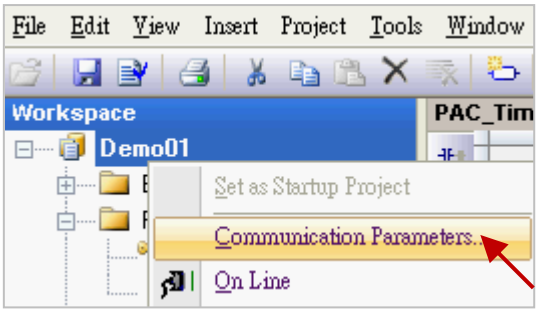


若使用 Demo Mode，"Build” 訊息區會提示 PAC 中的 Win-GRAF 專案只能運行 15 分鐘。

2.3.5 下載程式到 PAC

下載程式之前，請先設定好通訊參數 (僅支援 Ethernet 通訊埠的方式)。

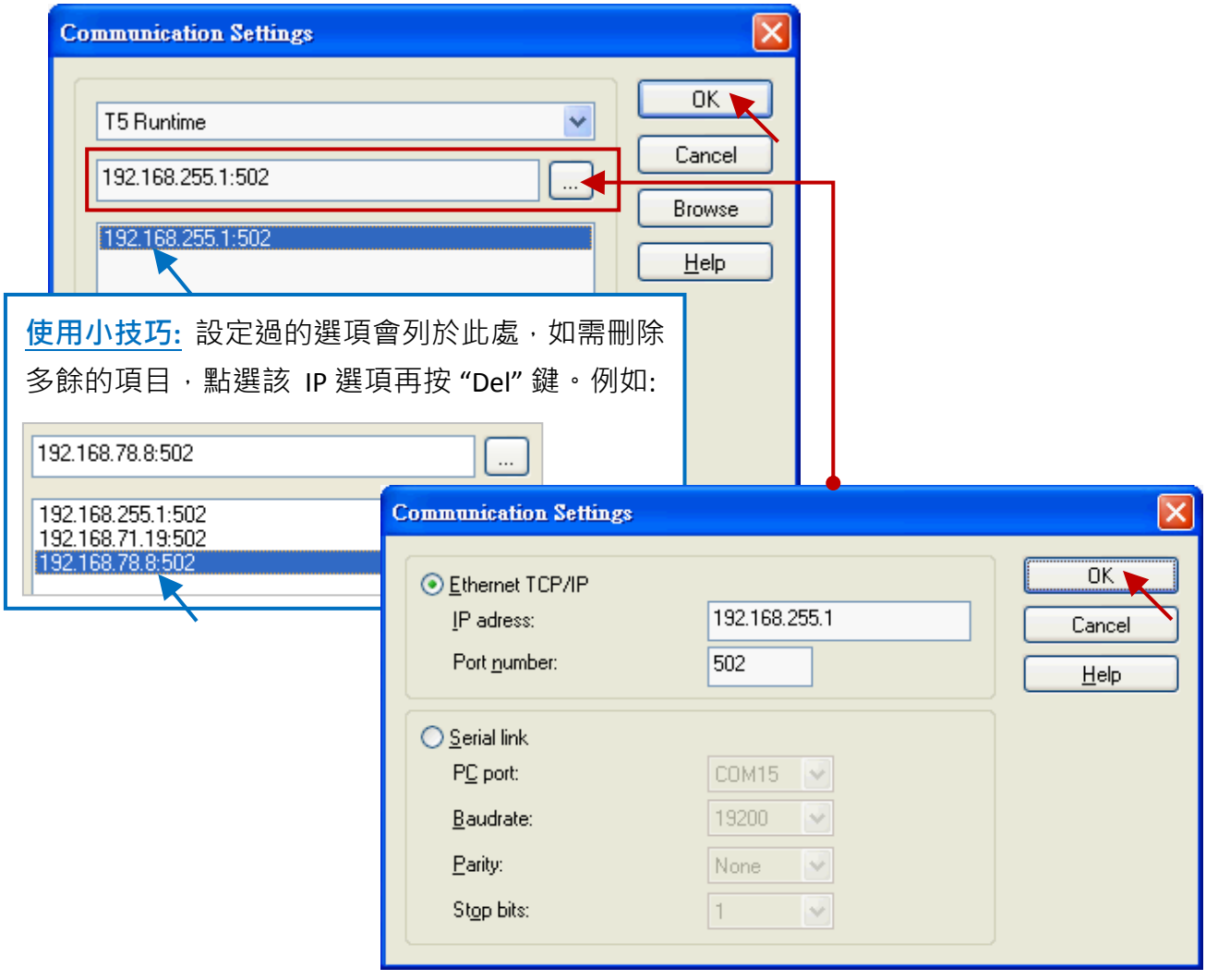
1. 滑鼠右鍵點選專案名稱 (即，Demo01)，再選擇 “Communication Parameters…” 開啟設定視窗。



2. 填入 “PAC IP:502” (例如: “192.168.255.1:502”) 來新增一個 IP 選項，並按 “OK”。

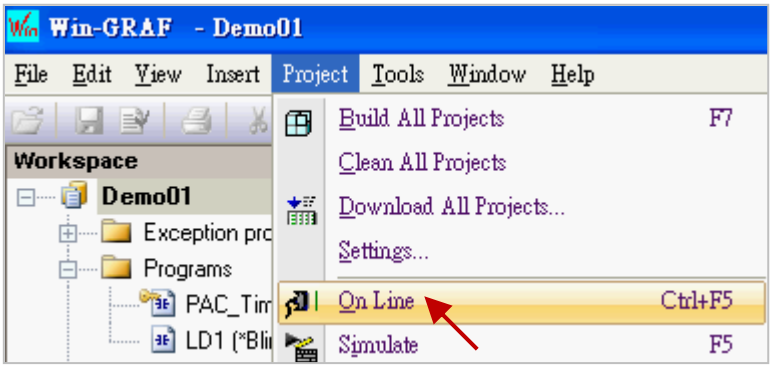
   也可點選  按鈕，來新增 或 修改 IP 位址。

   (**註:** PAC IP 出廠預設為 192.168.255.1，Port 號固定為 502)



3. 建立連線前，請確認您的 Win-GRAF PAC 已經開機且網路通訊正常。

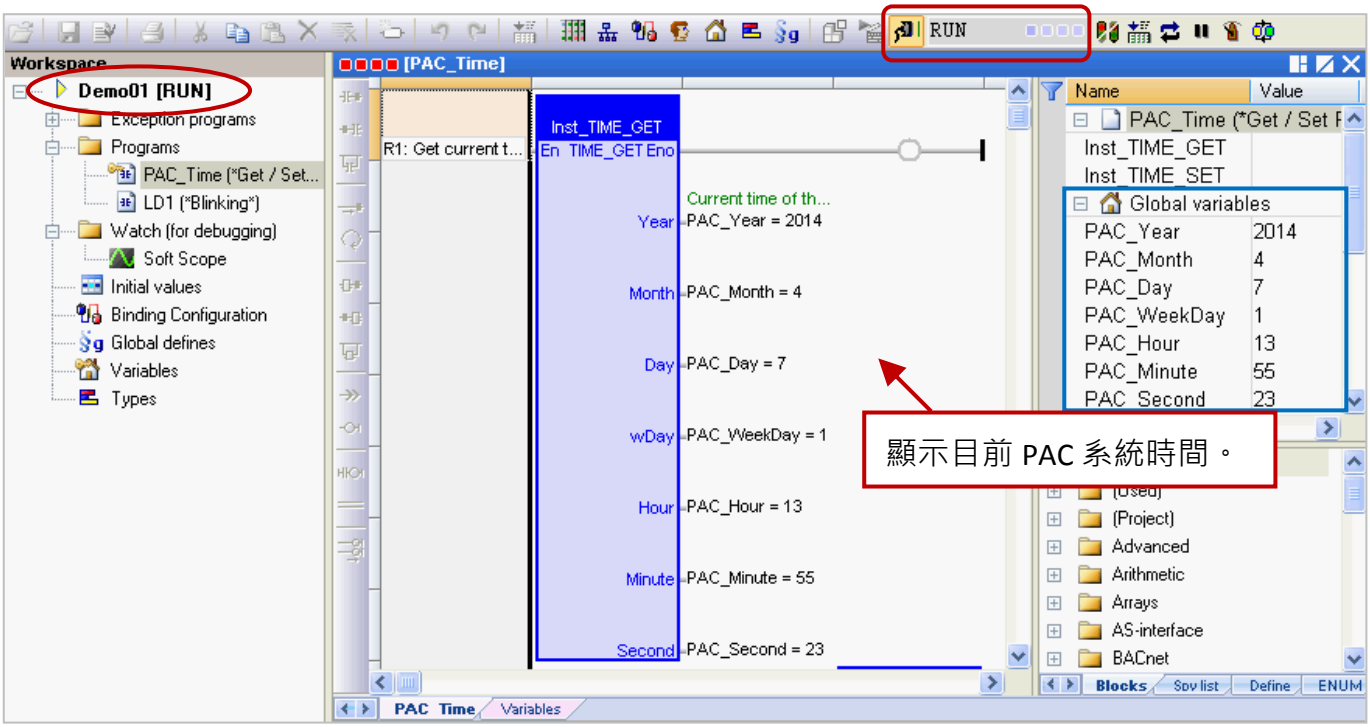
4. 點選功能表 “Project” 再選擇 “On Line”，或點選工具按鈕  來與 PAC 建立連線。



5. 如圖，若出現 “App: TEST” 字樣 (與目前專案名稱 “Demo01” 不同)，表示 PAC 內已有一個正在運行的專案 (名稱: “TEST”)，請點選工具按鈕 “Stop application” 停止該專案運行。

6. 接著，點選工具按鈕 “Download” 來下載 “Demo01” 專案。

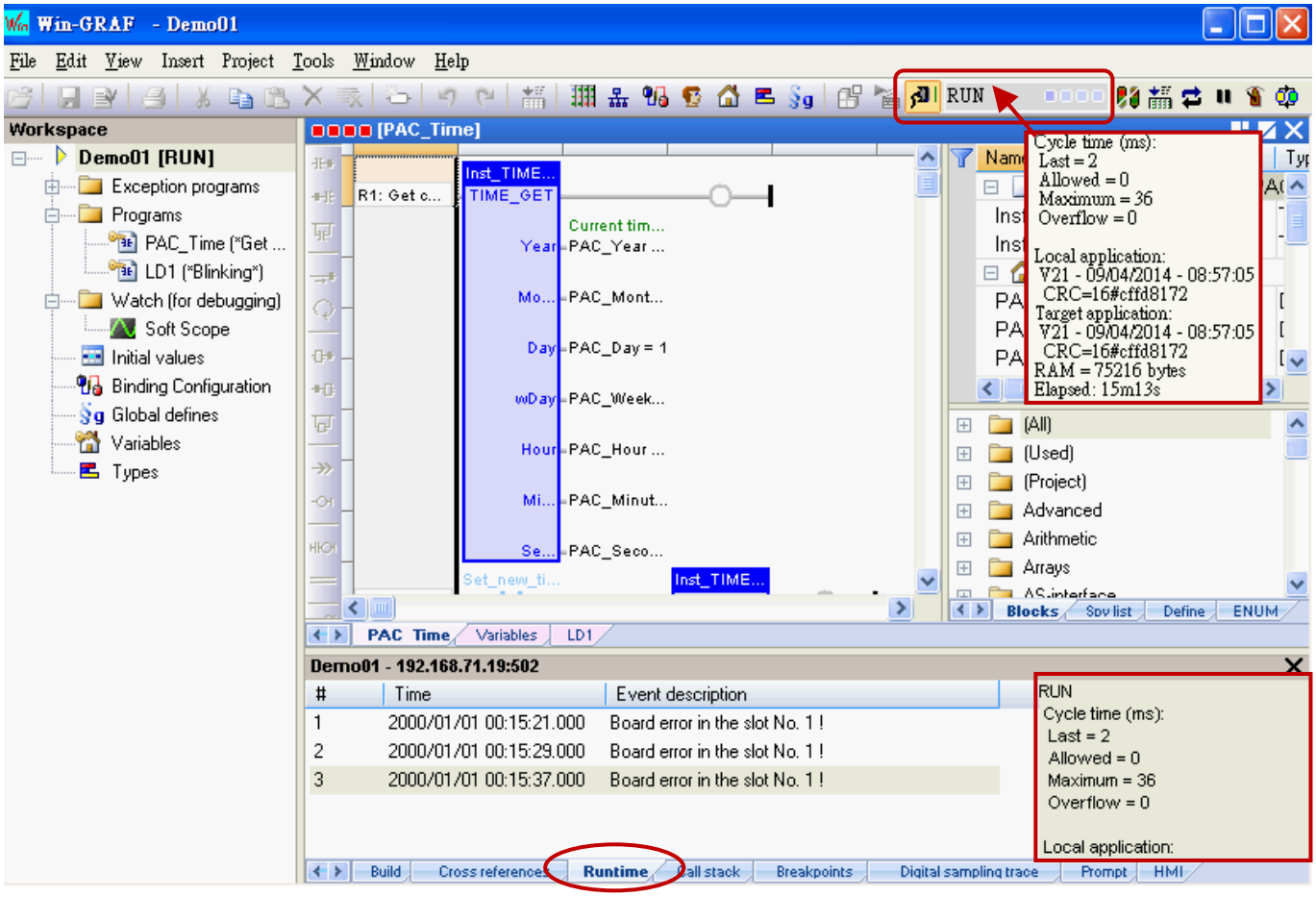
7. 若出現 “RUN” 字樣，表示連線成功且 “Demo01” 專案運行中。



註: 若下載過程中有出現錯誤訊息，請參考 附錄 B 來排除問題。

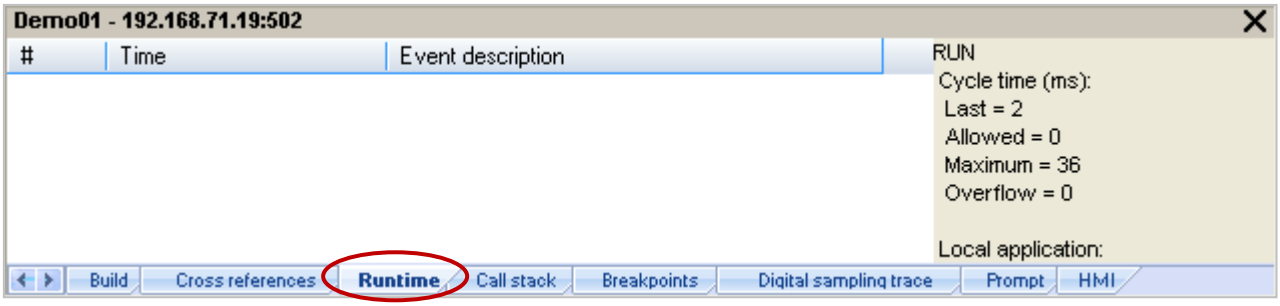
Cycle time

在連線的狀態下，滑鼠移到工具列 “RUN” 的位置可查看目前 PAC 內程式的 Cycle time。
您也可在右下方的訊息區中查看此資訊。



執行連線時 ()，會自動切換到 “Runtime” 訊息區，您可在此查看下載到 PAC 中的程式是否出現錯誤訊息。(例如: “Board error in the slot No. 1 !” 表示 PAC 的 Slot 1 上無插上 I/O 模組或是有異常，此例需在 PAC 的 Slot 1 插上 I-8055W 模組。)

請將 PAC 關機，插上 I-8055W 模組後再重新開機，並再執行連線時 () 一次。

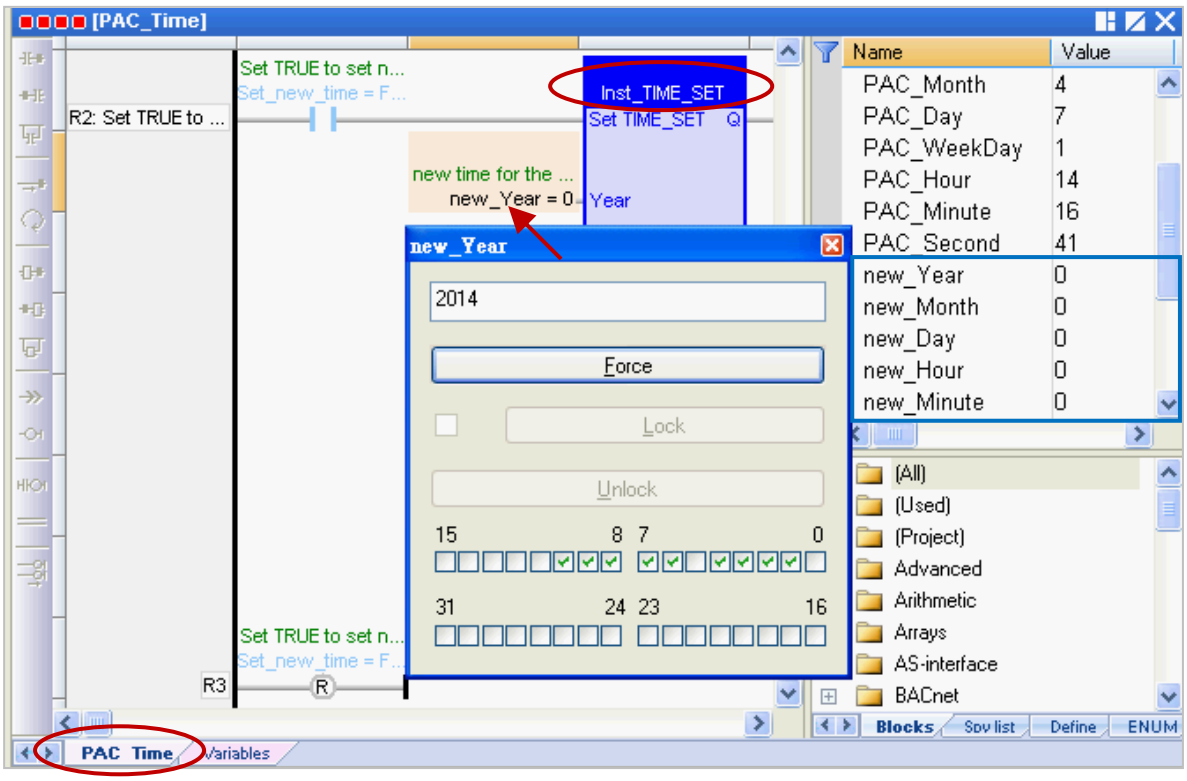


2.3.6 測試程式

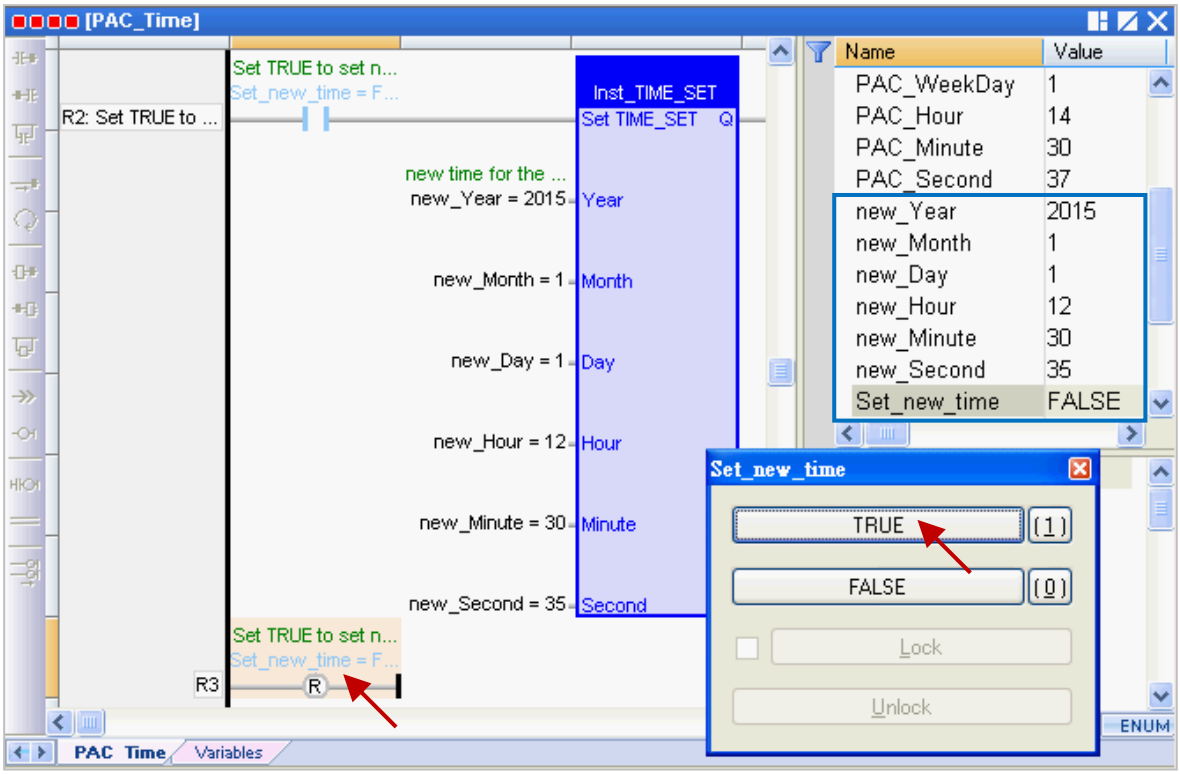
上一章節中，已成功的下載 “Demo01” 專案，接下來要介紹如何測試程式。

“PAC_Time” 程式:

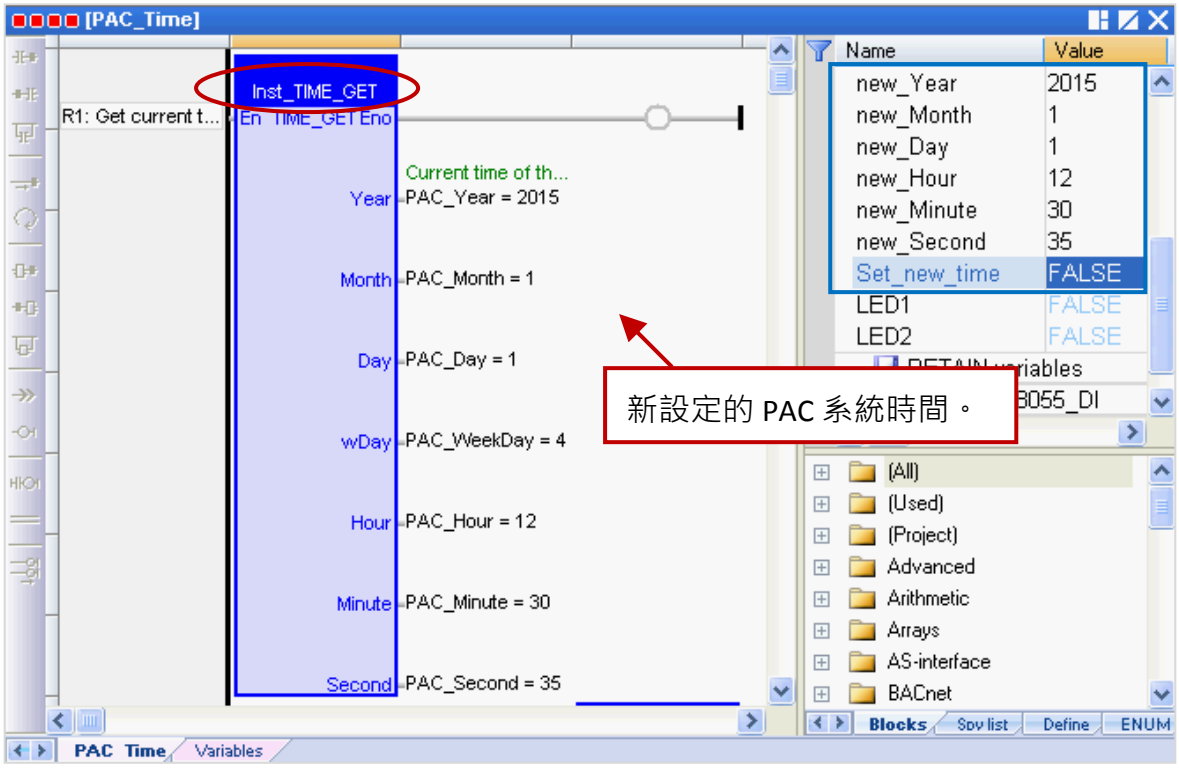
1. 在 “TIME_SET” 功能方塊中 (或變數區中)，滑鼠雙擊 各變數名稱 (例如: “new_Year”) 來依序修改系統時間。(假設: 將資料改為 2015 年 1 月 1 日，12 點 30 分 35 秒。)



2. 將 “Set_new_time” 變數設定為 “TRUE” 以寫入新的系統時間。

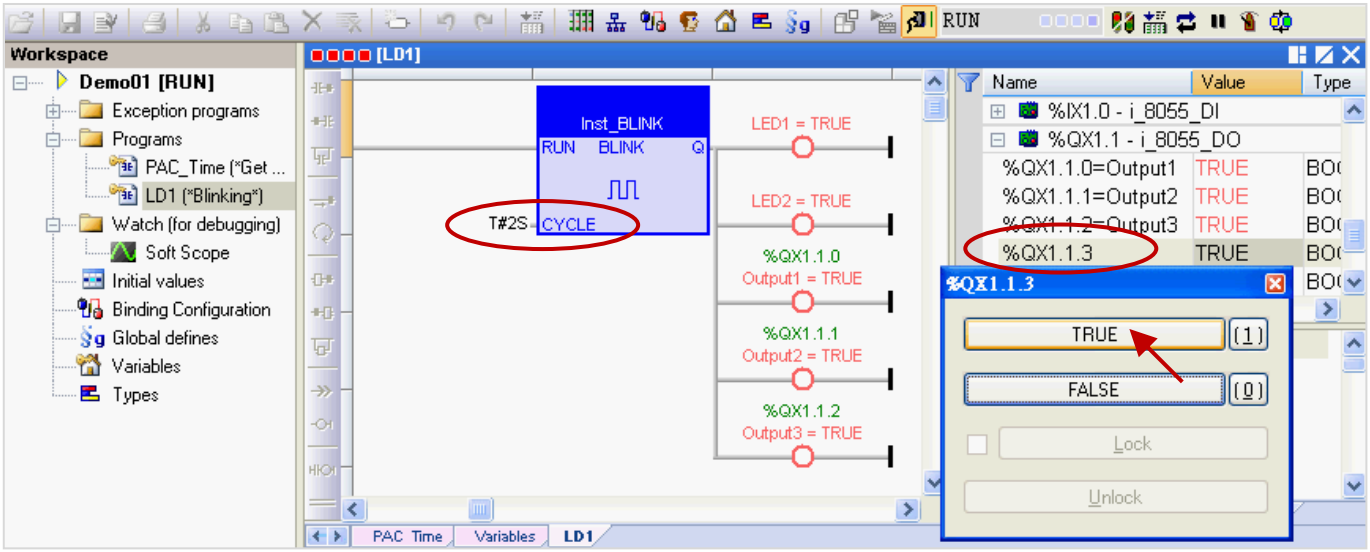


3. 在 “TIME_GET” 功能方塊中 (或變數區中)，顯示了新設定的 PAC 系統時間，且 “Set_new_time” 變數會自動重置為 “FALSE”。



“LD1” 程式:

4. 當 “Demo01” 專案運行時，可查看 PAC 上 (Slot 1) 的 I-8055W I/O 模組 - DO0、DO1、DO2 是否如同設定值 (即，"T#2S" 表示每 2 秒閃爍一次)。您也可改為在 “CYCLE” 左側指定一個資料型態為 “TIME” 的變數，設定方式參考 2.3.1 節，以便修改時間設定。)
5. 若將變數區的 “%QX1.1.3” 設定為 “TURE”，可見到 I-8055W I/O 模組 – DO3 的燈號為亮。



6. 您可再次點選工具按鈕  來取消與 PAC 的連線。

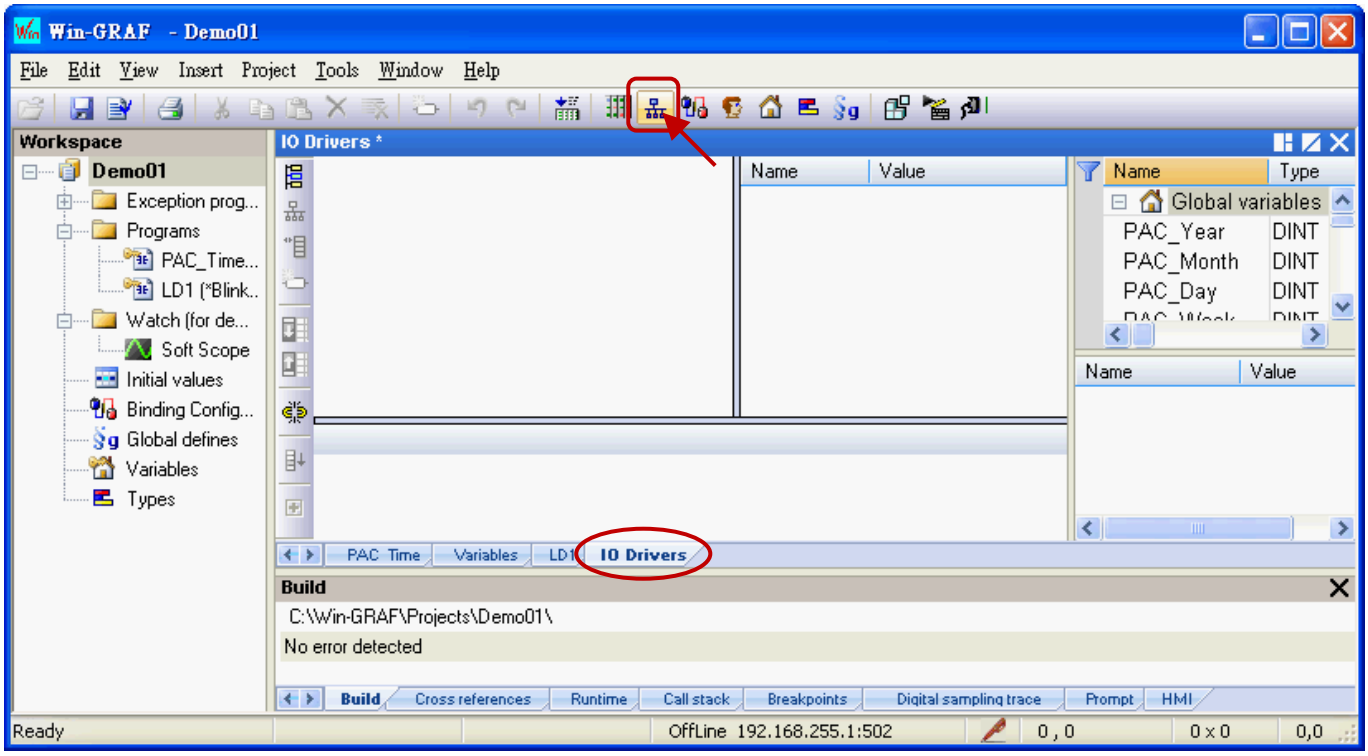
注意: 請勿點選 “Stop Application” 來停止連線，這會停止 PAC 中運行的程式。

第 3 章　Modbus Slave: 開放 Win-GRAF PAC 與圖控/HMI 軟體來相互溝通

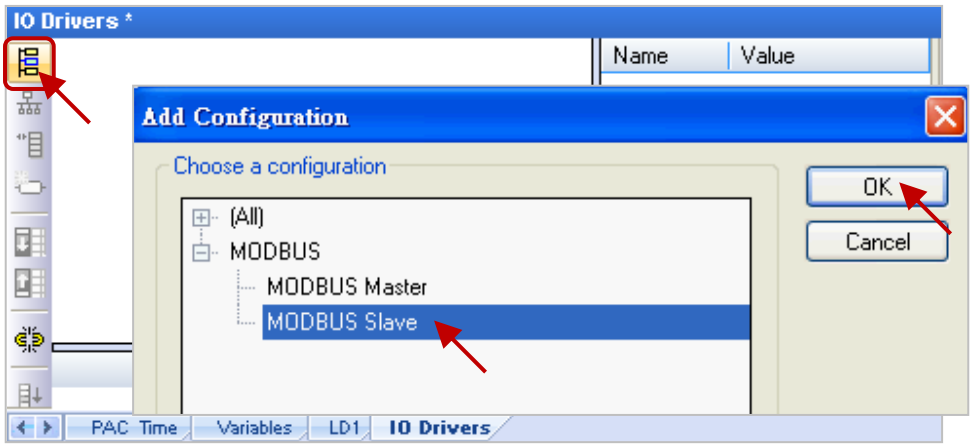
在第 2 章的 “Demo01” 專案中，說明了如何讀/寫 PAC 中的系統時間 (即 “PAC_Time” 程式) 與一個閃爍功能 (即 “LD1” 程式)，此章節將提供 “Demo01” 專案中的變數，讓圖控軟體 (例如: 泓格科技的 “InduSoft”) 或 HMI 軟體來進行存取。Win-GRAF 軟體提供了兩種方式，可開放 PAC 中的資料，一種是啟用 Win-GRAF PAC 為 Modbus TCP Slave，另一種是啟用 Win-GRAF PAC 為 Modbus RTU Slave (需先完成 3.1 節 Modbus Slave 設定，見 3.2 節)，請依照以下內容來進行設定。

3.1　啟用 Win-GRAF PAC 為 Modbus TCP Slave

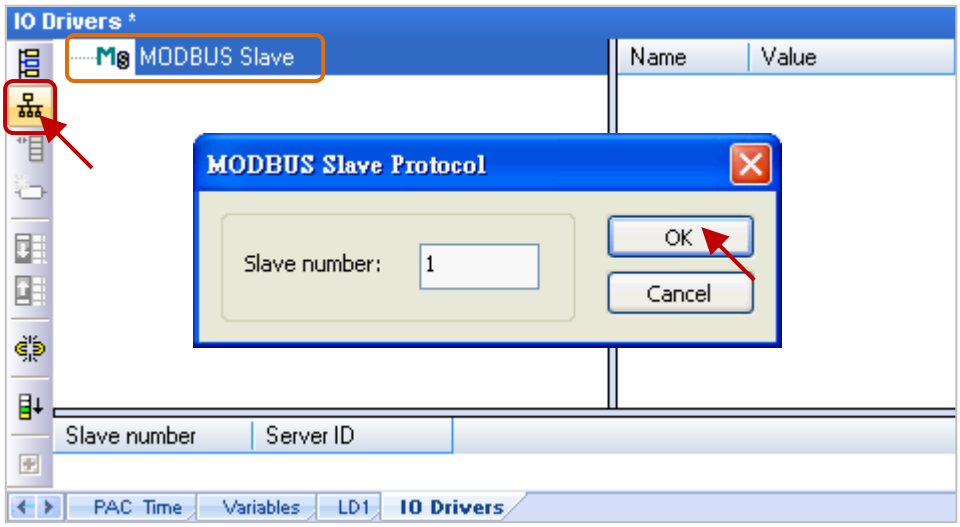
1. 滑鼠點選工具列上的 “Open Fieldbus Configuration” 按鈕來開啟 “I/O Drivers” 視窗。



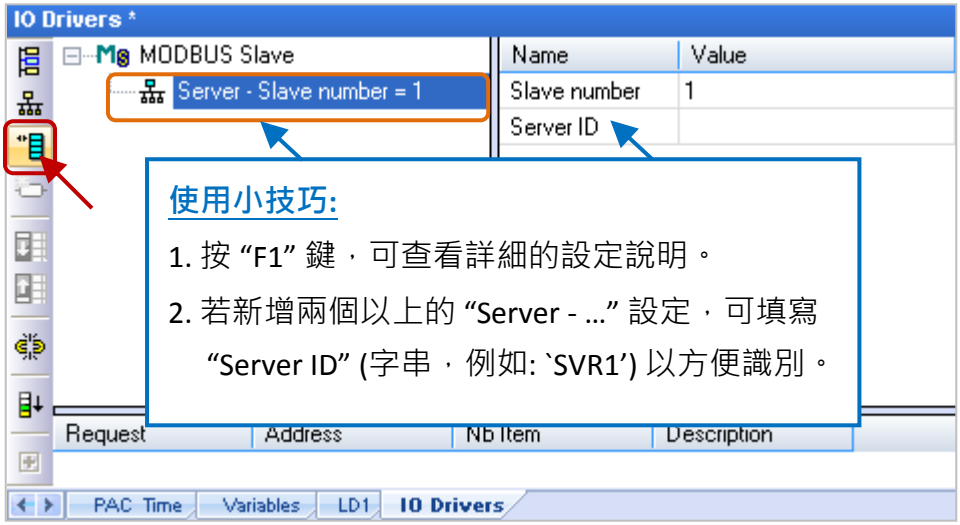
2. 點選 “I/O Drivers” 視窗左側的 “Insert Configuration” 按鈕，再點選 “MOSBUS Slave” 並點選 “OK” 來啟用一個 Modbus TCP Slave。



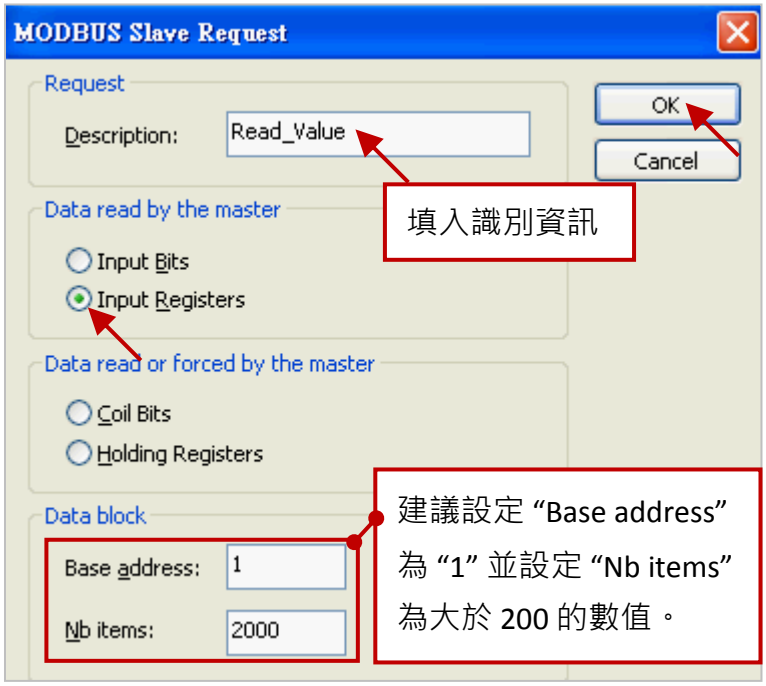
3. 點選左側的 “Insert Master/Port” 按鈕，並設定 “Slave number” (此例為 “1”)，再點選 “OK”。



4. 點選左側的 “Insert Slave/Data Block” 按鈕，來開啟 “MODBUS Slave Request” 設定視窗。



5. 在 “Description” 填入識別的資訊並點選 "Input Registers" 選項。



可供 Modbus Master 讀取資料:

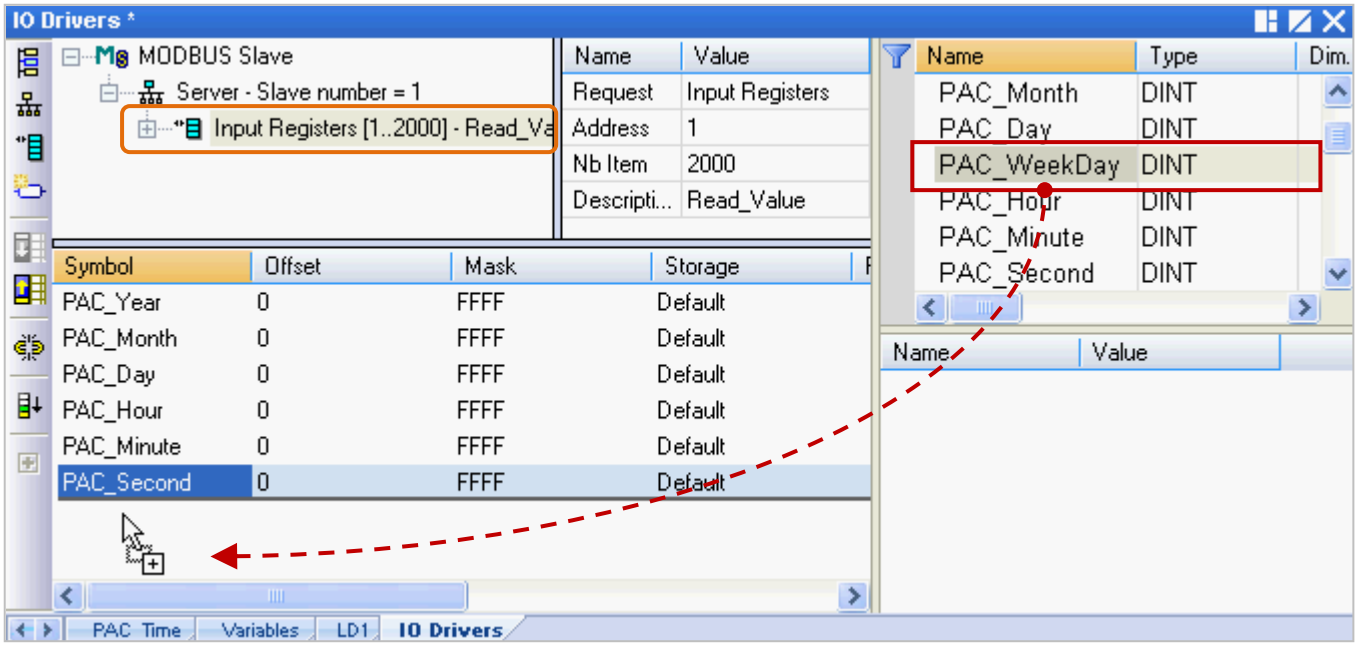
選項	資料型態
Input Bits	BOOL
Input Registers	BYTE, INT, DINT, REAL, …等。

可供 Modbus Master 寫入資料:

選項	資料型態
Coil Bits	BOOL
Holding Registers	BYTE, INT, DINT, REAL, …等。

(資料型態，可參考附錄 A)

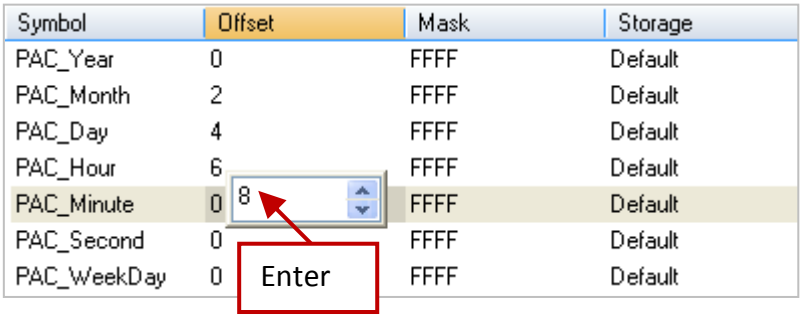
6. 如上圖，建議將 “Base address” 設定為 “1”，而 “Nb items” 表示一個 “Data block” 最多可提供多少個變數資料，若 Modbus Master (例如，圖控軟體) 要求的資料位址大於此數值 (此例設定為 “2000”)，則 Modbus Slave (即，Win-GRAF PAC) 將不回應。
7. 將變數區的變數 (此例為 “PAC_xxx”，資料型態: DINT) 以滑鼠拖曳的方式，拉到 “Symbol” 區塊。



8. 滑鼠雙擊 “Offset” 欄位並填入數值後，按 “Enter” 完成設定。

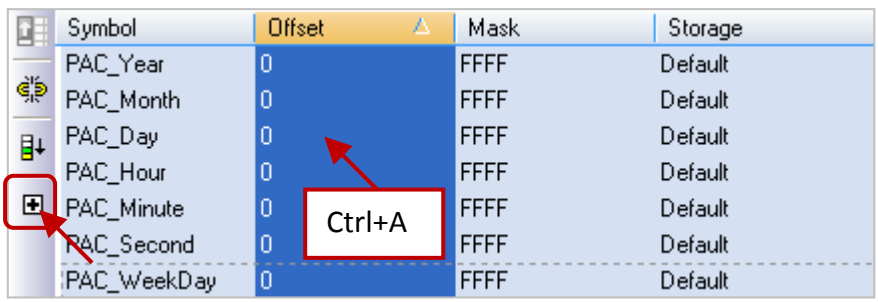
注意: (1) “Offset” 的值是由 “0” 開始，而 “Offset” 的值加 1 (Base address) 才是該變數的 Modbus 位址。

(2) 若選用的是 32-bit 或以上的資料型態 (此例為 DINT)，則需佔用 2 個 Modbus 位址如下表，“0, 2, 4, 6, ….”。(更多關於資料型態，可參考附錄 A。)

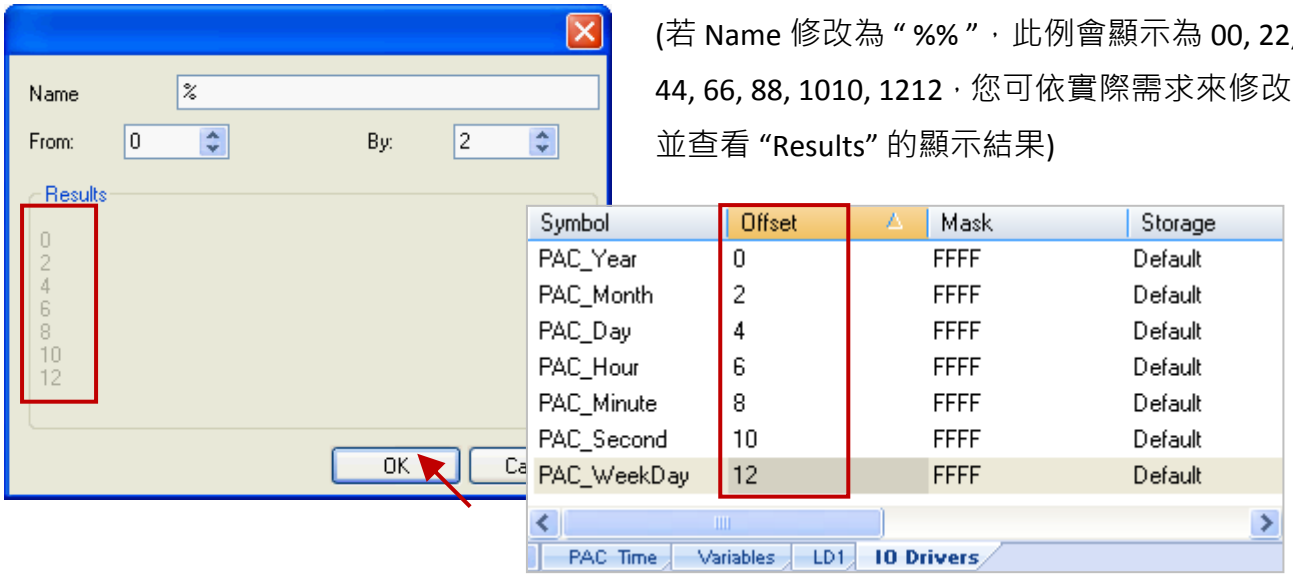


使用小技巧:

滑鼠點選任一欄位再按 “Ctrl+A” 全選，點選左側的 “Iterate Property” 按鈕，開啟設定視窗。

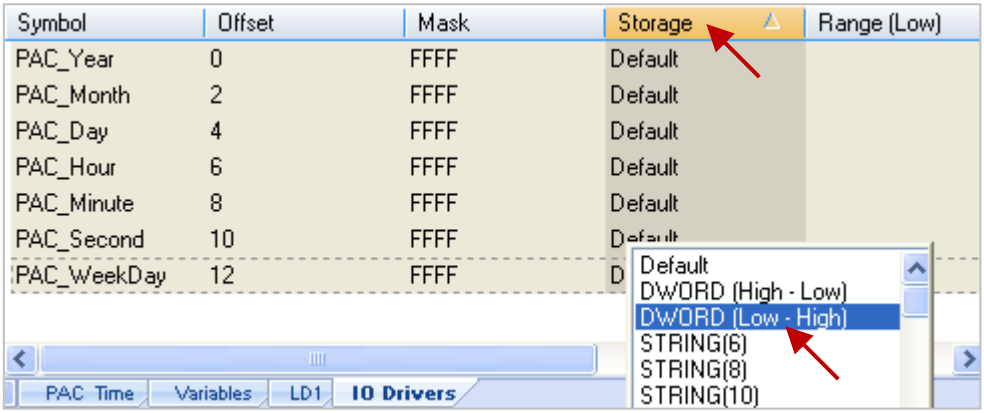


“Name” 維持設定 “ % ”、將 “From” 填入 0、“By” 填入 2，再點選 “OK”。

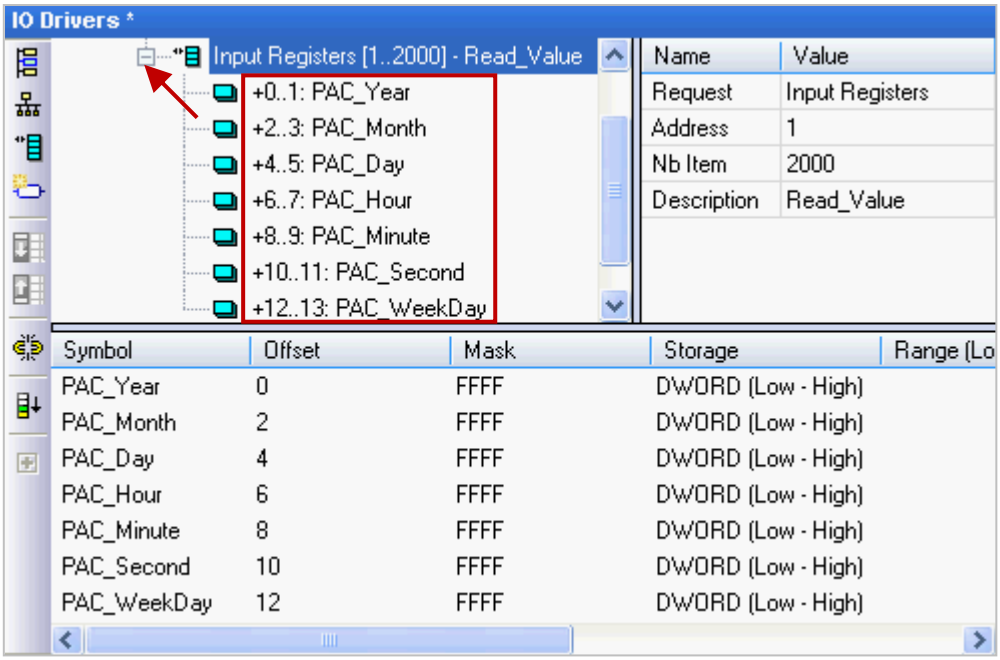


(若 Name 修改為 “ %% ”，此例會顯示為 00, 22, 44, 66, 88, 1010, 1212，您可依實際需求來修改，並查看 “Results” 的顯示結果)

9. 滑鼠點選 “Storage” 來選取整欄，再按 “Enter” 鍵顯示下拉選單，接著選擇 “DWORD (Low – High)” 再按 “Enter” 鍵完成設定。(若資料為 16-bit 或以下，不需設定 “Storage” 項目)



您可展開這個 “Data Block”，如圖，“Offset” 的值加 1 (Base address) 為該變數的 Modbus 位址。

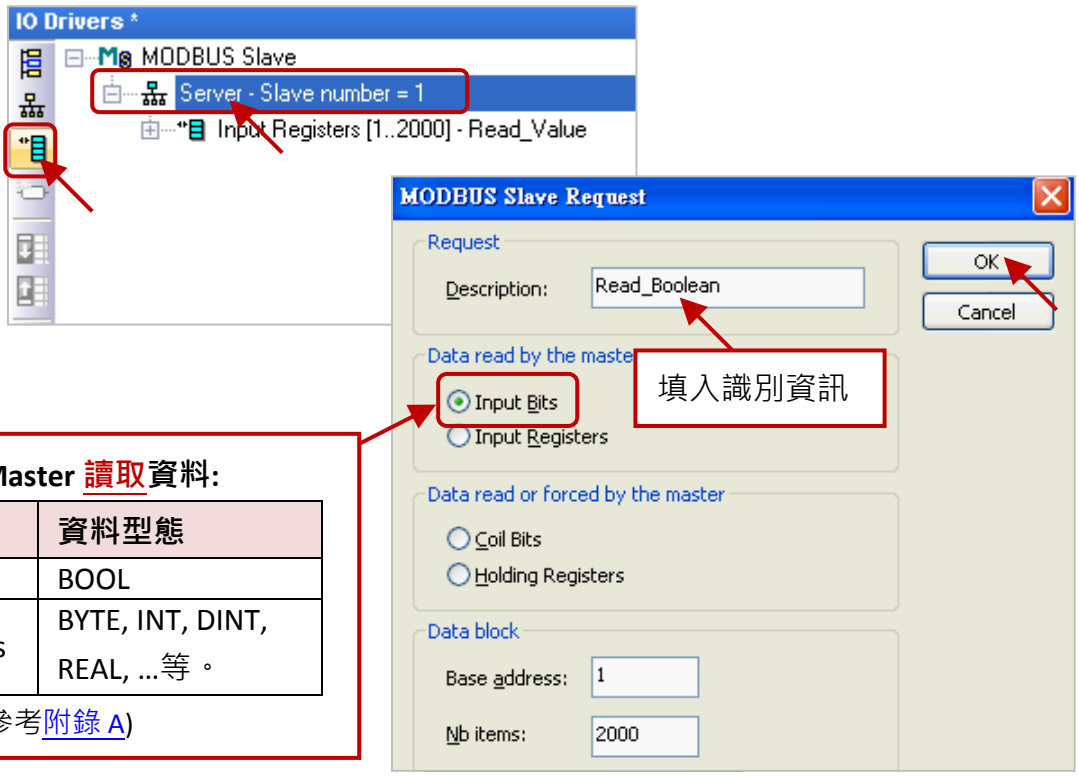


10. 接著，需再新增一個 “Data Block” 用來供 Modbus Master 讀取 Boolean 資料。

設定方式與步驟 4 ~ 8 類似:

(1) 滑鼠點選 “Server - …” 再點選左側的 “Insert Slave/Data Block” 按鈕，來開啟設定視窗。

(2) 設定視窗中，填入識別資料後，選擇供讀取的選項 “Input Bits”、設定 “Base address” 為 “1”、設定 “Nb items” 為 “2000”。

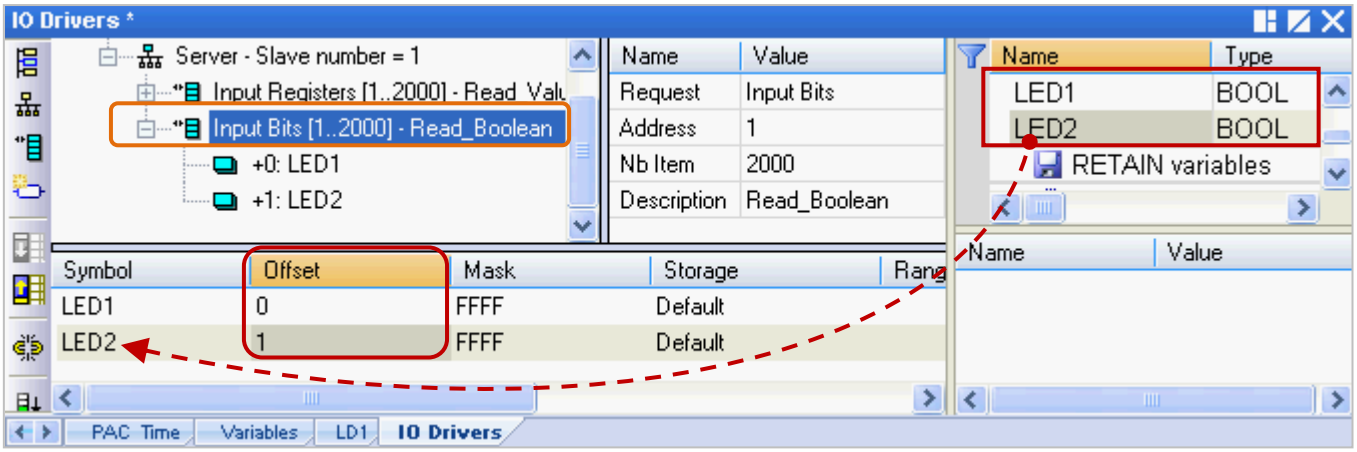


可供 Modbus Master 讀取資料:

選項	資料型態
Input Bits	BOOL
Input Registers	BYTE, INT, DINT, REAL, …等。

(資料型態，可參考附錄 A)

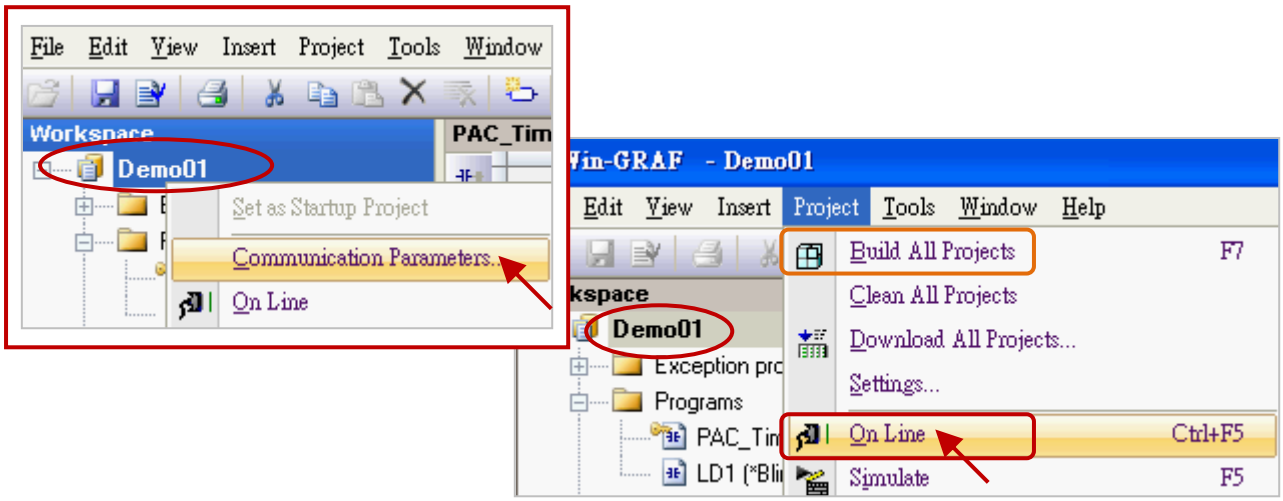
(3) 將變數區的 BOOL 變數 “LED1”、“LED2” 拖曳到 “Symbol” 區域並設定其 “Offset” 為 “0”、 “1”。

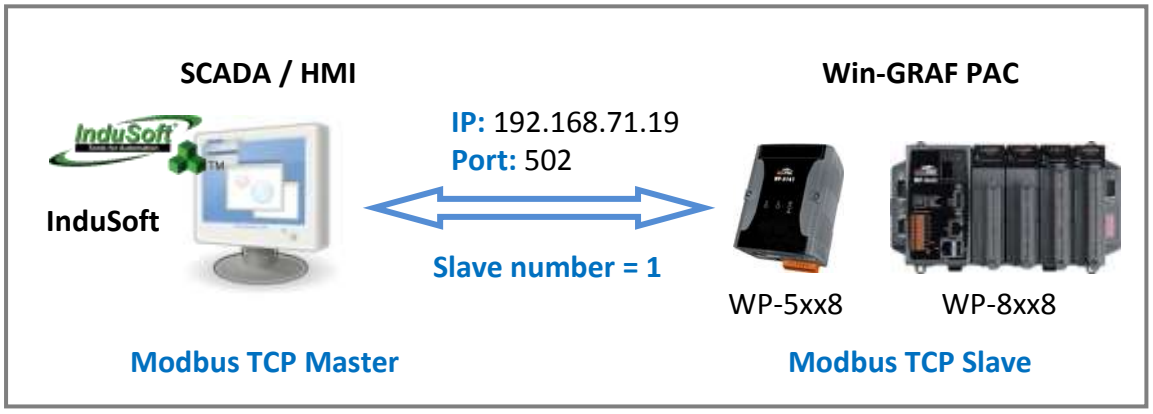


您已完成 Modbus Slave 設定，接下來請重新編譯並下載程式到 Win-GRAF PAC 中。

11. 點選功能表 “Project” > “Build All Projects” 重新編譯程式 (參考 2.3.4 節)。
出現 “No error detected” 表示編譯成功。

12. 滑鼠右鍵點選專案名稱 (即，Demo01)，再選擇 “Communication Parameters…” 設定 PAC IP (例如: 192.168.71.19:502)，點選功能表 “Project” > “On Line” (或 ) 來建立連線，並將專案下載到 Win-GRAF PAC 中 (參考 2.3.5 節)。





(您可參考 P1-1，來查看所有支援的 PAC 型號)

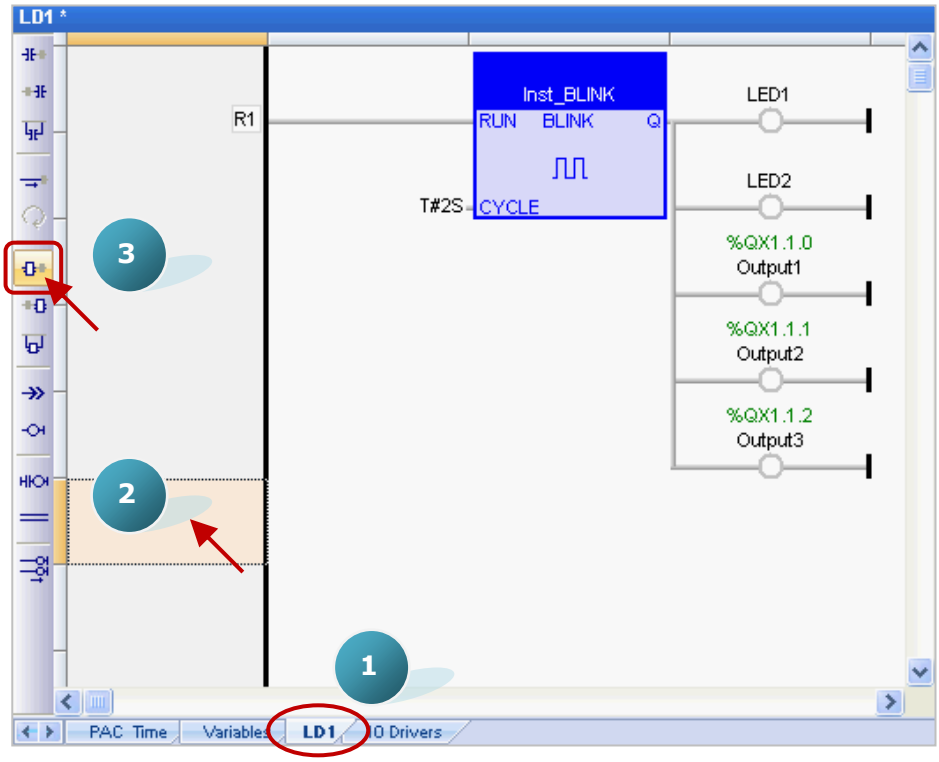
如此之後 HMI 或 SCADA 軟體便可使用 Modbus TCP protocol 來存取以上所列的、有設定 Modbus 位址的 Win-GRAF 變數。

3.2 啟用 Win-GRAF PAC 為 Modbus RTU Slave

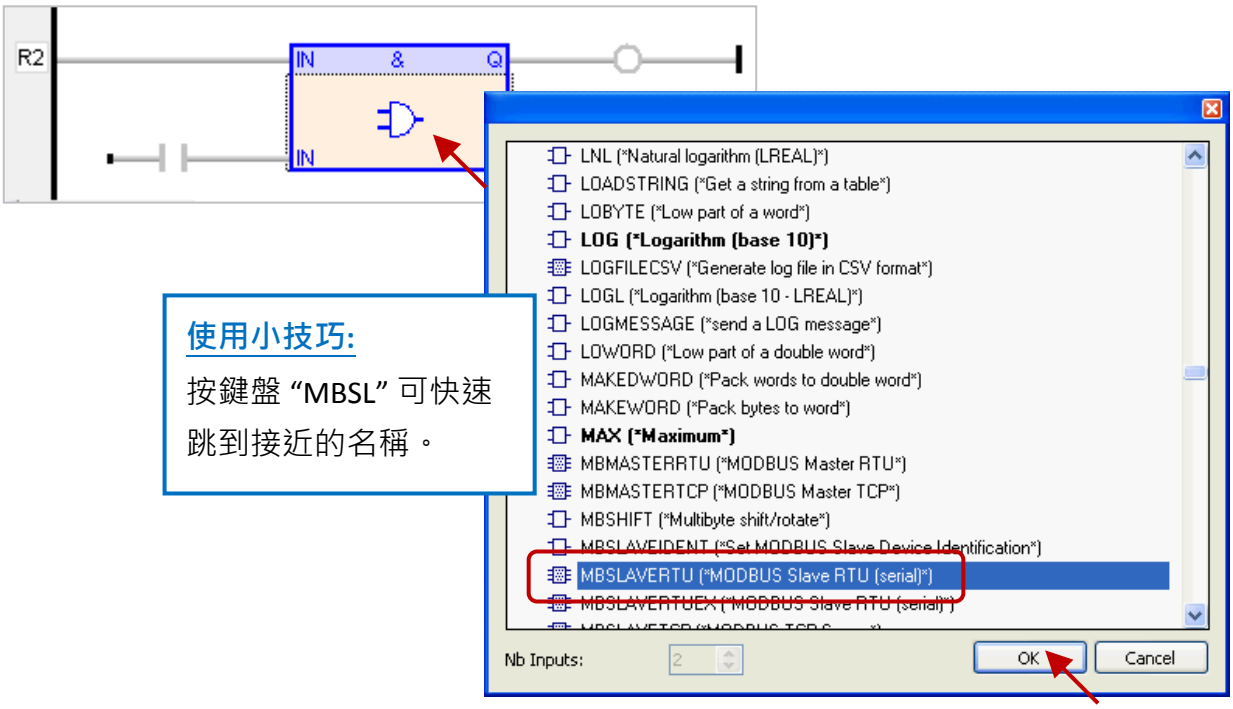
開始前，您必須先完成 3.1 節 中的內容、設定好需開放的 Modnus Slave 資料。啟用 Win-GRAF PAC 為 Modbus RTU Slave 的方式為，在程式中加入 “MBSLAVERTU” 或 “MBSLAVERTUEX” 功能方塊，請依照下列說明來完成設定。

加入 “MBSLAVERTU” 功能方塊

1. 在 “LD1” 視窗中，滑鼠點選需加入功能方塊的位置，再點選左側的 “Insert FB..” 按鈕。



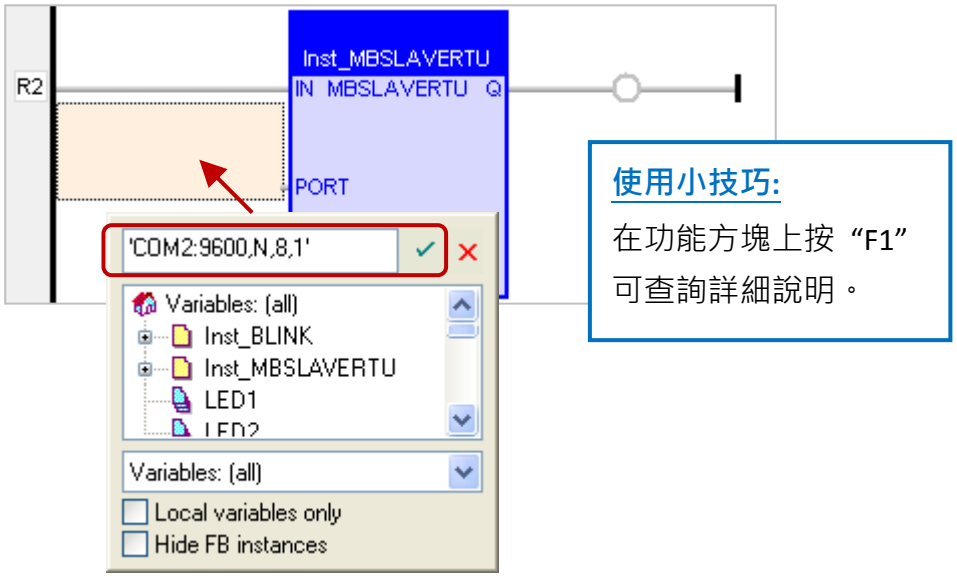
2. 滑鼠雙擊此功能方塊，並選取 “MBSLAVERTU” 再按 “OK”。



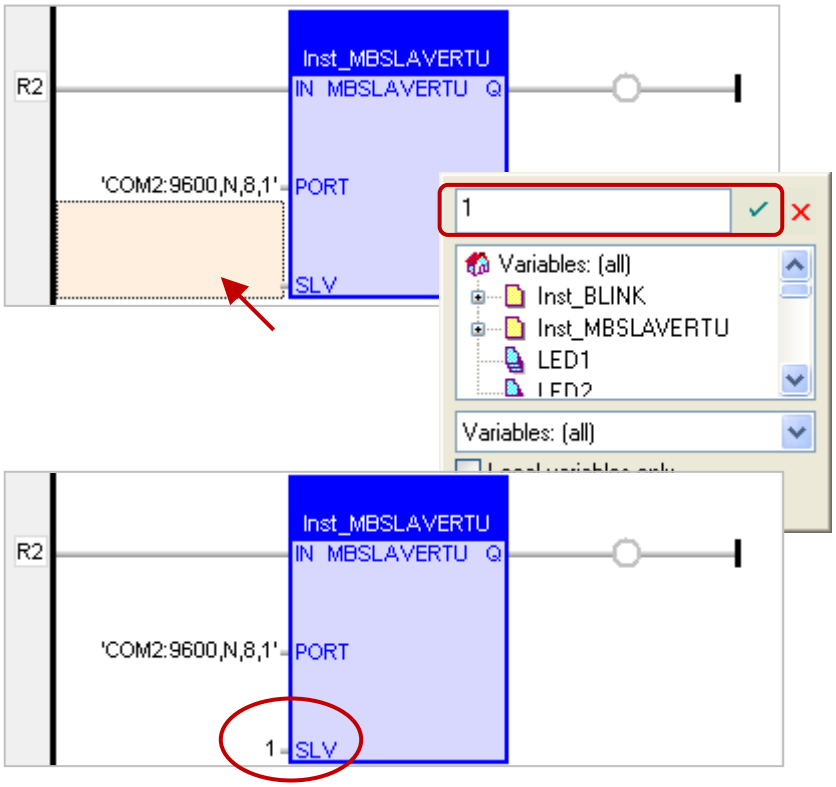
使用小技巧:

按鍵盤 “MBSL” 可快速跳到接近的名稱。

3. 於 “MBSLAVERTU” 功能方塊，滑鼠雙擊 “PORT” 左側的區域並輸入字串 'COM2:9600,N,8,1' (表示使用 Win-GRAF PAC 的 COM2 與 Modbus Master 進行溝通)，再點選 ✓ 完成設定。

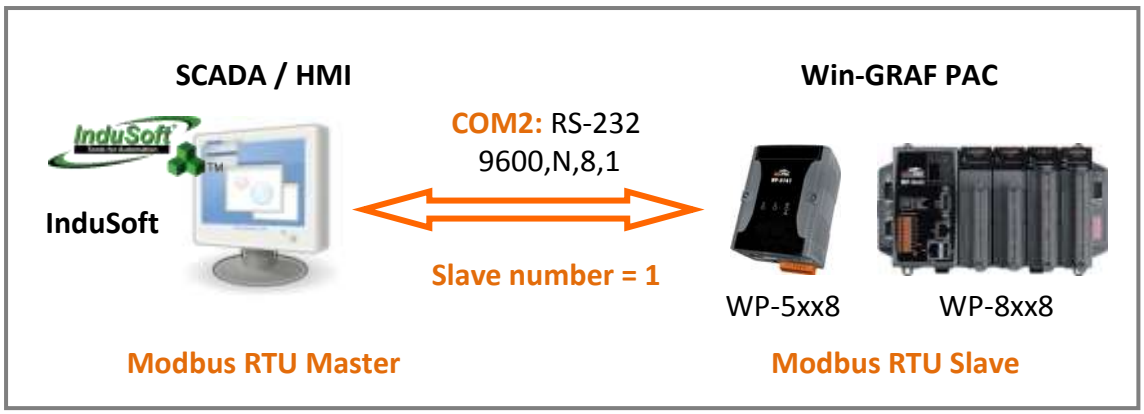


4. 滑鼠雙擊 “SLV” 左側的區域並輸入 “1” (即 3.1 節 - 步驟 3 設定的值)，再點選 ✓ 完成設定。



您已完成 “MBSLAVERTU” 功能方塊的設定，接下來請重新編譯並下載程式到 Win-GRAF PAC 中。(參考 2.3.4 節，2.3.5 節)

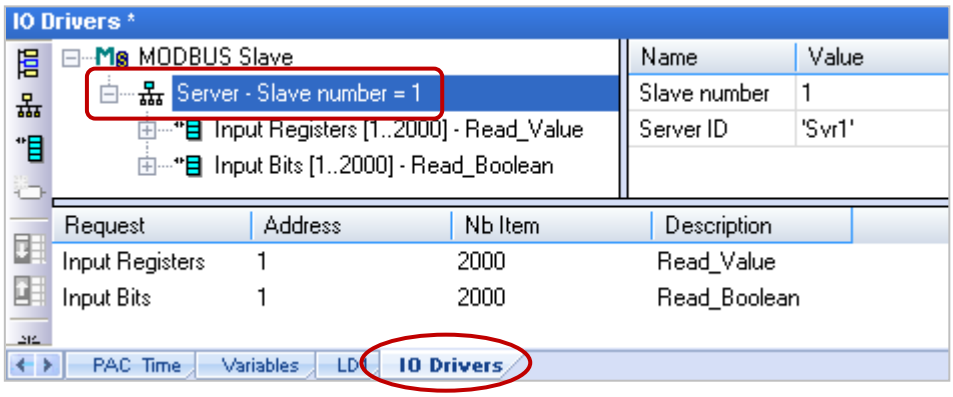
注意: 同一台 PAC 內可啟用多個 Modbus RTU Slave (建議不超過 16 Port)，方法為新增多個 “MBSLAVERTU” 功能方塊，並輸入不同的 Port 編號。



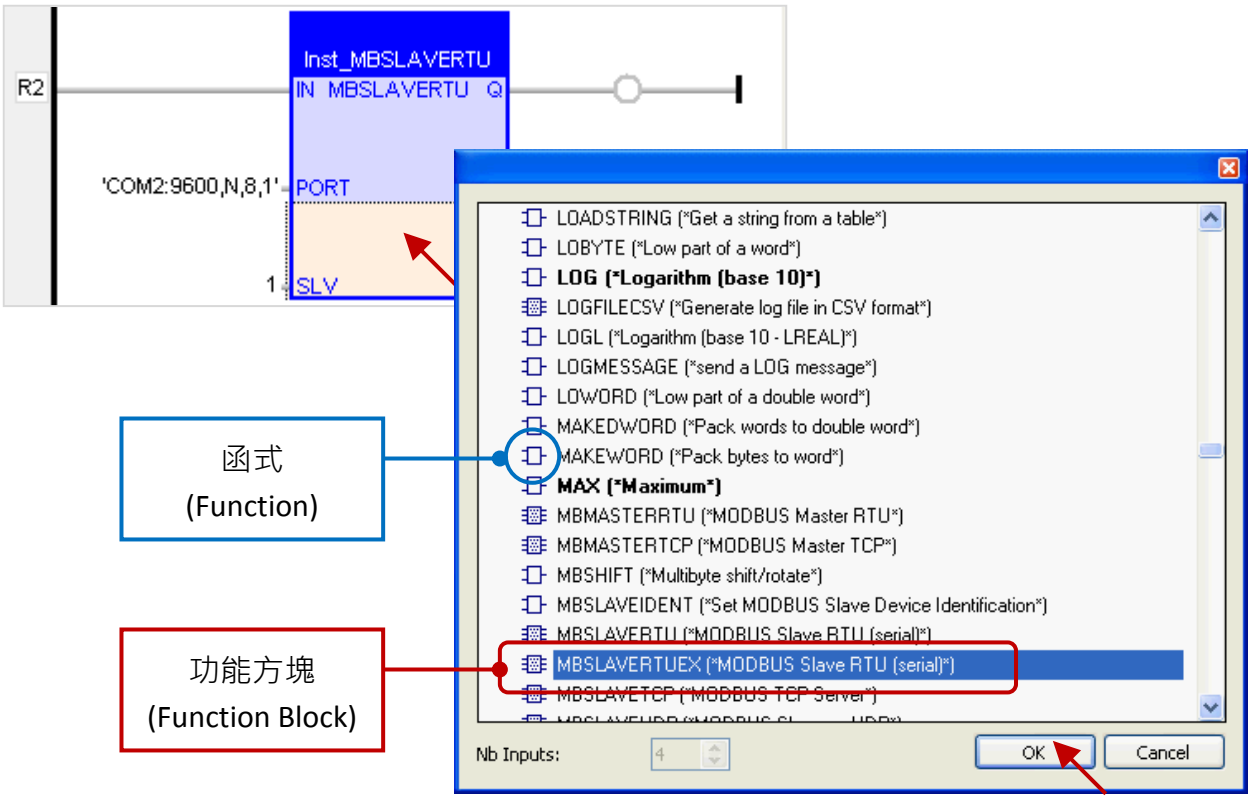
(您可參考 P1-1，來查看所有支援的 PAC 型號)

加入 “MBSLAVERTUEX” 功能方塊

若您在 “I/O Drivers” 中建立多個 “Server - …” 設定 (建議只使用一個)，則需使用 “MBSLAVERTUEX” 功能方塊。



1. 您可依照上述步驟 1 ~ 4 來建立一個 “MBSLAVERTUEX” 功能方塊。若您想變更先前已建立的功能方塊，可用滑鼠雙擊 “MBSLAVERTU” 並重新選取為 “MBSLAVERTUEX”，再點選 “OK”。

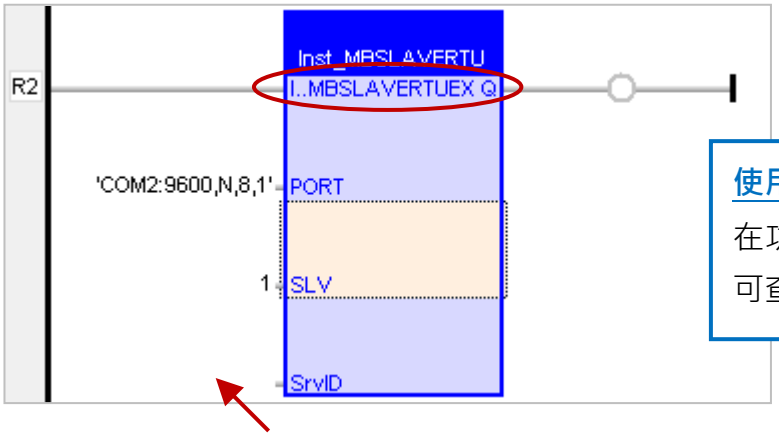


2. “MBSLAVERTUEX” 多了一個 “SrvID” 設定項目，滑鼠雙擊 “SrvID” 左側再填入所需的 “Server ID” (字串格式，例如 ‘Svr1’)。

注意:

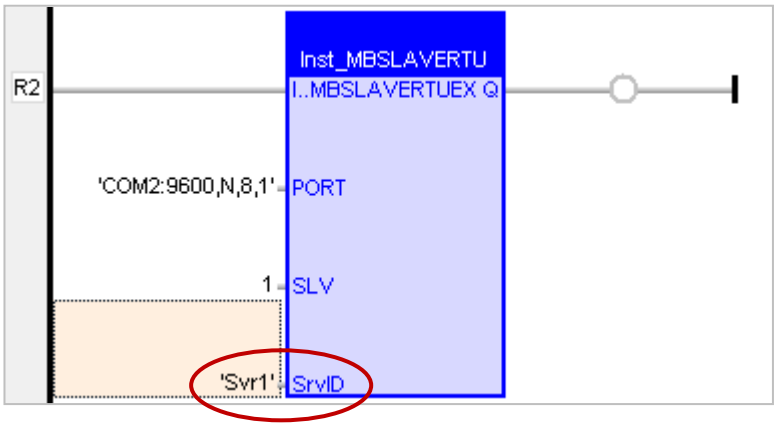
若使用 “MBSLAVERTU” 功能方塊，只會參考第一個 “Server – Slave number..” 設定。

若使用 “MBSLAVERTUEX” 功能方塊，則會參考具有相同 “Server ID” 的 “Server – Slave…” 設定。



使用小技巧:

在功能方塊上按 “F1” 可查詢詳細說明。



您已完成 “MBSLAVERTUEX” 功能方塊的設定，接下來請重新編譯程式並下載到 Win-GRAF PAC 中。(參考 2.3.4 節，2.3.5 節)

第 4 章　使用 “I/O Board” 功能

此章節將介紹如何使用 Win-GRAF Workbench 中的 “I/O Boards” 功能來對應到實體的 I/O 模組 或啟用其它功能。首先，請先了解每台 Win-GRAF PAC 的插槽編號 與 所支援的 PAC I/O 模組:

PAC 型號	插槽編號 (左至右)	插槽上支援的 I/O 模組
WP-8xx8	0 ~ 7	支援 I-8K 與 I-87K I/O 高卡模組。 (例如: I-8017HW 與 I-87055W) 但，不支援 I-8K 與 I-87K I/O 低卡模組。 (例如: I-8017H 與 I-87055)
XP-8x48 (*)	1 ~ 7	
XP-8x48-CE6 (*)		
XP-8x48-Atom-CE6		
VP-25W8	0 ~ 2	
VP-4138		
WP-5xx8	0	掌上型，可支援一片 XW-board (例如: XW-107)

(*) : XP-8048、XP-8048-CE6 為 0 槽的 PAC。(您可參考 P1-1，來查詢詳細的 PAC 型號)

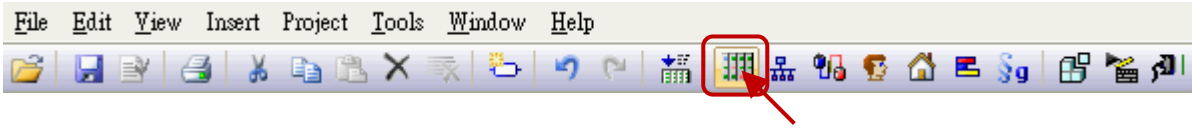
加入 “I/O board”

以下將 “I/O board” 稱為 “I/O 卡” 指的是軟體中的 I/O 功能 (例如: “i_8037_DO“)，
而 “I/O 模組” 指的是硬體設備 (例如: “I-8037W“)。

注意:

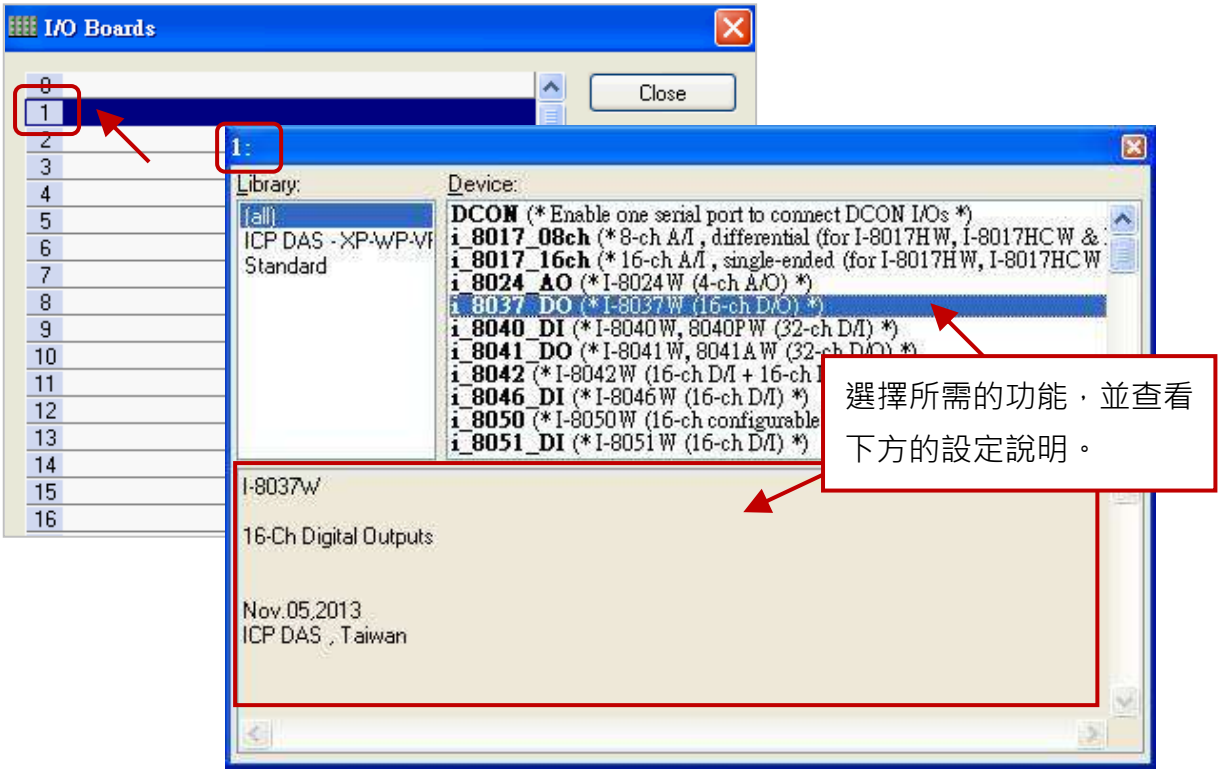
除了在 Win-GRAF 軟體設定 I/O 卡外，有些 I/O 模組還需進行硬體上的 Jumper 設定 (例如: Single-ended 與 Differential Jumper)。因此，使用前請先至各產品網頁上查詢設定資訊 或查看 I/O 模組的外殼說明 或 出貨附的文件。I-8K 與 I-87K 系列產品網頁:
http://www.icpdas.com/root/product/solutions/remote_io/rs-485/i-8k_i-87k/i-8k_i-87k_selection.html

1. 點選 Win-GRAF 工具列的 "Open I/Os" 按鈕來開啟 “I/O Boards” 視窗。

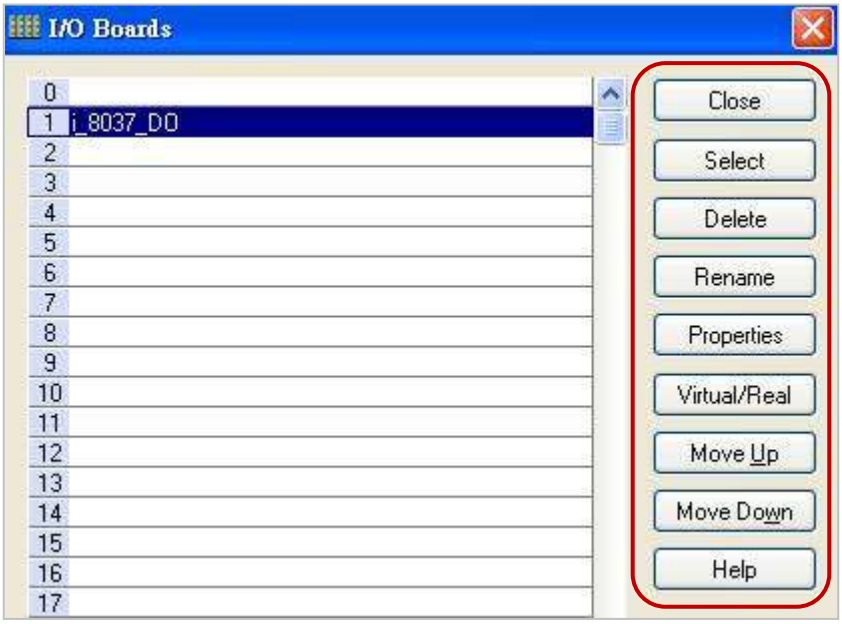


2. 如下圖，滑鼠雙擊 與 PAC I/O 模組相對應的 Slot 編號 (0 ~ 255)，再雙擊欲使用的 I/O 卡 (例如: i_8037_DO) 以完成選取。

注意: Slot0 ~ 7 是保留給 PAC I/O 模組，Slot8 (含) 以上供給其它用途使用。



按鈕說明: (點選以下按鈕可修改該 Slot 的設定)



“Close”:
關閉此視窗。

“Select”:
開啟 "Device" 功能選擇視窗。
(或按 “Enter” 開啟; “ESC” 離開)

“Delete”:
刪除此 I/O 卡。

“Rename”:
修改此 I/O 卡的名稱。

“Properties”: 查看此 I/O 卡的設定說明。

“Virtual/Real”: 將此 I/O 卡 切換為 虛擬 I/O (測試用) 或 真實 I/O。(快速鍵: Space)

“Move Up”: 將此 I/O 卡 往上移動。

“Move Down”: 將此 I/O 卡 往下移動。

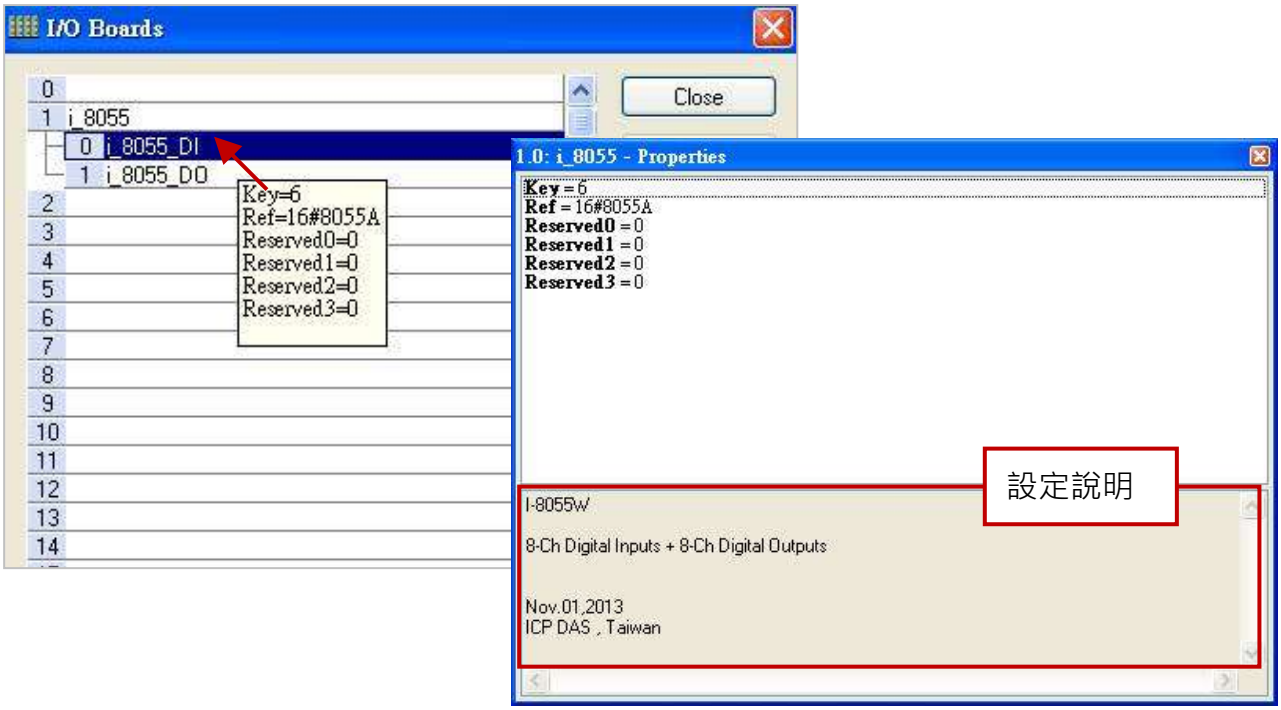
“Help”: 查看 I/O 卡使用說明。

4.1 DI/DO 卡

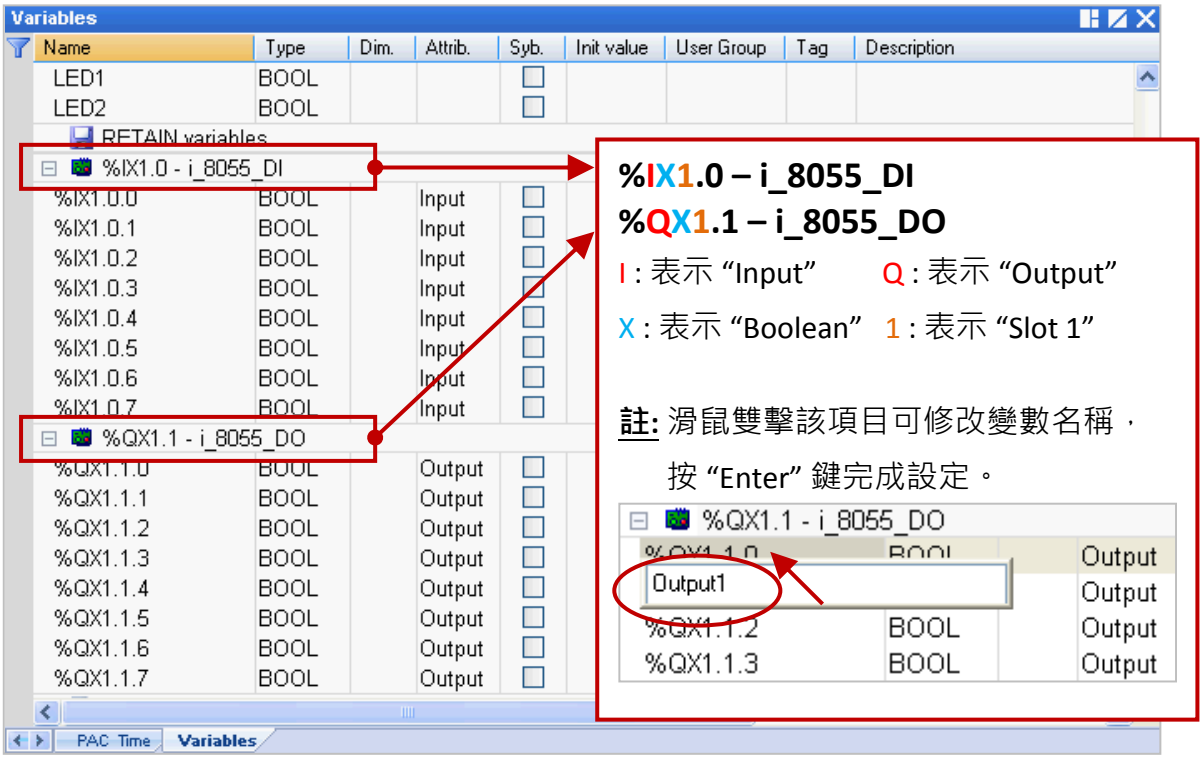
此章節以 I-8055W 為例，請參考 第四章 (P4-1) 來開啟並加入 I/O 卡。

