SmartQ Script Editor
入門指南

[Version 1.0]

ICP DAS CO., LTD.

泓格科技股份有限公司

免責聲明 Warning

泓格科技股份有限公司對於因為應用本產品所造成的損害並不負任何法律上的責任。本公司保留有任何時間未經通知即可變更與修改本文件內容之權利。本文所含資訊如有變更，恕不予另行通知。

本公司盡可能地提供正確與可靠的資訊，但不保證此資訊的使用或其他團體在違反專利或權利下使用。此處包涵的技術或編輯錯誤、遺漏，概不負其法律責任。

版權 Copyright

商標識別 Trademark

本文件提到的所有公司商標、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。

授權宣告 License

使用者僅被授權可以在單一電腦上與有限條件下使用、備份軟體與相關資料，不得同時於該單一電腦外使用本軟體。本公司仍保有此軟體與相關資料的著作權及其他智慧財產權。除非事先經過本公司的書面授權，否則禁止重製、傳送及散佈等方式取得部份或全部軟體或相關的複製品。

目錄：

1. SMARTQ SCRIPT EDITOR 簡介.....5

1.1 產品特色.....5
1.2 系統需求.....6
1.2.1 Script Editor 執行環境需求.....6
1.2.2 Script Runtime 執行環境需求.....6
1.2.2.1 支援的控制器.....6
1.2.2.2 支援的 I/O 模組.....6

2. 安裝與移除 SMARTQ SCRIPT EDITOR.....8

2.1 安裝 SCRIPT EDITOR.....8
2.2 啟動 SCRIPT EDITOR.....9
2.3 移除 SCRIPT EDITOR.....9

3. SMARTQ SCRIPT EDITOR 使用者介面.....12

3.1 介紹.....12
3.2 功能表（MENU BAR）.....13
3.3 工具列（TOOL BAR）.....14
3.4 專案管理區.....15
3.5 程式編輯區.....16
3.6 編譯資訊區.....16

4. 編輯與開發 SMARTQ SCRIPT 專案.....18

4.1 專案開發所需步驟.....18
4.2 新增專案.....18
4.2.1 步驟.....18
4.2.2 專案屬性設定.....19
4.3 新增控制器與 I/O 模組.....21
4.3.1 步驟.....21
4.3.2 控制器與 I/O 模組屬性設定.....22
4.4 新增 SCRIPT 檔案.....25
4.4.1 步驟.....25
4.4.2 Script 屬性設定.....26
4.5 編輯 SCRIPT 檔.....29
4.5.1 Auto Completion 支援.....29
4.5.2 插入 Keywords.....29
4.6 指定與 FLASH HMI 通訊對照的專案路徑.....30
4.7 編譯專案.....31
4.7.1 編譯前的設定確認.....31
4.7.2 編譯步驟.....31
4.8 下載專案.....32
4.9 遠端啟動 / 停止 / 暫停 / 回復 專案.....33
4.10 線上專案偵錯.....34
4.10.1 設定中斷點.....34
4.10.2 偵錯執行模式.....35
4.11 匯出資料通道清單.....37

5. 輕鬆上手六步驟.....39

5.1 案例描述 39
5.2 案例實作 40
5.2.1 步驟 1 – 建立新專案 40
5.2.2 步驟 2 – 撰寫 script 程式碼 40
5.2.3 步驟 3 – 編譯專案 46
5.2.4 步驟 4 – 下載專案 46
5.2.5 步驟 5 – 執行專案 47
5.2.6 步驟 6 – 對專案進行執行中線上偵錯 47

APPENDIX A SMARTQ 專案設計流程與重要觀念 49

A.1 SCRIPT EDITOR 與 FLASH EDITOR 的資訊整合 49
A.2 硬體 I/O 模組組態設定 50
A.3 階層式控制器物件資料模型 55
A.4 階層式控制器物件資料模型使用範例 56

APPENDIX B SCRIPT 語法簡介 58

B.1 概要 58
B.2 數字 58
B.3 字串 59
B.4 其他類型 59
B.5 變數 60
B.6 運算子 60
B.7 控制結構 61
B.8 陣列 63
B.9 函式 64

APPENDIX C 關鍵字(KEYWORDS) 66

C.1 保留關鍵字 66
C.2 擴充保留關鍵字 66
C.3 其他要避免的識別字 66

APPENDIX D SCRIPT RUNTIME 操作介面解說 68

D.1 專案選取介面（PROJECT SELECT INTERFACE） 68
D.2 專案運作介面（PROJECT OPERATION INTERFACE） 69
D.3 QP-500 狀態介面（QP-500 STATUS INTERFACE） 69
D.4 除錯資訊介面（DEBUG INTERFACE） 70
D.5 程式控制介面（CONTROL INTERFACE） 71

1. SmartQ Script Editor 簡介

SmartQ Script Editor 為一個完整的 Script 語言開發環境，其結合了強大的程式編輯功能，使用者可以直覺且快速的編輯 Script 檔案。透過這些 Script 檔案即可控制並擷取控制器上所提供的 I/O 模組資料與資訊，實作出完整的工控系統邏輯控制，並且能讓在遠端的 PAC (Programmable Automation Controller) 執行這些 Script 檔案。

SmartQ Script Editor 亦提供了即時線上偵錯功能(On-line Debugger)，使用者撰寫完成 Script 程式後可先下載至遠端控制器上執行；並逐步檢視此 Script 檔案在執行中的變數值變更過程及程式邏輯的執行順序，方便使用者對所撰寫的 Script 進行即時線上偵錯。

1.1 產品特色

SmartQ Script Editor 提供了以下重要的產品特色：

- **字詞自動完成功能(Auto-completion)：**在 SmartQ Script Editor 撰寫 Script 時，當輸入關鍵字的同時，將觸發 Auto-completion 功能自動產生選單，列舉變數或關鍵字的建議與提示，可減輕使用者在撰寫程式時需記憶變數、指令的負擔，並減少因使用者輸入錯誤造成程式有誤的機率。
- **提供 I/O 模組的參數與函式：**撰寫控制 I/O 模組的 Script 程式時，透過 Script Editor 專案上控制器及控制器所連接 I/O 模組的屬性設定，Script Editor 可以搭配 Auto-completion 功能，動態提供對應的參數與函式列表輔助使用者撰寫 Script，使用者可更直觀的撰寫出控制 I/O 模組的 Script 檔案。
- **錯誤資訊的紀錄：**SmartQ Script Editor 在 Script 檔完成編譯動作後，將逐一紀錄程式語法錯誤的訊息與所在位置。使用者可清楚的追蹤語法錯誤發生的位置，並根據所顯示的訊息瞭解錯誤原因，以便快速的進行程式除錯與修改。
- **即時線上偵錯器(On-line Debugger)：**SmartQ Script Editor 本身具備一個強大的 Script 即時線上偵錯器(On-line Debugger)，使用者進入偵錯模式，可設定中斷點，使程式可以在任何指定的程式碼暫停執行，便於進一步探索整個程式執行環境中變數值變更及程式邏輯的執行過程。

1.2 系統需求

1.2.1 Script Editor 執行環境需求

執行本軟體所需的系統需求如下：

- 作業系統：Windows Server 2003、Windows Server 2008、Windows Vista、Windows XP
- CPU：建議 1G MHz 以上
- RAM：建議 256MB 以上
- 硬碟大小：建議可用空間為 600MB 以上的硬碟
- 顯示器：支援 1024 x 768 ,32bit 高彩的顯示器

Script Editor 的執行環境需安裝 Microsoft .NET Framework Version 2.0(或以上的版本)，請在執行 Editor 前，先下載 Microsoft .Net Framework，相關下載參考網址如下：

- Microsoft .Net Framework Version 2.0 下載網址：
  http://www.microsoft.com/downloads/details.aspx?FamilyID=0856eacb-4362-4b0d-8edd-aab15c5e04f5&DisplayLang=en
- Microsoft .Net Framework Version 3.5 下載網址：
  http://www.microsoft.com/downloads/details.aspx?familyid=333325FD-AE52-4E35-B531-508D977D32A6&displaylang=en

1.2.2 Script Runtime 執行環境需求

1.2.2.1 支援的控制器

在 Script Editor 中撰寫完成 Script 檔案後，即可透過 Editor 的遠端下載功能，將檔案下載至泓格科技的 PAC (Programmable Automation Controller)上執行，下述的 QPAC 系列控制器皆內含 Script Runtime，可執行 Script 檔案。

QP-8x2 系列	QP-812、QP-842、QP-882

1.2.2.2 支援的 I/O 模組

QPAC 控制器(QP-8x2)所內含的 Script Runtime 目前所支援的 I/O 模組請參照下表。

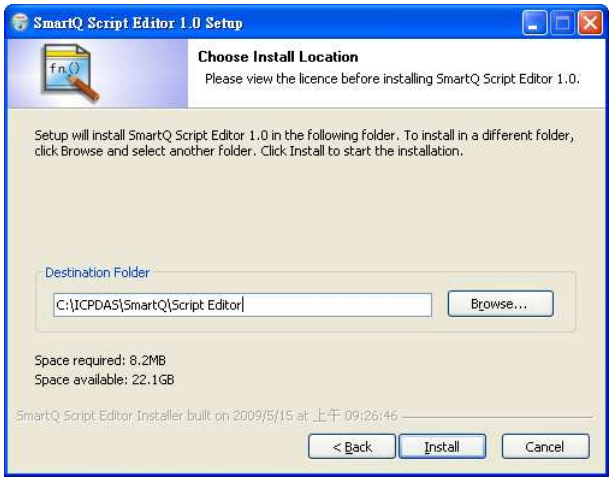
I-8K 系列 I/O 模組	
DI/DO	I-8037、I-8040、I-8041、I-8042、I-8048、I-8050、I-8051、I-8052、I-8053、I-8054、I-8055、I-8056、I-8057、I-8058、I-8060、I-8063、I-8064、I-8068、I-8069、I-8172
AI/AO	I-8017H、I-8017HS、I-8024
I-87K 系列 I/O 模組	
DI/DO	I-87040、I-87041、I-87052、I-87053、I-87054、I-87055、I-87057、I-87058、I-87063、I-87064、I-87065、I-87066、I-87068、I-87069
AI/DO	I-87013、I-87017、I-87018、I-87024
I-7K 系列 I/O 模組	
DI/DO	I-7041、I-7042、I-7043、I-7044、I-7045、I-7050、I-7051、I-7052、I-7053、I-7055、I-7058、I-7060、I-7063、I-7065、I-7066、I-7067
AI/AO	I-7011、I-7012、I-7013、I-7014、I-7016、I-7017、I-7018、I-7021、I-7022、I-7024

2. 安裝與移除 SmartQ Script Editor

2.1 安裝 Script Editor

- 將 ICP DAS V1.0(或以上版本)的光碟片放入安裝機器的光碟機中
- 至 SmartQ SW Tools\SmartQ Script Editor\ 的目錄下點選 SmartQ Script Editor 的安裝程式(SmartQ Script Editor Setup.exe)進行安裝
- 系統將進入下述畫面：