1. 滑鼠雙擊 “i_8055_DI” (或 “i_8055_DO”) 可開啟 “Properties” 視窗。

   註: 滑鼠停留在 “i_8055_DI” (或 “i_8055_DO”) 也會顯示設定說明。



連上 “i_8055” I/O 卡後，會自動在 “Variables” 視窗中新增 8 個 Input 與 Output 變數供程式使用。

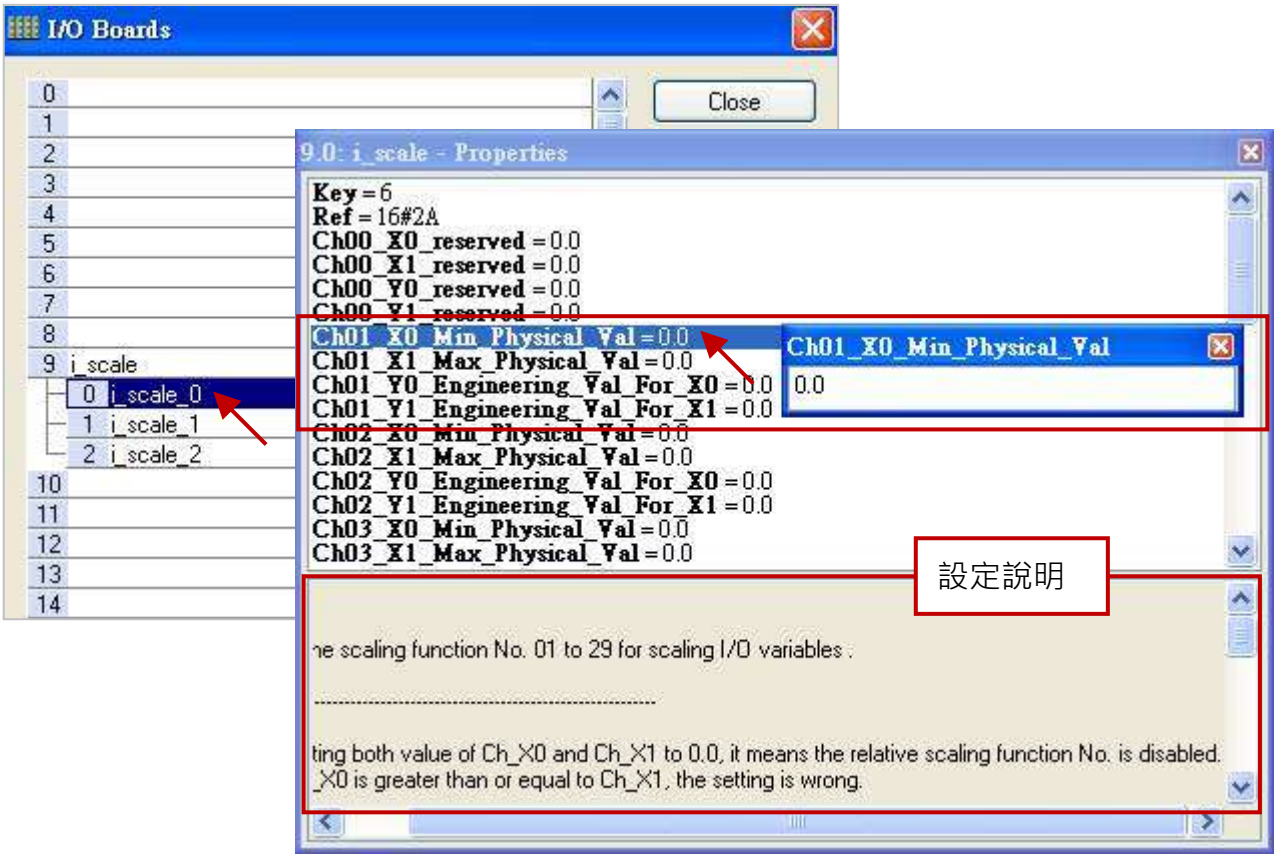


4.2 i_scale (轉換表功能)

“i_scale” 是用來設定轉換表功能，最多可設定 29 組，它適用於插在 Slot0 ~ 7 上的 AI/AO 模組，來進行數值轉換。請參考 第四章 (P4-1) 來開啟並加入此 I/O 卡。

1. 滑鼠雙擊 "i_scale_0" (或 "i_scale_1"、"i_scale_2") 開啟 “Properties” 視窗，來查看設定說明。

   注意: Slot0 ~ 7 是保留給 PAC I/O 模組，請使用在 Slot8 (含) 以上的位置。



參數說明: (Ch 表示 Ch01 ~ Ch29，Ch00 為保留值，無需設定。)

Ch_X0_Min_Physical_Val: AI (或 AO) board 的最小值 (X0)。

Ch_X1_Max_Physical_Val: AI (或 AO) board 的最大值 (X1)。

Ch_Y0_Engineering_Val_For_X0: X0 轉換後的工程應用值。

Ch_Y1_Engineering_Val_For_X1: X1 轉換後的工程應用值。

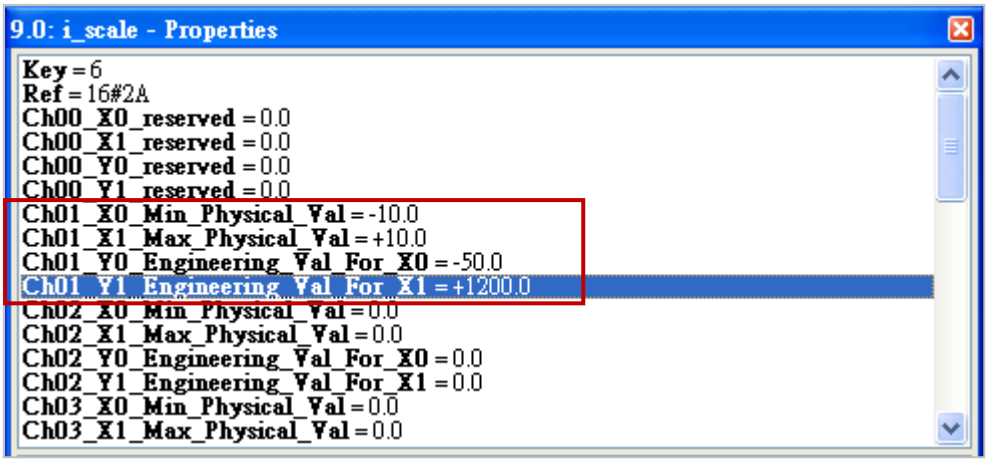
2. 滑鼠雙擊所需項目，來輸入數值並按 “Enter” 鍵完成設定。

   注意:

   1. 若 Ch_X0 與 Ch_X1 皆設定為 “0.0”，表示該對應編號不啟用轉換功能。
   2. 若 Ch_X0 的設定值 大於 或 等於 Ch_X1，表示設定錯誤。
   3. 若 Ch_Y0 的設定值 等於 Ch_Y1，表示設定錯誤。

   例如，若 AI board 的值為 “4 ~ 20 mA” 並想轉換為工程值 “0 ~ 10000”，請設定 Ch_X0 為 “4.0”、Ch_X1 為 “20.0”、Ch_Y0 為 “0.0” 與 Ch_Y1 為 “10000.0”。

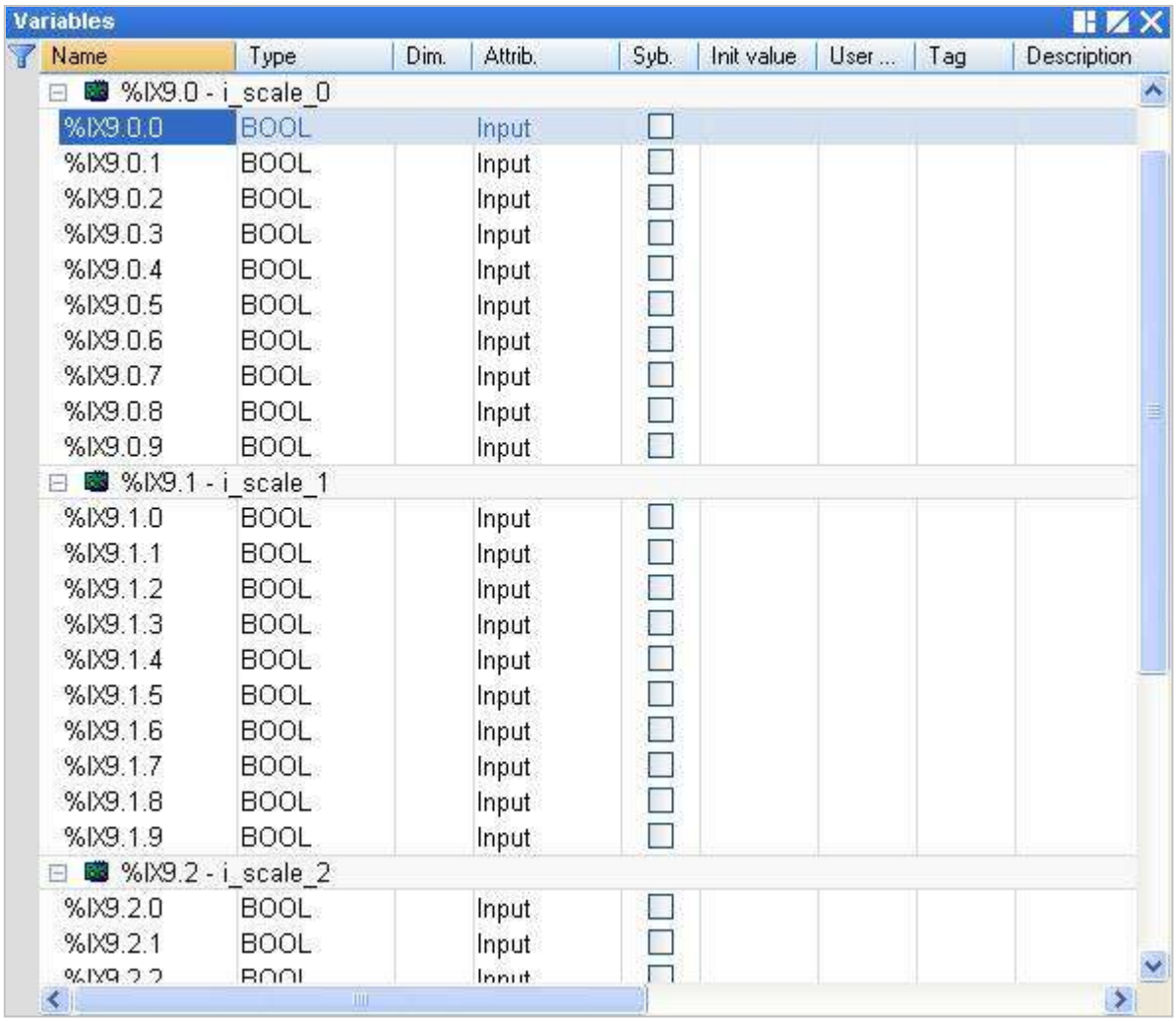
例如，若 AO board 的值為 “-10 ~ +10 V” 並想轉換為工程值 “-50 ~ 1200”，請設定 Ch_X0 為 “-10.0”、Ch_X1 為 “+10.0”、Ch_Y0 為 “-50.0” 與 Ch_Y1 為 “+1200.0”。



3. 在 “I/O Boards” 視窗內連上 “i_scale” 後，會自動在 “Variables” 視窗中新增 30 個布林變數，當 Win-GRAF 有連上 PAC 時，會顯示出轉換功能的狀態。

   True:　表示轉換功能 OK。

   FALSE:　表示未啟用轉換功能 或 設定錯誤。

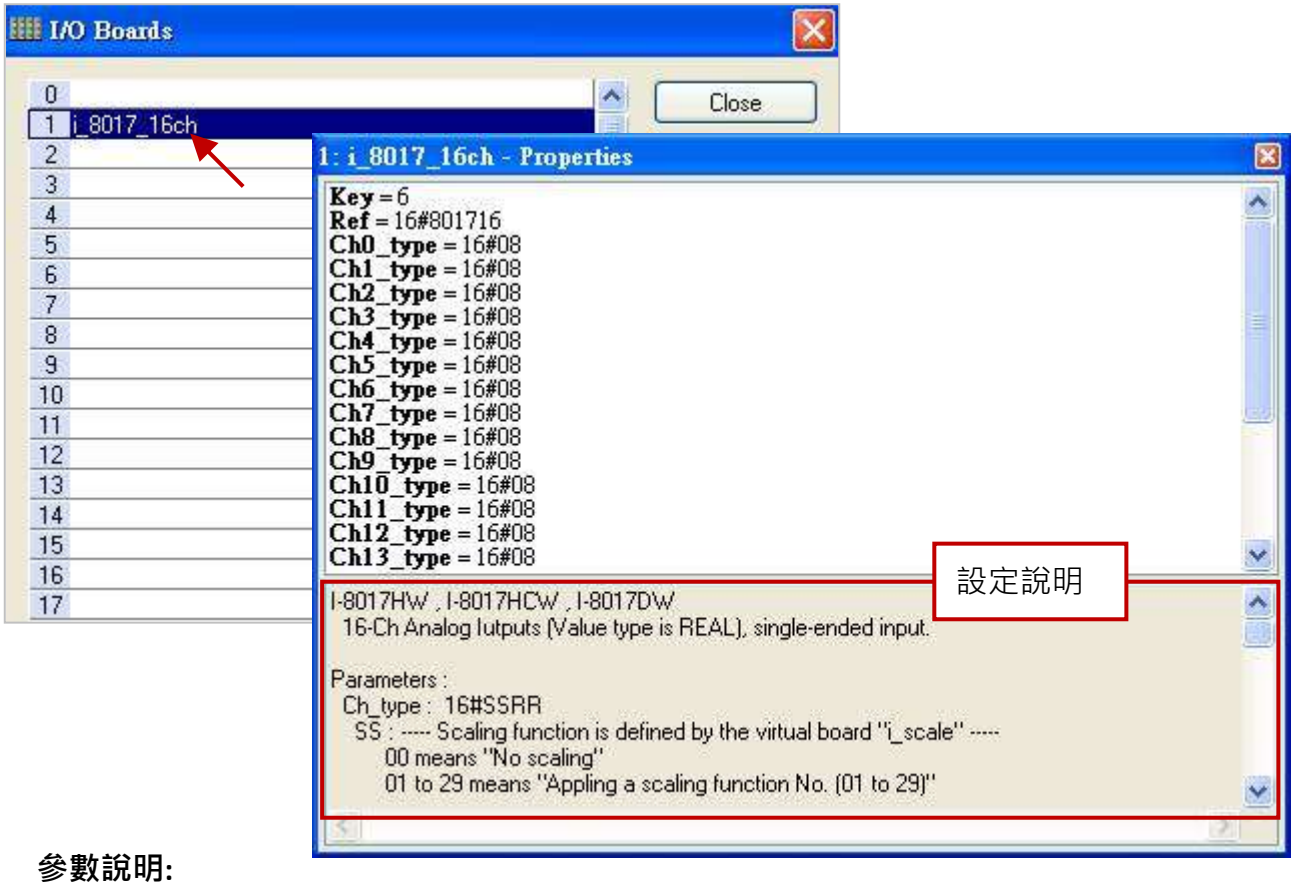


4.3 i_8017HW (8/16 通道 AI)

I-8017HW 可以是 8 通道差動式 (Differential) 或是 16 通道單端式 (Single-ended) 類比輸入模組 (資料型態為 “REAL”)。此處選擇 16 通道模組來做說明，請參考 第四章 (P4-1) 來開啟並加入 I/O 卡。

注意: 請先設定好 I-8017HW I/O 模組上的 Jumper (Differential / Single-ended)。

1. 滑鼠雙擊 "i_8017_16ch" 開啟 “Properties” 視窗，來查看設定說明。



參數說明:

Ch_type: 16#SSRR

SS : 此值定義在 "i_scale" 轉換表功能中。(請參考 4.2 節)

00 表示不使用轉換功能。

01 ~ 29 表示套用轉換表內編號 01 ~ 29 的設定。

若設定 SS 為其它值，將會採用預設值 “00”。

RR : 定義 AI 訊號型態/範圍。

05 表示 AI 模組的訊號型態/範圍為 “-2.5 ~ +2.5 V”。

06 表示 AI 模組的訊號型態/範圍為 “-20 ~ +20 mA”。

07 表示 AI 模組的訊號型態/範圍為 “-1.25 ~ +1.25 V”。

08 表示 AI 模組的訊號型態/範圍為 “-10 ~ +10 V”。

09 表示 AI 模組的訊號型態/範圍為 “-5 ~ +5 V”。

若設定 RR 為其它值，將會採用預設值 “08”。

2. 滑鼠雙擊所需項目，來輸入數值並按 “Enter” 鍵完成設定。

例如，16#08 表示 AI 模組的訊號型態/範圍為 “-10 ~ +10 V” 且不啟用轉換功能。
則通道值為 “5.67” 表示 AI 值為 “5.67 V”。

例如，16#209 表示 AI 模組的訊號型態/範圍為 “-5 ~ +5 V” 且啟用轉換功能 (即，"Ch02")。
則 AI 值會依據 "Ch02" 的設定，來轉換為一個工程值。

例如，16#1709 表示 AI 模組的訊號型態/範圍為 “-5 ~ +5 V” 且啟用轉換功能 (即，"Ch17")。
則 AI 值會依據 "Ch17" 的設定，來轉換為一個工程值。

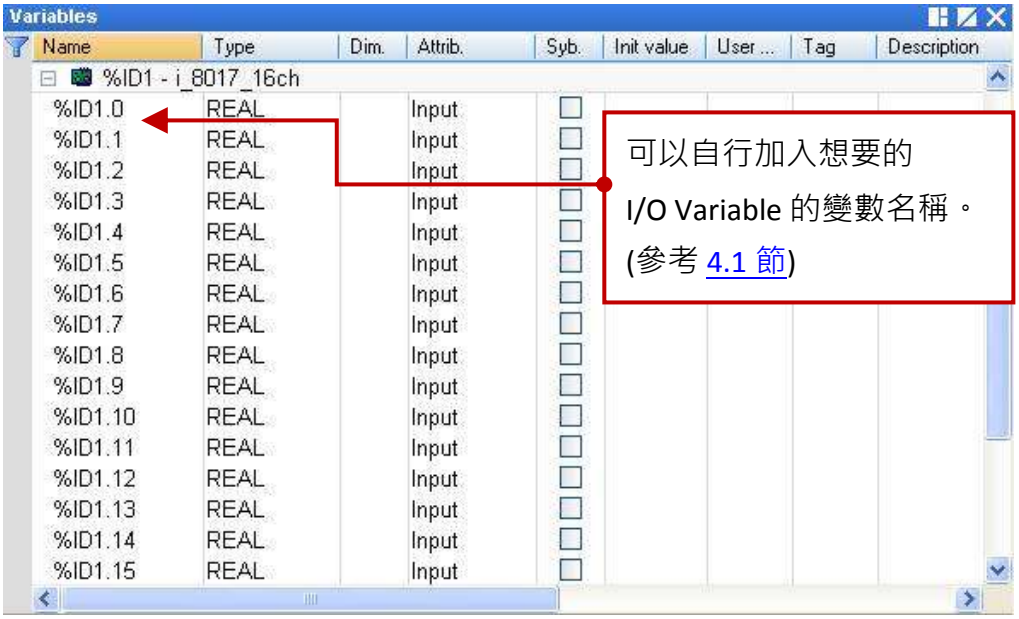
Noise_Filter_Max: 設定允許最大的上限值，高於該值會視為雜訊。
Filter 會過濾資料值，預設值為 "9999.9"。
例如，設定為 “7.9”，當資料值大於 7.9 V (或 7.9 mA) 該值會被過濾掉。

Noise_Filter_Min: 設定允許最小的下限值，低於該值會視為雜訊。
Filter 會過濾資料值，預設值為 "-9999.9"。
例如，設定為 “1.5”，當資料值大於 1.5 V (或 1.5 mA) 該值會被過濾掉。

註: 若設定值 Noise_Filter_Min >= Noise_Filter_Max，Filter 將不啟用。
若設定值 Noise_Filter_Min < “-1000” 與 Noise_Filter_Max > “1000”，Filter 將不啟用。

Sample_Number: 設定取樣多少次以後，才取其平均值並更新 AI 模組的資料值，預設值為 "1” (範圍: 1 ~ 500)。若設定為 “500” 表示最慢速的取樣率，取樣 500 次後，才會計算出平均值並更新 1 次。但此值越大，輸入波形越平滑。

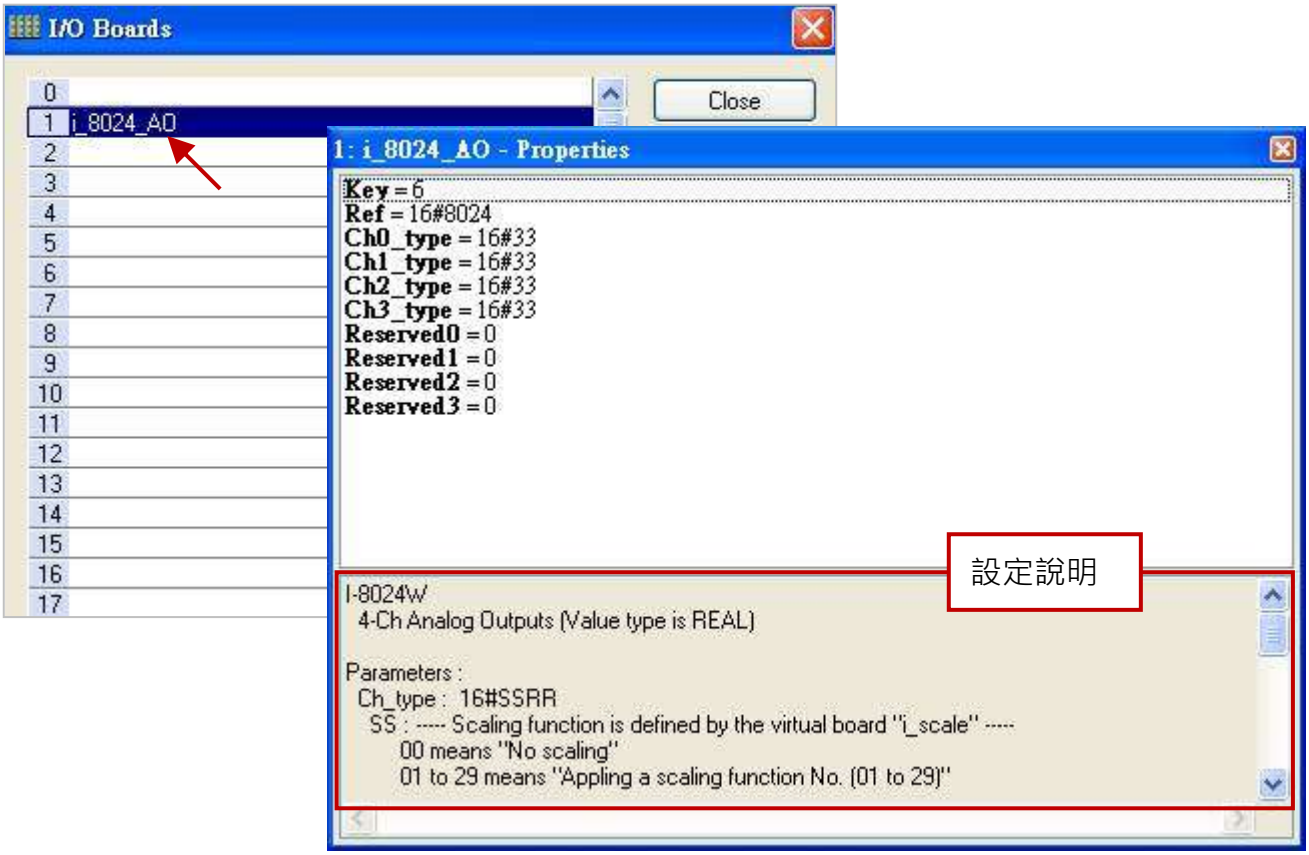
3. 在 “I/O Boards” 視窗內連上 “i_8017_16ch” 後，會自動在 “Variables” 視窗中新增 16 個 "REAL” 輸入變數，可供程式使用。



4.4 i_8024 (4 通道 AO)

I-8024W 是一款 4 通道類比輸出模組 (資料型態為 “REAL”)，可輸出 +/- 10 V 或 0 ~ +20 mA 的訊號。請參考 第四章 (P4-1) 來開啟並加入此 I/O 卡。

1. 滑鼠雙擊 "i_8024_AO" 開啟 “Properties” 視窗，來查看設定說明。



參數說明:

Ch_type: 16#SSRR

SS : 此值定義在 "i_scale" 轉換表功能中。(請參考 4.2 節)

00 表示不使用轉換功能。

01 ~ 29 表示套用轉換表，編號 01 ~ 29 的設定。

若設定 SS 為其它值，將會採用預設值 “00”。

RR : 定義 AO 訊號型態/範圍。

30 表示 AO 模組的訊號型態/範圍為 “0 ~ +20 mA”。

33 表示 AO 模組的訊號型態/範圍為 “-10 ~ +10 V”。

若設定 RR 為其它值，將會採用預設值 “33”。

2. 滑鼠雙擊所需項目，來輸入數值並按 “Enter” 鍵完成設定。

例如，16#33 表示 AO 模組的訊號型態/範圍為 “-10 ~ +10 V” 且不啟用轉換功能。

若通道值為 “5.67” 表示 AO 值為 “5.67 V”。

若通道值為 “-3.752” 表示 AO 值為 “-3.752 V”。

例如，16#133 表示 AO 模組的訊號型態/範圍為 “-10 ~ +10 V” 且啟用轉換功能 (即，"Ch01")。
則 AO 值會依據 "Ch01" 的設定，來轉換為一個工程值。

例如，16#30 表示 AO 模組的訊號型態/範圍為 “0 ~ 20 mA” 且不啟用轉換功能。
若通道值為 “12.5” 表示 AO 值為 “12.5 mA”; 值為 “6.27” 表示 AO 值為 “6.27 mA”。

例如，16#1730 表示 AO 模組的訊號型態/範圍為 “0 ~ 20 mA” 且啟用轉換功能 (即，"Ch17")。
則 AO 值會依據 "Ch017" 的設定，來轉換為一個工程值。

3. 在 “I/O Boards” 視窗內連上 “i_8024_AO” 後，會自動在 “Variables” 視窗中新增 4 個 "REAL” 輸出變數，可供程式使用。

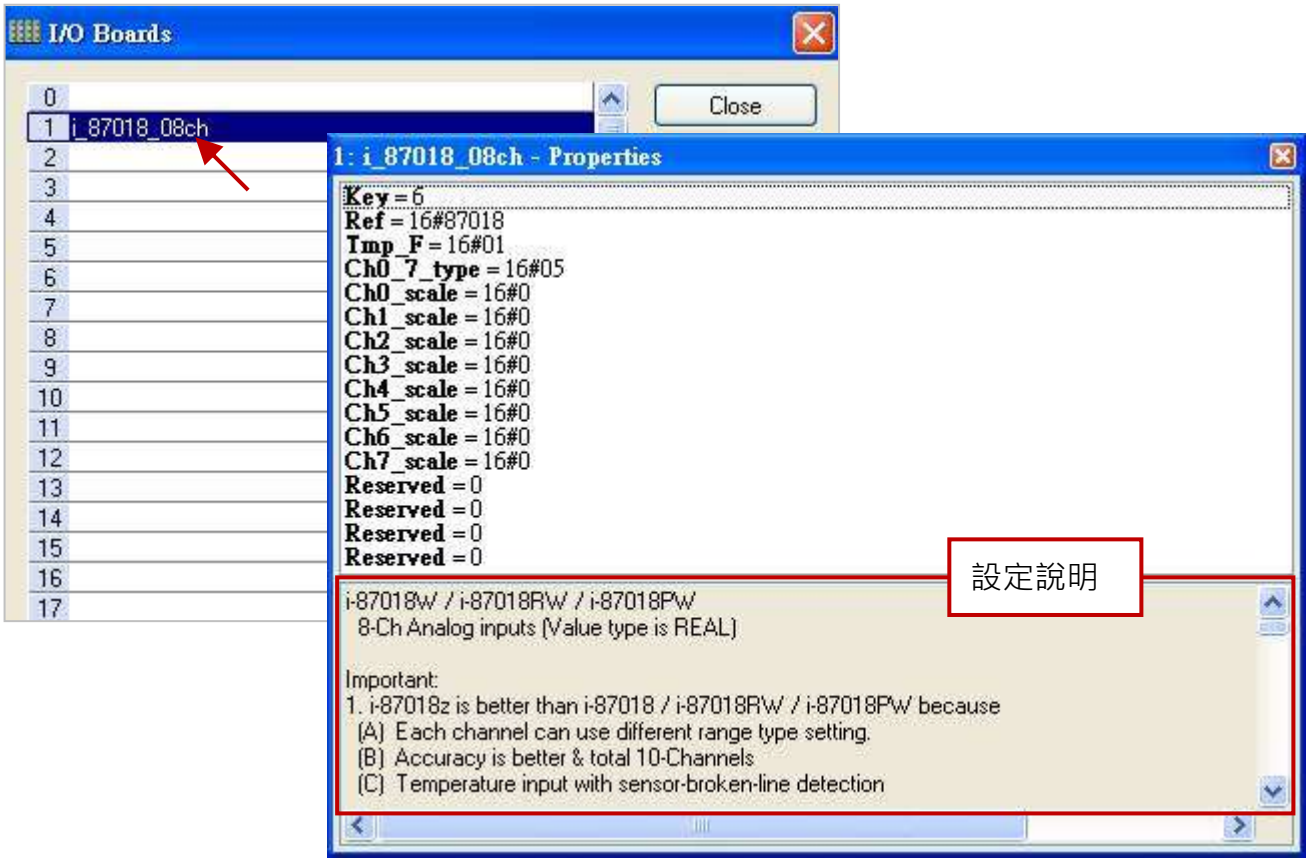
4.5 i_87018W (8 通道 AI)

I-87018W 是一款 8 通道類比輸入模組 (資料型態為 “REAL”)，可用來量測溫度 (Thermocouple)、電流 (-20 mA ~ +20 mA) 與 電壓輸入 (+/- 15 mV, +/- 50 mV, +/- 100 mV, +/- 500 mV, +/- 1 V, +/- 2.5 V)，請參考 第四章 (P4-1) 來開啟並加入 I/O 卡。

重要說明:

- I-87018ZW 優於 I-87018W / I-87018RW / I-87018PW，因為
  (A) 每個通道可設定為不同的輸入型態。
  (B) 量測準確度較高，且有 10 個通道。
  (C) 量測溫度輸入時，有斷線偵測功能。
  請參考產品網頁: http://www.icpdas.com/products/Remote_IO/i-87k/i-87018z.htm
- I-87018W 未支援斷線偵測功能。

1. 滑鼠雙擊 "i_87018_08ch" 開啟 “Properties” 視窗，來查看設定說明。



參數說明:

Tmp_F: 16#FF

FF : 溫度格式，僅適用在 AI 模組為溫度 (Thermocouple) 類型時。

01 表示單位為攝氏溫度。

02 表示單位為華氏溫度。

Ch0_7_type: 16#RR

RR : 定義 AI 訊號型態/範圍。

一般範圍: (適用於 “mA" 或 “V”)

設定值	AI 模組的訊號型態/範圍
00	-0.015 ~ +0.015 V
01	-0.05 ~ +0.05 V
02	-0.1 ~ +0.1 V
03	-0.5 ~ +0.5 V
0□	-1 ~ +1 V
05	-2.5 ~ +2.5 V
06	20 ~ +20 mA

當 I-87018W 與 I-87018RW 用來量測電流時，需外接一個 125 Ohm 電阻。

Thermocouple 範圍: (適用於溫度量測)

設定值	Ty□e	AI 模組的訊號型態/範圍
0E	J	-210 ~ +760 °C
0F	K	-270 ~ +1372 °C
10	T	-270 ~ +400 °C
11	E	-270 to +1000 °C
12	R	0 ~ +1768 °C
13	S	0 ~ +1768 °C
14	B	0 ~ +1820 °C
15	N	0 ~ +2320 °C
17	L	-200 ~ +800 °C
18	M	-200 ~ +100 °C
19	$L_{DIN43710}$	-200 ~ +900 °C

若設定 RR 為其它值，將會採用預設值 “05”。

2. 滑鼠雙擊所需項目，來輸入數值並按 “Enter” 鍵完成設定。

例如，16#05 表示 AI 模組的訊號型態/範圍為 “-2.5 ~ +2.5 V” 且不啟用轉換功能。

若通道值為 “1.28” 表示 AI 值為 1.28 V”。

若通道值為 “-0.752” 表示 AI 值為 “-0.752 V”。

例如，若 "Ch0_7_type" 設定為 16#0F 且 "Tmp_F" 設定為 16#01，表示

AI 模組的訊號型態/範圍為 “-270 ~ +1372 °C”，若通道值為 “25.75” 表示 “25.75 °C”。

例如，若 "Ch0_7_type" 設定為 16#10 且 "Tmp_F" 設定為 16#02，表示

AI 模組的訊號型態/範圍為 “-454 ~ +752 °F”，若通道值為 “25.75” 表示 “25.75 °F”。

注意: 若使用具有斷線偵測功能的溫度模組 (例如: I-87018ZW)，溫度值大於 "9000.0" 表示，

1. 溫度感測器可能斷線。
2. 溫度感測器可能損毀。
3. DCON 模組的設定 與 溫度感測器不符。
4. 感測器量到錯誤的電阻值。

Ch0_scale ~ Ch7_scale: 16#SS

SS : 此值定義在 "i_scale" 轉換表功能中。(請參考 4.2 節)

00　　表示不使用轉換功能。

01 ~ 29　表示套用轉換表，編號 01 ~ 29 的設定。

若設定 SS 為其它值，將會採用預設值 “00”。

例如，16#17 表示 AI 值會依據 "Ch17" 的設定，來轉換為一個工程值。

3. 在 “I/O Boards” 視窗內連上 “i_87018_08ch” 後，會自動在 “Variables” 視窗中新增 8 個 "REAL" 輸入變數，可供程式使用。

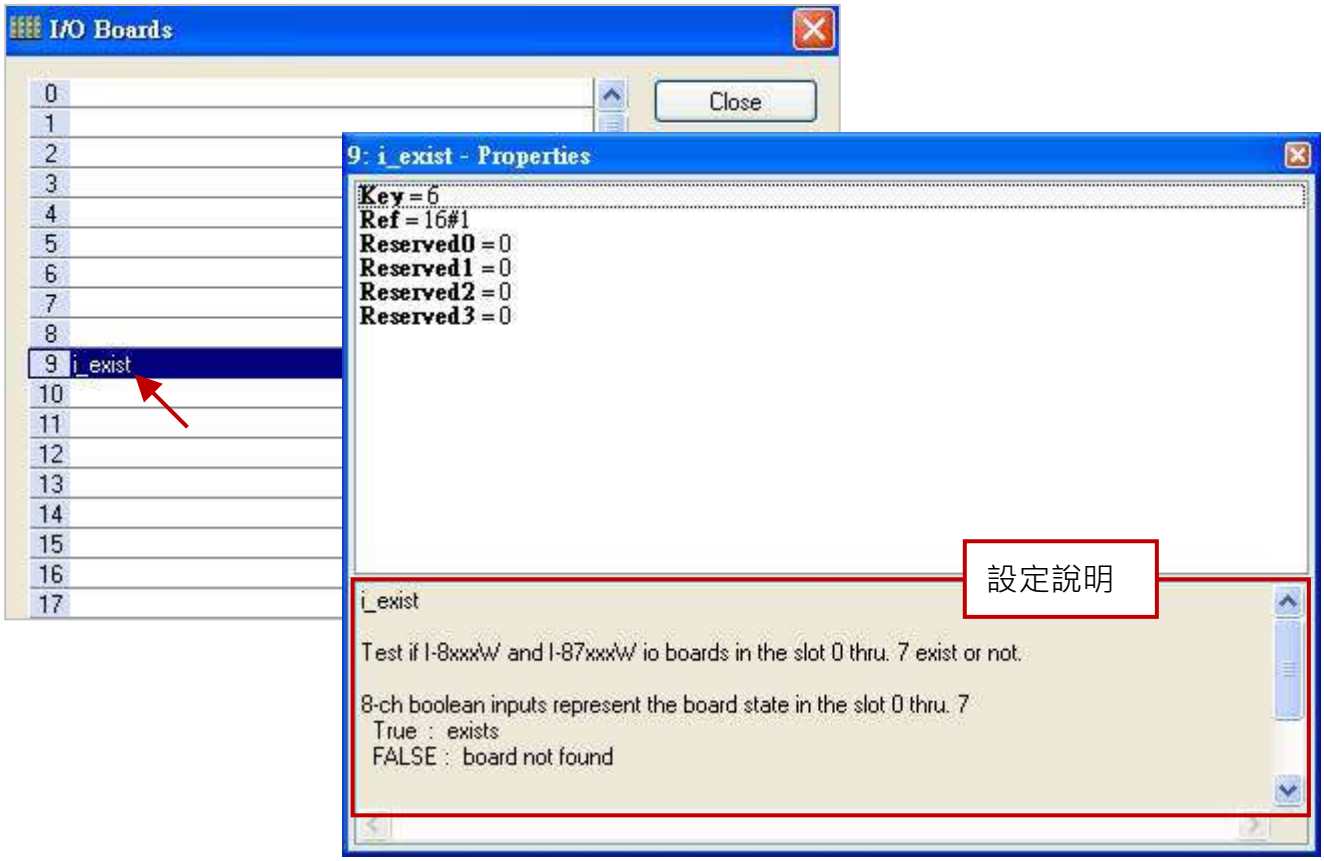
4.6 i_exist (測試 I/O 模組是否存在?)

“i_exist” 是用來測試 PAC 上 (Slot0 ~ 7) 的 I-8K 與 I-87K I/O 模組是否存在?

請參考 第四章 (P4-1) 來開啟並加入此 I/O 卡。

1. 滑鼠雙擊 "i_exist" 開啟 “Properties” 視窗，來查看設定說明。

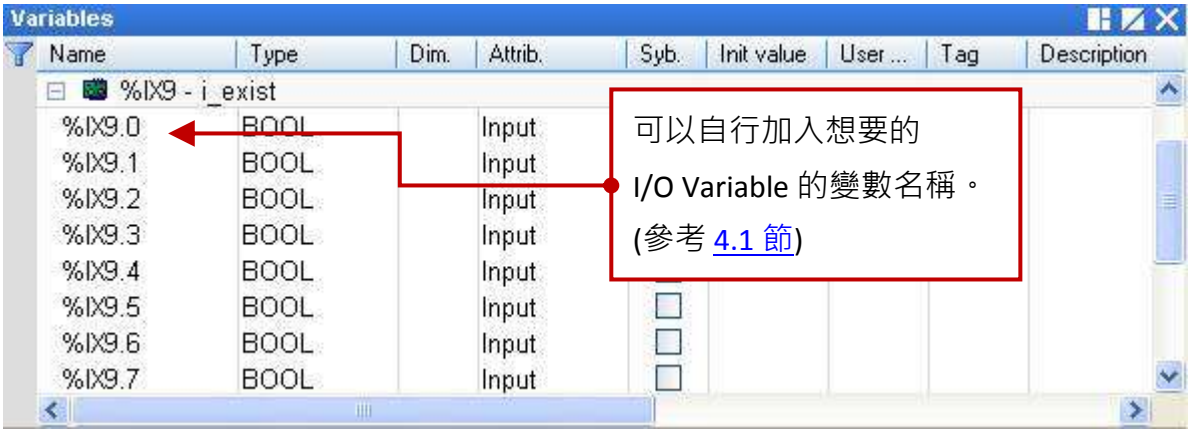
   注意: Slot0 ~ 7 是保留給 PAC I/O 模組，請使用在 Slot8 (含) 以上的位置。



2. 在 “I/O Boards” 視窗內連上 “i_exist” 後，會自動在 “Variables” 視窗中新增 8 個 "BOOL” 輸入變數，當 Win-GRAF 有連上 PAC 時，會顯示出 Slot 0 ~ 7 上 I/O 模組的狀態。

   TRUE: 表示 I/O 存在。

   FALSE: 表示找不到 I/O。

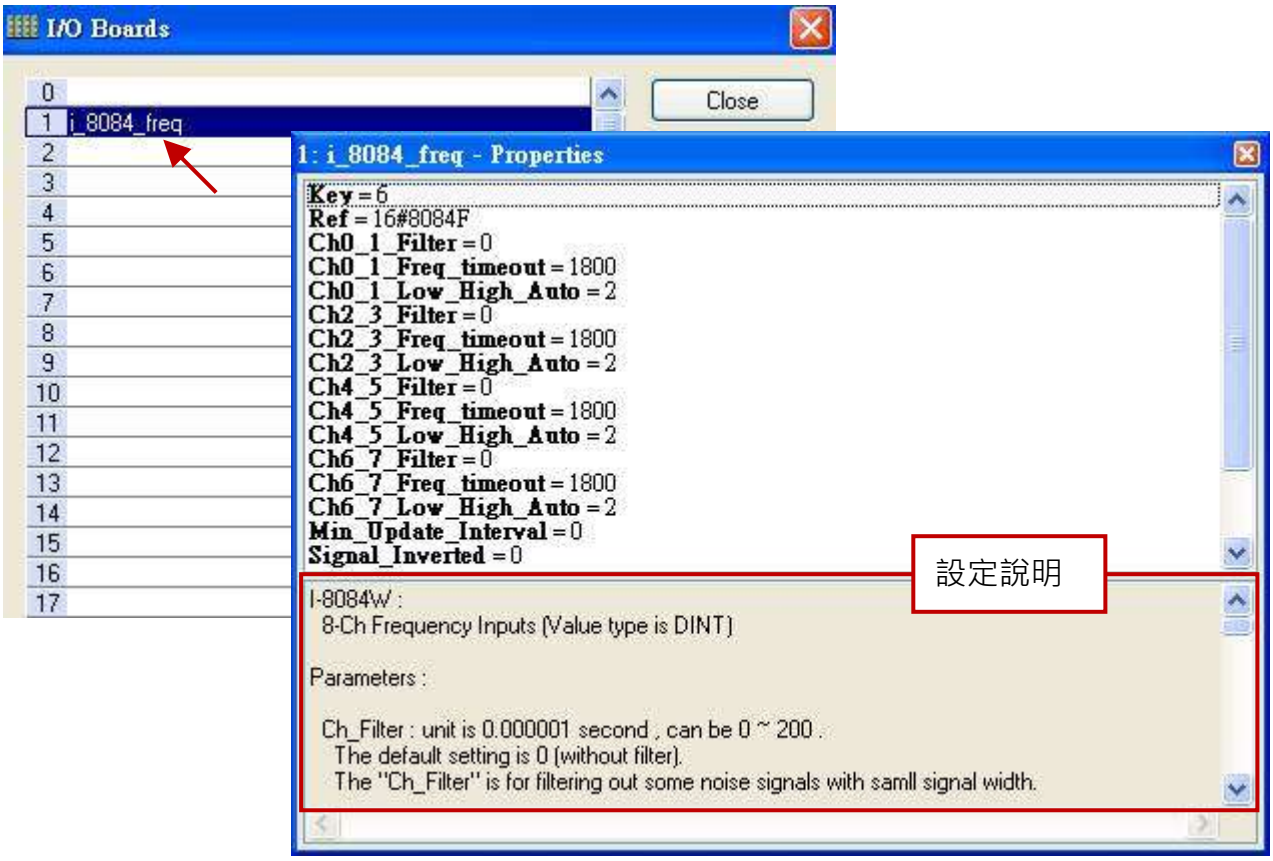


4.7 i_8084 (頻率量測, UP/Down 計數器, UP 計數器)

I-8084W 是一款高速的 4/8 通道頻率/計數器模組 (資料型態為 “DINT”)，可用於頻率量測、UP/Down 計數器、UP 計數器，以下將介紹此三種模式。請參考 第四章 (P4-1) 來開啟並加入此 I/O 卡。

4.7.1 i_8084_freq (頻率量測)

1. 滑鼠雙擊 "i_8084_freq" 開啟 “Properties” 視窗，來查看設定說明。



參數說明:

Ch_Filter: 單位為微秒 (μs = 0.000001 秒)，可設為 “0” ~ “200”，預設值為 “0” (不過濾)。"Ch_Filter" 是用來過濾掉一些很小的雜訊寬度。建議設為 “0”，若無雜訊考量或採用即時 (Real-time) 量測。下表為建議的設定:

最大輸入頻率 (Hz)	建議的設定值
1K	200
2K	100
5K	40
10K	20
20K	10
100K	2
450K	1
450K	0 (未過濾)

Ch_Freq_Timeout:

單位為毫秒 (ms = 0.001 秒)，可設為 “20” ~ “1800”，若設定為其它值，將會採用 “1800”。

若在 "Ch_Freq_Timeout" 間隔時間中，I-8084W 沒有任何訊號波形輸入，則相關通道的頻率值將設定為 “0”。

例如，設定此值為 100 ms 且輸入頻率為 500 Hz (表示約 2 ms 產生一個訊號波形)。

當輸入頻率下降到 9 Hz (表示約 111 ms 產生一個訊號波形)，此 “111” ms 已經超過了設定值 “100” ms (即 “Freq_Timeout”)。因為在 100 ms 的間隔時間內，沒有訊號波形產生，因此，頻率值將會設定為“0”。

設定為 20 ms 時，將無法檢測 50 Hz 以下的頻率值 (變為 “0”)。

設定為 100 ms 時，將無法檢測 10 Hz 以下的頻率值 (變為 “0”)。

設定為 1800 ms 時，可檢測 0 Hz、1 Hz ~ 450 KHz 的頻率值。

Ch_Low_High_Auto: (建議設定為 “2 : 自動模式”)

0 表示低頻模式。

1 表示高頻模式。

2 表示自動切換為 低頻 或 高頻模式。

若設定為其它值，將會採用預設值 “2 : 自動模式”。

模式 2 會自動切換頻率模式，當輸入頻率大於 3500 Hz 會切換到高頻模式，當輸入頻率小於 1000 Hz 會切換到低頻模式。

若輸入頻率通常小於 1000 Hz 或 常不正確，請勿設定為 1 (高頻模式)。

若輸入頻率通常大於 3500 Hz，請勿設定為 0 (低頻模式)。

Min_Update_Interval:

單位為毫秒 (ms = 0.001 秒)，可設為 "0” 或 “20” ~ “1000”。

預設值 "0” 表示 “每達到 PAC Cycle 時間，更新頻率一次”。

其餘設定值表示 “每達到此間隔時間，更新頻率一次”。

頻率的更新時間也取決於 Win-GRAF PAC 的 Cycle 時間，若 PAC Cycle 時間為 200 ms 而 "Min_Update_Interval" 設定小於 200，則實際的頻率更新時間將為 200 ms。

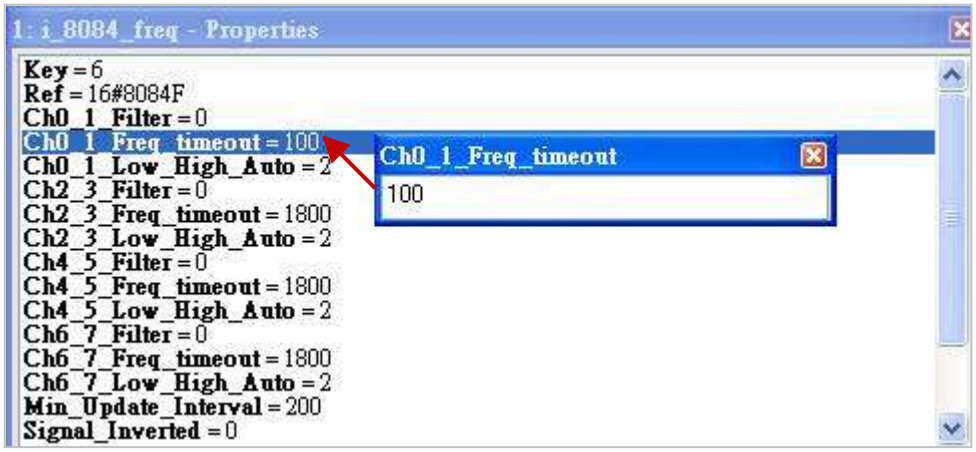
設定 "Min_Update_Interval" 為較大的值，將得到較平滑的頻率波形值，但頻率值更新緩慢。

Signal_Inverted:

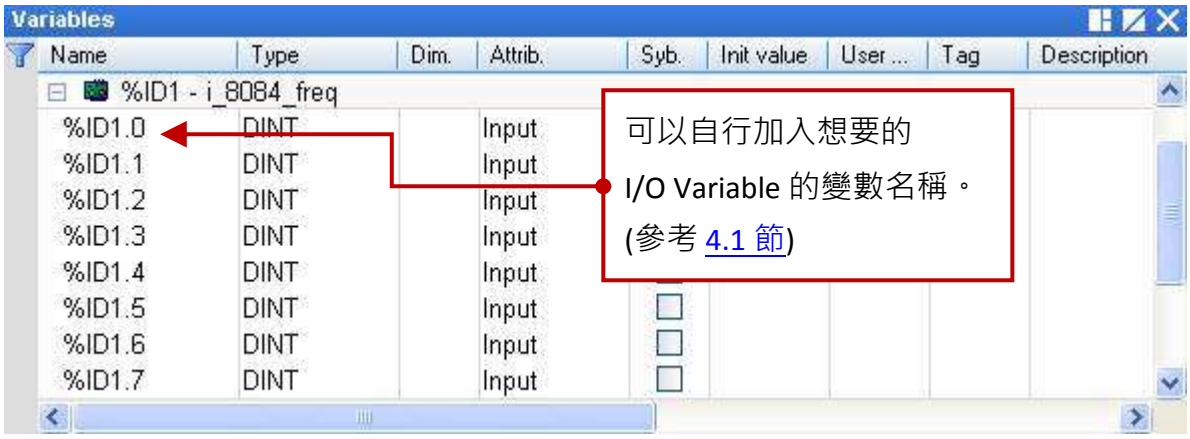
0: 輸入訊號正常 (無反相)

1: 輸入訊號為反相 (表示電壓 HIGH 會視為 LOW 處理，而電壓 LOW 會視為 HIGH 處理)

2. 滑鼠雙擊所需項目，來輸入數值並按 “Enter” 鍵完成設定。



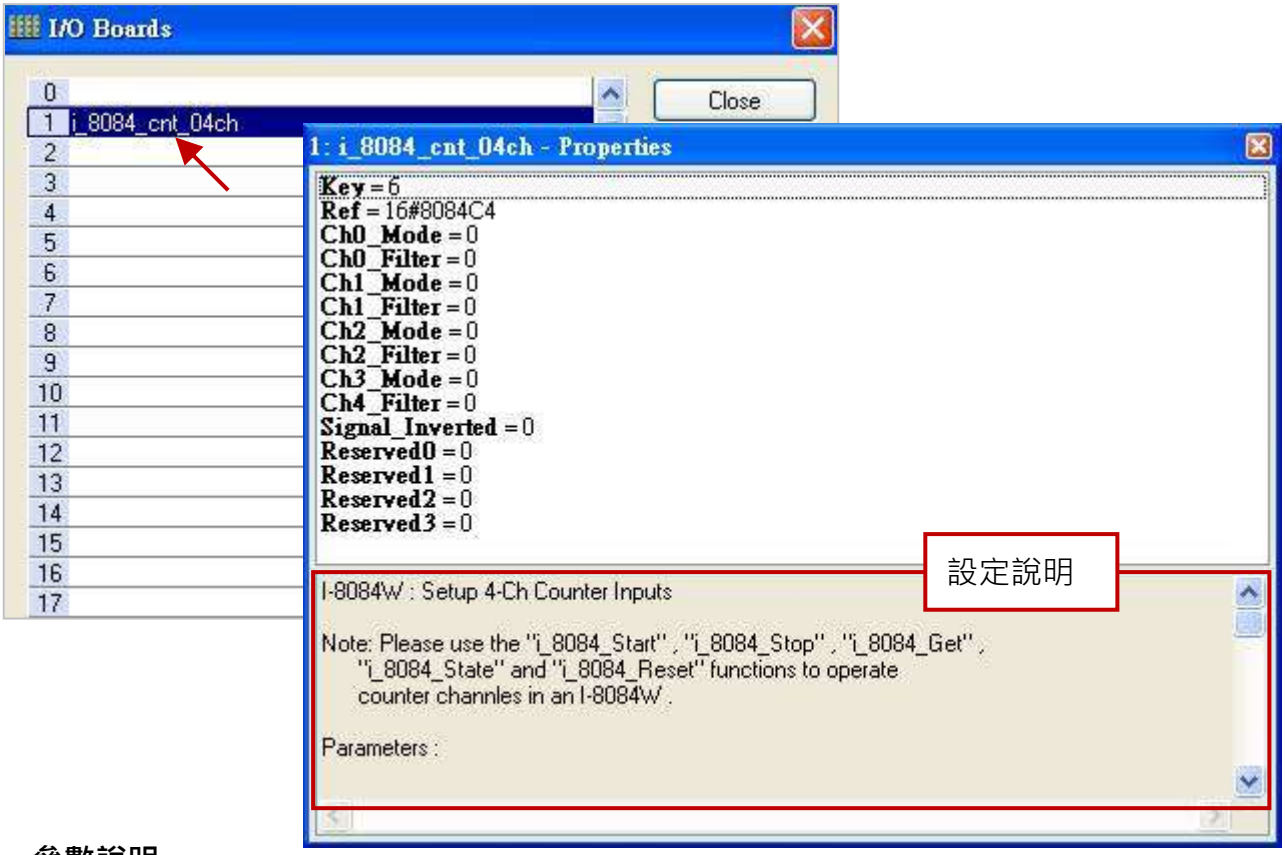
3. 在 “I/O Boards” 視窗內連上 “i_8084_freq” 後，會自動在 “Variables” 視窗中新增 8 個 "DINT” 輸入變數，可用來顯示各通道的頻率值。



4.7.2 i_8084_cnt_ch04 (4 通道 UP/Down 計數器)

註: 請使用 Win-GRAF Workbench 中的 "COUNTER_START"、"COUNTER_STOP"、"COUNTER _GET"、"COUNTER_STATE" 與 "COUNTER_RESET" 函式 (Function)，來操作 I-8084W 的計數通道。

1. 滑鼠雙擊 "i_8084_cnt_ch04" 開啟 “Properties” 視窗，來查看設定說明。



參數說明:

Ch_Mode: 輸入模式可設定為 “0”、“1” 與 “4”，若設定為其它值，將會採用 “0”。

0: Pulse/DIR 模式。

1: UP/DOWN 模式。

4: A/B phase (Quard.) 模式。

Ch_Filter: 單位為微秒 (μs = 0.000001 秒)，可設為 “0” ~ “200”，預設值為 “0” (不過濾)。"Ch_Filter" 是用來過濾掉一些很小的雜訊寬度。建議設為 “0”，若無雜訊考量或採用即時 (Real-time) 量測。下表為建議的設定:

最大輸入頻率 (Hz)	建議的設定值
1K	200
2K	100
5K	40
10K	20
20K	10
100K	2
450K	1
450K	0 (未過濾)

Signal_Inverted:

0: 輸入訊號正常 (無反相)。

1: 輸入訊號為反相 (表示電壓 HIGH 會視為 LOW 處理，而電壓 LOW 會視為 HIGH 處理)。

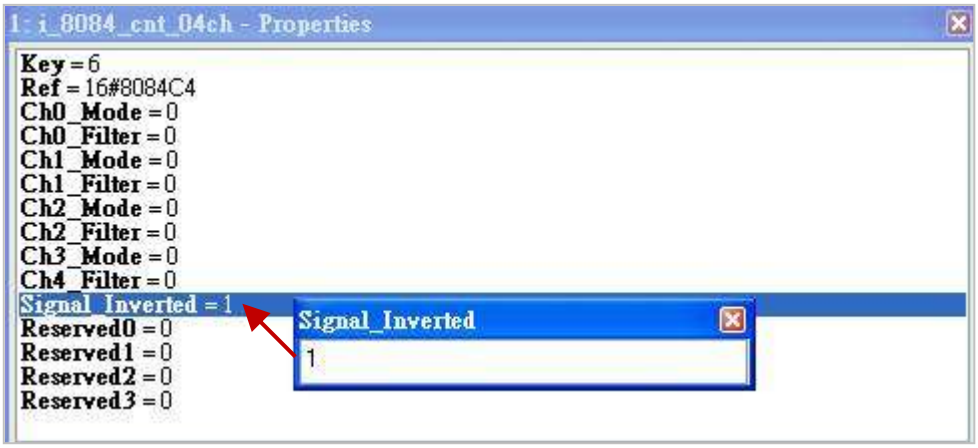
例如，若 "Signal_Inverted" 設為 “0” (無反相) 且 “Ch_Mode” 設為 “0” (Pulse/DIR)，

則當 "DIR" 訊號為 High 時，計數器將會開始計數。

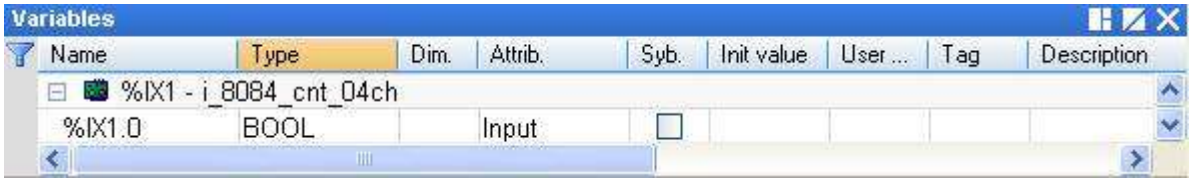
若 "Signal_Inverted" 設為 “1” (反相) 且 “Ch_Mode” 設為 “0” (Pulse/DIR)，

則當 "DIR" 訊號為 High 時，計數器將會倒數計數。

2. 滑鼠雙擊所需項目，來輸入數值並按 “Enter” 鍵完成設定。



3. 在 “I/O Boards” 視窗內連上 “i_8084_cnt_ch04” 後，會自動在 “Variables” 視窗中新增 1 個 "BOOL” 輸入變數 (無作用，固定為 “FALSE”)。

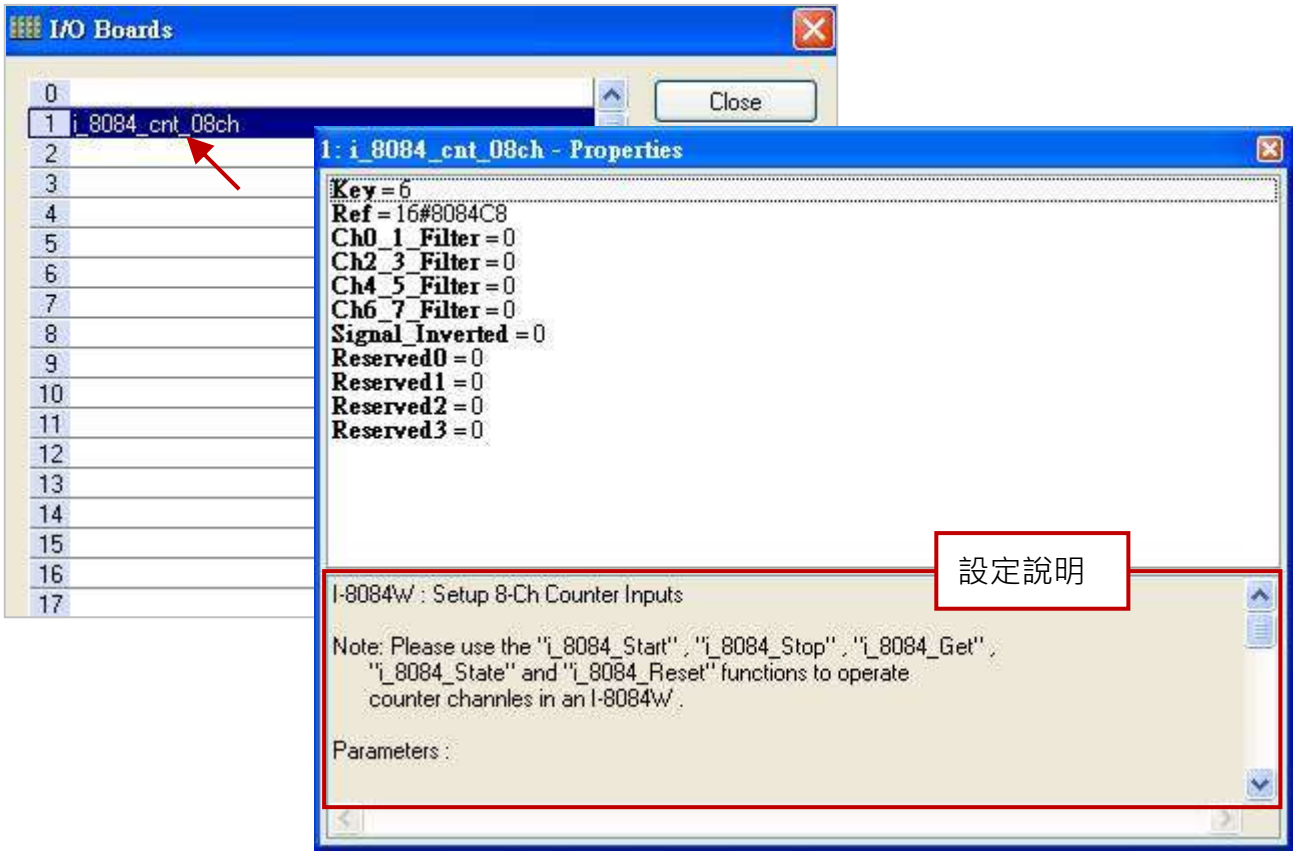


4. 完成 “I/O board” 連結後，請參考 4.9 節在 “程式區” 使用 "COUNTER_START"、"COUNTER_STOP"、"COUNTER _GET"、"COUNTER_STATE" 與 “COUNTER_RESET” 函式 或 ST 語法，來操作 I-8084W 的 Counter 功能。

4.7.3 i_8084_cnt_ch08 (8 通道 UP 計數器)

註: 請使用 Win-GRAF Workbench 中的 "COUNTER_START"、"COUNTER_STOP"、"COUNTER _GET"、"COUNTER_STATE" 與 "COUNTER_RESET" 功能方塊，來操作 I-8084W 的計數通道。

1. 滑鼠雙擊 "i_8084_cnt_ch08" 開啟 “Properties” 視窗，來查看設定說明。



參數說明:

Ch_Filter: 單位為微秒 (μs = 0.000001 秒)，可設為 “0” ~ “200”，預設值為 “0” (不過濾)。"Ch_Filter" 是用來過濾掉一些很小的雜訊寬度。建議設為 “0”，若無雜訊考量或採用即時 (Real-time) 量測。下表為建議的設定:

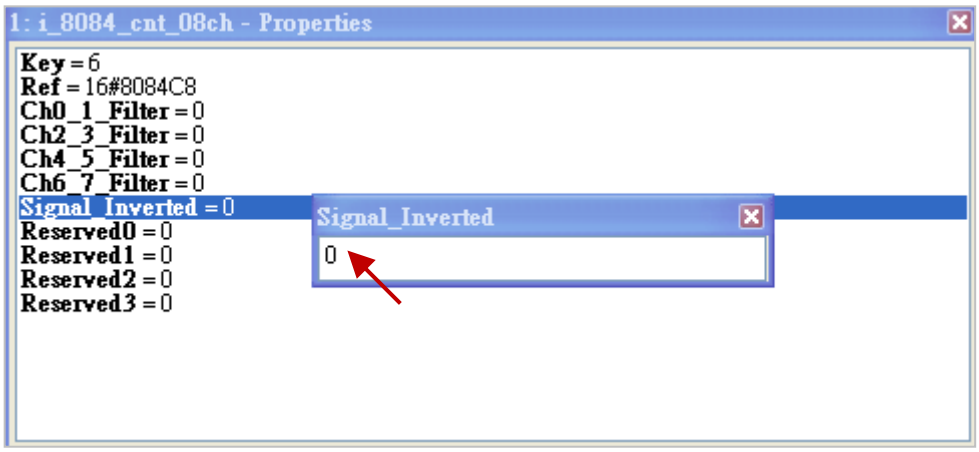
最大輸入頻率 (Hz)	建議的設定值
1K	200
2K	100
5K	40
10K	20
20K	10
100K	2
450K	1
450K	0 (未過濾)

Signal_Inverted:

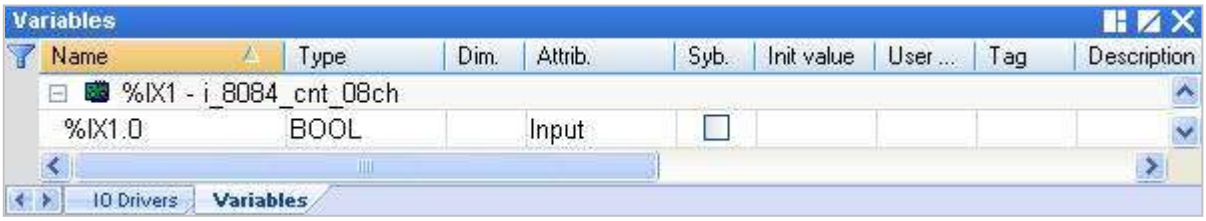
0: 輸入訊號正常 (無反相)。

1: 輸入訊號為反相 (表示電壓 HIGH 會視為 LOW 處理，而電壓 LOW 會視為 HIGH 處理)。

2. 滑鼠雙擊所需項目，來輸入數值並按 “Enter” 鍵完成設定。



3. 在 “I/O Boards” 視窗內連上 “i_8084_cnt_ch08” 後，會自動在 “Variables” 視窗中新增 1 個 "BOOL” 輸入變數 (無作用，固定為 “FALSE”)。

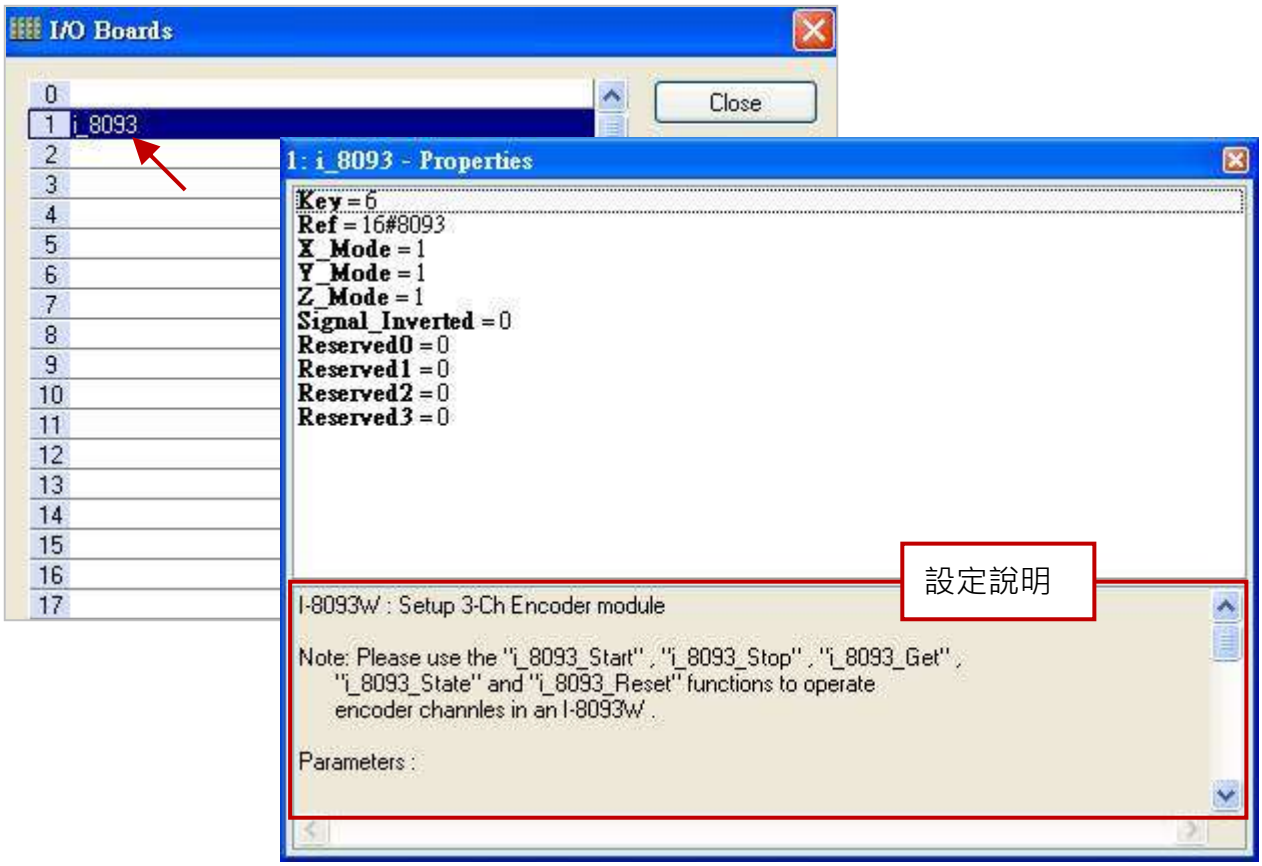


4. 完成 “I/O board” 連結後，請參考 4.9 節在 “程式區” 使用 "COUNTER_START"、"COUNTER_STOP"、"COUNTER _GET"、"COUNTER_STATE" 與 “COUNTER_RESET” 函式 或 ST 語法，來操作 I-8084W 的 Counter 功能。

4.8 i_8093 (3 軸之高速 Encoder 模組)

I-8093W 是一款 3 軸之高速 Encoder 模組，其每一軸都可獨立設定成 Quadrant、Pulse/Direction 或 CW/CCW 之輸入模式。若不熟悉 I/O 卡的開啟/加入方式，可參考 第四章 (P4-1)。

1. 滑鼠雙擊 "i_8093" 開啟 “Properties” 視窗，來查看設定說明。



參數說明:

X_Mode, Y_Mode, Z_Mode:

X、Y 與 Z 軸的輸入模式，可設定為 “1”、“2” 與 “3”，若設定為其它值，將會採用 “1”。

1: CW/CCW 計數模式。

2: Pulse/Direction 計數模式。

3: A/B phase (quadrant) 計數模式。

Signal_Inverted:

0: 輸入訊號正常 (無反相)。

1: 輸入訊號為反相 (表示電壓 HIGH 會視為 LOW 處理，而電壓 LOW 會視為 HIGH 處理)。

例如，若 "Signal_Inverted" 設為 “0” (無反相) 且 “X_Mode” 設為 “2” (Pulse/Direction)，

則當 " Direxion " 訊號為 High 時，計數器將會開始計數。

若 "Signal_Inverted" 設為 “1” (反相) 且 “X_Mode” 設為 “2” (Pulse/Direction)，

則當 " Direxion " 訊號為 High 時，計數器將會倒數計數。

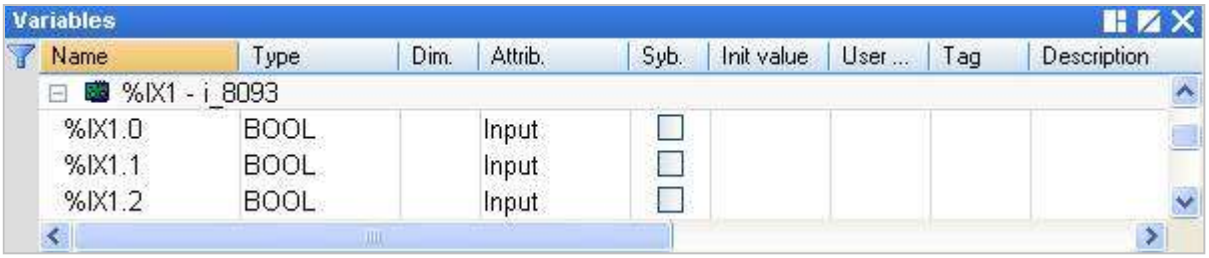
2. 滑鼠雙擊所需項目，來輸入數值並按 “Enter” 鍵完成設定。

3. 在 “I/O Boards” 視窗內連上 “i_8093” 後，會自動在 “Variables” 視窗中新增 3 個 "BOOL” 輸入變數可供使用。

   Ch0: X 軸的 Z-index

   Ch1: Y 軸的 Z-index

   Ch2: Z 軸的 Z-index



4. 完成 “I/O board” 連結後，請參考 4.9 節在 “程式區” 使用 "COUNTER_START"、"COUNTER_STOP"、"COUNTER _GET"、"COUNTER_STATE" 與 “COUNTER_RESET” 函式 或 ST 語法來操作 I-8093W 的 Encoder 功能。

4.9 I-8084W, I-8093W, I-87082W, I-87084W, I-7083 與 I-7080 模組的計數功能

此章節將介紹如何使用 LD 或 ST 語法 來使用 "COUNTER_START"、"COUNTER_STOP"、"COUNTER_GET"、"COUNTER_STATE" 與 “COUNTER_RESET” 函式 (Function)，來操作 I-8084W/I-8093W 的 Counter/Encoder 功能。若不熟悉程式 與 功能方塊的新增方式，可參考 2.3.3 節。

注意:

1. **此章節僅針對 PAC 上的 I-8084W 與 I-8093W 模組作說明。**
2. **使用下列功能方塊前，請先參考 4.7.2 節 (UP/Down 計數器)、4.7.3 節 (UP 計數器) 與 4.8 節 (Encoder) 來連結 "I/O Boards" 功能。**

4.9.1 COUNTER_START (開始計數)

假設: 在 PAC 的 Slot 2 使用 I-8084W 模組，並啟動通道 5 的計數功能。

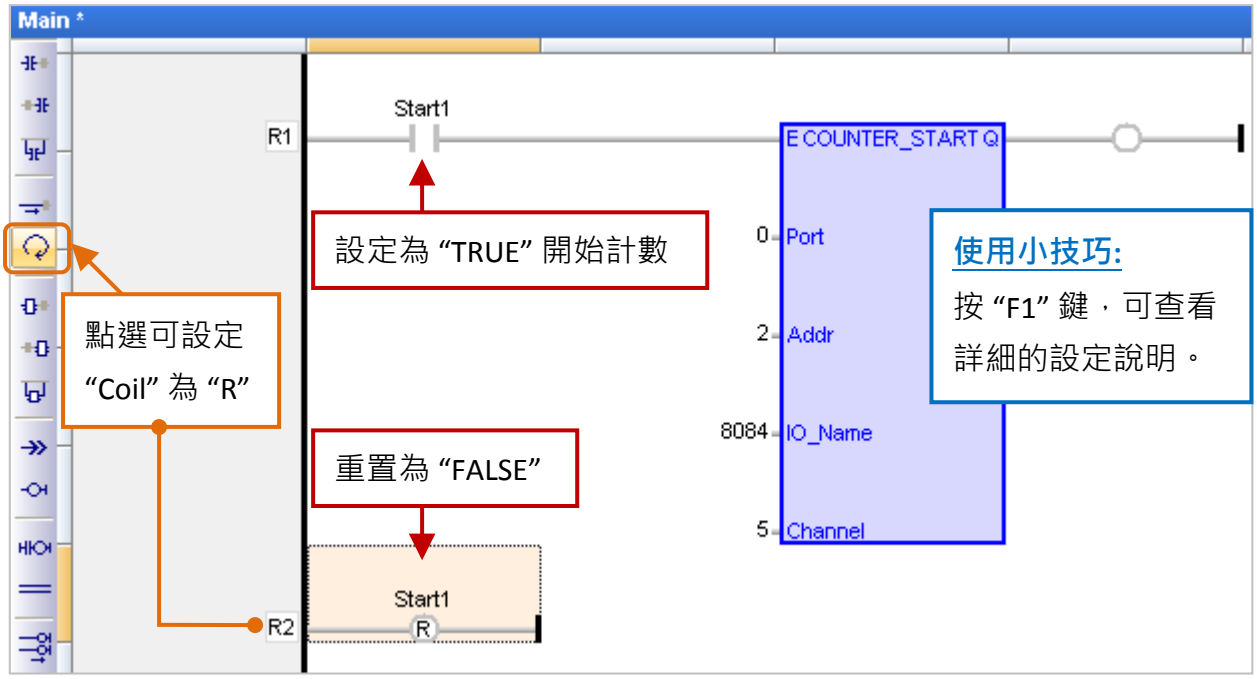
ST 語法:

```
IF  Start1 = TRUE  THEN
  Start1  :=  FALSE ;
  TMP_BOOL  :=  Counter_Start (0, 2, 8084, 5) ;
END_IF ;
```

註: 可先在 "變數區" 建立一個 "Start1"、"TMP_BOOL", 布林變數。

LD 語法:

(“Start1”: 布林變數，設定為 “TRUE” 開始計數，執行後重置 “Start1” 為 “FALSE”。)



Port: (資料型態: “DINT”)

若是在 PAC 的 Slot 0 ~ 7 使用的 I/O 模組，請設定為 "0"；若是透過 COM Port 連接的 DCON 遠端 I/O 模組，可設定為 "1 ~ 37" (視 PAC 而定，表示 COM1 ~ COM37)。

Addr: (資料型態: “DINT”)

若是在 PAC 的 Slot 0 ~ 7 使用的 I/O 模組，請設定為插槽編號 (0 ~ 7)。

若是透過 COM Port 連接的 DCON 遠端 I/O 模組，請設定為該模組的 Net-ID 位址 (1 ~ 255)。

IO_Name: (資料型態: “DINT”)

Counter/Encoder 模組的名稱，可設定為 "8084”、"8093"、"87084"、"87082"、"7083" 與 "7080"。

Channel: (資料型態: “DINT”)

Counter/Encoder 模組的通道編號，可設定為 "0"、"1"、...，視模組而定。

例如: I-8093W 中 "0" 表示 X 軸; "1" 表示 Y 軸; "2" 表示 Z 軸。

Q: 資料型態為 “BOOL”，“TRUE”: 表示 OK ； “FALSE”: 表示錯誤。

4.9.2 COUNTER_STOP (停止計數)

假設: 在 PAC 的 Slot 1 使用 I-8093W 模組，且停用 X 軸的計數功能。

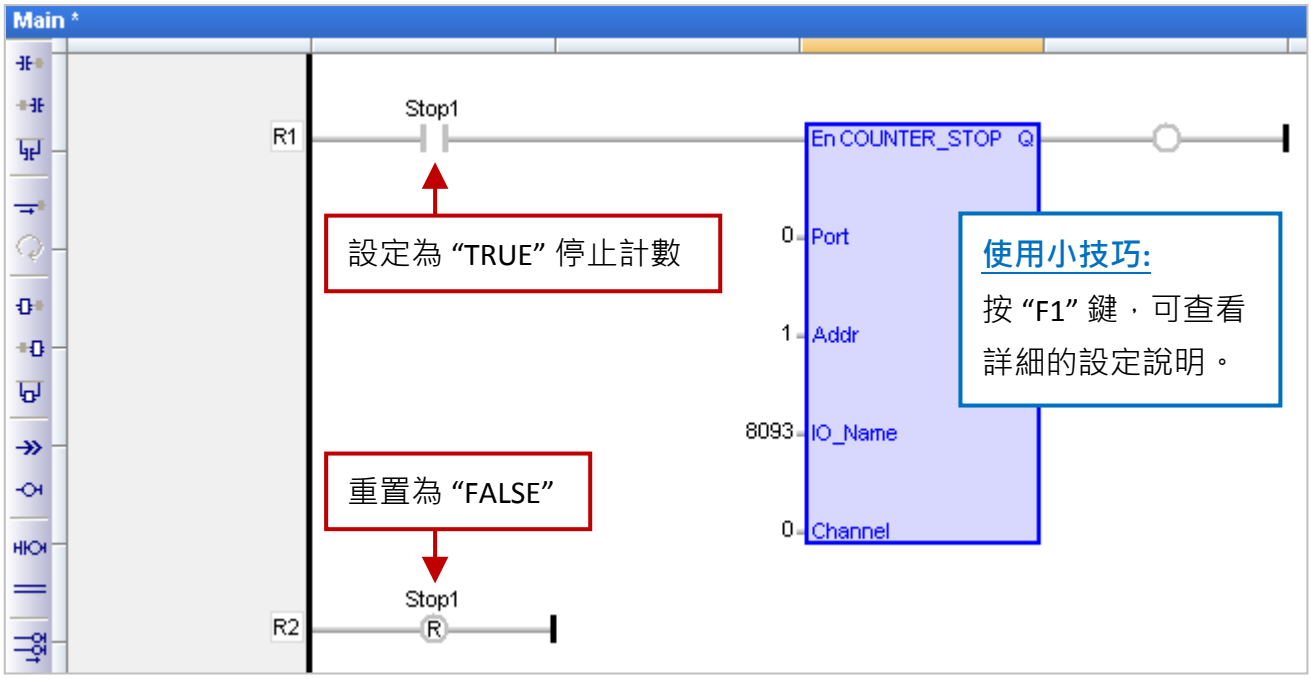
ST 語法:

```
IF  Stop1 = TRUE  THEN
  Stop1  :=  FALSE ;
  TMP_BOOL  :=  Counter_Stop (0, 1, 8093, 0) ;
END_IF ;
```

註: 可先在 "變數區" 建立一個 "Stop1"、"TMP_BOOL", 布林變數。

LD 語法:

("Stop1": 布林變數，設定為 "TRUE" 開始計數，執行後重置 "Stop1" 為 "FALSE"。)



Port: (資料型態: "DINT")

若是在 PAC 的 Slot 0 ~ 7 使用的 I/O 模組，請設定為 "0"；若是透過 COM Port 連接的 DCON 遠端 I/O 模組，可設定為 "1 ~ 37" (視 PAC 而定，表示 COM1 ~ COM37)。

Addr: (資料型態: "DINT")

若是在 PAC 的 Slot 0 ~ 7 使用的 I/O 模組，請設定為插槽編號 (0 ~ 7)。

若是透過 COM Port 連接的 DCON 遠端 I/O 模組，請設定為該模組的 Net-ID 位址 (1 ~ 255)。

IO_Name: (資料型態: "DINT")

Counter/Encoder 模組的名稱，可設定為 "8084"、"8093"、"87084"、"87082"、"7083" 與 "7080"。

Channel: (資料型態: "DINT")

Counter/Encoder 模組的通道編號，可設定為 "0"、"1"、...，視模組而定。

例如: I-8093W 中 "0" 表示 X 軸; "1" 表示 Y 軸; "2" 表示 Z 軸。

Q: 資料型態為 "BOOL"，"TRUE": 表示 OK ； "FALSE": 表示錯誤。

4.9.3 COUNTER_GET (取得計數值)

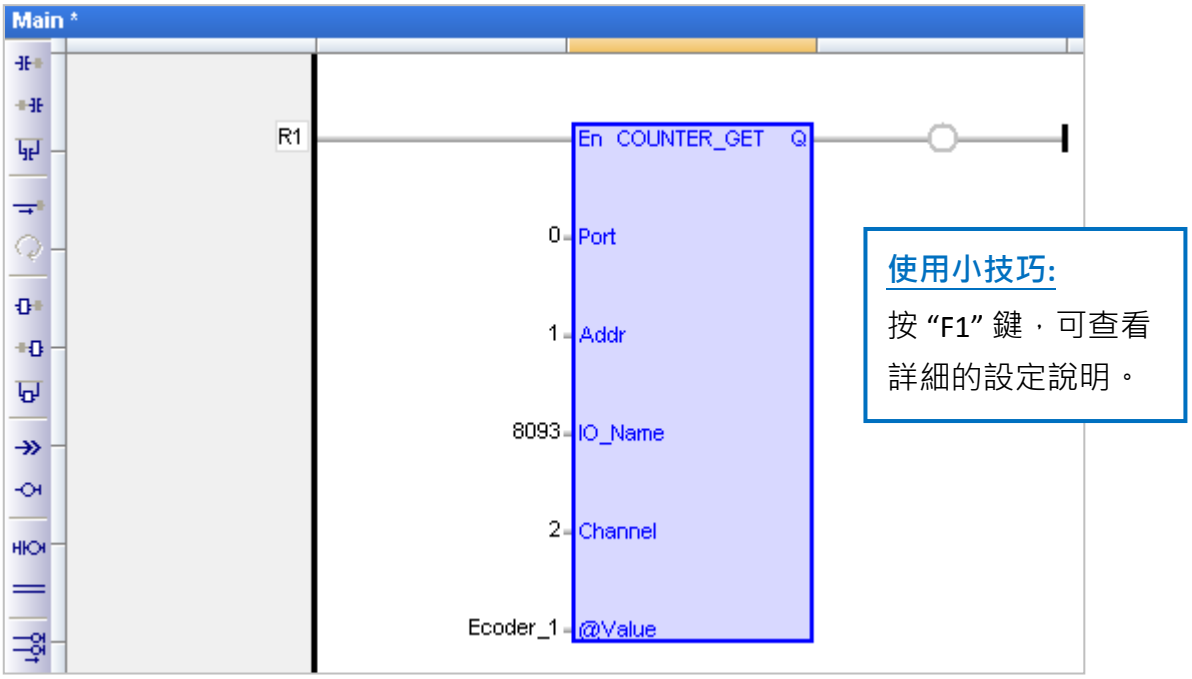
假設: 在 PAC 的 Slot 1 使用 I-8093W 模組，取得 Z 軸的 Encoder 值。

ST 語法:

```
TMP_BOOL := Counter_Get (0, 1, 8093, 2, Encoder_1) ;
```

註: 可先在 "變數區" 建立變數。
"TMP_BOOL" (BOOL)
“Encoder_1" (DINT)。

LD 語法:



Port: (資料型態: “DINT”)
若是在 PAC 的 Slot 0 ~ 7 使用的 I/O 模組，請設定為 "0"；若是透過 COM Port 連接的 DCON 遠端 I/O 模組，可設定為 "1 ~ 37" (視 PAC 而定，表示 COM1 ~ COM37)。

Addr: (資料型態: “DINT”)
若是在 PAC 的 Slot 0 ~ 7 使用的 I/O 模組，請設定為插槽編號 (0 ~ 7)。
若是透過 COM Port 連接的 DCON 遠端 I/O 模組，請設定為該模組的 Net-ID 位址 (1 ~ 255)。

IO_Name: (資料型態: “DINT”)
Counter/Encoder 模組的名稱，可設定為 "8084”、"8093"、"87084"、"87082"、"7083" 與 "7080"。

Channel: (資料型態: “DINT”)
Counter/Encoder 模組的通道編號，可設定為 "0"、"1"、...，視模組而定。
例如: I-8093W 中 "0" 表示 X 軸; "1" 表示 Y 軸; "2" 表示 Z 軸。

@Value: (資料型態 可為 “DINT”, “UDINT”, “DWORD”, “LINT” 與 “ULINT”)
用來顯示計數值。(關於數值範圍，可參考 附錄 A)

Q: 資料型態為 “BOOL”，“TRUE”: 表示 OK ； “FALSE”: 表示錯誤。

4.9.4 COUNTER_STATE (取得計數狀態)

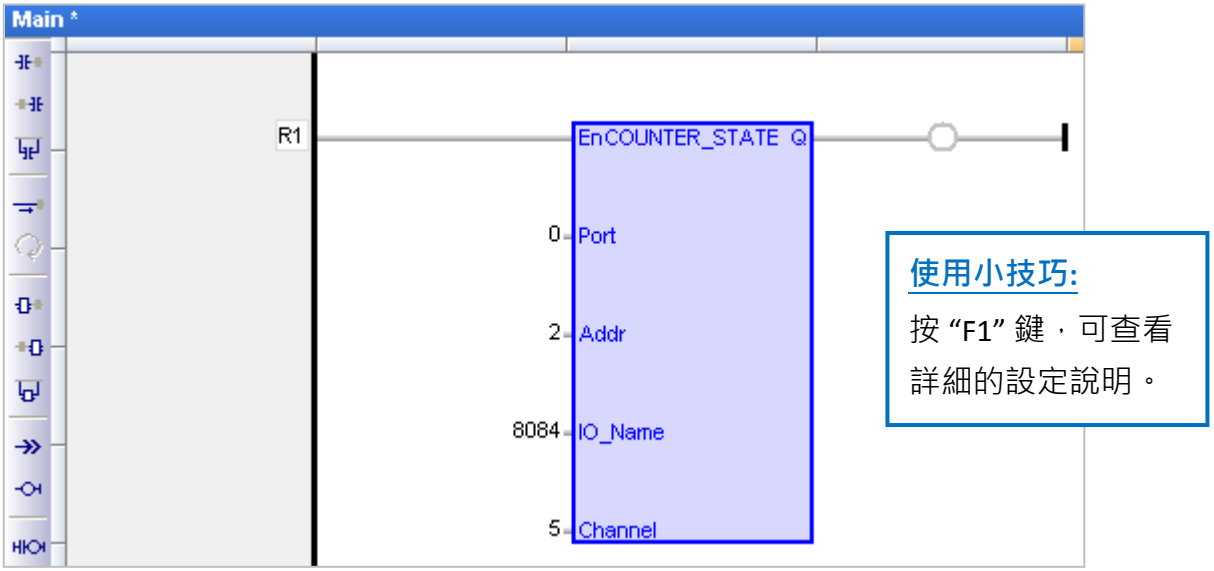
假設: 在 PAC 的 Slot 2 使用 I-8084W 模組，並取得通道 5 的計數狀態。

ST 語法:

```
TMP_BOOL := Counter_State (0, 2, 8084, 5) ;
```

註: 可先在 "變數區" 建立一個 "TMP_BOOL" 布林變數。

LD 語法:



Port: (資料型態: “DINT”)

若是在 PAC 的 Slot 0 ~ 7 使用的 I/O 模組，請設定為 "0"；若是透過 COM Port 連接的 DCON 遠端 I/O 模組，可設定為 "1 ~ 37" (視 PAC 而定，表示 COM1 ~ COM37)。

Addr: (資料型態: “DINT”)

若是在 PAC 的 Slot 0 ~ 7 使用的 I/O 模組，請設定為插槽編號 (0 ~ 7)。

若是透過 COM Port 連接的 DCON 遠端 I/O 模組，請設定為該模組的 Net-ID 位址 (1 ~ 255)。

IO_Name: (資料型態: “DINT”)

Counter/Encoder 模組的名稱，可設定為 "8084”、"8093"、"87084"、"87082"、"7083" 與 "7080"。

Channel: (資料型態: “DINT”)

Counter/Encoder 模組的通道編號，可設定為 "0"、"1"、...，視模組而定。

例如: I-8093W 中 "0" 表示 X 軸; "1" 表示 Y 軸; "2" 表示 Z 軸。

Q: 資料型態為 “BOOL”，“TRUE”: 表示 OK ； “FALSE”: 表示錯誤。

4.9.5 COUNTER_RESET (重置計數值)

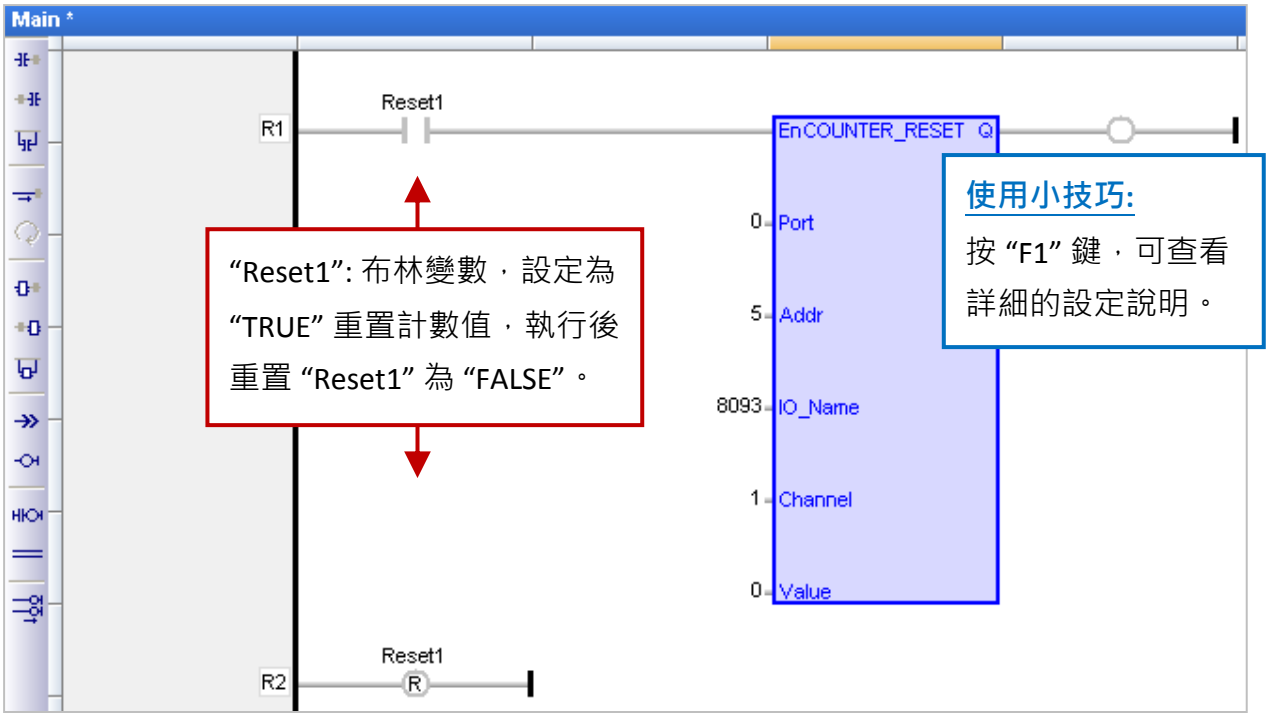
假設: 在 PAC 的 Slot 5 使用 I-8093W 模組，並重置 Y 軸的 Encoder 值為 "0“。

ST 語法:

```
IF  Reset1 = TRUE  THEN
  Reset1  :=  FALSE ;
  TMP_BOOL  :=  Counter_Reset (0, 5, 8093, 1, 0) ;
END_IF ;
```

註: 可先在 "變數區" 建立一個 "Reset1"、"TMP_BOOL", 布林變數。

LD 語法:



Port: (資料型態: “DINT”)

若是在 PAC 的 Slot 0 ~ 7 使用的 I/O 模組，請設定為 "0"；若是透過 COM Port 連接的 DCON 遠端 I/O 模組，可設定為 "1 ~ 37" (視 PAC 而定，表示 COM1 ~ COM37)。

Addr: (資料型態: “DINT”)

若是在 PAC 的 Slot 0 ~ 7 使用的 I/O 模組，請設定為插槽編號 (0 ~ 7)。

若是透過 COM Port 連接的 DCON 遠端 I/O 模組，請設定為該模組的 Net-ID 位址 (1 ~ 255)。

IO_Name: (資料型態: “DINT”)

Counter/Encoder 模組的名稱，可設定為 "8084”、"8093"、"87084"、"87082"、"7083" 與 "7080"。

Channel: (資料型態: “DINT”)

Counter/Encoder 模組的通道編號，可設定為 "0"、"1"、...，視模組而定。

例如: I-8093W 中 "0" 表示 X 軸; "1" 表示 Y 軸; "2" 表示 Z 軸。

Value: (資料型態 可為 “DINT”, “UDINT”, “DWORD”, “LINT” 與 “ULINT”)

設定欲重置的數值。

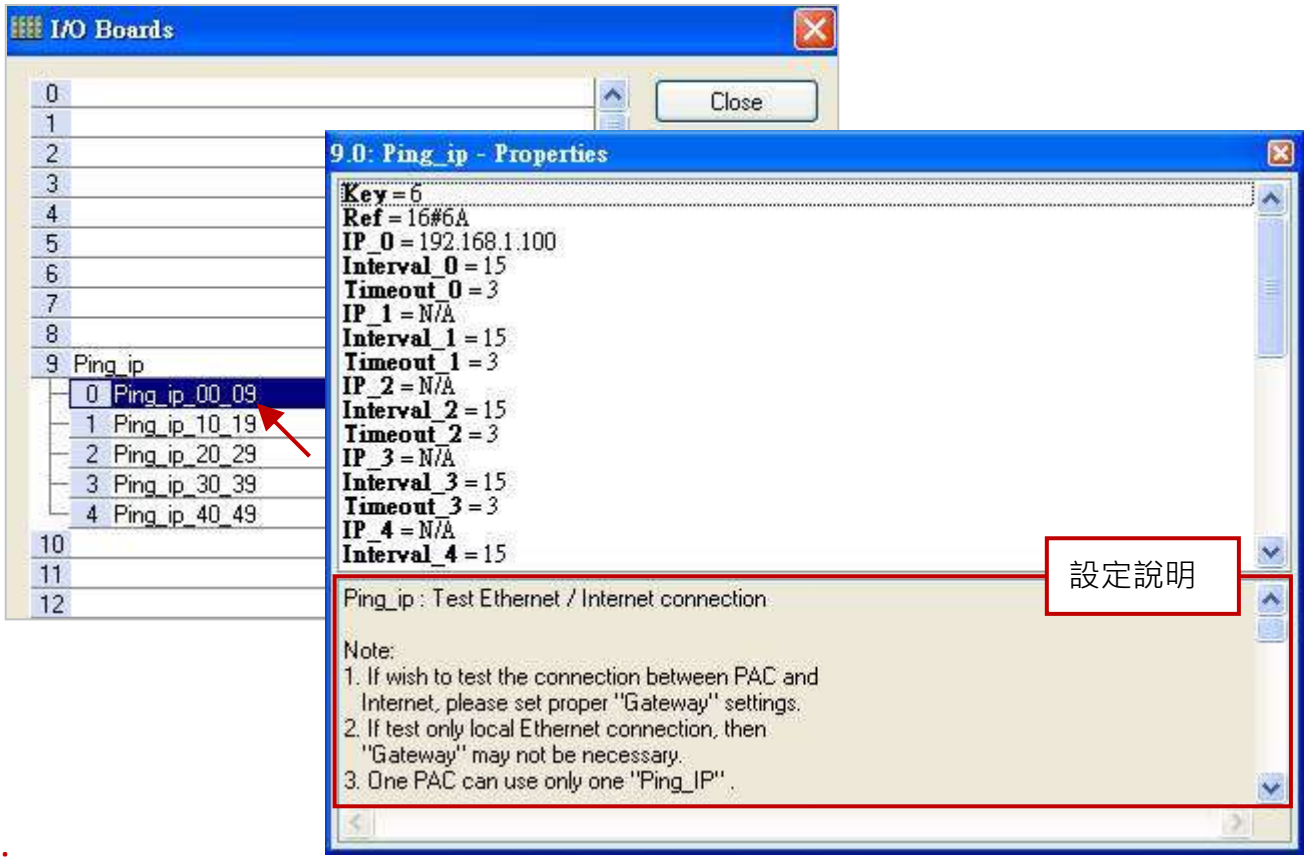
Q: 資料型態為 “BOOL”，“TRUE”: 表示 OK ； “FALSE”: 表示錯誤。