- 閱讀說明並確認 Microsoft .NET 2.0 runtime(或以上的版本)已經安裝完成，按〔Next >〕。
- 選擇欲安裝 Script Editor 檔案的目的地位置，預設的安裝路徑為 C:\ICPDAS\SmartQ\Script Editor\下，



- 確認後設定完成後，請按〔Install〕開始軟體安裝程序。

- 安裝完成後，按下〔Finish〕結束安裝程式的執行。若勾選[Run SmartQ Script Editor]選項，程式結束後將啟動 SmartQ Script Editor 的主程式。

2.2 啟動 Script Editor

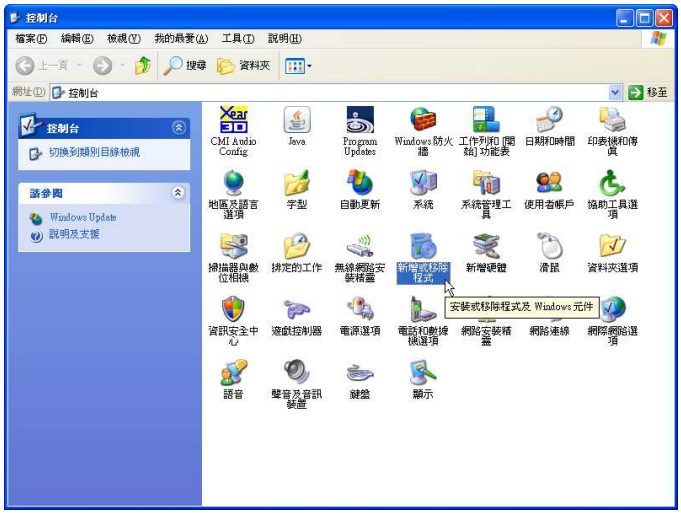
- 以 Windows XP 的開始功能表為例，執行〔開始〕/〔所有程式〕/〔ICPDAS〕/〔SmartQ〕/〔SmartQ Script Editor〕

- 也可以雙擊桌面或快速啟動列的捷徑來執行 SmartQ Script Editor

2.3 移除 Script Editor

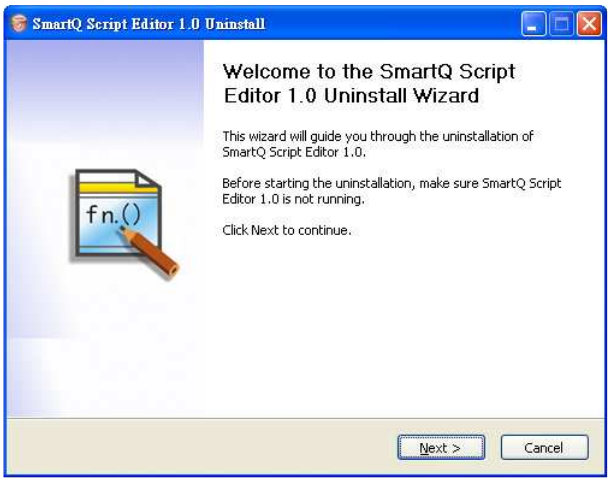
- 執行〔開始〕/〔設定〕/〔控制台〕，進入控制台視窗

- 點選〔新增或移除程式〕圖示並雙擊滑鼠左鍵

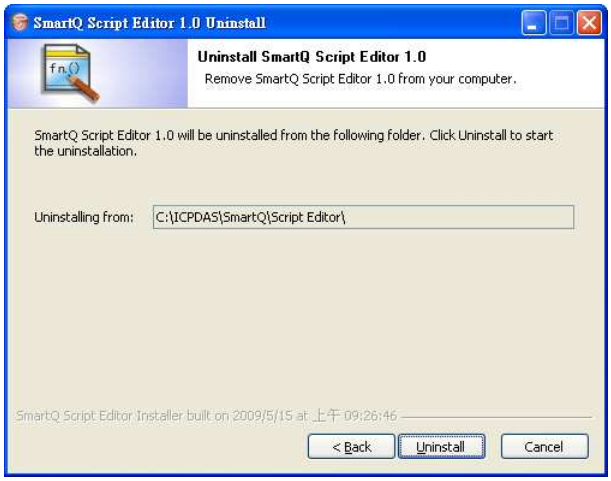


- 在目前安裝程式清單中點選〔SmartQ Script Editor〕，按下移除按鈕

- 此時將執行 Script Editor 的反安裝程式，按下〔Next >〕



- 確認要移除的程式所在的安裝目錄路徑。確認完畢，按下〔Uninstall〕



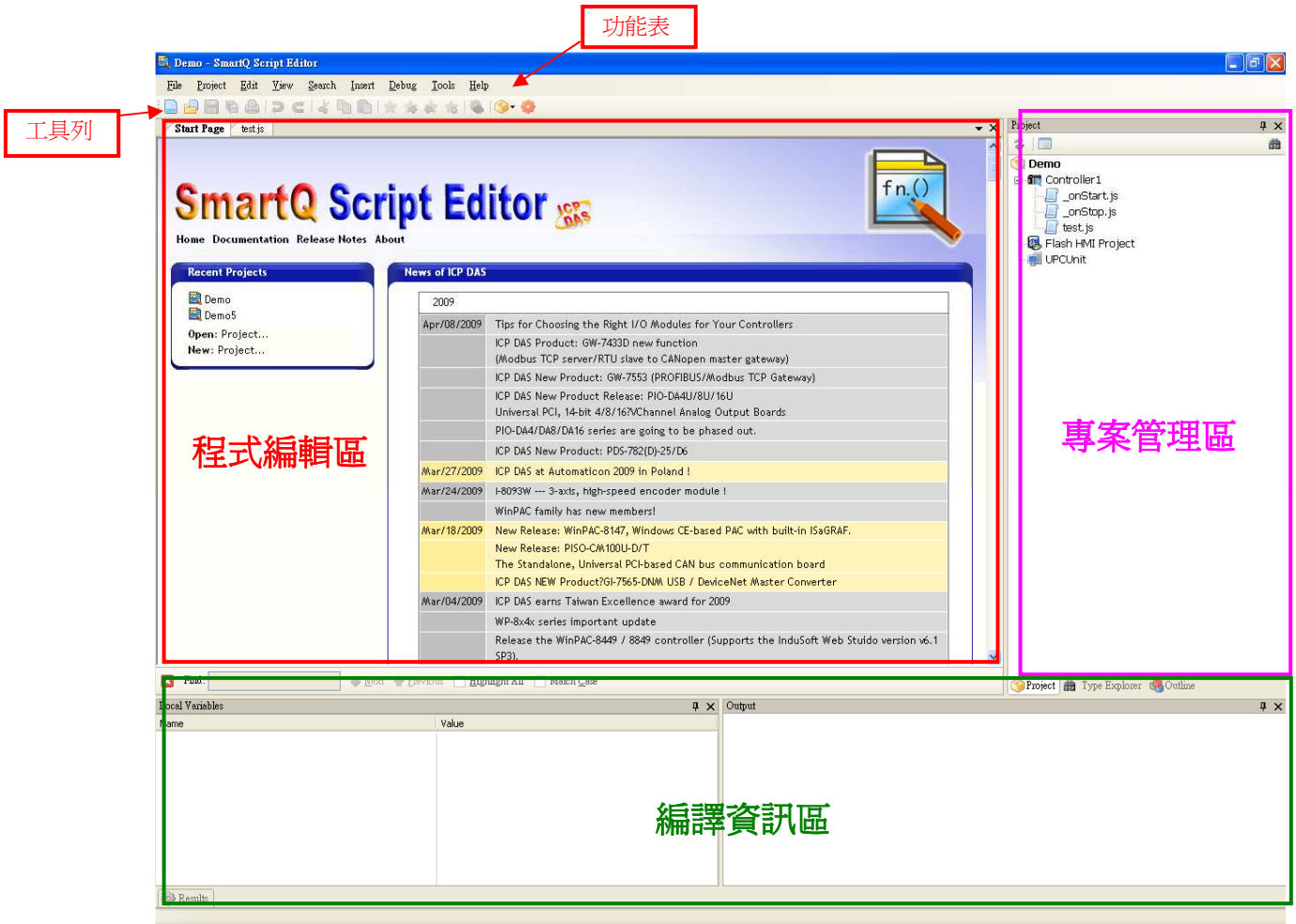
- 此時將進行程式的移除，請耐心等待。
- 移除程式執行完畢後，將出現完成解除安裝的畫面，按下〔完成〕結束移除程式

3. SmartQ Script Editor 使用者介面

3.1 介紹

SmartQ Script Editor 的操作視窗分為幾個工作區，每一個工作區包含一個或多個視窗(View)。主要的工作區為：

- 功能表(Menu bar)
- 工具列(Tool bar)
- 程式編輯區—包含首頁(Start Page)、程式編輯(Editor)等視窗。
- 專案管理區—包含專案管理員(Project Manager)、概要(Outline)、型別探索(Type Explorer)」等三個視窗。
- 編譯資訊區—包含結果(Result)、輸出(Output)、區域變數(Local Variables)等三個視窗。



3.2 功能表（Menu Bar）

SmartQ Script Editor 的功能表，具備多項選單提供執行各種檔案編輯與程式開發時所需要的指令。以下列表簡單列出功能表選單的基本功能：

功能表	說　明
檔案 (File)	1. 開啟 或 儲存 Script 檔案 2. 紀錄最近開啟的檔案與專案 3. 關閉 Script Editor 主程式
專案 (Project)	1. 新增 /開啟/ 儲存/關閉 專案 2. 編譯專案
編輯 (Edit)	撰寫 Script 時所需的編輯功能，如：剪下、複製、貼上、復原、取消復原、展開或收縮程式碼....等功能
檢視 (View)	1. 開啟檢視各視窗 2. 將視窗外觀配置還原為預設值
搜尋 (Search)	支援閱讀或編輯程式文件時需要的功能，如搜尋並替換文字、跳到對應的大括號、書籤的加入與刪除。
插入 (Insert)	可以插入其他內容、檔案資訊、時間戳記、關鍵字
偵錯 (Debug)	1. 偵錯相關指令：如開始偵錯、暫時偵錯、停止等。 2. 清除偵錯連線程式，結束偵錯 3. 加入與清除中斷點
工具 (Tools)	提供檔案比對與語言設定。
說明 (Help)	提供版本說明、公司網頁等資訊。

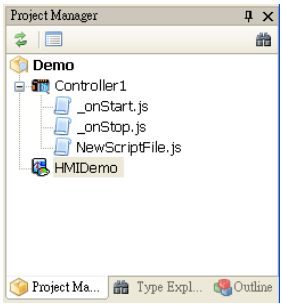
3.3 工具列（Tool Bar）

工具列提供常用的功能選項按鈕，可快速執行常用的指令。以下列表為工具列各按鈕說明：

工具列按鈕	說　明
	新增一個純文字檔(副檔名為*.txt)
	開啟檔案(＊.txt 或 *.js)
	儲存正在編輯的檔案
	儲存所有開啟的檔案
	列印
	復原
	取消復原
	剪下
	貼上
	複製
	在文件上新增一個書籤標記
	到下一個書籤位置
	到上一個書籤位置
	刪除書籤標記
	此檔語法檢查
	專案歷史清單
	編譯專案

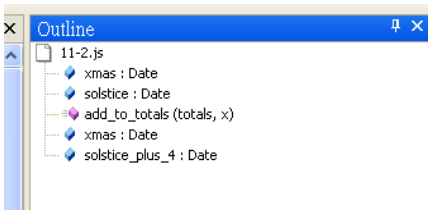
3.4 專案管理區

專案管理區中，與專案相關的檔案及項目將在**專案管理員視窗(Project Manager View)**工作區以樹狀結構的方式呈現，使用者可以展開或收合專案視窗中的資料夾，以便檢視、存取與編輯檔案。使用者可在專案管理視窗下：

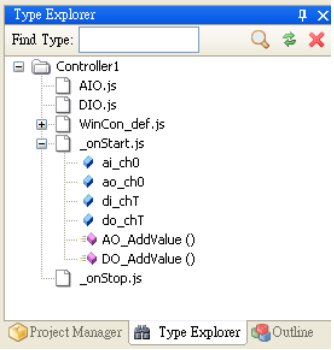


- 瀏覽所有的 Script 檔案和文字檔
- 編譯專案(Build project)
- 建立新專案、檔案
- 匯入專案與檔案
- 管理已存在的檔案(剪下、貼上、刪除、移動或更名)
- 在程式編輯區上開啟所選取的檔案內容

概要視窗（Outline View）顯示正在 Editor 中編輯的 Script 檔案概觀，此概觀由圖示組成，代表 Script 檔案中所宣告的變數或函式。當使用者在 Outline View 選擇了一個圖示，此圖示所對應的變數或函式名稱將呈現反白。



型別檢索視窗（Type Explorer）將顯現正在編輯的 QPAC 專案中所使用的變數與函式圖示概觀。點選 QPAC 專案上的檔案圖示，此檔案將開啟；圖示所對應的變數或函式名稱將呈現反白。



3.5 程式編輯區

開啟應用程式時，程式編輯區將出現首頁頁籤（目前為泓格公司新產品資訊與新聞），該頁籤即為**首頁視窗（Start Page）**。在首頁視窗亦可開啟或新增專案。

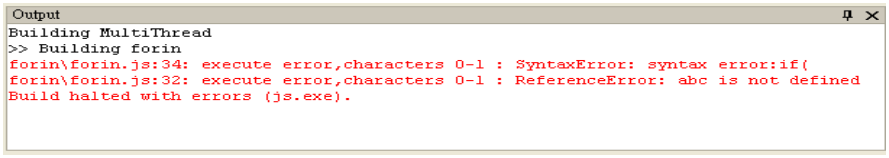
程式編輯區可在**編輯視窗（Editor View）**下多重開啟 Script 檔案、純文字檔案或其他檔案，以頁籤切換各檔案的編輯視窗。當 Script Editor 開啟檔案類型不同時，將依照檔案類型提供對應的編輯器，例如：編輯一個 Script 檔時，編輯器將提供下列功能：

- Code Auto-Completion：提供 Auto-Completion 的功能，可減輕使用者必須大量記憶程式碼的負擔，減少重複的編碼工作、節省開發的時間，並且提高 Script 的正確性。
- 語法明亮效果(Syntax high lighting)：色彩標示語法，方便辨識語法。
- 程式碼收合(Code Folding)：可自由收合或展開程式碼，使程式碼更容易閱讀。

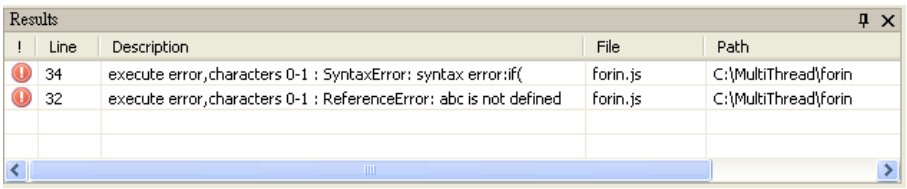
3.6 編譯資訊區

編譯資訊區為提供使用者在編譯或偵錯專案時，所需要參考的重要訊息，如：編譯語法錯誤、關鍵字拼法錯誤或型別不符。這些資訊將顯示於輸出視窗(Output View)、結果視窗(Result View)、區域變數視窗(Local Variable View)。

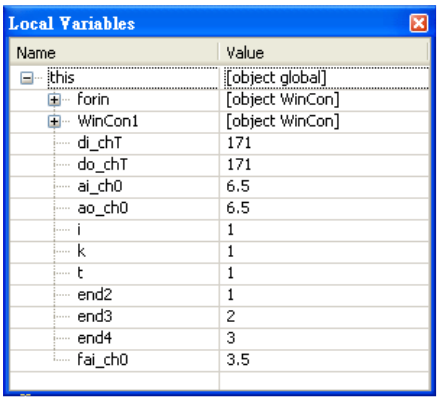
輸出視窗(Output View)：使用者在編譯 Script 程式時，問題、錯誤或警告紀錄將顯示於輸出視窗中。除了編譯的訊息，編譯結果、與遠端連線的狀況及偵錯程式時所執行的列印輸出等訊息也將顯示於輸出視窗中。



結果視窗(Result View)：輸出視窗所顯示的問題、錯誤或警告將以條列的方式紀錄下來，呈現於結果視窗中。以滑鼠雙擊某項條列的問題、錯誤或警告時，可在程式編輯區中自動開啟發生問題的 Script 檔；滑鼠游標將移到出現問題的原始程式處。



區域變數視窗(Local Variables View)：在偵錯時，程式執行的變數將以樹狀條列的方式顯示於區域變數視窗。在進行線上偵錯時，使用者可檢視變數在當下的實際數值及變數型態，並且可以手動更改部分變數的值。



4. 編輯與開發 SmartQ Script 專案

4.1 專案開發所需步驟

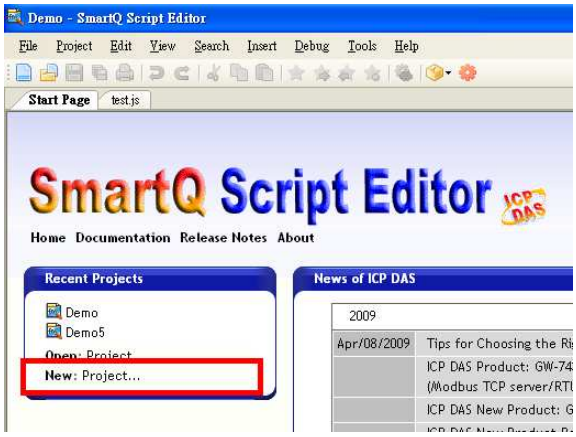
以 SmartQ Script Editor 開發控制器端控制邏輯的開發流程如下，各步驟的詳細操作程序請參考以下各章節。

確認所使用的控制器->確認控制器上所安裝的I/O模組->啟動Script Editor->新增專案->新增控制器與I/O模組設定->新增Script檔案->編輯Script檔案->編譯專案->下載專案->啟動

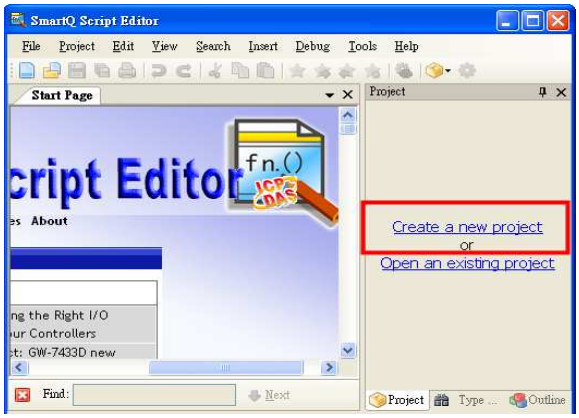
4.2 新增專案

4.2.1 步驟

- 按下開新檔案的連結(link)
  - 方法一：由首頁點選首頁左上方的最近開啟的專案區塊中的 [New：Project...] 連結

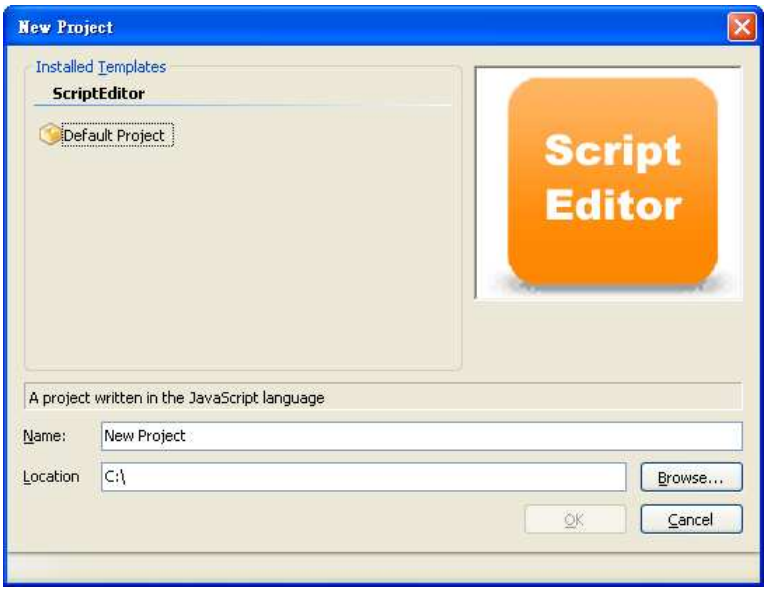


  - 方法二：由專案管理員視窗點選 [Create a new project] 連結



- 方法三：由功能表選取 **Project > New Project**

- 以專案範本建立專案

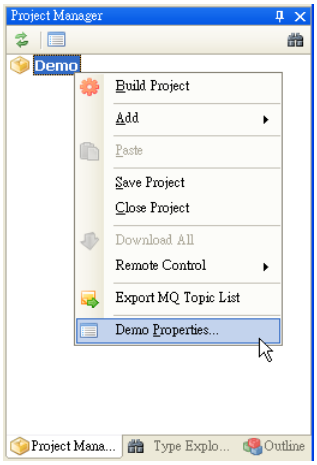


- Name：輸入專案名稱(ex：Demo)
  注意：請勿使用包含/\:;*?<>,$%字元的檔案名稱
- Location：輸入或瀏覽選擇儲存專案的路徑

- 按下 OK 按鈕

[註]：

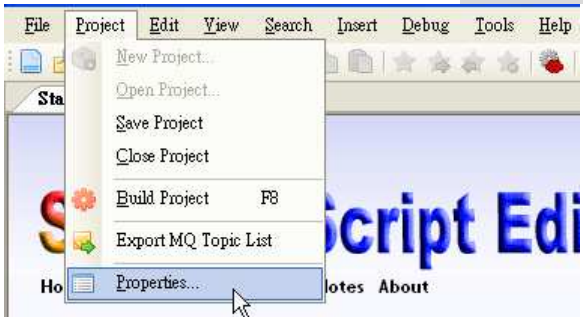
新增專案完成後，將在專案目錄下新建一個檔名為＊.jsproj 的專案組態檔，用以儲存專案的設定。