4.10 Ping_ip (測試遠端的 Ethernet/Internet 設備連線)

"Ping_ip" 功能可用來測試遠端的 Ethernet/ Internet 設備連線是否正常? 最多可設定 50 個 IP 位址。若不熟悉 “I/O board” 的開啟/加入方式，可參考 第四章 (P4-1)。

1. 滑鼠雙擊 "Ping_ip" 開啟 “Properties” 視窗，來查看設定說明。

   注意: Slot0 ~ 7 是保留給 PAC I/O 模組，請使用在 Slot8 (含) 以上的位置。



註:

1. 如欲測試 PAC 與 Internet 設備之間的連線，請在 PAC 內設置正確的 "Gateway" 設定。
2. 若僅測試區域網路 (Ethernet) 的連線，則無需設置 "Gateway"。
3. 一台 PAC 只允許使用 1 個 “ping_ip” 功能 (勿使用 2 個或多個)。
4. 當連線 (Ping) 成功後，會回傳一個布林值 "TRUE"。
5. 當連線 (Ping) 失敗後，會再傳一次。若仍失敗，會回傳一個布林值 "FALSE"。

參數說明:

IP_01 ~ IP_49: (資料型態: "STRING"。)

遠端設備的 IP 位址，設定為 'N/A' 表示不啟用

例如: 設定為 "192.168.1.100" 或 "52.19.125.242" 或 "N/A"。

Interval_01 ~ Interval_49: (資料型態: "DINT"。)

傳送 “ping” 命令的間隔時間。單位為秒，預設值為 15 秒，可設定為 6 ~ 86,400 秒，若設定值小於 6 會設定為 “6”，若設定值大於 86,400 (24 小時) 則會設定為 “86,400”。

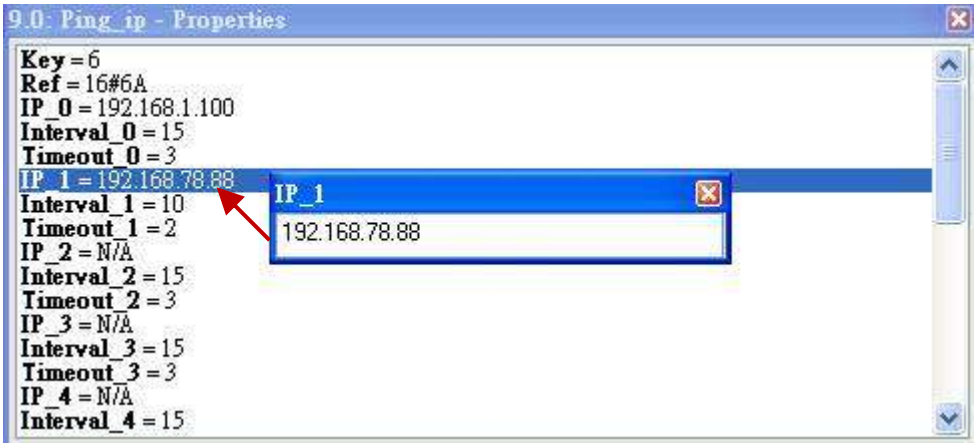
Timeout_01 ~ Timeout_49: (資料型態: "DINT"。)

“ping” 命令的逾時時間。單位為秒，預設值為 3 秒，可設定為 2 ~ 30 秒，若設定值小於 2 會設定為 “2”，若設定值大於 30 則會設定為 “30”。

注意: “Interval_x” 的值至少要設定為 “Timeout_x” 的三倍，否則 PAC 會採用三倍的值。

例如: “Timeout_00” 設定為 “10”，“Interval_00” 設定為 “20”，則 PAC 會採用的 “Interval_00” 值為 “30” (即，10 x 3 = 30)。

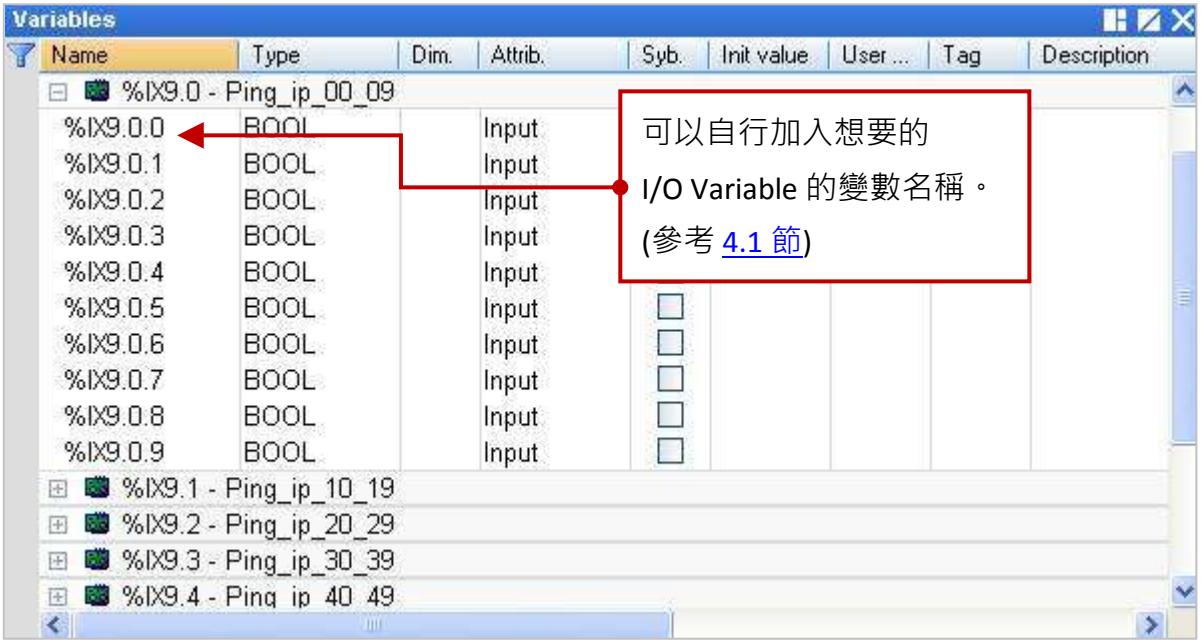
2. 滑鼠雙擊所需項目，來輸入數值並按 “Enter” 鍵完成設定。



3. 在 “I/O Boards” 視窗內連上 "Ping_ip" 後，會自動在 “Variables” 視窗中新增 50 個布林變數，當 Win-GRAF 有連上 PAC 時，會顯示出連線的狀態。

True: 表示連線 OK。

FALSE: 表示連線失敗 或 接線問題。

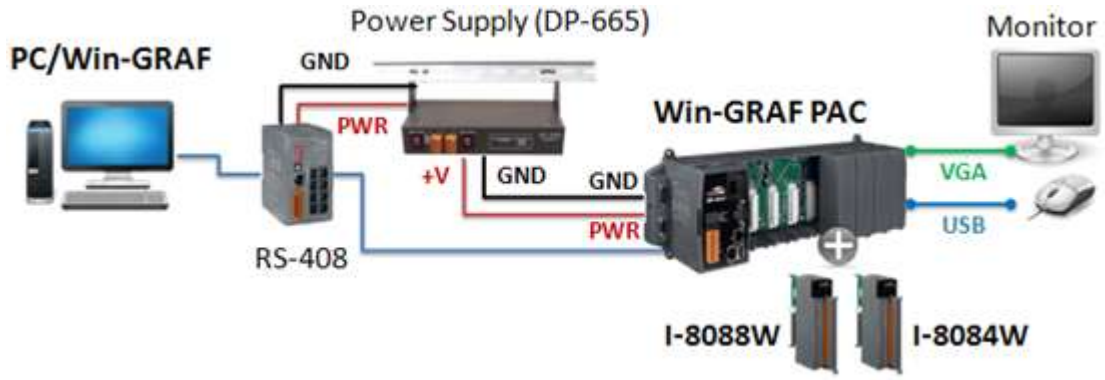


4.11 I-8088W (8 通道 PWM 輸出模組)

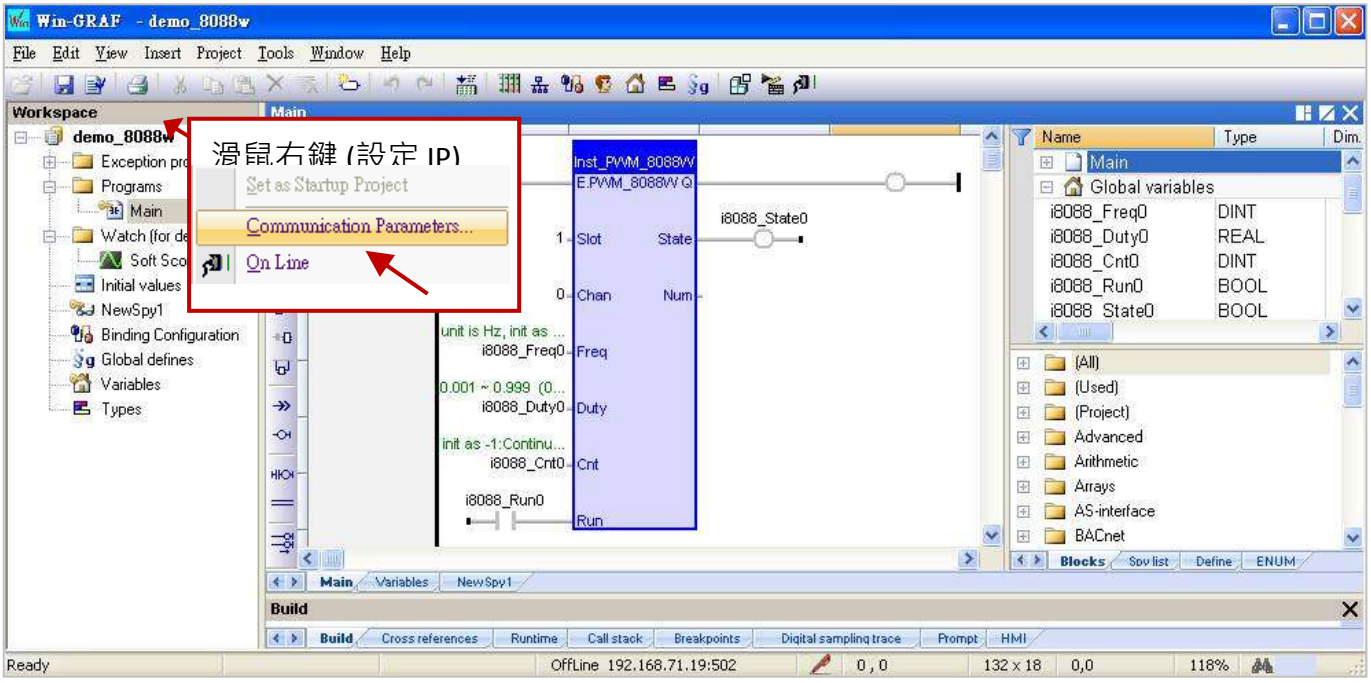
I-8088W 是一款 8 通道 脈寬調變 (PWM) 輸出模組。它的訊號輸出功率 (Duty = High / (High + Low)) 可以是 0.1 % ~ 99.9 %，它的訊號輸出頻率 (Frequency) 可以設成 1 Hz ~ 500 KHz。另外，I-8088W 支援 2 種 PWM 輸出模式，一種是連續輸出模式 (Continuous mode)，另一種是突發模式 (Burst mode)。設定為突發模式時，它只會輸出所指定的訊號波數量，然後就休息不輸出。設定為連續輸出模式，就會連續一直輸出。更多關於 I-8088W 的規格說明，請參訪網址: http://www.icpdas.com/products/Remote_IO/i-8ke/i-8088w_c.htm。

硬體連接圖:

本範例是使用 I-8084W (Slot 2) 來量測 I-8088W (Slot 1) 的 PWM 訊號的頻率 (實際的 PWM 應用應該不需使用 I-8084W)，請將 I-8088W 的 PWM 輸出 (通道 0，PW0) 接到 I-8084W 的頻率輸入 (通道 0，C0A+)。

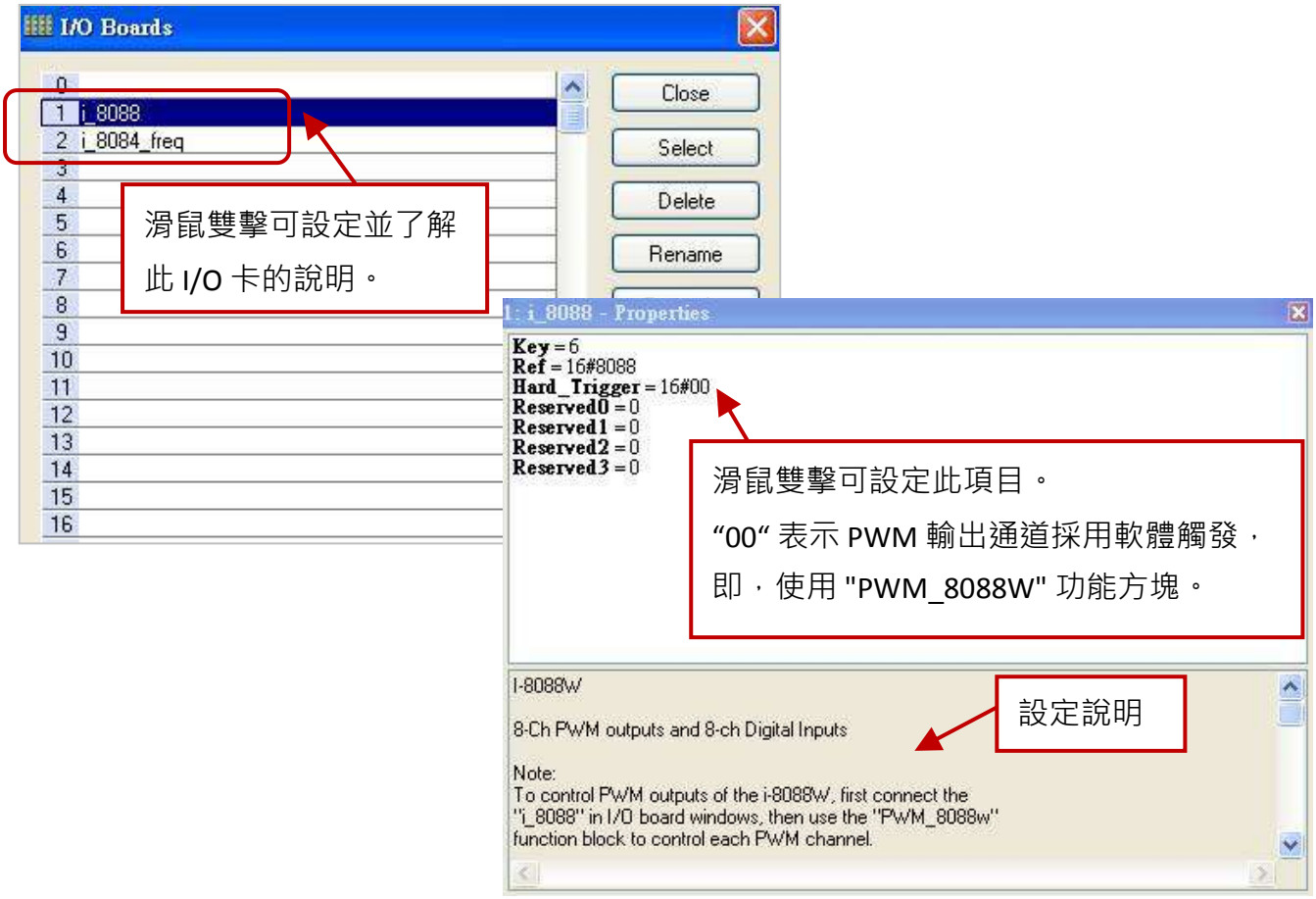


在出貨光碟中 (\Napdos\Win-GRAF\demo-project) 提供了以下將說明的範例程式，請參考 第 12 章 來回存/開啟此專案 (demo_8088w.zip) 並設定好 PAC 目前的 IP 位址。

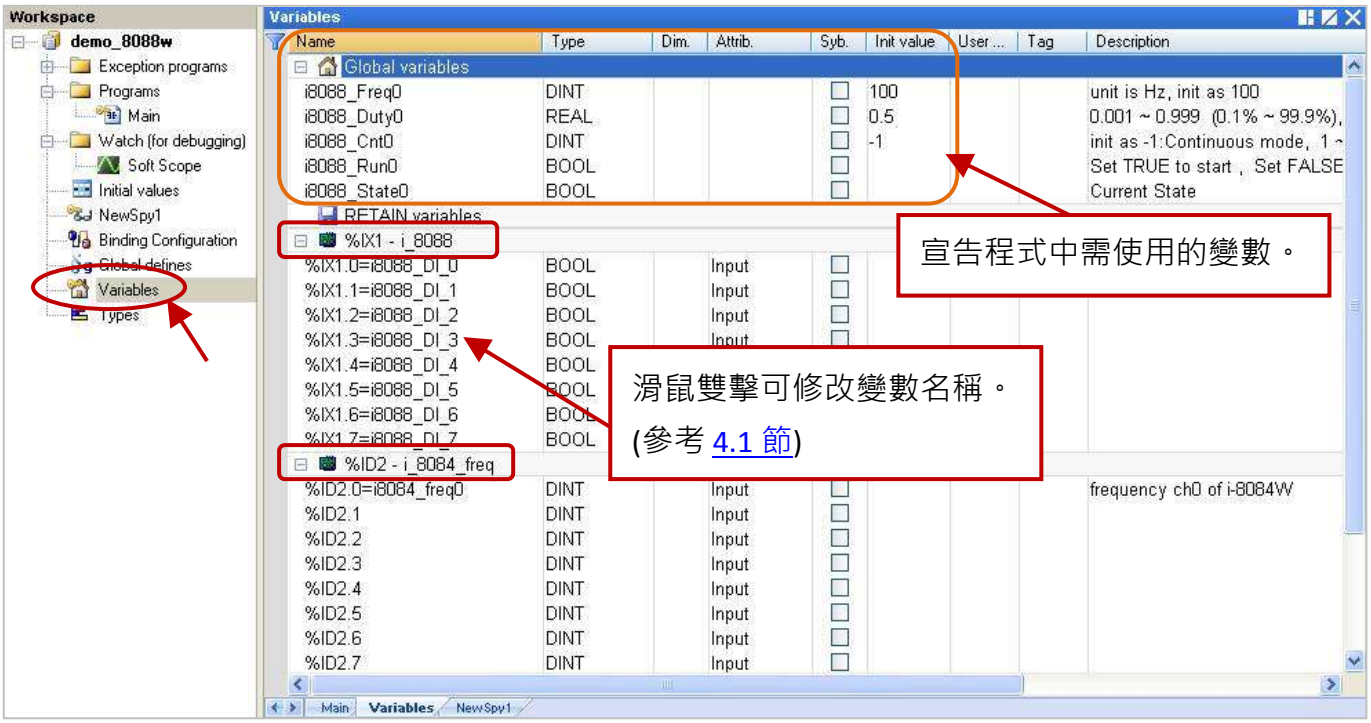


I/O Boards:

本範例在 "I/O Boards" 視窗內，加入 "i_8088" 與 "i_8084_freq" 於對應的 I/O 插槽編號上 (可參考 第四章)。滑鼠雙擊 Slot 編號可開啟 "Properties" 視窗，您可參考下方的使用說明來設定此 I/O 卡。

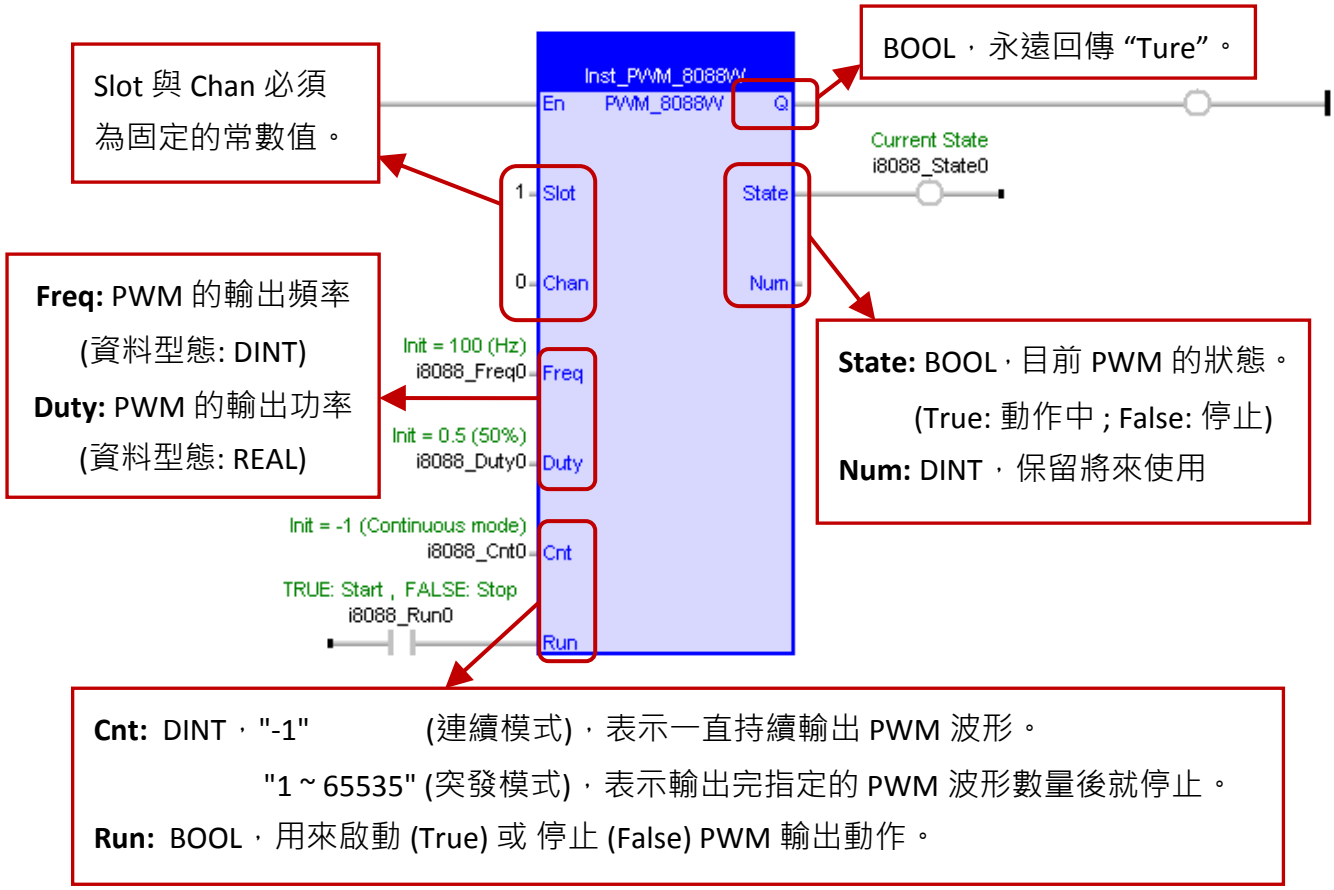


連上 "i_8088" 與 "i_8084_freq" I/O 卡後，會自動在 "Variables" 視窗 (或變數區) 新增相關的變數。另外，也可在此處新增程式中需使用的變數 (可參考 2.3.1 節)。



"PWM_8088W" 功能方塊:

接著，編寫類似下方的 階梯圖 (LD) 程式來控制 I-8088W 各別通道的 PWM 輸出。



參數說明:

Slot: 使用的 I/O 插槽編號，必須為固定的常數值 (資料型態: DINT)，此例為 "1"。

Chan: 使用的 I/O 通道編號，必須為固定的常數值(資料型態: DINT)，此例為 "0"。

Freq: PWM 的輸出頻率 (資料型態: DINT，單位: Hz)，值可以是 1 Hz ~ 500 KHz。
此例設定初值 (Init value) 為 "100" Hz。

Duty: PWM 的輸出功率 (資料型態: REAL)，值可以是 0.001 ~ 0.999 (即 0.1 % ~ 99.9 %)。
此例設定初值 (Init value) 為 0.5 (即 50 %)。

Cnt: 輸出模式 (資料型態: DINT)
連續模式: 設定為 “-1” (此例初值)，一直持續輸出 PWM 波形。
突發模式: 可設定 "1 ~ 65535"，輸出完指定的 PWM 波形數量後就停止。

Run: 使用一個 BOOL 變數來控制啟動 PWM 輸出。(True: 啟動 ; False: 停止)

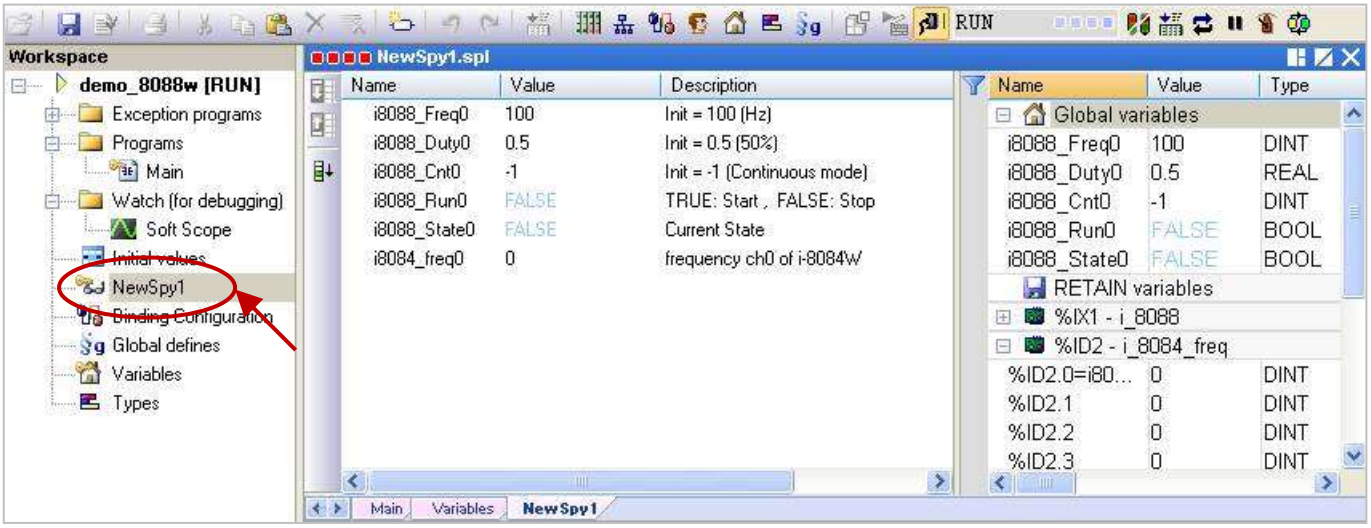
State: 目前 PWM 的狀態 (資料型態: BOOL)。(True: 動作中 ; False: 停止)

測試程式:

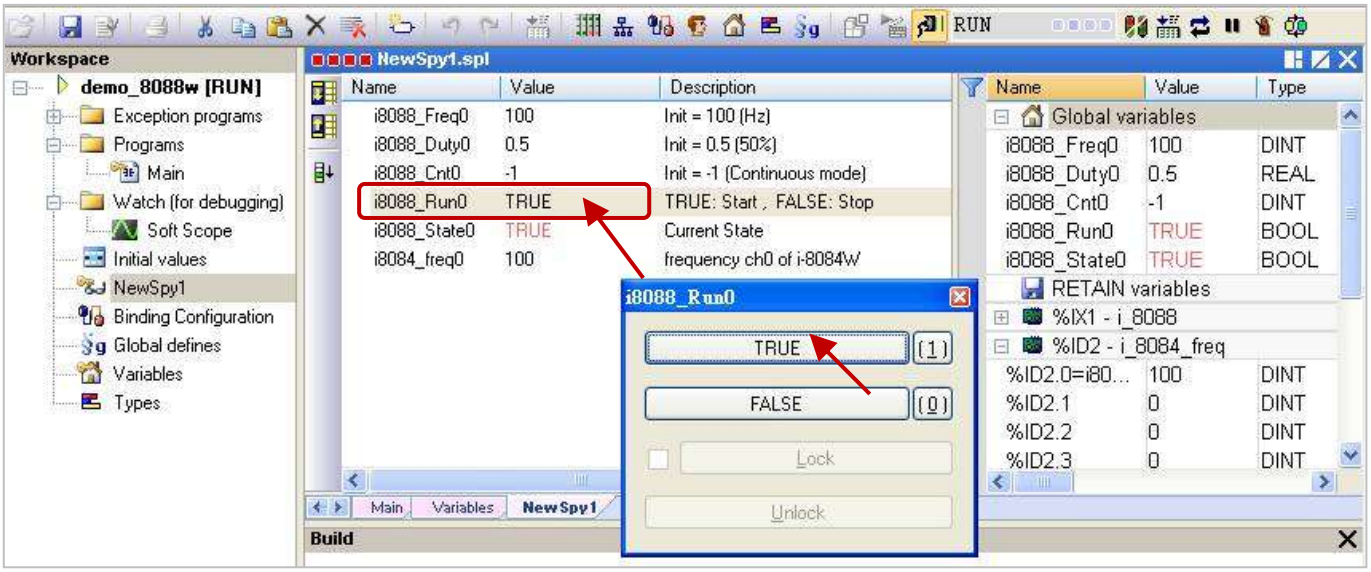
測試之前，請確認已設定好 PAC IP，再編譯/下載程式到 PAC 中。

(若不熟悉操作，可參考 2.3.4 節 與 2.3.5 節)

與 PAC 連線時，在觀測清單中 (Spy list，設定方式可參考 11.3 節) 可見到 I-8088W 的 PWM 輸出頻率 ("i8088_Freq0") 為 100 Hz，輸出功率 ("i8088_Duty0") 為 0.5，採用連續輸出模式 ("i8088_Cnt0" = -1)，且目前 I-8084W 量測到的頻率 ("i8084_Freq0") 為 0 Hz。



請將 "i8088_Run0" 設定為 "TRUE" 來啟動 PWM 輸出，此時 "i8088_State" 也會由 "FALSE" 變為 "TRUE" 並將 PWM 訊號輸出給 I-8084W， "i8084_Freq0" 的量測值將由 0 Hz 變為 100 Hz。



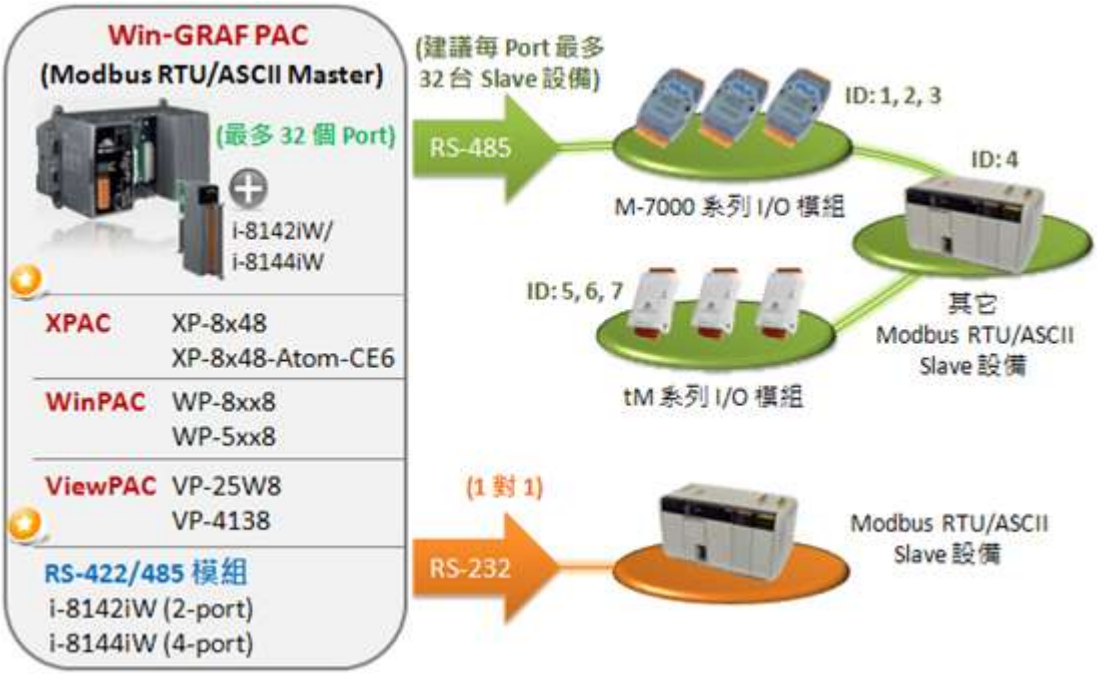
若將 "i8088_Cnt0" 設定為 "500" (Burst 模式)，則輸出完 500 個 PWM 波形後 "i8084_Freq0" 的量測值將變為 "0"，您可試著修改不同的設定值，再將 "i8088_Run0" 設定為 "TRUE" 來觀察輸出的變化。

第 5 章　Modbus Master: 連接其它 Modbus Slave 設備

此章節將介紹如何啟用 Win-GRAF PAC 為 Modbus Master 來連接 Modbus RTU/ASCII Slave 或 Modbus TCP/UDP Slave 設備。

5.1　啟用 Win-GRAF PAC 為 Modbus RTU/ASCII Master

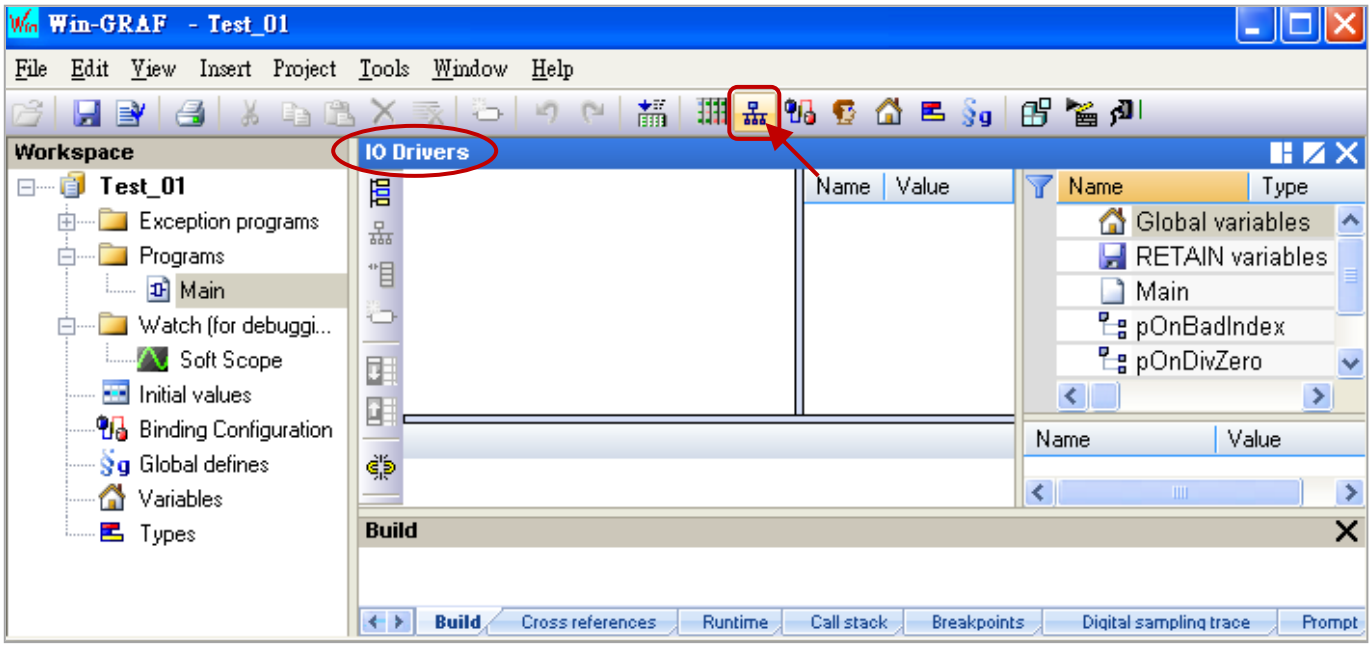
應用示意圖:



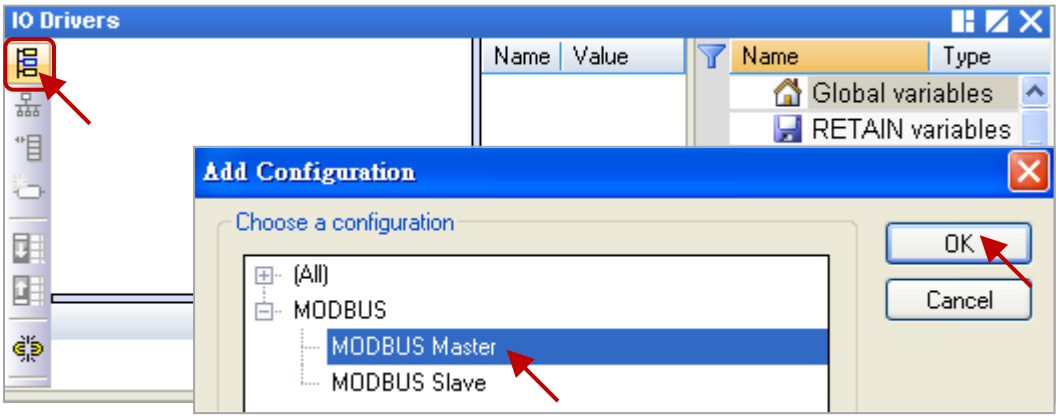
(您可參考 P1-1，來查詢詳細的 PAC 型號)

請參考以下操作步驟:

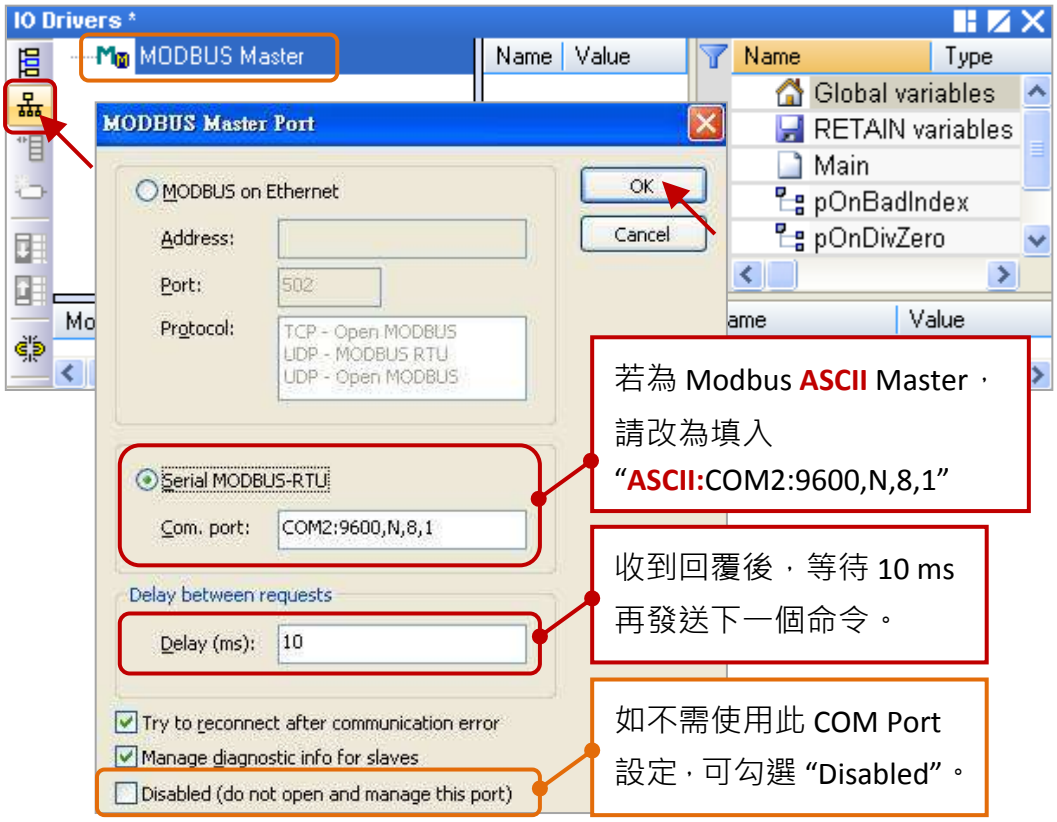
1. 滑鼠點選工具列上的 “Open Fieldbus Configuration” 按鈕來開啟 “I/O Drivers” 視窗。



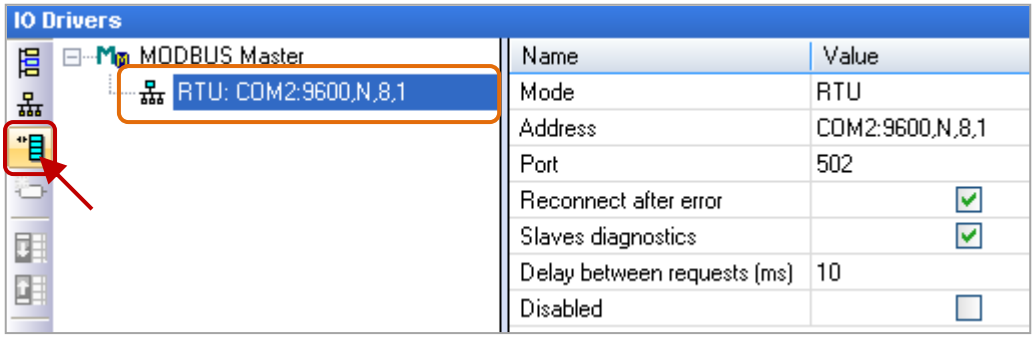
2. 點選 “I/O Drivers” 視窗左側的 “Insert Configuration” 按鈕，再點選 “MOSBUS Master” 並點選 “OK” 來啟用一個 Modbus Master 設定。



3. 點選左側的 “Insert Master/Port” 按鈕，開啟設定視窗。點選 “Serial MODBUS-RTU” 並設定 COM Port (例如: “COM2:9600,N,8,1”) 與 Delay (建議值: 10 ms，可設為 0 ~ 10000)，再點選 “OK”。



4. 點選左側的 “Insert Slave/Data Block” 按鈕，來建立一個 Data Block。



我們將介紹 5 個 Data Block，而每一個 Data Block 都代表一個 Modbus Master Request。

項目	Function Code	Modbus Request	說明
1	2	Read Input Bits	讀取 DI 資料
2	5	Write single coil bit	寫出 DO 資料
3	4	Read Input Registers	讀取 AI 資料
4	6	Write single holding register	寫出 AO 資料 (16-bit)
5	16	Write Holding Registers	寫出 AO 資料 (32-bit)

5.1.1 讀取 DI 資料

1. 於 “MODBUS Master Request” 設定視窗中，設定以下項目並於完成後按 “OK"。

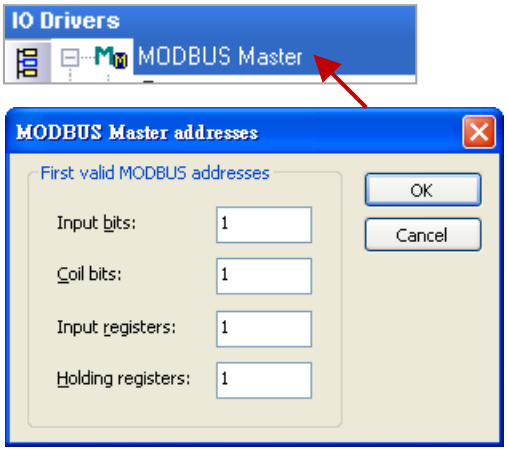
a. Slave/Unit:
填入 Slave 設備的站號 (Net-ID，此例為 “1”)。

b. MODBUS Request:
選擇 “<2> Read Input Bits” 選項。

c. Base address:
預設值由 1 開始。

Nb items:
讀取 DI 的數量 (此例為 16)。

註: 如需修改 “Base address”，可使用滑鼠右鍵點選 “MODBUS Master” 再選擇 “MODBUS Master Addresses” 修改其值。



d. Activacation: 表示 Modbus Request 發送的方式。

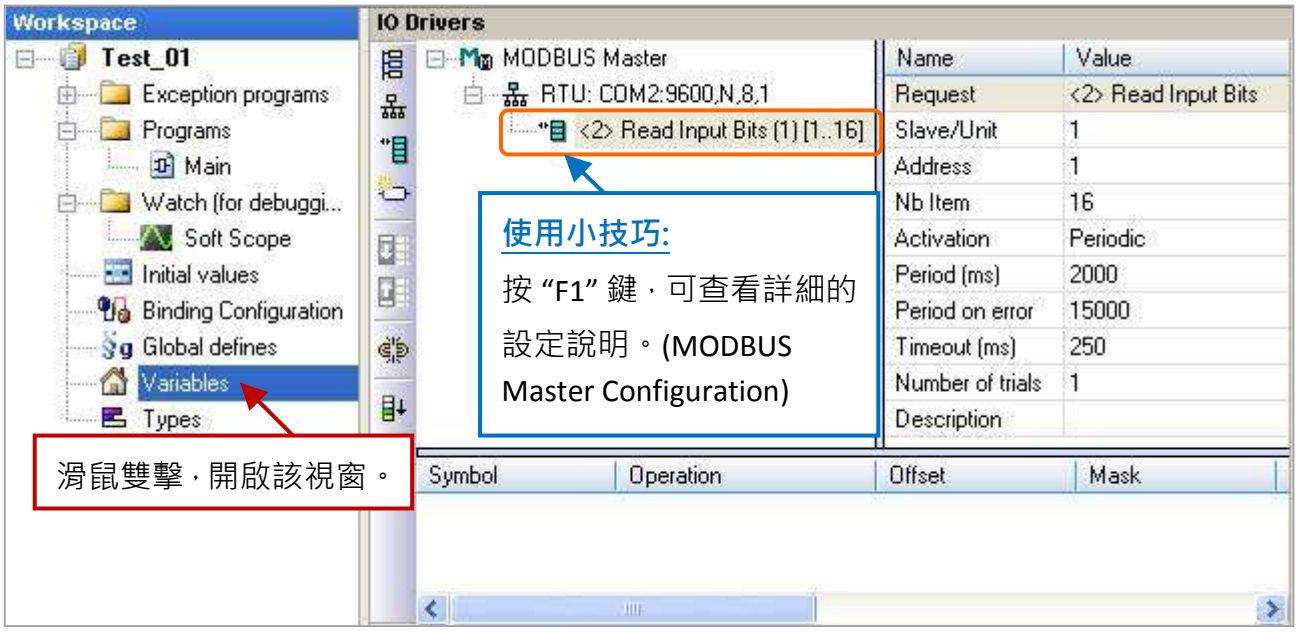
Periodic: 表示週期性的發送，此例為每 2 秒發送一次。“on error” 表示每當發生異常時，下一次的發送時間 (此例為 15 秒)。

On call: 表示程式有呼叫時，才進行發送一次。

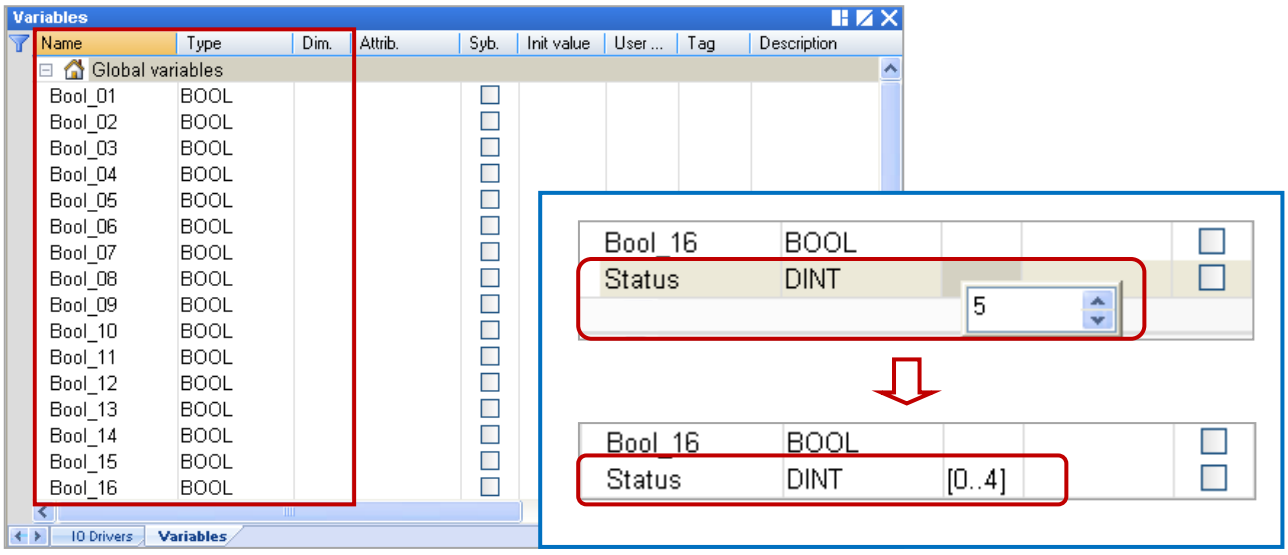
On change: 表示寫出的資料有改變時，才進行發送一次。

e. Timeout: 設定多久未回應，即表示異常。
(對於 Modbus RTU/ASCII 建議值: 200 ~ 1000 ms；此例為 250 ms)

2. 接著，請開啟 “Variables” 視窗，設定需使用的變數。



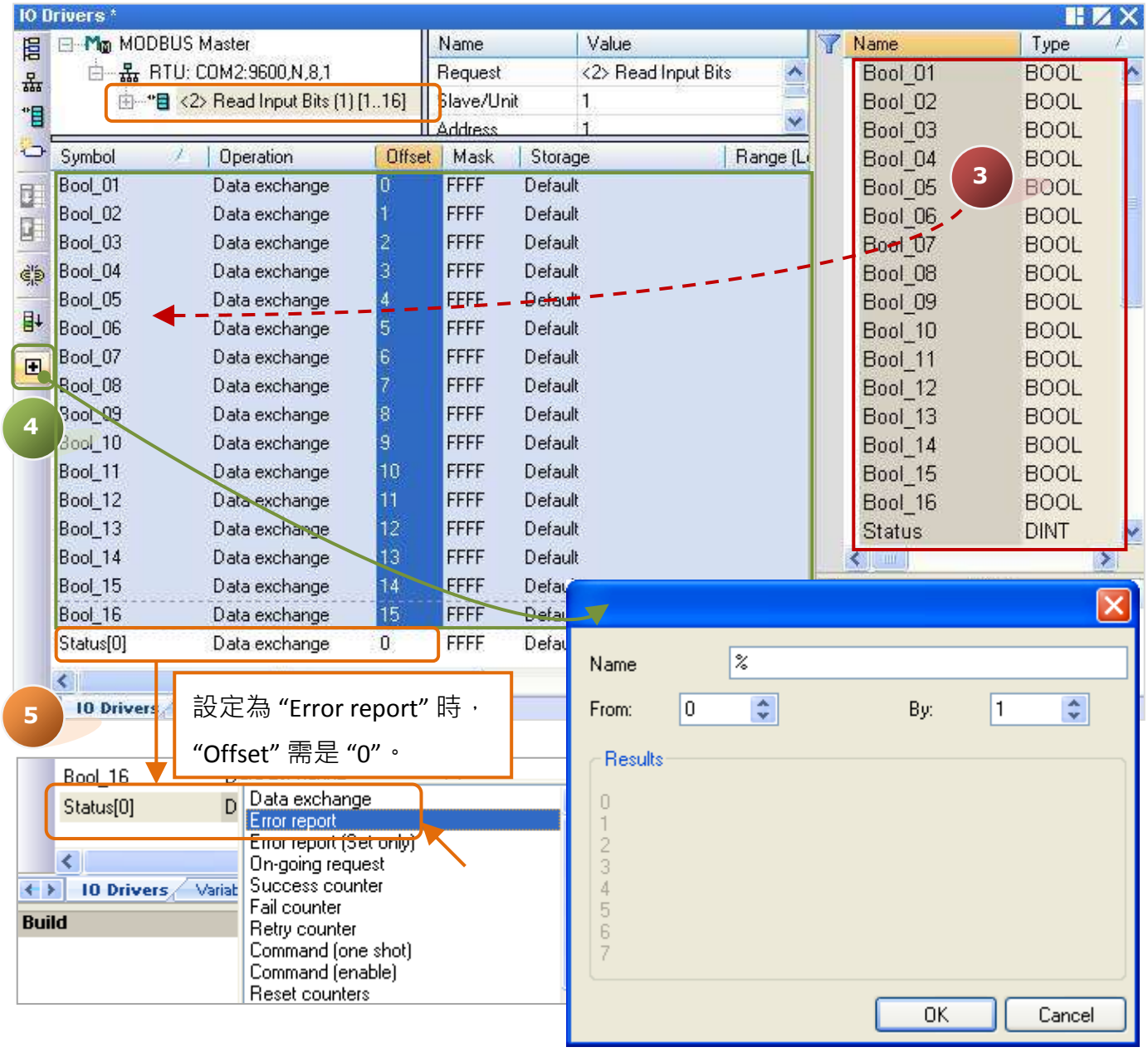
設定 16 個變數 (Name: “Boo_01 ~ Boo_16”; Type: BOOL) 用來讀取資料，與 1 個陣列變數 (Name: “Status"; Dim.: 5; Type: DINT) 用來記錄資料的存取狀況。設定方式可參考 2.3.1 節，設定完成後，畫面如下。



3. 如下圖，於 “I/O Drivers” 視窗，請將變數區中的變數 (“Boo_01” ~ “Boo_16” 與 “Status”) 拖曳到第 1 個 Data Block 的 “Symbol” 區域。 **注意:** “Status” 是一個陣列變數，拖曳到 “Symbol” 區域會是 “Status[0] ~ [4]"，請按 “Delete” 鍵來刪除 “Status[1] ~ [4]"。

4. 接著，選取 “Boo_01 ~ Boo_16” 的 “Offset” 欄位，並點選左側的 “Iterate Property” 按鈕，再設定 “Offset” 值 (From: 0 ; By: 1，可參考 3.1 節 – 步驟 8)。

5. 設定 “Status[0]” 的 “Operation” 為 “Error report” (表示讀取失敗時，該變數值為一個 “Error Code”，讀取成功時則會重置為 “0”)。

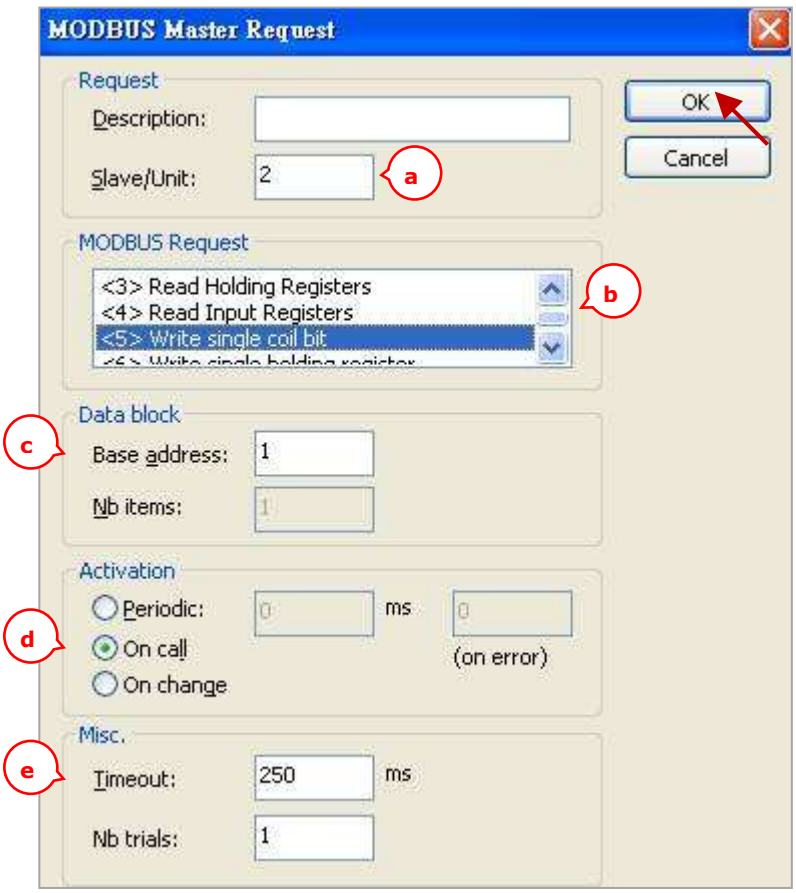


您可在此 “I/O Drivers” 視窗中，按 “F1” 鍵並查看詳細的 Modbus Master 設定說明。

Error Code	說明	Error Code	說明
0	OK (通訊正常)。	8	資料同位元檢查錯誤 (Data Parity Error)。
1	不支援 MODBUS 功能。	10	無效的閘道器 (Gateway) 路徑。
2	無效的 MODBUS 位址。	11	閘道器 (Gateway) 目標異常。
3	無效的 MODBUS 數值。	128	通訊異常 (Timeout)。
4	MODBUS Server 異常。	129	Bad CRC16。
6	Server 忙碌中。	130	RS232 通訊錯誤。

5.1.2 寫出 DO 資料

1. 參考 5.1 節 - 步驟 4，建立第 2 個 Data Block，於 “MODBUS Master Request” 設定視窗中，設定以下項目並於完成後按 “OK"。



a. Slave/Unit:
填入 Slave 設備的站號 (Net-ID，此例為 “2”)。

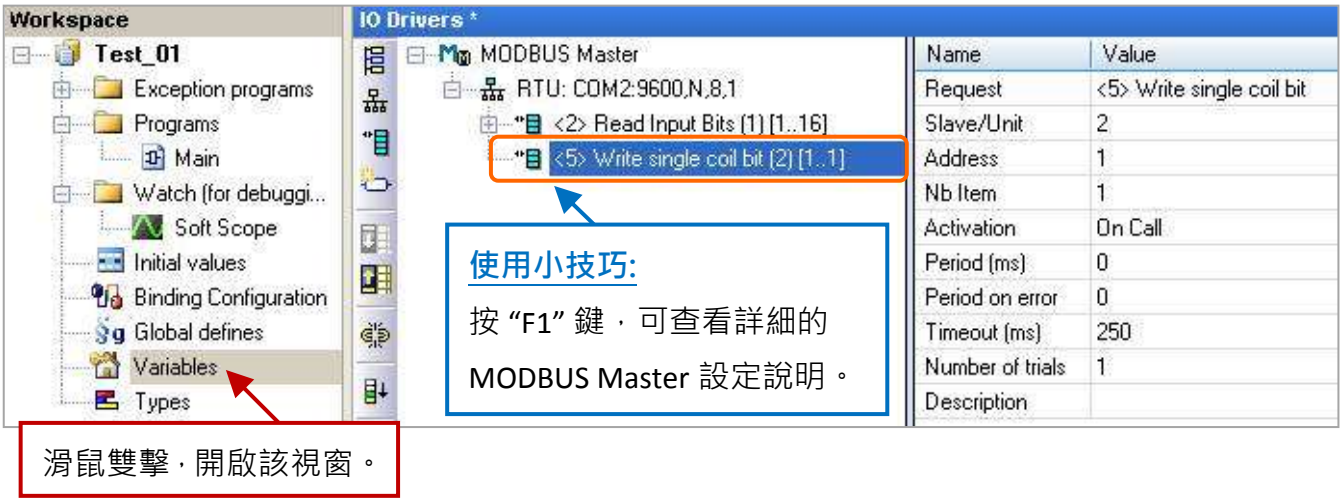
b. MODBUS Request:
選擇 “<5> Write single coil bit”。

c. Base address:
預設值由 1 開始。
(如需修改其值，可參考 5.1.1 節。)

d. On call: 表示程式有呼叫時，才進行發送一次。
(其它項目說明，可參考 5.1.1 節。)

e. Timeout:
設定多久未回應，即表示異常。
(對於 Modbus RTU/ASCII 建議值: 200 ~ 1000 ms；此例為 250 ms)

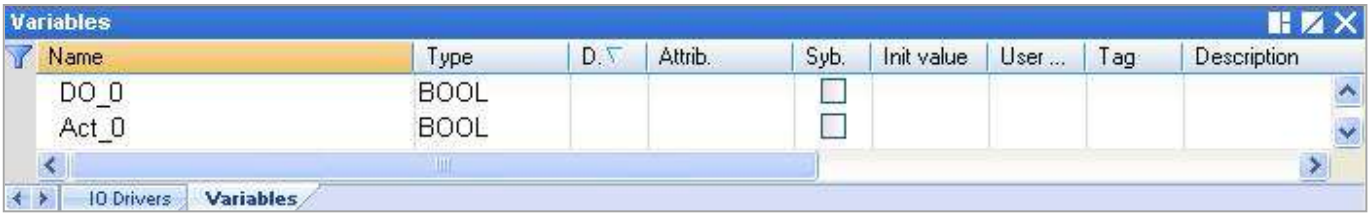
2. 接著，請開啟 “Variables” 視窗，設定需使用的變數。



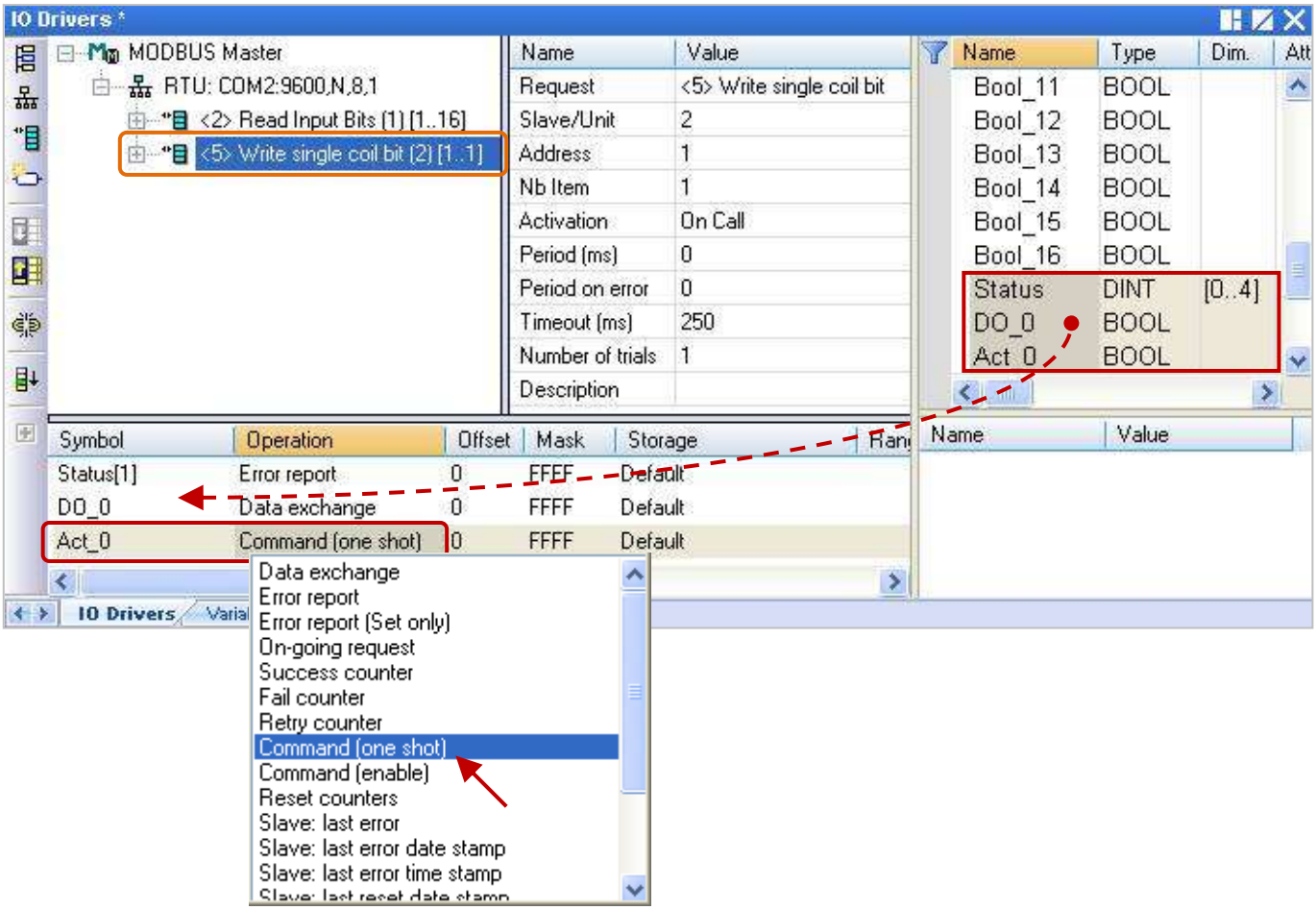
設定 2 個布林變數 (設定方式可參考 2.3.1 節)。

變數名稱	資料型態	說明
DO_0	BOOL	用來寫出 DO 資料。
Act_0	BOOL	此例選擇了 “On call” 的寫出方式，因此需設定此變數來啟動它。

設定完成後，畫面如下。

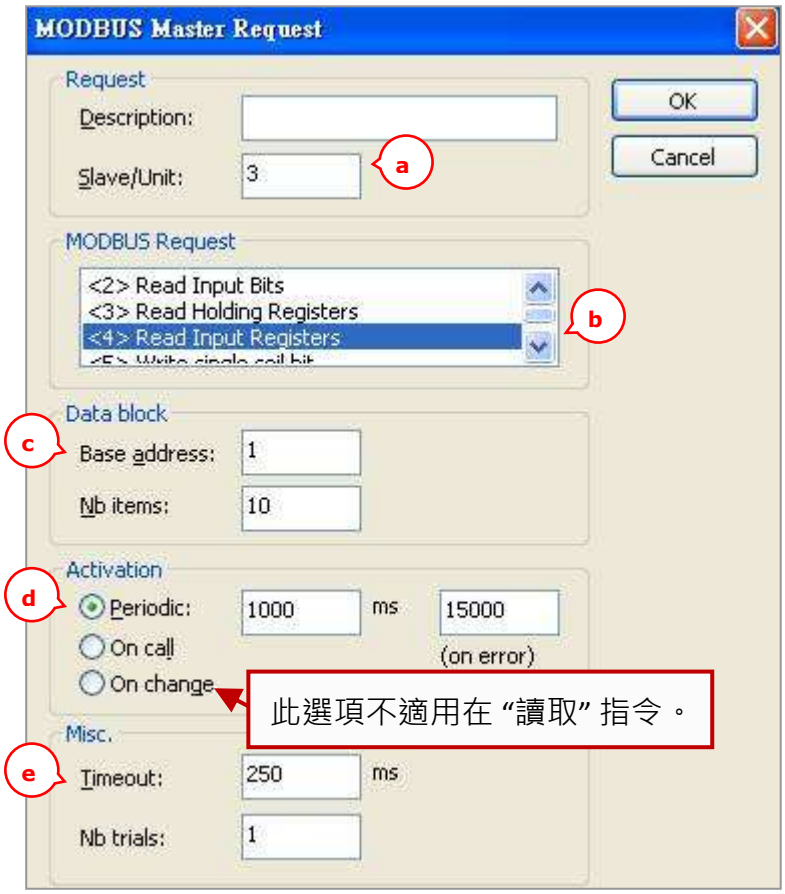


3. 於 “I/O Drivers” 視窗，請將變數區中的變數 (“DO_0”、 “Act_0” 與 5.1.1 節中建立的 “Status”) 拖曳到第 2 個 Data Block 的 “Symbol” 區域。**注意:** “Status” 是一個陣列變數，拖曳到 “Symbol” 區域會是 “Status[0] ~ Status[4]"，請按 “Delete” 鍵來刪除 “Status[0]” 與 “Status[2] ~ [4]"。
4. 設定 “Status[1]” 的 “Operation” 為 “Error report” (表示讀取失敗時，該變數值為一個 “Error Code”，讀取成功時則會重置為 “0”)，按 “F1” 鍵則可查看 Modbus Master 設定說明，於標題 “Status and command variables” 中有詳細的命令、“Error Code” 說明。
5. 設定 “Act_0” 的 “Operation” 為 “Command (one shot)”，表示當 “Act_0” 被設定為 “TRUE” 時，會發送指令一次，並自動重置為 “FALSE”；若選用 "Command (Enable)，表示當 “Act_0” 被設定為 “TRUE” 時，會輪流發送連續指令直到 “Act_0” 被設定為 “FALSE” 時，才會停止發送指令。



5.1.3 讀取 AI 資料

1. 參考 5.1 節 - 步驟 4，建立第 3 個 Data Block，於 “MODBUS Master Request” 設定視窗中，設定以下項目並於完成後按 “OK"。



a. Slave/Unit:
填入 Slave 設備的站號 (Net-ID，此例為 “3”)。

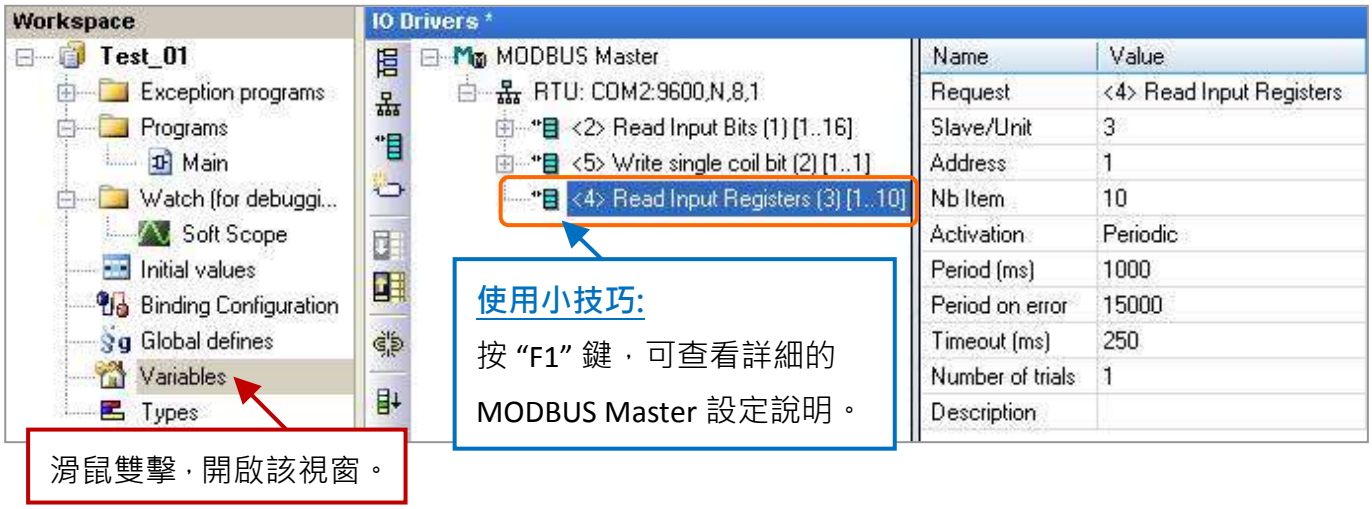
b. MODBUS Request:
選擇 “<4> Read Input Registers”。

c. Base address:
預設值由 1 開始。
(如需修改其值，可參考 5.1.1 節。)
Nb items:
讀取 AI 的數量 (此例為 “10”)。

d. Periodic: (可參考 5.1.1 節。)
表示週期性的發送請求，此例為每 1 秒發送一次。“on error” 表示每當發生異常時，下一次的發送時間 (此例為 15 秒)。

e. Timeout:
設定多久未回應，即表示異常。
(對於 Modbus RTU/ASCII 建議值: 200 ~ 1000 ms；此例為 250 ms)

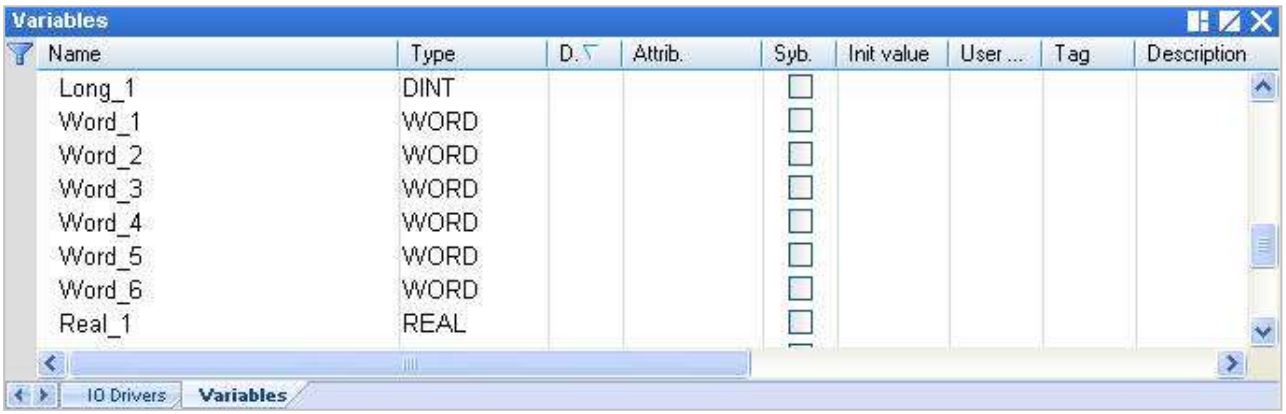
2. 接著，請開啟 “Variables” 視窗，設定需使用的變數。



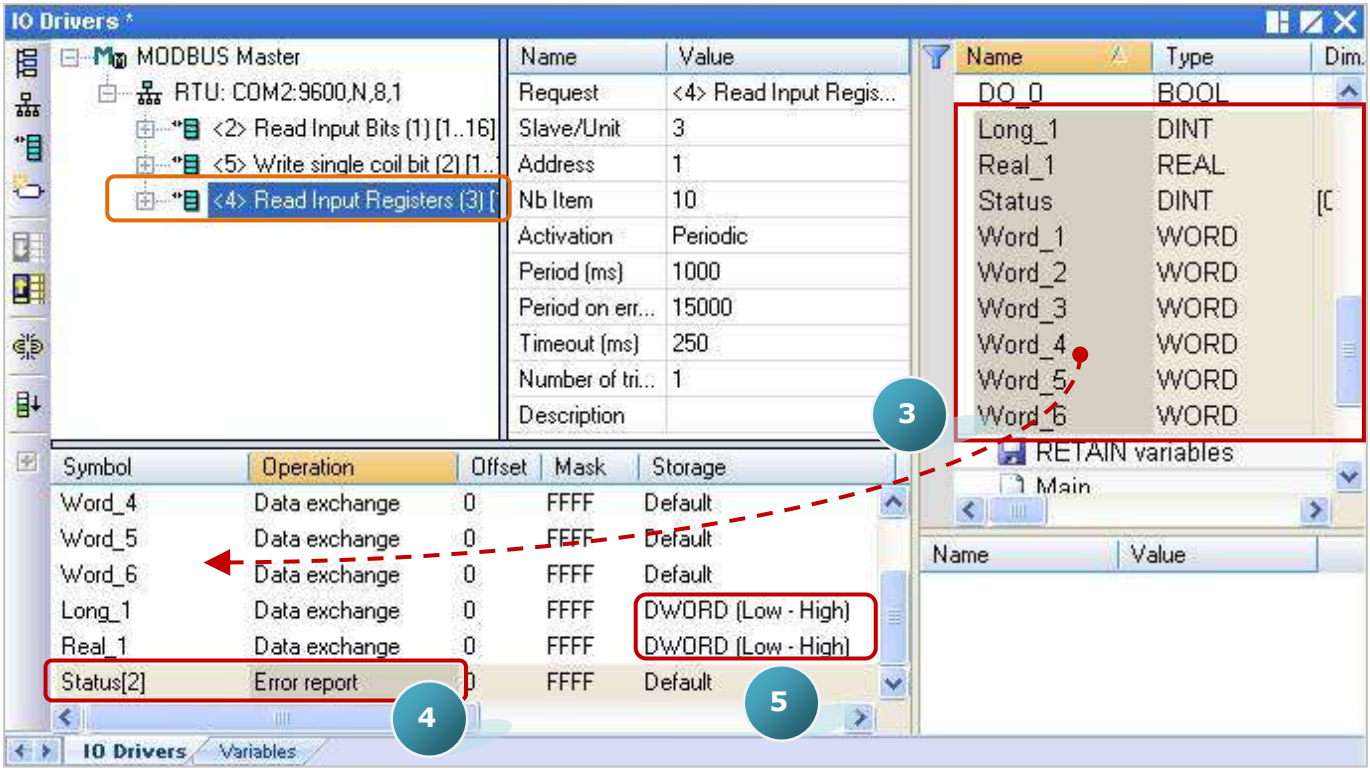
此處需設定 6 個 Word (16 bit)、1 個 Double Word (32 bit) 與 1 個 Real (32 bit) 變數。(設定方式可參考 2.3.1 節)，請依照下表來設定。

變數名稱	資料型態	說明
Word_1 ~ Word_6	WORD	用來讀取 AI 資料 (16 bit)。
Long_1	DINT	用來讀取 AI 資料 (32 bit)。
Real_1	REAL	用來讀取 AI 資料 (32 bit)。

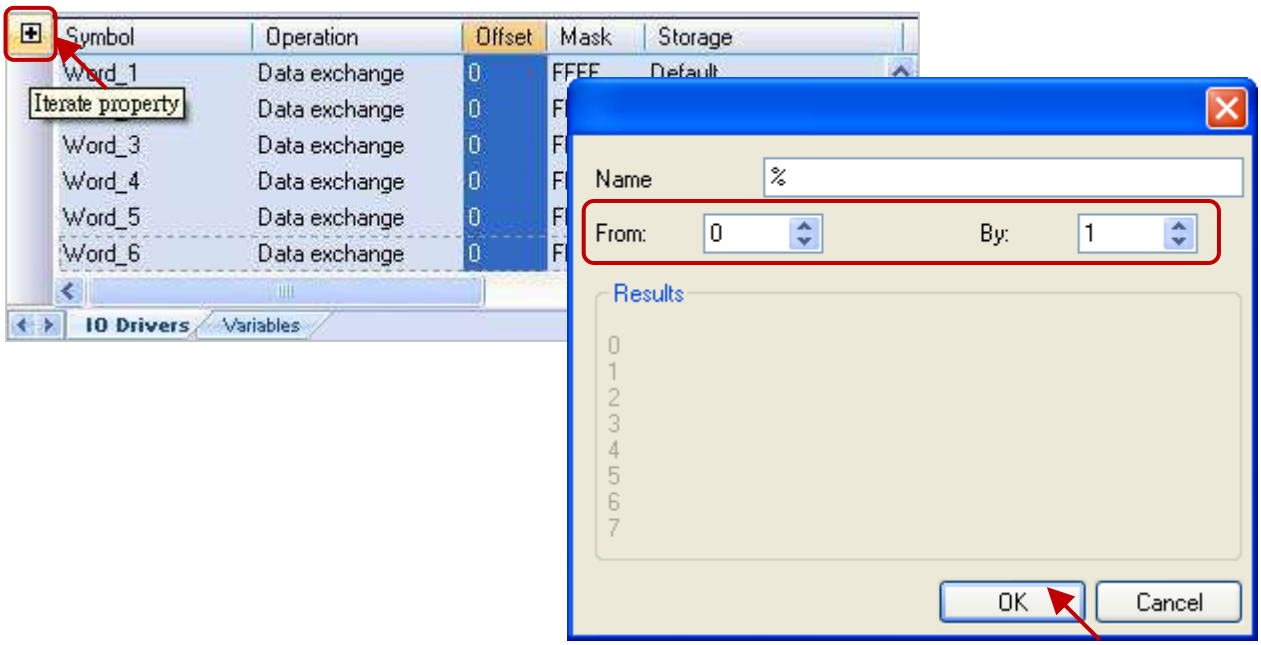
您可參考 附錄 A 來了解變數的資料形態與範圍，設定完成後，畫面如下。



3. 於 “I/O Drivers” 視窗，請將變數區中的變數 (“Word_1 ~ Word_6”、“Long_1”、“Real_1” 與 5.1.1 節中建立的 “Status”) 拖曳到第 3 個 Data Block 的 “Symbol” 區域。**注意:** “Status” 是一個陣列變數，拖曳到 “Symbol” 區域會是 “Status[0] ~ Status[4]"，請刪除 “Status[0] ~ [1]” 與 “Status[3] ~ [4]"。
4. 設定 “Status[2]” 的 “Operation” 為 “Error report” (表示讀取失敗時，該變數值為一個 “Error Code”，讀取成功時則會重置為 “0”)，按 “F1” 鍵則可查看 Modbus Master 設定說明，於標題 “Status and command variables” 中有詳細的命令、“Error Code” 說明。
5. “Long_1”、“Real_1” 為 32-bit 資料 (一個資料需占用 2 個 Modbus 位址)，設定其 “Storage” 為 “DWORD (Low – High)”。

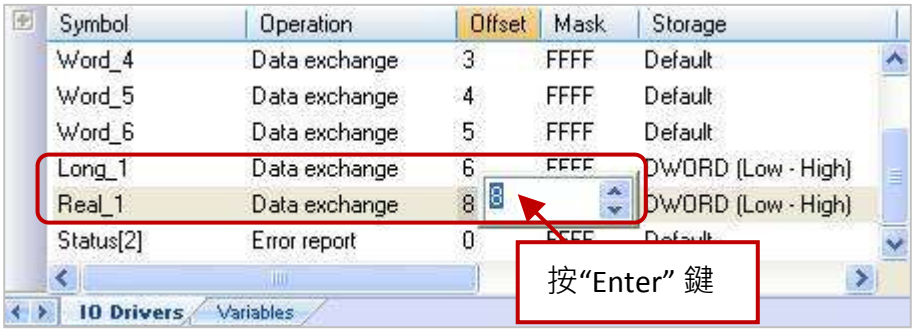


6. 如圖，選取 "Word_1 ~ Word_6" 並點選 "Iterate property" 設定 Offset 值 (From: 0；By: 1)。



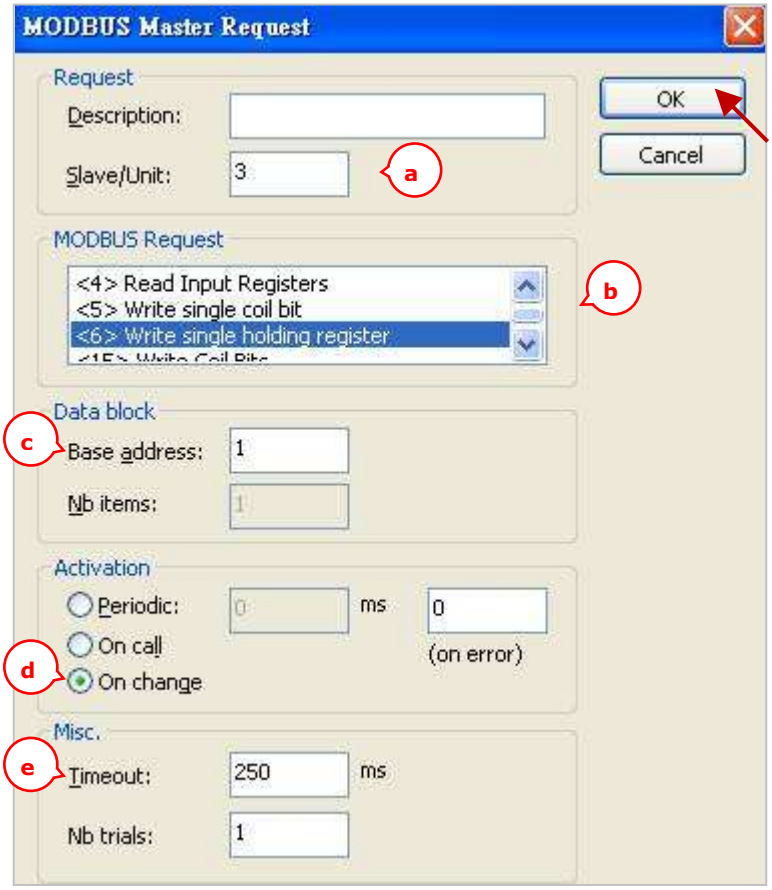
7. 接著，滑鼠雙擊 "Long_1"、"Real_1" 的 Offset 輸入其值為 "6"、"8" 並按 "Enter" 鍵完成設定。

   註: 32-bit 資料需占用 2 個 Modbus 位址，例如下方 "Long_1" 的 Offset 為 "6"，則下一個 "Real_1" 的 Offset 值 需設定為 "8"。



5.1.4 寫出 AO 資料 (16-bit)

1. 參考 5.1 節 - 步驟 4，建立第 4 個 Data Block，於 “MODBUS Master Request” 設定視窗中，設定以下項目並於完成後按 “OK"。



a. Slave/Unit:
填入 Slave 設備的站號 (Net-ID，此例為 “3”)。

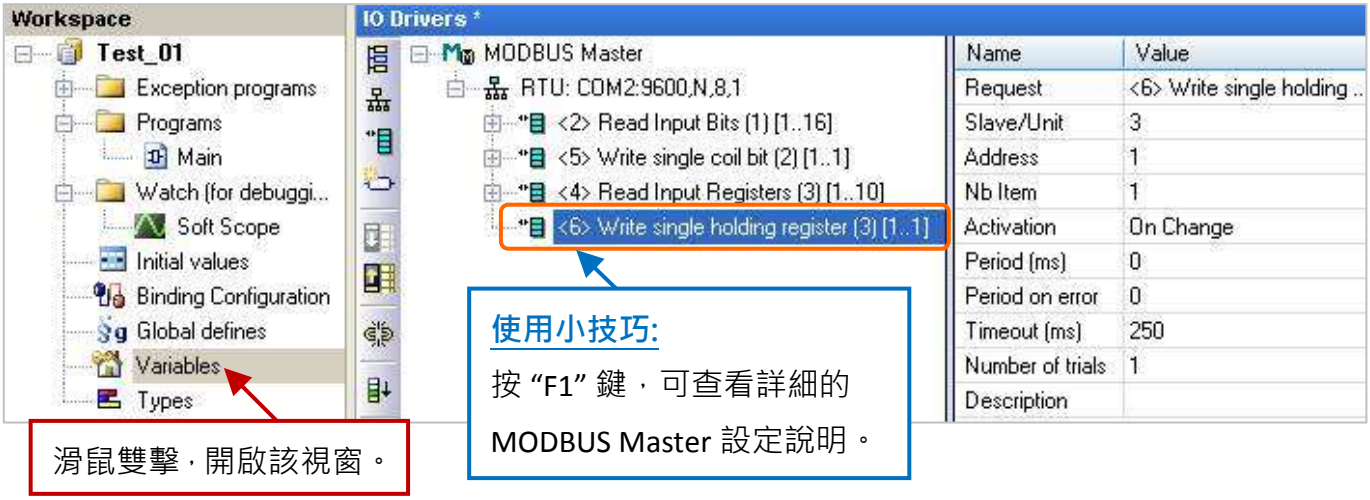
b. MODBUS Request: 選擇 “<6> Write single holding register”。

c. Base address:
預設值由 1 開始。
(如需修改其值，可參考 5.1.1 節。)

d. On change: 表示寫出的資料有改變時，才進行發送一次。
(其它項目說明，可參考 5.1.1 節。)

e. Timeout:
設定多久未回應，即表示異常。
(對於 Modbus RTU/ASCII 建議值: 200 ~ 1000 ms；此例為 250 ms)

2. 接著，請開啟 “Variables” 視窗，設定需使用的變數。

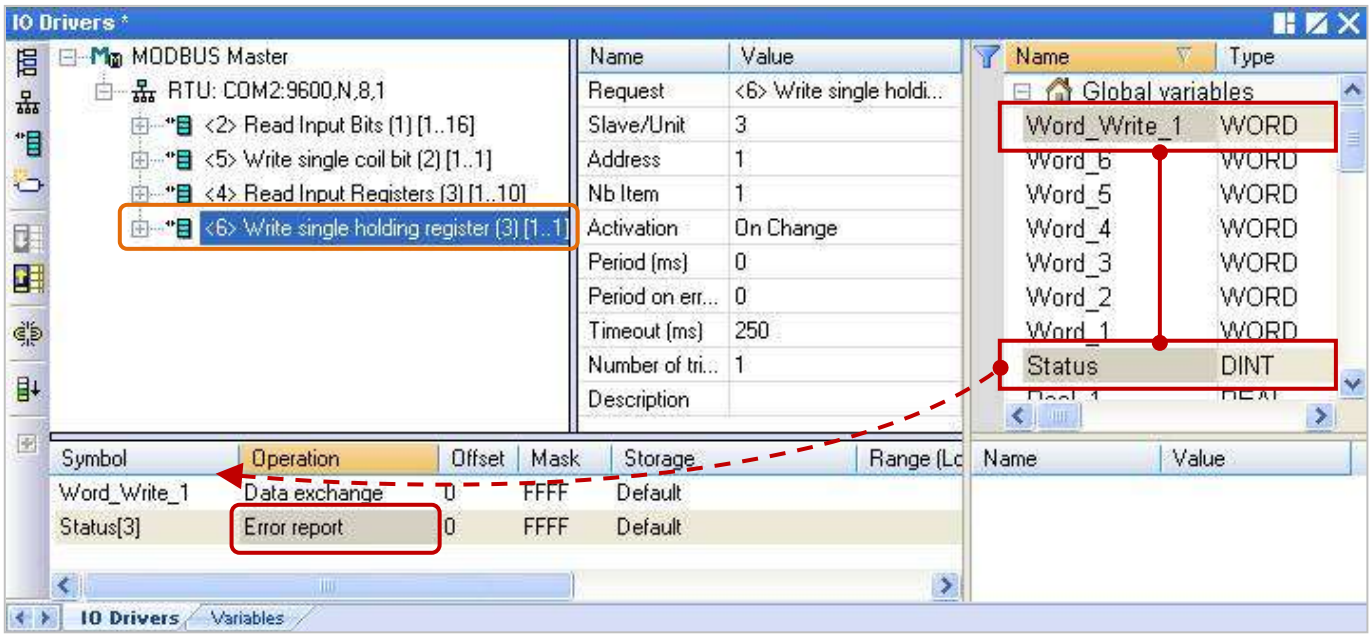


設定 1 個數值變數，您可參考 附錄 A 來了解變數的資料形態與範圍 (設定方式可參考 2.3.1 節)。

變數名稱	資料型態	說明
Word_Write_1	WORD	用來寫出 AO 資料 (16-bit)。

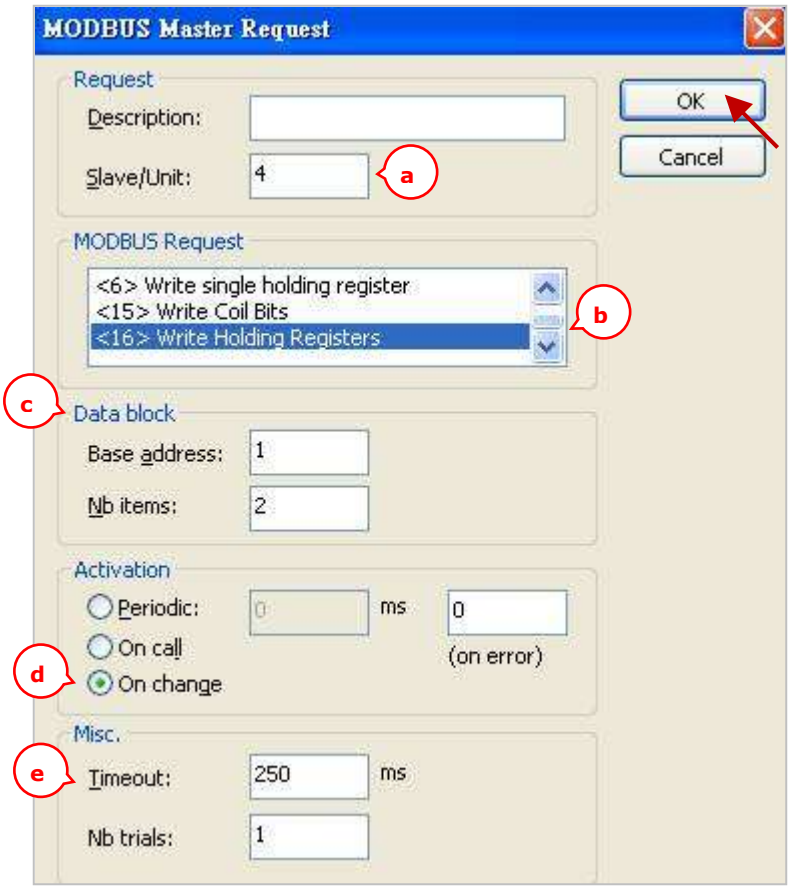
設定完成後，畫面如下。

3. 於 “I/O Drivers” 視窗，請將變數區中的變數 (“Word_Write_1” 與 5.1.1 節中建立的 “Status”) 拖曳到第 4 個 Data Block 的 “Symbol” 區域。**注意:** “Status” 是一個陣列變數，拖曳到 “Symbol” 區域會是 “Status[0] ~ Status[4]"，請按 “Delete” 鍵來刪除 “Status[0] ~ [2]” 與 “Status[4]"。
4. 設定 “Status[3]” 的 “Operation” 為 “Error report” (表示讀取失敗時，該變數值為一個 “Error Code”，讀取成功時則會重置為 “0”)，按 “F1” 鍵則可查看 Modbus Master 設定說明，於標題 “Status and command variables” 中有詳細的命令、“Error Code” 說明。



5.1.5 寫出 AO 資料 (32-bit)

1. 參考 5.1 節 - 步驟 4，建立第 5 個 Data Block，於 “MODBUS Master Request” 設定視窗中，設定以下項目並於完成後按 “OK"。



a. Slave/Unit:
填入 Slave 設備的站號 (Net-ID，此例為 “4”)。

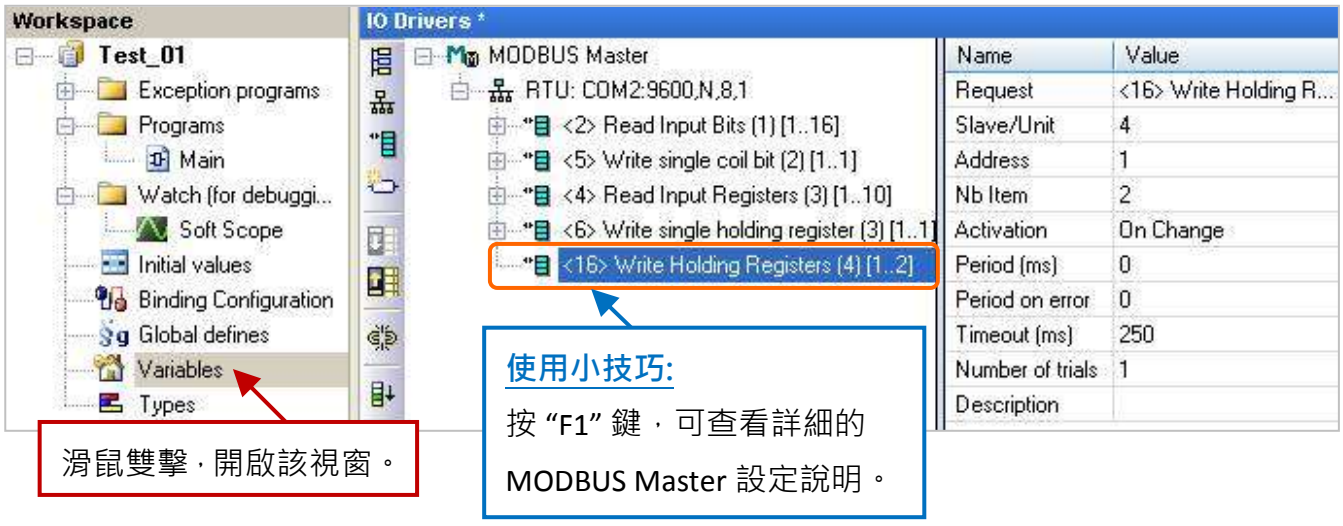
b. MODBUS Request: 選擇 “<16> Write Holding Registers”。

c. Base address:
預設值由 1 開始。
(如需修改其值，可參考 5.1.1 節。)
Nb items:
寫入 AO 的數量 (此例為 **“2”**，**占用 2 個 Modbus 位址**)。

d. On change: 表示寫出的資料有改變時，才進行發送一次。
(其它項目說明，可參考 5.1.1 節。)

e. Timeout:
設定多久未回應，即表示異常。
(對於 Modbus RTU/ASCII 建議值: 200 ~ 1000 ms；此例為 250 ms)