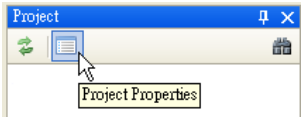
4.2.2 專案屬性設定

- 開啟專案屬性設定視窗
  - 方法一：點選專案管理員視窗的最上層節點，即專案的根節點，按滑鼠右鍵，顯示右鍵選單，點選 [${專案名稱} Properties…] (ex：Demo Properties)

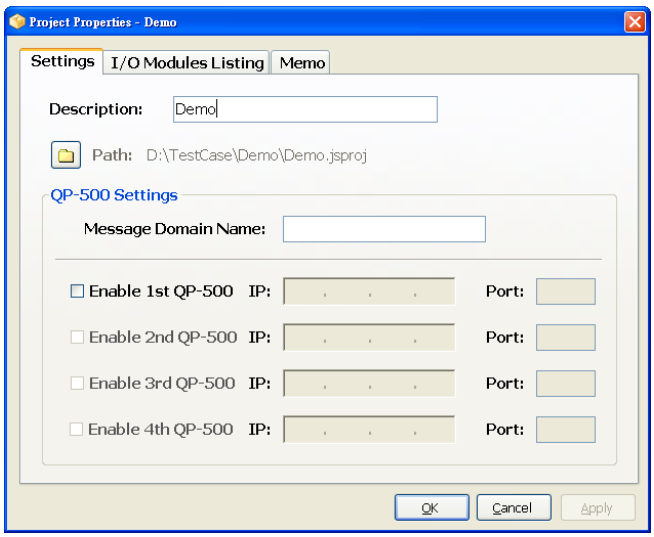
- 方法二：由功能表依序點選 **Project > Properties**



- 方法三：按下位於專案管理員視窗上方工具列的專案屬性按鈕



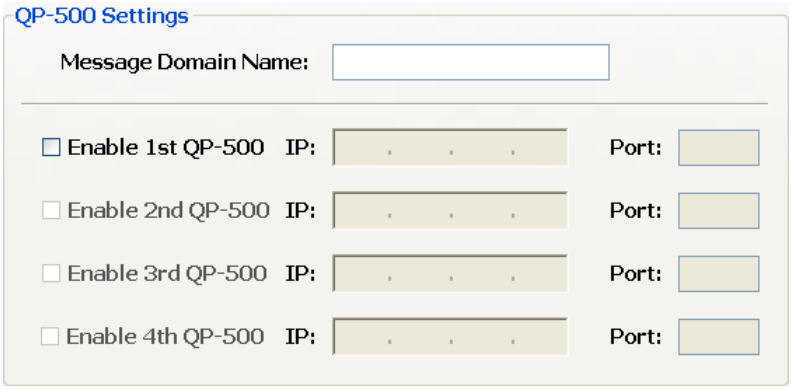
點選後，將開啟專案屬性設定視窗：



- [Settings]頁籤：顯示專案實體路徑與設定 QP-500。
- [I/O Modules Listing] 頁籤：控制器與 I/O 模組屬性設定完成後，在此頁籤將顯示各個模組的資訊。
- [Memo]頁籤：讓使用者填寫專案備註。

點選[Settings]頁籤的  按鈕，可開啟專案根目錄的資料夾，如圖示範例，則為「C：\Demo\」檔案資料夾

- 若專案中控制器彼此間或控制器與人機介面需要藉由資料通道溝通訊息時，則需要設定 QP-500 的相關資訊(相關說明可參考"SmartQ 系統概述")。勾選要設定的項目，即可編輯 IP 位址及 Port 號（Port 號預設值為 1883）。目前本系統最多可掛接 4 個獨立的 QP-500，以支援 QP-500 備援(Redundant)系統的架設。

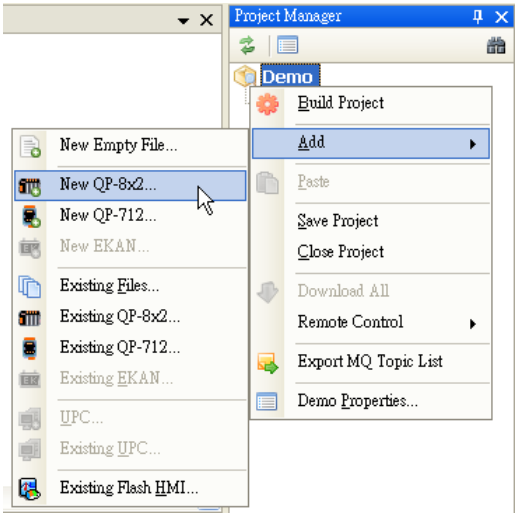


- 若系統環境中需要透過 SmartQ NetDB 資料庫系統將數個訊息群組的資料儲存在相同的資料庫時，則需要設定訊息網域設定名稱[Message Domain Name]，幫助使用者確認資訊是屬於哪個訊息群組所送出。
- 按下 OK 按鈕，完成專案屬性設定

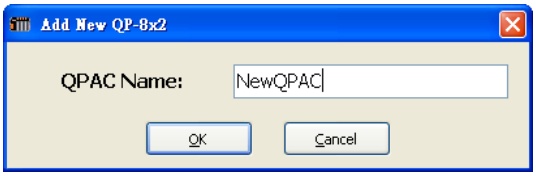
4.3 新增控制器與 I/O 模組

4.3.1 步驟

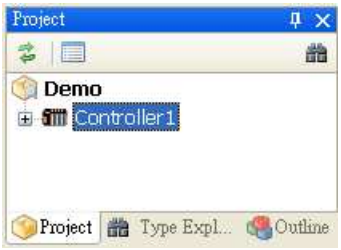
- 在專案管理員視窗點選專案的根節點
- 按滑鼠右鍵，顯示右鍵選單



- 按下 **ADD > New QP-8x2**
- 將跳出新增 QPAC 對話方塊



- 輸入 QPAC 專案名稱(Ex: Controller1)，按下 OK 按鈕
- 專案節點下將新增一個 QPAC 子專案

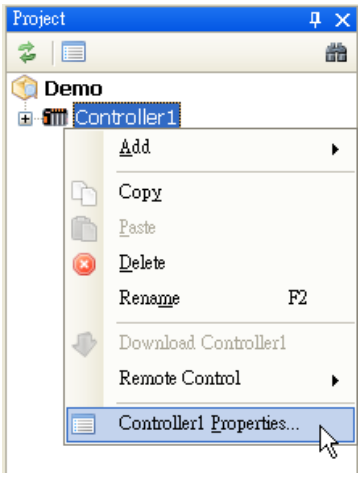


[註]：

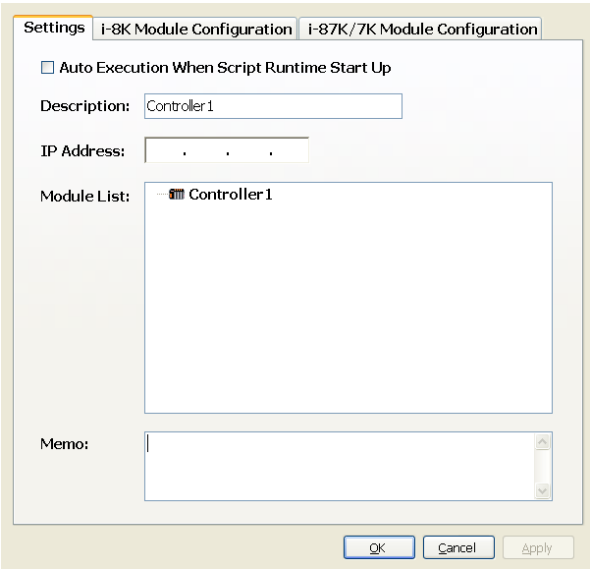
每一個 QPAC 專案名稱為一個資料夾名稱，也是一個 Script 內的變數名稱，故命名時必須遵循命名規則：第一個字元可以是任何大小寫英文字母、底線字元（_），第二個以後的字元則可以為英文字母、底線字元（_）或 0~9 數字。專案名稱長度限制：20 個字元。

4.3.2 控制器與 I/O 模組屬性設定

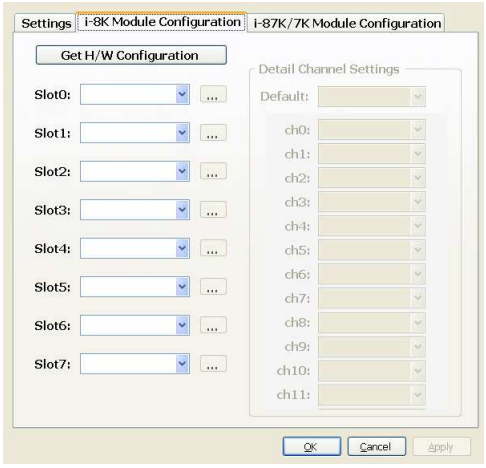
- 在專案管理員視窗中點選 QPAC 子專案節點，按滑鼠右鍵顯示右鍵選單，點選[${QPAC 專案名稱} Properties ...]（Ex: Controller1 Properties）設定屬性。



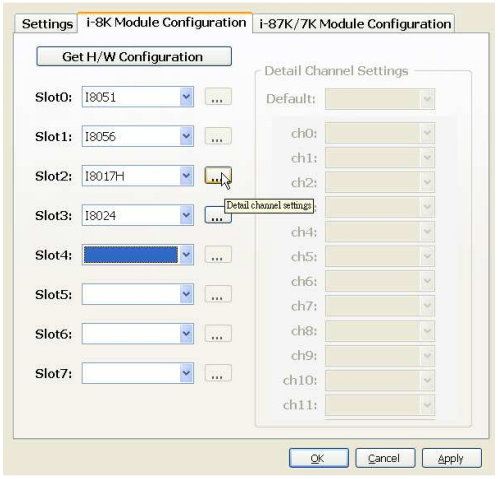
- 點選後，將開啟 QPAC 子專案屬性設定視窗



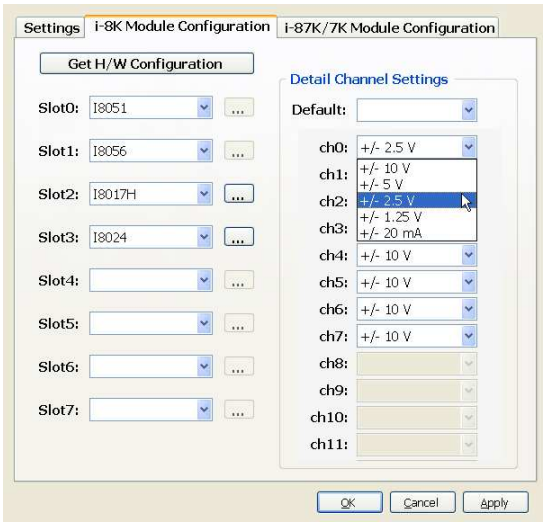
- 點選[Settings]頁籤，設定遠端 QPAC 的 IP 位址。IP 設定完成後，若要控制器在啟動 Script Runtime 後自動執行此專案，請勾選[Auto Execution When Script Runtime Start up]。
- i-8K 模組設定：
  - 點選[i-8K Module Configuration]頁籤開啟設定視窗：



  - 按下[Get H/W Configuration]按鈕掃描，此 QPAC 子專案上所安裝的 i-8k 模組將依序列出

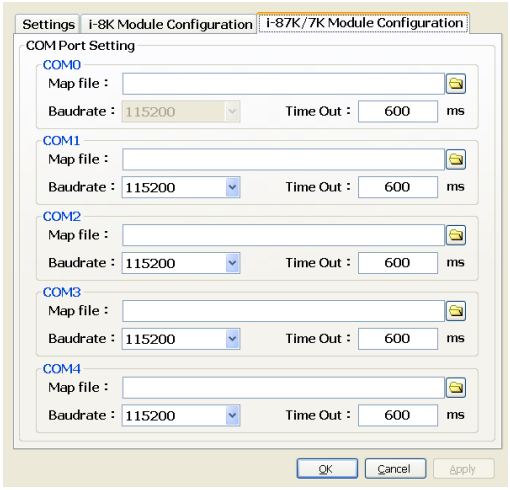


  - 若 Slot 上的模組為可設定 Channel 屬性的模組，如：I8017H、I8017HS、I8024 等模組，可按右側按鈕 ... ，此時視窗右側 [Detail Channel Settings]將呈現可設定的狀態。若所有的 Channel 的屬性要設定為相同屬性 (如:輸出入電壓或電流的範圍等)，點選[Default]選擇相關參數即可一次完成設定，將所有的 Channel 設為相同屬性。若要個別設定各 Chanel 的屬性，則依序點選 channel 個別設定相關參數。若此模組不需設定參數，右側按鈕 ... 將呈現鎖定狀態，無法點選設定。

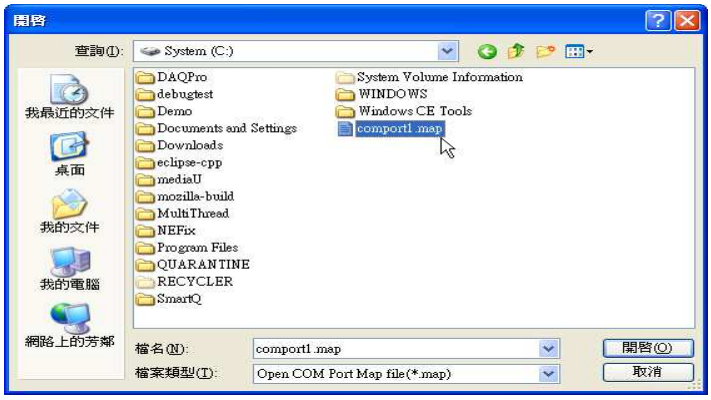


設定 i-87K/7K 模組

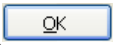
- 點選[i-87K/7K Module Configuration]頁籤

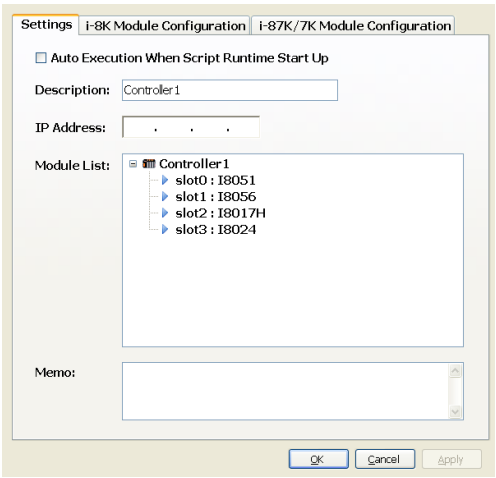


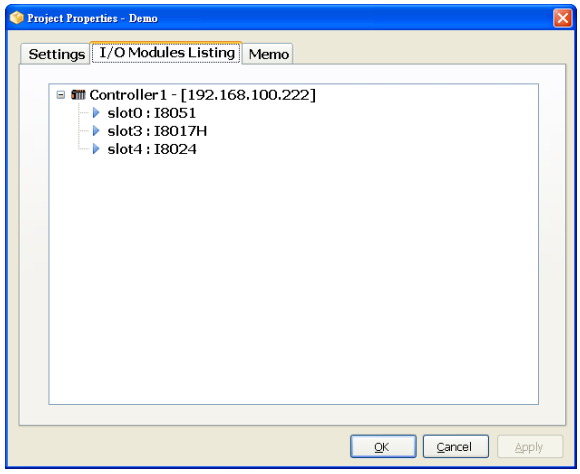
- 選擇要設定的 COM port ，在 map file 路徑右方按下  按鈕匯入 COM Port map 檔(副檔名：＊.map)



- 按下 OK 按鈕儲存目前設定，並關閉視窗結束設定。
- 按下 Apply 按鈕套用目前設定，可繼續設定下一個 COM Port。

- 若 COM port 匯入的 map 檔有誤，可再次點選 map 檔路徑右側按鈕  。開啟瀏覽視窗後，請勿匯入檔名，直接點選取消。此時將清除 COM port 匯入 map 檔路徑，點選  或  儲存設定，即可清除錯誤設定。
- 完成設定後，QPAC 子專案的[Settings]頁籤與專案屬性頁中的[I/O Modules Listing]頁籤將立即更新，顯示各硬體控制器及其模組的最新設定狀態。





[註]：

i-87K/7K 模組無法直接在 Script Editor 進行 COM Port 設定，必須匯入 map 檔案進行設定，map 檔請使用 DCON Utility 設定取得，請至以下網址下載 DCON Utility 軟體及相關文件說明：

http://www.icpdas.com/download/7000/7000.htm

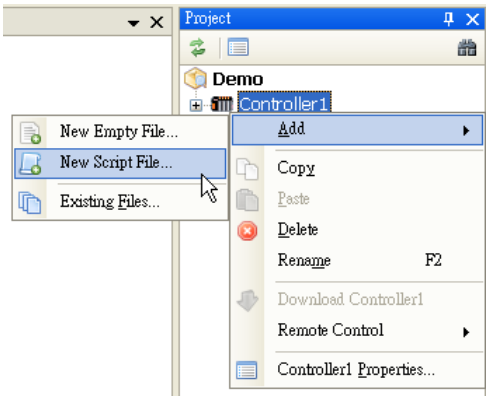
[註]：

此程序完成後，SmartQ Flash HMI Editor 可以由專案的根目錄下取得名稱為[${專案名稱}_Settings.xml]（如:Demo_Settings.xml)設定檔，此檔案為記錄 Script Editor 中目前此專案內所有 QPAC 控制器可以使用的資料通道(包含 QPAC 控制器所連接的 IO 模組實體資料通道、虛擬資料通道及系統資料通道等)，以便建立與 SmartQ Flash HMI Editor 內 Flash HMI Component 的通訊管道。

4.4 新增 Script 檔案

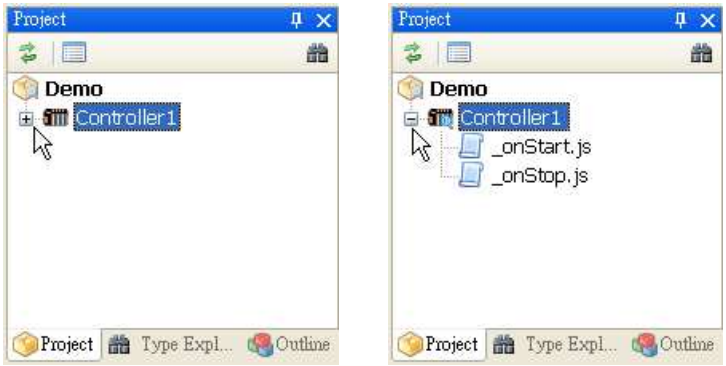
4.4.1 步驟

- 在專案管理員視窗中，選擇 QPAC 子專案的節點，點選並按滑鼠右鍵，顯示右鍵選單
- 由右鍵選單中按下 **Add > New Script File**



- 輸入檔案名稱後，按下 OK 按鈕

- 若 QPAC 子專案內含已建立的檔案，在專案管理員視窗中，此 QPAC 子專案前方將出現 ⊞ 記號。按下此加號，將展開子專案下的檔案節點。若此時新增檔案，此新加入的檔案節點也將立刻顯示於專案管理員視窗中。以滑鼠右鍵雙擊檔案節點，編輯視窗將開啟檔案內容（尚未編輯的檔案內容將呈現空白）。



- 每一個 QPAC 子專案下，使用者最多可以自行新增 10 個 Script 檔（不包含_onStart.js 和_onStop.js）

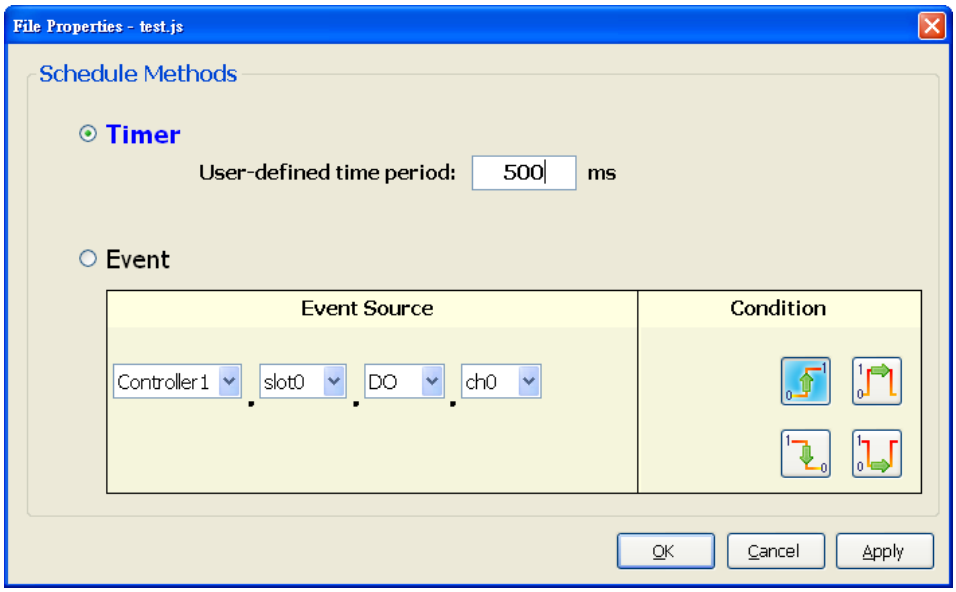
[註]：

每一個新增的 QPAC 子專案中將自動內建 2 個 Script 檔，即：『_onStart.js』與『_onStop.js』檔案。『_onStart.js』為 Runtime 啟動執行後，必須先行執行的指令。主要的目的為進行變數與函式的初始化以及 QPAC 所需的初始設定，只需要初始化執行一次的變數與函式必須在此處宣告。『_onStop.js』則為 QPAC 上 Script Runtime 停止執行前，最後必須執行的動作。