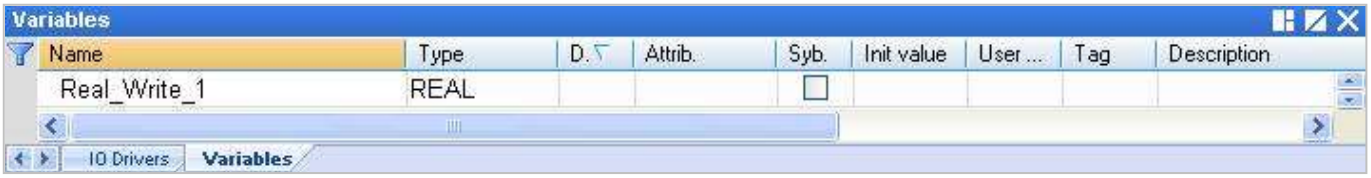
2. 接著，請開啟 “Variables” 視窗，設定需使用的變數



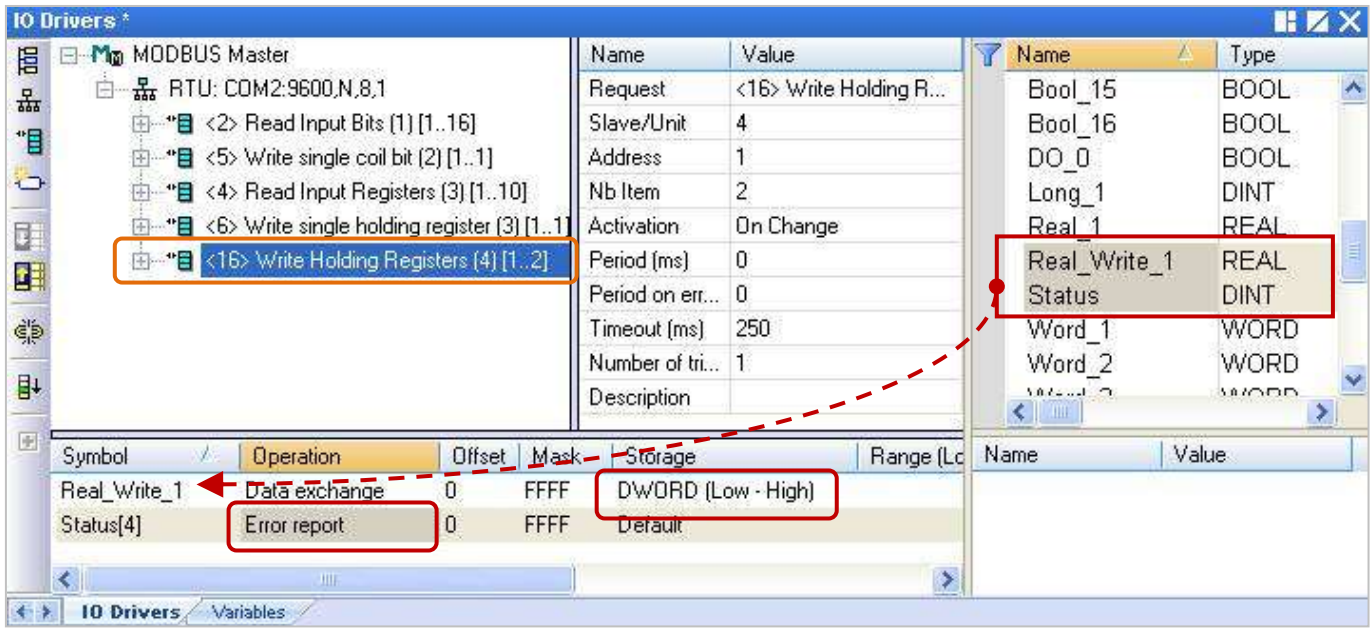
設定 1 個實數變數，您可參考 附錄 A 來了解變數的資料形態與範圍 (設定方式可參考 2.3.1 節)。

變數名稱	資料型態	說明
Real_Write_1	REAL	用來寫出 AO 資料 (32-bit)。

設定完成後，畫面如下。

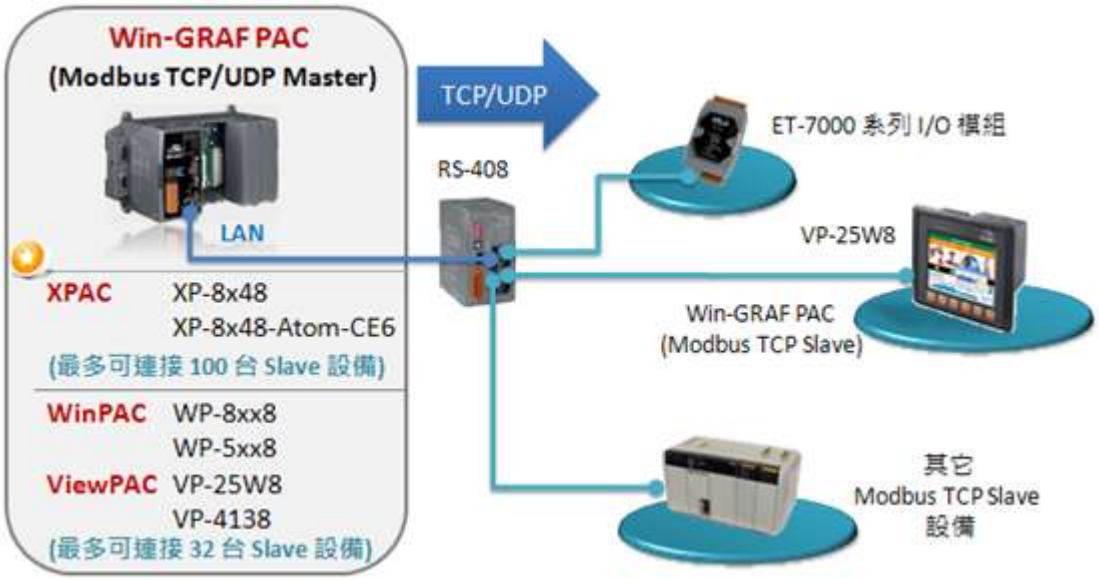


3. 於 “I/O Drivers” 視窗，請將變數區中的變數 (“Real_Write_1” 與 5.1.1 節中建立的 “Status”) 拖曳到第 5 個 Data Block 的 “Symbol” 區域。**注意:** “Status” 是一個陣列變數，拖曳到 “Symbol” 區域會是 “Status[0] ~ Status[4]"，請刪除 “Status[0] ~ [3]”。
4. 設定 “Status[4]” 的 “Operation” 為 “Error report” (表示讀取失敗時，該變數值為一個 “Error Code”，讀取成功時則會重置為 “0”)，按 “F1” 鍵則可查看 Modbus Master 設定說明，於標題 “Status and command variables” 中有詳細的命令、“Error Code” 說明。
5. “Real_Write_1” 為 32-bit 資料 (一個資料需占用 2 個 Modbus 位址)，設定其 “Storage” 為 “DWORD (Low – High)”。



5.2 啟用 Win-GRAF PAC 為 Modbus TCP/UDP Master

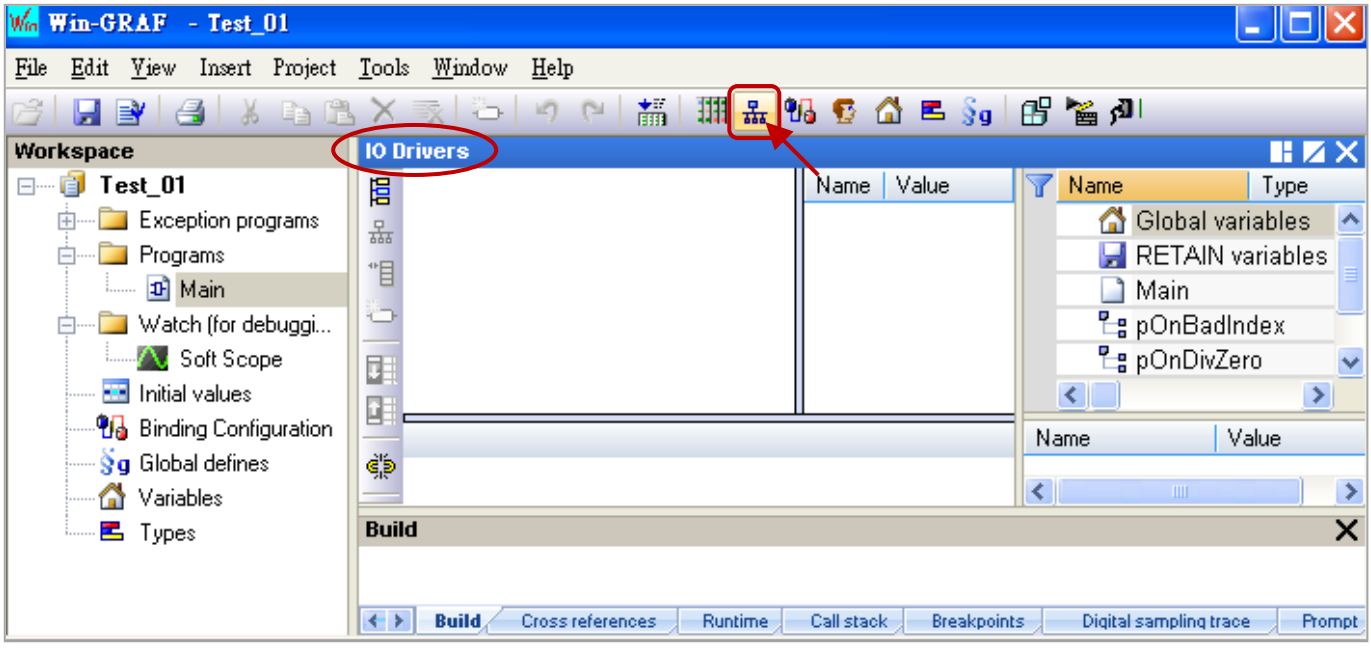
應用示意圖:



(您可參考 P1-1，來查詢詳細的 PAC 型號)

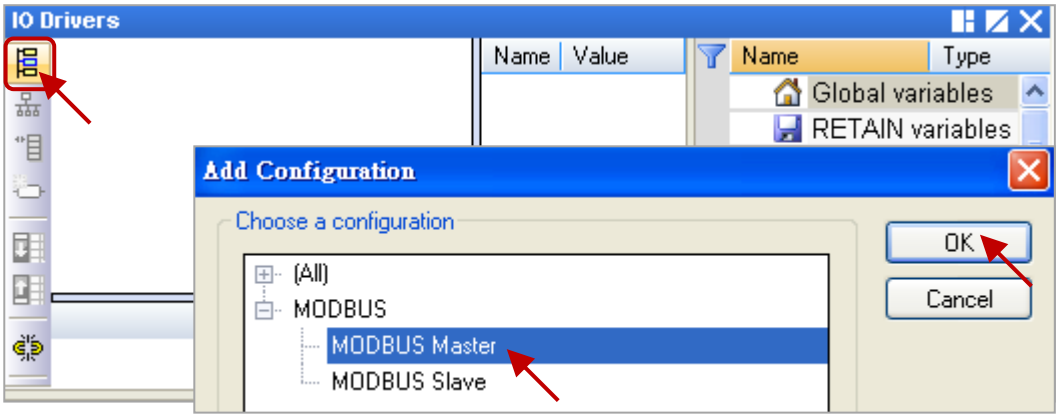
請參考以下操作步驟:

1. 滑鼠點選工具列上的 “Open Fieldbus Configuration” 按鈕來開啟 “I/O Drivers” 視窗。



2. 點選 “I/O Drivers” 視窗左側的 “Insert Configuration” 按鈕，再點選 “MOSBUS Master” 並點選 “OK” 來啟用一個 Modbus Master 設定。

註: 一個 “Modbus Master” 可有多個 Port 設定 (參考下一步驟)，可設定為 Modbus Master RTU/ASCII Port (參考 5.1 節) 或是 Modbus Master TCP/UDP Port，也可設定是否啟用該設定。



3. 點選左側的 “Insert Master/Port” 按鈕，開啟設定視窗並選擇 “MODBUS on Ethernet” 選項。設定以下項目後，再點選 “OK”。

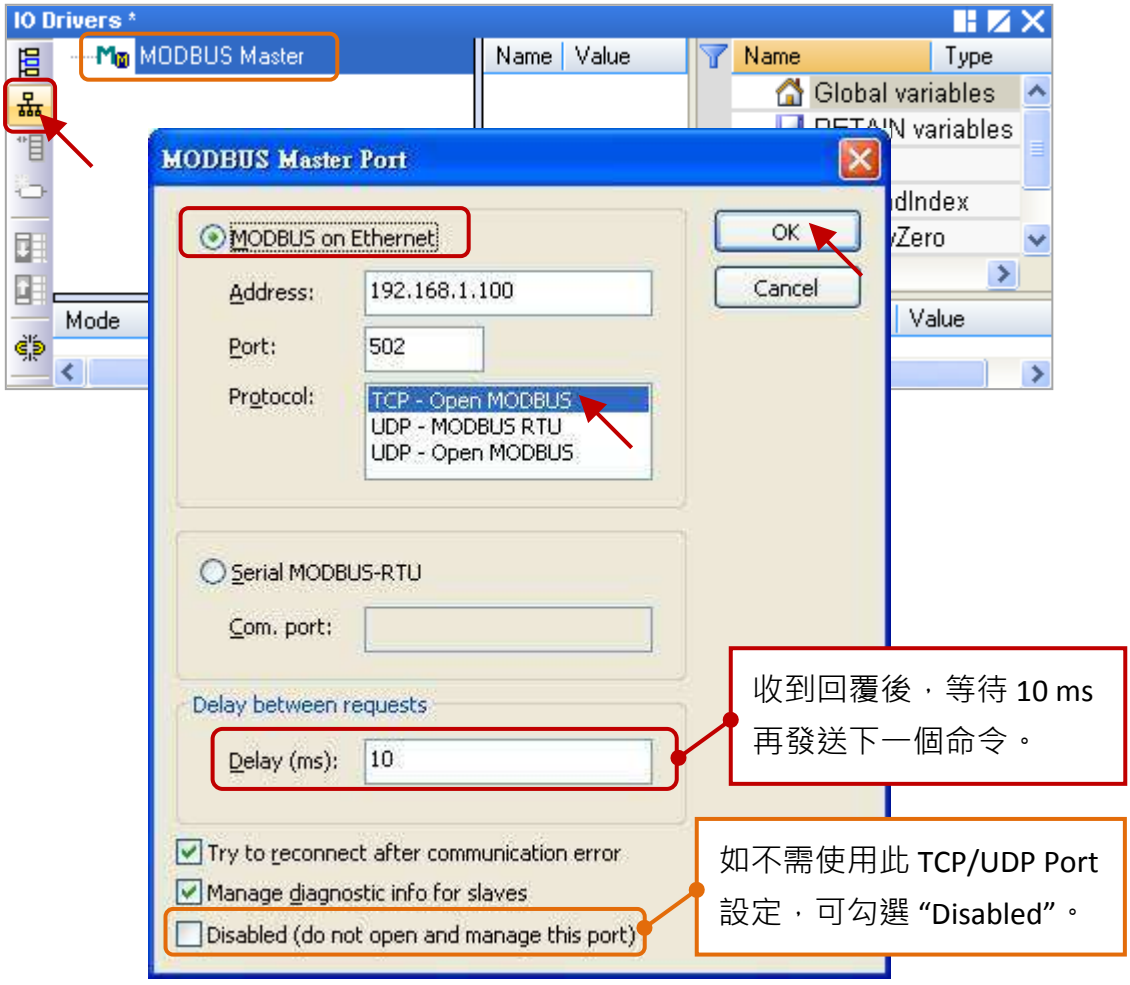
   Address: 填入 Modbus Slave 設備的 IP 位址。(例如: “192.168.1.100”)

   Port: 固定填入 “502”。

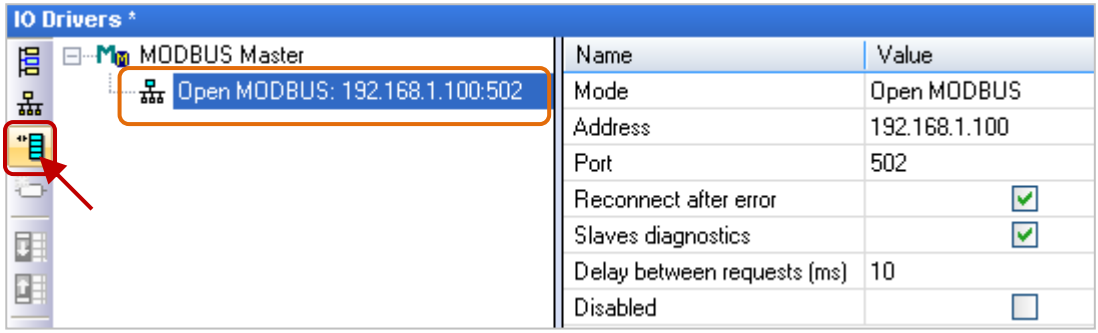
   Protocol: 若為 Modbus TCP Master，請選擇 “TCP – Open MODBUS”。

   若為 Modbus UDP Master，請選擇 “UDP – Open MODBUS”。

   Delay: 填入命令的間隔時間 (例如: 10 ms，可設為 0 ~ 10000)。



4. 點選左側的 “Insert Slave/Data Block” 按鈕，來建立一個 Data Block。



讀取 AI 資料

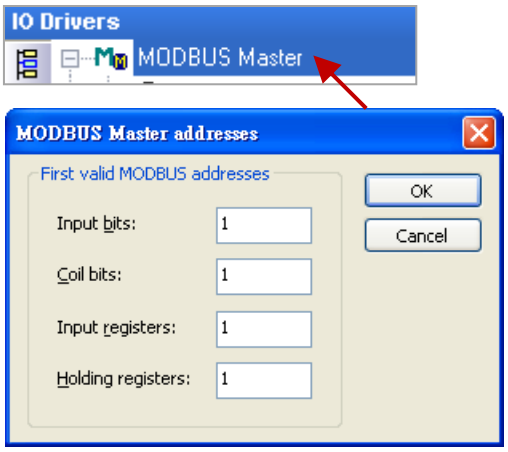
5. 於 “MODBUS Master Request” 設定視窗中，設定以下項目並於完成後按 “OK"。

a. Slave/Unit:
填入 Slave 設備的站號 (Net-ID，通常為 “1”)。

b. MODBUS Request:
選擇 “<4> Read Input Registers”。

c. Base address:
預設值由 1 開始。
Nb items:
讀取 AI 的數量 (此例為 4)。

註: 如需修改 “Base address”，可使用滑鼠右鍵點選 “MODBUS Master” 再選擇 “MODBUS Master Addresses” 修改其值。



d. Activacation: 表示 Modbus Request 發送的方式。

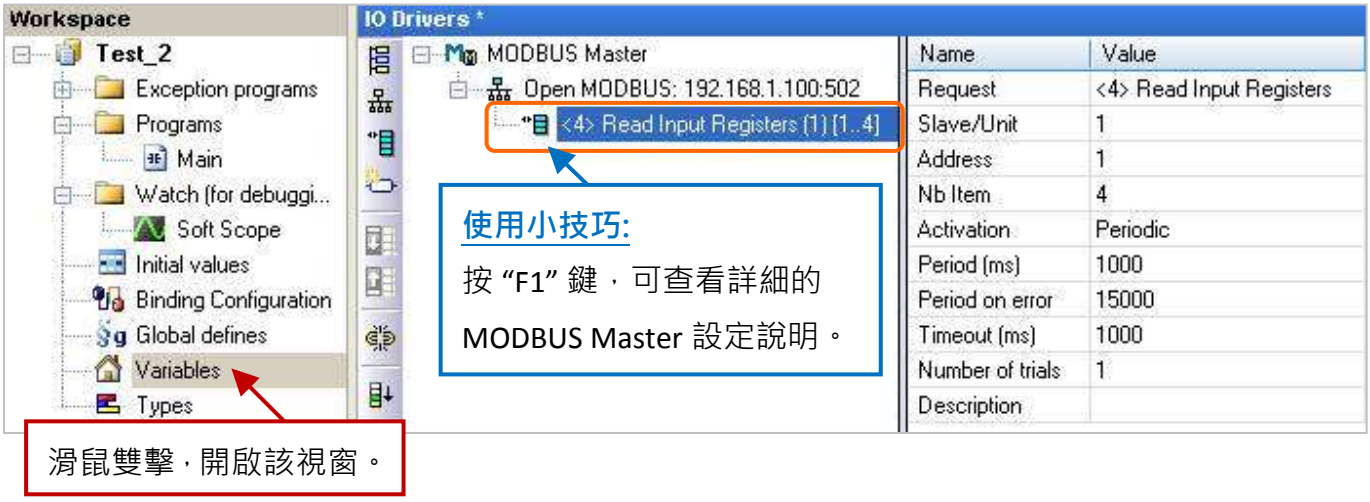
Periodic: 表示週期性的發送，此例為每 2 秒發送一次。“on error” 表示每當發生異常時，下一次的發送時間 (此例為 15 秒)。

On call: 表示程式有呼叫時，才進行發送一次。

On change: 表示寫出的資料有改變時，才進行發送一次。

e. Timeout: 設定多久未回應，即表示異常。
(對於 Modbus TCP/UDP 建議值: 1000 ~ 3000 ms；此例為 1000 ms)

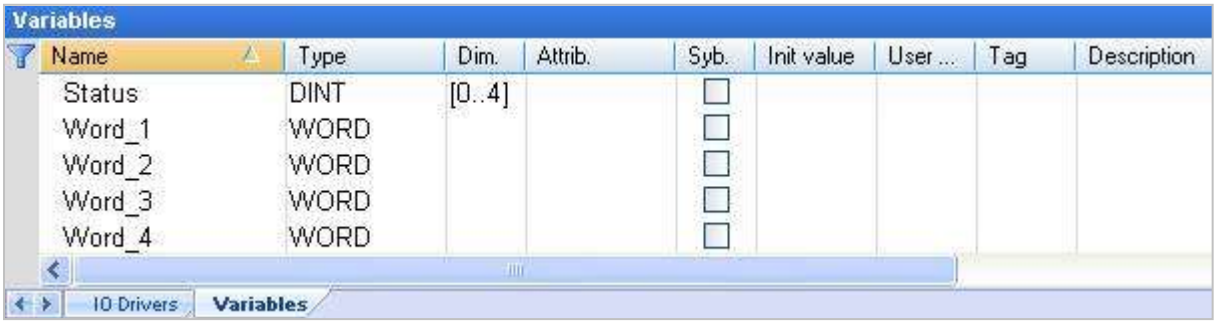
6. 接著，請開啟 “Variables” 視窗，設定需使用的變數。



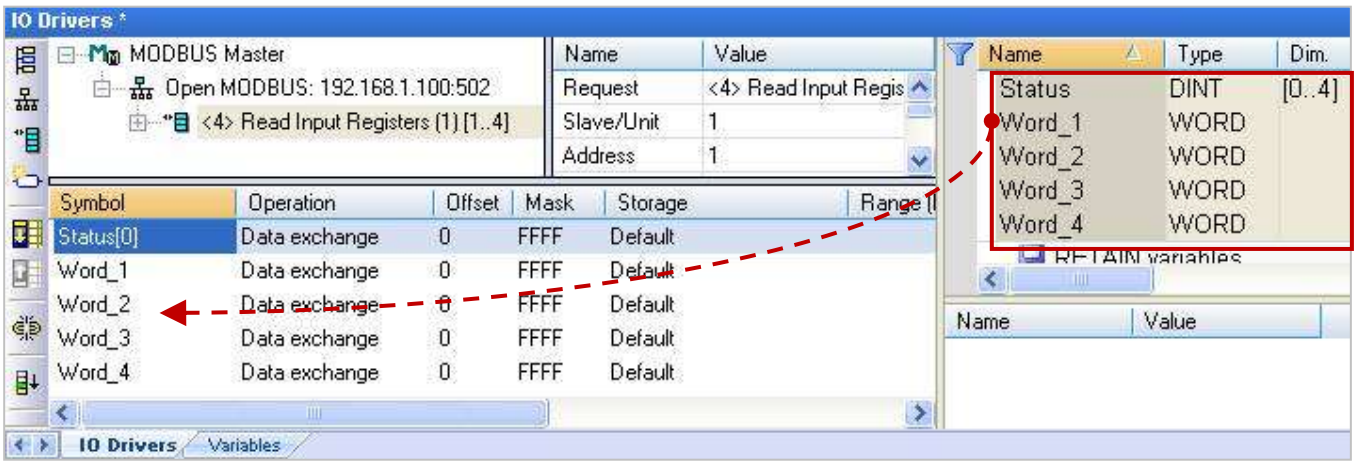
此處需設定 4 個 Word (16 bit) 變數 (設定方式可參考 2.3.1 節)，請依照下表來設定。

變數名稱	資料型態	Dim.	說明
Word_1 ~ Word_4	WORD	---	用來讀取 AI 資料 (16 bit)。
Status	DINT	5	用來記錄資料的存取狀況。

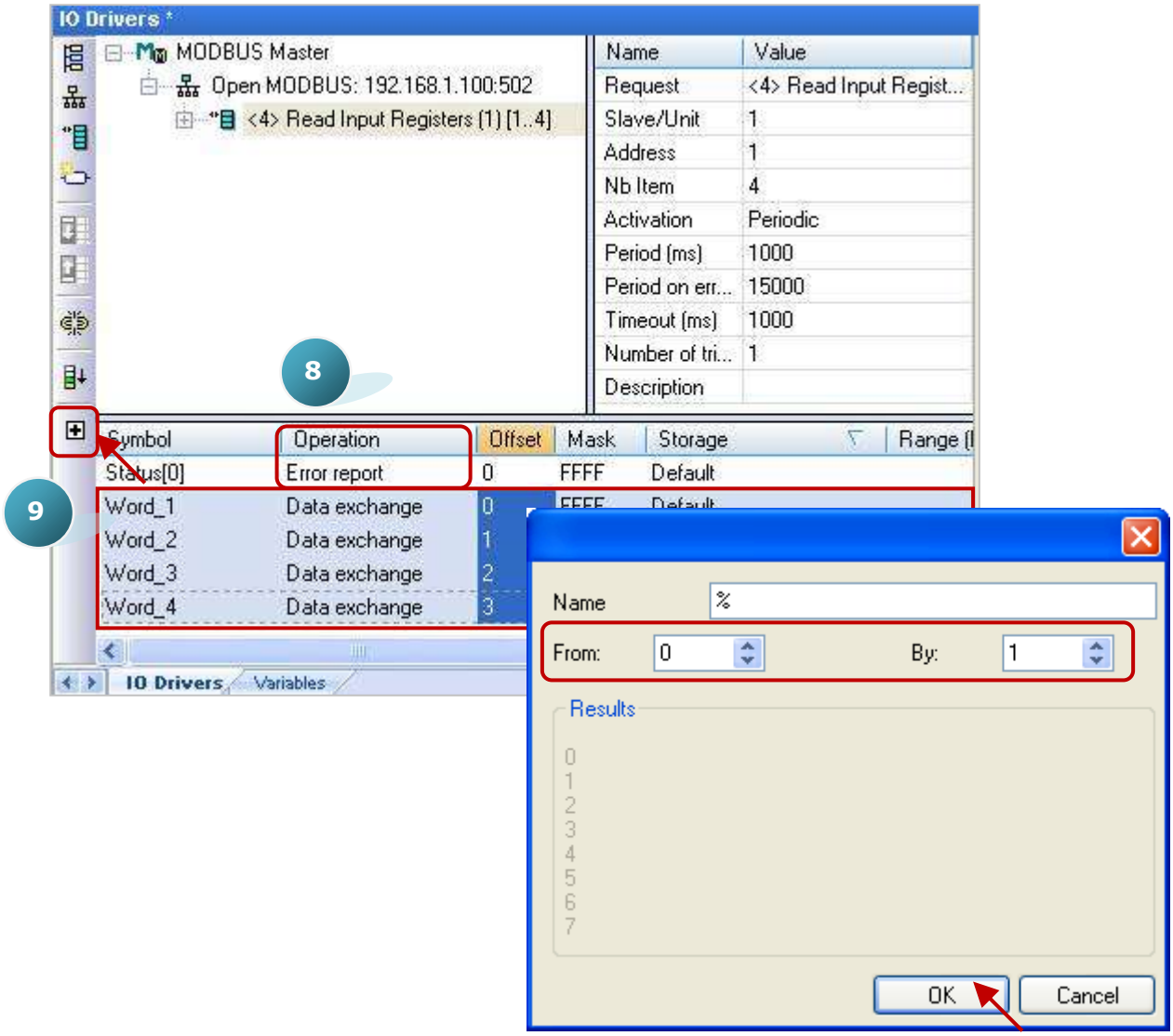
您可參考 附錄 A 來了解變數的資料形態與範圍，設定完成後，畫面如下。



7. 於 “I/O Drivers” 視窗，請將變數區中的變數 (“Word_1 ~ Word_4” 與 “Status”) 拖曳到 Data Block 的 “Symbol” 區域。**注意:** “Status” 是一個陣列變數，拖曳到 “Symbol” 區域會是 “Status[0] ~ Status[4]"，請按 “Delete” 鍵刪除 “Status[1] ~ [4]"。



8. 設定 “Status[0]” 的 “Operation” 為 “Error report” (表示讀取失敗時，該變數值為一個 “Error Code”，讀取成功時則會重置為 “0”)，按 “F1” 鍵則可查看 Modbus Master 設定說明，於標題 “Status and command variables” 中有詳細的命令、“Error Code” 說明。
9. 選取 "Word_1 ~ Word_4” 並點選 “Iterate property” 設定 Offset 值 (From: 0；By: 1)。



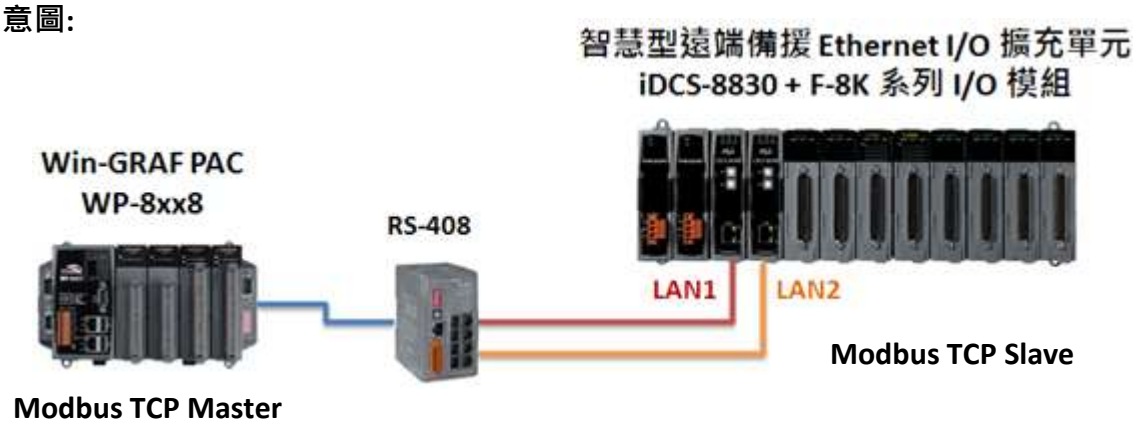
無論是 Modbus Master RTU/ASCII Port (參考 5.1 節) 或是 Modbus Master TCP/UDP Port，設定 “Modbus Master Request” 的方式是相同的，以上已完成了讀取 AI 資料的設定，其它的讀/寫方式可點選項目連結，來參考 5.1.1 ~ 5.1.5 節的範例內容。

項目	Function Code	Modbus Request	說明
1	2	Read Input Bits	讀取 DI 資料
2	5	Write single coil bit	寫出 DO 資料
3	4	Read Input Registers	讀取 AI 資料
4	6	Write single holding register	寫出 AO 資料 (16-bit)
5	16	Write Holding Registers	寫出 AO 資料 (32-bit)

5.3 連接具有 2 個 IP 位址的 Modbus TCP Slave 設備

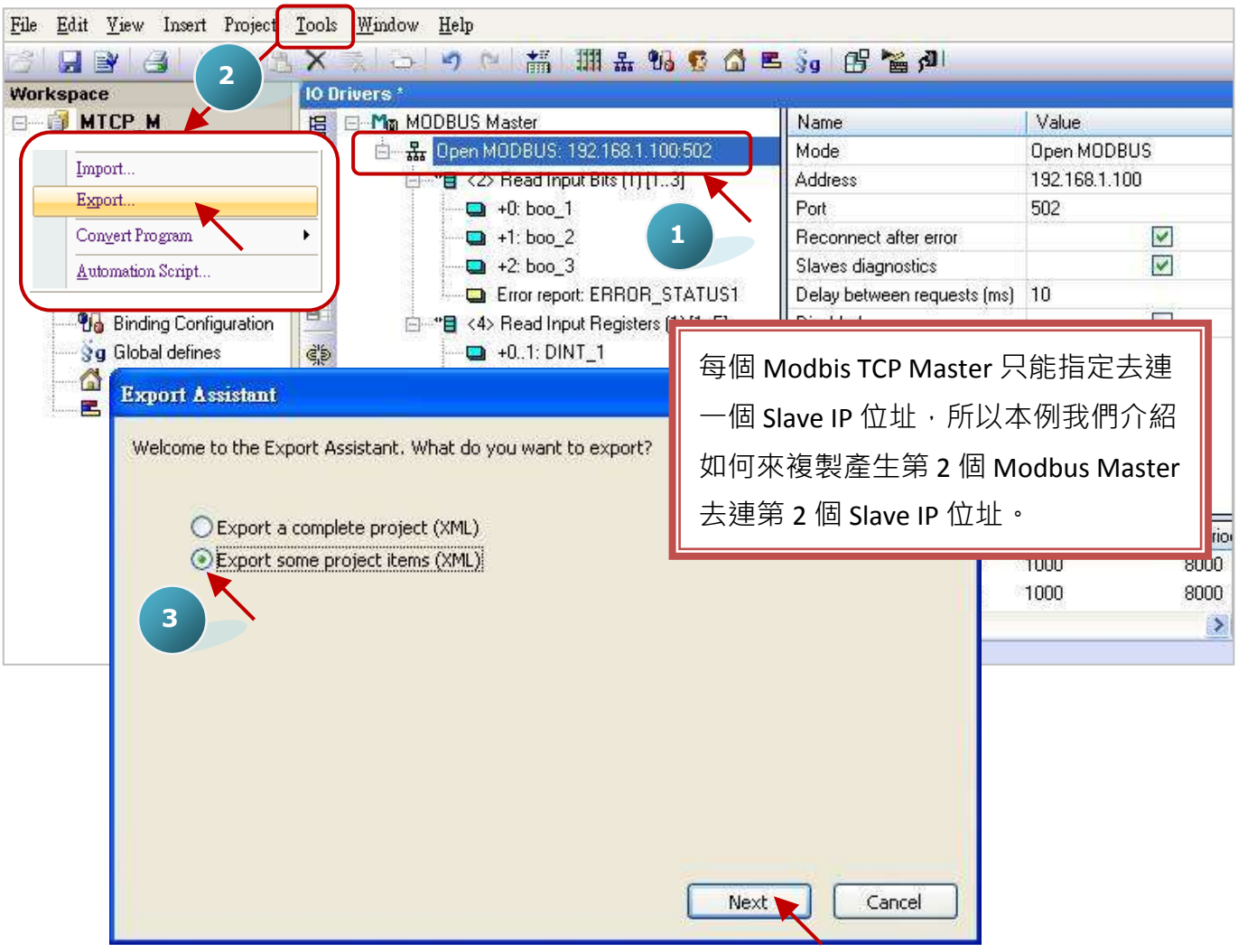
上一節說明了如何啟用 Win-GRAF PAC 為 Modbus TCP Master 設備，也說明了去讀/寫 Modbus TCP Slave 設備的方式。此章節將說明如何建立備援的 “Modbus Master Request” 設定，當 Modbus TCP Slave 設備的其中一個 IP 無法使用時，另一個 IP 仍會正常的讀/寫資料。

應用示意圖:

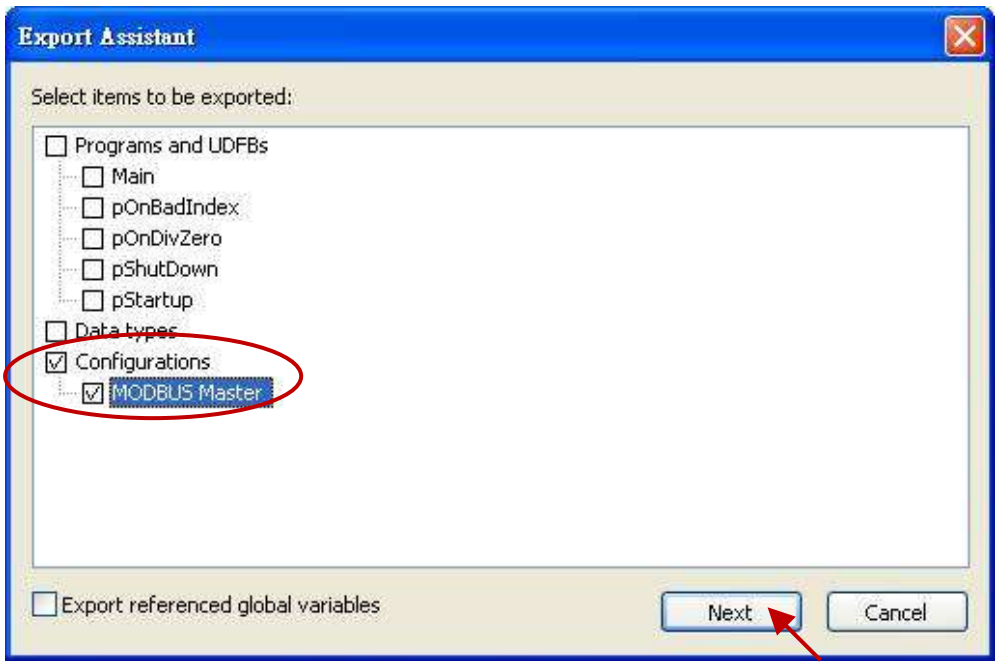


請參考以下操作步驟:

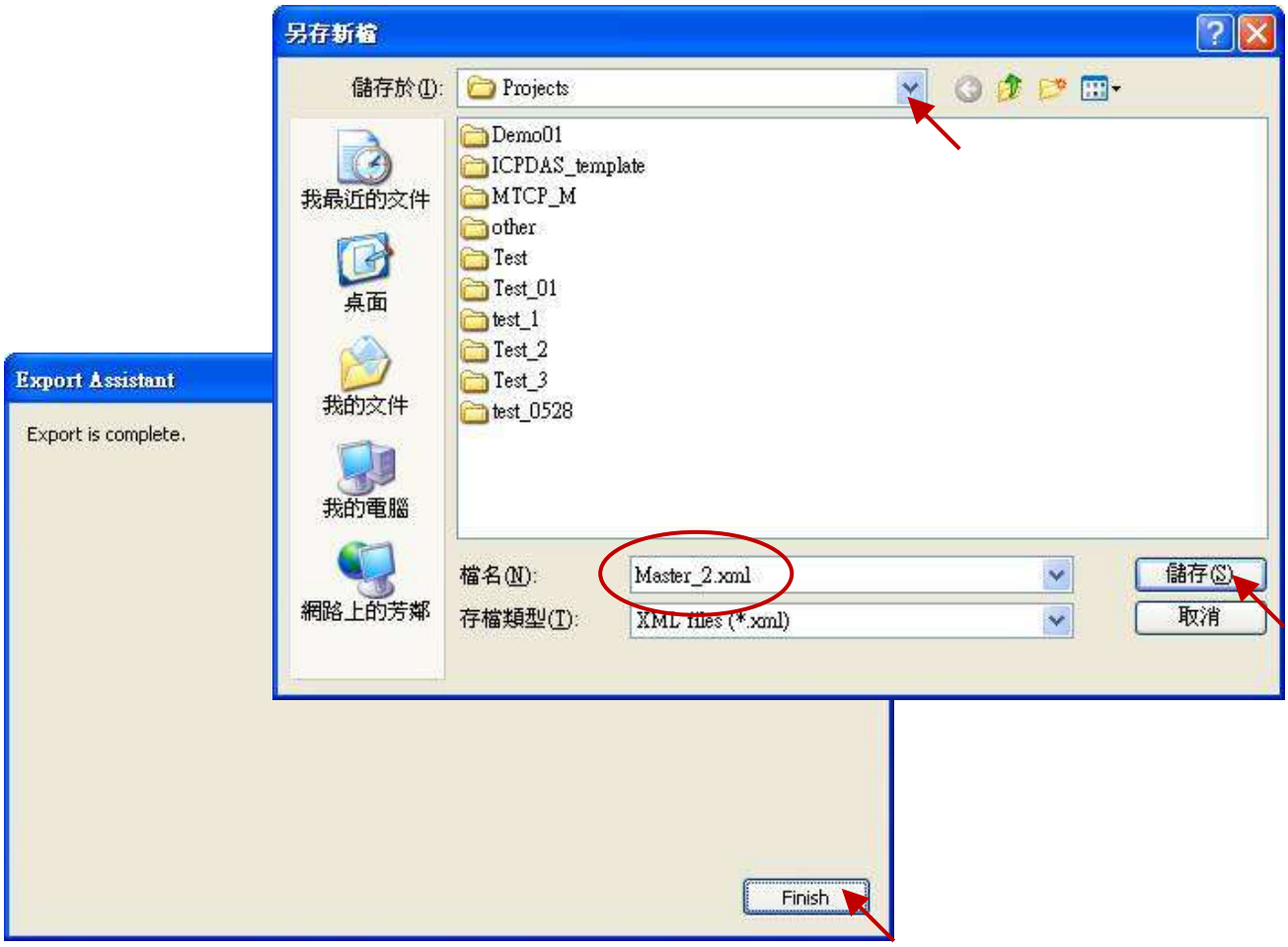
1. 滑鼠點選在 “Open MODBUS:”，再點選功能表 “Tools” 並選擇 “Export” 項目。
2. 於 “Export Assistant” 視窗中，點選 “Export some project items (XNL)” 再點選 “Next” 進行下一步。



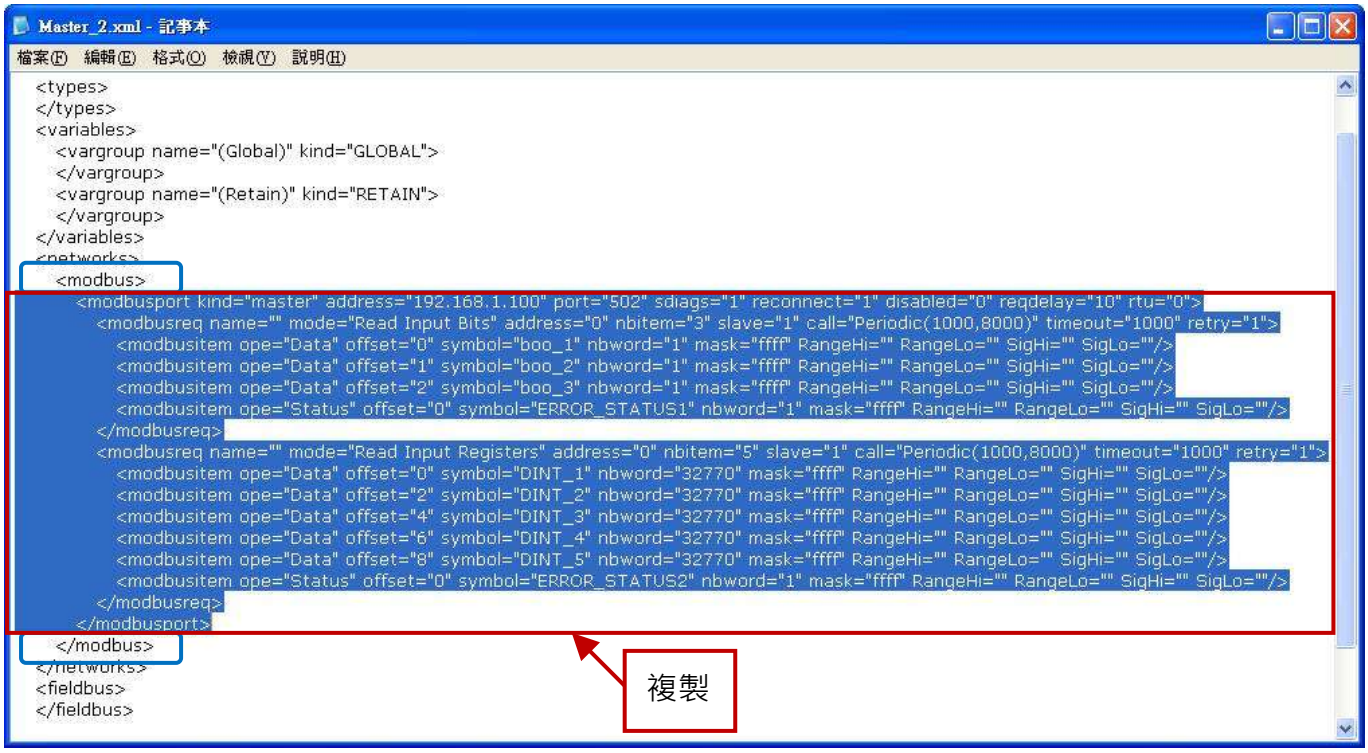
3. 取消其它勾選，僅保留 “Configurations” 勾選並點選 “Next” 進行下一步。



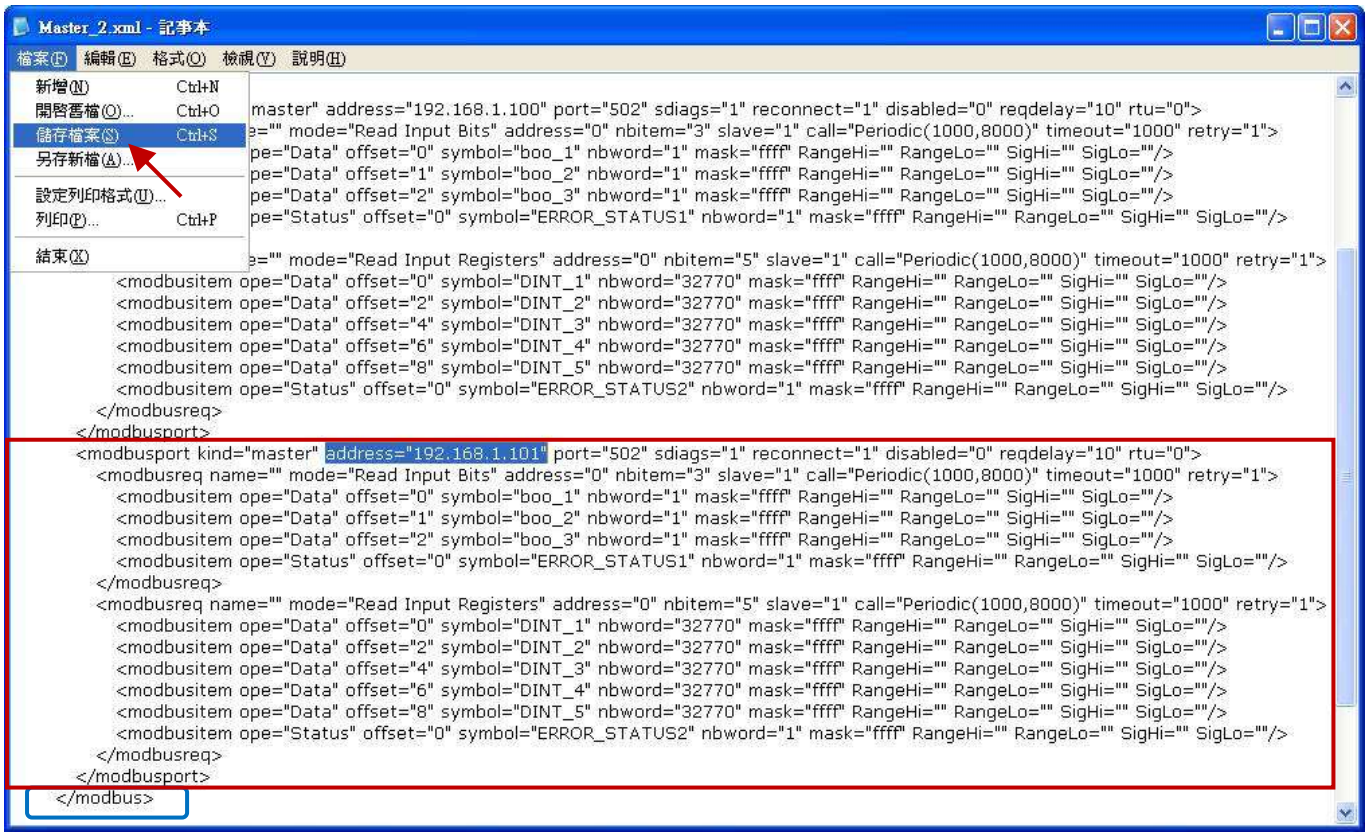
4. 尋找適合的路徑 (預設在 C:\Win-GRAF\Projects) 並為此檔案命名 (例如: Master_2.xml)，再點選 “儲存” 按鈕。最後，點選 “Finish” 完成匯出設定。



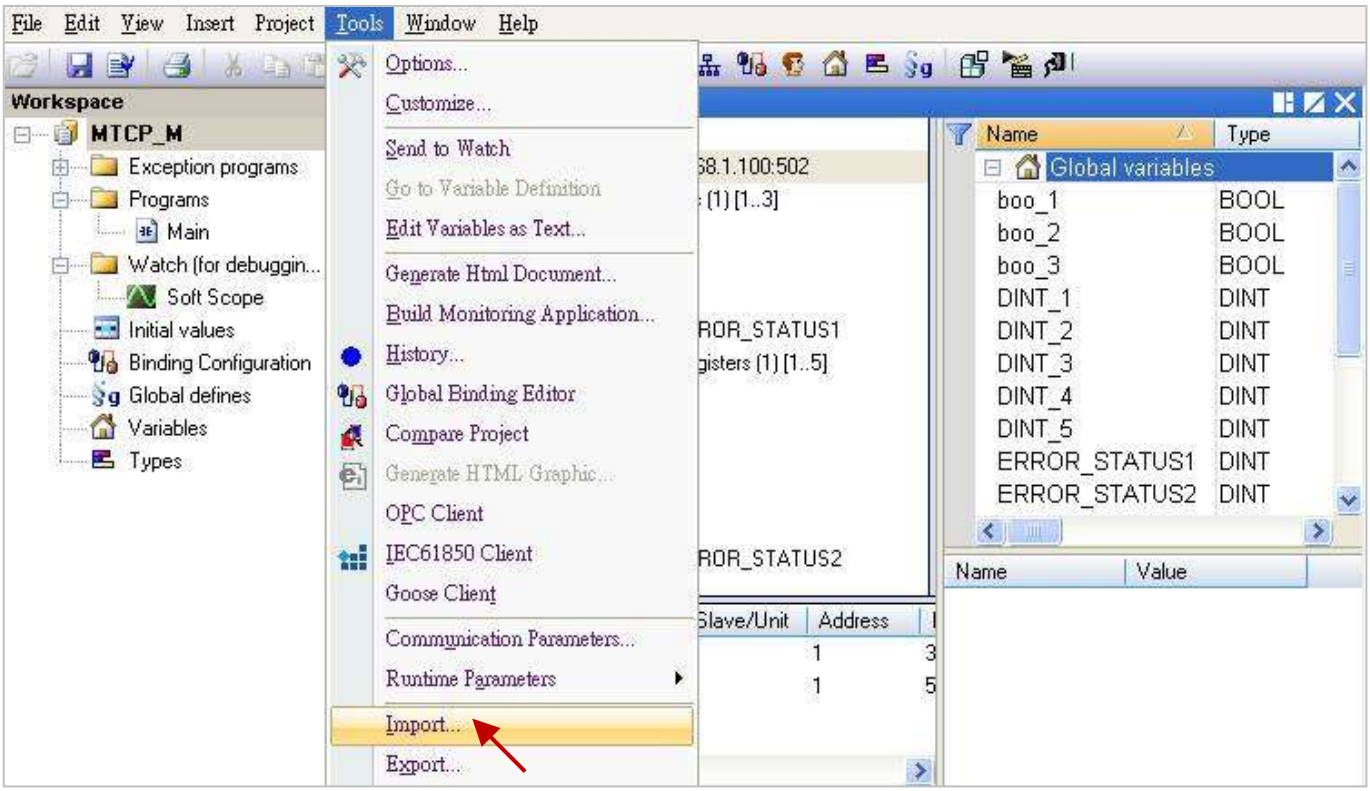
5. 以記事本 (Notepad) 開啟於步驟 4 匯出的 .xml 檔案，並複製 <modbus> 與 </modbus> 中間的內容。



6. 將複製的內容貼在 </modbus> 之上，並修改 address 為 Modbus Slave 設備的第二個 IP 位址 (例如: “192.168.1.101”)，再儲存並關閉檔案。

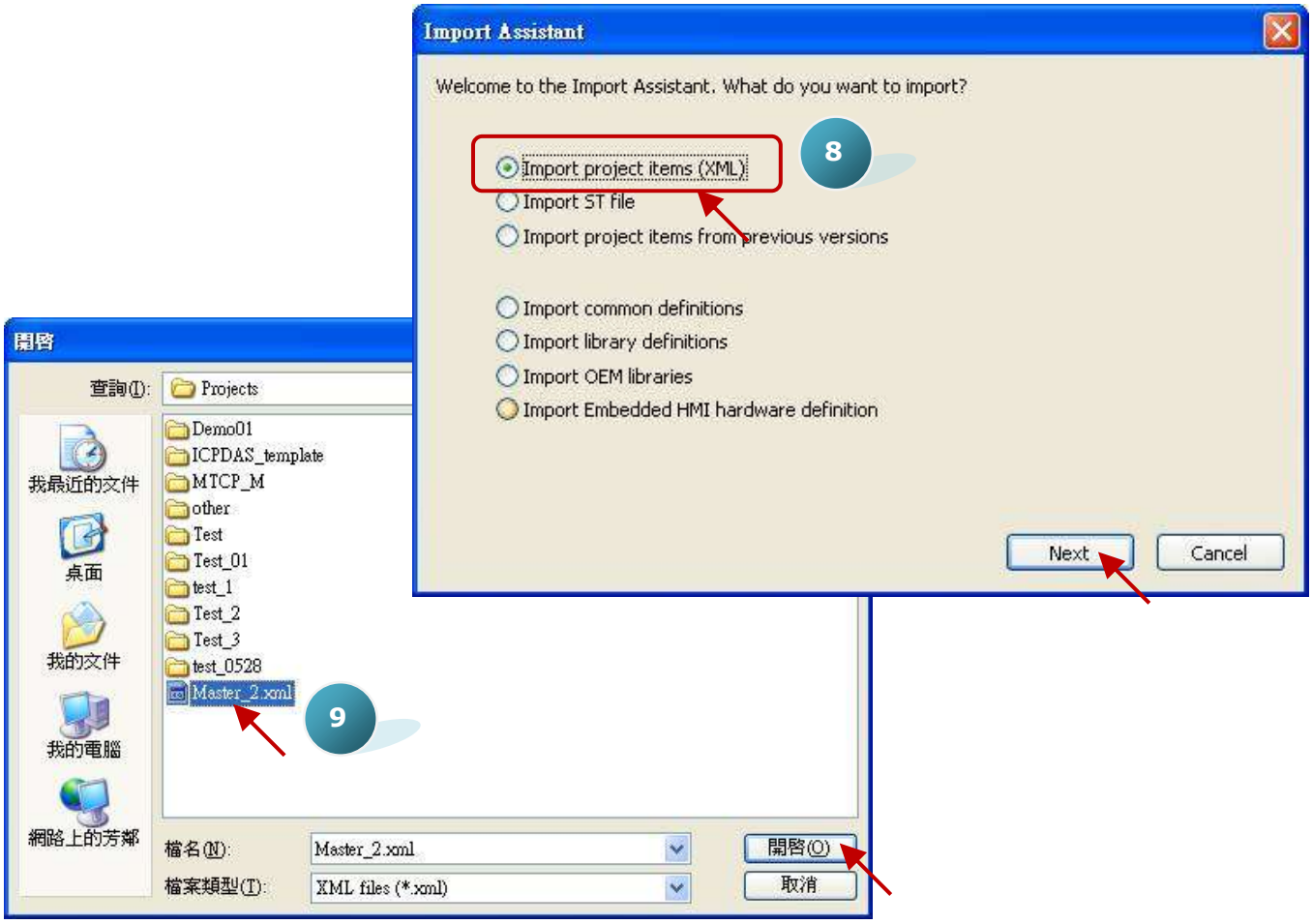


7. 點選 Win-GRAF 的功能表 “Tools” 並選擇 “Import” 項目。



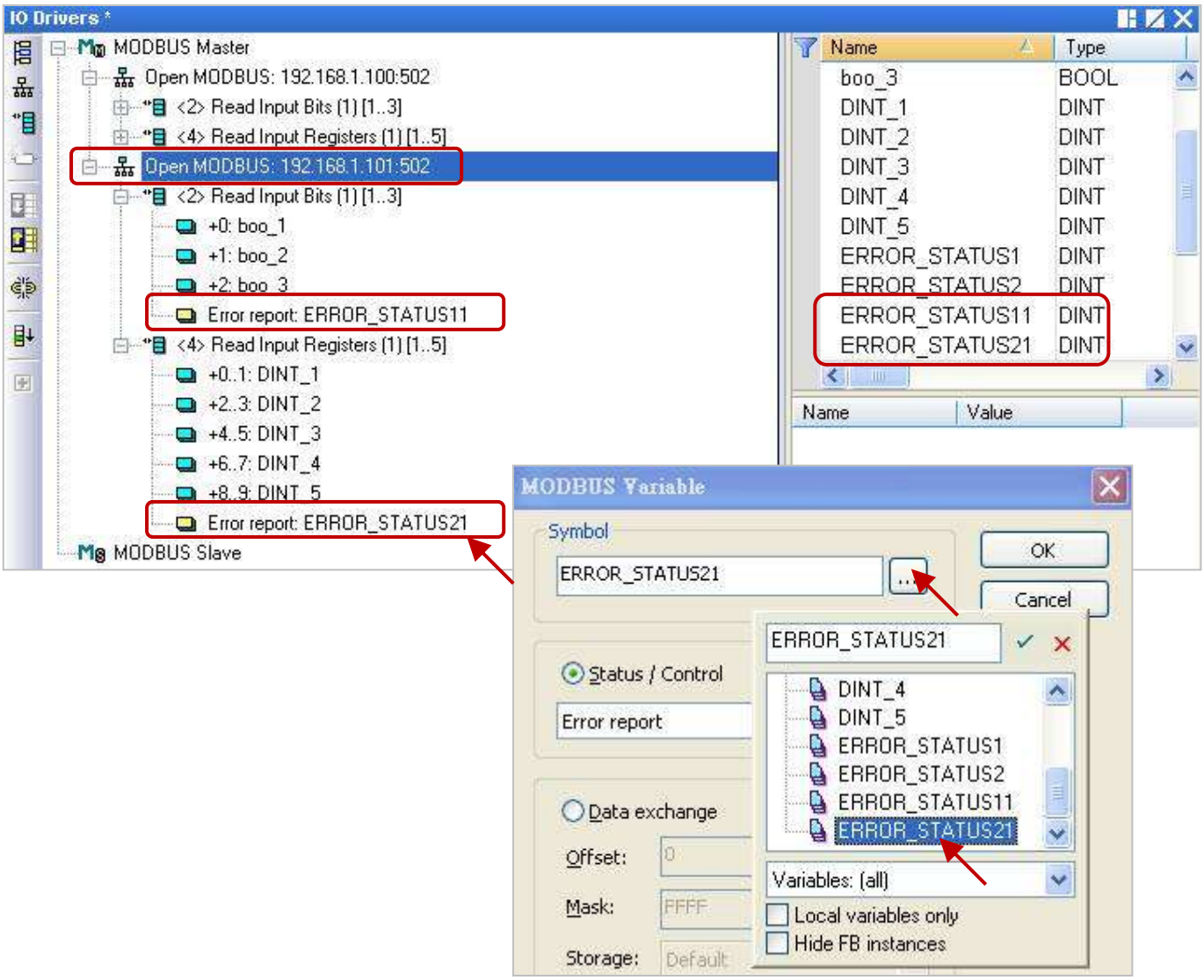
8. 於 “Import Assistant” 視窗中，點選 “Import project items (XNL)” 再點選 “Next” 進行下一步。

9. 選取欲匯入的檔案 (例如: “Master_2.xml”) 並點選“開啟” 按鈕。



10. 點選 “Finish” 完成匯入設定。

11. 在 “I/O Drivers” 視窗中，已新增了一個 “Open MODBUS” 設定，此設定包含了 2 個 “Modbus Master Request”，用來讀取 DI 與 AI，而其中 “Error report” 是用來表示 IP 的連線狀態，因此請在變數區新增 2 個 “DINT” 變數 (例如: “ERROR_STATUS11”, “ERROR_STATUS21”) 並滑鼠雙擊 “Error report” 來修改指定的變數。



第 6 章　可保存變數與資料儲存

6.1　可保存變數 (Retain Variable)

此章節將介紹如何使用 "RETAIN_VAR"、"RETAIN_ARY"、"RETAIN_FLAG_GET"、"RETAIN_FLAG_SET" 與 "RETAIN_FLAG_CLR" 函式 (Function)，Win-GRAF 系列 PAC 內建有一個資料保存記憶區，可供使用者保存變數資料，此資料不會因關機而消失，下次開機時仍為上一次的值。

注意: WP-5xx8 為掌上型的 PAC 並沒有資料保存記憶區 (Retain Memory)，無法使用以上函式。因此，請直接參考 6.2 節 的內容 。

於出貨光碟中 (\Napdos\Win-GRAF\demo-project)，有提供此章節的範例程式 (demo_retain.zip)，請參考 第 12 章 來回存此專案 (執行 File > Add Existing Project > From Zip) 並設定好 PAC 目前的 IP 位址。

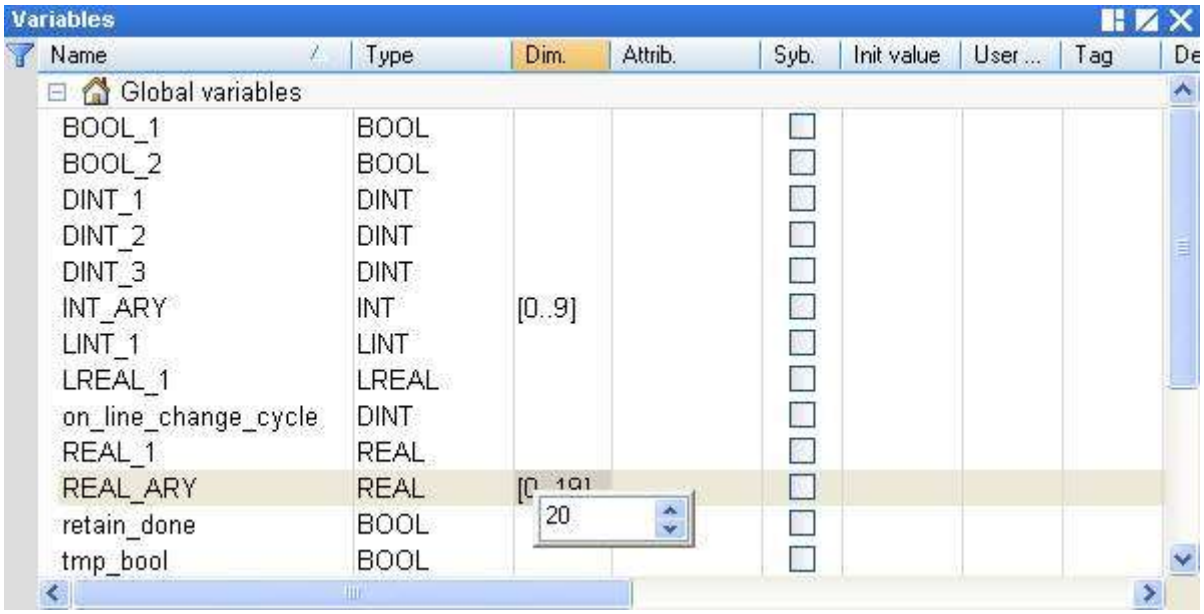
註: "Retain_Var()" 或 Retain_Ary() 函式，只能在第一個 PAC Cycle 或 執行線上更新 (On-line Change) 的那個 Cycle 內使用，於其它 Cycle 內呼叫此函式，將會回傳 "FALSE"。當保存變數尚未指定過任何初值時，PAC 程式執行此函式所讀到的值並無意義，使用者至少需為所有的保存變數指定適合的初值一次。

ST 語法: 此範例說明 Retain_Var() 與 Retain_Ary() 函式。

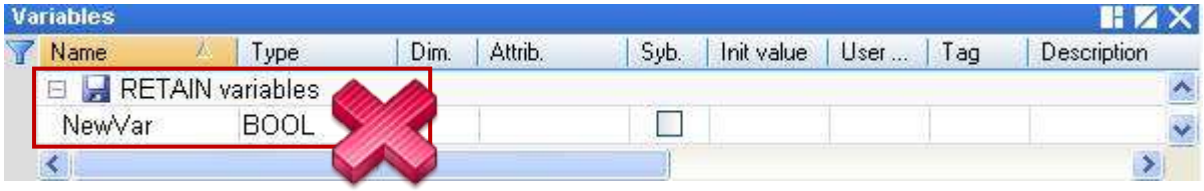
```
(* 宣告 "on_line_change_cycle" 為 "DINT" 變數，
   非 “0” 表示正在執行線上更新 (On-line Change) 的那個 Cycle 內。
   宣告 "retain_done" 為 "BOOL" 變數且設定初始值為 “FALSE” 一次。
   宣告 "tmp_bool" 為 "BOOL" 變數。
*)

on_line_change_cycle  :=  GetSysInfo (_SYSINFO_CHANGE_CYCLE) ;
if (retain_done = FALSE) or (on_line_change_cycle <> 0) then
      retain_done := TRUE ;  (* 只執行一次即可 *)

      tmp_bool := Retain_Var ( DINT_1 , 1) ; (* 設定保存一個 “DINT” 變數 *)
      tmp_bool := Retain_Var ( DINT_2 , 2) ;
      tmp_bool := Retain_Var ( REAL_1 , 3) ; (* 設定保存一個 “REAL” 變數 *)
      tmp_bool := Retain_Var ( BOOL_1 , 4) ; (* 設定保存一個 “BOOL” 變數 *)
      tmp_bool := Retain_Var ( BOOL_2 , 5) ;

      (* 設定保存 “INT” 陣列變數內的 10 個元素 *)
      tmp_bool := Retain_Ary ( INT_ARY ,  6 ,  10) ;
```

```
    (* 設定保存 “REAL” 陣列變數內的 20 個元素 *)
    tmp_bool := Retain_Ary ( REAL_ARY , 16 , 20) ;
    tmp_bool := Retain_Var ( DINT_3 , 36) ;
    (* 64-bit 變數只能使用位址編號 10001 ~ 12000 *)
    tmp_bool := Retain_Var ( LINT_1  , 10001) ;  (* 設定保存一個 “LINT” 變數 (64-bit) *)
    tmp_bool := Retain_Var ( LREAL_1 , 10002) ; (* 設定保存一個 “LREAL” 變數 (64-bit) *)
end_if ;
```

於 "Variables“ 視窗可查詢/設定變數，若您不熟悉變數的宣告方式，可參考 2.2.2 節 與 2.3.1 節。



注意: ICP DAS Win-GRAF PAC 並不支援 "Variable" 視窗中的 "Retain Variable“ 功能，因此請參考下列章節的 5 個函式來使用可保存變數功能。



6.1.1 RETAIN_VAR (設定保存一個變數)

使用小技巧:
按 “F1” 鍵，可查看詳細的設定說明。

Name:
想要保存資料的變數名稱 (勿使用陣列變數 或 字串)，資料型態可為 BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, LINT, ULINT, LWORD, LREAL。

Addr: (資料型態: “DINT”)
用來保存變數的位址編號，可設定為 1 ~ 12000。

Q: 資料型態為 “BOOL”，“TRUE”: 表示 OK ； “FALSE”: 表示錯誤。

註:

1. 一個位址只能儲存一個變數，**請勿**將相同的位址指定給兩個變數 (或多個)，否則保存值將會出現錯誤。
2. 64-bit 的資料型態 (例如: LINT, ULINT, LREAL, LWORD) 只能使用位址編號 10001 ~ 12000。
3. 其它資料型態 (例如: BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME) 可使用位址編號 1 ~ 12000。

6.1.2 RETAIN_ARY (設定保存一個陣列變數)

使用小技巧:
按 “F1” 鍵，可查看詳細的設定說明。

Name[]:
想要保存資料的陣列變數名稱 (勿使用字串 或 非陣列變數)，資料型態可為 BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, LINT, ULINT, LWORD, LREAL。

Addr: (資料型態: “DINT”)
用來保存陣列變數的起始位址編號，可設定為 1 ~ 12000。

Num: (資料型態: “DINT”)
陣列變數內想要保存的資料數量。
例如，若陣列變數有 100 個元素，設定 "Num" 為 "1 ~ 100" 是正確的，但若設為大於 100 就不對。
若陣列變數有 5 個元素，則設定 "Num" 為 "1 ~ 5" 是正確的，但若設為大於 5 就不對。

Q: 資料型態為 “BOOL”，“TRUE”: 表示 OK ； “FALSE”: 表示錯誤。

註:

1. 一個位址只能儲存一個變數 (或陣列中的一個元素)，**請勿**將相同的位址指定給兩個變數 (或多個)，否則保存值將會出現錯誤。
2. 64-bit 的資料型態 (例如: LINT, ULINT, LREAL, LWORD) 只能使用位址編號 10001 ~ 12000。
3. 其它資料型態 (例如: BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME) 可使用位址編號 1 ~ 12000。

6.1.3 RETAIN_FLAG_SET/GET/CLR (設定/取得/刪除 Flag 的狀態)

如何使用:

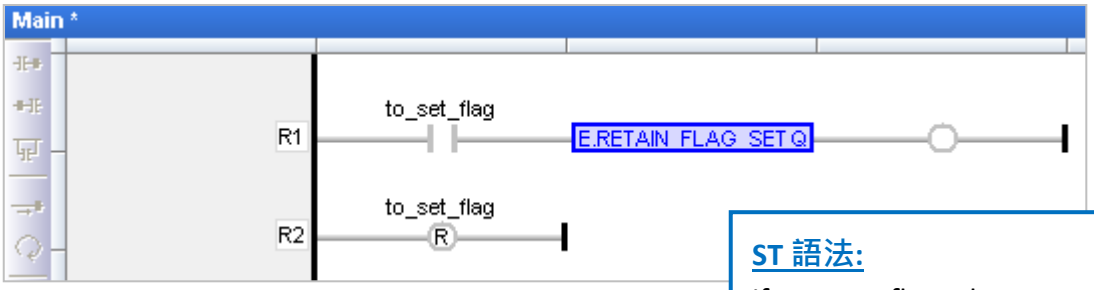
“RETAIN_FLAG” 是一個儲存在保存記憶體中的旗標 (TRUE / FALSE)，可讓使用者判斷保存資料是否為有效值。若先前保存變數未設置過任何初值，則 PAC 一開機從保存記憶體所讀到的值並不是正確的 (一般會是個亂數值)，因此為了讓程式可正常運作，使用者至少需為所有的保存變數設定過一次適當的值，然後可呼叫 "Retain_Flag_Set()" 將保存旗標設定為 "TRUE“，表示 "所有的保存變數已設置了適當的值"; 呼叫 "Retain_Flag_Get()" 可取得保存旗標的狀態; 呼叫 "Retain_Flag_Clr()" 則可刪除保存旗標的狀態。

ST 語法:

```
(*  宣告 "on_line_change_cycle" 為 "DINT" 變數，非 “0” 表示正在執行線上更新
    (On-line Change) 的那個 Cycle 內。
    宣告 "retain_done" 為 "BOOL" 變數且初始值為 “FALSE”。
    宣告 "tmp_bool"、"retain_flag" 與 "to_set_flag" 為 "BOOL" 變數。
*)
on_line_change_cycle  :=  GetSysInfo (_SYSINFO_CHANGE_CYCLE) ;

if (retain_done = FALSE) or (on_line_change_cycle <> 0) then
  retain_done := TRUE ;  (* 只執行一次即可 *)
  tmp_bool := Retain_Var( DINT_1 , 1) ;  (* 設定保存一個 “DINT” 變數 *)
  tmp_bool := Retain_Var( DINT_2 , 2) ;
  tmp_bool := Retain_Var( REAL_1 , 3) ; (* 設定保存一個 “REAL” 變數 *)
  tmp_bool := Retain_Var( BOOL_1 , 4) ; (* 設定保存一個 “BOOL” 變數 *)

  (* ... 在執行所有的 Retain 函式後 ... *)
  retain_flag := Retain_Flag_Get();
  if  (retain_flag = FALSE)  then
    (* 若先前保存變數並未設定過適當的值，您可在此執行一些適當的操作。*)
    (* ... *)
  end_if ;

end_if ;

 (* 當您為所有的保存變數都指定了適當的值後，請記得將以下的 "to_set_flag" 設定為
    “TRUE”，以便調用 "Retain_Flag_Set() 一次，如此，下次使用到 "Retain_Flag_Get()  時
    就會回傳 “TRUE”。 *)
 if  (to_set_flag = TRUE)  then
   to_set_flag := FALSE ;
   tmp_bool := Retain_Flag_Set() ;
 end_if ;
```

LD 語法: (按 “F1” 鍵，可查看詳細的設定說明。)

RETAIN_FLAG_SET : 設定保存旗標。



ST 語法:

```
if  to_set_flag  then
    to_set_flag  :=  FALSE  ;
    TMP_BOOL  :=  Retain_Flag_Set() ;
end_if ;
```

Q: 資料型態為 “BOOL”，只回傳 “TRUE”。

RETAIN_FLAG_GET : 取得保存旗標的狀態。

ST 語法:

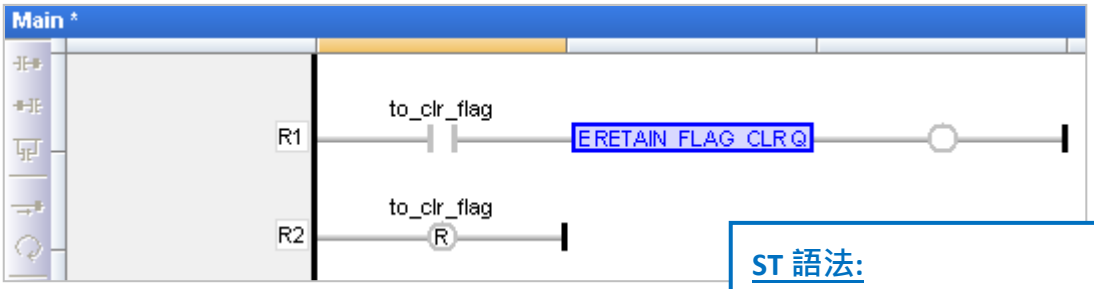
```
retain_flag  :=  Retain_Flag_Get() ;
```

Q: 資料型態為 “BOOL”，

“TRUE”: 表示已設立旗標 (Flag) ;

“FALSE”: 表示未設立旗標 (Flag)。

RETAIN_FLAG_CLR : 刪除保存旗標的狀態。



ST 語法:

```
if  to_clr_flag  then
    to_clr_flag  :=  FALSE  ;
    TMP_BOOL  :=  Retain_Flag_Clr() ;
end_if ;
```

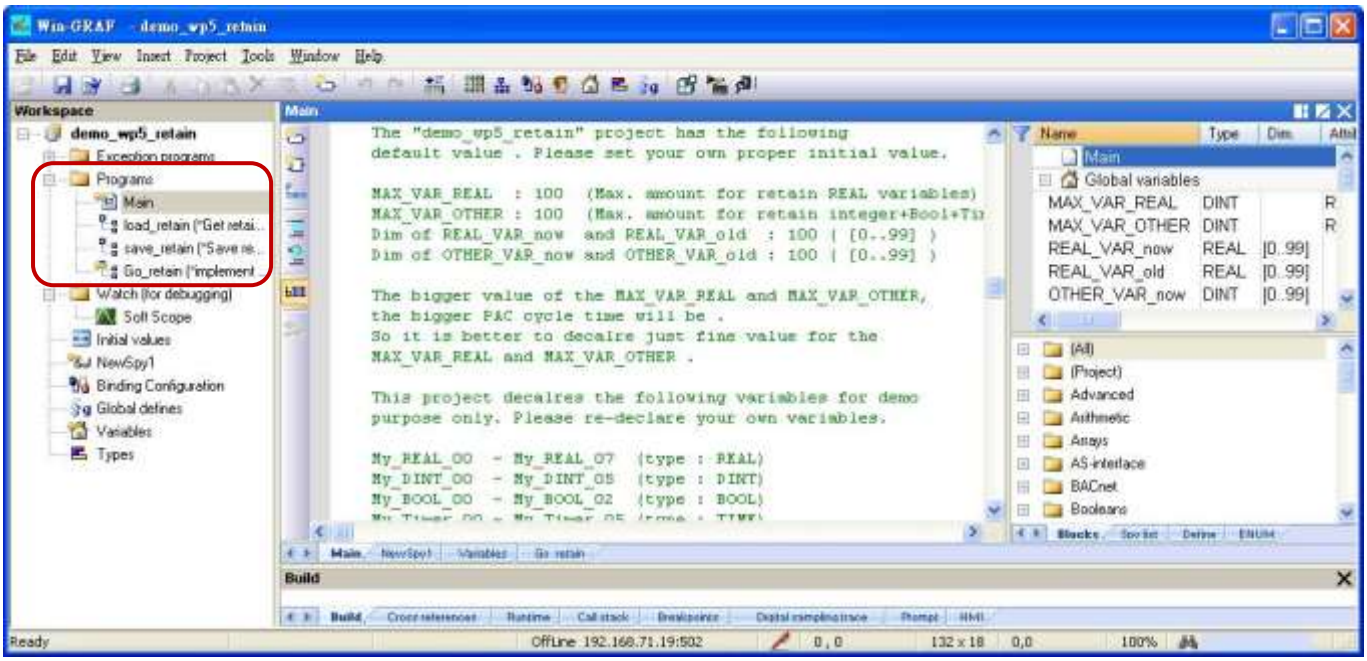
Q: 資料型態為 “BOOL”，只回傳 “TRUE”。

6.2 可保存變數 (僅適用 WP-5xx8)

由於 WP-5xx8 沒有資料保存記憶區 (Retain Memory)，所以未支援 Retain_VAR() 與 Retain_Ary() 函式，您可使用此章節所提供的範例程式 (demo_wp5_retain.zip) 將需保存的變數資料儲存在 PAC 內 \System_disk\Win-GRAF\ 目錄下的檔案中。

於出貨光碟中 (\Napdos\Win-GRAF\demo-project) 有提供此範例程式 (demo_wp5_retain.zip)，請參考 第 12 章 來回存此專案 (執行 File > Add Existing Project > From Zip) 並設定好 PAC 目前的 IP 位址。

此專案包含了 1 個 ST 主程式 (Main) 與 3 個 ST 副程式 (load_retain, save_retain 與 Go_retain)。



使用限制:

此專案並不適合處理數值會頻繁變動的保存變數 (例如: 每秒或每分鐘需變更數值)。這是因為變數資料是儲存在 \System_disk\ 目錄下，若保存數值常常變動，則檔案更新會消耗很多 CPU 時間進而造成 PAC 效能低弱。

以下為 "demo_wp5_retain" 專案中設定的初值，請依實際需求設定適當的值。

MAX_VAR_REAL: **100** (最多可使用 REAL 保留變數的數量)
MAX_VAR_OTHER: **100** (最多可使用 Integer, BOOL, TIMER 保留變數的全部數量)
"REAL_VAR_now" 與 "REAL_VAR_old" 的 Dim. 值: **100** ([0..99]，需和 "MAX_VAR_REAL" 值相同)
"OTHER_VAR_now" 與 "OTHER_VAR_old" 的 Dim. 值: **100** ([0..99]，需和 "MAX_VAR_OTHER" 值相同)

註: "MAX_VAR_REAL" 與 "MAX_VAR_OTHER" 的設定值越大，表示 PAC 的 Cycle Time 也越大。因此，請將此兩個數值設定為接近實際的使用數量即可。

以下為此專案中宣告的變數，請依實際需求來宣告變數。

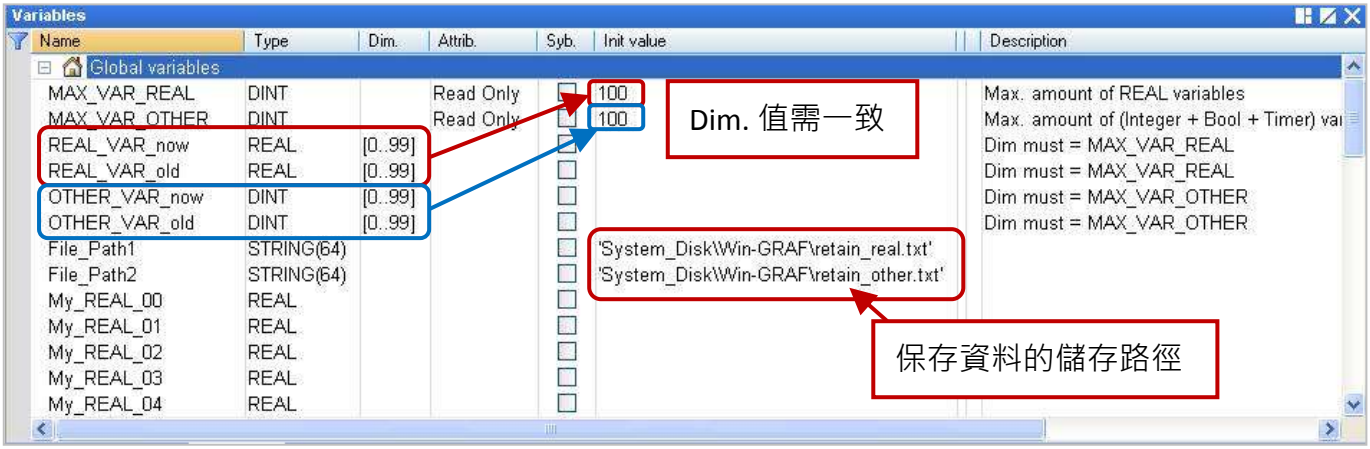
My_REAL_00 ~ My_REAL_07 (資料型態: REAL)

My_DINT_00 ~ My_DINT_05 (資料型態: DINT)

My_BOOL_00 ~ My_BOOL_02 (資料型態: BOOL)

My_Timer_00 ~ My_Timer_05 (資料型態: TIME)

您可在 "Variables" 視窗查看更多的變數。



"Go_retain" 副程式是用來執行檔案中的保存動作，請記得修改以下的 4 個區塊 (註解為 "Add your REAL variables for retain here" 與 "Add your integer, BOOL, Timer variables for retain here")，請依實際宣告的變數來修改:

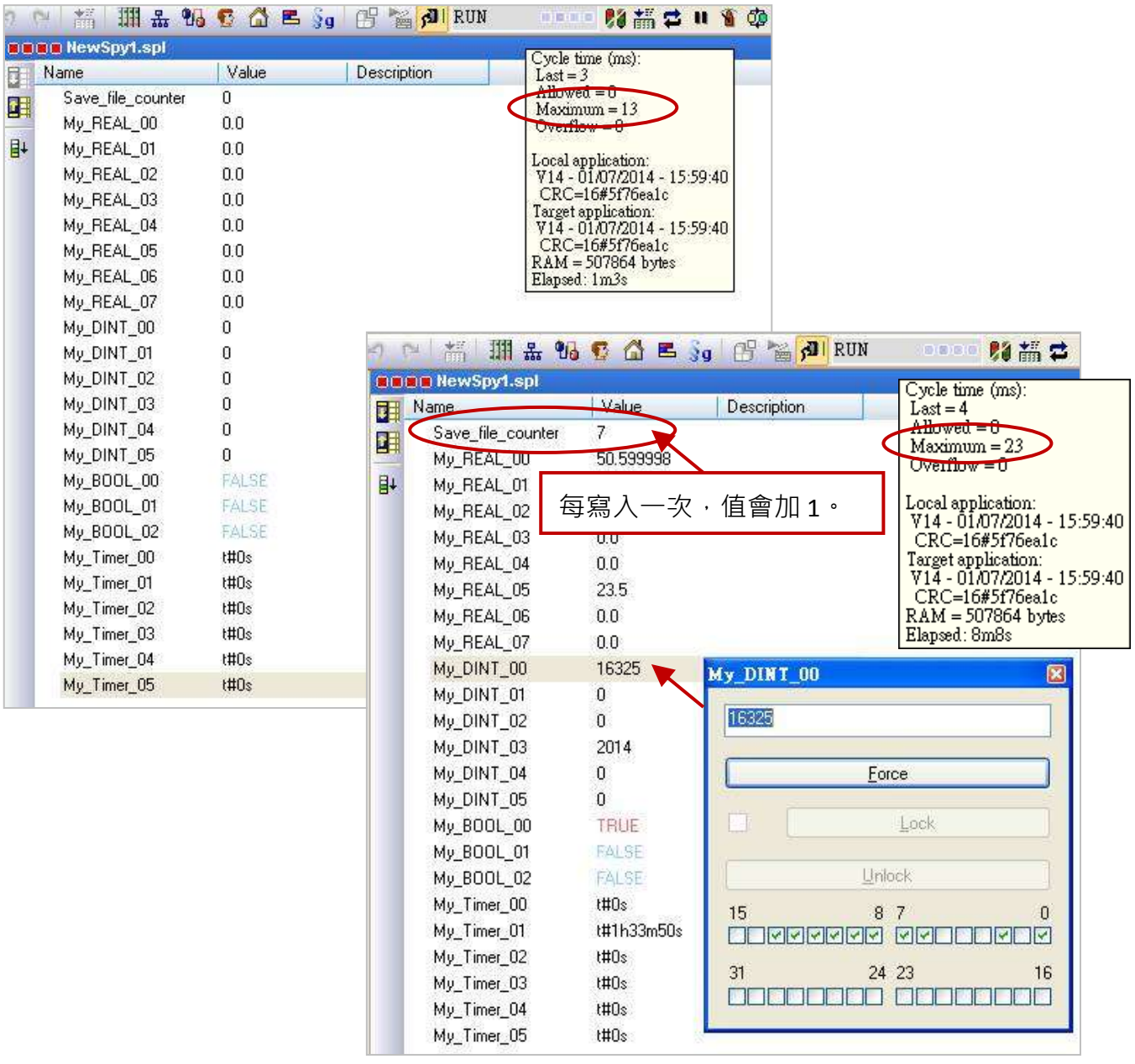
```
(* Add your REAL variables for retain here *)
(* ------------------------------------------------ *)
My_REAL_00 := REAL_VAR_now[0] ;
My_REAL_01 := REAL_VAR_now[1] ;
My_REAL_02 := REAL_VAR_now[2] ;
My_REAL_03 := REAL_VAR_now[3] ;
My_REAL_04 := REAL_VAR_now[4] ;
My_REAL_05 := REAL_VAR_now[5] ;
My_REAL_06 := REAL_VAR_now[6] ;
My_REAL_07 := REAL_VAR_now[7] ;
(* ------------------------------------------------ *)


(* Add your integer, BOOL, Timer variables for retain here *)
(* ..................................................... *)
My_DINT_00 := OTHER_VAR_now[0] ;
My_DINT_01 := OTHER_VAR_now[1] ;
My_DINT_02 := OTHER_VAR_now[2] ;
My_DINT_03 := OTHER_VAR_now[3] ;
My_DINT_04 := OTHER_VAR_now[4] ;
My_DINT_05 := OTHER_VAR_now[5] ;
```

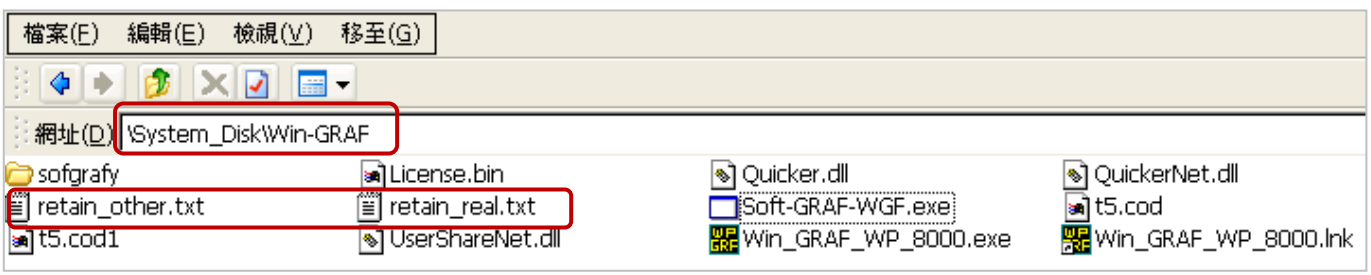
```
   My_BOOL_00 := Any_to_BOOL( OTHER_VAR_now[6] ) ;
   My_BOOL_01 := Any_to_BOOL( OTHER_VAR_now[7] ) ;
   My_BOOL_02 := Any_to_BOOL( OTHER_VAR_now[8] ) ;

   My_Timer_00 := Any_to_TIME( OTHER_VAR_now[9] ) ;
   My_Timer_01 := Any_to_TIME( OTHER_VAR_now[10] ) ;
   My_Timer_02 := Any_to_TIME( OTHER_VAR_now[11] ) ;
   My_Timer_03 := Any_to_TIME( OTHER_VAR_now[12] ) ;
   My_Timer_04 := Any_to_TIME( OTHER_VAR_now[13] ) ;
   My_Timer_05 := Any_to_TIME( OTHER_VAR_now[14] ) ;
   (* ..................................................... *)


(* Add your REAL variables for retain here *)
 (* ------------------------------------------------- *)
 REAL_VAR_now[0]  := My_REAL_00 ;
 REAL_VAR_now[1]  := My_REAL_01 ;
 REAL_VAR_now[2]  := My_REAL_02 ;
 REAL_VAR_now[3]  := My_REAL_03 ;
 REAL_VAR_now[4]  := My_REAL_04 ;
 REAL_VAR_now[5]  := My_REAL_05 ;
 REAL_VAR_now[6]  := My_REAL_06 ;
 REAL_VAR_now[7]  := My_REAL_07 ;
 (* ------------------------------------------------- *)


 (* Add your integer, BOOL, Timer variables for retain here *)
 (* ..................................................... *)
 OTHER_VAR_now[0]  := My_DINT_00 ;
 OTHER_VAR_now[1]  := My_DINT_01 ;
 OTHER_VAR_now[2]  := My_DINT_02 ;
 OTHER_VAR_now[3]  := My_DINT_03 ;
 OTHER_VAR_now[4]  := My_DINT_04 ;
 OTHER_VAR_now[5]  := My_DINT_05 ;

 OTHER_VAR_now[6]  := Any_to_DINT( My_BOOL_00 ) ;
 OTHER_VAR_now[7]  := Any_to_DINT( My_BOOL_01 ) ;
 OTHER_VAR_now[8]  := Any_to_DINT( My_BOOL_02 ) ;

 OTHER_VAR_now[9]  := Any_to_DINT( My_Timer_00 ) ;
 OTHER_VAR_now[10] := Any_to_DINT( My_Timer_01 ) ;
 OTHER_VAR_now[11] := Any_to_DINT( My_Timer_02 ) ;
 OTHER_VAR_now[12] := Any_to_DINT( My_Timer_03 ) ;
 OTHER_VAR_now[13] := Any_to_DINT( My_Timer_04 ) ;
 OTHER_VAR_now[14] := Any_to_DINT( My_Timer_05 ) ;
 (* ..................................................... *)
```

測試程式:

測試之前，請確認已設定好 PAC IP，再編譯/下載程式到 PAC 中 (可參考 2.3.4 節 與 2.3.5 節)。與 PAC 連線時，在觀測清單中 (Spy list，設定方式可參考 11.3 節) 一開始所有數值為 "0" (或 "FALSE")，請任意的輸入幾個數值，當數值有變更時會在 PAC 內建立文字檔 (\System_disk\Win-GRAF\retain_real.txt 與 retain_other.txt) 並將資料寫入到該檔案中。

註: "Save_file_counter" 會顯示寫入檔案的次數，若此數值變化很快 (例如: 每秒/分鐘寫入好幾次)，則不適合此應用 (因為頻繁地在 \System_disk 下寫入檔案，會降低 PAC 效能)。





6.3 備份資料到 EEPROM

Win-GRAF 系列 PAC 內建有一個 EEPROM 記憶體可供使用者用來讀取與寫入資料，此資料不會因 PAC 關機而消失。相較於 SRAM 的讀寫方式，EEPROM 有以下優缺點:

優點: 提供 "可保存變數" (Retain Variable，參考 6.1 節) 之外，另一個可保存重要資料的方式。

缺點: 1. EEPROM 的讀寫動作很耗 CPU 時間 (約 5 ~ 50 ms)，而使用 "可保存變數" 方式，CPU 時間遠小於 1 ms。因此，**請勿**太頻繁的使用 "EEP_Read" 與 "EEP_Write" 函式，會增加許多 PAC Cycle 時間。

2. EEPROM 有寫入限制 (視 PAC 而定)，並不適合對同一筆資料進行多次寫入，因此，**請勿**在每個 PAC Cycle 內呼叫 "EEP_Write" 函式來進行寫入動作。

ST 語法: (下列分為 安全 與 危險 寫法)

```
(* 宣告 "FIRST_CYCLE" 為 “BOOL” 變數且初始值為 “TRUE”。
   宣告 "tmp_bool" 為 “BOOL” 變數。
   宣告 "New_Val" 與 "Old_Val" 為 “DINT” 變數。 *)
(* 於第一個 Cycle 讀取 EEPROM 一次 *)
if FIRST_CYCLE then
  FIRST_CYCLE := FALSE ; (* 表示不再是第一個 Cycle *)
  tmp_bool := EEP_Read ( 1 , New_Val) ;
end_if ;
(* 安全寫法: 只有數值改變時才寫入到 EEPROM *)
if New_Val <> Old_Val then
  Old_Val := New_Val ;
  tmp_bool := EEP_Write ( 1 , New_Val ) ;
end_if ;
```

```
(* 危險寫法: EEPROM 可能很快被損毀 *)
(* 宣告 "FIRST_CYCLE" 為 “BOOL” 變數且初始值為 “TRUE”。
   宣告 "tmp_bool" 為 “BOOL” 變數。
   宣告 "New_Val" 與 "Old_Val" 為 “DINT” 變數。 *)
(* 於第一個 Cycle 讀取 EEPROM 一次 *)
if FIRST_CYCLE then
  FIRST_CYCLE := FALSE ; (* 表示不再是第一個 Cycle *)
  tmp_bool := EEP_Read ( 1 , New_Val) ;
end_if ;
(* 危險寫法: 於每個 Cycle 內將 "New_Val" 值寫入到 EEPROM 一次 *)
tmp_bool := EEP_Write ( 1 , New_Val ) ;
```

6.3.1 EEP_READ (讀取 EEPROM)

> **使用小技巧:**
> 按 “F1” 鍵，可查看詳細的設定說明。

Addr: (資料型態: “DINT”)

位址可設定為 1 ~ 1200。若 "@Name" 參數的資料型態為 64-bit 資料 (例如: LINT、ULINT、LREAL 或 LWORD)，則 "Addr" 只能設定為 1001 ~ 1200。

@Name :

指定一個變數名稱，用來儲存讀到的 EEPROM 的資料。

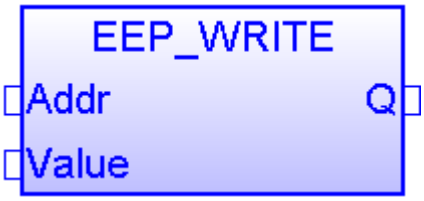
(請勿使用字串變數，資料型態可為 BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, LINT, ULINT, LWORD, LREAL。)

Q:

資料型態為 “BOOL”，“TRUE”: 表示 OK ； “FALSE”: 表示錯誤。

若 "@Name" 參數的資料型態為 "REAL" 或 "LREAL"，當讀取的資料不是實數或有其它錯誤發生，"Q" 會回傳 "FALSE"。若此資料非實數資料，"REAL" 或 "LREAL" 變數將會讀到 "0.0"。

6.3.2 EEP_WRITE (寫入 EEPROM)



Addr: (資料型態: “DINT”)

位址可設定為 1 ~ 1200。若 "Value" 參數的資料型態為 64-bit 資料 (例如: LINT、ULINT、LREAL 或 LWORD)，則 "Addr" 只能設定為 1001 ~ 1200。