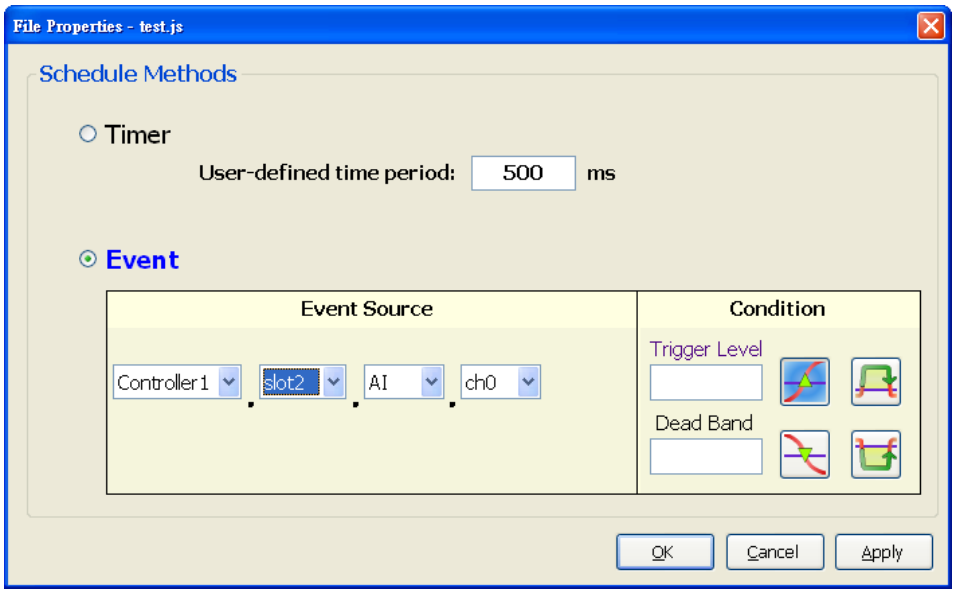
4.4.2 Script 屬性設定

- 在專案管理員視窗中，選擇要設定的 Script File，點選按滑鼠右鍵，顯示右鍵選單。由右鍵選單中點選[＄{Script File 名稱} Properties…](ex: test.js Properties…)
- 在 Script File 屬性視窗可設定每一個 Script File 在遠端 QPAC 執行時，所需要的排程特性，包含「計時器控制」(Timer)、或「事件觸發」(Event)
  - 「計時器控制」(Timer)設定此 Script 檔案將按照指定的時間間隔執行（即每隔多久的時間要執行一次）

- 「事件觸發」(Event)則是在符合使用者自行設定的條件時才能觸發執行此 Script

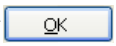


點選事件觸發(Event)時，觸發事件的條件（Condition）設定介面將按照事件根源（Event Source）為數位或類比資料呈現其對應的設定介面。如：上圖為數位型態的介面，下圖則為類比型態的介面



	圖示	說　明
數位		Rising Edge Trigger，當外部數位訊號由 0 轉換為 1 時，將觸發事件，Script檔案內容僅執行一次
		Falling Edge Trigger，當外部數位訊號由 1 轉換為 0 時，將觸發事件， Script檔案內容僅執行一次

		Level Trigger (Above)，當外部數位訊號維持在 1 的時候，執行Script檔案內容，直到數位訊號更改其值為 0 時，則停止執行。
		Level Trigger (Below)，當外部數位訊號維持在 0 的時候，執行Script檔案內容，直到數位訊號更改其值為 1 時，則停止執行。
類比		Rising Edge Trigger，當外部類比訊號值大於[Trigger Level]設定值時，將觸發事件，並且只執行一次Script檔案內容。事件觸發後，事件狀態將一直維持在新的狀態。當類比訊號改變小於[Trigger Level]減去[Dead Band]的值時，則事件狀態才將回歸未觸發前的狀態。
		Falling Edge Trigger，當外部類比訊號值小於[Trigger Level]設定值時，將觸發事件，並且只執行一次Script檔案內容。事件觸發後，事件狀態將一直維持在新的狀態。當類比訊號改變大於[Trigger Level]加上[Dead Band]的值時，則事件狀態才將回歸未觸發前的狀態。
		Level Trigger (Above)，當外部類比訊號值大於[Trigger Level]設定值時，將開始執行Script檔案內容，直到類比訊號小於[Trigger Level]減去[Dead Band]的值時，才停止執行。
		Level Trigger (Below)，當外部類比訊號值大於[Trigger Level]設定值時，將開始執行Script檔案內容，直到類比訊號大於[Trigger Level]加上[Dead Band]的值時，才停止執行。
	Trigger Level	設定類比訊號觸發事件的觸發點
	Dead Band	設定類比訊號判斷事件狀態的遲滯區，此數值必須為正數。

- 按下  按鈕確認

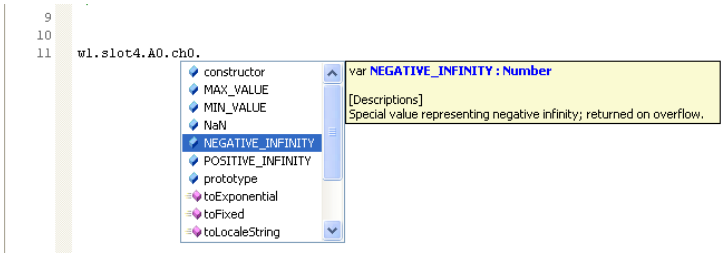
[註]：

若 Script 檔案的排程特性屬性為 Timer，專案或 QPAC 子專案編譯完成後，系統將產生一個最低底限的週期時間建議值，使用者可以依據其值來調整每一個 Script 執行週期。

4.5 編輯 Script 檔

4.5.1 Auto Completion 支援

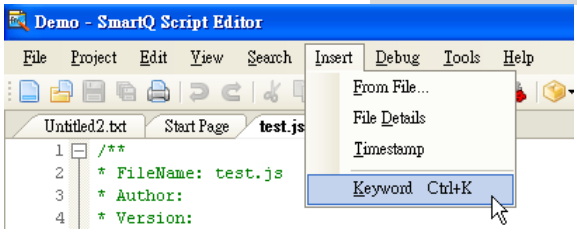
- 在 Editor 上，若鍵入符合關鍵字、變數或函式的前幾個字元（最少兩個字元以上），將自動啟動 Auto-Completion 功能，跳出一個輔助列表，顯示相關的關鍵字、變數或函式等。
- 若在變數後面鍵入「.」時，將自動啟動 Auto-Completion，若該變數具有相關的屬性或函式，將顯示於輔助列表中。

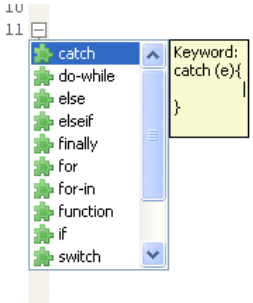


- 若持續輸入字元，符合的字串越少，輔助列表的顯示內容也將隨之減少。若列表中僅剩一個符合選項，將自動插入此選項。
- 可直接選擇列表中的項目，雙擊滑鼠左鍵將此選項插入程式碼中。
- 按下[Esc]可關閉輔助列表

4.5.2 插入 Keywords

- 插入 Keywords：
  - 方法一：按下快速鍵[Ctrl+K]
  - 方法二：由功能表點選 **Insert > Keyword**

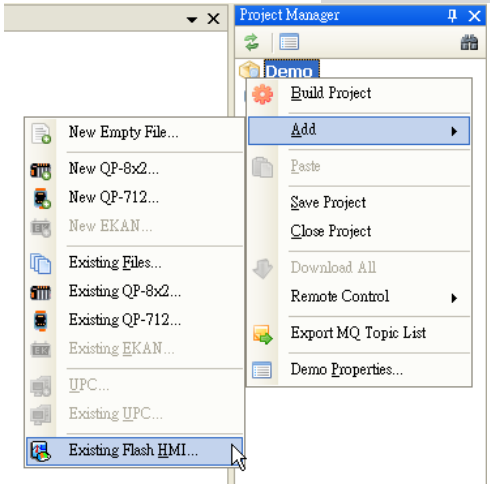




- 可直接選擇輔助列表中的項目，雙擊滑鼠左鍵將此選項插入程式碼中。
- 按下[Esc]可關閉輔助列表

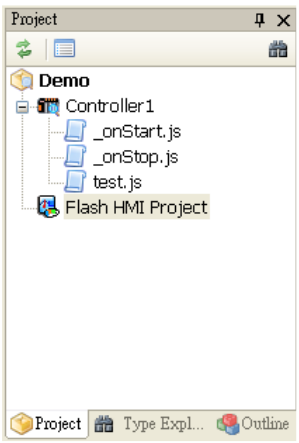
4.6 指定與 Flash HMI 通訊對照的專案路徑

- 在專案管理員視窗中，點選專案的節點，按滑鼠右鍵，顯示右鍵選單
- 由右鍵選單中按下 **Add > Existing Flash HMI…**



- 加入 Flash HMI 專案目錄路徑，當 Script Runtime 執行時便可建立資料通道以便與 Flash HMI 進行訊息溝通。

- 此時在專案管理員視窗中可看到新增的 Flash HMI 專案節點 



[註]：

完成此程序後， Script Editor 便可取得 SmartQ Flash HMI Editor 開發專案內所有 Flash HMI Component 使用的資料通道(包含 QPAC 控制器所連接的 IO 模組實體資料通道、虛擬資料通道及系統資料通道等)，Script Editor 將這些資訊加以編譯並加入控制器子專案（如:QPAC 子專案)的設定檔，便可下載至控制器端的 Script Runtime 執行設定。

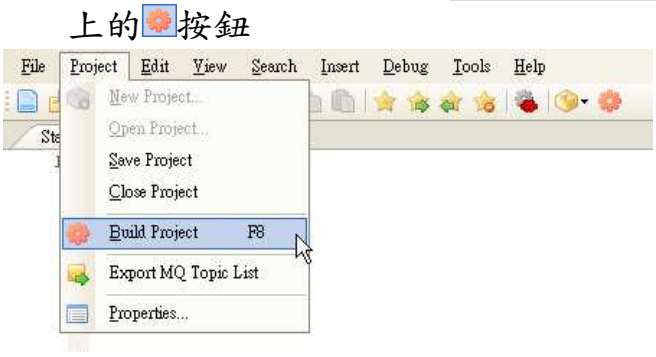
4.7 編譯專案

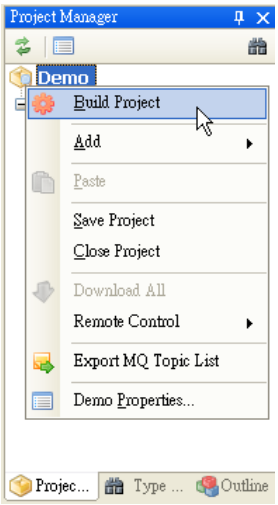
4.7.1 編譯前的設定確認

- 定義編譯屬性
  - 進行編譯前請確認專案屬性與每一個子專案屬性已正確設定無誤 (如：QP-500 的 IP、QPAC 的 IP 等設定)
  - 相關的屬性設定方式請參考前面章節
- 若應用環境中將與 Flash HMI 連結互動，必須先指定與 Flash HMI 通訊對照的專案路徑，Script 執行時才能與 Flash HMI 互相通訊。
- ***每次修改變更 Script Editor 或 Flash Editor 的專案請務必重新編譯以更新所有 QPAC 組態記錄，並下載至各個 QPAC 進行組態變更。***

4.7.2 編譯步驟

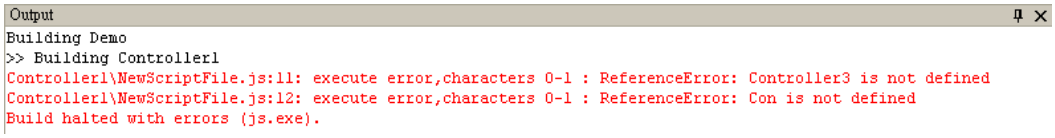
- 執行編譯(Build)
  - 方法一：從功能表下按 **Project > Build Project** 或是工具列上的按鈕



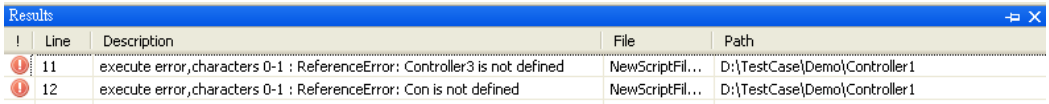


  - 方法二：以快速鍵[Ctrl+F]執行編譯
  - 方法三：在專案管理員視窗中，點選專案的節點，按滑鼠右鍵顯示右鍵選單，由右鍵選單中點選[Build Project]

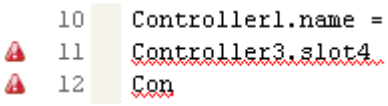
- 執行編譯專案，Editor 將自動儲存編輯的檔案
- 編譯時相關編譯訊息將顯示於輸出視窗



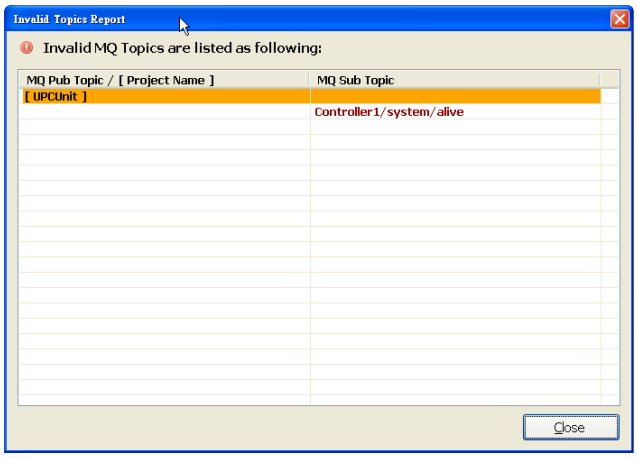
- 發生錯誤時，錯誤訊息、檔案、行數將條列於結果視窗



- 同時， 在發生錯誤的程式碼也將以紅色的波浪形底線標示。



- 點選結果視窗上的錯誤訊息，雙擊滑鼠左鍵，可自動切換至 Editor 上錯誤的程式碼所在位置
- 編譯成功後，將自動檢視 Flash HMI 專案所設定的資料通道，若發現不存在或是錯誤的資料通道設定時，將跳出視窗列表顯示錯誤的對照設定



[註]：

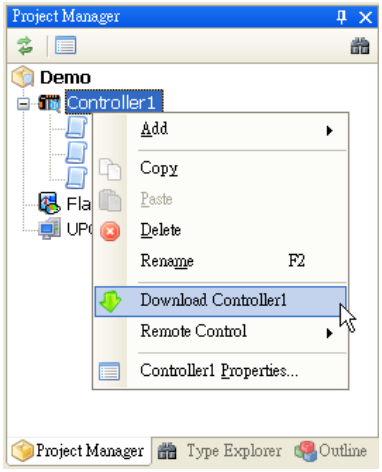
在進行專案編譯時，遠端控制命令（Ex: start、suspend）將顯示灰階狀態，禁止使用。

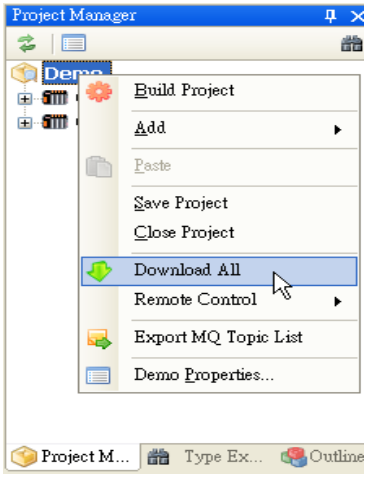
4.8 下載專案

- 確認專案下所有子專案都已編譯成功，若尚未完成請先執行 4.7 的步驟編譯專案。
- ***若 QPAC 間設定將互相通訊(資訊相互分享)，則修改任何一個檔案（即使只更動其中一個 QPAC 子專案檔案，未改變其他檔案）重新編譯後，務必以 [Download All] 更新所有的 QPAC 組態記***

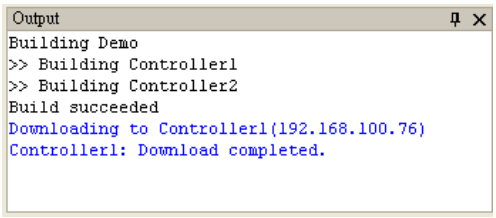
錄。如此，所有 QPAC 才能完整獲得更新後的組態，以免進行通訊時發生錯誤。

- 選擇下載方式
  - 下載單一 QPAC 子專案的檔案到遠端 QPAC 硬體上：在專案管理員視窗中，選擇 **QPAC 子專案**的節點，點選並按滑鼠右鍵，顯示右鍵選單，由右鍵選單中按下[Download ${QPAC 專案名稱}]（ex:Download Controller1）





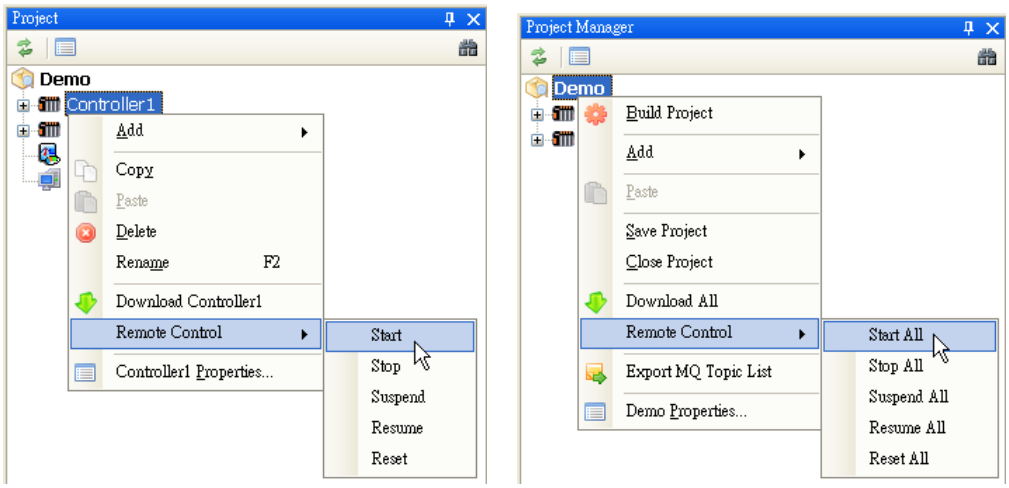
  - 將專案下所有的 QPAC 子專案檔案下載至遠端各個 QPAC 設備：在專案管理員視窗中，選擇**專案**的節點，點選並按滑鼠右鍵，顯示右鍵選單，從右鍵選單中按下[Download All]。
- 連線至遠端 QPAC 的訊息將顯示於輸出視窗中



- 檔案下載完成後，將存放於遠端 QPAC 設備的 Script Runtime 資料夾下(/System_Disk/Tools/SmartQ/)

4.9 遠端 啟動 / 停止 / 暫停 / 回復 專案

- 確認專案的檔案已全部成功下載至 QPAC。
- 選擇執行命令的目標
  - 單一 QPAC 子專案下的程式：在專案管理員視窗中，選擇欲執行的 QPAC 子專案節點，點選並按滑鼠右鍵，顯示右鍵選單。由右鍵選單中按下 **Remote Control > Start**，即可啟動遠端 QPAC 執行先前所下載的程式



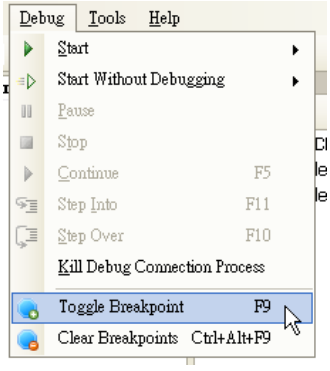
- 專案下所有程式：在專案管理員視窗中，選擇專案節點，點選並按滑鼠右鍵，顯示右鍵選單。由右鍵選單中按下 **Remote Control > Start All**，即可啟動所有的遠端 QPAC 設備，執行各個 QPAC 先前所下載的程式

- Stop、Suspend、Resume、Reset 等功能： Remote Control 下其他與遠端執行相關的功能操作方式與 Start 相同:
  - Stop(停止)：停止 script 執行
  - Suspend（暫停）：讓遠端的 QPAC 設備暫時停止 script 執行。
  - Resume（回復）：暫停執行後回復執行，script 繼續執行因暫停而未完成的工作。
  - Reset(重開機)：使用此命令重新啟動 QPAC 設備，回復為初始開機狀態。

4.10 線上專案偵錯

4.10.1 設定中斷點

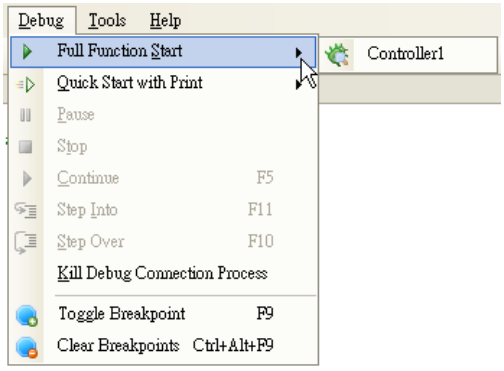
- 確認專案的檔案已全部成功下載至 QPAC，且遠端的 QPAC 設備正常運作。
- 游標移至預設中斷點的該行程式上，按下[F9]或是從功能表下按 **Debug > Toggle Breakpoint**
- 完成中斷點設定時，該行程式左側將出現此標記：● 。當此行程式碼再度被執行前，editor 將在此處暫停。
- 若要清除已設定的中斷點，可在該行程式碼