Value :

將資料寫入 EEPROM。

(請勿使用字串變數，資料型態可為 BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, LINT, ULINT, LWORD, LREAL。)

Q:

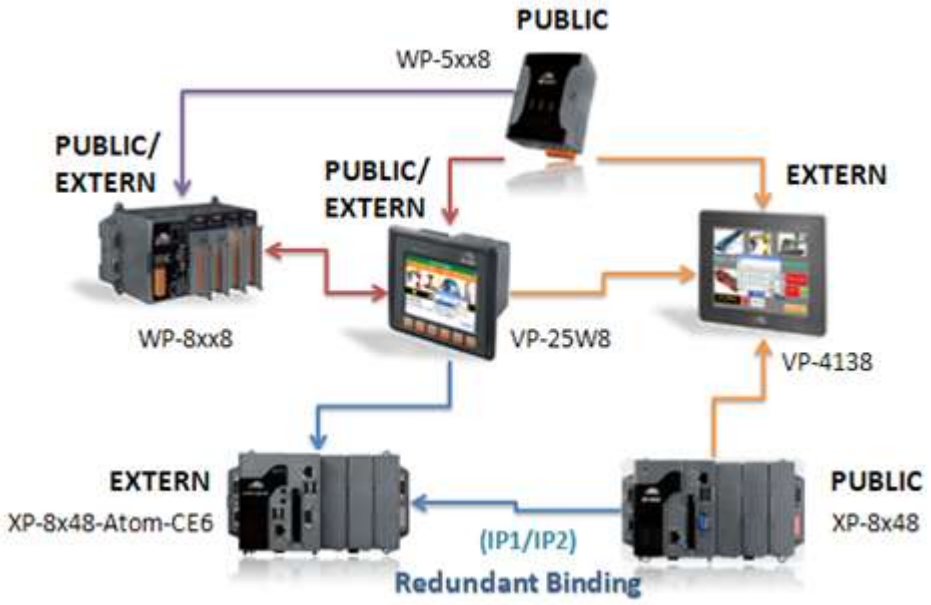
資料型態為 “BOOL”，“TRUE”: 表示 OK ； “FALSE”: 表示錯誤。

第 7 章　在 PAC 間互傳資料 (Data Binding)

"Binding“ 功能可讓多台 PAC 間互相傳遞彼此的資料，Win-GRAF 提供了兩種設定 Binding 方式:

- **PUBLIC:** 是指公開 PAC 自己的資料，或給同一台 PAC 內的 VB .net, C#, C 或 Soft-GRAF HMI 來使用。
- **EXTERN:** 是指從別台 PAC 取資料回來。

應用示意圖:



注意:

每台 Win-GRAF PAC 最多可使用的 “Binding” 功能 (EXTRN):

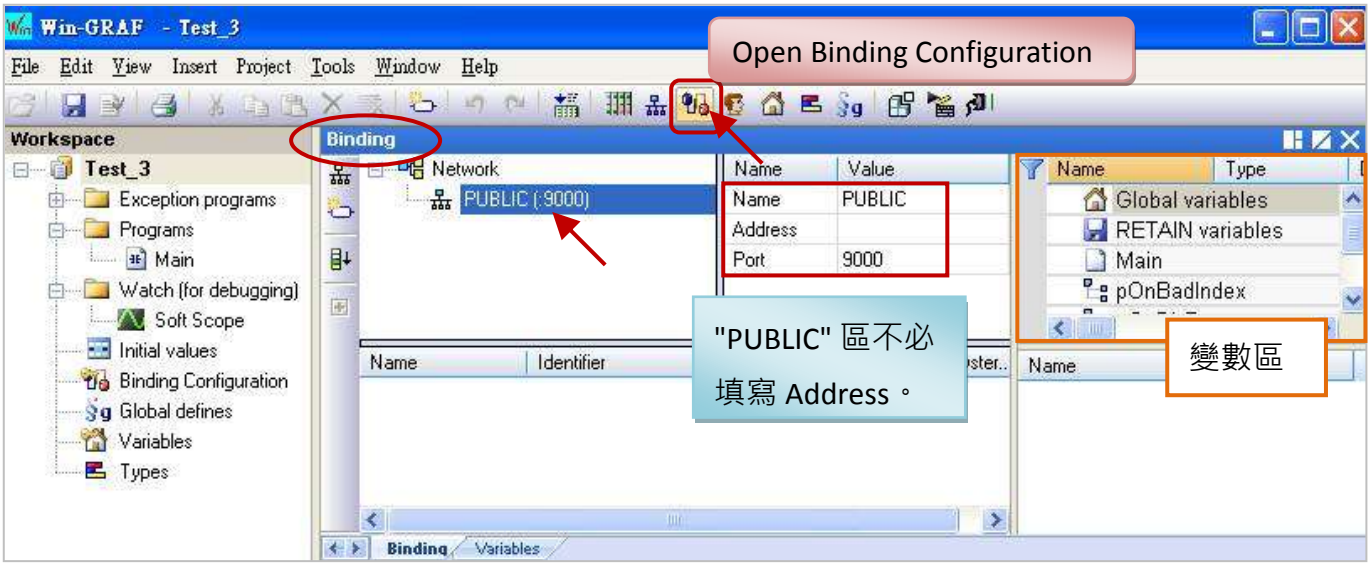
XPAC:　Max. 32
WinPAC:　Max. 16
ViewPAC:　Max. 16

(詳細型號，請見 P1-1)

"PUBLIC" 設定步驟如下:

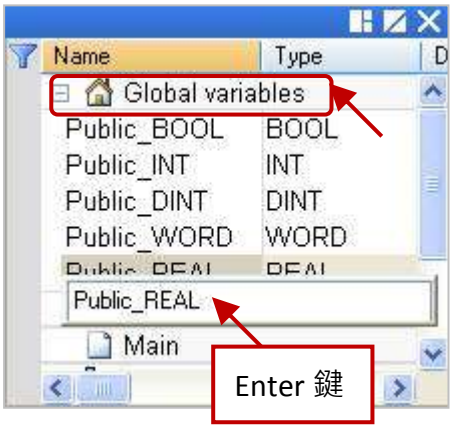
當 PAC 有指定 "PUBLIC" 區表示公開自己的資料。

1. 滑鼠點選工具列上的 “Open Binding Configuration” 按鈕來開啟 “Binding” 視窗。
2. 點選 "PUBLIC (:9000)" 來設定要公開的資料，您可修改 "Name" 欄位的名稱，"Address" 欄位無需填寫，"Port" 欄位固定為 "9000"，請勿更動。

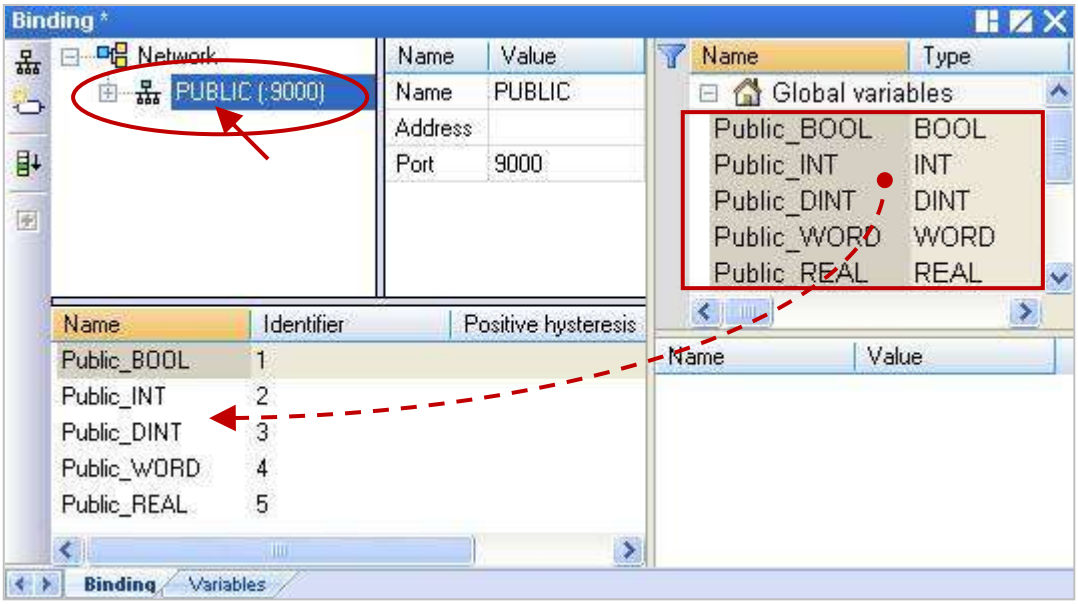


3. 在設定公開資料之前，您必須在變數區先建立好要公開的變數。滑鼠點選 “Global variables” 再按 “Ins” 鍵來新增變數項目，下表為此範例所使用的變數，您可依實際需求來設定，設定完成後，畫面如下。

變數名稱	資料型態
Public_BOOL	BOOL
Public_INT	INT
Public_DINT	DINT
Public_WORD	WORD
Public_REAL	REAL



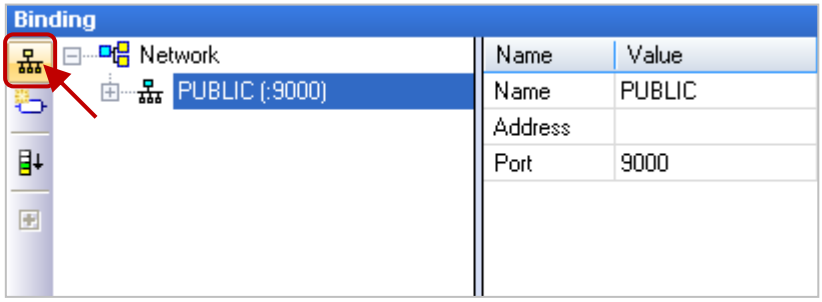
4. 滑鼠點選在 "PUBLIC (:9000)"，再選取變數區欲公開的變數資料，並拖曳到 "Name" 區域。"Identifier" 欄位會自動產生編號 (若其它台 PAC 想取用該資料，需設定一樣的 ID 編號)。
注意: "PUBLIC" 最多可使用 8192 個變數，"Identifier" 編號只能是 "1 ~ 8192"。



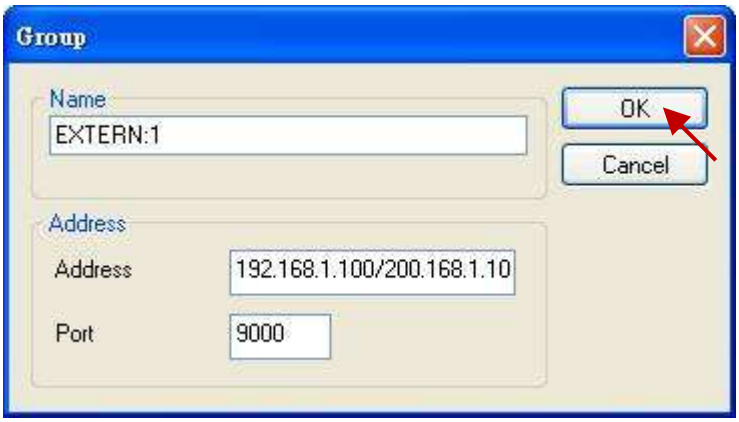
"EXTERN" 設定步驟如下:

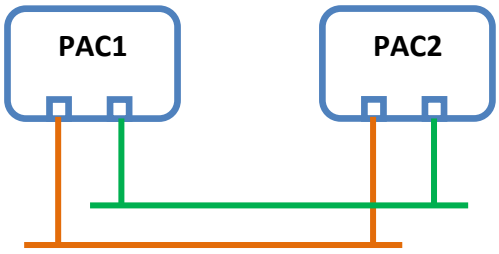
當 PAC 有指定 "EXTERN" 區表示要從別台 PAC 取回它的資料。

5. 點選左邊的 "Insert Master/Port" 按鈕，將會出現 "Group" 視窗 (見下一頁)。



請設定好以下欄位，並按 "OK“ 按鈕。





Name: 可修改為所需的名稱。

Address: 輸入要取得資料的那台 PAC 的 IP 位址 (例如: "192.168.1.100")，也可輸入兩個 IP 位址 (例如: "192.168.1.100/200.168.1.100"; 對方 PAC 需有使用 2 個 Ethernet Port)，如此當其中之一發生問題，會嘗試去連第 2 個 IP 位址。

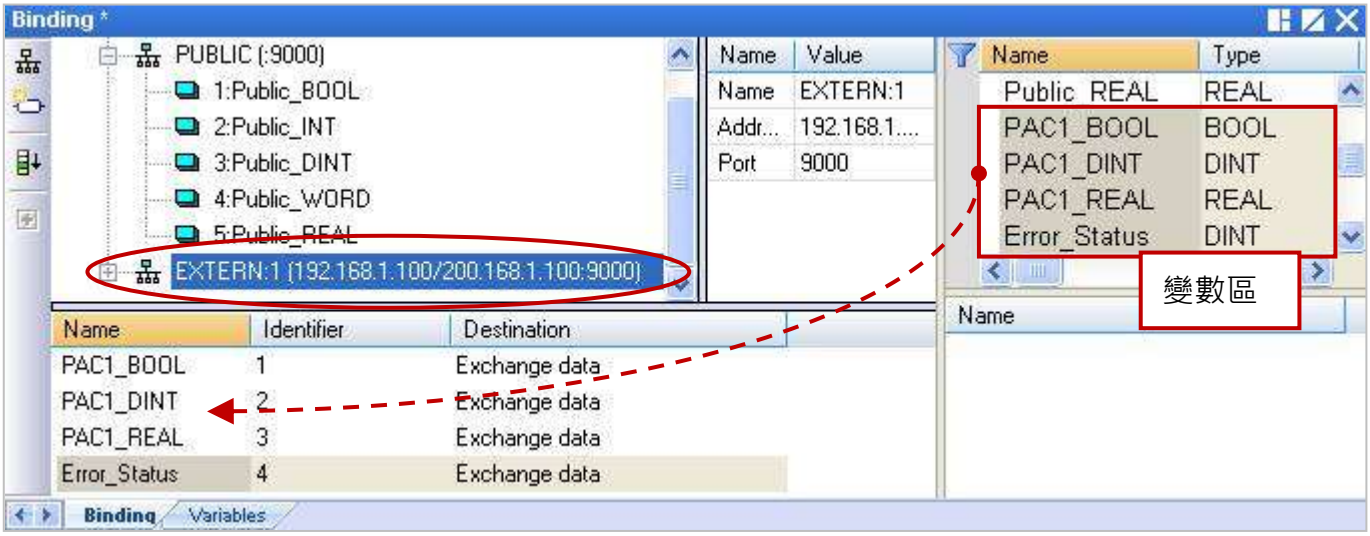
Port: 固定為 "9000"，請勿更動。

6. 您必須在變數區先設定好要取得的資料型態。(可參考步驟 3 - 滑鼠點選 “Global variables” 再按 “Ins” 鍵來新增變數項目)，右邊列表是此範例想取用的資料型態，您可依實際需求來設定，設定完成後，畫面如下。

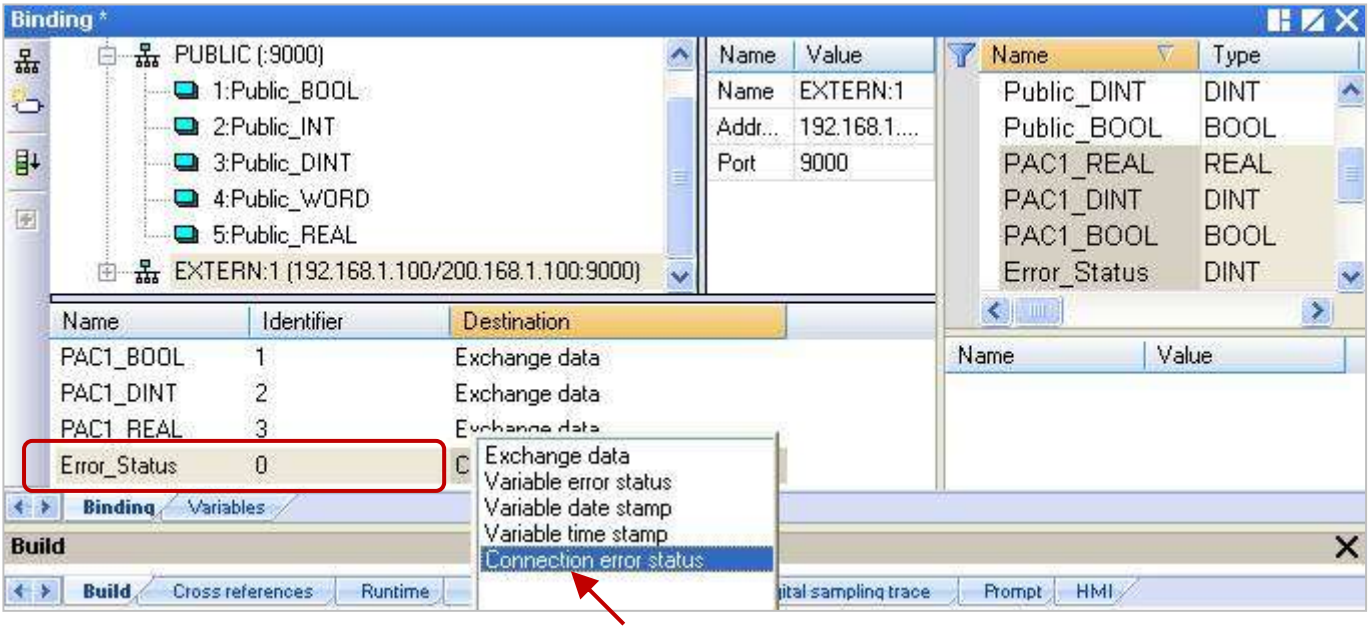
變數名稱	資料型態
PAC1_BOOL	BOOL
PAC1_DINT	DINT
PAC1_REAL	REAL
Error_Status	DINT

7. 請將所需的變數拖曳到 "EXTERN:1" 的 "Name" 區域。

   註: "Identifier" 欄位會自動產生編號，請修改為和 要取得資料的那台 PAC 所開放的 ID 一致。



8. 如圖，“Error_Status” 變數是用來判斷該 PAC 的通訊狀況，請將此 ID 設定為 "0" 再雙擊 "Destination" 欄位將其設定為 "Connection error status"。



註:

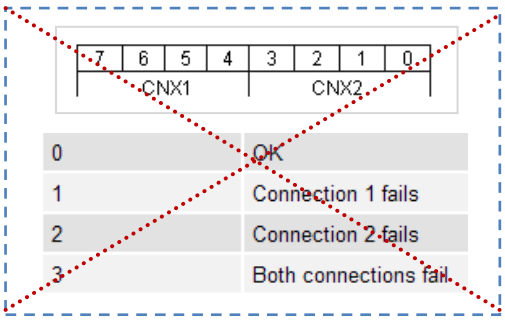
a. 若 “EXTERN” 設定了 2 個 IP 位址 (步驟 5)，則 “Error_Status” 會回傳 2 個通訊狀態。如下表，以 8 個 bit 來表示通訊狀態，bit 0 ~ 3 表示第 1 個 IP 的通訊狀態 (bit 皆為 1 時，值為 15)，bit 4 ~ 7 表示第 2 個 IP 的通訊狀態 (bit 皆為 1 時，值為 240)，只要不等於 0，即表示有通訊異常。

IP2 的通訊狀態				IP1 的通訊狀態				狀態說明
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
0				0				通訊 OK
0				≠0 (1 ~ 15)				IP1 通訊異常
≠ 0 (16 ~ 240)				0				IP2 通訊異常
≠ 0				≠ 0				IP1，IP2 皆通訊異常

b. “Error_Status” 的回傳值是一個整數值，以下提供了一個除法的判斷方式，將此數值除以 16，**商數**代表第 **2** 個 IP 的通訊狀態，**餘數**代表第 **1** 個 IP 的通訊狀態，不等於 0，即表示有通訊異常。例如，若 “Error_Status” = 16，除以 16 的結果，商數 = 1 (≠0，IP2 通訊異常) 且 餘數 = 0 (IP1 通訊 OK); 若 “Error_Status” = 3，除以 16 的結果，商數 = 0 (IP2 通訊 OK) 且 餘數 = 3 (≠0，IP1 通訊異常);

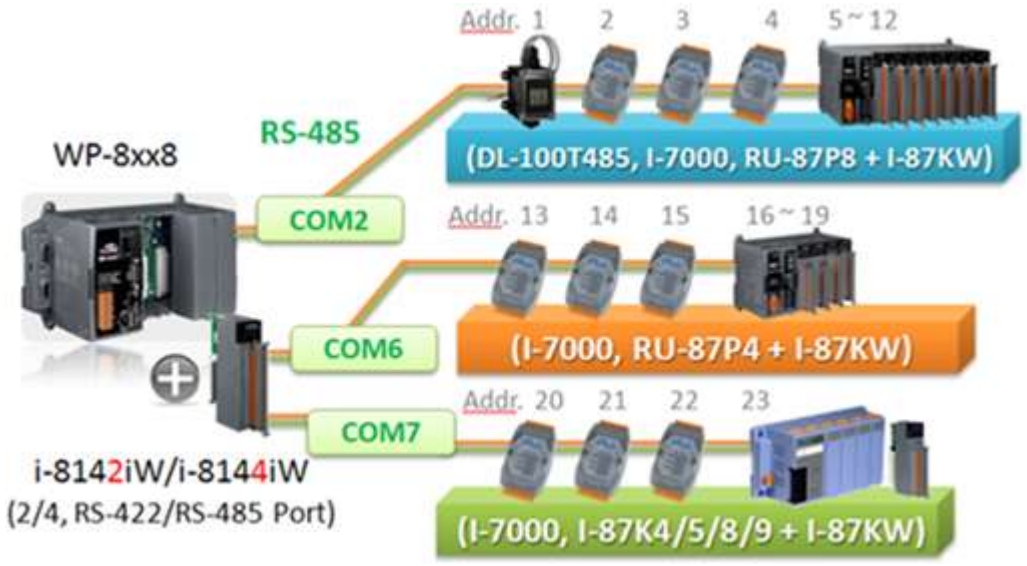
注意:

若按 "F1“ 鍵開啟 "HTML Help" 說明，主題 - "Networked applications - Dual binding on redundant ETHERNET" 中 "Connection status" 與 "Variable status" 的說明是錯誤的，不符合 ICP DAS Win-GRAF PAC 的使用方式，請忽略該說明。



第 8 章　連接 DCON I/O 模組

Win-GRAF PAC 可透過 COM Port (RS-485) 來連接 ICP DAS 的 "I-7000" 與 "I-87KW" 遠端 DCON I/O 模組，每台 PAC 最多可啟用 16 個 DCON Port，而每個 Port 最多可連接 50 個遠端 DCON 模組 (建議不要超過 32 個) 。若選用 "I-87KW" 系列 I/O 模組，必須搭配使用 RS-485 I/O 擴充單元 (例如: I-87K4/5/8/9 或 RU-87P4/8)。您可在泓格科技的網站上查看詳細的產品資訊:
www.icpdas.com/root/product/solutions/remote_io/remote_io_products_tc.html



在開始使用 "I-7000" 或 "I-87KW" 遠端 DCON I/O 模組前，需先以 "DCON Utility" 軟體設定每一個模組的 Protocol (請選 DCON 模式)、Address (1 ~ 255)、Baudrate (需與 Win-GRAF PAC 一樣，建議設為 9600)、Checksum (一般為 Disable) 、Data format 與其它 Input/Output 設定 (依需求來設定)。

注意:

A. 若 I-7000 及 I-87KW 是 AI 模組，則資料格式 (Data format) 必需設定為 **"2's Complement"**。
例如: I-7005, I-7013, I-7014D, I-7015, I-7016, I-7017R, I-7018Z, I-7019R, I-7033; I-87005W, I-87013W, I-87015W, I-87015PW, I-87016W, I-87017W, I-87017RCW, I-87017ZW, I-87017DW, I-87018W, I-87018RW, I-87018ZW, I-87019RW, I-87019ZW, ...等類比輸入模組。

B. 若 I-7000 及 I-87KW 是 AO 模組，則資料格式 (Data format) 必需設定為 **"Engineering"**。
例如: I-7021, I-7022, I-7024, I-7024R; I-87024W, I-87024UW, I-87024CW, I-87028UW, I-87028CW, I-87028VW, I-87028VW-20V 等類比輸出模組。

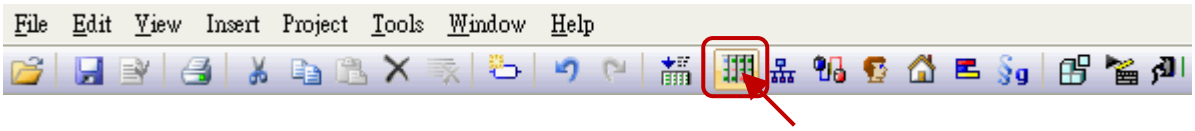
"DCON Utility" 是一個方便好用的軟體工具，可以協助網路搜尋、設定與測試 I/O 模組。
請至以下網址取得軟體程式與使用手冊 :
www.icpdas.com/products/dcon/introduction.htm

接下來將說明，Win-GRAF Workbench 中的設定方式。

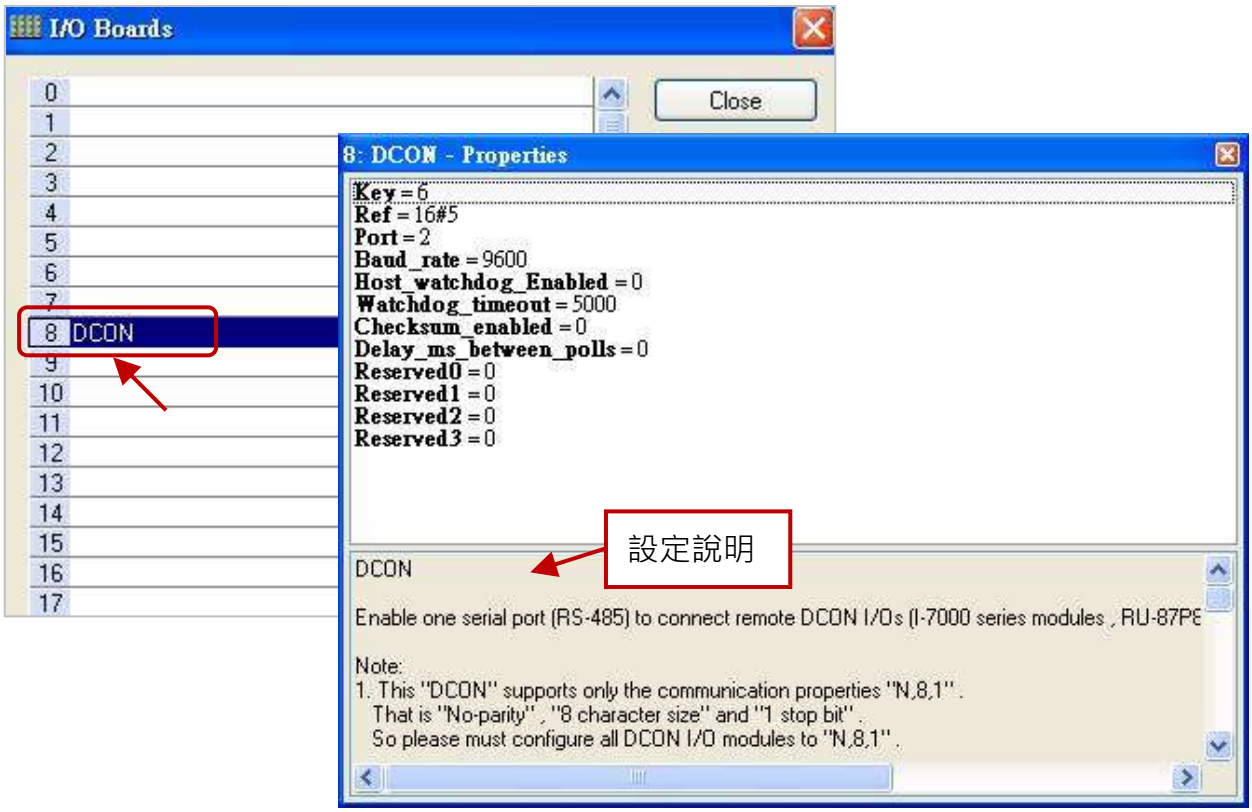
8.1 設定 "DCON" I/O 卡

“DCON” 可用來啟用一個 RS-485 Port 來連接遠端的 DCON I/O 模組 (例如: I-7000 系列模組、RU-87P8 I/O 擴充單元 + I-87KW I/O 模組 或 I-87K8 I/O 擴充單元 + I-87KW I/O 模組)。如需啟用多個 DCON Port， 請設定多個 “DCON” I/O 卡。(一台 PAC 最多可啟用 16 個 "DCON")

1. 點選 Win-GRAF 工具列的 "Open I/Os" 按鈕來開啟 “I/O Boards” 視窗。



2. 於 Slot8 加入 "DCON" I/O 卡 (參考第四章)，再以滑鼠雙擊該項目來開啟 “Properties” 視窗。
   注意: Slot0 ~ 7 是保留給 PAC I/O 模組，Slot8 (含) 以上供給其它用途使用。



參數說明:

註: "DCON" 僅支援通訊屬性為 "N,8,1"，表示 "無同位元"、"8 個資料位元" 與 " 1 個停止位元"，因此請將所有的 DCON I/O 模組設置為 "N,8,1"。

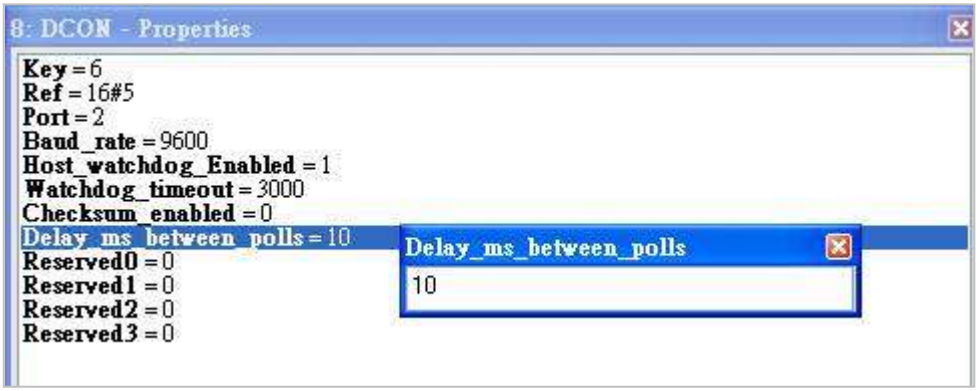
Port:	COM Port 編號 (可設定為 1 ~ 37，視 PAC 而定)。
Baud_rate:	通訊速率，可設定為 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 (bps)，若設定為其它值，將會採用預設值 "9600"。
Host_watchdog_Enabled:	1: 表示啟用 Host-watchdog，0: 表示不啟用 設定為非 0 的值，將會採用 "1"。

Watchdog_timeout: 單位: ms，可設定為 “3000 ~ 25500”，設定為大於 “25500” 將會採用 25500 ms (即 25.5 秒); 設定為小於 “3000” 將會採用 3000 ms (即 3 秒)。若 "Host_watchdog_Enabled" 設定為 "0" 會忽略此設定。

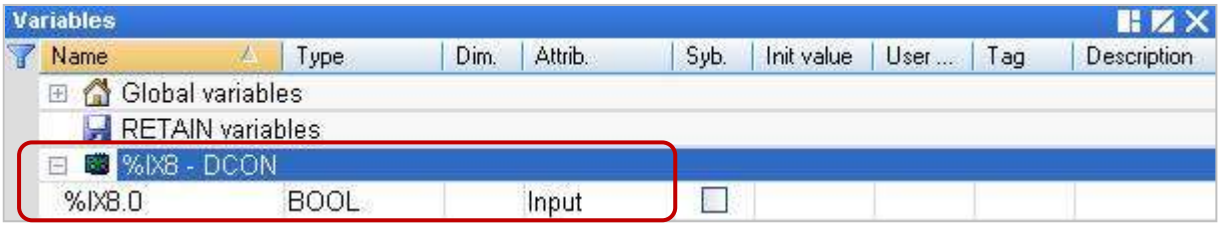
Checksum_enabled: 1: 表示啟用，0: 表示不啟用。
若設定為非 0 的值，將會採用 "1"。

Delay_ms_between_polls: 單位: ms，預設值為 0 ms，有效範圍為 "0 ~ 1000"。設定為小於 “0” 將會採用 0 ms ; 設定為大於 “1000” 將會採用 1000 ms。
若沒有連接無線模組，請設定為較小的值 (例如: 0 ~ 10)。
若連接無線模組 (例如: ICP DAS 的 ZT-2570、ZT-2571 Ethernet/RS-485/RS-232 至 ZigBee 轉換器 或 ZB-2000 系列 DIO/AIO 模組)，請設定為較大的值 (例如: 30 ~ 100 或 其它值)，而設定較大的值，輪詢 (Polling) 效率會較慢。

3. 滑鼠雙擊欲設定的項目，並輸入設定值。

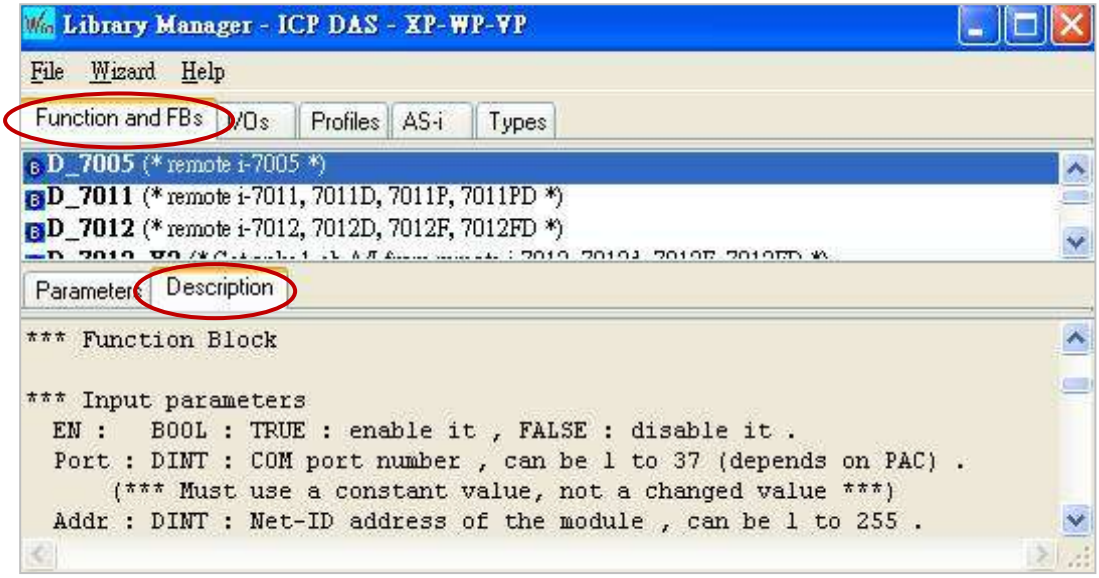


4. 在 “I/O Boards” 視窗內連上 “DCON” 後，會自動在 “Variables” 視窗中新增 1 個 "BOOL” 輸入變數，當 Win-GRAF 有連上 PAC 時，會顯示出 COM Port 的連接狀態。
(TRUE: 表示 OK ； FALSE: 表示錯誤。)

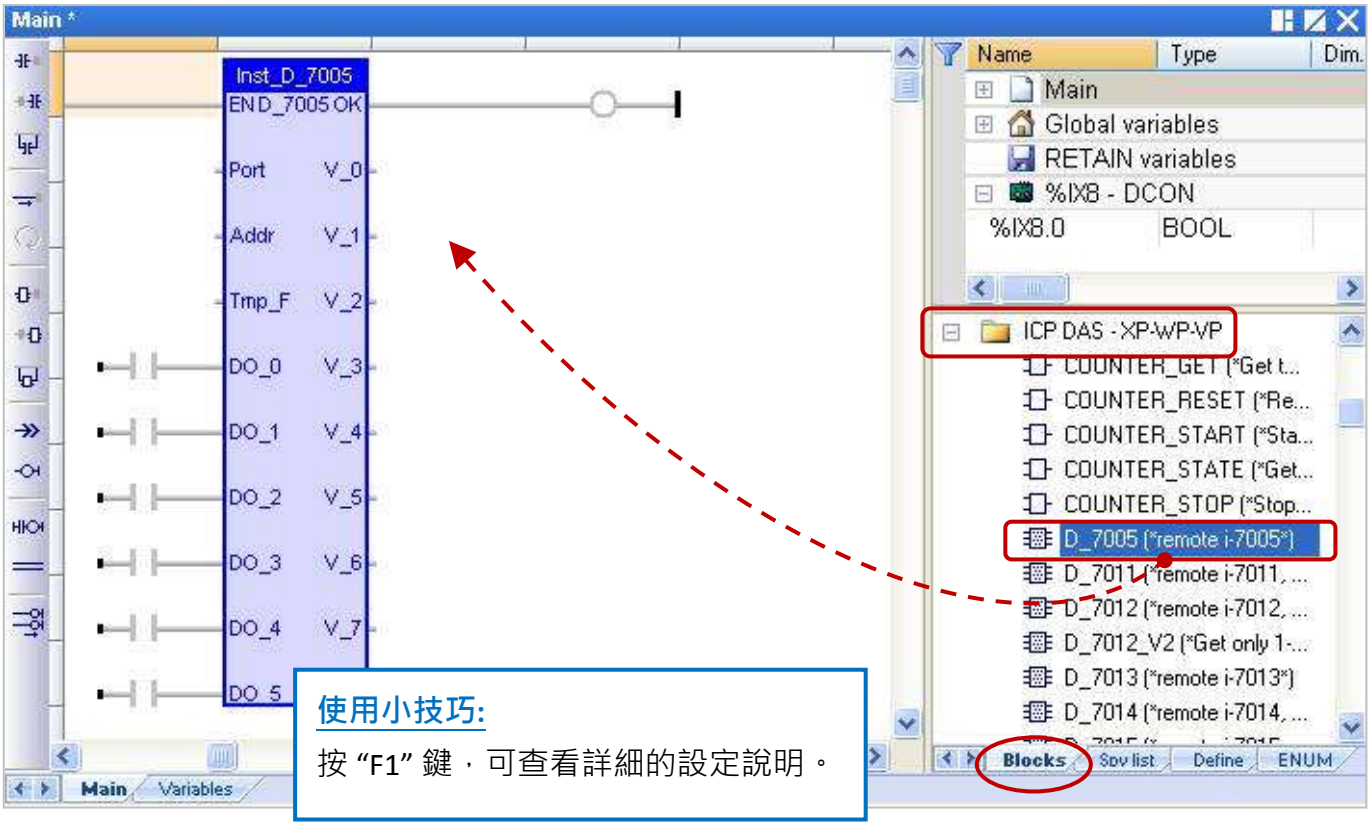


8.2 使用 I/O 功能方塊

Win-GRAF 支援許多 ICP DAS 的 DCON 遠端 I/O 模組，您可開啟 “程式庫管理員” (參考 1.2.3 節) 或在 I/O 功能方塊上按 “F1” 鍵來查看這些 I/O 功能方塊的設定說明，本章節將介紹 “D_7065”, “D_7018Z”, “D_7083”, “D_87084_freq”, “D_87084_cnt4”, “D_87084_cnt8”, “DL_100T485” I/O 功能方塊。



在 LD 程式 - 功能方塊區中，可在 “Blocks” 面板中展開 "ICP DAS – XP-WP-VP" 資料夾，裡面列有許多函式 與 功能方塊，您可選擇所需的項目，並將它拖曳到程式編輯區來使用。



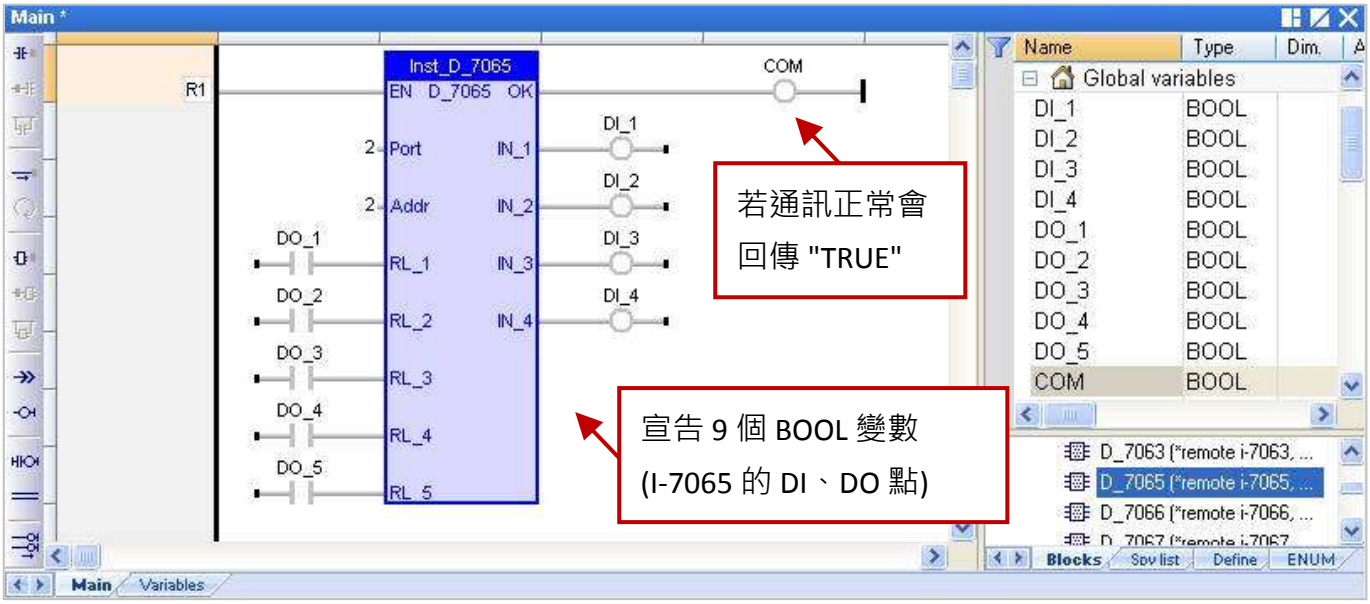
8.2.1 “D_7065” 功能方塊

“D_7065” 可用來連接一個遠端的 I-7065, I-7065D 繼電器 (Relay) 輸出模組 與 I-7065A, I-7065AD, I-7065B, I-7065BD (固態繼電器輸出模組)。

註:

1. 所有連接的 DCON I/O 模組需使用 "DCON Utility" 軟體 (見 P8-1) 設定過一次。
2. 請在 “I/O boards” 視窗加入 “DCON” (見 8.1 節)，並設定適當的參數 (例如: Port、Baud_rate...)。
3. 只有在通訊狀態為 "TRUE" 的情況下 (若 "OK" 會回傳 "TRUE")，DI 通道的回傳值才具意義。
4. 您可參考 第 12 章，點選功能表 "File" > "Add Existing Project" > "From Zip"，來回存出貨光碟中的範例程式 (CD-ROM: \Napdos\Win-GRAF\demo-project\demo_d_7065.zip) 並查看詳細的程式內容。

假設: 使用 PAC 的 COM2 來連接 I-7065 (位址 = 2)，使用 4 個數位輸入通道 與 5 Relay 輸出通道。



輸入參數:

EN: 資料型態: BOOL，設定為 “TRUE": 啟用；設定為 “FALSE“: 不啟用。

Port: 資料型態: DINT，COM Port 編號 (可設定為 1 ~ 37，視 PAC 而定)。
(*** 需設定為常數，即不可變動的值 ***)

Addr: 資料型態: DINT，模組的 Net-ID 位址 (可設定為 1 ~ 255)。
(*** 需設定為常數，即不可變動的值 ***)

RL_1 ~ RL_5: 資料型態: BOOL，5 通道 DO 值。

輸出參數:

OK: 資料型態: BOOL，"TRUE": 表示通訊正常；"FALSE": 表示通訊失敗。

IN_1 ~ IN_4: 資料型態: BOOL，4 通道 DI 值。

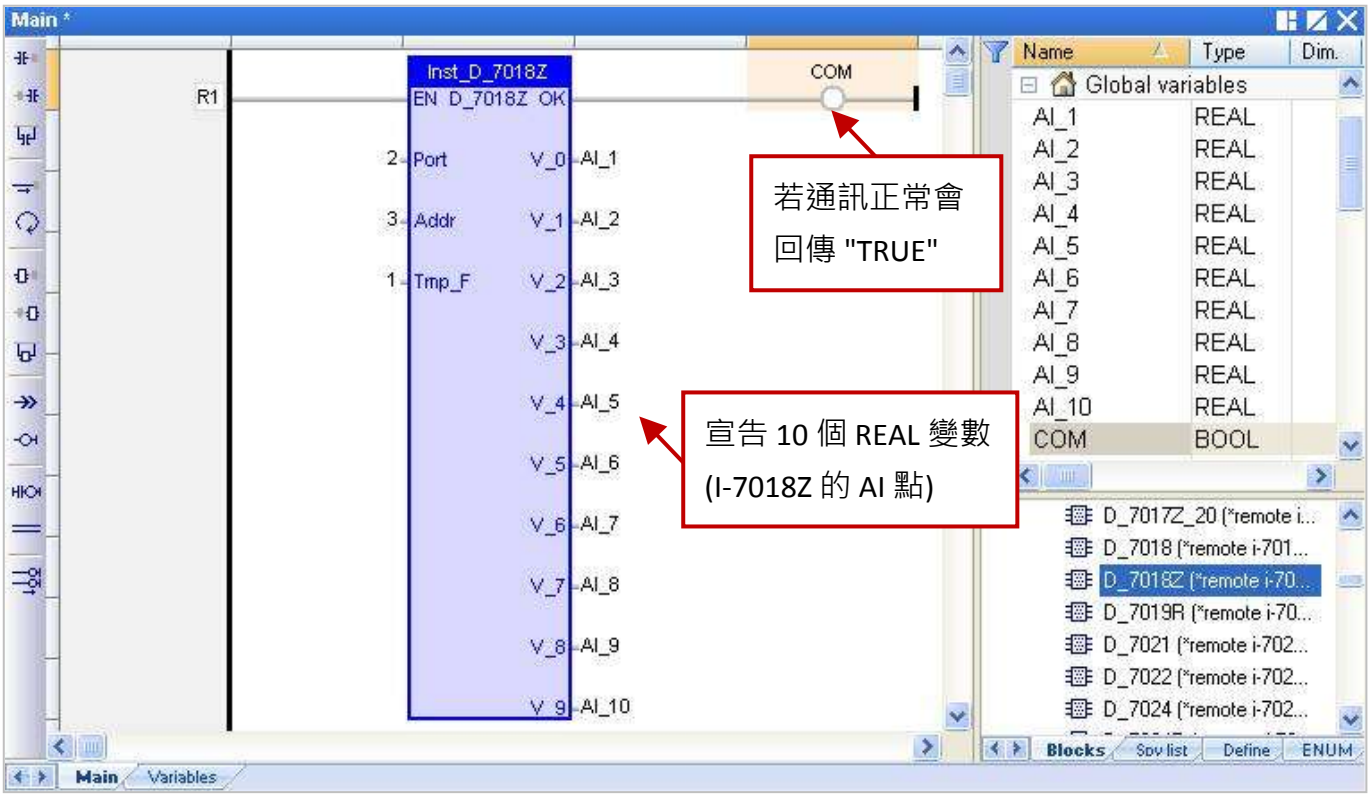
8.2.2 “D_7018Z” 功能方塊

“D_7018Z” 可用來連接一個遠端的 I-7018Z 模組，此模組是 10 通道熱電耦類比輸入模組，可用來量測電壓、電流或溫度，通道可個別設定並具有斷線偵測、過電壓保護功能。

註:

1. 請先使用 "DCON Utility" 軟體 (見 P8-1) 設定好該模組的適當參數 (例如: Address、Baudrate...)，AI 模組的資料格式需設定為 **"2's Complement"**，否則 Win-GRAF PAC 無法正確地讀取該值。
2. 請在 “I/O boards” 視窗加入 “DCON” (見 8.1 節)，並設定適當的參數 (例如: Port、Baud_rate...)。
3. 只有在通訊狀態為 "TRUE" 的情況下 (若 "OK" 會回傳 "TRUE")，AI 通道的回傳值才具意義。
4. 您可參考 第 12 章，點選功能表 "File" > "Add Existing Project" > "From Zip"，來回存出貨光碟中的範例程式 (CD-ROM: \Napdos\Win-GRAF\demo-project\demo_d_7018z.zip) 並查看詳細的程式內容。

假設: 使用 PAC 的 COM2 來連接 I-7018Z (位址 = 3)，且用來量測攝氏溫度。



輸入參數:

EN: 資料型態: BOOL，設定為 “TRUE": 啟用；設定為 “FALSE“: 不啟用。

Port: 資料型態: DINT，COM Port 編號 (可設定為 1 ~ 37，視 PAC 而定)。
(*** 需設定為常數，即不可變動的值 ***)

Addr: 資料型態: DINT，模組的 Net-ID 位址 (可設定為 1 ~ 255)。
(*** 需設定為常數，即不可變動的值 ***)

Tmp_F: 資料型態: DINT，溫度格式可設定為 1 或 2。

1 : 表示為攝氏溫度

2 : 表示為華氏溫度

若設定為其它值，則會取用 "1: 攝氏溫度"。

輸出參數:

OK: 資料型態: BOOL，"TRUE": 表示通訊正常； "FALSE": 表示通訊失敗。

V_0 ~ V_9: 資料型態: REAL，10 通道 AI 值。

若在 "DCON Utility" 選用了通道為電壓類型，表示該回傳值的單位為 "V"。

例如，回傳值為 0.85421 表示 0.85421 V 或 854.21 mV。

若在 "DCON Utility" 選用了通道為電流類型，表示該回傳值的單位為 "mA"。

例如，回傳值為 1.5567 表示 1.5567 mA。

若在 "DCON Utility" 選用了通道為溫度類型，表示該回傳值的單位為 "度"。

例如，回傳值為 25.75 表示 25.75 度。

斷線偵測:

若溫度值大於 "9000.0" 表示，

1. 溫度感測器可能斷線。
2. 溫度感測器可能損毀。
3. DCON 模組的設定 與 溫度感測器不符。
4. 感測器量到錯誤的電阻值。

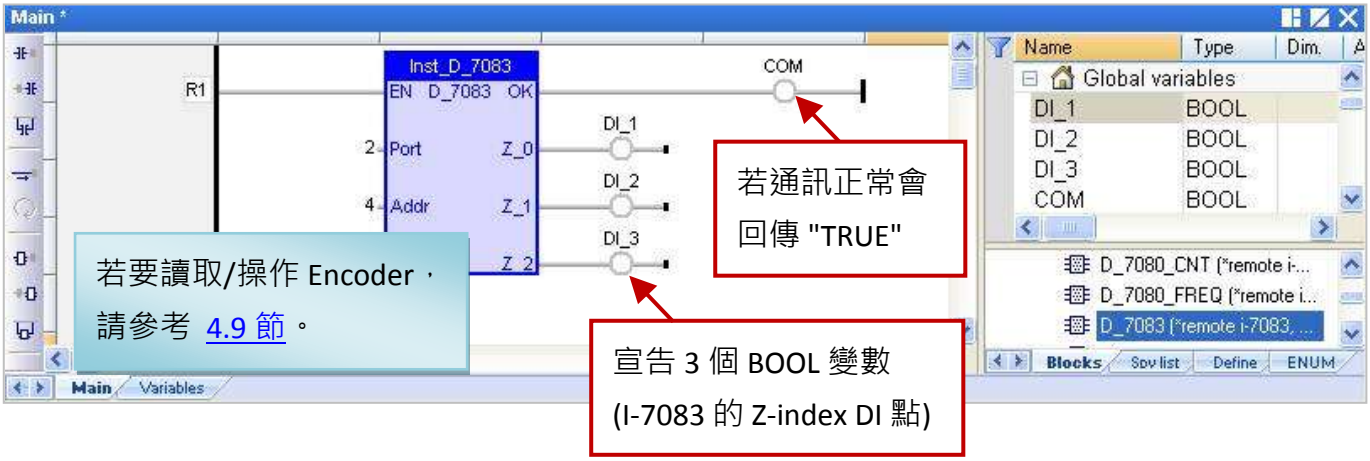
8.2.3 “D_7083” 功能方塊

“D_7083” 可用來連接一個遠端的 I-7083, I-7083D, I-7083B, I-7083BD 模組，此模組是 3 軸 32 位元編碼器輸入模組。

註:

1. 為了取得 I-7083, I-7083D, I-7083B, I-7083BD 模組的 Encoder 值，需先使用 "D_7083" 功能方塊之後，再搭配 "Counter_Start", "Counter_Stop", "Counter_Get", "Counter_State" 與 "Counter_Reset" 函式 (可參考 4.9 節) 來操作這些模組的 Encoder 通道。
2. 請先使用 "DCON Utility" 軟體 (見 P8-1) 設定好該模組的適當參數 (例如: Address、Baudrate...)，AI 模組的資料格式需設定為 **"2's Complement"**，否則 Win-GRAF PAC 無法正確地讀取該值。
3. 請在 “I/O boards” 視窗加入 “DCON” (見 8.1 節)，並設定適當的參數 (例如: Port、Baud_rate...)。
4. 只有在通訊狀態為 "TRUE" 的情況下 (若 "OK" 會回傳 "TRUE")，AI 通道的回傳值才具意義。

假設: 使用 PAC 的 COM2 來連接 I-7065 (位址 = 4)，使用 3 個數位輸入通道。



輸入參數:

EN: 資料型態: BOOL，設定為 “TRUE": 啟用； 設定為 “FALSE“: 不啟用。

Port: 資料型態: DINT，COM Port 編號 (可設定為 1 ~ 37，視 PAC 而定)。
(*** 需設定為常數，即不可變動的值 ***)

Addr: 資料型態: DINT，模組的 Net-ID 位址 (可設定為 1 ~ 255)。
(*** 需設定為常數，即不可變動的值 ***)

輸出參數:

OK: 資料型態: BOOL，"TRUE": 表示通訊正常； "FALSE": 表示通訊失敗。

Z_0 ~ Z_2: 資料型態: BOOL，3 軸 Z-index DI 值。

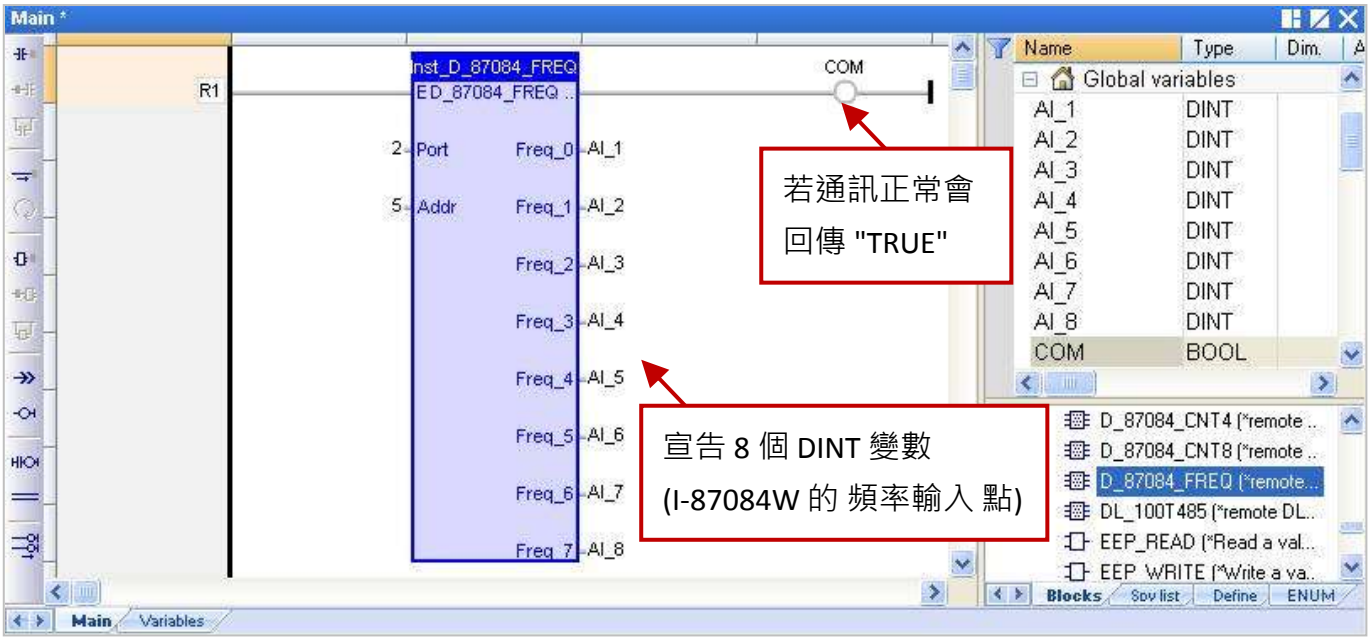
8.2.4 “D_87084_FREQ” 功能方塊

"D_87084_freq" 可連接一個 I/O 擴充單元 (例如: I-87K4/5/8/9、RU-87P4 或 RU-87P8) 上的 I-87084W 模組，並用來量測 8 通道的頻率值。

註:

1. 請先使用 "DCON Utility" 軟體 (見 P8-1) 設定好該模組的適當參數 (例如: Address、Baudrate...)，頻率的資料格式需設定為 **" Hex format"**，否則該功能將無效。
2. 請在 “I/O boards” 視窗加入 “DCON” (見 8.1 節)，並設定適當的參數 (例如: Port、Baud_rate...)。
3. 只有在通訊狀態為 "TRUE" 的情況下 (若 "OK" 會回傳 "TRUE")，AI 通道的回傳值才具意義。

假設: 使用 PAC 的 COM2 來連接 I-87084W (位址 = 5)，使用 8 個頻率輸入通道。



輸入參數:

EN: 資料型態: BOOL，設定為 “TRUE": 啟用；設定為 “FALSE“: 不啟用。

Port: 資料型態: DINT，COM Port 編號 (可設定為 1 ~ 37，視 PAC 而定)。
(*** 需設定為常數，即不可變動的值 ***)

Addr: 資料型態: DINT，模組的 Net-ID 位址 (可設定為 1 ~ 255)。
(*** 需設定為常數，即不可變動的值 ***)

輸出參數:

OK: 資料型態: BOOL，"TRUE": 表示通訊正常；"FALSE": 表示通訊失敗。

Freq_0 ~ Freq_7: 資料型態: DINT，8 通道頻率值 (單位: Hz)。

8.2.5 “D_87084_CNT4” 功能方塊

"D_87084_CNT4" 可連接一個 I/O 擴充單元 (例如: I-87K4/5/8/9、RU-87P4 或 RU-87P8) 上的 I-87084W 模組，並用來量測 4 通道的計數值。

註:

1. 請先使用 "DCON Utility" 軟體 (見 P8-1) 設定好該模組的適當參數 (例如: Address、Baudrate...)，頻率的資料格式需設定為 **" Hex format"**，否則該功能將無效。
2. 請在 “I/O boards” 視窗加入 “DCON” (見 8.1 節)，並設定適當的參數 (例如: Port、Baud_rate...)。
3. 為了由遠端 I-87084W 模組取得 4 通道的 Counter 值，需先使用 "D_87084_CNT4" 功能方塊，並搭配 "Counter_Start", "Counter_Stop", "Counter_Get", "Counter_State" 與 "Counter_Reset" 函式 (可參考 4.9 節) 來操作該模組的 Counter 通道。
4. 只有在通訊狀態為 "TRUE" 的情況下 (若 "OK" 會回傳 "TRUE")，AI 通道的回傳值才具意義。

假設: 使用 PAC 的 COM2 來連接 I-87084W (位址 = 6)，使用 4 個計數通道。

輸入參數:

EN: 資料型態: BOOL，設定為 “TRUE": 啟用；設定為 “FALSE“: 不啟用。

Port: 資料型態: DINT，COM Port 編號 (可設定為 1 ~ 37，視 PAC 而定)。
(*** 需設定為常數，即不可變動的值 ***)

Addr: 資料型態: DINT，模組的 Net-ID 位址 (可設定為 1 ~ 255)。
(*** 需設定為常數，即不可變動的值 ***)

輸出參數:

OK: 資料型態: BOOL，"TRUE": 表示通訊正常；"FALSE": 表示通訊失敗。

8.2.6 “D_87084_CNT8” 功能方塊

"D_87084_CNT8" 可連接一個 I/O 擴充單元 (例如: I-87K4/5/8/9、RU-87P4 或 RU-87P8) 上的 I-87084W 模組，並用來量測 8 通道的計數值。

註:

1. 請先使用 "DCON Utility" 軟體 (見 P8-1) 設定好該模組的適當參數 (例如: Address、Baudrate...)，頻率的資料格式需設定為 **" Hex format"**，否則該功能將無效。
2. 請在 “I/O boards” 視窗加入 “DCON” (見 8.1 節)，並設定適當的參數 (例如: Port、Baud_rate...)。
3. 為了由遠端 I-87084W 模組取得 8 通道的 Counter 值，需先使用 "D_87084_CNT8" 功能方塊，並搭配 "Counter_Start", "Counter_Stop", "Counter_Get", "Counter_State" 與 "Counter_Reset" 函式 (可參考 4.9 節) 來操作該模組的 Counter 通道。
4. 只有在通訊狀態為 "TRUE" 的情況下 (若 "OK" 會回傳 "TRUE")，AI 通道的回傳值才具意義。

假設: 使用 PAC 的 COM2 來連接 I-87084W (位址 = 7)，使用 8 個計數通道。

輸入參數:

EN: 資料型態: BOOL，設定為 “TRUE": 啟用；設定為 “FALSE“: 不啟用。

Port: 資料型態: DINT，COM Port 編號 (可設定為 1 ~ 37，視 PAC 而定)。
(*** 需設定為常數，即不可變動的值 ***)

Addr: 資料型態: DINT，模組的 Net-ID 位址 (可設定為 1 ~ 255)。
(*** 需設定為常數，即不可變動的值 ***)

輸出參數:

OK: 資料型態: BOOL，"TRUE": 表示通訊正常；"FALSE": 表示通訊失敗。

8.2.7 “DL_100T485” 功能方塊

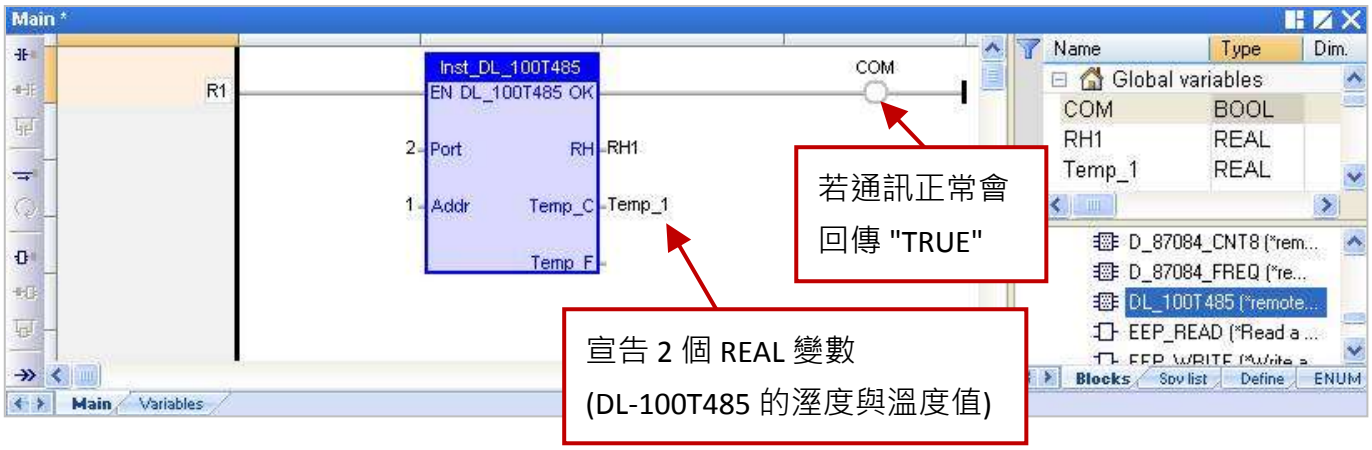
"DL_100T485" 可連接一個遠端的 DL_100T485 模組，來讀取溫度與濕度值。

產品網頁: www.icpdas.com/root/product/solutions/remote_io/rs-485/dl_series/dl-100t485.html

註:

1. 請先使用隨貨光碟中的 "DL-100T485 Utility" 軟體，設定好該模組的適當參數 (例如: Module ID)，DL-100T485 預設 Address (ID) 為 "1"，Baudrate 為 "9600"，Checksum 為 "Disable"。
2. 請在 “I/O boards” 視窗加入 “DCON” (見 8.1 節)，並設定適當的參數 (例如: Port、Baud_rate...)。
3. 只有在通訊狀態為 "TRUE" 的情況下 (若 "OK" 會回傳 "TRUE")，AI 通道的回傳值才具意義。

假設: 使用 PAC 的 COM2 來連接 DL_100T485 (位址 = 1)，用來量測濕度值。



輸入參數:

EN:	資料型態: BOOL，設定為 “TRUE": 啟用；設定為 “FALSE“: 不啟用。
Port:	資料型態: DINT，COM Port 編號 (可設定為 1 ~ 37，視 PAC 而定)。 (*** 需設定為常數，即不可變動的值 ***)
Addr:	資料型態: DINT，模組的 Net-ID 位址 (預設為 "1"，可設定為 1 ~ 255)。 (*** 需設定為常數，即不可變動的值 ***)

輸出參數:

OK:	資料型態: BOOL，"TRUE": 表示通訊正常；"FALSE": 表示通訊失敗。
RH:	資料型態: REAL，回傳值為相對溼度 (單位: %)。 例如: 回傳值為 "45.7" 表示 45.7%。
Temp_C:	資料型態: REAL，回傳值為攝氏溫度 例如: 回傳值為 "25.7" 表示 25.7 ℃。
Temp_F:	資料型態: REAL，回傳值為華氏溫度 例如: 回傳值為 "78.26" 表示 78.26 ℉。

第 9 章　即時線上更新 (On Line Change)

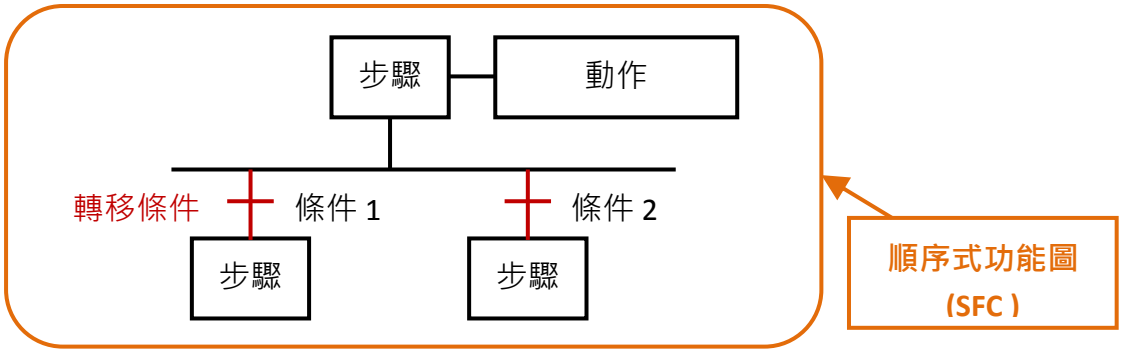
"On Line Change" 功能允許 Win-GRAF PAC 在運行的同時更新小幅修改過的程式，此修改的程式名稱必須與 PAC 中目前正在 Run 的程式是同一個。"On Line Change" 功能主要提供給緊急狀況使用，像是應用場所不允許因為要更換應用程式而短暫的停機或停止運作，也找不到時間可供更換新的應用程式 (例如，需 24 小時運作不能停止的設備)。若非以上狀況，最好不要使用此功能，您可先停止運行中的應用程式，再將修改過的程式下載到 PAC 中 (見 2.3.5 節)，這是比較安全的作法。

9.1　"On Line Change" 功能的使用限制

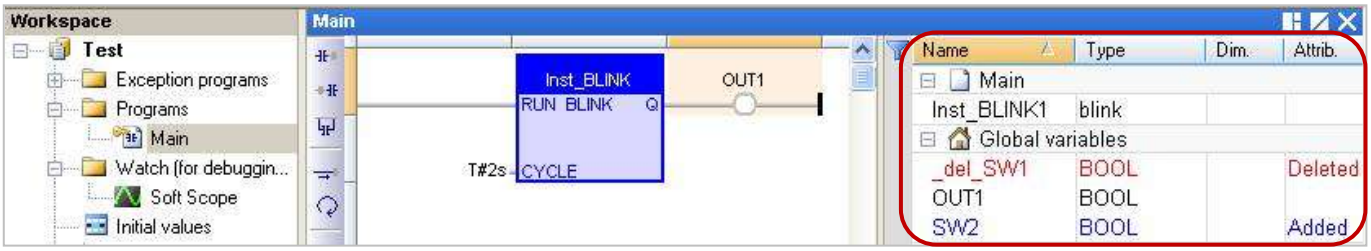
在啟用 "On Line Change" 功能之前，請先了解下列使用限制:

啟用 "On Line change" 功能後，可執行下列變更 (不停止原有程式):

- 變更 單一程式中的程式碼。
- 變更 單一順序式功能圖 (SFC) 的條件 或 動作。

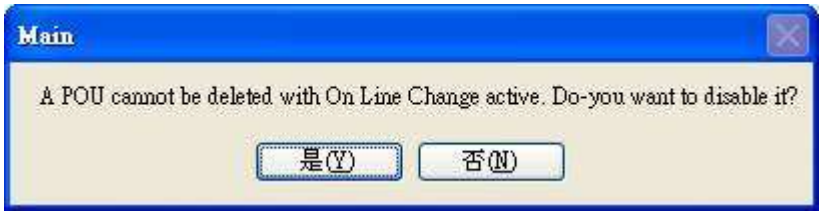


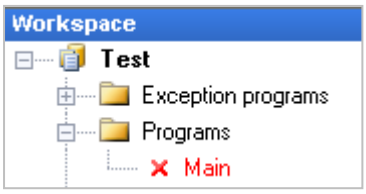
- 新增、更名、刪除 全域 (Global) 或 區域 (Local) 變數。
- 新增、更名、刪除 全域 (Global) 或 區域 (Local) 功能方塊 的樣例變數。



啟用 "On Line change" 功能後，不允許以下變更:

- 新增、更名、刪除程式。(若刪除程式，會出現以下警告訊息)



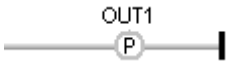
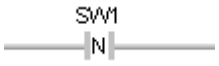
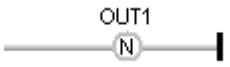


- 變更 順序式功能圖 (SFC)。
- 變更 使用者自定功能方塊 (UDFB) 的區域參數或變數。
- 變更 變數 或 功能方塊樣例變數 的類型、陣列 Dim 長度與字串長度。
- 變更 "I/O boards" 視窗中的設定。

此外，啟用 "On Line change" 功能後，以下的程式寫法是不安全的:

- 脈波 (P 或 N) 接點 與 線圈 (邊緣偵測)。

  ✍ 請改成使用已定義的 "R_TRIG" 與 "F_TRIG" 功能方塊的樣例變數。

上升脈波偵測		
P (False > True)	**修改前**	**修改後**
		
		
下降脈波偵測		
N (True > False)		
		

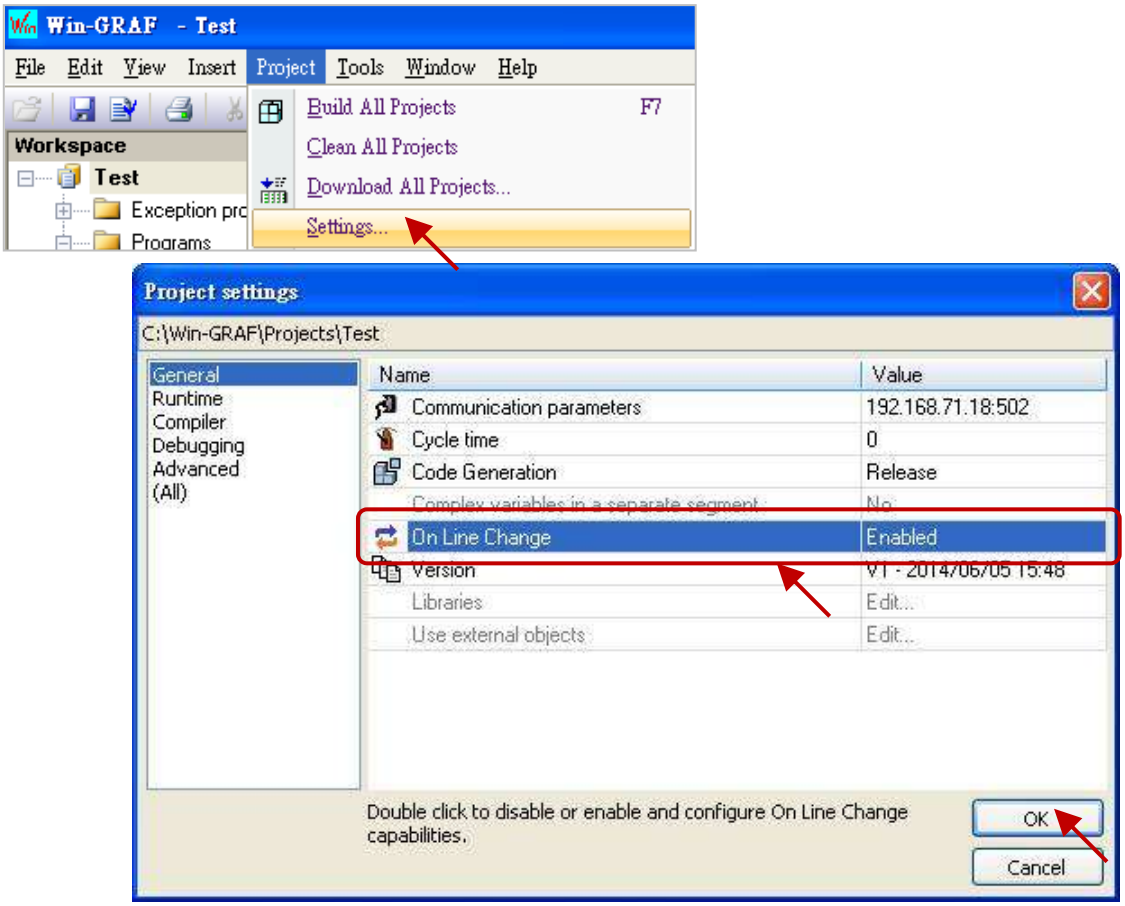
- FBD 的迴圈中含有未定義的變數連結。

  ✍ 您必須明確地指定迴圈中的變數。

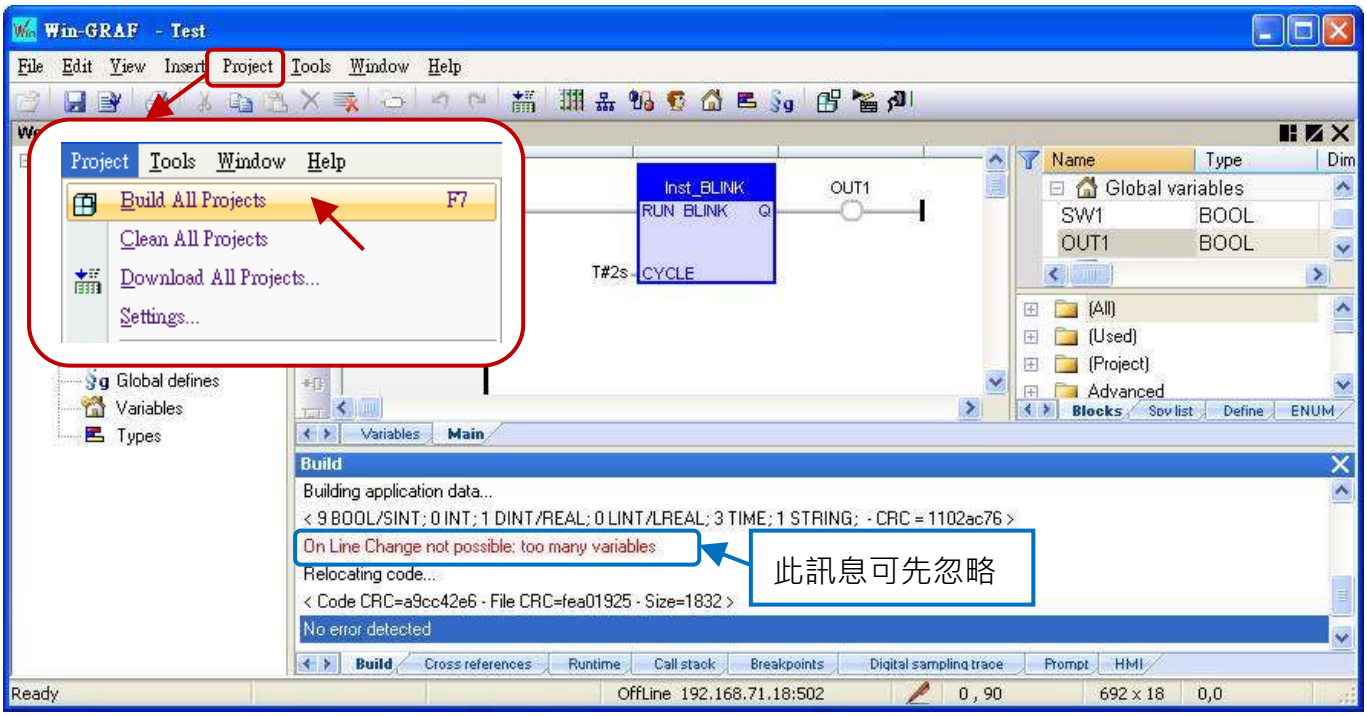
9.2 使用 "On Line change" 功能

啟用 "On Line change" 功能:

1. 滑鼠點選功能表 "Project > Settings..."，再雙擊 "On Line Change" 項目將其設定為 "Enabled"。



2. 接著，需點選功能表 "Project > Build All Projects"，執行程式編譯才能執行後續設定。



配置變數使用量:

當啟用 "On Line Change" 功能，為了允許新的變數與方塊宣告，您必須為各種資料型態的變數預留可使用的數量。

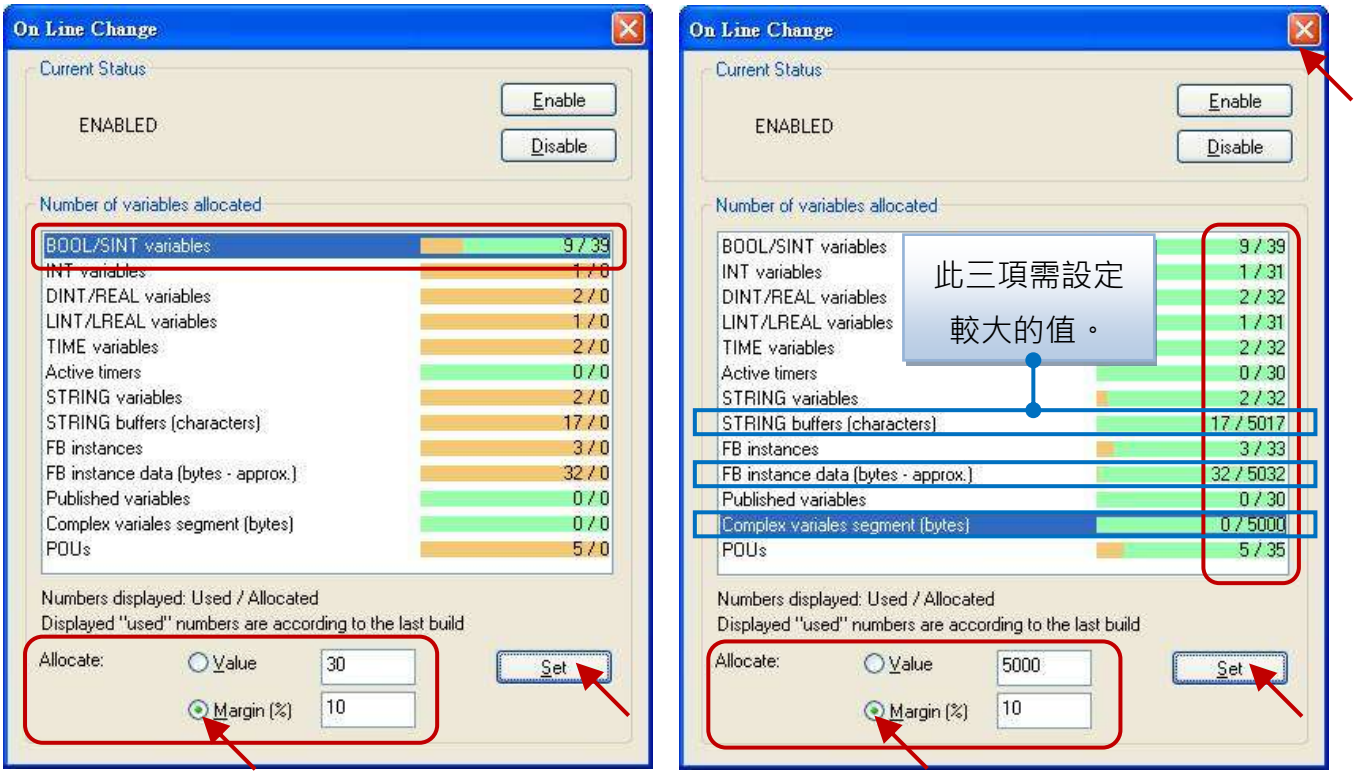
3. 同步驟 1，滑鼠點選功能表 "Project > Settings..."，再雙擊 "On Line Change" 項目進入設定視窗。請在 “Value" 或 “Margin" 欄位設定需新增的數量。

   註: 若 “Value" 與 “Margin" 兩者皆有設定值，會取用較大的值。此例，"Value" 填入 "30" 且 “Margin" 填入"10"，由於顯示數值 x 10% 小於 30，因此會啟用較大值 "30"。

4. 接著，點選所需的 資料型態項目 並按 "Set“ 按鈕以完成設定。
   (例如: 點選 "BOOL/SINT variables" 並按 "Set“ 後，數量為 9 + 30 = 39)。

   註: "STRING buffers (characters)"、"FB instance data (bytes – approx.)" 與 "Complex variables segment (bytes)，此三項需設定較大的值 (此例設定為 "5000")。

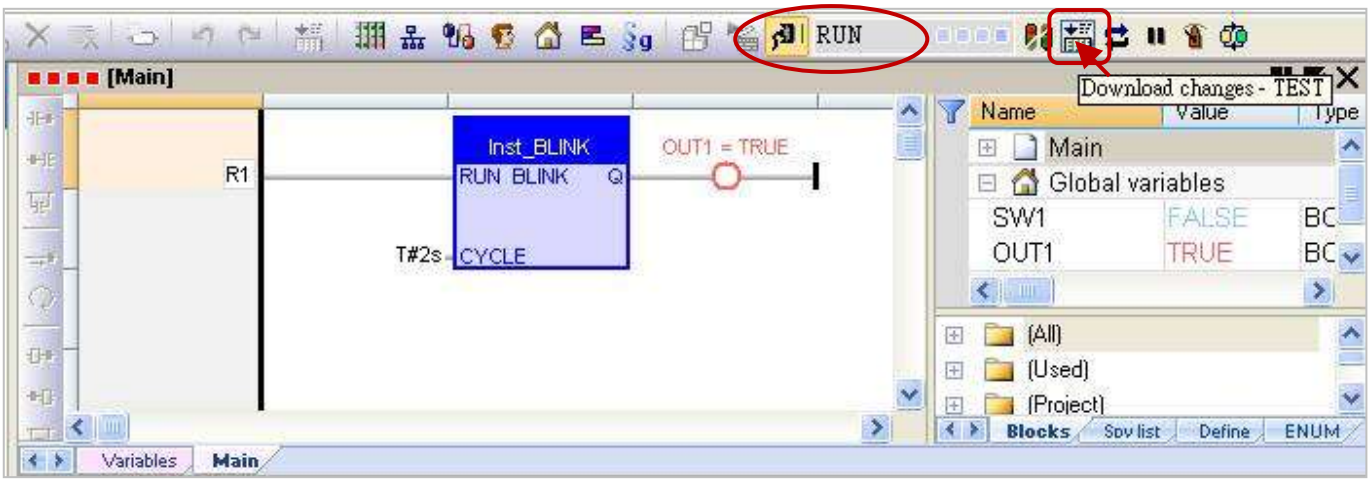
5. 設定完成後畫面如下，可點選右上角的 "X" 按鈕離開設定畫面。



6. 點選功能表 "Project" 再選擇 "Build All Projects"，再次編譯程式。接著，再選擇 “On Line”，或點選工具按鈕  來與 PAC 建立連線。(若不熟悉連線設定，可參考 2.3.5 節)

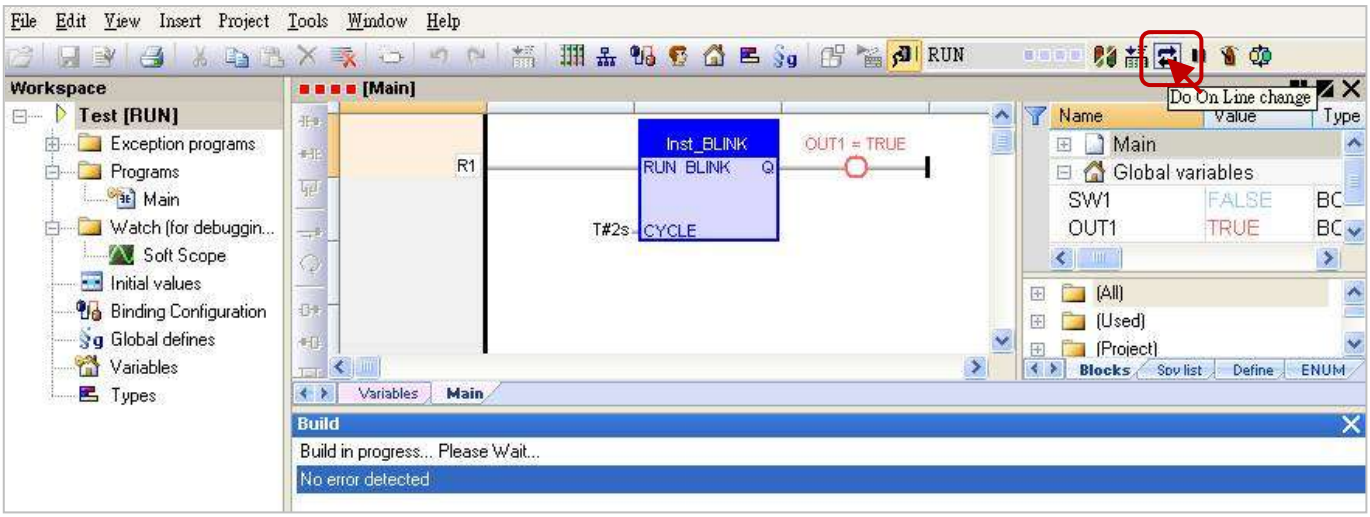
7. 連線成功後，點選工具列 "Download changes" 按鈕，將程式下載到 PAC 中。

注意: "On Line Change" 功能，僅適用在對原本的程式進行小幅度的修改 (不需停止運行)，若 PAC 中原先運行的程式與目前檔名不同，則需停止程式運行再重新下載 (參考附錄 B)。



8. 點選工具列 "Do On Line change" 按鈕，來執行此功能。

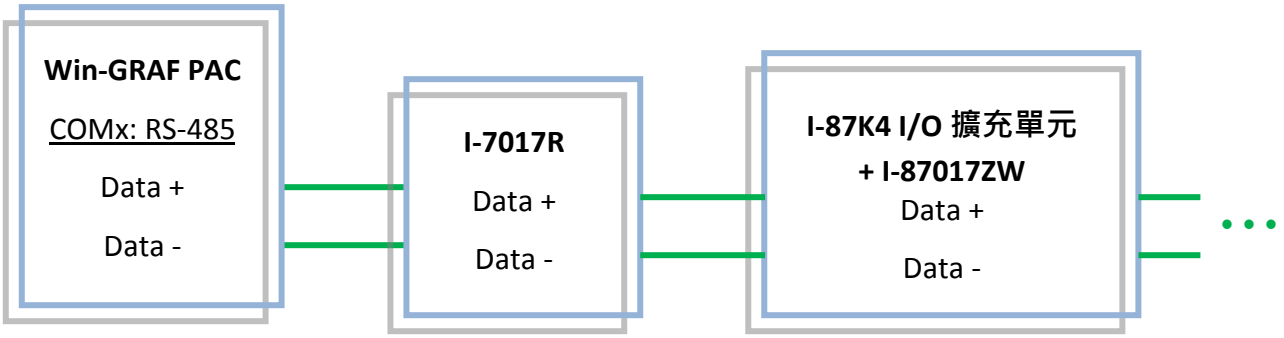
注意: 由於執行 "On Line Change" 後，為了保護系統的正常運作會有一些使用限制 (參考 9.1 節)，因此請確認程式無誤後，再執行此功能。



第 10 章 資料/型態轉換 與 使用 PAC 時間

10.1 AI 資料轉換

若在 PAC 的 Slot 0 ~ 7 使用 AI 模組 (例如: I-8017HW)，並且想把 AI 輸入訊號 (例如: "4 ~ 20 mA" 或 "0 ~ 10 V") 轉換為使用者所需的工程應用值 (例如: 0 ~ 10000) 可參考 4.3 節。然而，若是使用遠端的 AI 模組 (例如: 透過 PAC 的 RS-485 Port 去連接的 I-87017ZW 或 I-7017R 模組)，則可參考以下設定:



1. 首先，在 "I/O Boards" 視窗中，連上 "i_scale"，並滑鼠雙擊 "i_scale_x" 開啟屬性視窗。
   (若不熟悉操作，請參考 4.2 節)
   注意: "I/O Boards" 視窗中只能使用一個 "i_scale" (請勿連接兩個或多個)。

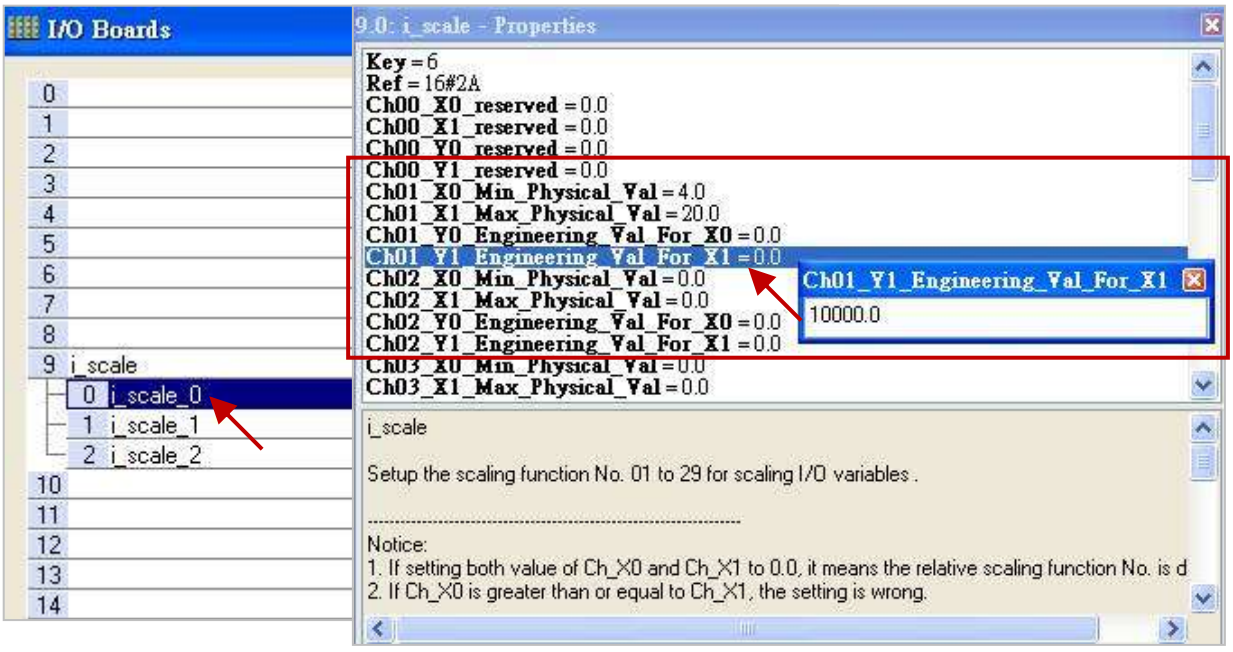
2. 設定需啟用的轉換函式編號 與 數值 (例如: 使用函式 1 並將 "4 ~ 20 mA" 轉換為 "0 ~ 10000")。

   Ch**01**_X0_Min_Physical_Val: “4.0”。

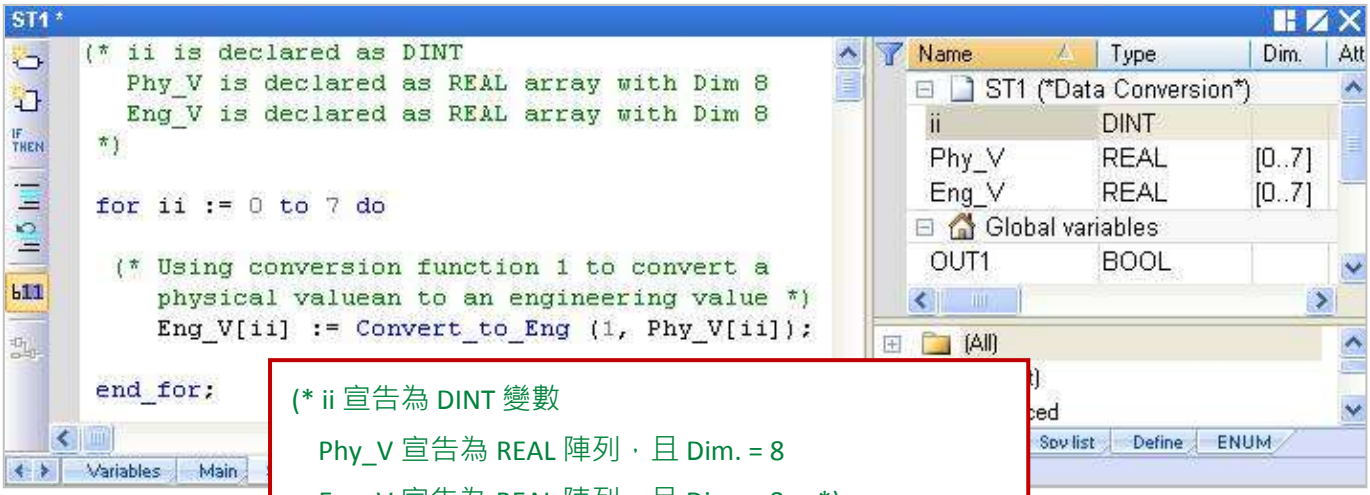
   Ch**01**_X1_Max_Physical_Val: “20.0”。

   Ch**01**_Y0_Engineering_Val_For_X0: “0.0”。

   Ch**01**_Y1_Engineering_Val_For_X1: “10000.0”。



3. 編寫一個 ST 程式，用來將實體值 (例如: Phy_V[0] ~[7]) 轉換為 工程值 (例如: Eng_V[0] ~[7])。
(若不熟悉程式建立方式，可參考 2.3.3 節)

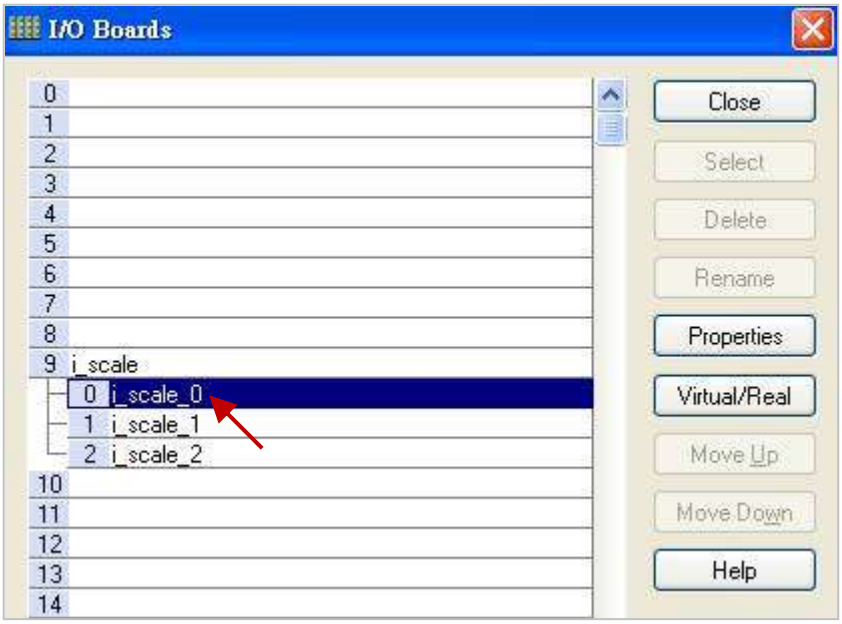



```
(* ii 宣告為 DINT 變數
   Phy_V 宣告為 REAL 陣列，且 Dim. = 8
   Eng_V 宣告為 REAL 陣列，且 Dim. = 8    *)
for ii := 0 to 7 do
   (* 套用轉換函式 1，將實體值轉換為工程值 *)
   Eng_V[ii] := Convert_to_Eng (1, Phy_V[ii]);
end_for;
```

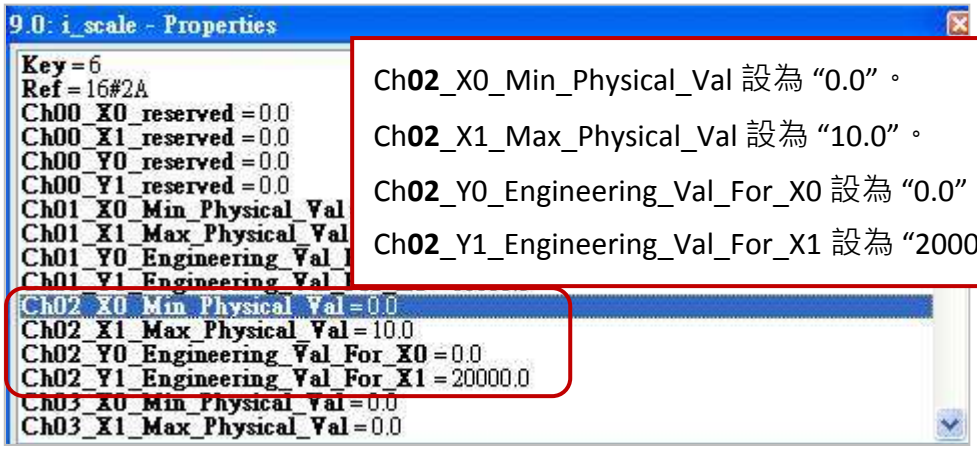
10.2 AO 資料轉換

若在 PAC 的 Slot 0 ~ 7 使用 AO 模組 (例如: I-8024W)，並且想把使用者自己的 工程應用值 轉換為 AO 輸出訊號 (例如: "0 ~ 20000" 轉換為 "0 ~ 10 V") 可參考 4.4 節。然而，若是使用 DCON 遠端 AO 模組 (例如: 透過 PAC 的 RS-485 Port 去連接的 I-7024 模組)，則可參考以下設定:

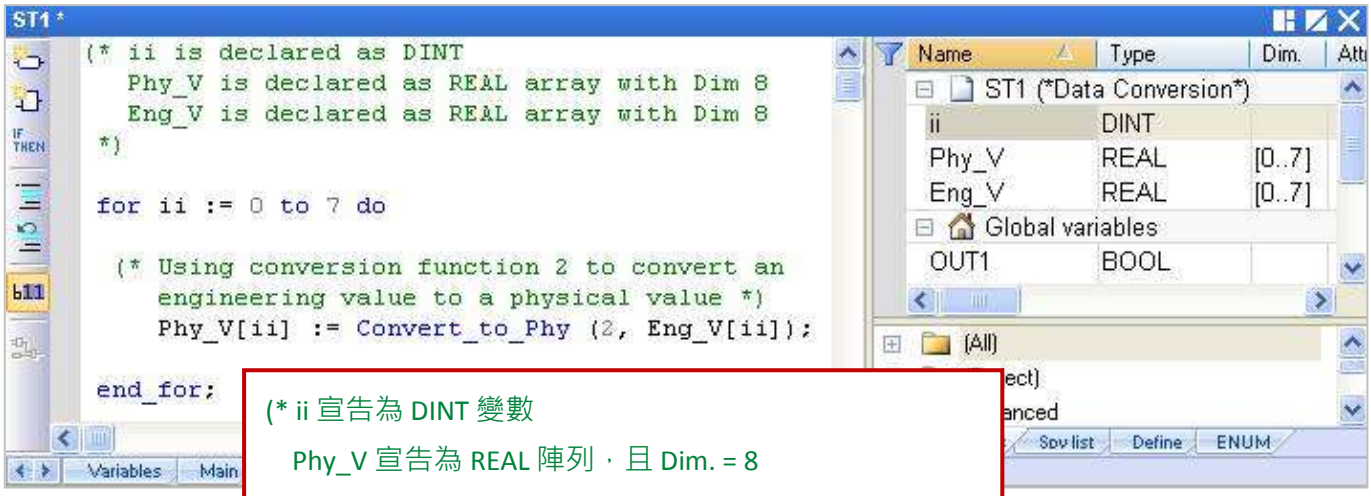
1. 在 "I/O Boards" 視窗中，連上 "i_scale"，並滑鼠雙擊 "i_scale_x" 開啟屬性視窗。(參考 10.1 節)。
注意: "I/O Boards" 視窗中只能使用一個 "i_scale" (請勿連接兩個或多個)。



2. 設定需啟用的 轉換函式編號 與 數值 (例如: 使用函式 2 並將 "0 ~ 20000" 轉換為 "0 ~ 10 V")。



3. 編寫一個 ST 程式，用來將工程值 (例如: Eng_V[0] ~[7]) 轉換為 實體值 (例如: Phy_V[0] ~[7])。
(若不熟悉程式建立方式，可參考 2.3.3 節)




```
(* ii 宣告為 DINT 變數
   Phy_V 宣告為 REAL 陣列，且 Dim. = 8
   Eng_V 宣告為 REAL 陣列，且 Dim. = 8    *)
for ii := 0 to 7 do
   (* 套用轉換函式 2，將工程值轉換為實體值 *)
   Phy_V[ii] := Convert_to_Phy (2, Eng_V[ii]);
end_for;
```

10.3 資料型態轉換

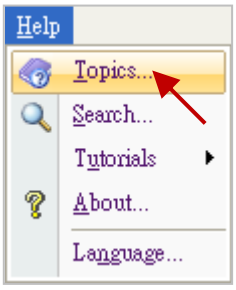
當不同的資料型態要進行 " + , - , * , / " 運算 或 " > , < , = , <= , >= , <> (不等於)" 判斷，或使用函式內參數的 變數或型態不一樣時，需先使用 型態轉換函式 (如下表) 來轉換為一樣的資料型態，才能正常使用。

型態轉換函式	說明	型態轉換函式	說明
ANY_TO_BOOL	轉換為 布林 (Boolean)	ANY_TO_REAL	轉換為 實數
ANY_TO_SINT	轉換為 短整數 (8 bit)	ANY_TO_LREAL	轉換為 雙精準實數
ANY_TO_INT	轉換為 整數 (16 bit)	ANY_TO_STRING	轉換為 字元字串
ANY_TO_DINT	轉換為 長整數 (32 bit – 預設)	NUM_TO_STRING	將數字轉為字串， 可指定小數點後轉換位數
ANY_TO_LINT	轉換為 Large 整數 (64 bit)	ATOH	將 16 進制字串轉為整數
ANY_TO_TIME	轉換為 時間格式	HTOA	將整數轉為 16 進制字串

例如，以下的 ST 程式，會將 DINT 變數先轉換為實數後，再來進行運算。

```
REAL_Val_1  :=  ANY_TO_REAL (DINT_Val_1) * 3.5 + 4.8 ;
```

您可在功能表中開啟 "HTML Help"，並輸入欲搜尋的字串，來了解詳細的設定說明。



10.4 BCD 轉換

BCD 碼是用 4-bit 來表示十進制數字 0 ~ 9。假設有個數值十進制是 "1 3 2" 換算為 BCD 碼是 "0001 0011 0010"，換算為二進制則是 10000100 (即 $2^7 + 2^2 = 128 + 4 = 132$)。

十進制	BCD				說明
	2^3	2^2	2^1	2^0	
0	0	0	0	0	0
1	0	0	0	1	$2^0 = 1$
2	0	0	1	0	$2^1 = 2$
3	0	0	1	1	$2^1 + 2^0 = 3$
4	0	1	0	0	$2^2 = 4$
5	0	1	0	1	$2^2 + 2^0 = 5$
6	0	1	1	0	$2^2 + 2^1 = 6$
7	0	1	1	1	$2^2 + 2^1 + 2^0 = 7$
8	1	0	0	0	$2^3 = 8$
9	1	0	0	1	$2^3 + 2^0 = 9$

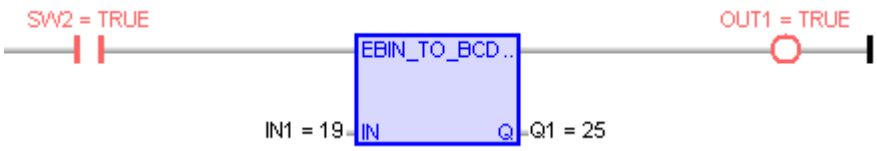
註:
BCD 碼只能表示 0 ~ 9 的數字。此 6 種數值 (1010, 1011, 1100, 1101, 1110, 1111)，將無法使用，會回傳 "0"。

下表中的函式可用來進行 BCD (Binary Coded Decimal) 數值轉換。

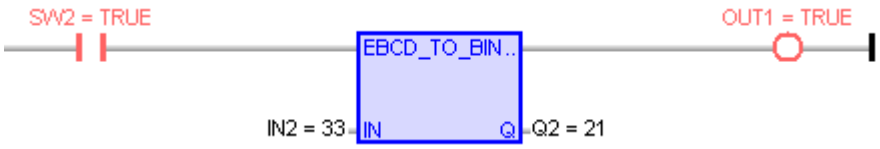
型態轉換功能	說明
BIN _TO_BCD	將二進碼 轉換為 BCD 值。
BCD_TO_BIN	將 BCD 值 轉換為 二進碼。

註: 若輸入 <= 0，皆會顯示 "0"。
IN1, IN2, Q1, Q2 宣告為 DINT。

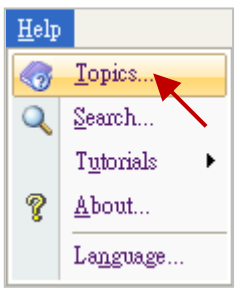
BIN _TO_BCD:
19$_{(10)}$ = 0001 1001$_{(BCD)}$
= $2^4 + 2^3 + 2^0$ = 16 + 8 + 1 = **25**



BCD_TO_BIN:
33$_{(10)}$ = 0010 0001$_{(2)}$ = **21**$_{(BCD)}$
15$_{(10)}$ = 0000 ~~1111~~$_{(2)}$ = **0**$_{(BCD)}$



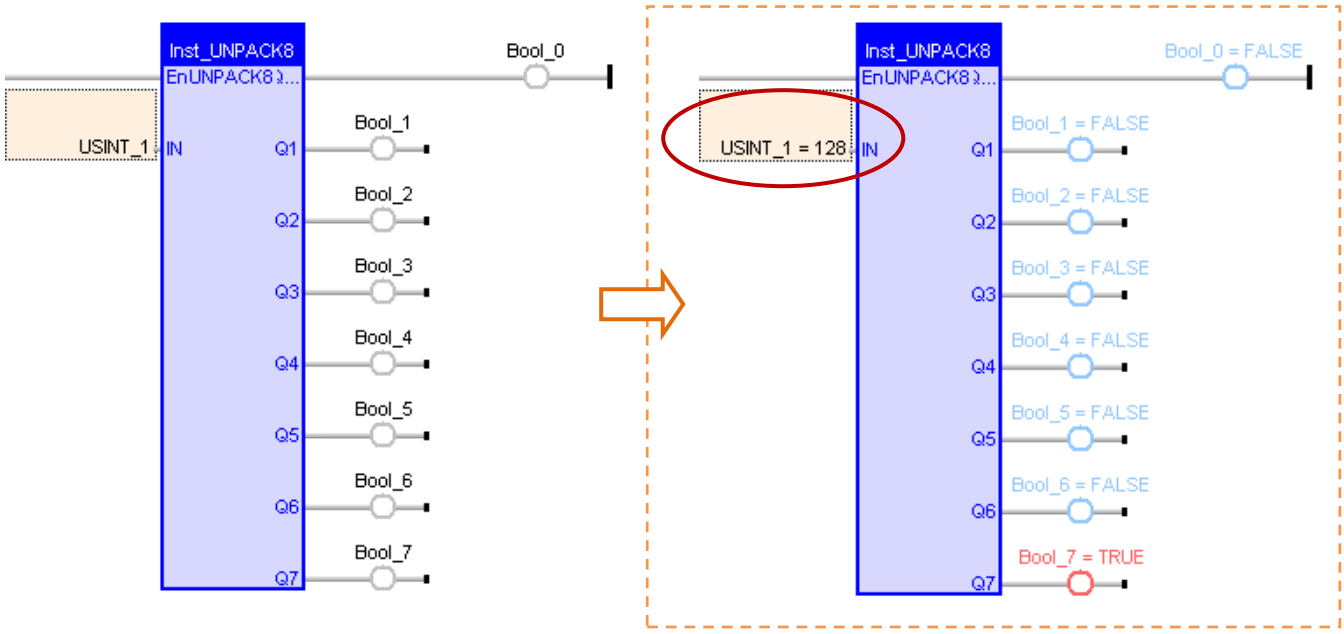
您可在功能表中開啟 "HTML Help"，並輸入欲搜尋的字串，來了解詳細的設定說明。



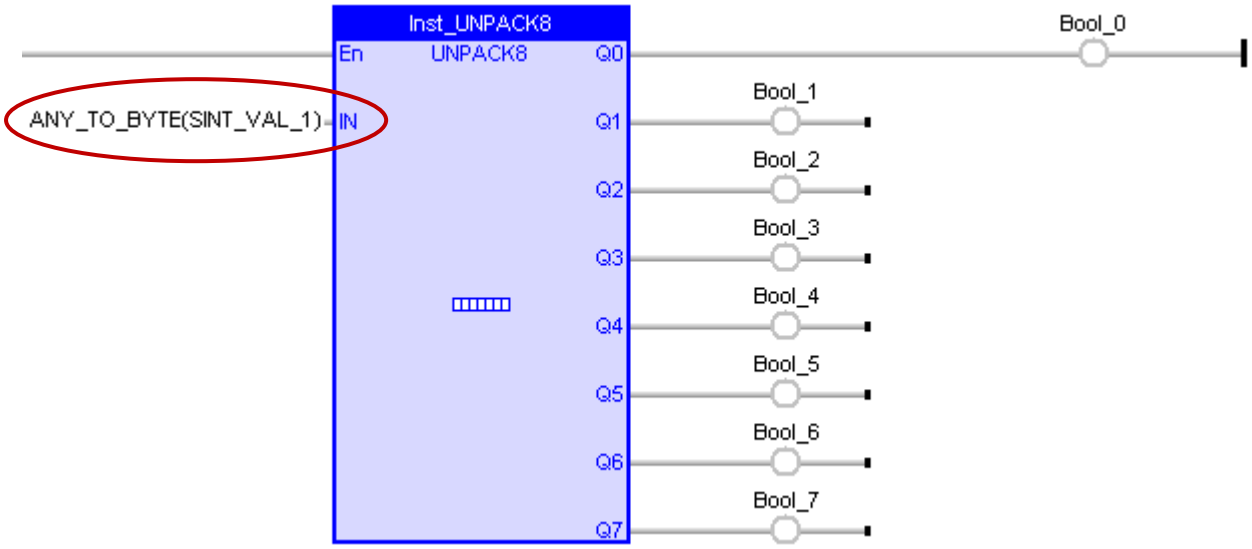
10.5 組合/拆解 整數或布林值

將整數值拆解為布林值:

若想把 1 個 BYTE (或 USINT，範圍: 0 ~ 255) 拆解為 8 個 Boolean，可使用 "UNPACK8" 函式。



若想把 1 個 SINT 拆解為 8 個 Boolean，需先使用 ST 程式 "ANY_TO_BYTE()" 將 SINT 轉為 BYTE 型態，方法如下:



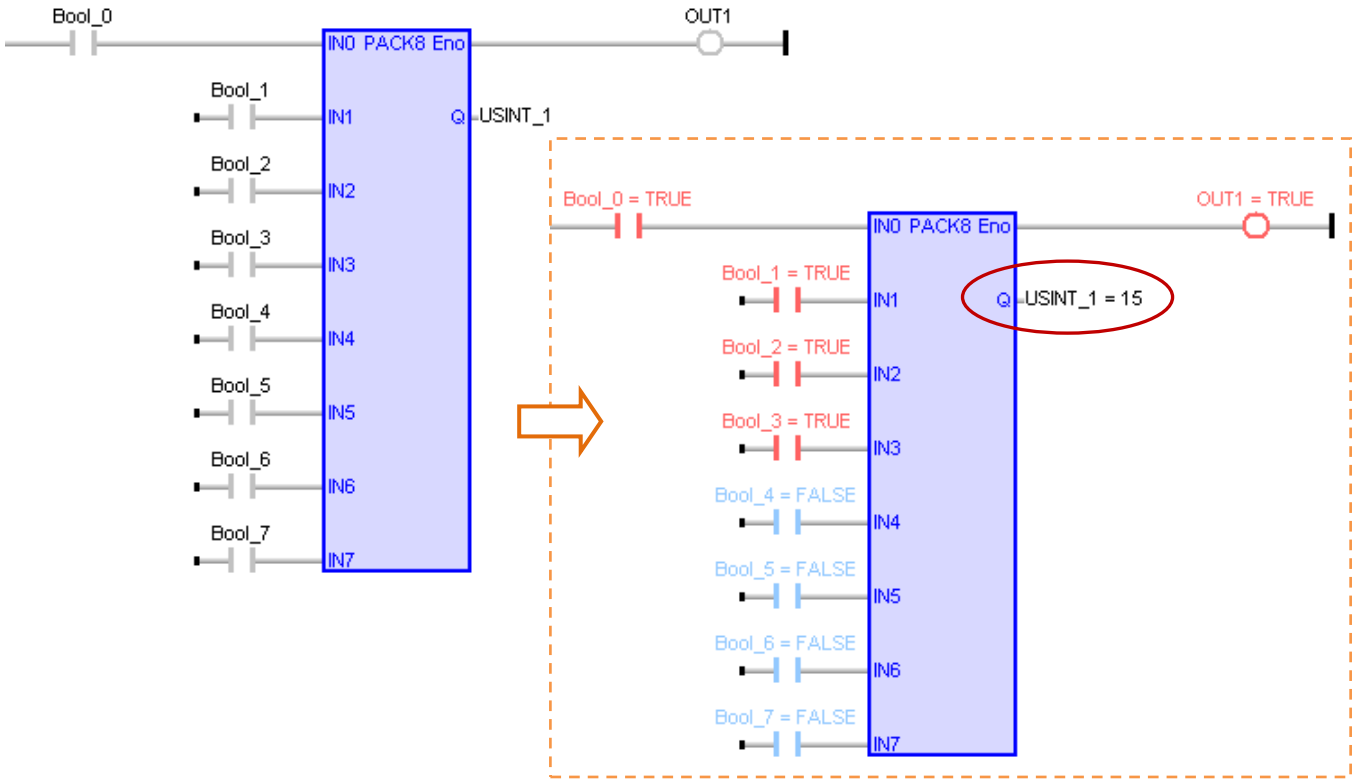
將布林值組合為整數值:

若想把 8 個 Boolean 組合成 1 個 BYTE (或 USINT，範圍: 0 ~ 255) ，可使用 "PACK8" 函式。

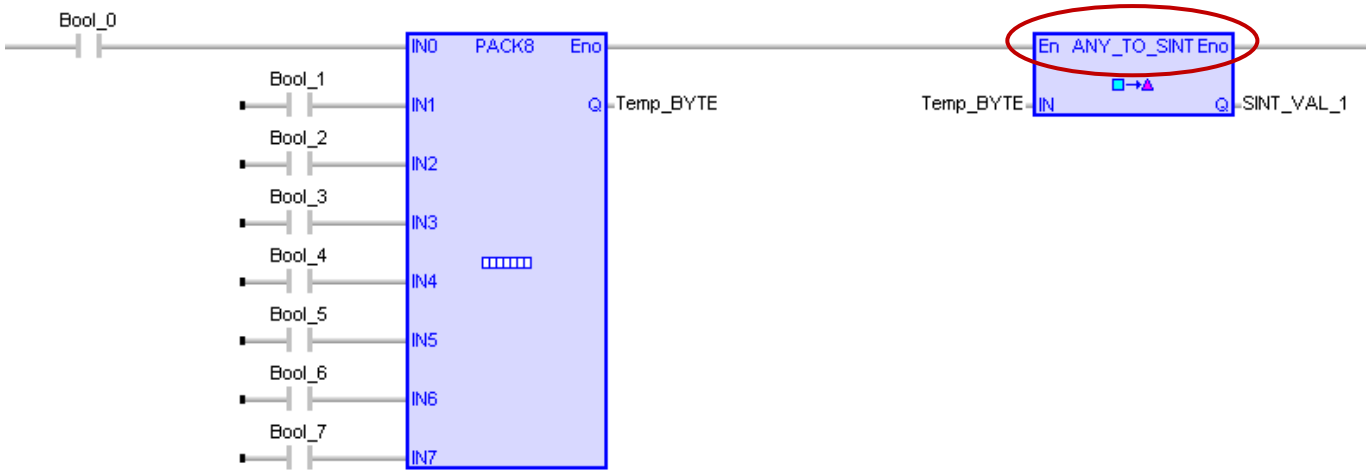
ST 語法:

```
USINT_1  :=  PACK8 (Bool_0, I Bool_1, Bool_2, Bool_3, Bool_4, Bool_5, Bool_6, Bool_7);
```

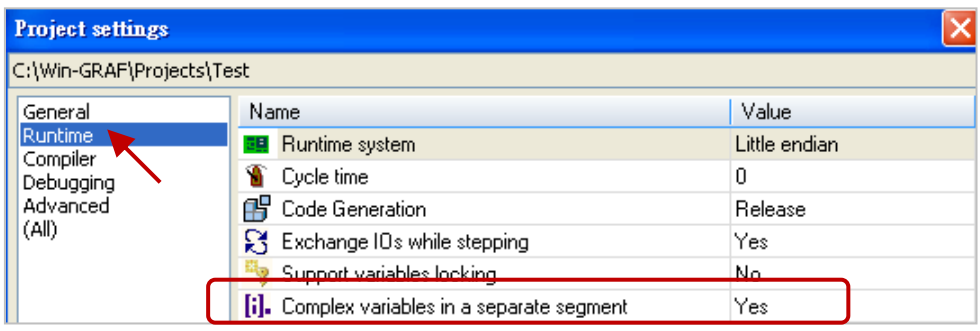
LD 語法:



若想把 8 個 Boolean 組合成 1 個 SINT，輸出 (Q) 需指定一個 "BYTE" 變數來暫存數值，並使用一個 "ANY_TO_SINT" 函式將 BYTE 轉為 SINT 型態，方法如下:



注意: 如有編譯失敗的情況，請點選功能表 "Project" > "Settings"，並檢查 "Runtime" 設定中的 "Complex variables in a separate segment" 是否為 "Yes"。



10.6 組合/拆解 BYTE, WORD, DWORD

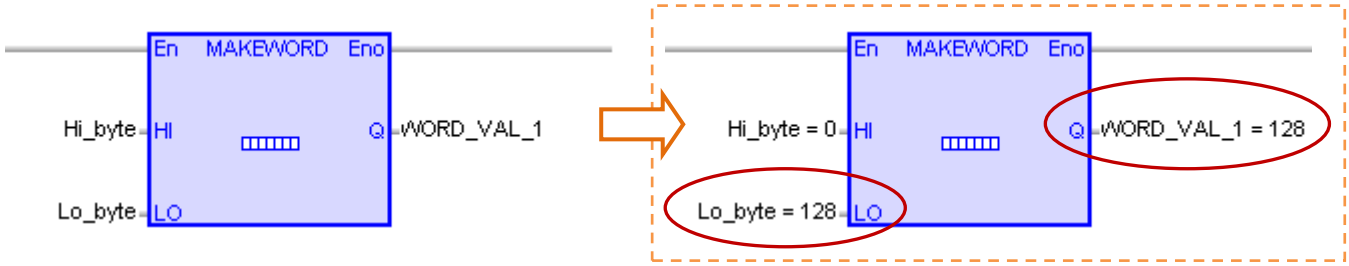
將 2 個 8-bit 組成 16-bit 資料

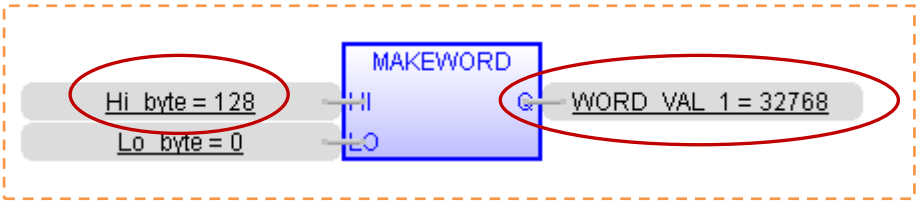
若想將 2 個 Byte (或 USINT) 組成 1 個 WORD (或 UINT)，可使用 "MAKEWORD" 函式。

ST 語法:

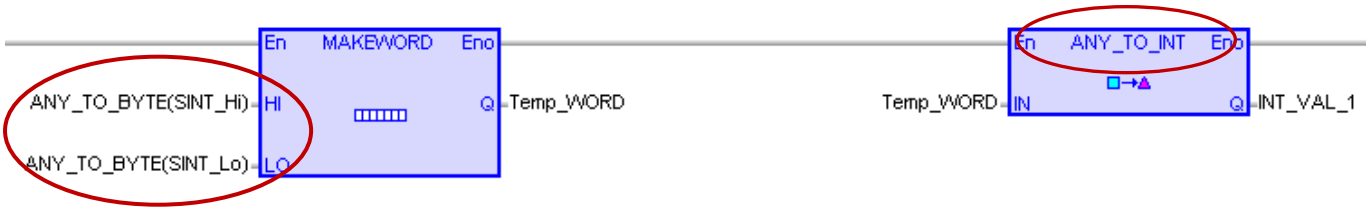
```
WORD_VAL_1 :=  MAKEWORD (Hi_byte, Lo_byte);
```

LD/FBD 語法:





若想將 2 個 SINT 組成 1 個 INT，需先使用 ST 程式 "ANY_TO_BYTE()" 將 SINT 轉為 BYTE，再使用一個 "ANY_TO_INT" 函式，將組成的 WORD 轉為 INT 型態。



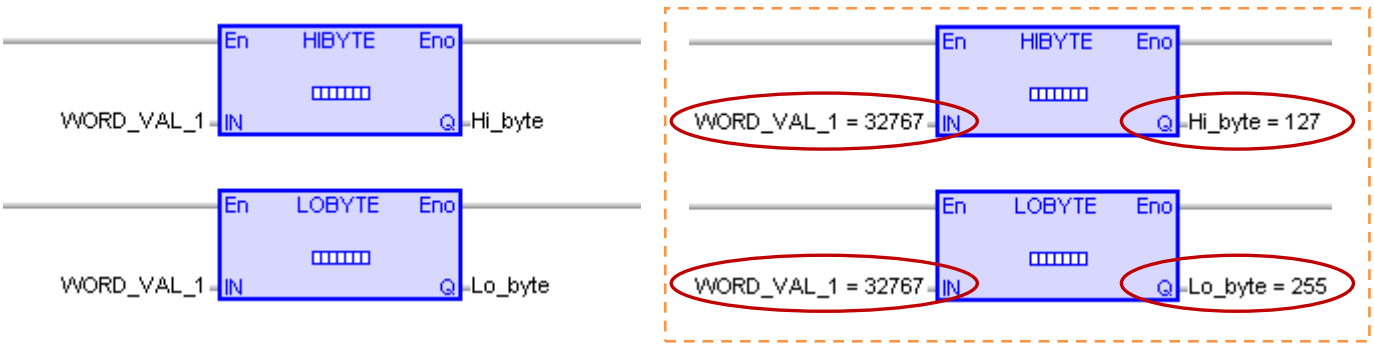
將 1 個 16-bit 拆解為 2 個 8-bit 資料

若想將 1 個 WORD (或 UINT) 拆解為 2 個 Byte (或 USINT)，可使用 "HIBYTE"、"LOBYTE" 函式。

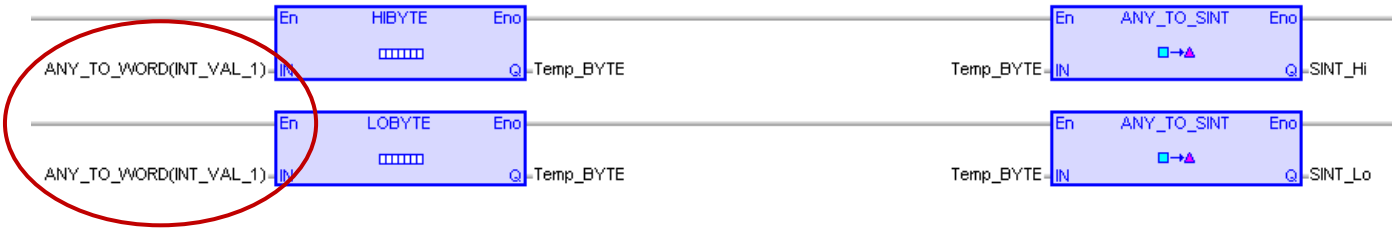
ST 語法:

```
Hi_byte  :=  HIBYTE (WORD_VAL_1);
Lo_byte  :=  LOBYTE (WORD_VAL_1);
```

LD/FBD 語法:



若想將 1 個 INT 拆解為 2 個 SINT，需先使用 ST 程式 "ANY_TO_WORD()" 將 INT 轉為 WORD，再使用一個 "ANY_TO_SINT" 函式將拆解的 BYTE 轉為 SINT 型態。



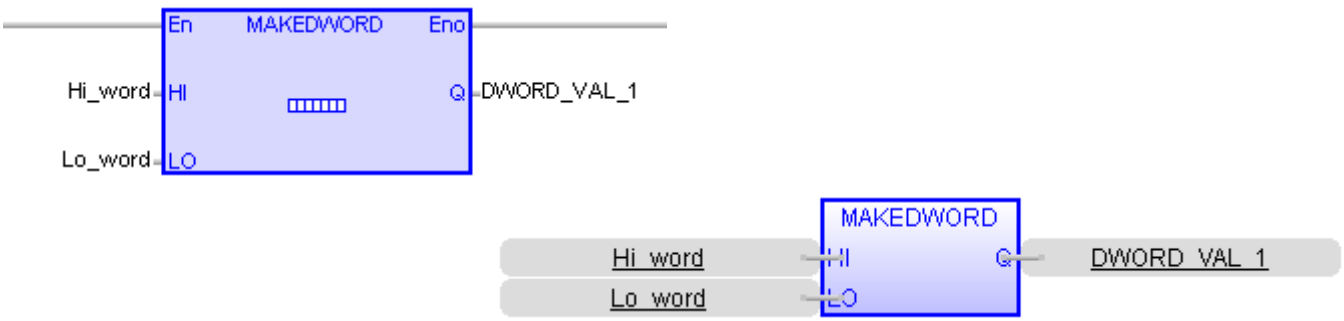
將 2 個 16-bit 組成 32-bit 資料

若想將 2 個 WORD (或 UINT) 組成 1 個 DWORD (或 UDINT)，可使用 "MAKE**D**WORD" 函式。

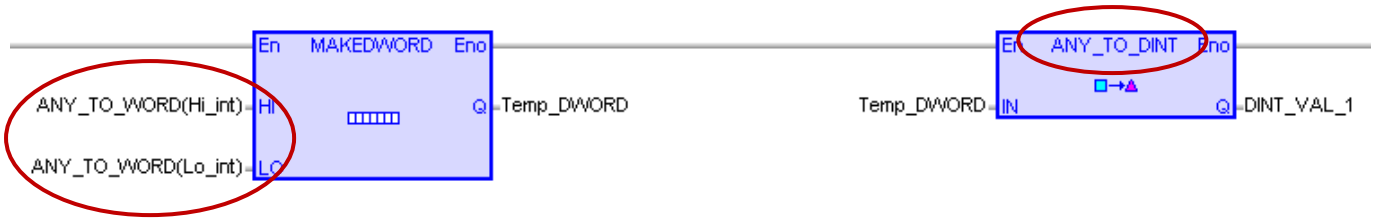
ST 語法:

```
DWORD_VAL_1 :=  MAKEDWORD (Hi_word, Lo_word);
```

LD/FBD 語法:



若想將 2 個 INT 組成 1 個 DINT，需先使用 ST 程式 "ANY_TO_WORD()" 將 INT 轉為 WORD，再使用一個 "ANY_TO_DINT" 函式，將組成的 DWORD 轉為 DINT 型態。



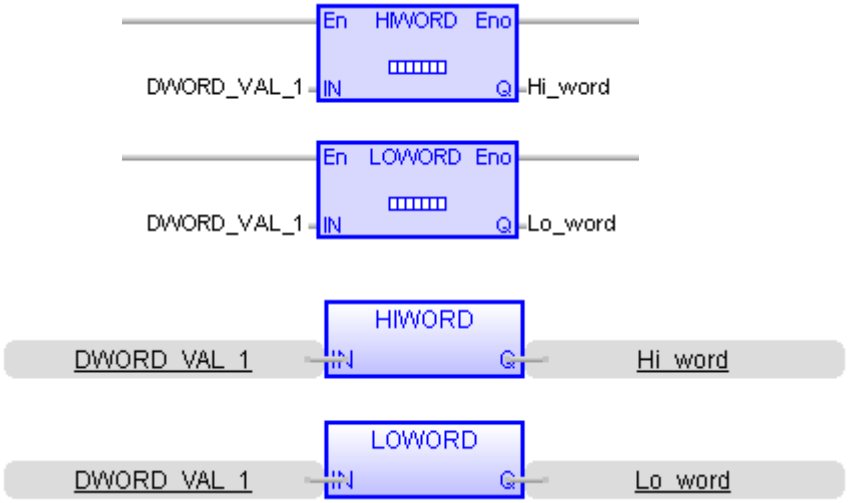
<u>將 1 個 32-bit 拆解為 2 個 16-bit 資料</u>

若想將 1 個 DWORD (或 UDINT) 拆解為 2 個 WORD (或 UINT)，可使用 "HIWORD"、"LOWORD" 函式。

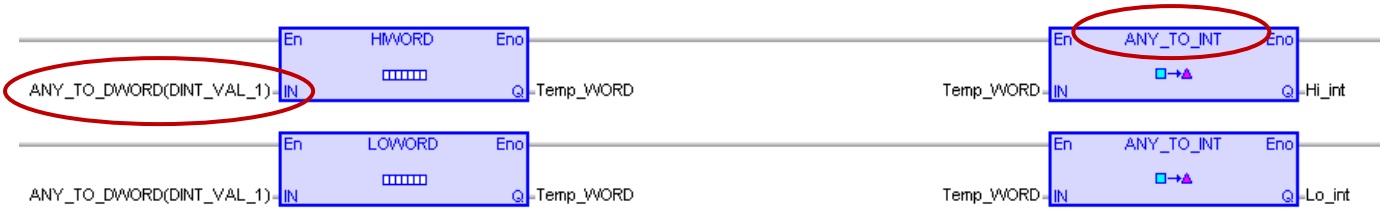
<u>ST 語法:</u>

```
Hi_word  :=  HIWORD (DWORD_VAL_1);
Lo_word  :=  LOWORD (DWORD_VAL_1);
```

<u>LD/FBD 語法:</u>



若想將 1 個 DINT 拆解為 2 個 INT，需先使用 ST 程式 "ANY_TO_DWORD()" 將 DINT 轉為 DWORD，再使用一個 "ANY_TO_INT" 函式，將拆解的 WORD 轉為 INT 型態。



10.7 將變數拆成 Byte Array 或將 Byte Array 組成變數

"SerializeOut" 函式可以將一個 Win-GRAF 變數值拆解成 Byte Array (或 USINT Array)；
"SerializeIn" 函式則可以將一個 Byte Array (或 USINT Array) 組成一個 Win-GRAF 變數值。

註: 1. Array 的 Dim. 至少要設定為 "8"。
2. 此 "Serialize" 函式不可使用 STRING 變數。

您可在功能表中開啟 "HTML Help"，並輸入欲搜尋的字串，來了解詳細的說明。

SerializeOut() 與 SerializeIn() 若回傳 0，表示儲存位置錯誤或 Array 空間不足。

(* 宣告 TMP_DINT 為一個 DINT,
buf 為一個 BYTE Array, Dim 為 10,
DINT_Val 為一個 DINT,
Word_Val 為一個 WORD,
REAL_Val 為一個 REAL *)

註:

資料型態	Byte
BOOL, SINT, USINT, BYTE	1
INT, UINT, WORD	2
DINT, UDINT, DWORD, REAL	4
LINT, LREAL	8

範例 1

(* 將 DINT_Val 拆解成 4 個 byte，從 BYTE Array 位置 2 開始分別存放在 buf[2], buf[3], buf[4], buf[5]，
採用 "Little Endian" 排序 *)

```
TMP_DINT := SerializeOut (buf, DINT_Val, 2, FALSE) ;
```

註: 最後一個參數為 "FALSE" 表示採用 "Little Endian" 排序 (即，將低序位元組存儲在起始地址)

範例 2

(* 將 Word_Val 拆解成 2 個 byte，從 BYTE Array 位置 0 開始分別存放在 buf[0], buf[1]，
採用 "Big Endian" 排序 *)

```
TMP_DINT := SerializeOut (buf, Word_Val, 0, TRUE) ;
```

註: 最後一個參數為 "TRUE " 表示採用 "Big Endian" 排序 (即，將高序位元組存儲在起始地址)

範例 3

(* 將 BYTE Array 內的 buf[0], buf[1] , buf[2] , buf[3] 組合成 1 個 REAL_Val，採用 "Little Endian" 排序 *)

```
TMP_DINT := SerializeIn (buf, REAL_Val, 0, FALSE) ;
```

註: 最後一個參數為 "FALSE" 表示採用 "Little Endian" 排序 (即，將低序位元組存儲在起始地址)

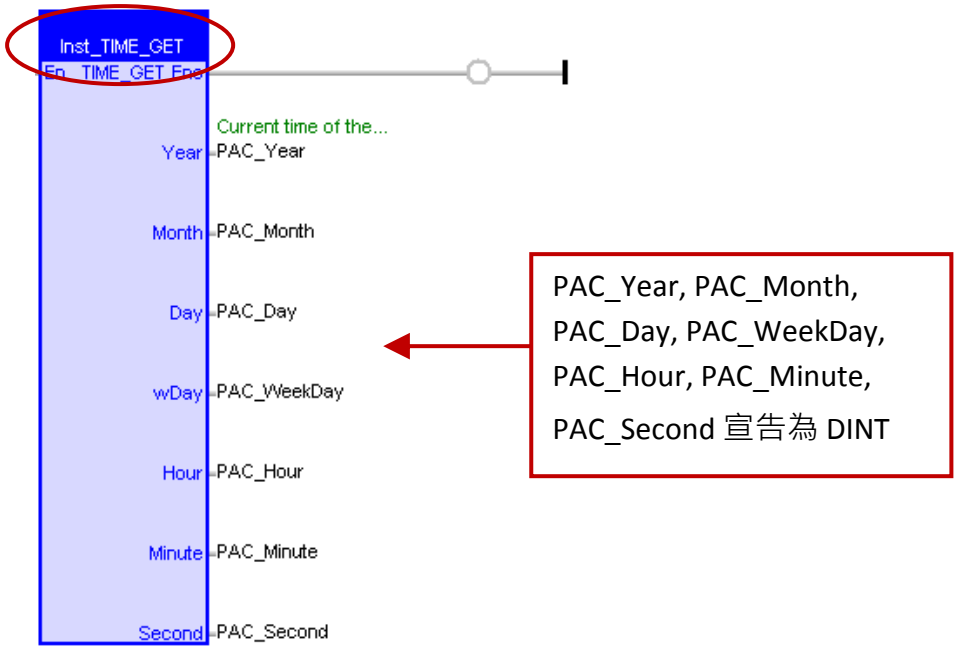
範例 4

(* 將 一個 DINT_Val 對應成 1 個 REAL_Val，採用 "Little Endian" 排序 *)

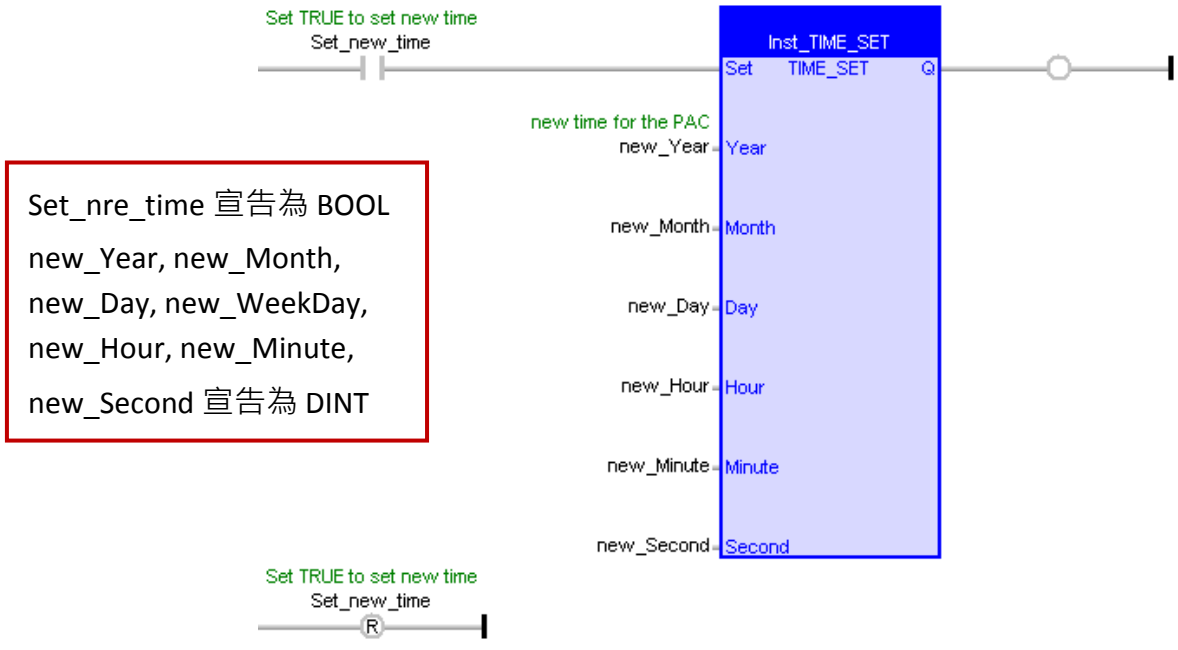
```
TMP_DINT := SerializeOut (buf, DINT_Val, 0, FALSE) ;
TMP_DINT := SerializeIn (buf, REAL_Val, 0, FALSE) ;
```

10.8 取得/設定 PAC 時間

若要取得 Win-GRAF PAC 目前的時間，可使用 "TIME_GET" 功能方塊。(可參考 2.2.1 節)



若要調整 Win-GRAF PAC 目前的時間，可使用 "TIME_SET" 功能方塊。(可參考 2.3.6 節)
先填入新的時間到 new_Year, new_Month, new_Day, new_WeekDay, new_Hour, new_Minute, new_Second 變數後，再將 Set_new_time 變數設定為 "TRUE" 一次。



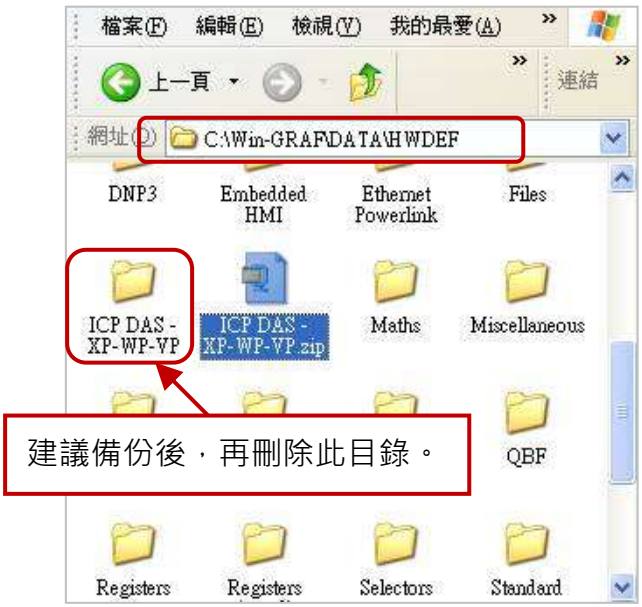
第 11 章 一般常用工具 與 有用的技巧

11.1 更新 Win-GRAF 程式庫 (Library)

註: 以後會提供程式庫的網頁連結。

Win-GRAF 的程式庫 (包括 Function、Function Block 與 I/O Board)，一般儲存在 C:\Win-GRAF\DATA\HWDEF 路徑下的 “ICP DAS - XP-WP-VP” 目錄中。基於某一些情況，會將程式庫更新至新版本以支援更多的 Function 或 新的 I/O Board，請依照以下步驟來執行此程序:

1. 首先，關閉所有 Win-GRAF Workbench 的視窗。
2. 可先壓縮 “ICP DAS - XP-WP-VP” 目錄並備份在其它路徑 (例如: D:\temp\xxx.zip)，再刪除該目錄。



3. 將新的 “ICP DAS - XP-WP-VP” 目錄複製到 C:\Win-GRAF\DATA\HWDEF 路徑內，再重新開啟 Win-GRAF Workbench。

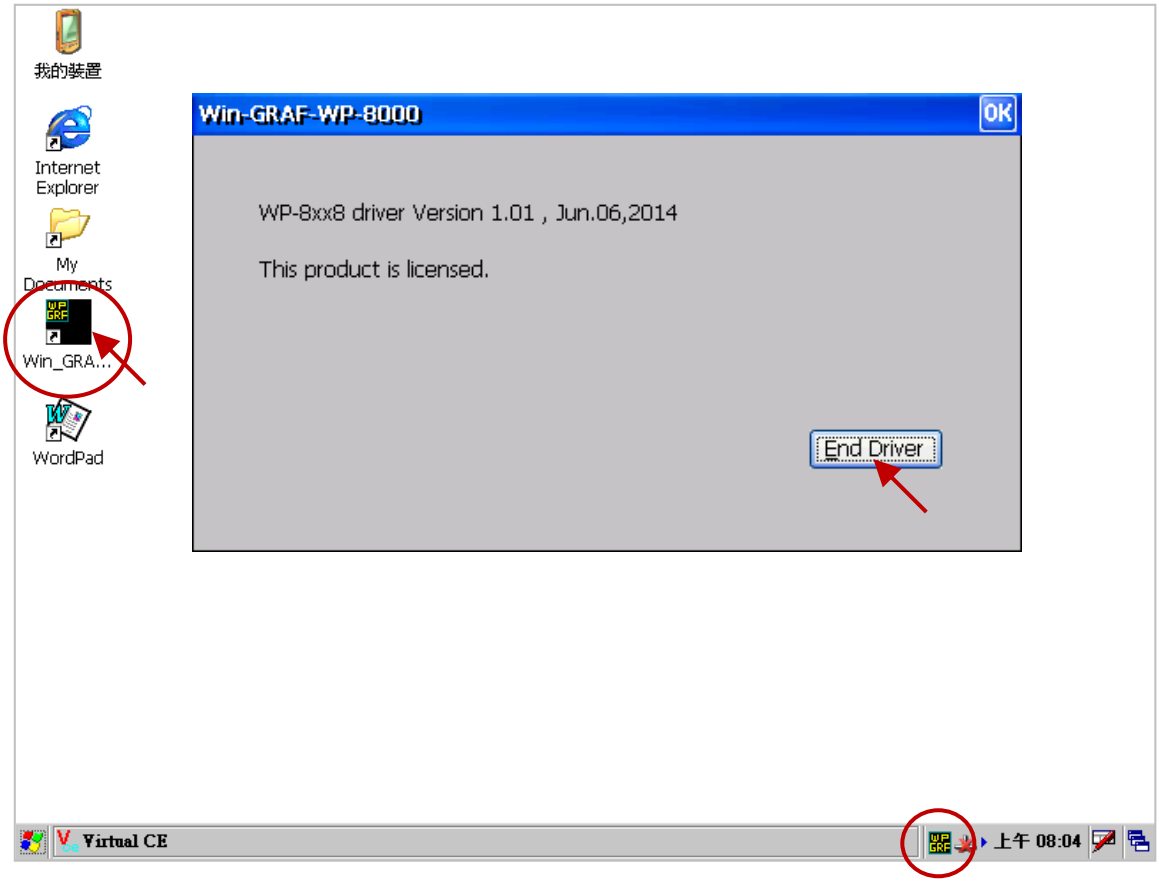
11.2 更新 Win-GRAF 驅動程式 (Driver)

註: 以後會提供程式庫的網頁連結。

為了因應新增加的功能、IO Board 或 其他用途，將會不定時的發佈新版的 Win-GRAF 驅動程式 (Driver)，您可在網頁中取得最新的版本並依照以下步驟更新到 PAC 中。

XPAC (XP-8xx8, XP-8xx8-CE6, XP-8xx8-Atom-CE6)、WinPAC (WP-8xx8, WP-5xx8) 與 ViewPAC (VP-25W8, VP-4138) 會把 Win-GRAF Driver 放在 \System_Disk\Win-GRAF\ 路徑內。

1. PAC 中 (以 WP-8xx8 為例)，滑鼠雙擊桌面上的 "Win-GRAF_WP_8000" 再點選 "End Driver" 停止正在運行的 Driver。



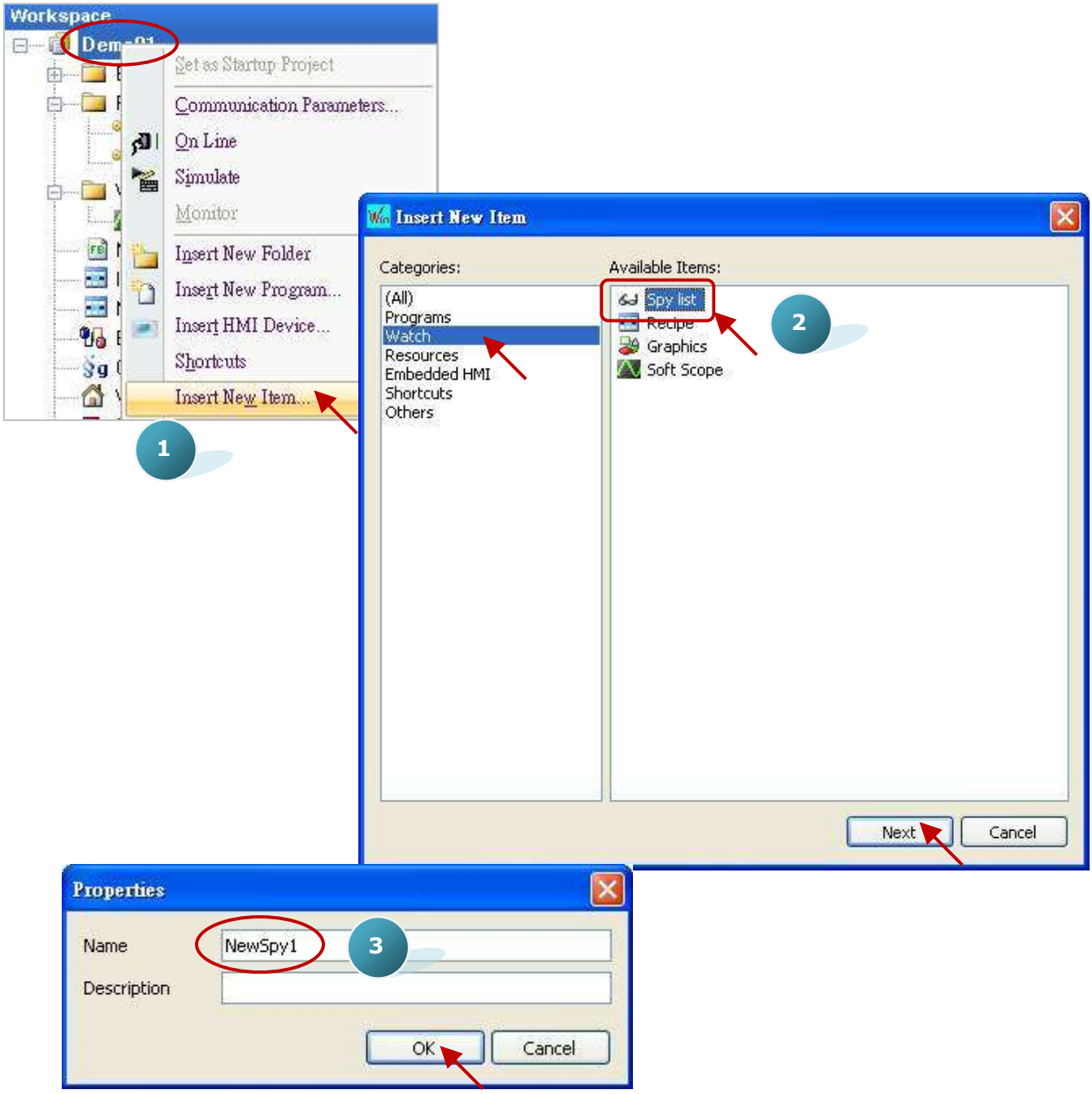
2. PC 中，以 FTP 方式先將新的 Driver 丟到 PAC 的 \Temp\ 路徑內。
3. PAC 中，將 \Temp\ 路徑內的新 Driver 覆寫到 \System_Disk\Win-GRAF\ 路徑內，再將 PAC 重新開機即可。

11.3 觀測清單 (Spy List)

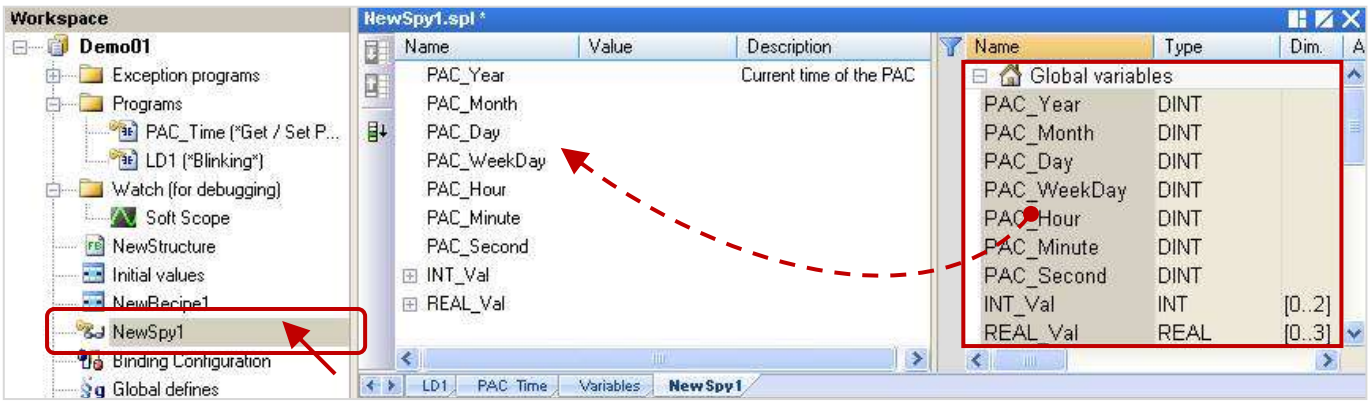
在程式運行時，觀測清單 (Spy List) 可提供使用者快速掌握變數的數值或狀態。有時，一個程式可能宣告了數百 ~ 數千個變數，使用者無需費心尋找，只需切換到自己建立的 Spy List 即可看到所需的資料。

方法如下:

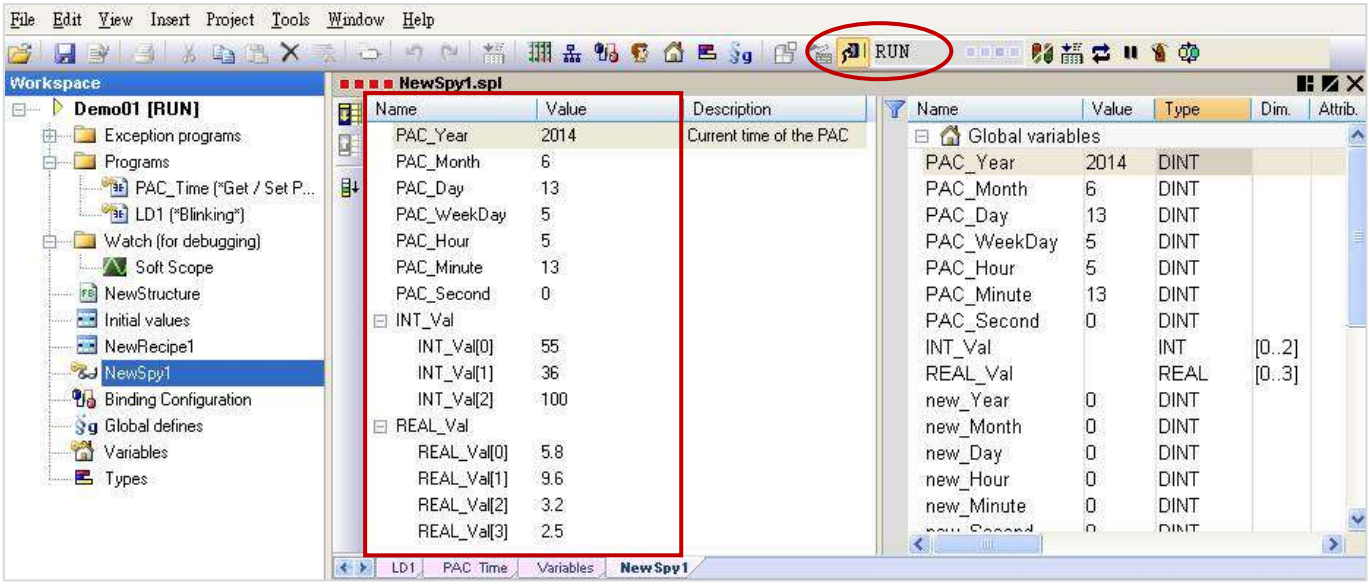
1. 滑鼠右鍵點選專案名稱 (例如: "Demo01") 並選擇 "Insert New Item" 項目。
2. 在 "Watch" 區中選擇 "Spy List"，再點選 "Next" 進行下一步驟。
3. 接著，再輸入清單名稱 (例如: "NewSpy1") 並按 "OK"。



4. 滑鼠雙擊左側的 "NewSpy1" 開啟視窗，並將想要顯示的變數拖曳到此視窗內。



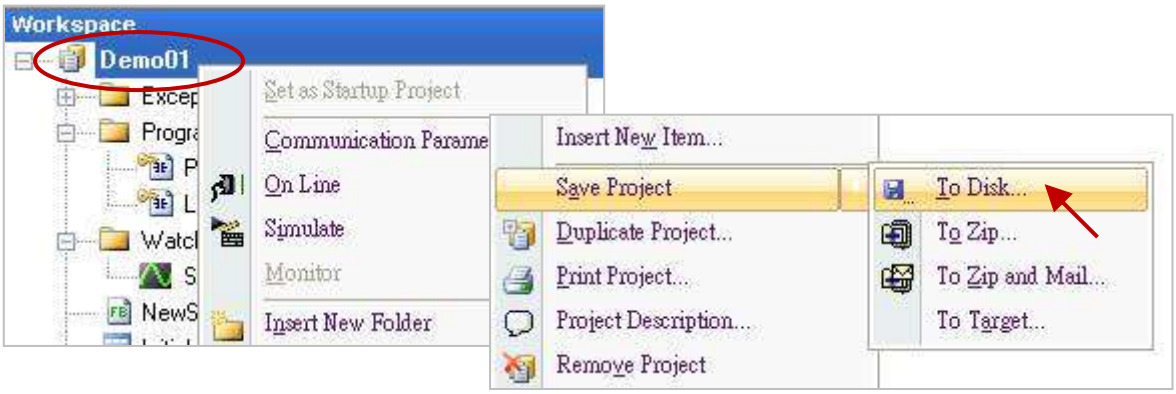
5. 當 Win-GRAF 與 PAC 連線時，此 "NewSpy1" 視窗將會清楚顯示這些變數資料。



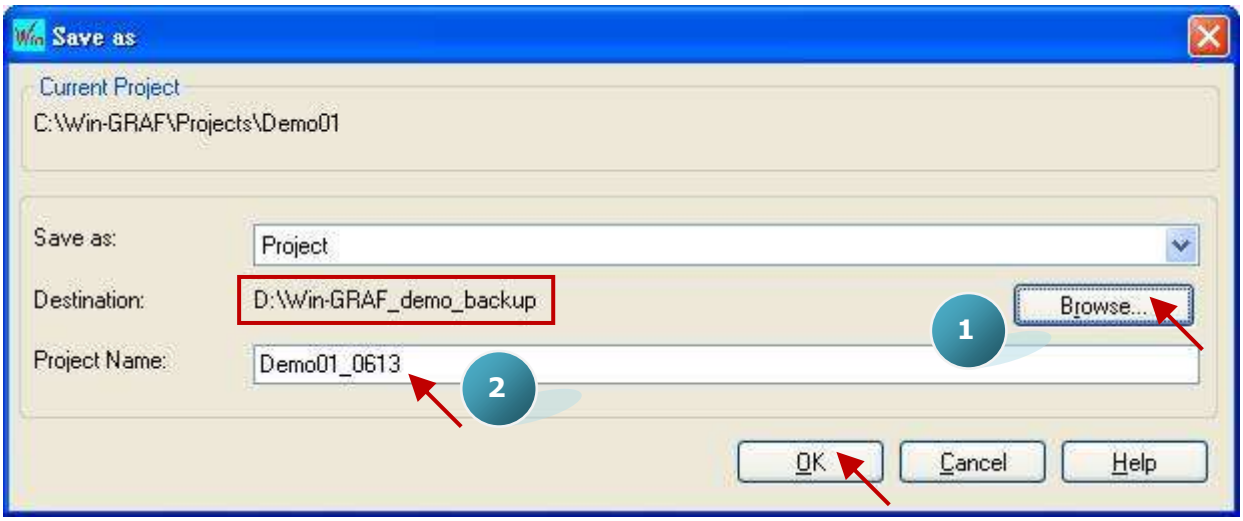
11.4 備份/回存一個 Win-GRAF 專案 (Project)

備份 Win-GRAF 專案:

1. 滑鼠右鍵點選專案名稱 (例如: "Demo01") 並選擇 "Save Project" 內的 "To Disk"。

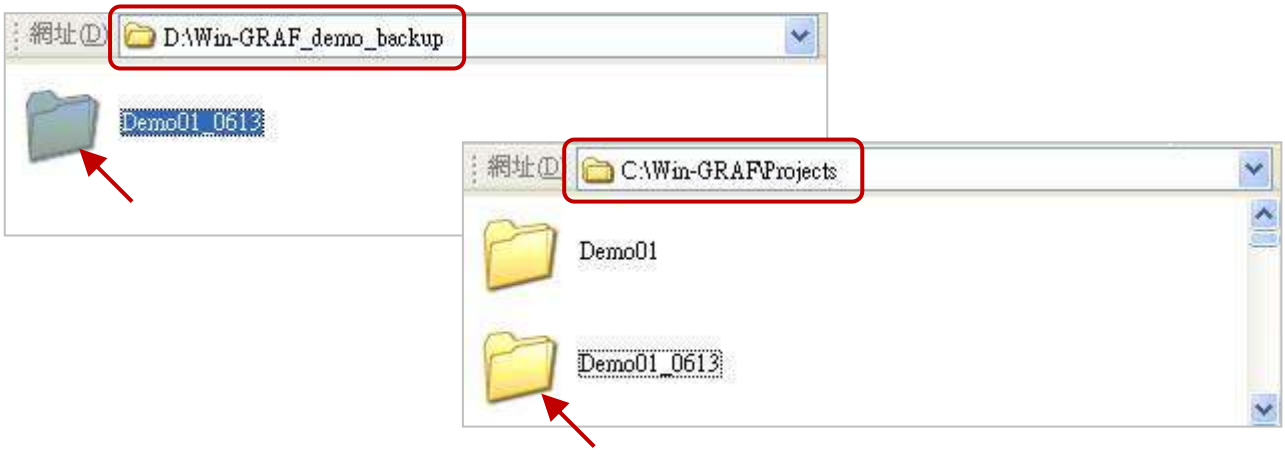


2. 點選 "Browser" 按鈕來指定存檔的路徑 (例如: D:\Win-GRAF_demo_backup)，並輸入專案名稱 (例如: "Demo01_0613")，再點選 "OK" 即完成備份。

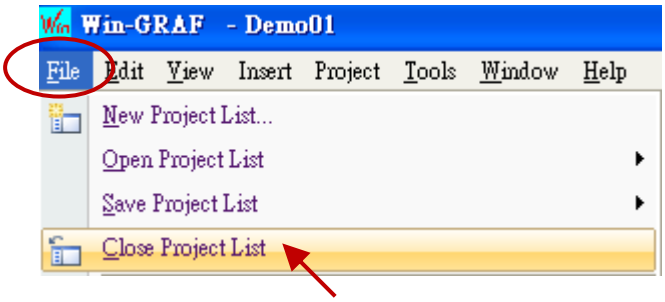


回存 Win-GRAF 專案:

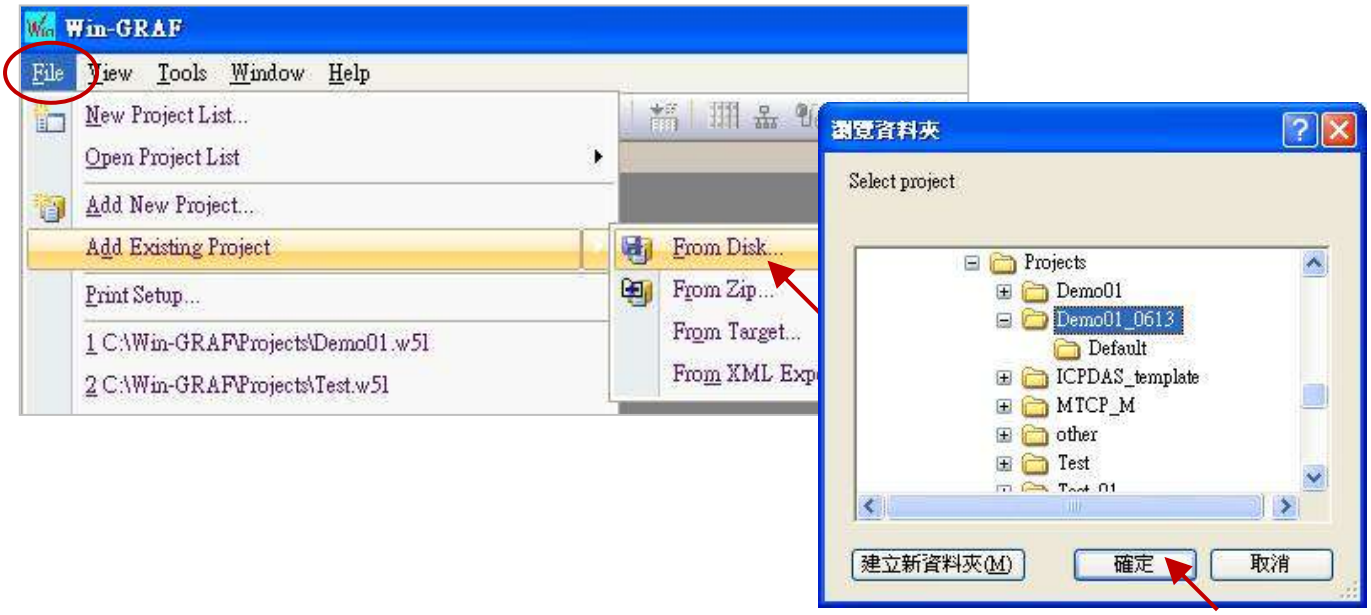
1. 請將先前備份的專案目錄 (例如: "Demo01_0613") 複製到 C:\Win-GRAF\Projects 路徑內。

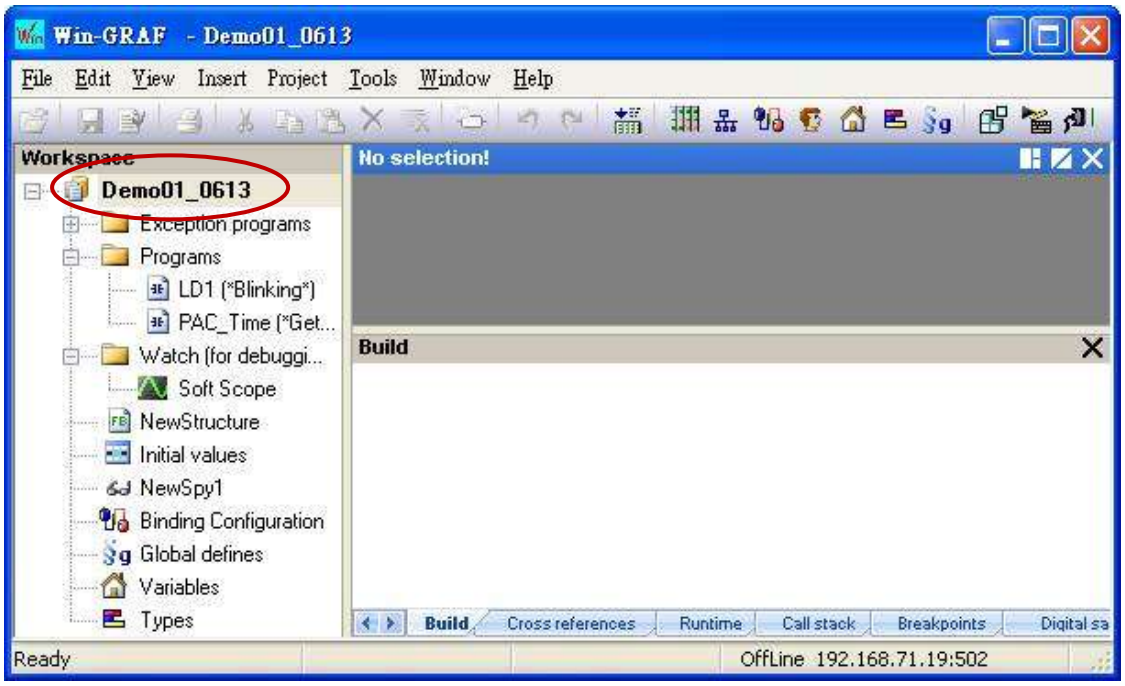


2. 滑鼠點選功能表 "File" > "Close Project List" 關閉所有已開啟的視窗。



3. 接著，再點選功能表 "File" > "Add Existing Project" > "From Disk" 項目，並選取 C:\Win-GRAF\Projects 路徑內，要回存的專案目錄 (例如: "Demo01_0613")，再點選 "確定" 即完成回存。





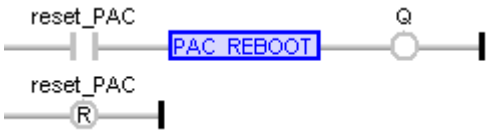
11.5 以軟體重新啟動 PAC

基於某一些情況，使用者可能需要以軟體的方式來重新啟動 PAC。Win-GRAF 提供了 "PAC_Reboot" 函式 (Function) 可調用它來重開機。

註: 請勿在每個 PAC Cycle 調用此功能，否則 PAC 會一直重新開機。

安全寫法:

```
(*  "reset_PAC" 宣告為 BOOL 且初值為 "FALSE"
    "TMP_BOOL" 宣告為 BOOL *)
(* 只有 "reset_PAC" 設定為 "TRUE" 時，才會重開機  *)
   if  reset_PAC  then
     reset_PAC := FALSE ;
     TMP_BOOL  :=  PAC_Reboot( ) ;
   end_if ;
```



危險的寫法:

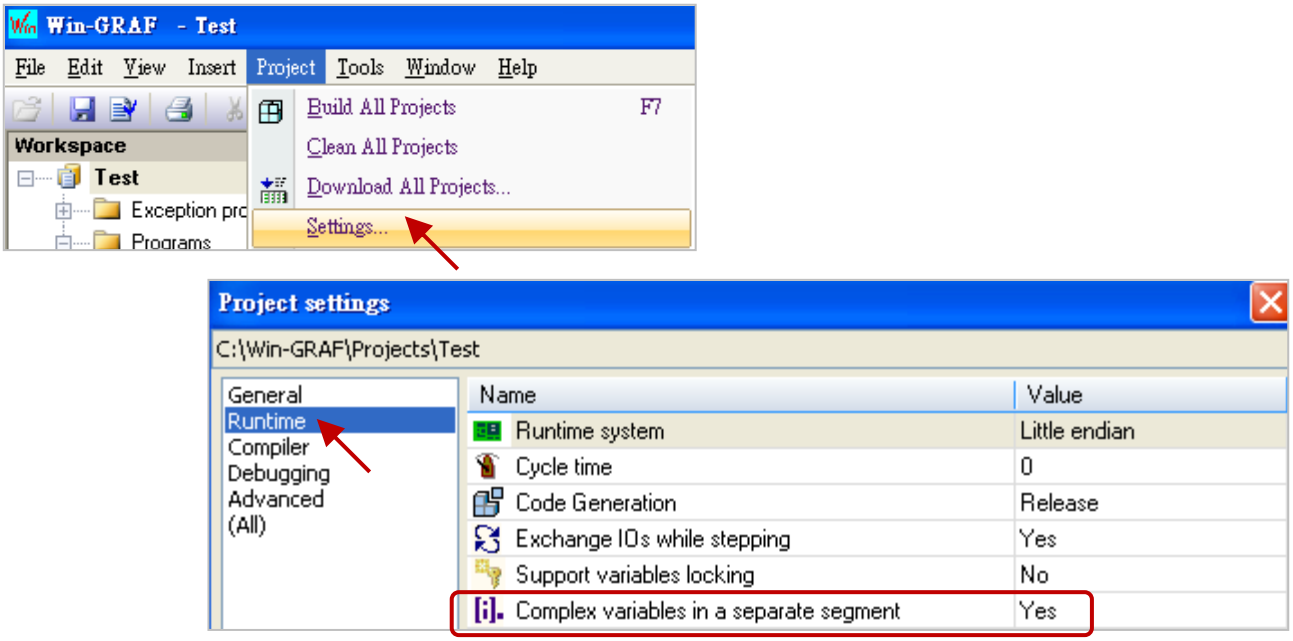
```
(*  "TMP_BOOL" 宣告為 BOOL *)
(* 危險！此寫法會導致 PAC 一直重新開機。 *)
    TMP_BOOL  :=  PAC_Reboot( ) ;
```

若出現 PAC 一直重開機的錯誤，請將 Win-GRAF PAC 上的旋轉式開關 (Rotary Switch) 轉到 "1" 並重新開機。開機後會進入安全模式，請修改 PAC 中 Win-GRAF 應用程式為無效的名稱 (例如: "t5.cod1")，再將旋轉式開關轉到 "0"， 重新開機後會進入一般模式 (且無應用程式)。

註: Win-GRAF PAC (即，XP-8xx8, XP-8xx8-CE6, WP-8xx8, WP-5xx8, VP-25W8 and VP-4138) 裡的應用程式存放在 "\System_Disk\Win-GRAF\t5.cod".

11.6 在 LD 與 FBD 內使用 ST 語法

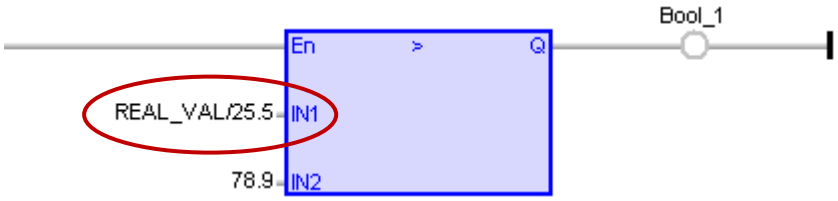
Win-GRAF Workbench 允許使用者在階梯圖 (LD) 與 功能方塊圖 (FBD) 內使用簡單的 ST 語法，以方便程式設計。使用前，請先至功能表 "Project" > "Settings" 的 "Runtime" 區內將 "Complex variables in a separate segment" 啟用為 "Yes"。



以下為使用的例子:

LD 語法:

使用除法 (REAL_VAL/25.5)。



FBD 語法:

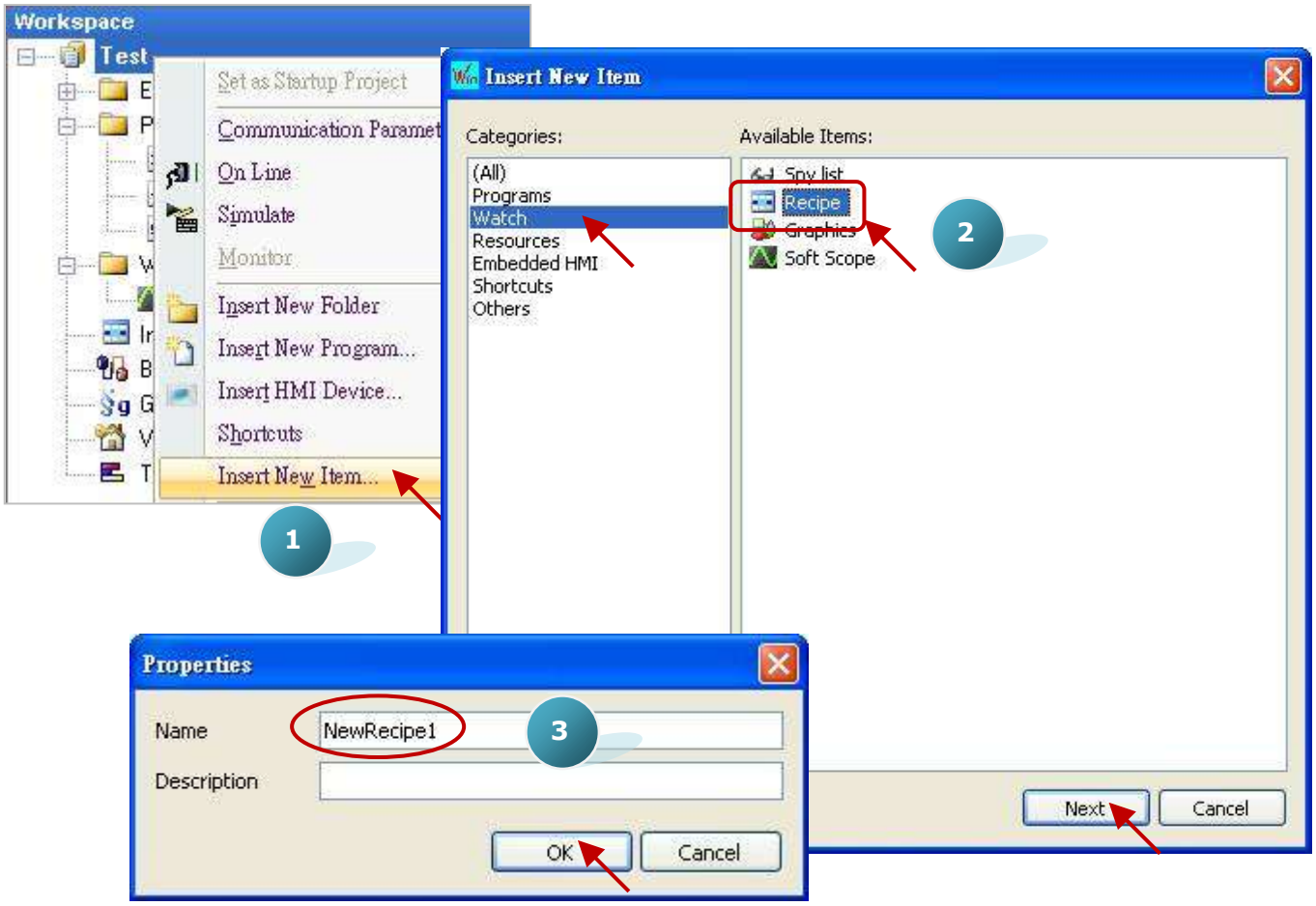
呼叫一個 ANY_TO_BYTE() 函式，將 SINT 轉為 BYTE 型態。

11.7 置換 PAC 內的配方表 (Recipe)

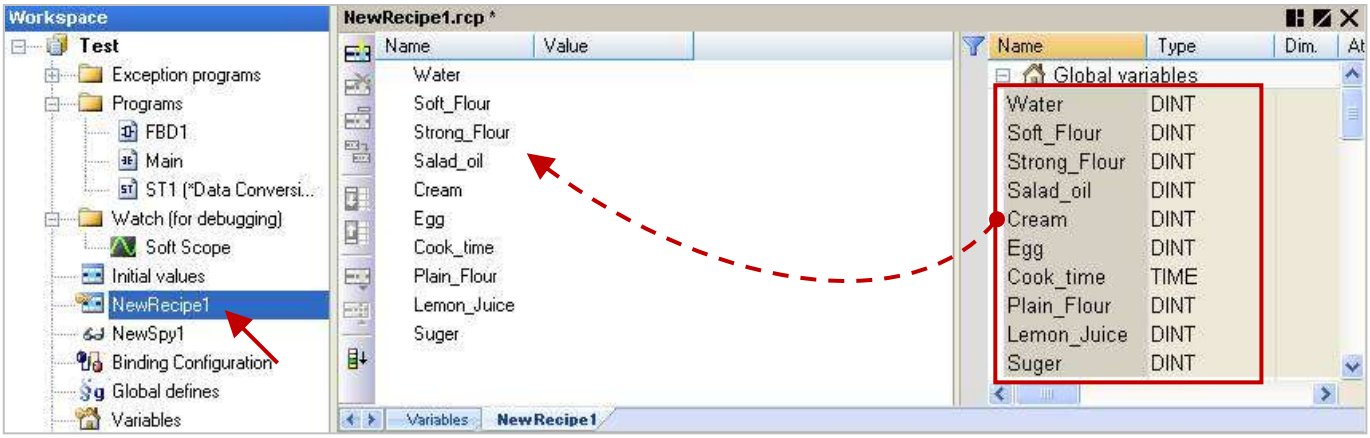
有些應用會預先定義好不同產品所需的配方表 (Recipe) 與數值，而此配方表 (Recipe) 可對應到 Win-GRAF PAC 內一群變數的組合，當某天要變更 PAC 製程來生產不同的產品時，可使用 Win-GRAF Workbench 與 PAC 連線，並選取想要更換的新配方表，再套用到 PAC 內即可。

方法如下:

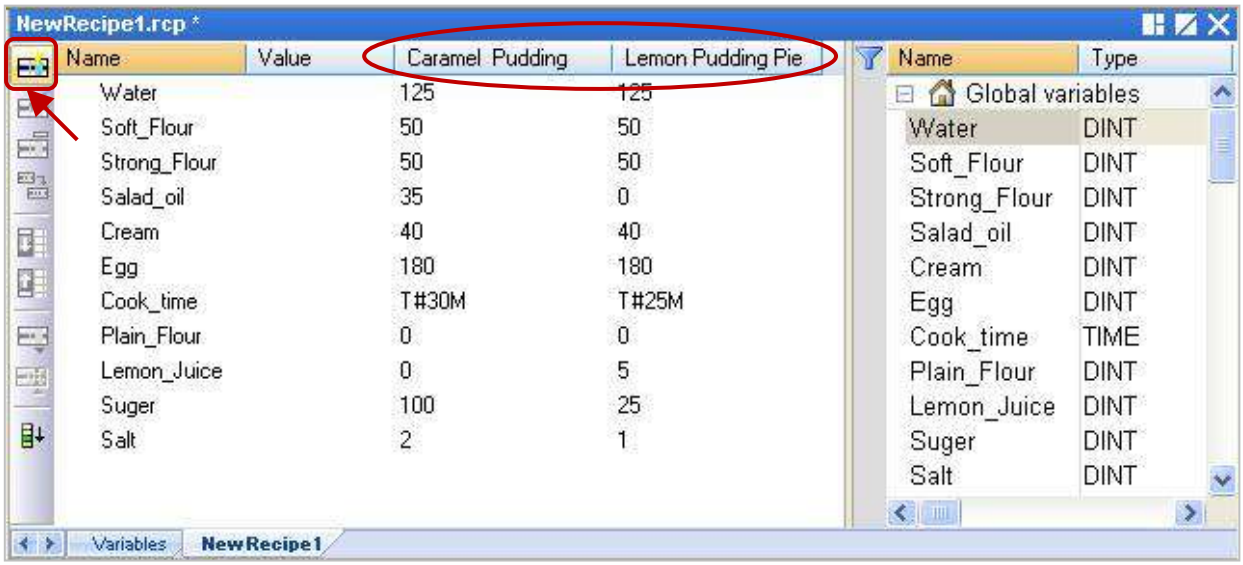
1. 滑鼠右鍵點選 專案名稱 (例如: "Test") 並選擇 "Insert New Item" 項目，在 "Watch" 區中選擇 "Recipe" 並點選 "Next"，接著，再輸入清單名稱 (例如: "NewRecipe1") 並按 "OK"。



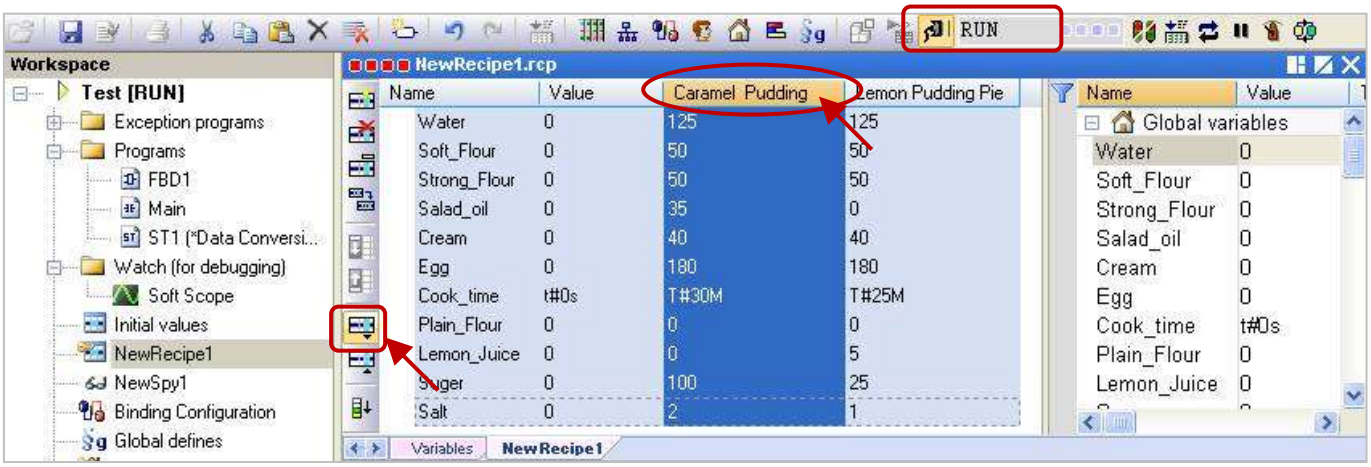
2. 滑鼠雙擊左側的 "NewRecipe1" 開啟視窗，並將需使用的變數拖曳到此視窗內。

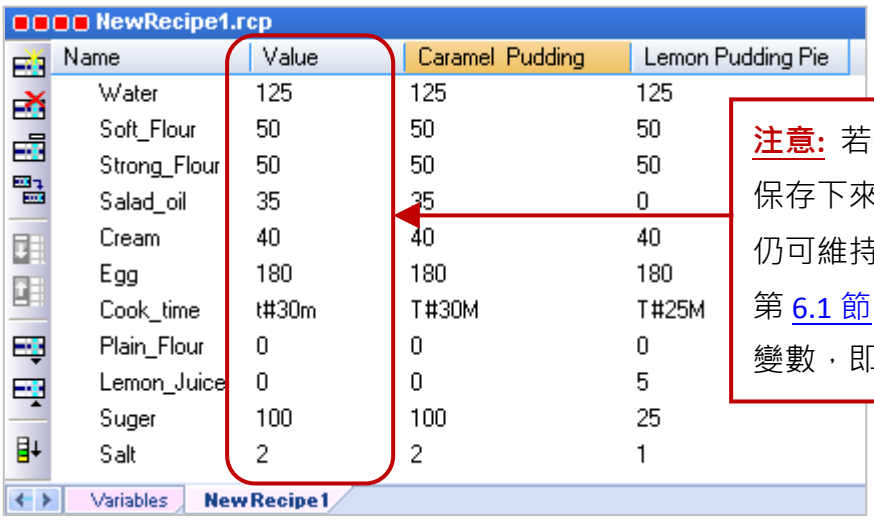


3. 點選 "Insert Column" 來新增配方表，並填入適當的數值。



4. 點選 "On Line" 按鈕與 PAC 連線。一開始值皆為 "0"，請點選產品名稱欄位再點選 "Send recipe" 將此配方表套用到 PAC 中。

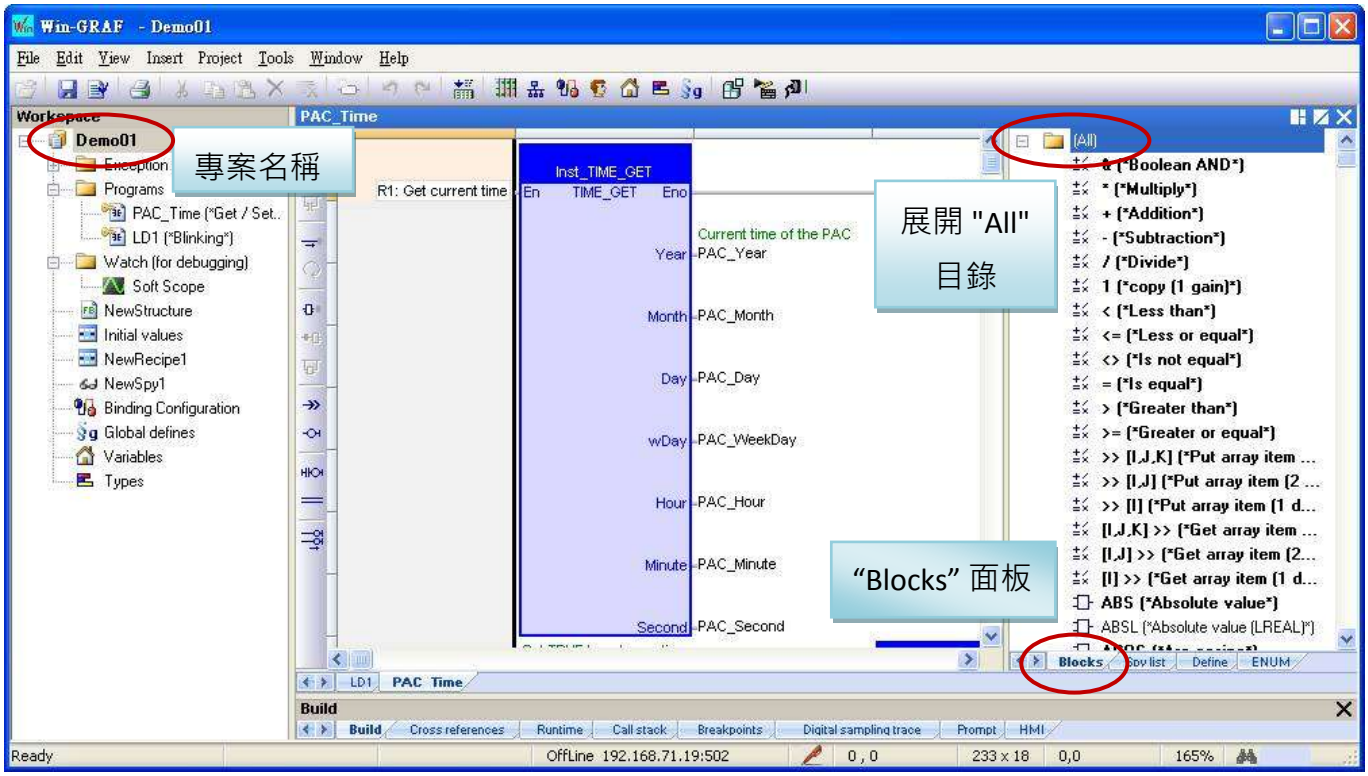




注意: 若想將 PAC 內的配方數值保存下來 (即 PAC 關機後再開機，仍可維持之前的數值)，請參考第 6.1 節 的方法來使用可保存變數，即可達成。

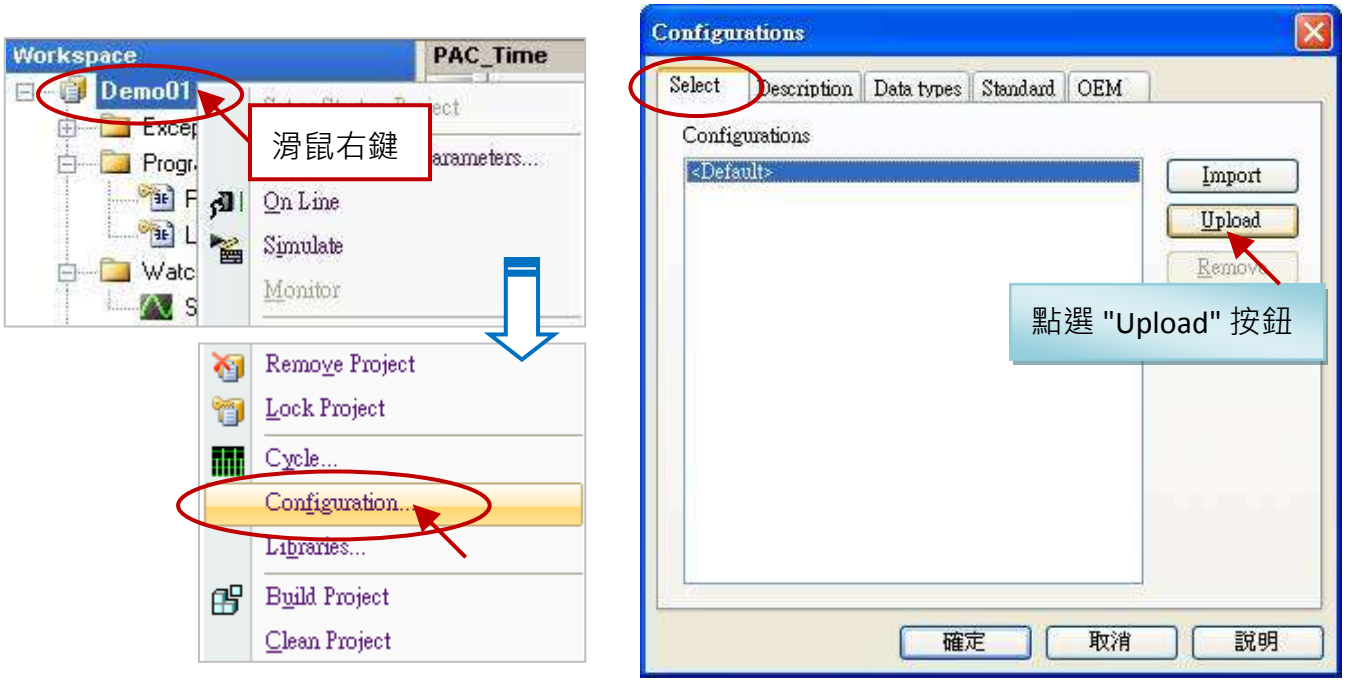
11.8 取得 PAC 所支援的函式 (Function) 與功能方塊 (Function Block)

在 Win-GRAF Workbench 的 "Blocks" 面板內，展開 "All" 目錄可見到相當多的函式 (Function) 與 功能方塊 (Function Block)，然而有一些在 Win-GRAF PAC 內並未支援，以下將說明如何快速分辨 PAC 內所支援的函式 與 功能方塊。

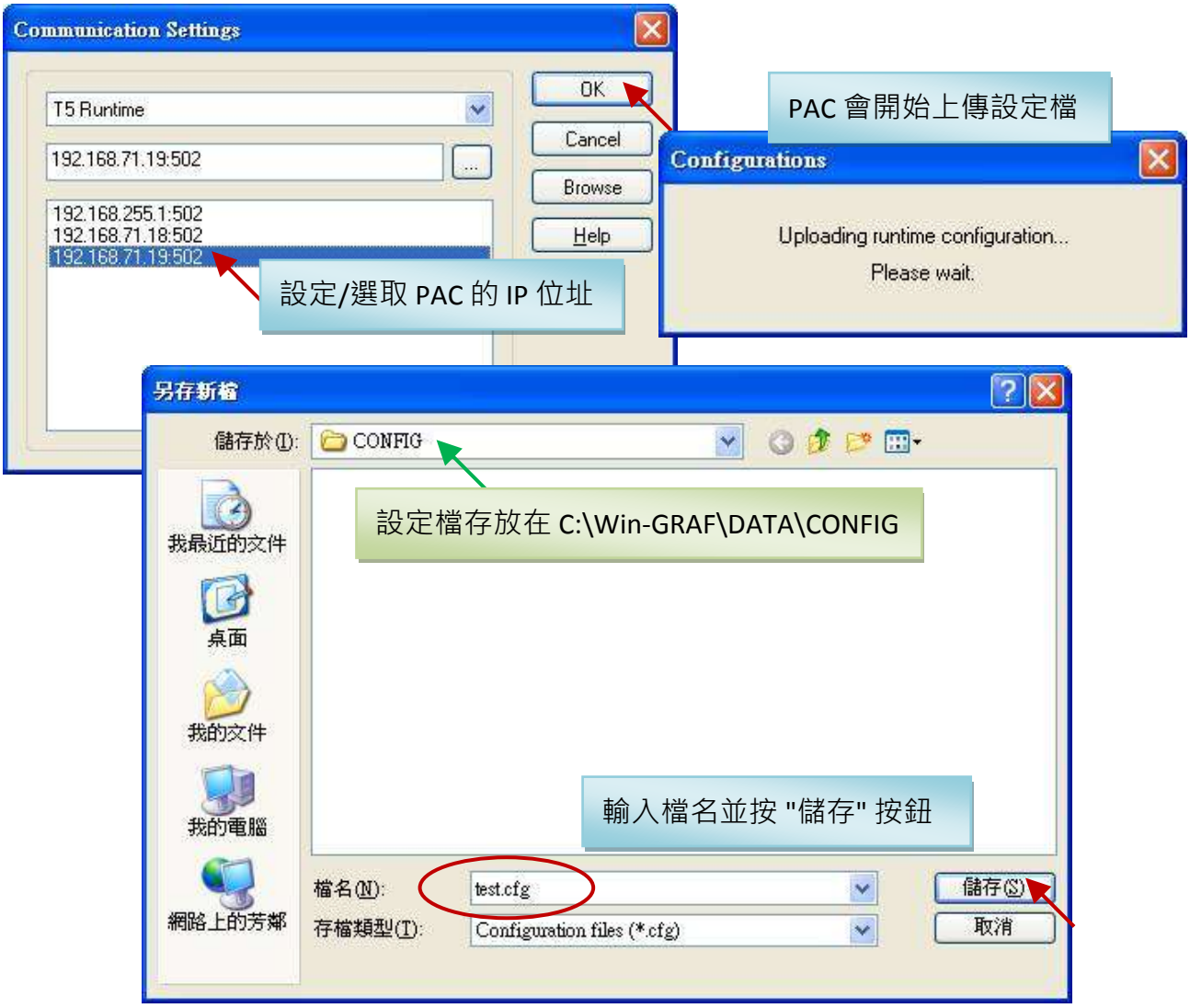


設定方式如下:

1. 請確認 PAC 有開機且和 PC 有用 Ethernet 連上。
2. 於 Win-GRAF Workbench，滑鼠右鍵點選專案名稱 (例如: "Demo01") 再選擇 "Configuration"，並於 "Select" 頁籤中點選 "Upload" 按鈕開啟設定視窗。

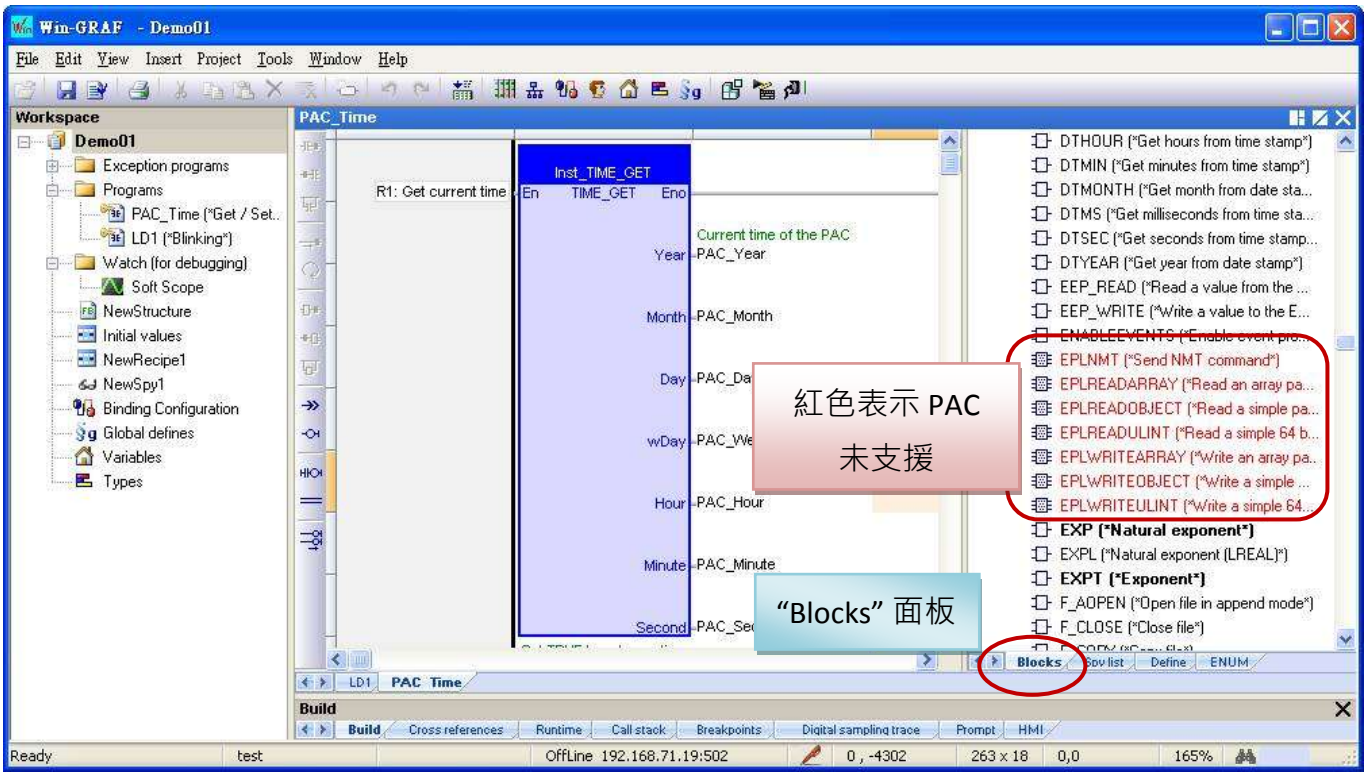


3. 設定/選取 PAC 的 IP 位址並點選 "OK" 按鈕後，PAC 會開始上傳設定檔。接著，請輸入檔名 (例如: "test.cfg") 並按 "儲存" 來儲存此設定檔。



4. 回到 "Configurations" 視窗，可見到剛剛設定的檔名 (test)，請按 "確定" 離開此視窗。

5. 在 "Blocks" 面板內，可見到紅色部分表示 PAC 未支援。

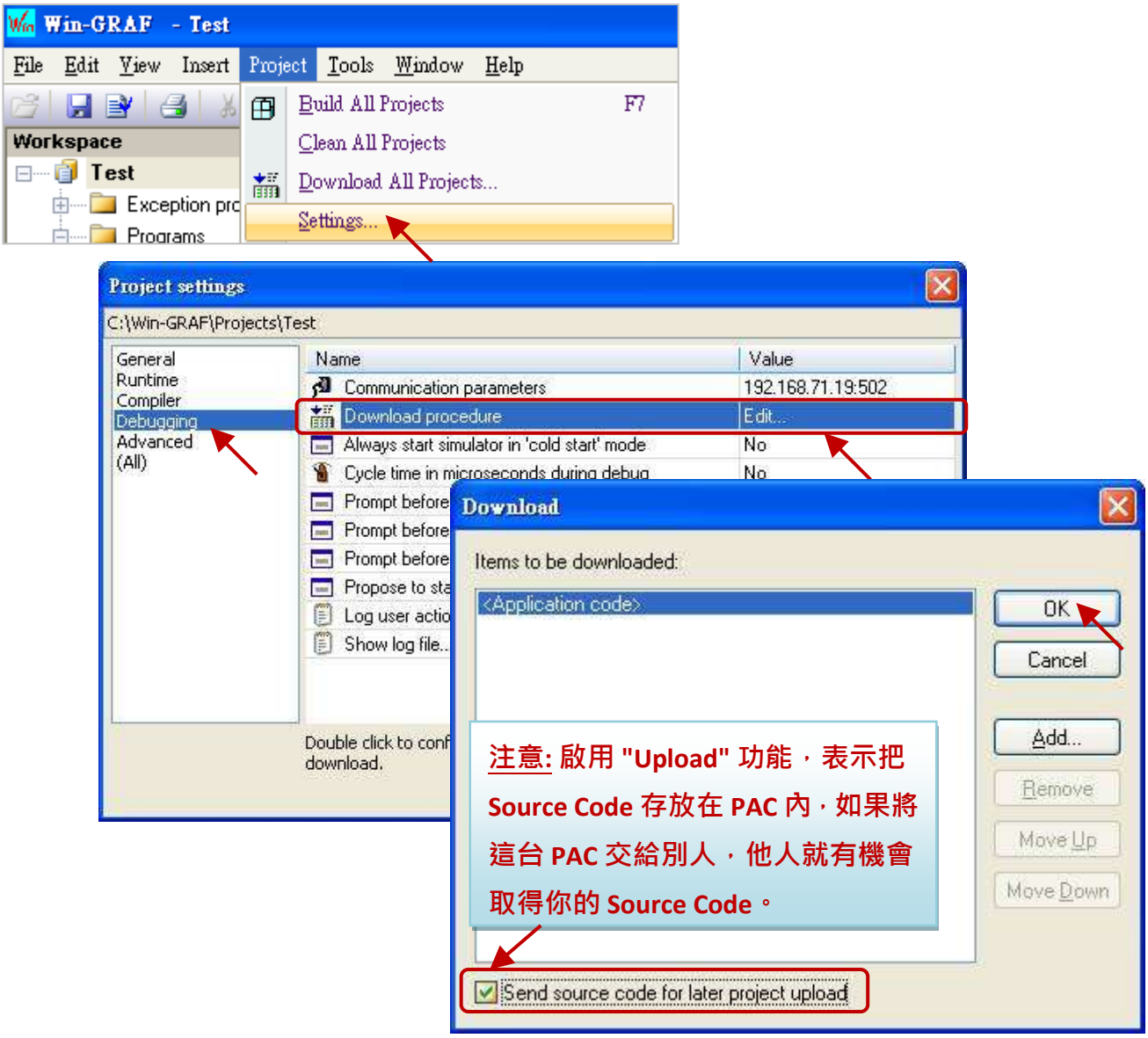


11.9 上傳 Win-GRAF 專案原始碼

有些應用可能會在一段時日後，需要從 PAC 內將 Win-GRAF 專案的原始碼 (Source Code) 抓回到 PC 內，這個功能就叫做 "Upload"。此功能可防止專案的原始碼遺失 或 前一位工作者移交原始碼不完全，仍可取得 PAC 內的專案原始碼。

啟用/下載專案原始碼:

1. 滑鼠點選功能表 "Project" > "Settings" 開啟設定視窗。
2. 滑鼠雙擊 "Debugging" 內的 "Download procedure" 項目並勾選 "Send source code for later project upload" 再點選 "OK " 完成設定。



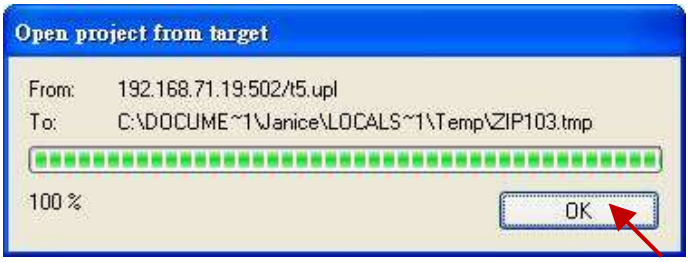
3. 點選功能表 "Project" > "Build All Projects" 編譯程式，並點選工具按鈕  與 PAC 建立連線，接著再將目前的專案下載到 PAC 中 (若不熟悉操作，可參考 2.3.5 節)。下載專案後，Source Code 會存放在 PAC 的 \Systen_Disk\Win-GRAF\t5.upl，此檔案會隨著專案內容的增加而變大，當專案很大很複雜時，檔案大小有可能會達到幾百 K Byte 或甚至超過 1 MB。

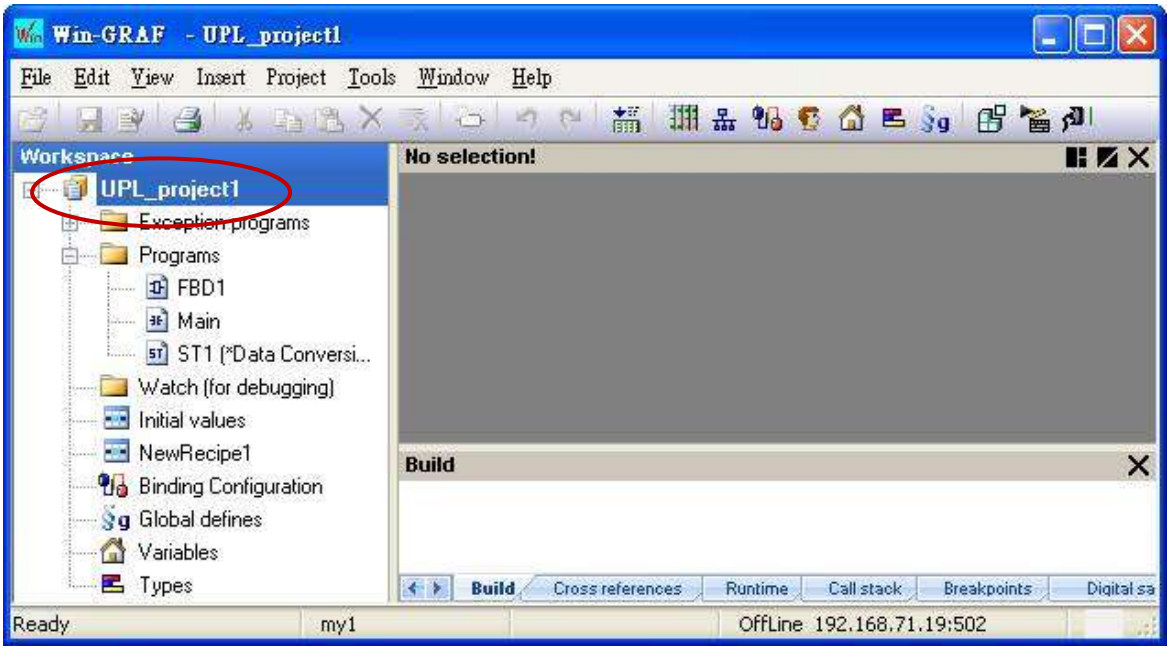
上傳專案原始碼:

請先關閉所有已開啟的 Win-GRAF 視窗 (滑鼠點選功能表 "File" > "Close Project List")。

4. 點選功能表 "File" > "Add Existing Project" > "From Target"，接著 選取 PAC 的 IP 位址 並設定上傳檔案的名稱 (例如: "UPL_project1")，再點選 "OK" 開始上傳檔案。

5. 上傳完成後，請點選 "OK" 按鈕。此時，Win-GRAF 會自動開啟此專案。

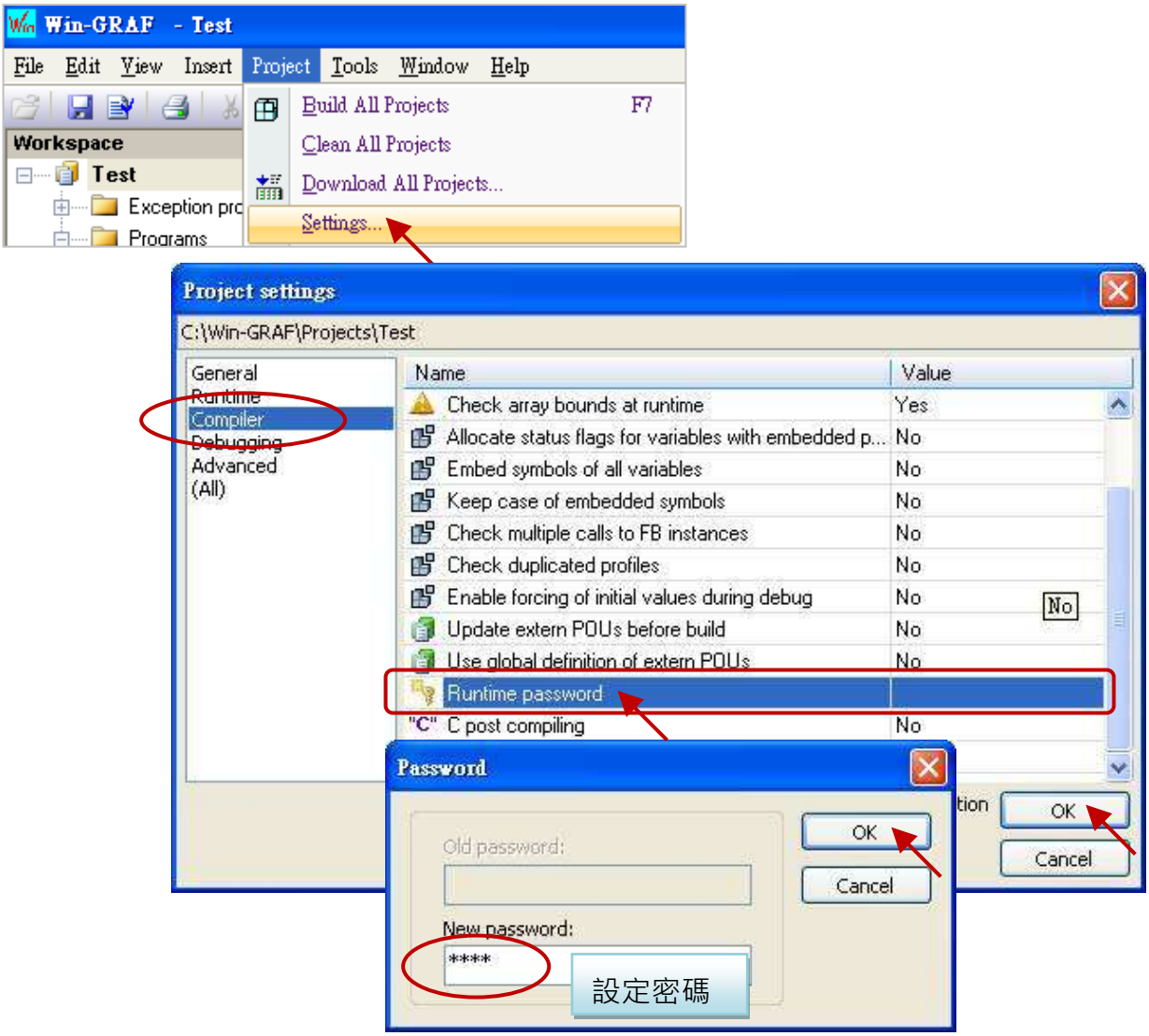




11.10 設定 PAC 的密碼

為了避免 PAC 內正在運行的重要程式，被某一台 PC 連線並任意的更動或停止運行，您可為 PAC 設定一組密碼來防止未經授權的操作。

1. 滑鼠點選功能表 "Project" > "Settings" 開啟設定視窗。
2. 滑鼠雙擊 "Compiler" 內的 "Runtime password" 項目，並設定密碼 再點選 "OK" 完成設定。

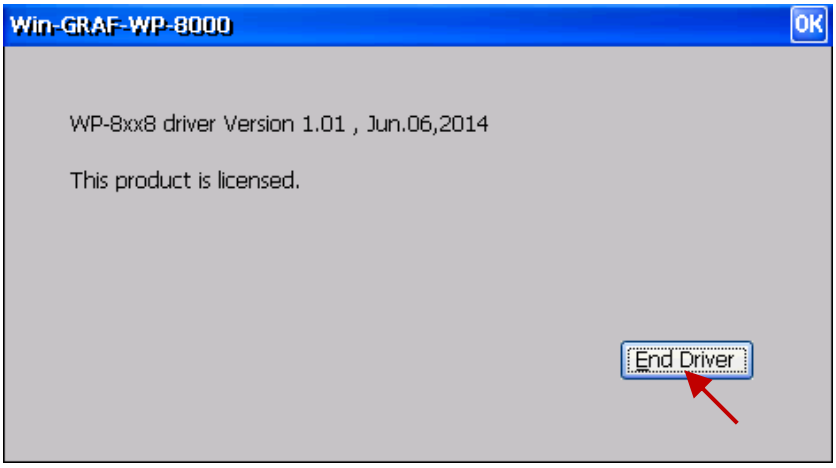


3. 點選功能表 "Project" > "Build All Projects" 再次編譯程式，並將目前的專案下載到 PAC 中即完成設定 (若不熟悉操作，可參考 2.3.5 節)。下次執行 "On Line" 連線時，將會要求輸入密碼。

注意: 啟用密碼後，請務必記得您所設置的密碼，否則日後您將無法連上 PAC。

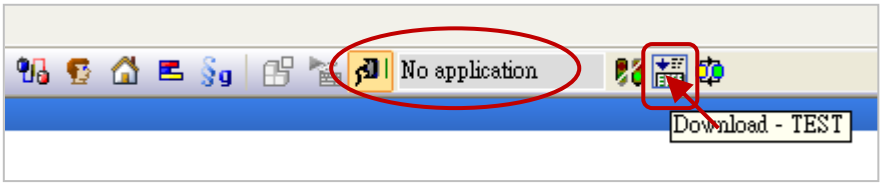
唯一的解決方法:

1. 將 PAC 接上 USB 滑鼠與螢幕。
2. PAC 中，開啟 Win-GRAF Driver 並執行 "End Driver" (可參考 11.2 節)。



3. 將 \System_Disk\win-graf 內的 "t5.cod" 檔案重新命名 (例如: "t5.cod1") 或刪除後，再將 PAC 重新開機。

如此，PAC 內將變成 "No application"，此時 Win-GRAF Workbench 可再重新下載應用程式。



11.11 使用 ST 語法來操作功能方塊

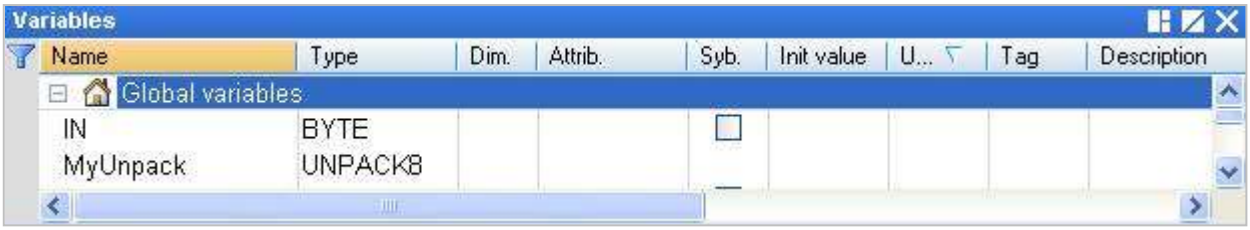
使用者如需在 ST 語法內操作函式 (Function) 方法很簡單，只要呼叫該函式 與 填入對應的參數即可。如下的程式在一開始會開啟 COM3，之後每隔 5 秒會從 COM3 送出一個字串 `Hello‘。

```
(* 宣告 “INIT1” 為 BOOL 並指定初值為 TRUE,
   宣告 “TMP_BOO” 為 BOOL, “TMR1” 為 TIME *)

IF   INIT1   THEN
   INIT1 := FALSE ;
     TMR1 := T#0s ;
   TSTART (TMR1) ;
END_IF;
IF COM_Status(3) = FALSE THEN
   TMP_BOO := COM_open (3, `19200,N,8,1’) ;
END_IF ;
IF   TMR1 >= T#5s   THEN
   TMR1 := T#0s ;
   COM_send_str (3, `Hello: ’) ;
END_IF;
```

若要在 ST 內使用功能方塊 (Function Block)，需先在變數區宣告要使用的功能方塊的樣例變數 (Instance)，之後的用法就類似函式的用法如下:

以下的程式碼可以把 1 個 byte 拆解成 8 個 BOOL:

1. 宣告 "MyUnpack" 變數為 UNPACK8 (FB Instance)，"IN" 變數為 BYTE。



2. 然後寫一個 ST 程式。

```
MyUnpack(IN) ;
Q0 := MyUnpack.Q0 ;
Q1 := MyUnpack.Q1 ;
Q2 := MyUnpack.Q2 ;
Q3 := MyUnpack.Q3 ;
Q4 := MyUnpack.Q4 ;
Q5 := MyUnpack.Q5 ;
Q6 := MyUnpack.Q6 ;
Q7 := MyUnpack.Q7 ;
```

11.12 如何保護您的 Win-GRAF 程式，讓盜用者無法使用?

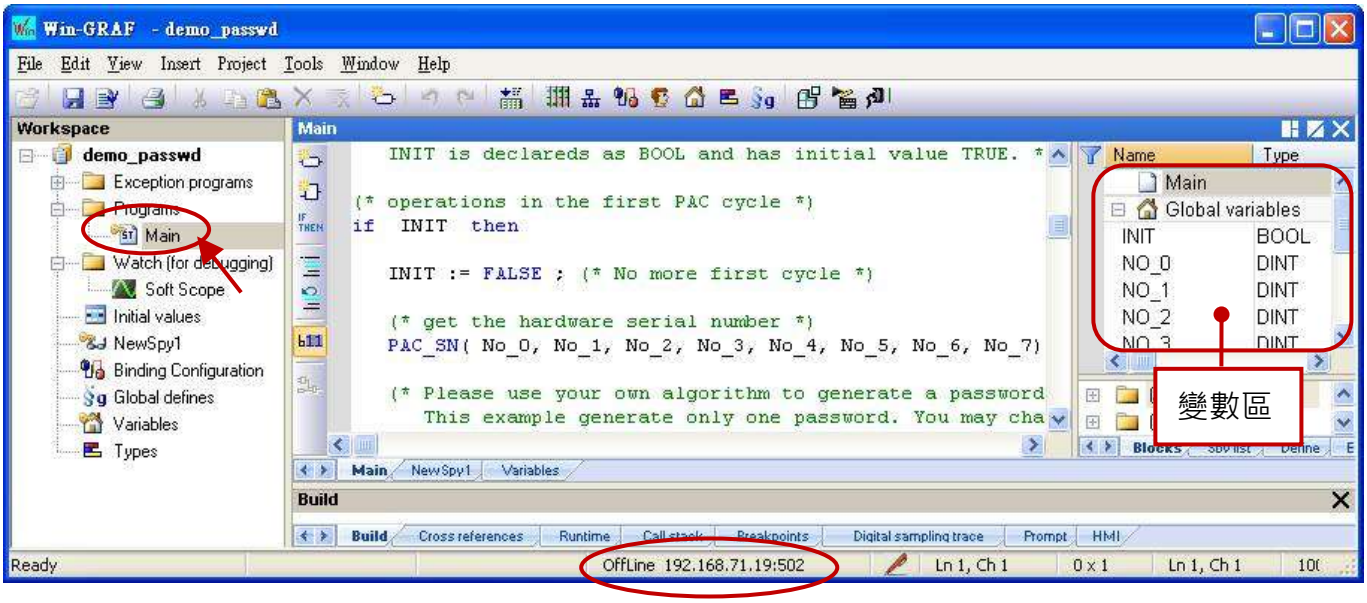
當您完成 Win-GRAF 應用程式開發，並準備交貨給客戶時，請先想一想，是否有可能您交出去的 Win-GRAF PAC 內的應用程式，將被複製到另一台同型號的 PAC 內？！小心！或許第三者就是這樣盜用了您辛苦開發完的成果，以下提供一個簡單好用的方法來保護您的應用程式。

注意: 如果您是把 Win-GRAF 應用程式的 Source Code 交給了客戶，那很抱歉，以下的方法將無法保護您的程式不被盜用。因為有了 Source Code，盜用者就可以自行修改程式碼並套用在另一台 PAC 內。

每一台 ICP DAS Win-GRAF PAC 都會有一個序號 (Serial Number)，此序號有 8 個 Byte (或稱 64 Bit)，而且每台 PAC 的序號都不同。因此，可利用這個序號再加上您自訂的運算來產生一組密碼，再將此密碼預先存到 PAC 的 EEPROM 內。之後，在您的應用程式裡去驗證它，若不通過，該應用程式將不允許運作。其方法如下:

此範例使用了 2 個 Win-GRAF 專案，一個是 "demo_passwd" 用來產生 PAC 的密碼並存到 EEPROM 內，另一個是 "demo_my_ap" 已開發好並準備出貨給客戶的應用程式。每次 PAC 要出貨之前，必需先把 "demo_passwd" 程式下載到 PAC 內 Run 一次，它會產生專屬於那台 PAC 的密碼。然後，使用者再下載 "demo_my_ap" 到同一台 PAC 內，之後再出貨給客戶。此後，第三者若複製此 PAC 內的 Win-GRAF 應用程式到另一台 同型號的 PAC 內，會因為密碼驗證不通過 而 運作失敗。

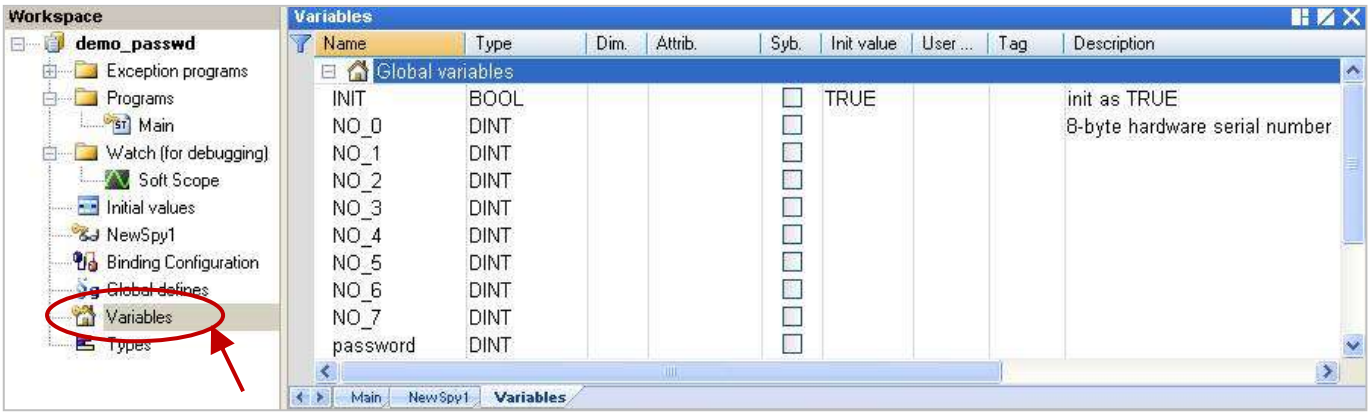
於出貨光碟中 (\Napdos\Win-GRAF\demo-project) 提供了此 2 個範例程式 (demo_passwd.zip 與 demo_my_ap.zip)，請參考 第 12 章 來回存此專案 (執行 File > Add Existing Project > From Zip) 並設定好 PAC 目前的 IP 位址。



"demo_passwd" 應用程式:

此程式先使用 "PAC_SN“ 來讀出序號，再經過使用者自訂的演算法來產生密碼。最後，將該密碼存到 EEPROM 記憶體的某個位址內 (使用者可自行決定要存放何處)。

變數宣告:

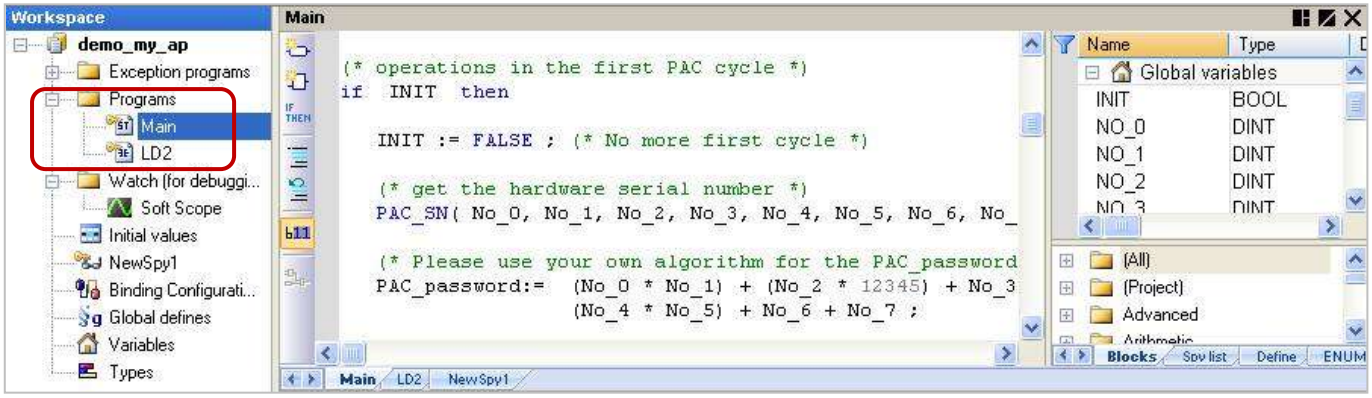


ST 程式:

```
(* 此 "demo_passwd" 範例程式會依據 PAC 內 8-byte 的硬體序號來產生一組密碼，
   並將它儲存到 PAC 的 EEPROM 內。

(* 宣告 "No_0" ~ "No_7" 與 "password" 變數為 DINT。
   宣告 "INIT" 變數為 BOOL 且初值 (Initial value) 為 TRUE。 *)

(* 第一個 PAC Cycle 的操作 *)
if INIT then
    INIT := FALSE ; (* 表示不再是第一個 Cycle *)

  (* 取得硬體的序號 *)
  PAC_SN( No_0, No_1, No_2, No_3, No_4, No_5, No_6, No_7) ;

 (* 請使用您自定的演算法來產生一組密碼。本範例僅產生一組密碼，您可修改程式來產生
    多組密碼 *)
  password := (No_0 * No_1) + (No_2 * 12345) + No_3 + (No_4 * No_5) + No_6 + No_7 ;

  (* 將密碼儲存到 EEPROM，位址是 "157" *)
  EEP_Write( 157 , password ) ;

end_if ;
```

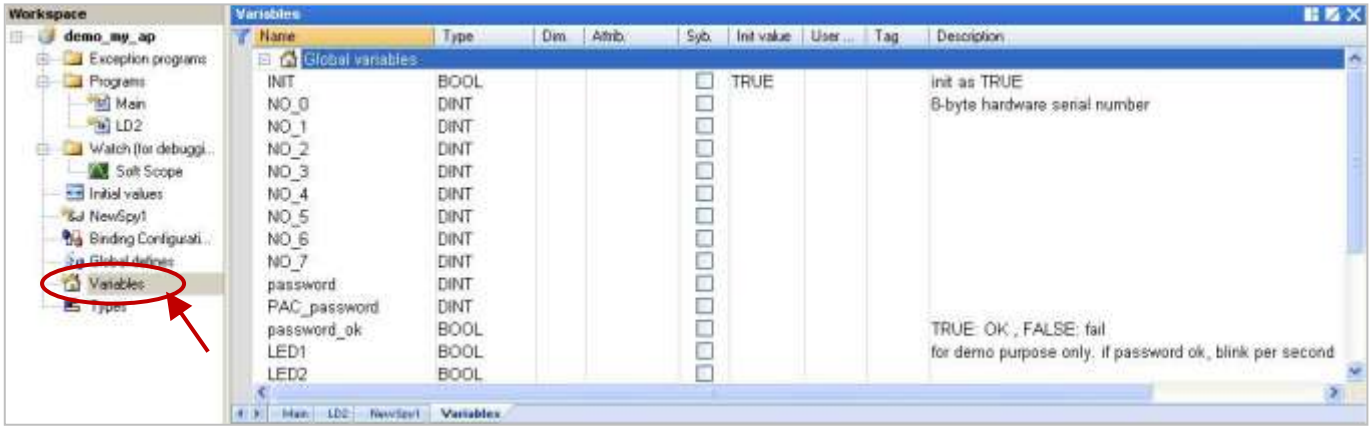
"demo_my_ap" 應用程式:

一開始會使用 "PAC_SN" 函式來讀出序號，接著再算出密碼，另外也會從 EEPROM 內讀出密碼來比對密碼是否正確?



(**註:** 您可參考 2.1.2 節，讓程式依照執行的順序來排列)

變數宣告:



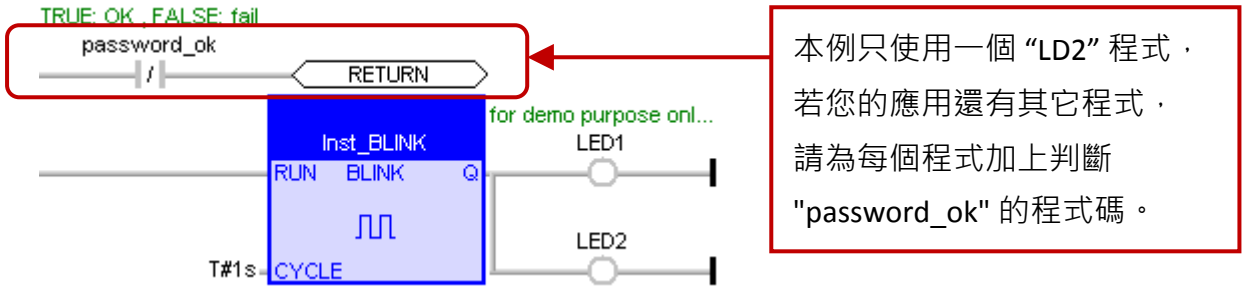
ST 程式 - Main:

```
(* 此 "demo_my_ap" 範例程式可由 PAC 的 EEPROM 內讀出密碼，並與使用者自訂的演算結果
   相互比對看密碼是否正確? *)

(* 宣告 "No_0" ~ "No_7", "password"  與 "PAC_password" 變數為 DINT。
   宣告 "INIT" 變數為 BOOL 且初值 (Initial value) 為 TRUE。
   宣告 "password_ok" 變數為 BOOL *)

(* 第一個 PAC Cycle 的操作 *)
if INIT then
    INIT := FALSE ; (* 表示不再是第一個 Cycle *)

  (* 取得硬體的序號 *)
  PAC_SN( No_0, No_1, No_2, No_3, No_4, No_5, No_6, No_7) ;
```

```
(* 使用您自訂的演算法來產生 "PAC_password" 的值 *)
PAC_password:=  (No_0 * No_1) + (No_2 * 12345) + No_3 + (No_4 * No_5) + No_6 + No_7 ;

(* 讀出先前儲存在 EEPROM，位址為 157 的密碼 *)
EEP_Read( 157 , password ) ;

(* 比對密碼是否正確? *)
password_ok := FALSE ; (* 一開始先設為 "FALSE" *)
if  password = PAC_password  then
  password_ok := TRUE ;  (* 密碼正確 *)
end_if ;

end_if ;
```

LD 程式 – LD2

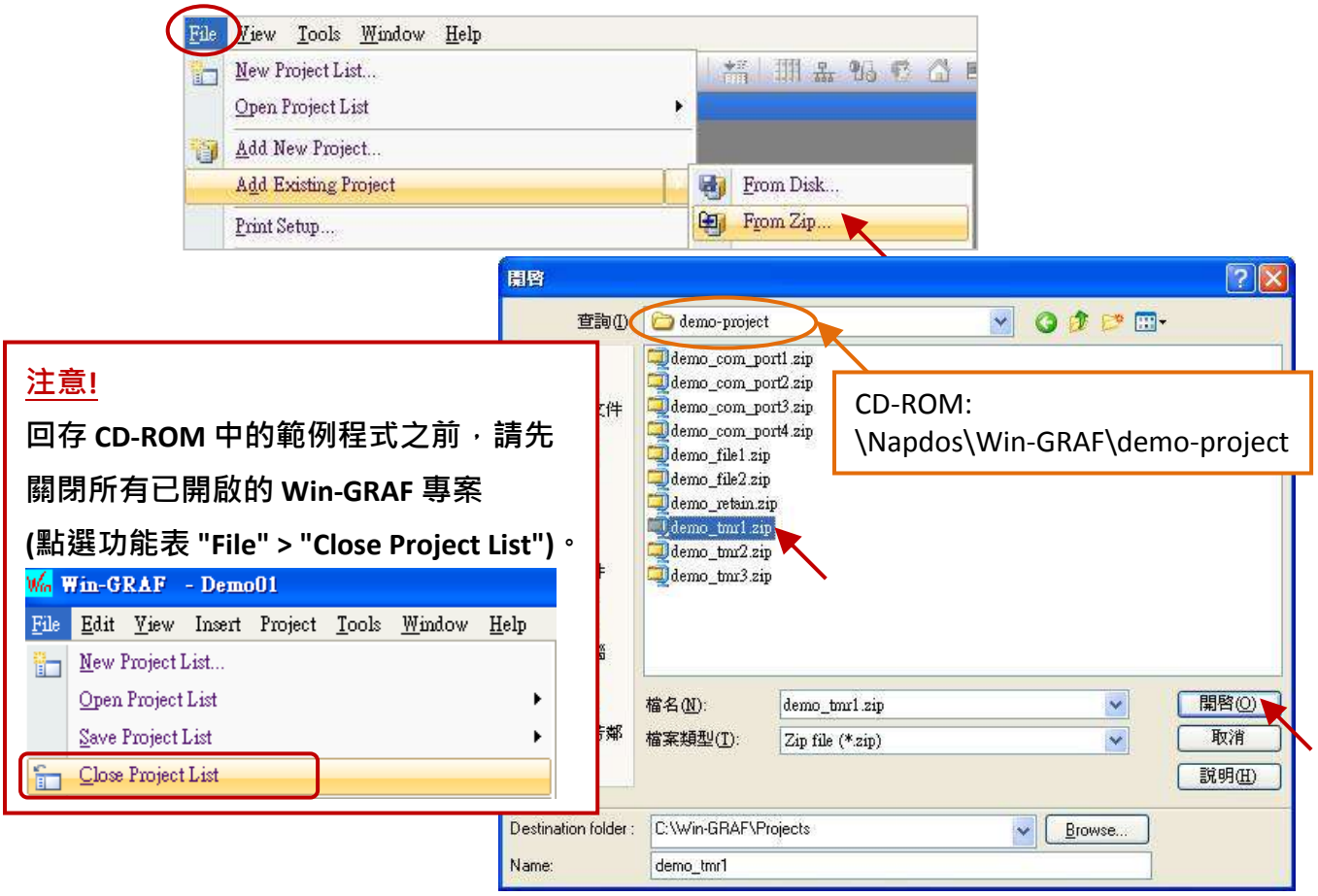
如果 "password_ok" 為 “FALSE“ 表示密碼不正確，會離開此程式。只有密碼正確時，才能執行後續的程式，如此即可保護您的應用程式無法被盜用者使用。



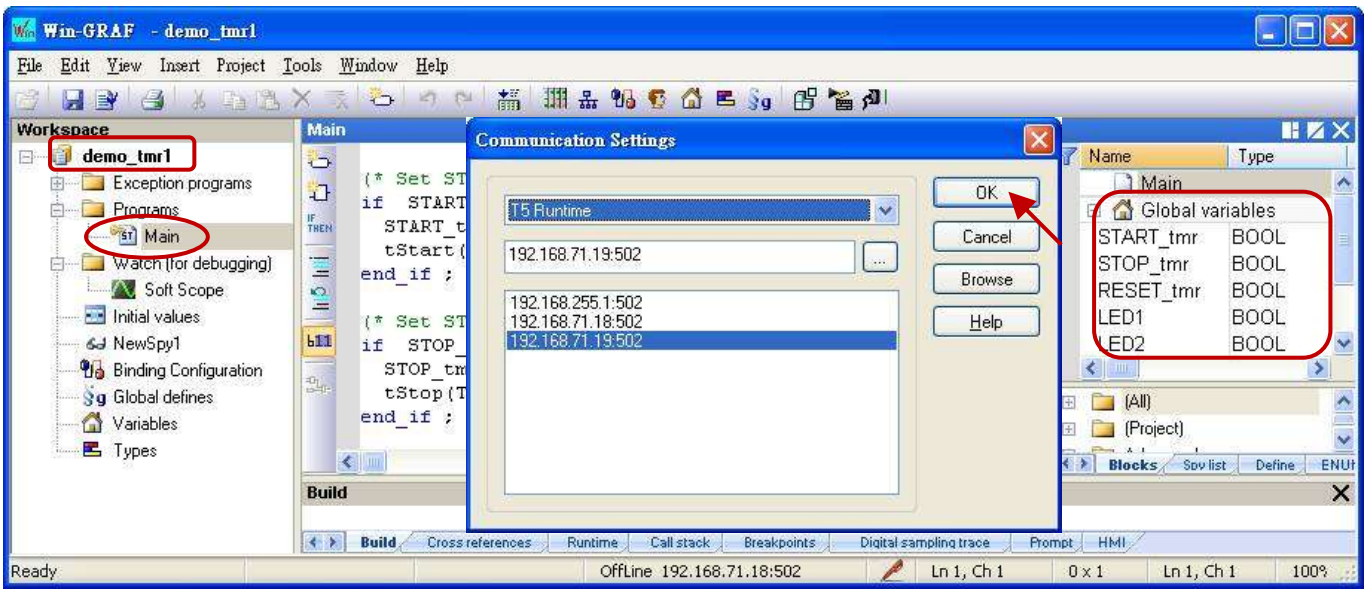
第 12 章 範例程式說明

在 Win-GRAF 的出貨光碟中 (CD-ROM: \Napdos\Win-GRAF\demo-project)，提供了以下章節的範例程式。使用前，請先進行以下步驟:

1. 點選功能表 "File" > "Add Existing Project" > "From Zip"，來開啟專案 (例如: "demo_tmr1.zip")。



2. 滑鼠雙擊 "Main" 可開啟此 ST 程式，於變數區可查看/建立變數。
3. 滑鼠右鍵點選專案名稱 (“demo_tmr1“) 再選擇 "Communication Parameters" 設定 PAC 目前的 IP 位址。(可參考 2.3.5 節)



12.1 計時器 (Timer) 操作

12.1.1 啟動、停止、重置計時

請參考 P12-1 來開啟此專案 ("demo_tmr1.zip")，於變數區可查看/建立變數。

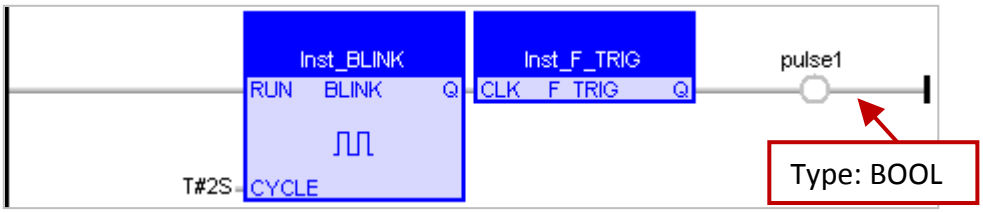
ST 程式:

```
(* 宣告 “START_tmr”, “STOP_tmr”, “RESET_tmr”, “LED1”, “LED2” 為 BOOL
   宣告 “TMR1” 為 TIME *)
(* 設定 “START_tmr” 為 TRUE, 以開始 "TMR1" 計時 *)
 IF   START_tmr   THEN
   START_tmr := FALSE ;
   TSTART (TMR1) ;
END_IF;

(* 設定 “STOP_tmr” 為 TRUE, 以停止 "TMR1" 計時 *)
 IF   STOP_tmr   THEN
    STOP_tmr := FALSE ;
    TSTOP (TMR1) ;
 END_IF;

(* 設定 “RESET_tmr” 為 TRUE, 以重置 "TMR1" 為 T#0s *)
 IF   RESET_tmr   THEN
    RESET_tmr := FALSE ;
    TMR1 := T#0s ;
 END_IF;

(* 當 "TMR1" 為 T#3s ~ T#10s 時, “LED1 ~ LED2” 為 ON *)
 LED1 := FALSE ;
 LED2 := FALSE ;
 IF   (TMR1 >= T#3s) and  (TMR1 <= T#10s)  THEN
      LED1 := TRUE ;
      LED2 := TRUE ;
 END_IF;

(* 當 "TMR1" 為 T#15s 時, 自動重置 “TMR1” 為 T#0s *)
 IF   TMR1 >= T#15s  THEN
      TMR1 := T#0s ;
 END_IF;
```

12.1.2 週期性的操作

請參考 P12-1 來開啟此專案 ("demo_tmr2.zip")，於變數區可查看/建立變數。"BLINK" 功能搭配 "F_TRIG" 功能方塊，可每隔一段時間產生一個 Pluse TRUE，因此它可以應用在週期性的操作上。

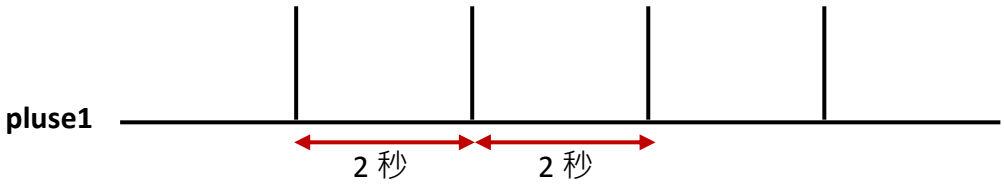
LD 程式:



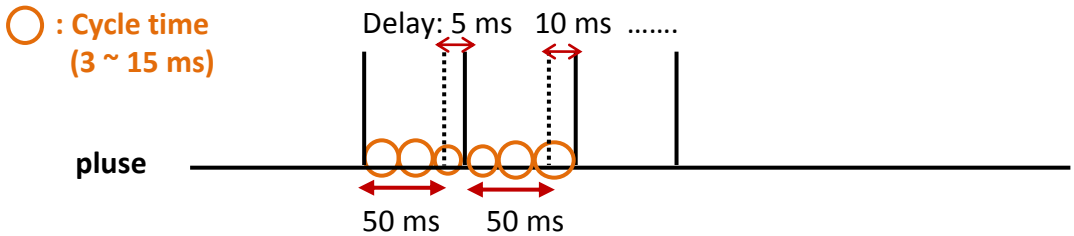
ST 程式:

```
IF   pluse1   THEN
  (*  執行操作 *)
  • • •
END_IF;
```

上方的 "BLINK" 與 "F_TRIG" 將會每 2 秒產生一個 Pluse TRUE，但是此方式有個缺點:



若週期的間隔時間較短時 (例如: 間隔時間為 100 ms 或更小；或是 PAC 的 Cycle time 較大時，像是 20 ~ 50 ms，一般為 3 ~ 15 ms)，則此週期性的操作會不準確。例如，每 50 ms 執行一次週期性操作，由於相較於 250 ms 或 2 秒，間隔時間為 50 ms 相當接近 PAC 的 Cycle time，若使用功能方塊的方式很容易累積延遲輸出的時間，因此最終的操作時間也變得不準確。



為了改善此狀況，使用以下的編寫方式會較精確:

請參考 P12-1 來開啟此專案 ("demo_tmr3.zip")，於變數區可查看/建立變數。

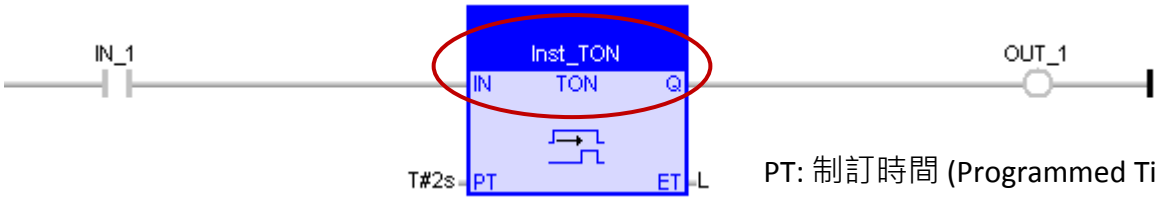
ST 程式:

```
(* 宣告 “INIT” 為 BOOL 且設定初值為 TRUE
   宣告 “TMR1”, “TMR1_next” 為 TIME *)

 IF  INIT  THEN
    INIT := FALSE ;
    TMR1 := T#0s ;
   TMR1_next := TMR1 + T#50 ms ;
   TSTART (TMR1);
END_IF;

IF  TMR1 >= TMR1_next  THEN
   IF  TMR1 > T#10h  THEN
       TMR1 := T#0s ;
       TMR1_next := T#0s ;
   END_IF;
   TMR1_next := TMR1_next + T#50 ms ;
   (* 執行操作 *);
   • • •
END_IF;
```

當計時到 T#23h59m59s999ms 時，Timer 的值會產生溢位 (Overflow)。因此，可設定它在 10 或 18 小時，自動重置為 "0"。

12.1.3 偵測穩定的 ON 或 OFF 訊號

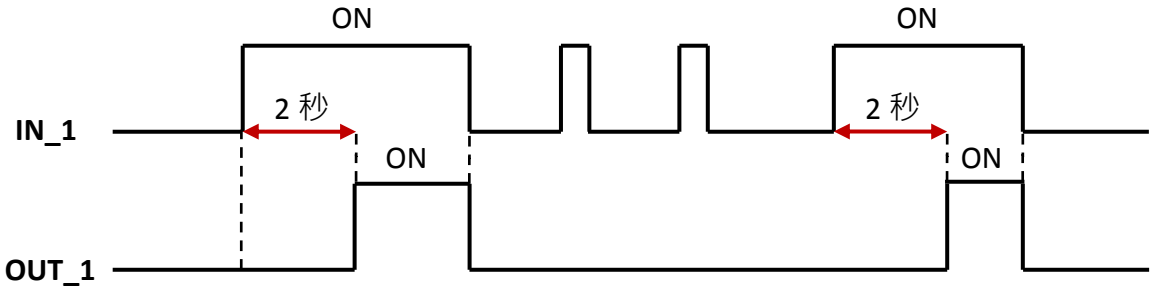
"TON" 功能方塊可偵測經過一段時間，仍維持 **"ON"** 的穩定訊號。



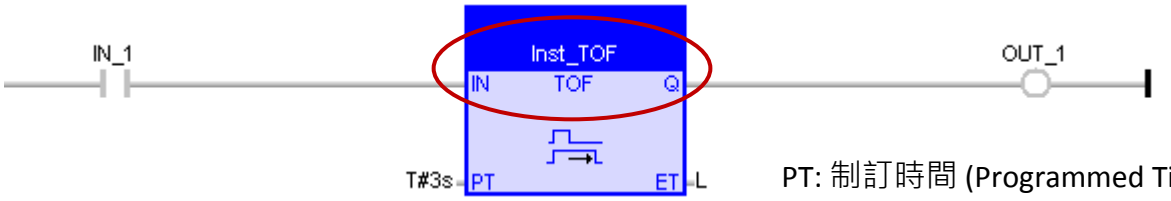
PT: 制訂時間 (Programmed Time)

EL: 經過時間 (Elapsed Time)

如上圖，此功能可偵測維持 2 秒仍 "On" 的穩定訊號。



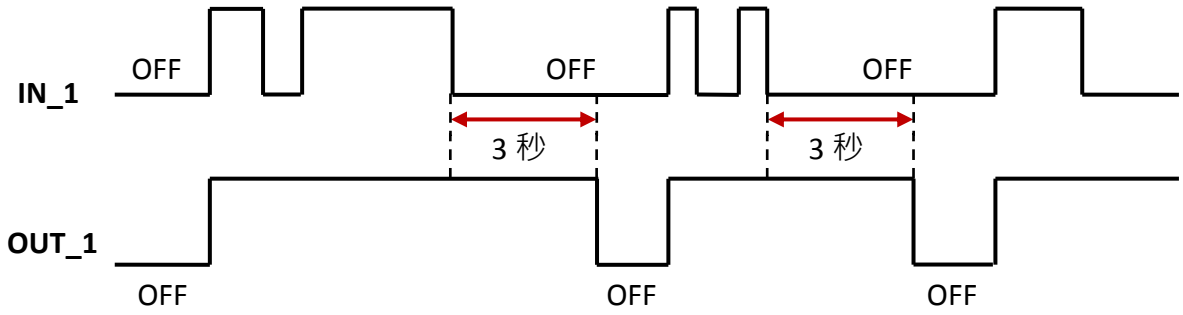
"TOF" 功能方塊可偵測經過一段時間，仍維持 **"OFF"** 的穩定訊號。



PT: 制訂時間 (Programmed Time)

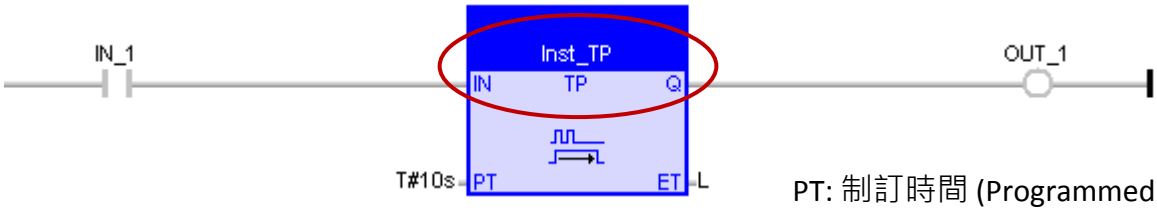
EL: 經過時間 (Elapsed Time)

如上圖，此功能可偵測維持 3 秒仍 "OFF" 的穩定訊號。



12.1.4 觸發後維持 "ON" 一段時間

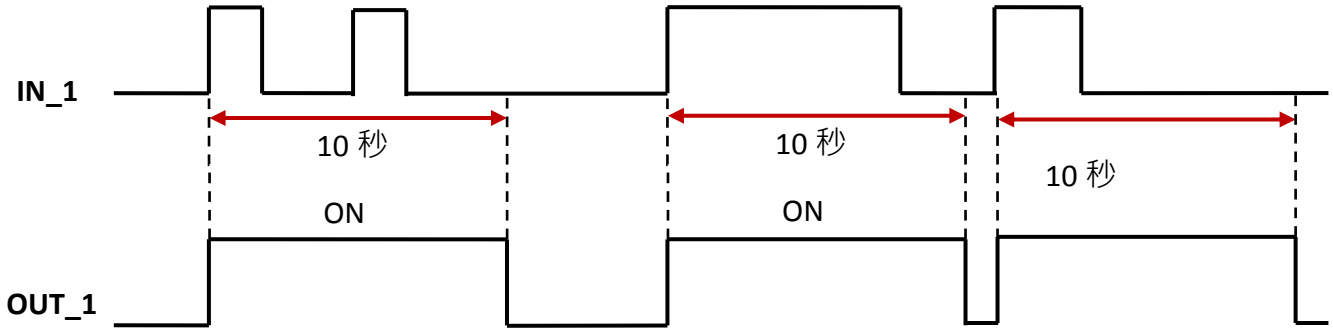
"TP" 功能方塊可在觸發後 (即，由 OFF 變成 ON 時) 維持輸出 **"ON"** 一段時間。



PT: 制訂時間 (Programmed Time)

EL: 經過時間 (Elapsed Time)

如上圖，觸發後維持 10 秒 "ON" 的訊號輸出。

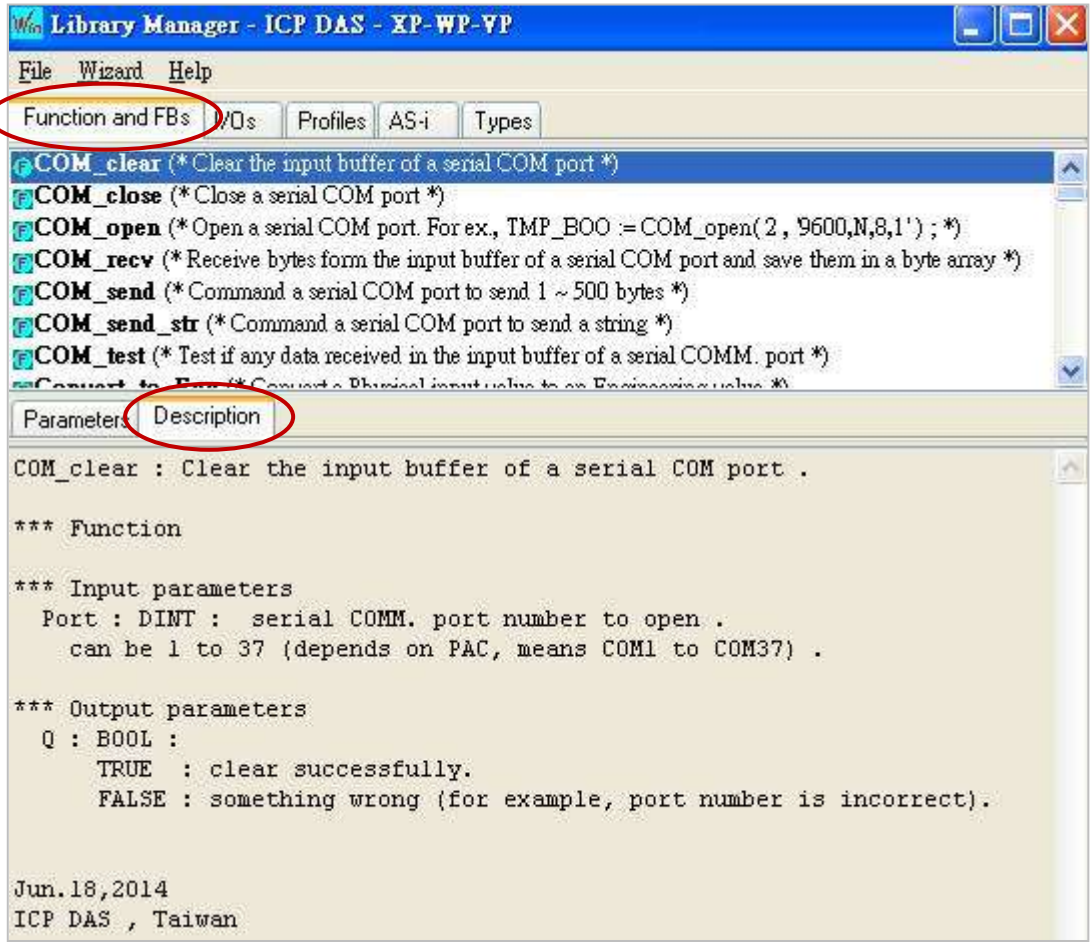


12.2 序列埠的通訊操作

使用者可直接操作序列埠 (即，RS-232, RS-485 或 RS-422 Port) 來實現一些特定的通訊協定。以下有 7 個函式 (Function) 可用來直接操作序列埠。

函式 (Function)	說明
COM_open	開啟一個序列埠。
COM_close	關閉一個序列埠。
COM_clear	清除序列埠的輸入暫存區。
COM_test	測試序列埠是否有接收到任何 Byte?
COM_send	傳送一些 Byte 給序列埠。
COM_send_str	傳送一些 String 給序列埠。
COM_recv	接收一些 Byte 給序列埠。
COM_status	取得 COM Port 是否已正常開啟的狀態。

可參考 1.2.3 節 來開啟程式庫管理員 (Library Manager) 並查詢各函式 (Function) 的使用說明。



12.2.1 使用 COM Port 來傳送一個字串

請參考 P12-1 來開啟此專案 ("demo_com_port1.zip")，於變數區可查看/建立變數。

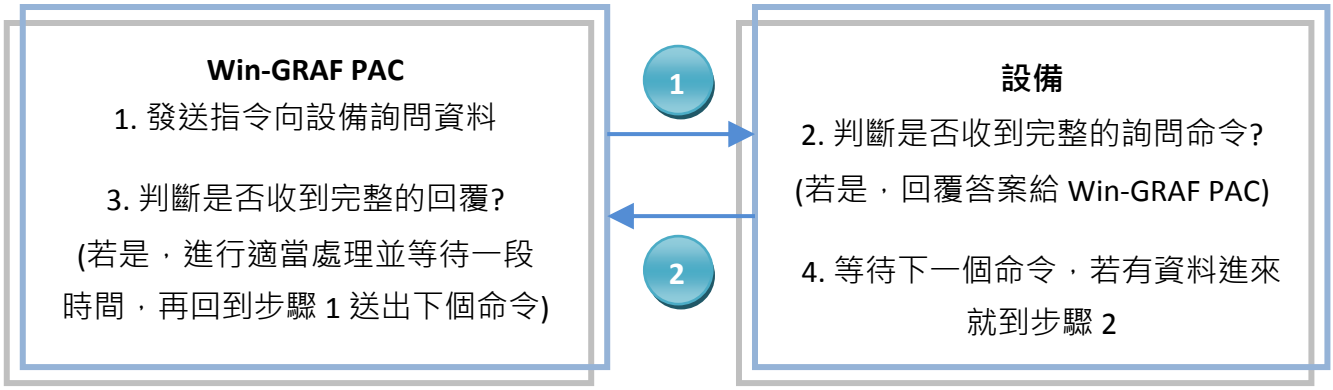
ST 程式: 此程式可以每 2 秒從 PAC 的 COM1 (通訊參數: `9600, N, 8, 1') 送出一個字串 (例如: < CNT1 = 1 > 或 < CNT1 = 25 >, ...等)。

宣告 "INIT" 為 BOOL 且初值為 TRUE ;
"Port_OK" 為 BOOL ;
"CNT1" 為 DINT ;
"TMR1", "TMR1_next" 為 TIME
"Port_number" 為 DINT 且初值 為 "1"。

```
(* 在第一個 PAC cycle 執行操作 *)
if INIT then
  INIT := FALSE ; (* 非第一個 cycle *)
  CNT1 := 0 ;
  TMR1 := T#0s ;
  TMR1_next := TMR1 + T#2s ;
  (* 開始 "TMR1" 計時 *)
  tStart(TMR1) ;
end_if ;

(* 若 COM port 的狀態變為 FALSE (未開啟), 則開啟它 *)
if COM_Status(Port_number) = FALSE then
  (* 開啟一個序列 COM port *)
  Port_OK := COM_open(Port_number, '9600,N,8,1' ) ;
end_if ;

(* 到達時間時 , ... *)
if TMR1 >= TMR1_next then

  (* 避免 "TMR1" 達到 T#23h59m59s999ms 發生溢位 *)
  if TMR1 > T#10h then
    TMR1 := T#0s ;
    TMR1_next := T#0s ;
  end_if ;

  (* 為 "TMR1_next" 設定新值 *)
  TMR1_next := TMR1_next + T#2s ;

  (* 由 COM port 傳送字串 *)
  COM_send_str( Port_number, '<CNT1=' + Any_to_STRING(CNT1) + '>' ) ;

  (*到達 100 時, 重置 CNT1 為 0 *)
  CNT1 := CNT1 + 1 ;
  if CNT1 >= 100 then
    CNT1 := 0 ;
  end_if ;
end_if;
```

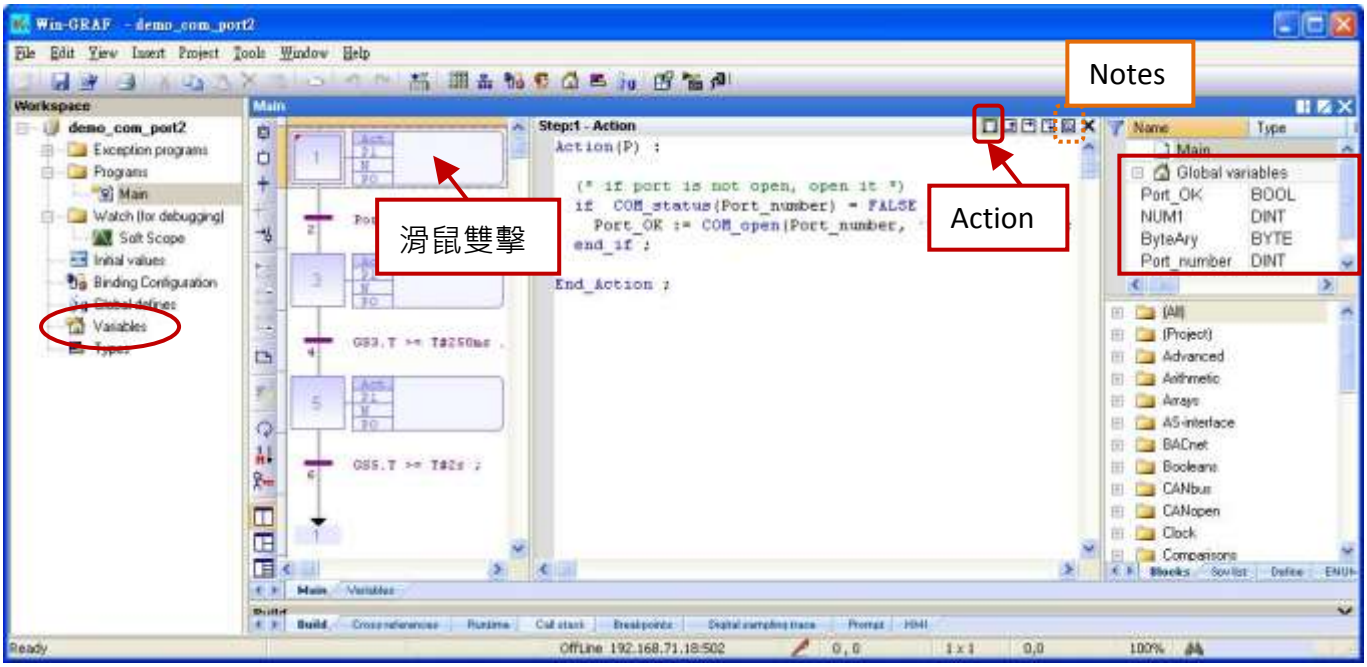
12.2.2 使用 COM Port 對設備一問一答

如有應用需使用 RS-232/485/422 Port 來取得其它設備的資料，其一問一答的方式如下:

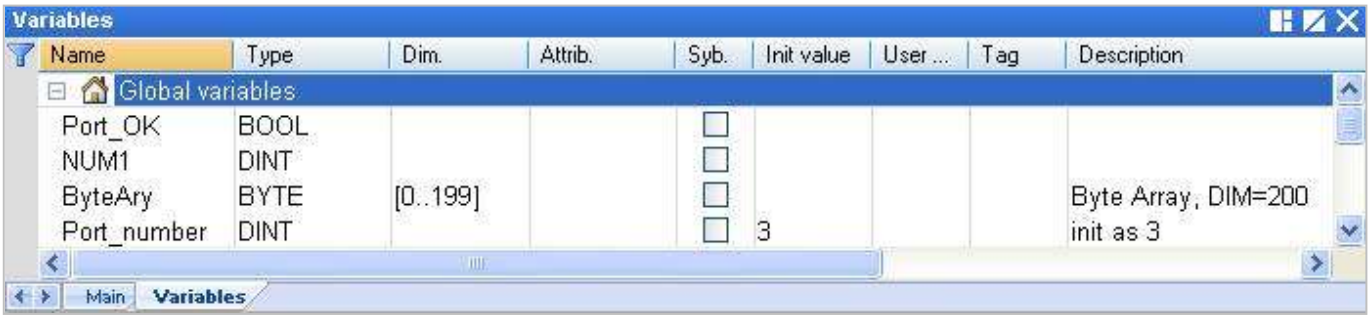


請參考 P12-1 來開啟此專案 ("demo_com_port2.zip")，於變數區可查看/建立變數。

註: 滑鼠雙擊 “Action” 會先開啟 “Notes” 視窗，請切換到 “Action” 視窗來查看程式碼。



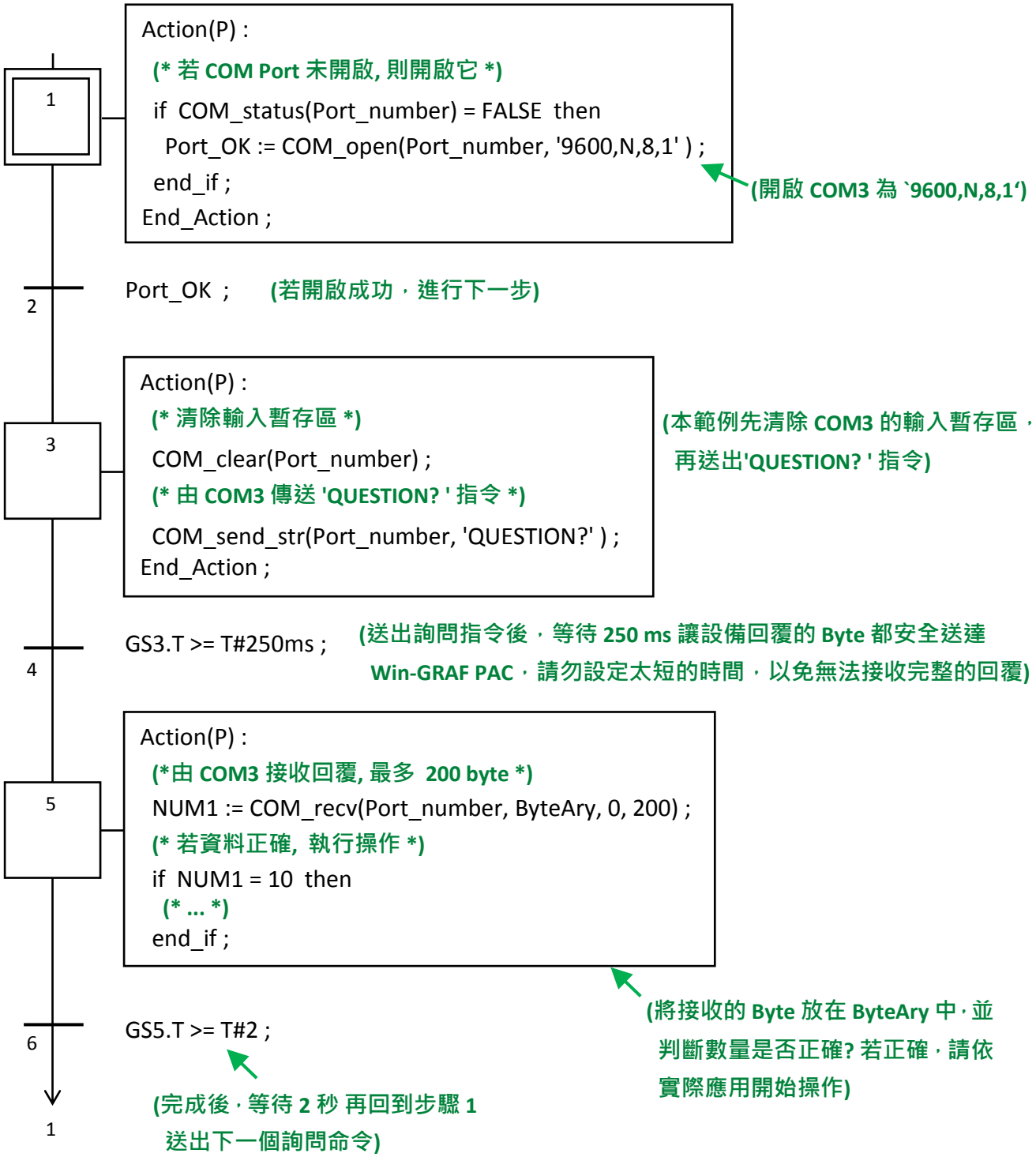
您也可點選 “Variables” 開啟變數視窗。



本範例先由 Win-GRAF PAC 的 COM3 送出一個字串 `QUESTION?' 給設備，接著等待設備的回覆並做處理，完成後等待 2 秒，再送出同樣的 `QUESTION?' 命令，...如此重複進行。

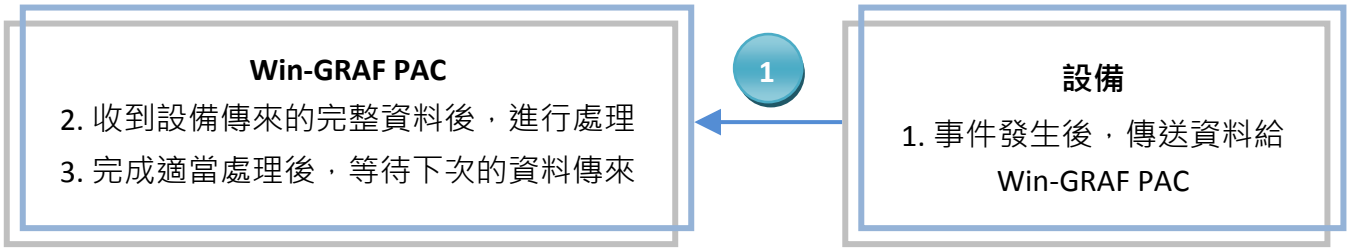
SFC 程式:

(變數 "Port_OK" 為 BOOL ; "NUM1" 為 DINT ; "ByteAry" 為 BYTE 且 Dim. 為 "200" ;
"Port_number" 為 DINT 且 初值 為 "3" 。)



12.2.3 COM Port 等待遠端設備傳來的資料

此種方式在一般的商場或便利商店很常見，像是使用條碼閱讀機。當讀取到商品的條碼後，它會傳送 條碼資料 到 Win-GRAF PAC 的 COM Port (RS-232/485/422)，並且不需回覆任何訊息。



請參考 P12-1 來開啟此專案 ("demo_com_port3.zip")，於變數區可查看/建立變數。

ST 程式:

```
(* 在第一個 PAC cycle 執行操作 *)
if INIT then
  INIT := FALSE ;
  T1 := T#0s ;
  STEP1 := 0 ;
end_if ;

(* 若 COM Port 未開啟, 則開啟它 *)
if COM_status(Port_number) = FALSE then
  Port_OK := COM_open( Port_number , '9600,N,8,1' ) ;
end_if ;

(* 若開啟 COM Port 失敗，離開此 ST 程式 *)
if Port_OK = FALSE then
  return ;
end_if ;

CASE STEP1 OF

  (* 是否有至少 1 Byte 的資料傳來 *)
  0:
    if COM_test(Port_number) then
      STEP1 := 1 ;
      T1 := T#0s ;
      Tstart(T1) ;
    end_if ;
```

宣告 "INIT" 為 BOOL 且初值為 TRUE ;
"Port_OK" 為 BOOL ;
"STEP1", "NUM1" 為 DINT ;
"T1" 為 TIME ;
"ByteAry" 為 BYTE 且 Dim. 為 "200" ;
“Port_number” 為 DINT 且初值 為 "3"。

STEP1 = 0，表示等待中並測試 COM3 是否有傳資料進來?
若回傳 TRUE，表示有資料。
將 STEP1 設定為 "1"，T1 設定為 "0" 並開始計時。

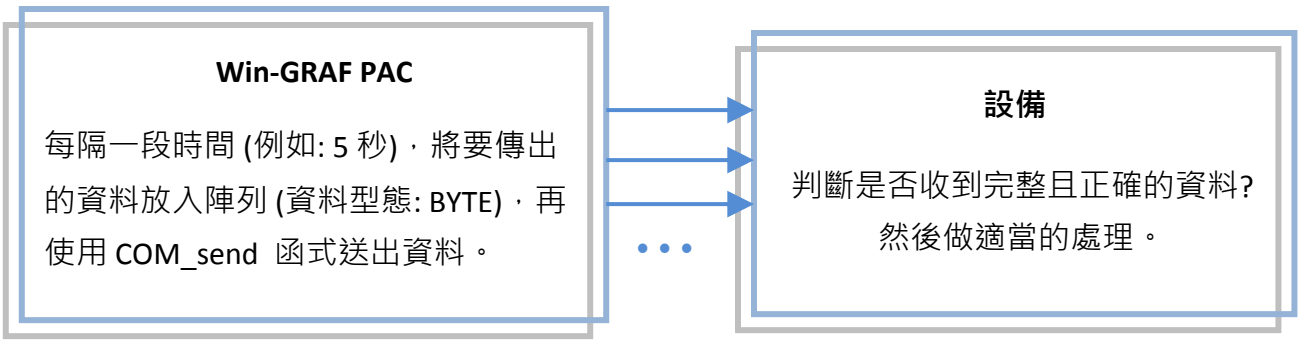
```
  (* 等待 250 ms, 並由 COM port 接收所有的 Byte *)
1:
   if  T1 >= T#250ms  then
     Tstop(T1) ;
     T1 := T#0s ;
     STEP1 := 0 ;

     (* 最多可接收 200 Byte *)
     NUM1 := COM_recv(Port_number , ByteAry , 0 , 200 ) ;

     (* 若資料是正確的, 此例為 25 Byte 則執行操作 *)
     if  NUM1 = 25  then
       (* ... *)
     end_if ;

   end_if ;

END_CASE ;
```

STEP = 1，表示資料正傳送進來，等待 250 ms 將所有接收的資料放在陣列中，此等待時間和 設備規格 與 Baud Rate 有關，若設定的時間太短可能會接收不完全。
記得將 STEP1 設定為 "0" 來等待下次資料再傳進來。

收到資料後，判斷資料對不對? 再依實際應用開始操作。

12.2.4 使用 COM Port 定期回報資料給遠端設備

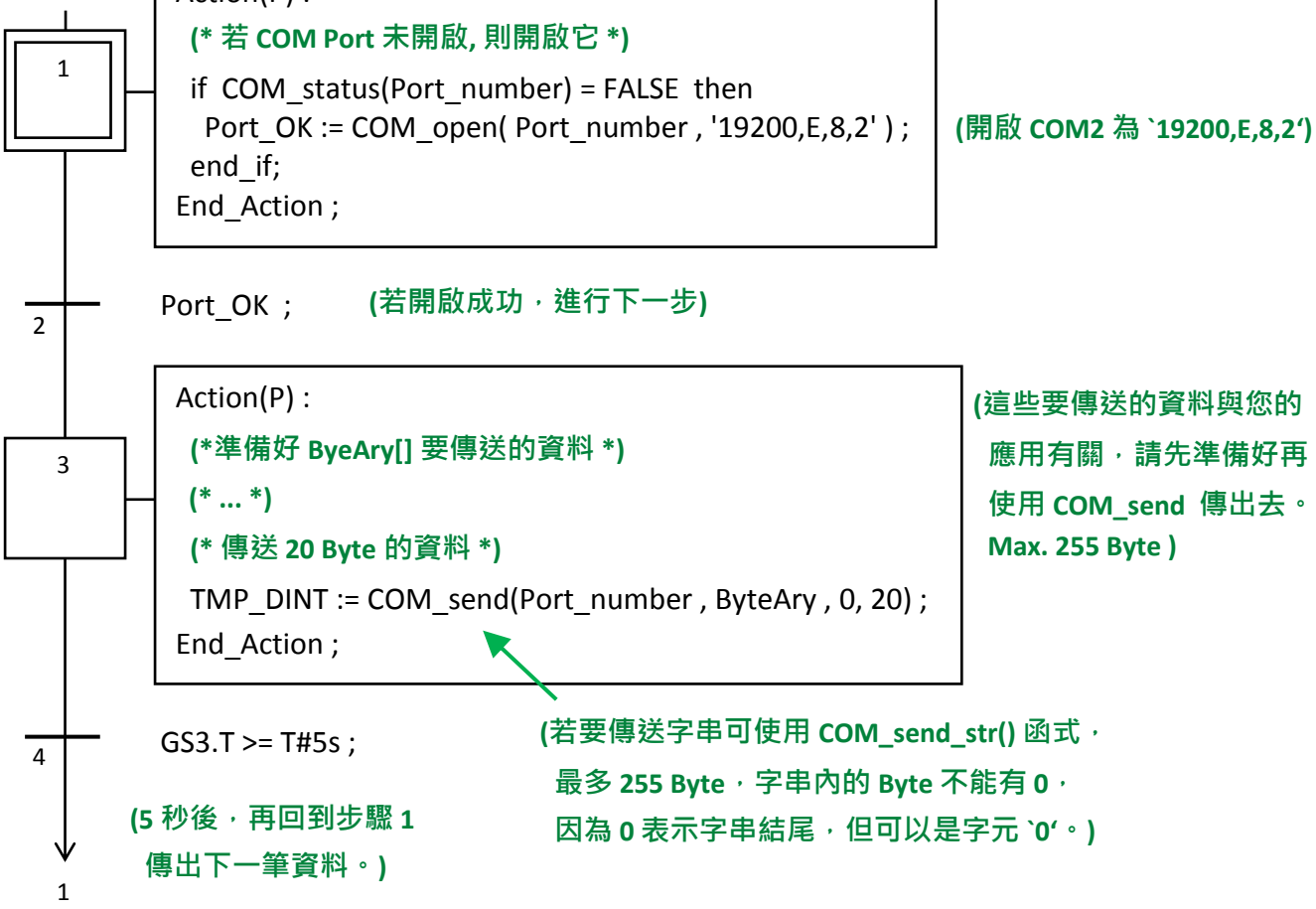
如需每隔一段時間使用 RS-232/485/422 Port 將資料回報給其它的設備，方式如下:



請參考 P12-1 來開啟此專案 ("demo_com_port4.zip")，於變數區可查看/建立變數。

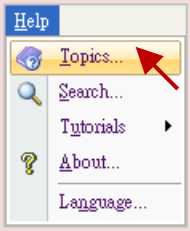
SFC 程式: (可參考 12.2.2 節 來開啟 "Action" 視窗。)

(宣告變數: "Port_OK" 為 BOOL ； "TMP_DINT" 為 DINT ； "ByteAry" 為 BYTE 且 Dim. 為 100 ；
"Port_number" 為 DINT 且初值 為 "2"。)



12.3 讀/寫 PAC 內儲存裝置的檔案 (File)

Win-GRAF Workbench 提供了下列函式 (Function)，可對 PAC 內的儲存裝置進行讀/寫操作。

函式名稱	說明
請點選功能表 "Help" > "Topics" 並輸入搜尋字串 “File”，於 “File Management functions” 主題中來了解詳細的使用說明。 	
F_ROPEN	開啟一個檔案以供讀取
F_WOPEN	開啟一個檔案以供寫入
F_AOPEN	以附加方式來建立/開啟檔案
F_CLOSE	關閉一個開啟中的檔案
F_EOF	測試若到達檔案尾端時，開啟檔案以供讀取
FA_READ	從二進位 (Binary) 檔案中，讀出整數值 (DINT)
FA_WRITE	從二進位 (Binary) 檔案中，寫入整數值 (DINT)
FM_READ	從文字檔中，讀出字串 (STRING)
FM_WRITE	從文字檔中，寫入字串(STRING)
FB_READ	從檔案中，讀出二進位值 (Binary)
FB_WRITE	從檔案中，寫入二進位值 (Binary)
F_EXIST	測試檔案是否存在?
F_GETSIZE	取得檔案大小
F_COPY	複製檔案
F_DELETE	刪除檔案
F_RENAME	將檔案重新命名
請參考 1.2.3 節 來查詢以下 Function (函式) 的使用說明明	
F_dir	建立一個目錄
F_cp_dir	複製一個目錄與其內的檔案到另一個目錄 (不含子目錄)
F_del_dir	刪除一個目錄與其內的檔案 (不含子目錄)

註: ICP DAS 的 Win-GRAF PAC 並不支援 “F_SAVERETAIN” 與 “F_LOADERETAIN” 函式。

12.3.1 寫入資料到 PAC 內的檔案

請參考 P12-1 來開啟此專案 ("demo_file1.zip")，於變數區可查看/建立變數。

ST 程式: 此程式可用來寫入 10 個 "REAL" 數值到 PAC 內的檔案。

```
(*    此 "demo_file1" 專案會把 10 個 REAL 數值儲存到 \System_Disk\Real_data1.txt
      檔案格式 :
        每一列有一個 REAL 數值且包含了換行字元 (即，<CR><LF>)。
        1.08
        2.786
        38.45
        41.5
        59.875
        60.76
        71.23
        80.5
        99.8
        100.7        *)

(*  變數宣告:
    Write_File       : BOOL
    Tmp_string       : String, len=255
    File_ID          : DINT
    REAL_val[0..9] : REAL
    ii               : DINT
    File_Status      : String, len=128    *)

(* 將 "Write_File" 設定為 "TRUE"，將資料寫入檔案中 *)
if  Write_File  then

  Write_File := FALSE ;
  File_ID := F_Wopen( '\System_Disk\Real_data1.txt' );

 if  File_ID = 0  then
   (* 在寫入模式將無法開啟檔案 *)
   File_Status := 'Can not open file in write mode !' ;
 else
  (* 若開啟檔案成功, 將 REAL[0] ~ [9] 的值儲存到檔案中, 每列包含 1 個 REAL 值 與
     結束字元 <CR><LF> *)
   File_Status := 'Open file ok.' ;
   for ii := 0 to 9  by 1  do
```

由於 \System_Disk\ 的容量較小，建議您將檔案路徑改為以下位置:

WP-8xx8, WP-5xx8, VP-25W8, VP-4138:
\Micro_SD 或是

XP-8x48, XP-8x48-CE6, XP-8x48-Atom-CE6:
\System_Disk2

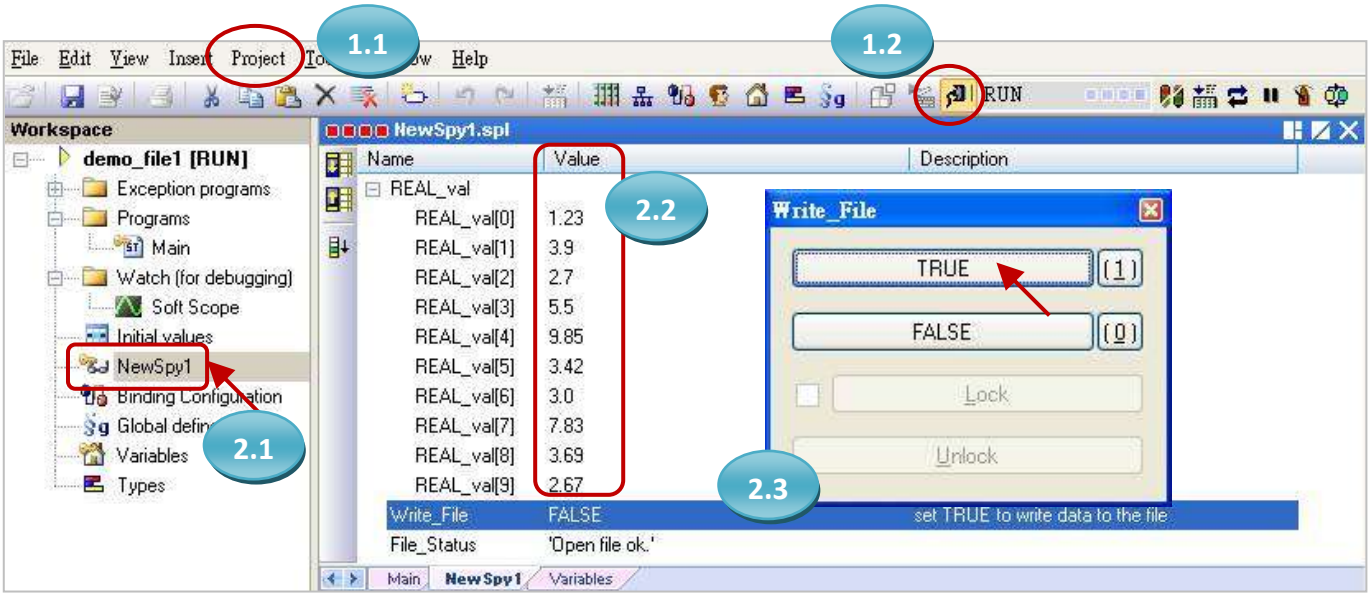
```
    Tmp_string := Any_to_string( REAL_val[ii] ) ;
    FM_write( File_ID , Tmp_string ) ;
  end_for ;

  (* 關閉檔案 *)
  F_close( File_ID ) ;

 end_if ;
end_if ;
```

若想存成**整數**資料，可修改為:
Tmp_string := Any_to_string(**DINT_val[ii]**) ;
且變數區也需宣告 “DINT_val” 為 DINT 且 Dim. 至少為 "10"。

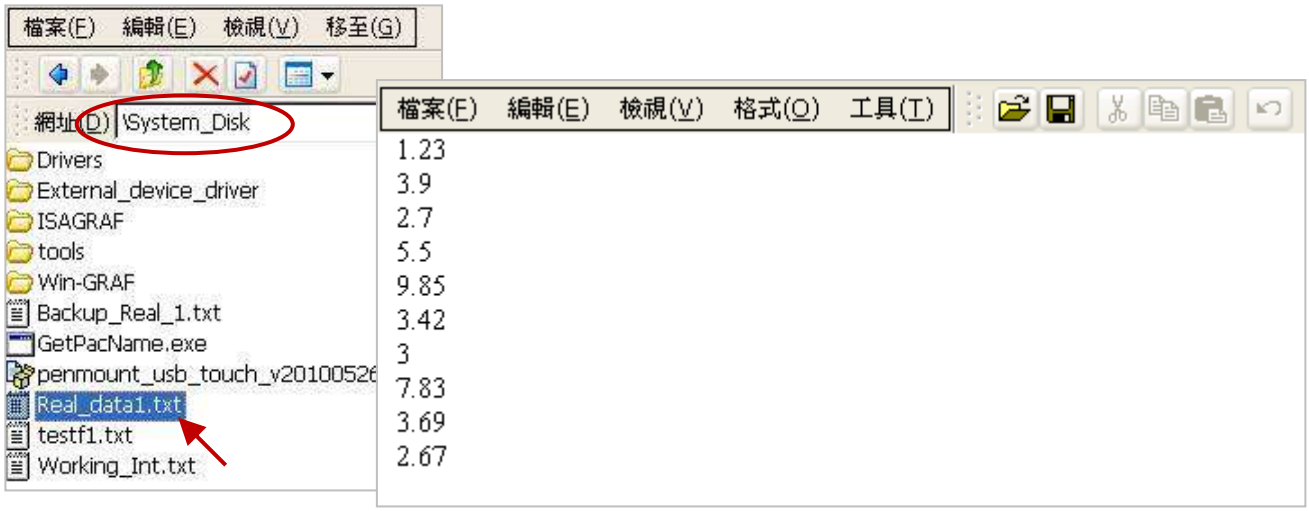
測試程式:

此例當 "Write_File" 設為 "TRUE" 時，會將數值寫入 PAC 內的 \System_Disk\Real_data1.txt 檔案中。

1. 請先完成 IP 設定 (見 P12-1)，編譯並下載程式到 PAC 中。
   (點選 "Project" > "Build All Projects" / "On Line"，若不熟悉操作，可參考 2.3.4 節、2.3.5 節)
2. 點選 "NewSpy1" 開啟觀測清單，並填入要寫入的數值，再將 "Write_File" 設定為 "TRUE"，來寫入資料。(若 OK，"File_Status" 會顯示 "Open file ok")



3. PAC 中，開啟 "Real_data1.txt" 檔案，可見到步驟 5 所填入的值。



12.3.2 讀取 PAC 內的檔案資料

請參考 P12-1 來開啟此專案 ("demo_file2.zip")，於變數區可查看/建立變數。

ST 程式: 此程式可用來讀取 PAC 內檔案中的 10 個 "REAL" 數值。

```
(*  此 "demo_file2" 專案會從 \System_Disk\Real_data2.txt 檔案中讀取 10 個 REAL 數值
     檔案格式 :
       每一列有一個 REAL 數值且包含了換行字元 (即，<CR><LF>)。
       1.08
       2.786
       38.45
       41.5
       59.875
       60.76
       71.23
       80.5
       99.8
       100.7
*)

(*
  變數宣告:
  Write_File      : BOOL
  Tmp_string      : String, len=255
  File_ID         : DINT
  REAL_val[0..9] : REAL
  ii              : DINT
  File_path       : String, len = 128, initial val = '\System_Disk\Real_data2.txt'
  File_Status     : String, len=128
*)

(* 將 "Read_File" 設定為 "TRUE"，來讀取檔案資料 *)
if  Read_File  then

   Read_File := FALSE ;
   (* 檢查檔案是否存在? *)
   if  F_exist( File_path ) = FALSE  then
     (*檔案不存在 *)
     File_Status := 'File "' + File_path +'" does not exist !' ;
   else

     (* 若檔案存在, 開起並讀取檔案 *)
     File_ID := F_Ropen( File_path );

     if  File_ID = 0  then
       (* 開啟檔案失敗 *)
       File_status := 'Can not open File "' + File_path +'" !' ;
```

由於 \System_Disk\ 的容量較小，建議您將檔案路徑改為以下位置:

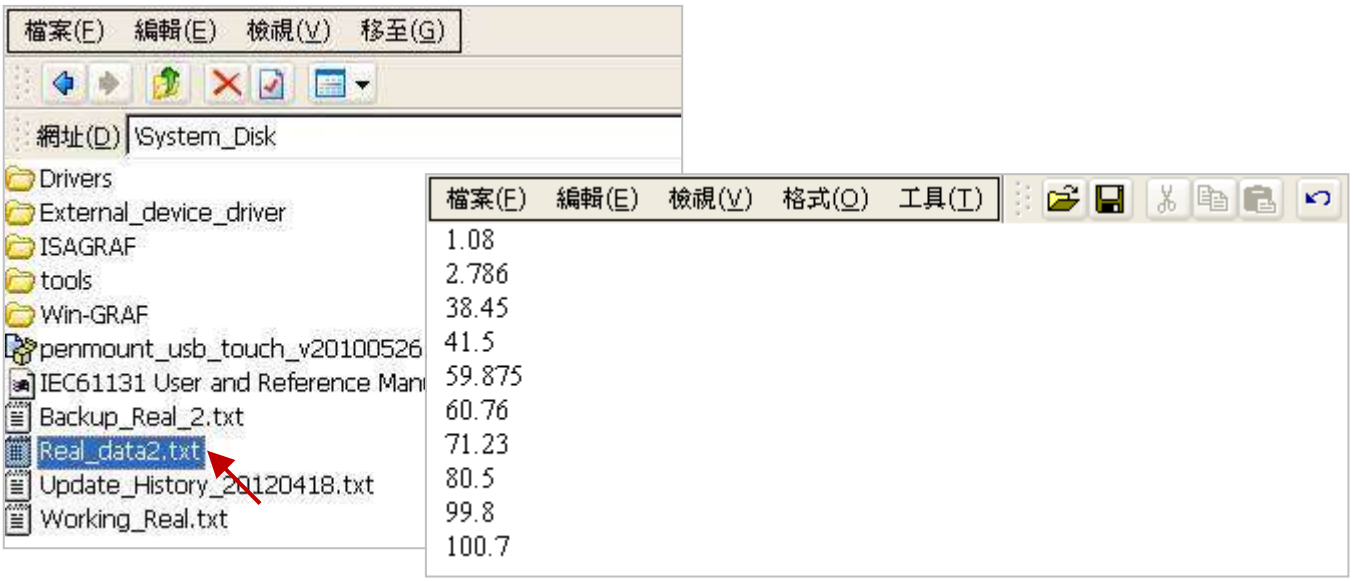
WinPAC, ViewPAC 系列:
\Micro_SD 或是

XP PAC 系列:
\System_Disk2

```
(* 開啟檔案成功, 讀取檔案中 REAL[0] ~ [9] 的數值,
   每列包含 1 個 REAL 值與結束字元 <CR><LF> *)
File_status := 'Open File "' + File_path +'" Ok .' ;
for ii := 0 to 9  by 1  do

  (*  測試是否讀到檔案的最後一列? *)
  if  F_EOF( File_ID )  then
    (*讀到檔案的最後一列, 離開 "for" 迴圈 *)
     exit ;
  end_if ;

  (* 讀取檔案中的一列 String 資料 *)
  Tmp_string := FM_READ( File_ID ) ;

  (* 將 String 轉換為 REAL 數值 *)
  REAL_val[ii] := Any_to_REAL( Tmp_string ) ;

end_for ;

(* 關閉檔案 *)
F_close( File_ID ) ;

  end_if ;
 end_if ;
end_if ;
```

若想讀取**整數**資料，可修改為:
DINT_val[ii] := Any_to_**DINT**(Tmp_string) ;
且變數區也需宣告 “DINT_val” 為 DINT 且 Dim. 至少為 "10"。

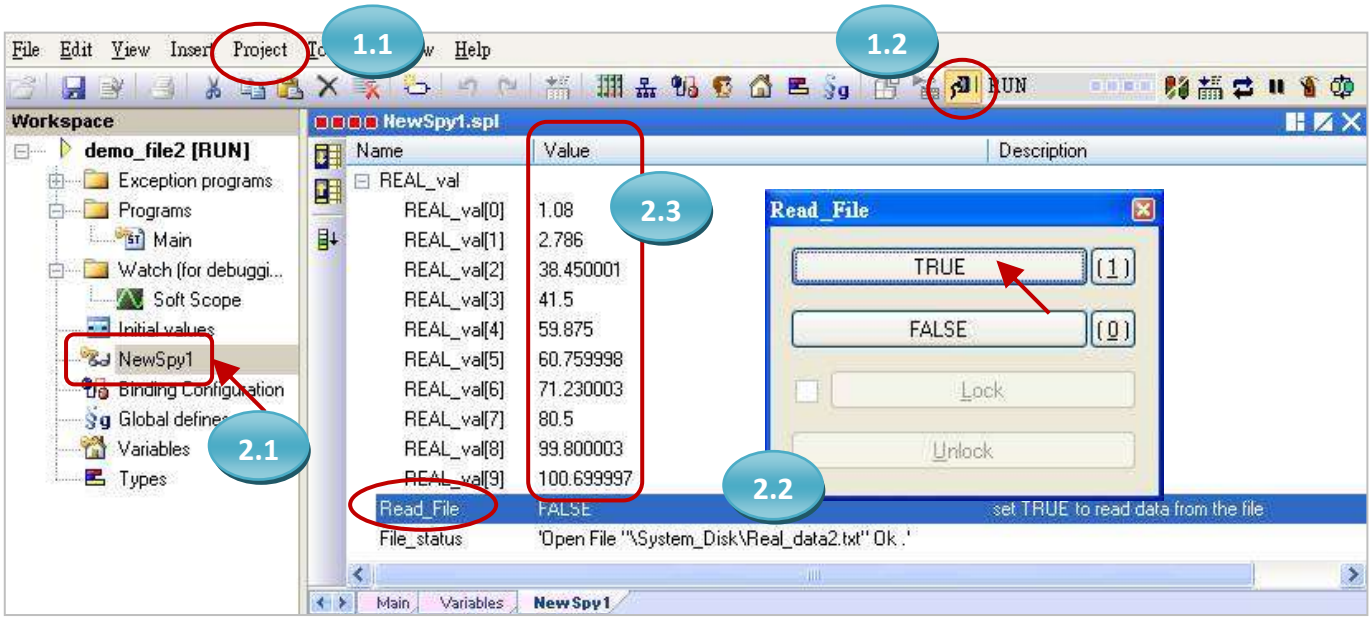
註: 此範例當 "Read_File" 設定為 "TRUE" 時，會去讀取 PAC 內 "\System_Disk\Real_data2.txt" 檔案，請確認此檔案已存放在 PAC 中。



測試程式:

1. 請先完成 IP 設定 (見 P12-1)，編譯並下載程式到 PAC 中。
   (點選 "Project" > "Build All Projects" / "On Line"，若不熟悉操作，可參考 2.3.4 節、2.3.5 節)

2. 點選 "NewSpy1" 開啟觀測清單，再將 "Read_File" 設定為 "TRUE"，來讀取資料。
   (若 OK，"File_Status" 會顯示 "Open File "\System_Disk\Real_data2.txt" Ok.")



第 13 章 使用 VB.net 2008/2010 程式來 讀/寫 Win-GRAF 變數

本章以 Visual Studio .NET 2008 開發工具建立一個範例程式的方式來說明，範例程式可以在 XP-8xx8-CE6 , WP-8xx8, VP-25W8 , VP-4138 , WP-5xx8 產品包裝盒內附的 CD-ROM 內找到。

VB .NET 範例:

光碟 : \napdos\Win-GRAF\demo-project\vb.net_2008_demo\

demo_vb01 : 數位 I/O 範例，搭配 I-87055W 模組 (於 Slot 0)

demo_vb02 : 類比 I/O 範例，搭配 I-87024W (Slot 1) 與 I-8017HW (Slot 2) 模組

demo_vb03 : 讀/寫 Win-GRAF Internal Integers, Timer, Real 以及 String 變數 (無需 I/O 模組)

demo_vb04 : 讀/寫 Win-GRAF Internal String 變數 (無需 I/O 模組)

Win-GRAF 範例:

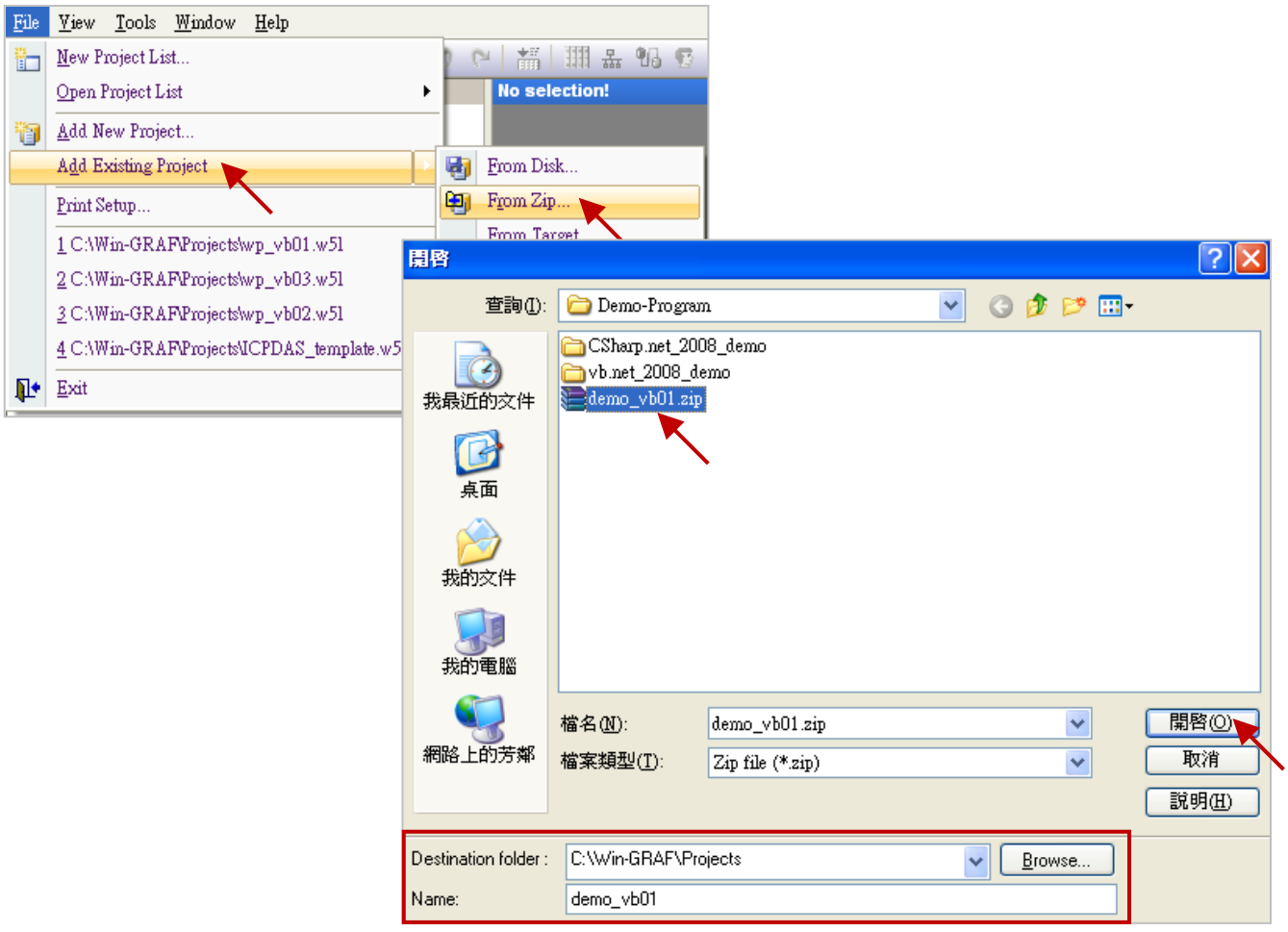
光碟 : \napdos\Win-GRAF\demo-project\

"demo_vb01.zip", "demo_vb02.zip", "demo_vb03.zip", "demo_vb04.zip"

13.1 如何回存 Win-GRAF 專案?

以下介紹如何把 Win-GRAF 範例程式，回存到 Win-GRAF Workbench 中。

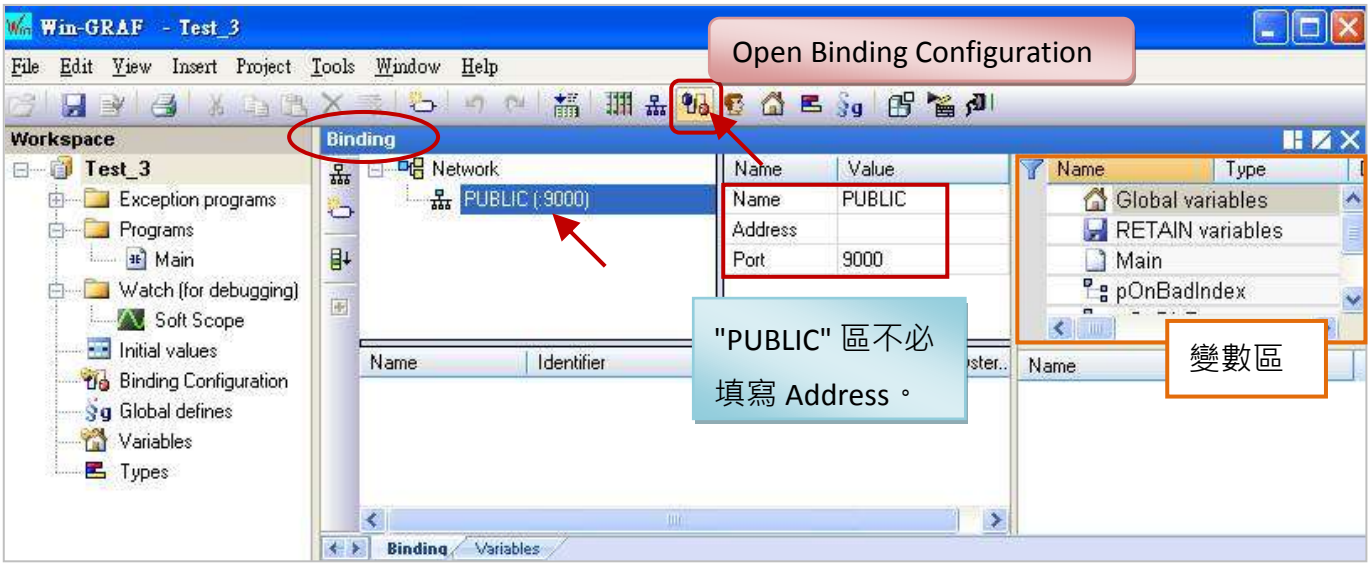
首先，滑鼠點選工具列 “File” -> “Add Exiting Project” -> “FromZip…”，並選擇想要回存的 Win-GRAF 專案 (zip 檔)，即可回存該專案。之後，請執行編譯並將專案下載到 PAC 內。



13.2 如何開放 Win-GRAF 變數給 .NET 程式使用?

Win-GRAF 中，除了字串 (String) 變數之外，其它欲開放的變數需在 "Binding" 設定中產生一個位址才能提供 .NET 程式存取。以下將示範如何開放變數:

1. 滑鼠點選工具列上的 “Open Binding Configuration” 按鈕來開啟 “Binding” 視窗。
2. 點選 "PUBLIC (:9000)" 來設定要公開的資料，您可修改 "Name" 欄位的名稱，"Address" 欄位無需填寫，"Port" 欄位固定為 "9000"，請勿更動。

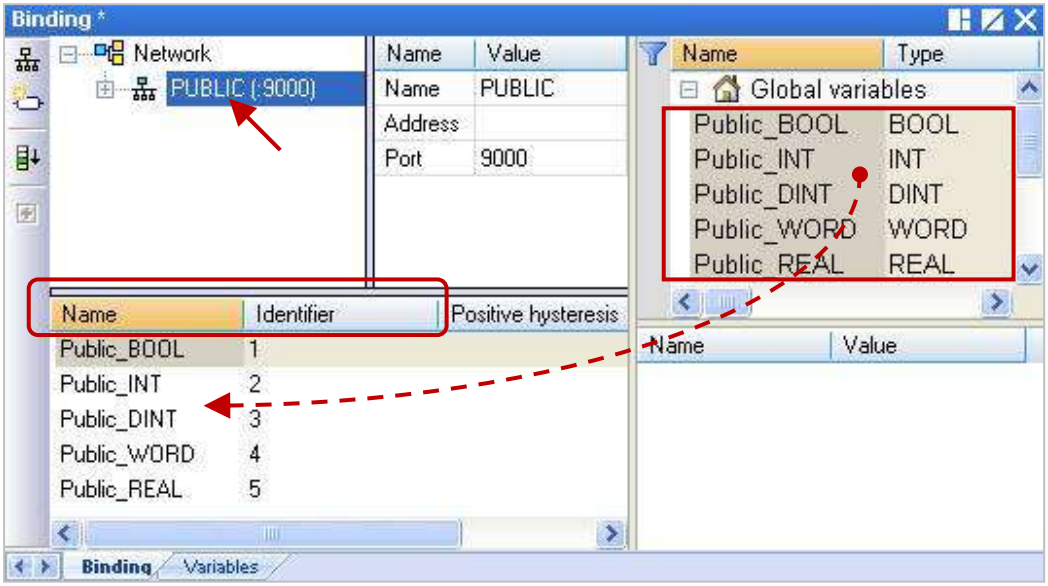


3. 在設定公開資料之前，您必須在變數區先建立好要公開的變數。滑鼠點選 “Global variables” 再按 “Ins” 鍵來新增變數項目，下表為 “Test_3” 範例所使用的變數，您可依實際需求來設定，設定完成後，畫面如下。

變數名稱	資料型態
Public_BOOL	BOOL
Public_INT	INT
Public_DINT	DINT
Public_WORD	WORD
Public_REAL	REAL

4. 如下圖，滑鼠點選在 "PUBLIC(:9000)"，接著選取變數區內欲公開的變數資料，並將其拖曳到 "Name" 區域。"Identifier" 欄位會自動產生編號，若其他 VB 或 .Net 程式想取用變數資料，需設定一樣的 ID 編號 。

注意: "PUBLIC" 最多可使用 8192 個變數，"Identifier" 編號只能是 "1 ~ 8192"。



另外，如需開放 Win-GRAF 的 String 變數，可使用 "Pub_String" 函式。在 Win-GRAF 專案的程式裡，加入此段 ST 語言程式。(您可回存 Win-GRAF demo_vb04 範例程式，來查看以下程式內容)

```
Pub_string(Address, String_val) ;
```

Address: 開放的位址編號，範圍可以是 1 ~ 1024

String_val: String 變數的名稱

變數說明:

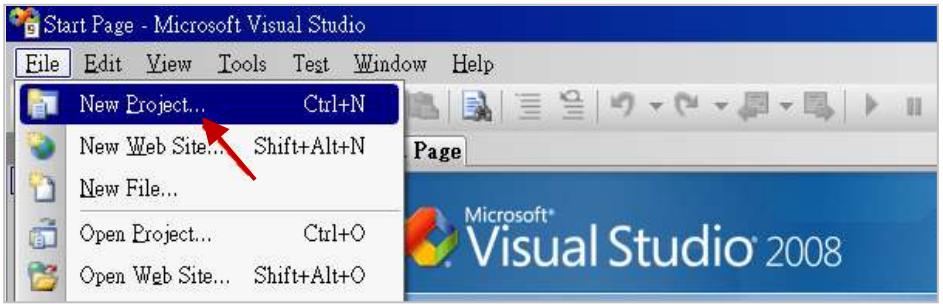
名稱	型態	說明
Init	BOOL	初始化用，初值設定為 True。
Tmp_val	BOOL	判斷開放位址是否成功。 “True” 表示成功，"False" 表示失敗。
msg1	STRING, 字串長度為 100	欲開放的 String 變數。 **注意：字串長度可以是 1 ~ 255。**
msg2	STRING, 字串長度為 32	
msg3	STRING, 字串長度為 60	

程式內容:

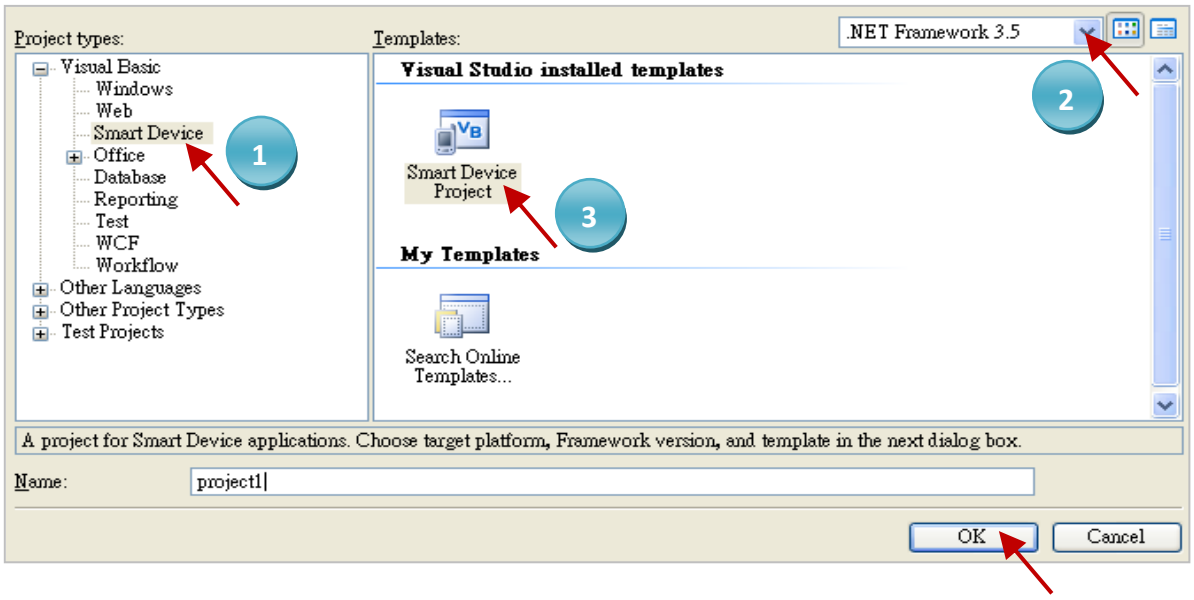
```
If  init  then
  Init := false;
(* 開放位址為 1 的字串值 *)
  Tmp_val := pub_string(1,msg1);

(* 開放位址為 2 的字串值 *)
  Tmp_val := pub_string(2,msg2);

(* 開放位址為 3 的字串值 *)
  Tmp_val := pub_string(3,msg3);

End_if;
```

13.3 建立 VB.NET 新專案

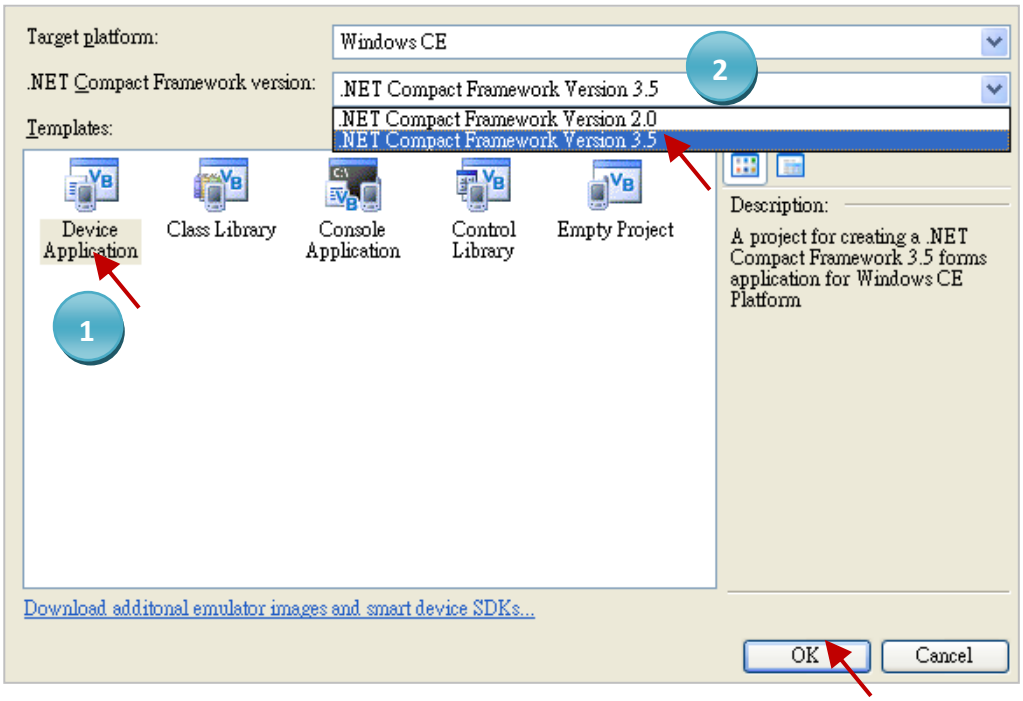
1. 開啟微軟 Visual Studio .NET 2008 軟體，點選 [File] > [New Project]。



2. 點選 [Smart Device] > [.NET frame work 3.5] > [Smart Device Project]，在下方輸入專案名稱 (本例: project1)，然後按 “OK”。



3. 點選 [Device Application] > [Windows CE] > [.NET Compact Framework Version 3.5]，然後按 “OK”。

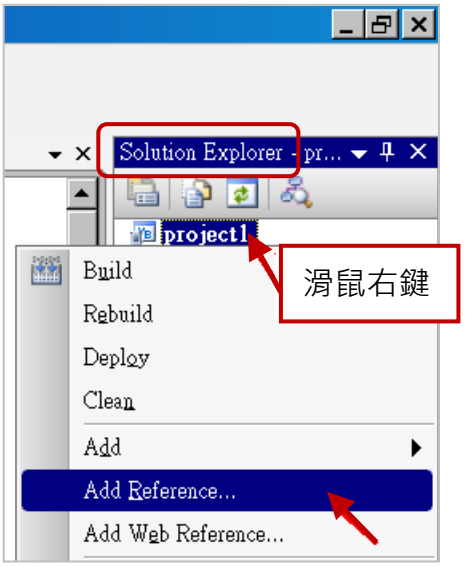


13.3.1 加入專案參考 (Project Reference)

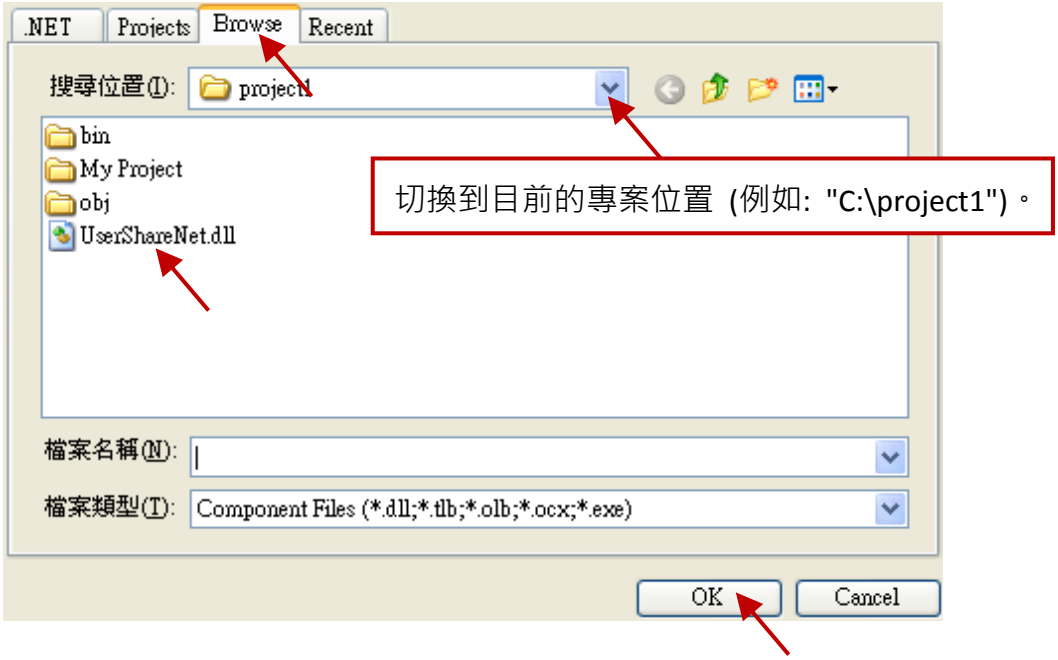
“UserShareNet” 程式庫 (Library) 中包含了所有要跟 Win-GRAF 專案交換資料用的函式 (Function)，而在程式裡使用 “UserShare” 關鍵字之前，您必須在應用程式的參考清單裡加入參考: “UserShareNet.dll”。

1. 請於出貨光碟中 (\napdos\Win-GRAF\demo-project \demo_vb01\vb01\)，將 “UserShareNet.dll” 檔複製到目前專案的位置底下(例如: "C:\project1")。

2. 滑鼠右鍵點擊 “Solution Explorer” 視窗中的專案名稱 (例如: "project1")，再選擇 “Add Reference …”。

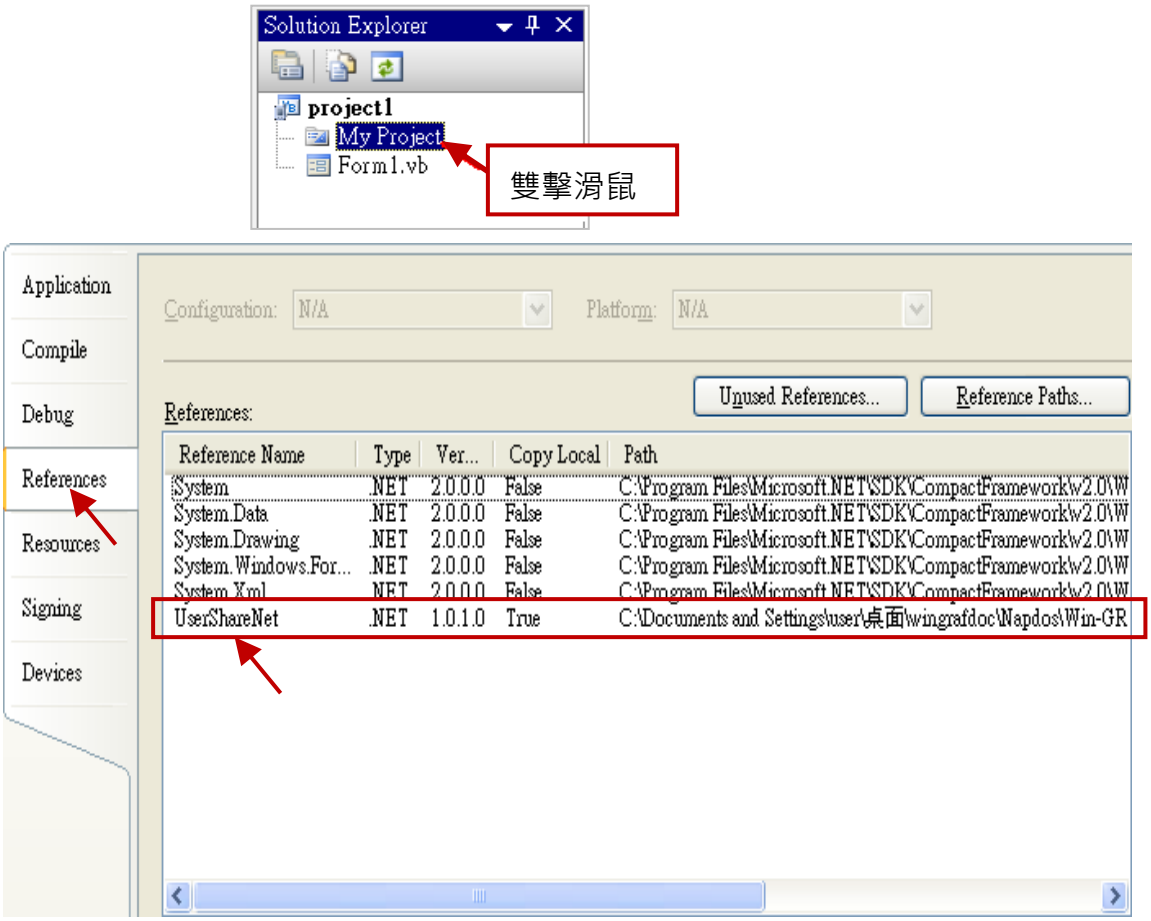


3. 點選 “Browse” 頁籤，並切換到目前的專案位置 (例如: "C:\project1")，再選擇 “UserShareNet.dll”，並按 “OK” 即完成了。

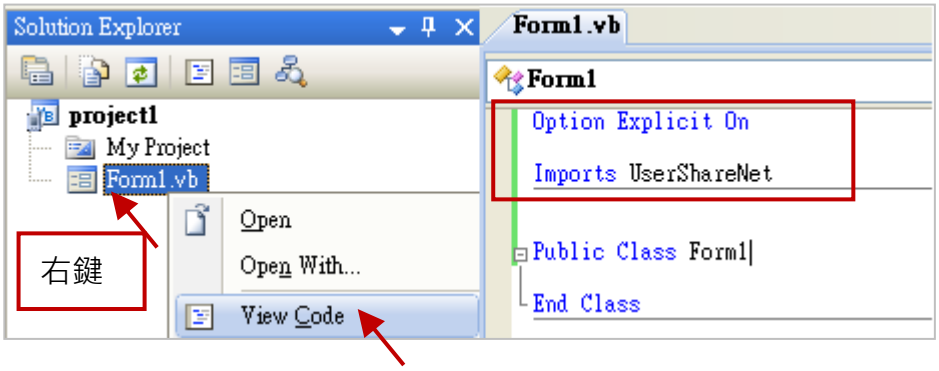


註: 加入 “UserShareNet.dll” 後，下次需開發新專案時，即可從先前開發的專案目錄中 (例如: "C:\project1")，將此檔案複製並加入到新的專案中，或是，可預先從光碟中複製 “UserShareNet.dll” 到固定路徑中 (例如: "C:\dll_lib")。

4. 加入 “UserShareNet.dll” 後，請雙擊專案的 “My Project”，確認是否已加入 “UserShareNet.dll”。



5. 以滑鼠右鍵點選 “Form1.vb”，選擇 “View Code”，並在第一、二行插入 “Option Explicit On” 與 “Imports UserShareNet” (如下圖)。

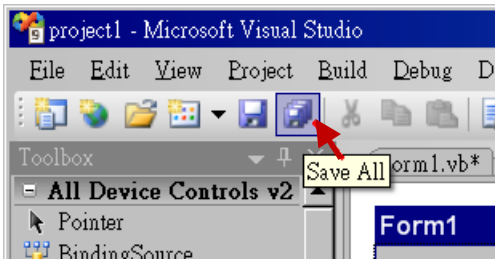


接著，就可以在您的 VB 表格中設計所需的物件與動作了。請參考 13.5 節 的說明來使用 “UserShareNet.dll" 內的函式，來讀/寫 Win-GRAF 專案內的變數資料。

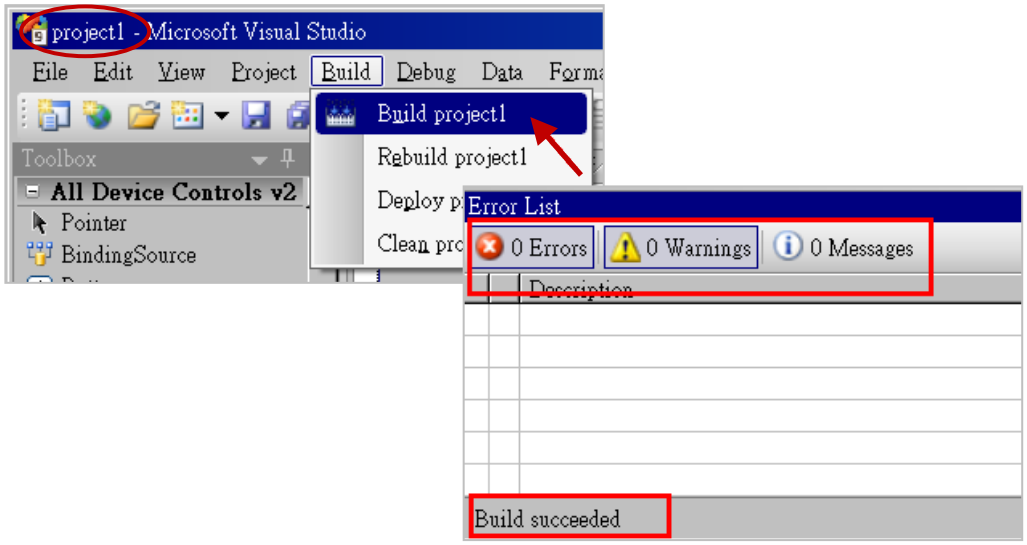
13.4 編譯應用程式

程式編寫完成後，請依照以下步驟來編譯 (Build) 並產生一個執行檔 (.exe)。

1. 請記得隨時按下 “Save All” 工具按鈕來存檔。



2. 點選 “Build” > “Build project1” 來編譯此專案 (project1)，於 “Error List” 視窗中會顯示編譯結果。



3. 編譯完成後，執行檔會存放在以下目錄中。

<您的 VB.net 專案資料夾> \bin\Release\ **<project_name>.exe**

請將此執行檔 (例如: “project1.exe”) 複製到 PAC 的 \System_Disk\Win-GRAF\ 目錄下來執行。

注意:

使用者可複製 VB.net 執行檔到其他目錄下執行，但請記得同時複製相關的 DLL 檔案，否則執行會有錯誤。例如: 要在 \Micro_SD\ 目錄下執行 "project1.exe"，該目錄必須有以下 3 個檔案，即 “project1.exe”、“UserShareNet.dll”、“Quicker.dll” 檔。

(可在光碟的 “\System_disk\Win-GRAF\” 目錄下取得 “UserShareNet.dll” 與 “Quicker.dll” 檔案)

13.5 "UserShareNet.DLL" 內的函式說明

本節著重於 "UserShareNet.DLL" 內的函式 (Function) 來進行說明。

"UserShareNet.DLL" 提供了許多函式，可用來讀/寫 Win-GRAF 專案裡的變數，以下分為幾類：

1. 讀/寫 Boolean
2. 讀/寫 8-bit 整數
3. 讀/寫 16-bit 整數
4. 讀/寫 32-bit 整數
5. 讀/寫 64-bit 整數
6. 讀/寫 32-bit 實數
7. 讀/寫 64-bit 實數

※請參考附錄 A 來查看 Win-GRAF 變數的資料型態與範圍

13.5.1 讀/寫 Boolean 的函式

■ Set_BOOL

說明:

設定指定位址編號的 Boolean 變數值。

語法:

```
UserShare.Set_BOOL ( iUserAddress  As System.UInt16 , ByVal iStatus As byte) as Byte
```

參數:

iUserAddress : 指定變數的位址編號。(1 ~ 8192)

iStatus : 設定變數的狀態；例如: iStatus = 1 表示 "True"，iStatus = 0 表示 "False"

範例:

‘ 設定位址 “1” 的 Win-GRAF Boolean 變數為 "True".
UserShare.Set_BOOL(Convert.ToUInt16(1), 1)

範例程式:

光碟: \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb01

■ Get_BOOL

說明:

讀出指定位址編號的 Boolean 變數值。

語法:

UserShare.Get_BOOL (iUserAddress As System.UInt16 , ByRef iStatus As byte)

參數:

iUserAddress : 指定變數的位址編號 (1 ~ 8192)

iStatus : 取得變數的狀態；iStatus = 1 表示 "True"，iStatus = 0 表示 "False"

範例:

‘ 取得 Win-GRAF Boolean 位址編號 “1” 的變數狀態.

```
Dim iStatus As Byte
UserShare.Get_BOOL(Convert.ToUInt16(1), iStatus)
```

範例程式 :

光碟: \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb01

13.5.2 讀/寫 整數 的函式

■ **Set_SINT** ■ **Set_INT** ■ **Set_DINT** ■ **Set_LINT**

說明:

設定指定位址編號的 Win-GRAF 8-bit Integer, 16-bit Integer, 32-bit Integer 及 64-bit Integer 變數值。

語法:

UserShare.Set_SINT (ByVal *iUserAddress* As System.UInt16 , ByVal *iStatus* As SByte) As Byte

UserShare.Set_INT (ByVal *iUserAddress* As System.UInt16 , ByVal *iStatus* As Short) As Byte

UserShare.Set_DINT (ByVal *iUserAddress* As System.UInt16 , ByVal *iStatus* As Integer) As Byte

UserShare.Set_LINT (ByVal *iUserAddress* As System.UInt16 , ByVal *iStatus* As long) As Byte

參數:

iUserAddress : 指定變數的位址編號 (1 ~ 8192)

iStatus : 設定 8-bit Integer, 16-bit Integer, 32-bit Integer 及 64-bit Integer 值

範例:

‘設定 32-bit integer 值 “1234567” 到位址編號 “1” 的變數.
UserShare.Set_DINT(Convert.ToUInt16(1), Convert.ToInt32(1234567))

‘ 設定 integer 值 “-1234” 到位址編號 “2” 的變數.
UserShare.Set_INT(Convert.ToUInt16(3), Convert.ToInt16(-1234))

‘ 設定 64-bit integer 值 “123456789012345” 到位址編號 “3” 的變數.
UserShare.Set_LINT(Convert.ToUInt16(3), Convert.ToInt64(123456789012345))

‘ 設定 8-bit integer 值 “125” 到位址編號 “4” 的變數.
UserShare.Set_SINT(Convert.ToUInt16(3), Convert.ToSByte(125))

範例程式:

光碟 :

1. 讀/寫 類比 I/O: \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb02
2. 讀/寫 Internal long integer、Timer 及 Real (浮點數) 的值:
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb03

■ Get_SINT ■ Get_INT ■ Get_DINT ■ Get_LINT

說明:

讀出指定位址編號的 Win-GRAF 8-bit Integer, 16-bit Integer, 32-bit Integer 及 64-bit Integer 變數值。

語法:

UserShare. Get_SINT (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As SByte) As Byte

UserShare. Get_INT (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As Short) As Byte

UserShare.Get_DINT (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As Integer) As Byte

UserShare. Get_LINT (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As long) As Byte

參數:

iUserAddress : 指定變數的位址編號 (1 ~ 8192)

iStatus : 取得 8-bit Integer, 16-bit Integer, 32-bit Integer 及 64-bit Integer 值

範例:

```
Dim Dlong_val As Int64
Dim short_val As Int16
Dim long_val As Int32
Dim Sbyte_val as sbyte

‘ 取得位址編號 “7” 的 64-bit 整數的變數值.
UserShare.Get_LINT(Convert.ToUInt16(7), Dlong_val)

‘取得位址編號 “8” 的 32-bit 整數的變數值.
UserShare.Get_DINT(Convert.ToUInt16(8), long_val)

‘取得位址編號 “9” 的 16-bit 整數的變數值.
UserShare.Get_INT(Convert.ToUInt16(9), short_val)

‘取得位址編號 “10” 的 8-bit 整數的變數值.
UserShare.Get_SINT(Convert.ToUInt16(9), sbyte_val)
```

範例程式:

光碟:

1. 讀/寫 類比 I/O:
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo \demo_vb02
2. 讀/寫 Internal long integer、Timer 及 Real (浮點數) 的值:
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb03

13.5.3 讀/寫 實數 的函式

■ Get_REAL ■ Get_LREAL

說明:

讀出指定位址編號的 Win-GRAF 32-bit Real、64-bit Real 變數值。

語法:

UserShare. Get_REAL (ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As Single) As Byte

UserShare. Get_LREAL(ByVal *iUserAddress* As System.UInt16 , ByRef *iStatus* As Double) As Byte

參數:

iUserAddress : 指定變數的位址編號 (1 ~ 8192)

iStatus : 取得浮點數數值

範例:

```
Dim float_val As Single
Dim double_val As Double

‘ 取得位址編號 “7” 的 double 變數值.
UserShare.Get_LREAL(Convert.ToUInt16(7), double_val)

‘取得位址編號 “8” 的 Single 變數值.
UserShare.Get_REAL(Convert.ToUInt16(8), float_val)
```

範例程式:

光碟:

1. 讀/寫 類比 I/O:
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb02
2. 讀/寫 Internal long integer、Timer 及 Real (浮點數) 的值:
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb03

■ Set_REAL ■ Set_LREAL

說明:

寫入指定位址編號的 Win-GRAF 32-bit Real、64-bit Real 變數值。

語法:

UserShare. Set_REAL (ByVal *iUserAddress* As System.UInt16, ByVal *iStatus* As Single) As Byte

UserShare. Set_LREAL(ByVal *iUserAddress* As System.UInt16, ByVal *iStatus* As Double) As Byte

參數:

iUserAddress : 指定變數的位址編號 (1 ~ 8192)

iStatus : 設定浮點數數值

範例:

‘ 寫入 11234.234567 到位址編號 ”7” 的變數

UserShare.Set_LREAL(Convert.ToUInt16(7),Convert.ToDouble(11234.234567))

‘寫入 123.12 到位址編號 ”8” 的變數

UserShare.Set_REAL(Convert.ToUInt16(8), Convert.ToSingle (123.12))

範例程式:

光碟:

1. 讀/寫 類比 I/O: \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb02
2. 讀/寫 Internal long integer、Timer 及 Real (浮點數) 的值: \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb03

13.5.4 讀/寫 字串 的函式

■ Get_STRING

說明:

讀取指定位址編號的 Win-GRAF 字串變數值。

語法:

UserShare. Get_STRING (ByVal *iUserAddress* As System.UInt16, ByVal *msg()* As Byte) As Byte

參數:

iUserAddress : 指定變數的位址編號 (1 ~ 1024)

msg() : 取得/設定 Win-GRAF 的字串值

範例:

```
Dim str_val As String
Dim msg() As Byte
‘ 取得位址編號 “7” 的 String 變數值.
UserShare.Get_STRING(Convert.ToUInt16(7),msg )
str_val= byte_array_to_unicode(msg)

 Private Function byte_array_to_unicode(ByVal buf() As Byte) As String
     Dim tmpmsg As String
     If buf.Length > 255 Then
        Return Nothing
     End If
tmpmsg = System.Text.Encoding.GetEncoding("UTF-8").GetString(buf, 0,  buf.Length)

     Return tmpmsg
  End Function
```

範例程式:

光碟:

1. 讀/寫 STRING 的值:
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb04

■ Set_STRING

說明:

寫入指定位址編號的 Win-GRAF 字串值。

語法:

UserShare. Set_STRING (ByVal *iUserAddress* As System.UInt16, ByVal *msg()* As Byte) As Byte

參數:

iUserAddress : 指定變數的位址編號 (1 ~ 1024)

msg() : 設定 Win-GRAF 的字串值

範例:

```
Dim str_val As String=”Hellow World”
Dim msg() As Byte

msg= unicode_to_byte_array(str_val)
‘ 寫入位址編號 “7” 的 String 變數值.
UserShare.Set_STRING(Convert.ToUInt16(7),msg )

‘轉換字串成 byte array
Private Function unicode_to_byte_array(ByVal msg As String) As Byte()
    Dim tmpbuf() As Byte
    If msg.Length > 255 Then
       Return Nothing
    End If
    tmpbuf = System.Text.Encoding.GetEncoding("UTF-8").GetBytes(msg)
    Return tmpbuf

End_Function
```

範例程式:

光碟:

1. 讀/寫 STRING 的值:
   \napdos\Win-GRAF\demo-project\vb.net_2008_demo\demo_vb04

13.5.5 如何讓 VB.NET 程式讀取 Win-GRAF 字串變數?

.NET 應用程式如果要寫入字串變數，必須根據目前使用的語言編碼 (例如: UTF-8) 轉換成 byte 陣列才能寫入，若是讀出字串變數的陣列內容，則需要根據語言編碼轉換成字串。以下提供 VB.NET 程式轉換的範例

(編碼為 UTF-8)：

String 轉成 byte 陣列

```
  Private Function unicode_to_byte_array(ByVal msg As String) As Byte()
    Dim tmpbuf() As Byte
    If msg.Length > 255 Then
      Return Nothing
    End If

    tmpbuf = System.Text.Encoding.GetEncoding("UTF-8").GetBytes(msg)
    Return tmpbuf
  End Function
```

byte 陣列轉字串

```
  Private Function byte_array_to_unicode(ByVal buf() As Byte) As String
    Dim tmpmsg As String
    If buf.Length > 255 Then
      Return Nothing
    End If
tmpmsg = System.Text.Encoding.GetEncoding("UTF-8").GetString(buf, 0, buf.Length)

    Return tmpmsg
  End Function
```

第 14 章 使用 C# 程式來 讀/寫 Win-GRAF 變數

本章以 Visual Studio .NET 2008 開發工具建立一個範例程式的方式來說明，範例程式可以在 XP-8xx8-CE6 , WP-8xx8, VP-25W8, VP-4138, WP-5xx8 產品包裝盒內附的 CD-ROM 內找到。

C# 範例:

光碟 : \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\

demo_CSharp01： 數位 I/O 範例，搭配 I-87055W 模組 (於 Slot 0)

demo_CSharp02： 類比 I/O 範例，搭配 I-87024W (Slot 1) 與 I-8017HW (Slot 2) 模組

demo_CSharp03： 讀/寫 Win-GRAF Internal Integer, Timer, 及 Real 變數 (無需 I/O 模組)

demo_CSharp04： 讀/寫 Win-GRAF String 變數 (無需 I/O 模組)

Win-GRAF 範例:

光碟 : \napdos\Win-GRAF\demo-project\

"demo_vb01.zip", "demo_vb02.zip", "demo_vb03.zip", "demo_vb04.zip"

14.1 如何回存 Win-GRAF 專案?

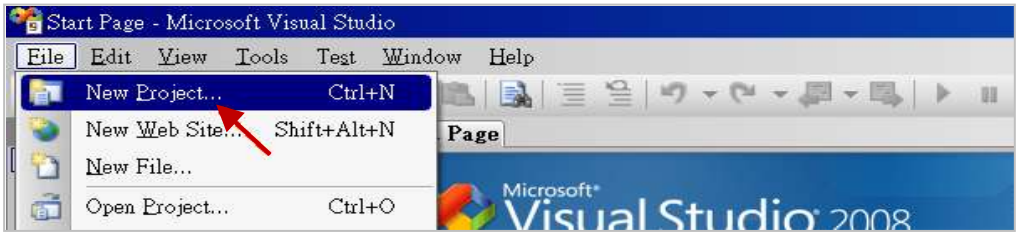
請參考 13.1 節 來回存 Win-GRAF 專案。

14.2 如何開放 Win-GRAF 變數給 C# 程式使用?

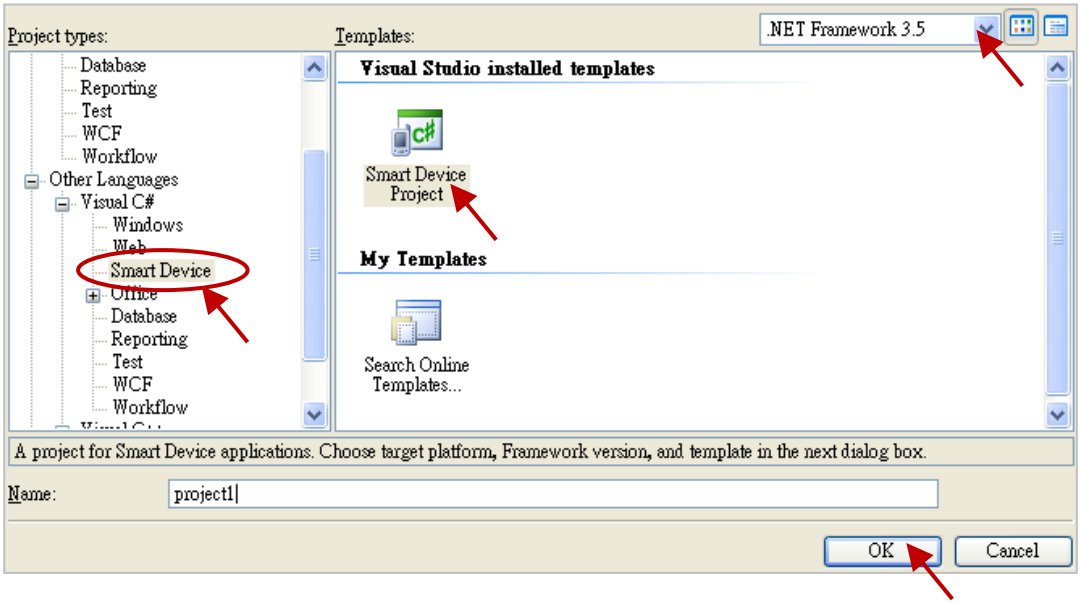
請參考 13.2 節 - 開放 Win-GRAF 變數給 .NET 程式使用的方式。

14.3 建立 C# 新專案

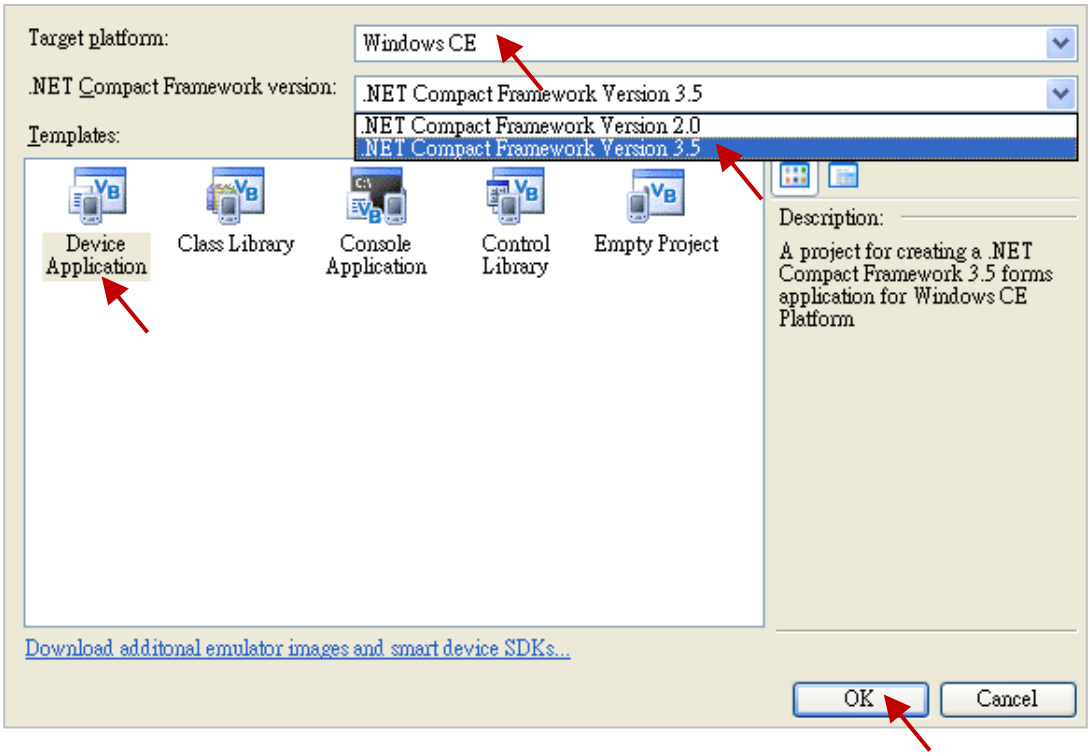
1. 開啟微軟 Visual Studio .NET 2008，點選功能表 [File] > [New Project]。



2. 點選 [Smart Device] > [.NET frame work 3.5] > [Smart Device Project]，並輸入專案名稱 (例如: "project1")，然後點選 "OK"。



3. 點選 [Device Application] > [Windows CE] > [.NET Compact Framework Version 3.5]，然後點選 "OK"。

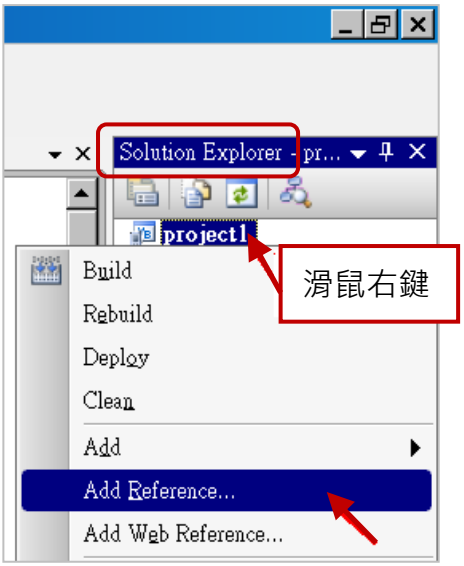


14.3.1 加入 C# 專案參考

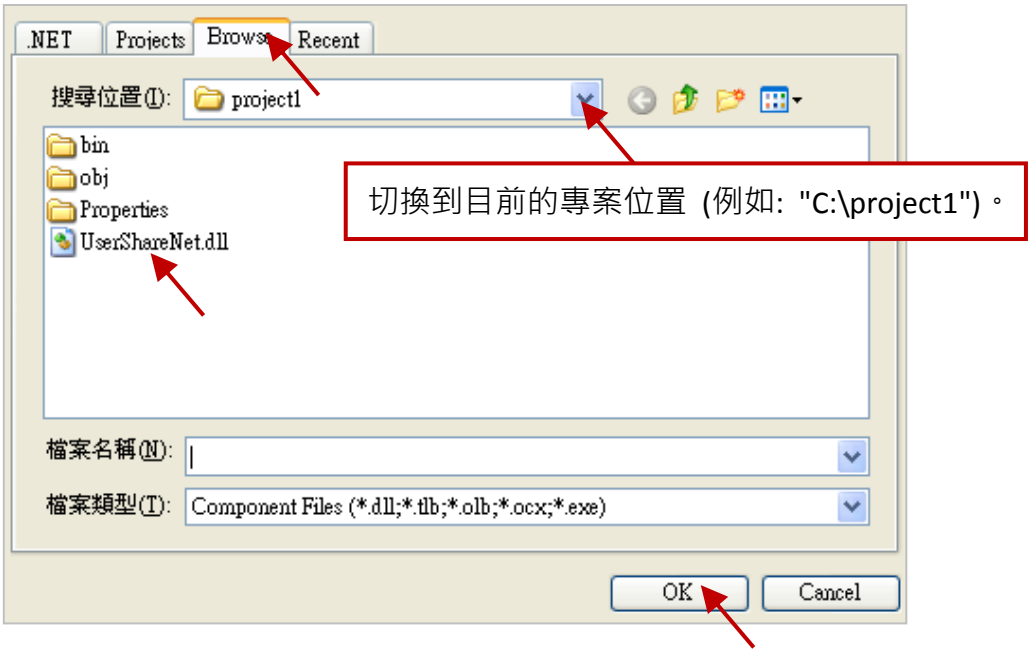
“UserShareNet” 程式庫 (Library) 中包含了所有要跟 Win-GRAF 專案交換資料用的函式，而在程式裡使用 “UserShare” 關鍵字之前，您必須在應用程式的參考清單裡加入參考: “UserShareNet.dll”。

1. 請於出貨光碟中 (\napdos\Win-GRAF\demo-project \demo_CSharp01\demo_CSharp01\)，將 “UserShareNet.dll” 檔複製到目前專案的位置底下(例如: "C:\project1")。

2. 滑鼠右鍵點選 “Solution Explorer” 視窗中的專案名稱 (例如: "project1")，再選擇 “Add Reference …”。

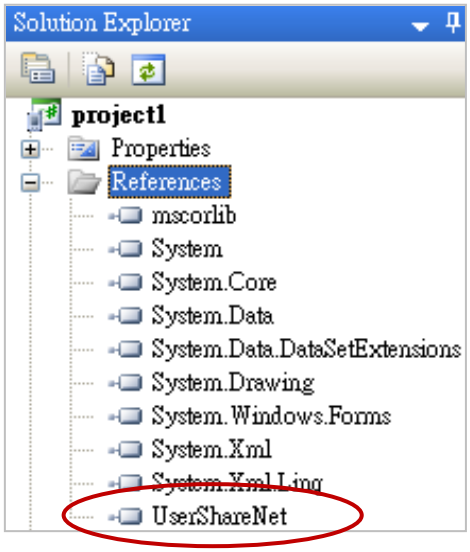


3. 點選 “Browse” 頁籤，並切換到目前的專案位置 (例如: "C:\project1")，再選擇 “UserShareNet.dll”，並按 “OK” 即完成了。

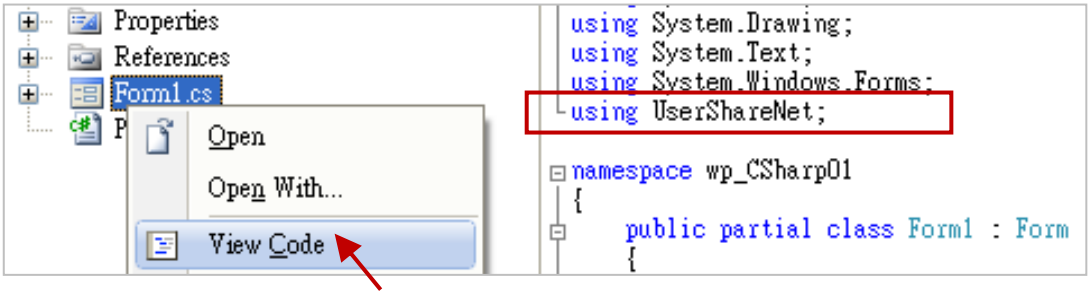


註: 加入 “UserShareNet.dll” 後，下次需開發新專案時，即可從先前開發的專案目錄中 (例如: "C:\project1")，將此檔案複製並加入到新的專案中。或是，可預先從光碟中複製 “UserShareNet.dll” 到固定路徑中 (例如: "C:\dll_lib")。

4. 加入 “UserShareNet.dll” ，會出現在 “Solution Explorer” 視窗。



5. 以滑鼠右鍵點選 “Form1.cs”，選擇 “View Code”，將游標移到最上方，在第一個區段中加入 “using UserShareNet” 。

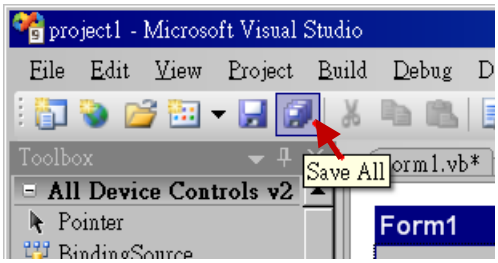


接著，就可以在您的 C# 表單中設計所需的物件與動作了。請參考 14.5 節 的說明來使用 “UserShareNet.dll” 內的函式，來讀/寫 Win-GRAF 內的變數資料。

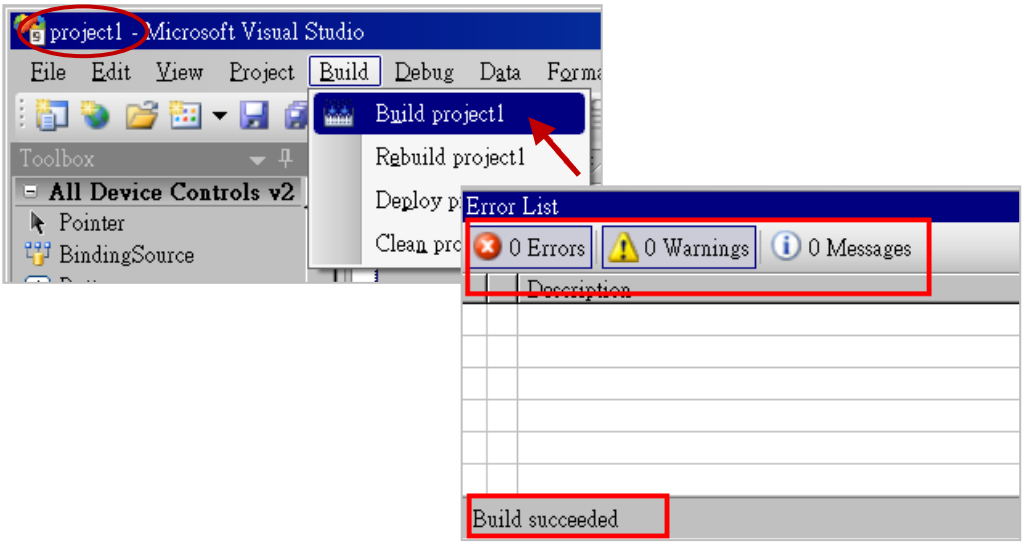
14.4 編譯應用程式

程式編寫完成後，請依照以下步驟來編譯 (Build) 並產生一個執行檔 (.exe)。

1. 請記得隨時按下 “Save All” 工具按鈕來存檔。



2. 點選 “Build” > “Build project1” 來編譯此專案 (project1)，於 “Error List” 視窗中會顯示編譯結果。



3. 編譯完成後，執行檔會存放在以下目錄中。

<您的 C# .net 專案資料夾> \bin\Release\ **<project_name>.exe**

請將此執行檔 (例如: “project1.exe”) 複製到 PAC 的 \System_Disk\Win-GRAF\ 目錄下來執行。

注意:

使用者可複製 C# .net 執行檔到其他目錄下執行，但請記得同時複製相關的 DLL 檔案，否則執行會有錯誤。例如: 要在 \Micro_SD\ 目錄下執行 "project1.exe"，該目錄必須有以下 3 個檔案，即 “project1.exe”、“UserShareNet.dll”、“Quicker.dll” 檔。

(可在光碟的 “\System_disk\Win-GRAF\” 目錄下取得 “UserShareNet.dll” 與 “Quicker.dll” 檔案)

14.5 "UserShareNET.DLL" 內的函式說明

本節將針對 "UserShareNet.DLL" 內的函式 (Function) 來進行說明。

"UserShareNet.DLL" 提供了許多函式，可用來讀/寫 Win-GRAF 專案裡的變數，以下分為幾類：

1. 讀/寫 Boolean
2. 讀/寫 8-bit 整數
3. 讀/寫 16-bit 整數
4. 讀/寫 32-bit 整數
5. 讀/寫 64-bit 整數
6. 讀/寫 32-bit 實數
7. 讀/寫 64-bit 實數

※請參考附錄 A 來查看 Win-GRAF 的變數資料型態與範圍

14.5.1 讀/寫 Boolean 的函式

■ Set_BOOL

說明:

設定指定位址編號的 Win-GARF Boolean 變數值。

語法:

```
UserShare.Set_BOOL(ushort iUserAddress, byte iStatus)
```

參數:

iUserAddress : 指定變數的位址編號 (1 ~ 8192)

iStatus : 設定變數的狀態；例如: iStatus = 1 表示 "True"，iStatus = 0 表示 "False"。

範例:

//設定位址 ”1” 的 Win-GRAF 變數為 True.

```
UserShare.Set_BOOL(Convert.ToUInt16(1), 1);
```

範例程式:

光碟 : \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp01

■ Get_BOOL

說明:

讀出指定位址編號的 Win-GARF Boolean 變數值。

語法:

UserShare.Get_BOOL(ushort iUserAddress, out byte iStatus)

參數:

iUserAddress : 指定變數位址編號 (1 ~ 8192)

iStatus : 取得變數的狀態；iStatus = 1 表示 "True"，iStatus = 0 表示 "False"。

範例:

```
Byte iStatus=0;
//取得位址編號 “1” 的變數狀態.
UserShare.Get_BOOL(Convert.ToUInt16(1),out iStatus);
```

範例程式:

光碟: \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp01

14.5.2 讀/寫 整數 的函式

■ Set_SINT ■ Set_INT ■ Set_DINT ■ Set_LINT

說明:

設定指定位址編號的 Win-GRAF 8-bit Integer, 16-bit Integer, 32-bit Integer 及 64-bit Integer 變數值。

語法:

```
UserShare.Set_SINT(ushort iUserAddress , sbyte iStatus)

UserShare.Set_INT(ushort iUserAddress , short iStatus)

UserShare.Set_DINT(ushort iUserAddress, int iStatus)

UserShare.Set_LINT(ushort iUserAddress, long iStatus)
```

參數:

iUserAddress : 指定變數的位址編號 (1 ~ 8192)

iStatus : 設定 8-bit Integer, 16-bit Integer, 32-bit Integer 及 64-bit Integer 值

範例:

```
//設定 32-bit integer 值 “1234567” 到位址編號 “1” 的變數.
int temp1=1234567;
UserShare.Set_DINT(Convert.ToUInt16(1), temp );

//設定 16-bit Integer 值 “-1234” 到 Modbus 位址編號 “2” 的變數.
short temp2= -1234;
UserShare.Set_INT(Convert.ToUInt16(2), temp2 );

//設定 64-bit Integer 值 “123456789012345” 到位址編號 “3” 的變數.
long temp3=123456789012345;
UserShare.Set_LINT(Convert.ToUInt16(3),  temp3 );

//設定 8-bit Integer “125” 值到位址編號 ”4” 的變數.
Sbyte temp4=125;
UserShare.Set_SINT(Convert.ToUInt16(4),  temp4 );
```

範例程式:

光碟:

1. 讀/寫 類比 I/O: \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp02
2. 讀/寫 Internal long integer、Timer 及 Real (浮點數) 的值:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp03

■ Get_SINT ■ Get_INT ■ Get_DINT ■ Get_LINT

說明:

讀出指定位址編號的 Win-GRAF 8-bit Integer, 16-bit Integer, 32-bit Integer 及 64-bit Integer 變數值。

語法:

UserShare.Get_SINT(ushort iUserAddress, out sbyte iStatus)

UserShare.Get_INT(ushort iUserAddress, out short iStatus)

UserShare.Get_DINT(ushort iUserAddress, out int iStatus)

UserShare.Get_LINT(ushort iUserAddress, out long iStatus)

參數:

iUserAddress : 指定變數的位址編號 (1 ~ 8192)

iStatus : 取得 8-bit Integer, 16-bit Integer, 32-bit Integer 及 64-bit Integer 值

範例:

```
Int64 Dlong_val;
Int16 short_val;
Int32 long_val ;
sbyte sbyte_val;

//取得位址編號 “7” 的 64-bit 整數的變數值.
UserShare.Get_LINT(Convert.ToUInt16(7),out Dlong_val);

//取得位址編號 “8” 的 32-bit 整數的變數值.
UserShare.Get_DINT(Convert.ToUInt16(8),out long_val);

//取得位址編號 “9” 的 16-bit 整數的變數值.
UserShare.Get_INT(Convert.ToUInt16(9),out short_val);

//取得位址編號 “10” 的 8-bit 整數的變數值.
UserShare.Get_SINT(Convert.ToUInt16(9),out sbyte_val)
```

範例程式:

光碟:

1. 讀/寫 類比 I/O:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp02
2. 讀/寫 Internal long integer、Timer 及 Real (浮點數) 的值:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp03

14.5.3 讀/寫 實數 的函式

■ Get_REAL ■ Get_LREAL

說明:

讀出指定位址編號的 Win-GRAF 32-bit Real、64-bit Real 變數值。

語法:

> **UserShare. Get_REAL (System.UInt16 *iUserAddress*, out float *iStatus*)**
>
> **UserShare. Get_LREAL(ByVal *iUserAddress* As System.UInt16 , out Double *iStatus*)**

參數:

iUserAddress : 指定變數的位址編號 (1 ~ 8192)

iStatus : 取得浮點數數值

範例:

```
float float_val;
double double_val;

// 取得位址編號 “7” 的 double 變數值.
UserShare.Get_LREAL(Convert.ToUInt16(7),out double_val);

//取得位址編號 “8” 的 float 變數值.
UserShare.Get_REAL(Convert.ToUInt16(8),out float_val);
```

範例程式:

光碟:

1. 讀/寫 類比 I/O:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp02
2. 讀/寫 Internal long integer、Timer 及 Real (浮點數) 的值:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_ demo_CSharp03

■ Set_REAL ■ Set_LREAL

說明:

寫入指定的位址編號的 Win-GRAF 32-bit Real、64-bit Real 變數值。

語法:

UserShare. Set_REAL (ushort *iUserAddress* , float *iStatus*)

UserShare. Set_LREAL(ushort *iUserAddress* , Double *iStatus*)

參數:

iUserAddress : 指定變數的位址編號 (1 ~ 8192)

iStatus : 設定浮點數數值

範例:

```
// 寫入 11234.234567 到位址編號 ”7” 的變數
UserShare.Set_LREAL(Convert.ToUInt16(7),Convert.ToDouble(11234.234567));

//寫入 123.12 到位址編號 ”8” 的變數
UserShare.Set_REAL(Convert.ToUInt16(8), Convert.ToSingle (123.12));
```

範例程式:

光碟:

1. 讀/寫 類比 I/O:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp02
2. 讀/寫 internal long integer、Timer 及 Real (浮點數) 的值:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp03

14.5.4 讀/寫 字串 的函式

■ Set_STRING

說明:

寫入指定位址編號的 Win-GRAF 字串值。

語法:

UserShare.Set_STRING (ushort addr , Byte [] msg)

參數:

addr : 指定變數的位址編號 (1 ~ 1024)

msg[] : 設定 Win-GRAF String 變數值。

範例:

```
String str_val;
Byte[] msg;

// 寫入位址編號 “7” 的 String 變數值.
msg= unicode_to_byte_array(str_val);
UserShare.Set_STRING(Convert.ToUInt16(7),msg );

//String 轉成 byte 陣列
private byte[] unicode_to_byte_array(string msg)
    {
      byte[] tmpbuf;
      if (msg.Length > 255)
        return null;

      tmpbuf = Encoding.GetEncoding("UTF-8").GetBytes(msg);
      return tmpbuf;
    }
```

範例程式:

光碟:

1. 讀/寫 STRING 的值:
   \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp04

■ Get_STRING

說明:

讀取指定位址編號的 Win-GRAF 字串變數值。

語法:

UserShare.Set_STRING (ushort addr , Byte [] msg)

參數:

addr : 指定變數的位址編號 (1 ~ 1024)

msg[] : 設定 Win-GRAF String 變數值。

範例:

```
String str_val= “Hello World”;
Byte[] msg;

// 設定位址編號 “7” 的 String 變數值.
UserShare.Get_STRING(Convert.ToUInt16(7),msg );
str_val= byte_array_to_unicode(msg);



//byte 陣列轉字串
private string byte_array_to_unicode(byte[] buf)
 {
        string tmpmsg;
        if (buf.Length > 255)
           return null;

        tmpmsg = Encoding.GetEncoding("UTF-8").GetString(buf, 0, buf.Length);
        return tmpmsg;
  }
```

範例程式:

光碟:

1. 讀/寫 STRING 的值: \napdos\Win-GRAF\demo-project\CSharp.net_2008_demo\demo_CSharp04

14.5.5 如何讓 C # 程式讀取 Win-GRAF 字串變數?

.NET 程式如果要寫入字串變數，必須根據目前使用的語言編碼 (例如: UTF-8) 轉換成 byte 陣列才能寫入，若是讀出字串變數的陣列內容，則需要根據語言編碼轉換成字串。以下提供 C# 程式轉換的範例 (編碼為 UTF-8)：

//String 轉成 byte 陣列

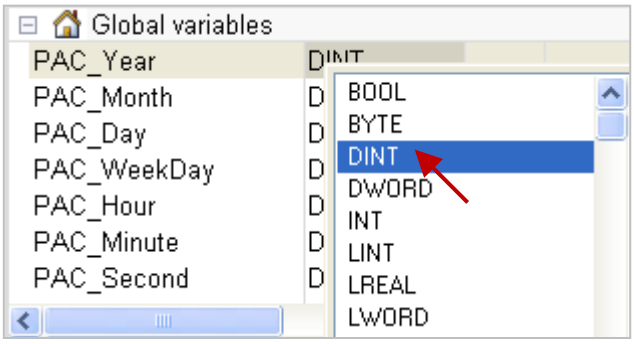
```
private byte[] unicode_to_byte_array(string msg)
    {
       byte[] tmpbuf;
       if (msg.Length > 255)
          return null;

       tmpbuf = Encoding.GetEncoding("UTF-8").GetBytes(msg);
       return tmpbuf;
    }
```

//byte 陣列轉字串

```
private string byte_array_to_unicode(byte[] buf)
 {
       string tmpmsg;
       if (buf.Length > 255)
          return null;

       tmpmsg = Encoding.GetEncoding("UTF-8").GetString(buf, 0, buf.Length);
       return tmpmsg;
  }
```

附錄 A　資料型態與數值範圍

您可在程式的變數區 (參考 2.2.1 節) 或 變數視窗 (參考 2.2.2 節) 中設定變數的資料型態。



下表為變數的基本資料型態與數值範圍:

資料型態	位元數	數值範圍
BOOL (*)	---	TRUE，FALSE
SINT	8 bits (Small int, signed)	-128 ~ +127
USINT BYTE	8 bits (Unsigned small int)	0 ~ +255
INT	16 bits (Int, signed)	-32768 ~ +32767
UINT WORD	16 bits (Unsigned int)	0 ~ +65535
DINT (*)	32 bits (Double int, signed)	-2147483648 ~ +2147483647
UDINT DWORD	32 bits (Unsigned double int)	0 ~ +4294967295
LINT	64 bits (Large int, signed)	-2^{63} ~ $+(2^{63}-1)$
ULINT LWORD	64 bits (Unsigned large int)	0 ~ $+(2^{64}-1)$
REAL (*)	32 bits (Floating point)	$\pm3.4\times10^{-38}$ ~ $\pm3.4\times10^{38}$
LREAL	64 bits (Floating point)	$\pm1.7\times10^{-308}$ ~ $\pm1.7\times10^{308}$
STRING (*)	最多 255 個字元	---
TIME (*)	32 bits	T#0ms ~ T#23h59m59s999ms

(*): 表示常用的資料型態

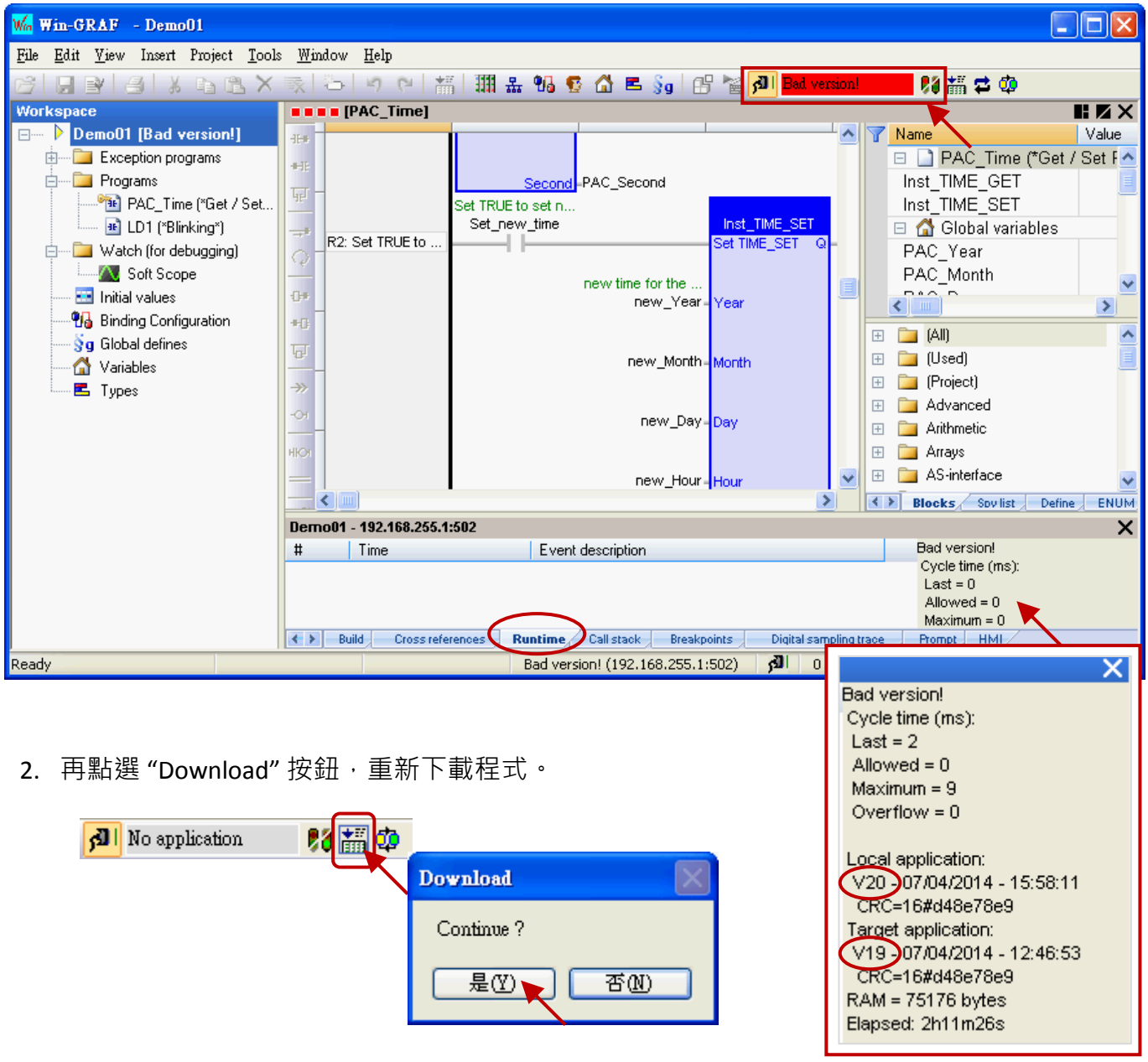
附錄 B　錯誤訊息排除

若 PC 與 PAC 之間的連線出現了錯誤訊息，請參考以下內容來排除問題。

- **若出現 “Bad version!” 錯誤訊息:**

  表示程式有修改並重新編譯過，因此 PC 與 PAC 中的程式 (編譯) 版本不一致。

1. 點選 “Stop application” 按鈕，停止 PAC 中運行的程式。

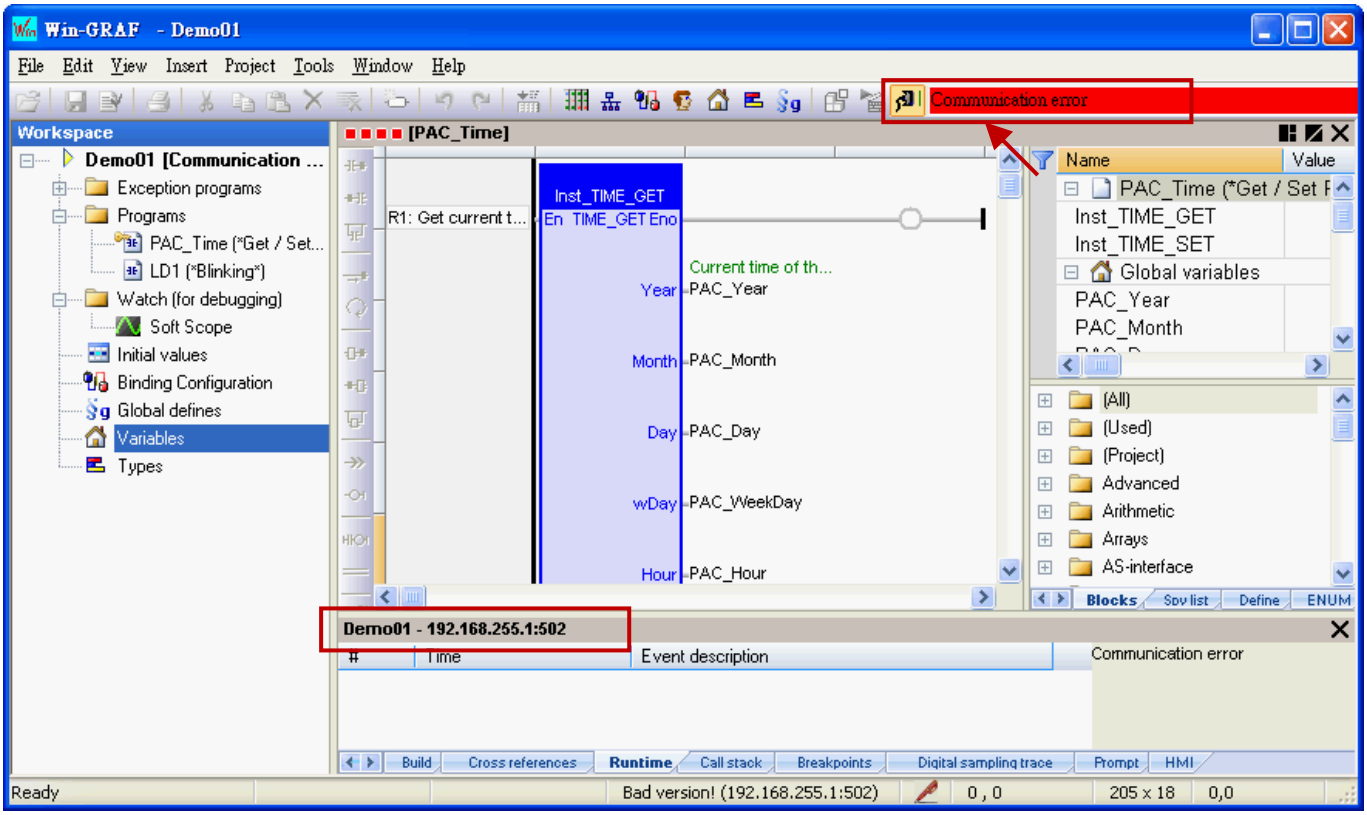


2. 再點選 “Download” 按鈕，重新下載程式。

3. 出現 “RUN” 表示程式正常運行。

- **若出現 “Communication error” 錯誤訊息:**

  表示 PC 與 PAC 之間通訊失敗。



1. 確認您的 Win-GRAF PAC 已經開機且網路通訊正常。
2. 確認 PAC IP 與專案中的設定一致 (參考 2.3.5 節，此例為 “192.168.255.1:502”)。
3. 確認您的 PC 網路通訊正常。

附錄 C　啟動 WinCE PAC 螢幕保護功能

啟動 WinCE PAC 的螢幕保護功能，請設定下列兩個項目。

1. 請執行 **“控制台”** > “電源” > “配置”，在 “電源配置” 項目選擇 “AC 電源”，並將 “使用者閒置” 與 “系統閒置” 設定同樣的值 (或設定 “系統閒置” 的值比 “使用者閒置” 的值大)。
2. 然後，記得執行 “**WinPAC Utility**” > “File” > “Save and Reboot” 存檔及重新啟動。

下方以 WP-8xx8 為例來說明:

如果使用者沒有碰觸螢幕或按鍵，設定的時間到時，WP-8xx8 會關閉背光 來啟動螢幕保護功能。之後只要使用者碰觸螢幕或按鍵，WP-8xx8 就會再次開啟螢幕背光。

若不想使用螢幕保護功能，請設定 “User Idle” 與 “System Idle” 為 “Never”，並記得要執行 “WinPAC Utility” > “File” > “Save and Reboot”，儲存設定並重新啟動。

附錄 D　使用 RS-232/485/422 擴充卡

客戶可以在 Win-GRAF PAC (見 P1-1) 的插槽 (Slot 0 ~ 7) 使用下列模組來擴充 16 個以上的 COM 埠。

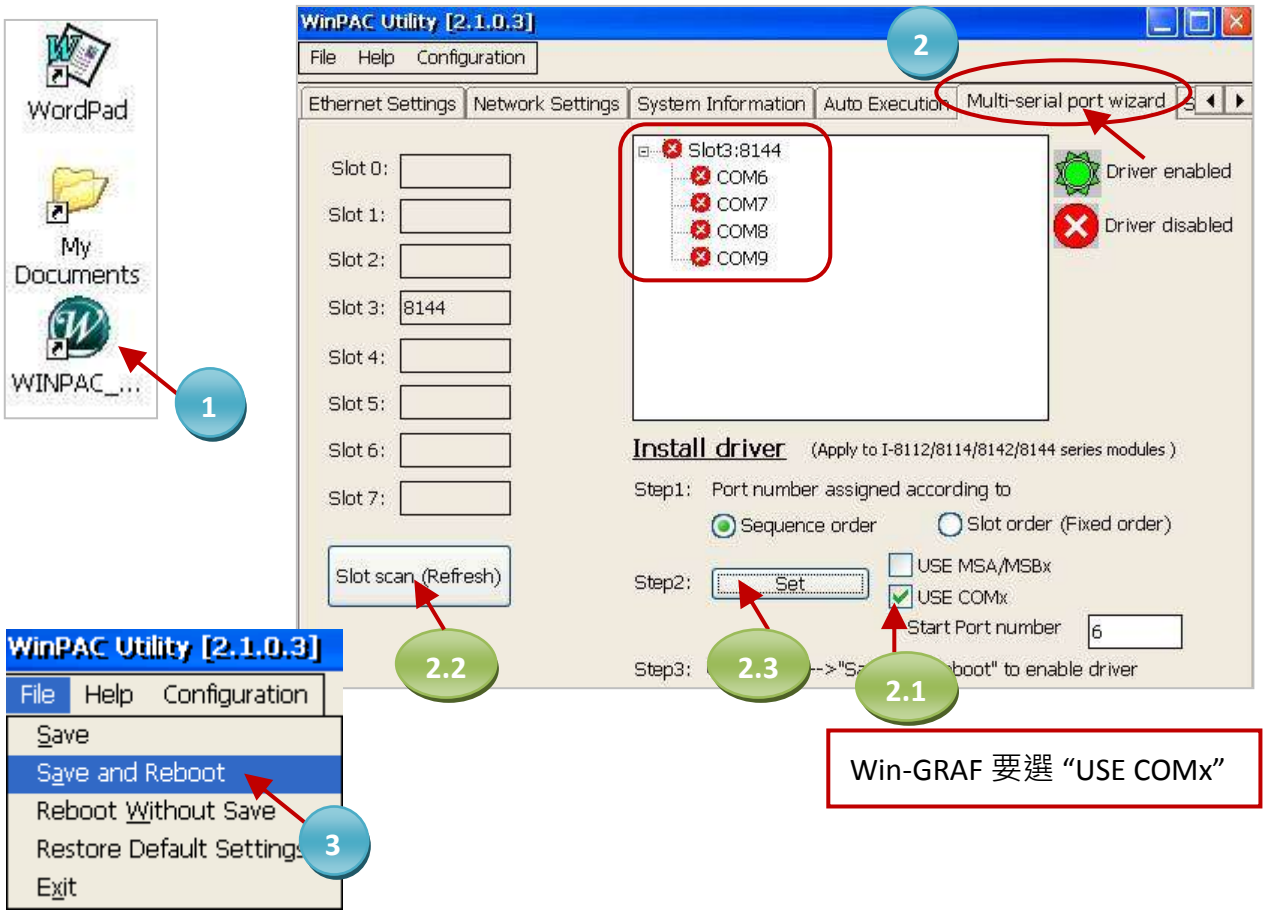
I-8112iW： 2 埠 隔離式 RS-232 模組
I-8114iW： 4 埠 隔離式 RS-232 模組
I-8114W ： 4 埠 無隔離式 RS-232 模組
I-8142iW： 2 埠 隔離式 RS-422/RS-485 模組
I-8144iW： 4 埠 隔離式 RS-422/RS-485 模組

註: WP-5xx8 不支援 XW-5xx 序列埠擴充子卡 (無法擴充序列埠)

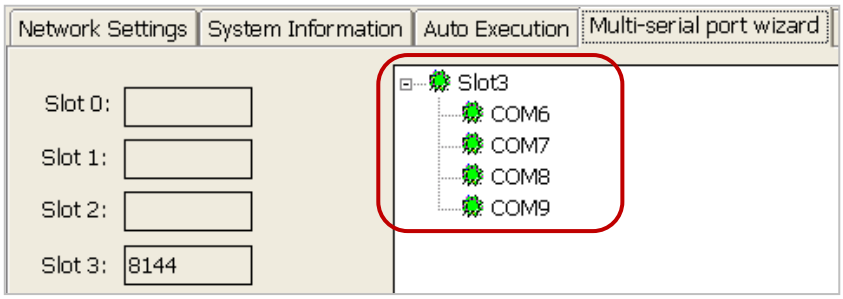
下方以 WP-8xx8 為例來說明:

使用上列模組前，必須先以 “WinPAC Utility” 進行設定。請將模組插入 WP-8xx8 的 0 ~ 7 槽 (最好是 0 ~ 3 槽)，再執行 WinPAC Utility :

- 點選 “Multi-serial port wizard” 頁籤。
- 勾選 “USE COMx” (**註:** Win-GRAF 不支援 “USE MSA/MSBx”)。
- 點選 “Slot scan” 按鈕，會自動搜尋目前 PAC 插槽上的多序列埠模組，並在左欄中顯示卡號。(若先前有設定過，會在右欄中顯示 COM 埠的原始設定。)
- 按下 “Set” 按鈕，更新原始設定為模組目前的設定。
- 執行 [File] > [Save and Reboot] 儲存新設定，並重新啟動 WP-8xx8。



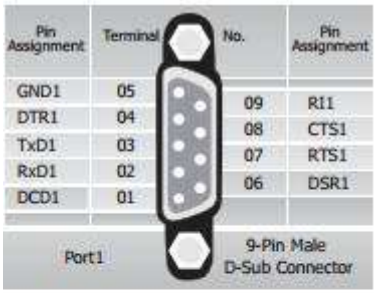
設定成功後，Win-GRAF 定義擴充卡的 COM 埠編號是 COM6 ~ COM37 (圖中的例子為 COM6 ~ COM9)。

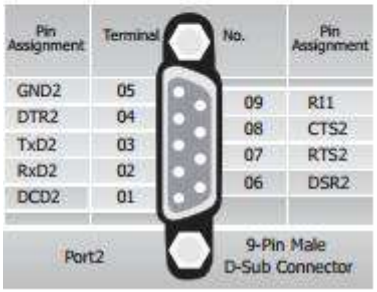


接腳圖:

I-8112iW

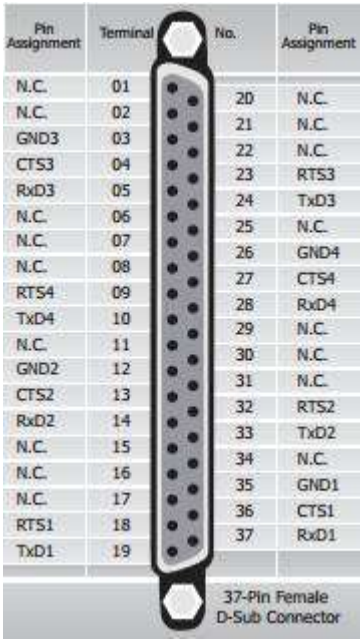
2-port Isolated RS-232





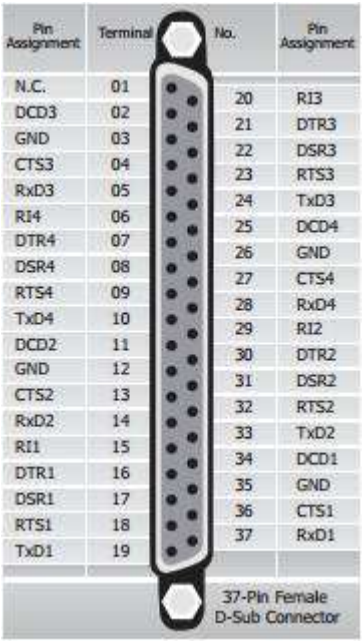
I-8114iW

4-port Isolated RS-232



I-8114W

4-port RS-232



I-8142iW

Terminal No.	Pin Assignment
01	D1+/TxD1+
02	D1-/TxD1-
03	RxD1+
04	RxD1-
05	GND1
06	D2+/TxD2+
07	D2-/TxD2-
08	RxD2+
09	RxD2-
10	GND2
11	N.C.
12	N.C.
13	N.C.
14	N.C.
15	N.C.
16	N.C.
17	N.C.
18	N.C.
19	N.C.
20	N.C.

I-8144iW

Terminal No.	Pin Assignment
01	D1+/TxD1+
02	D1-/TxD1-
03	RxD1+
04	RxD1-
05	GND1
06	D2+/TxD2+
07	D2-/TxD2-
08	RxD2+
09	RxD2-
10	GND2
11	D3+/TxD3+
12	D3-/TxD3-
13	RxD3+
14	RxD3-
15	GND3
16	D4+/TxD4+
17	D4-/TxD4-
18	RxD4+
19	RxD4-
20	GND4

I-8142iW (2-port Isolated RS-422/485)

RS-485 port1: (D1+ , D1-)
RS-485 port2: (D2+ , D2-)

RS-422 port1: (TxD1+ , TxD1-, RxD1+, RxD1-)
RS-422 port2: (TxD2+ , TxD2-, RxD2+, RxD2-)

I-8144iW (4-port Isolated RS-422/485)

RS-485 port1: (D1+ , D1-)
RS-485 port2: (D2+ , D2-)
RS-485 port3: (D3+ , D3-)
RS-485 port4: (D4+ , D4-)

RS-422 port1: (TxD1+ , TxD1-, RxD1+, RxD1-)
RS-422 port2: (TxD2+ , TxD2-, RxD2+, RxD2-)
RS-422 port3: (TxD3+ , TxD3-, RxD3+, RxD3-)
RS-422 port4: (TxD4+ , TxD4-, RxD4+, RxD4-)