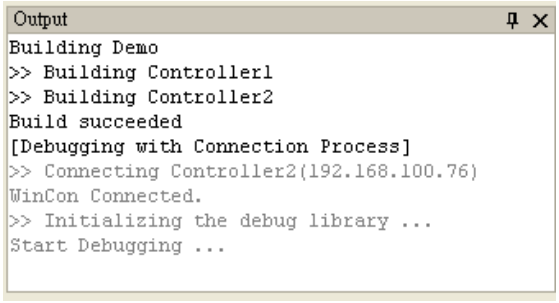


按下[F9]即可清除該行中斷點，或是由功能表點選 **Debug > Clear Breakpoint** 清除所有的中斷點。

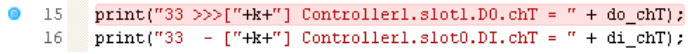
4.10.2 偵錯執行模式

- 若要讓程式中斷偵錯，請先確認在欲中斷的程式上已設定中斷點。
- 將滑鼠移動到功能表上 **Debug > Full Function Start** ，將出現專案下所有 QPAC 專案名稱，選擇要進行偵錯的 QPAC 專案。

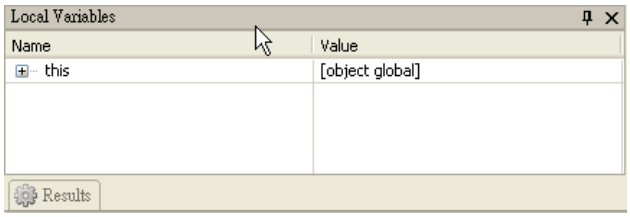




- 此時輸出視窗將顯示 Script Editor 與遠端 QPAC 專案進行連線的訊息。
- 若連線成功，將執行 Script 程式並停止於設定中斷點的程式碼，該行程式碼將以背景紅色的樣式標示其為程式暫停的中斷點。



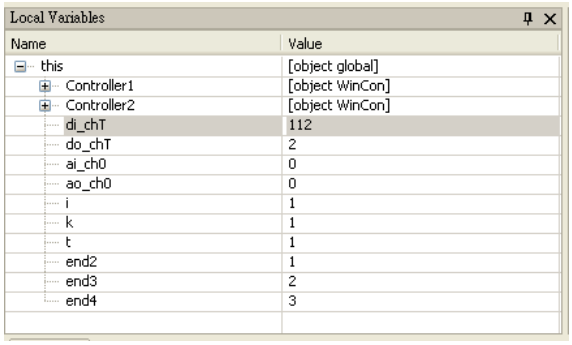
- 此時區域變數視窗將顯示所使用的變數及相對應的變數值。



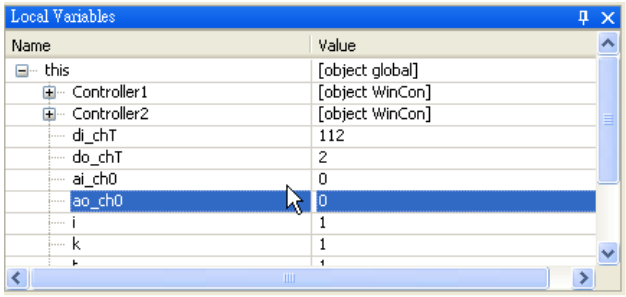
[註]：

區域變數視窗中將自動顯示一個「this」變數，代表正在執行程式所在的範圍（Scope）變數。若是在函式中，this 則為該函式範圍。

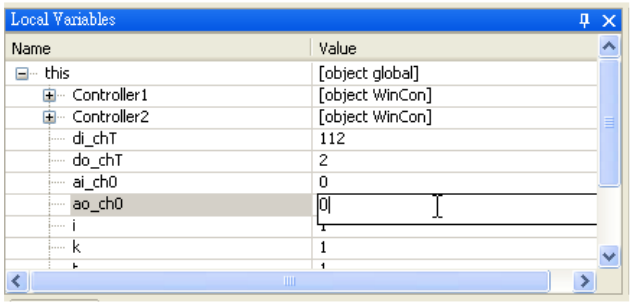
- 若變數左側有⊞標記，可按下標記顯示該變數下所包含的下一層變數值。



- 變數的值為數字或字串時，可在對應的右側[Value]欄位上變更其值。變更步驟為點選該行，該行將呈現反白，在對應的[Value]欄位上，點擊滑鼠左鍵。



- 該對應的[Value]欄位將呈現可輸入的狀態，此時可以更改其值。待程式重新執行時，遠端程式將更新此變數值。若其輸入狀態無法更改，表示該屬性無法編輯。

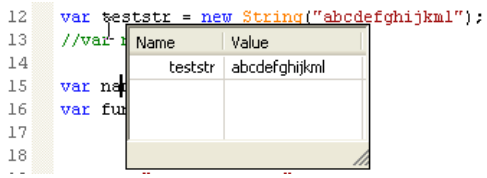


- 進入偵錯模式時，可以以 Tool bar 上五個按鈕控制偵錯進度。由左至右分別為

  - 暫停（Pause）：當程式執行時，若未設定中斷點，以此按鈕暫時停止程式的執行。
  - 終止（Stop）：停止並結束偵錯程序。
  - 繼續執行 (Continue)：繼續執行程式，直到遇到中斷點才停止。
  - 逐步執行(Step Into)：一次執行一行程式碼，若遇到函式呼叫，則進入函式中一行一行執行。

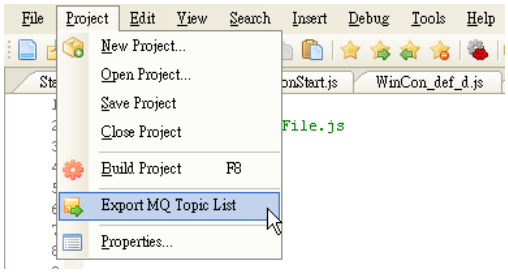
- 跨步執行(Step Over)：與逐步執行相似，一次執行一行程式碼。相異之處為遇到函式時，會將整個函式視為一行敘述，不進入函式內部執行。

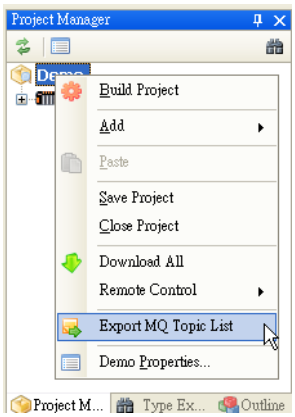
- 在偵錯模式時，如果程式碼已經被執行過，且變數為非函式變數名稱，當滑鼠移到該變數時也能將浮動顯示出該變數的值。



4.11 匯出資料通道清單

- 確認專案的檔案已全部編譯成功。
- 開啟匯出清單對話方塊
  - 方法一：由功能表選取 **Project > Export MQ Topic List**

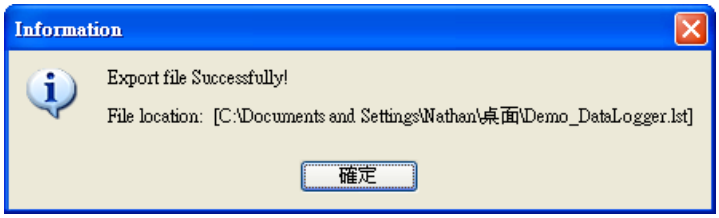




  - 方法二：在專案管理員視窗點選專案的根節點，按滑鼠右鍵，顯示右鍵選單。

- 在檔案瀏覽對話方塊中，選擇一個存放路徑後，按下 確定 。

- 若匯出成功將顯示對話方塊告知儲存位置與檔名，檔名為[${專案名稱}_DataLogger.lst]。(ex：Demo_DataLogger.lst)



- 匯出的資料通道清單可由 SmartQ NetDB 資料庫系統匯入，並讓使用者依照資料的類別將資料通道分成數個群組與設定警報條件。SmartQ NetDB 資料庫系統則會依照設定將傳送的資料儲存並記錄在資料庫中。(參考 SmartQ NetDB 快速上手手冊)

5.輕鬆上手六步驟

本章節將以一個簡單的實例來示範 SmartQ Script Editor 的設定與操作流程。藉由本範例的介紹使用者將可初步的瞭解 SmartQ Script Editor 的操作步驟。此專案範例位於光碟目錄：Example\Demo1，請將此專案完整複製到 C:\ICPDAS\SmartQ\Example\Demo1 目錄下，再開啟 Script Editor。

5.1 案例描述

本案例使用兩個控制器，每一個控制器上各有一組 DI/O 與 AI/O 模組。DO 模組與 DI 模組間各通道(Channel)對接，AI 模組與 AO 模組間各通道(Channel)對接，而類比資料是以電壓模式輸出，輸出電壓採用預設。範例使用的設備如下：

設備	型號	說　明
PAC	QP-842	QPAC 系列可程式自動控制器, 4 插槽
DO 模組	I-8056W	DO x 16 通道, 安裝在 QP-842 的插槽 0
DI 模組	I-8051W	DI x 16 通道, 安裝在 QP-842 的插槽 1
AO 模組	I-8024W	AO x 4 通道, 安裝在 QP-842 的插槽 2
AI 模組	I-8017HW	AI x 8 通道, 安裝在 QP-842 的插槽 3
PC 或 NB		用於安裝 SmartQ Script Editor
電源供應器	DP-665	+10V ~ +30VDC

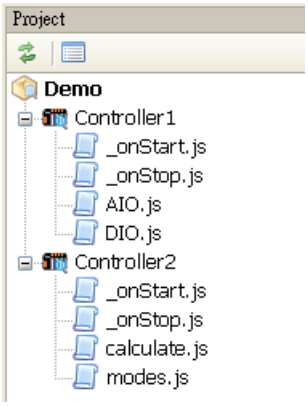
本專案範例中兩個控制器彼此的運作是各自獨立，專案內每一個控制器所執行的工作內容與目的如下表：

QPAC 名稱	Controller1
工作內容	DO 或 DI 其輸出或接收的資料為累加的狀況，由 0 累加至 65535，使模組上的燈號以 2 進位模式亮燈； AO 或 AI 其輸出或接收的資料亦為累加的狀況，由 0v 累加至 10v，每次遞增 0.5v
程式目的	瞭解函式宣告、如何取得模組的現值、以及不同 Script 間如何同時獨自運作且不影響彼此
QPAC 名稱	Controller2
工作內容	使一個變數由 1 開始以 2 的次方增長，程式將依次方運算值來

	決定 DO 模組的工作模式(此範例為 16 種：0～15 工作模式)，並使模式對應的 DO channel 輸出訊號。而相對應的 DI channel 將收到此訊號，並列印出模式代號。而 DI 與 DO 的模組上，其相對應的燈號也將亮起，每次僅亮一個燈號； 依照 DI 上的 channel 訊號去控制 AO 上的 channel 輸出電壓伏特數。
程式目的	瞭解如何透過 Script 來控制模組間的連動

5.2 案例實作

5.2.1 步驟 1 – 建立新專案



由於此案例中使用兩個控制器，因此在專案管理員視窗中必須在專案目錄下針對這兩個控制器各自建立一個 QPAC 子專案。專案結構如左圖。每一個控制器下除了預設的_onStart.js 和_onStop.js 外，將再各自增加兩個 Script 檔案，每個 Script 檔案屬性皆設定為 Timer：500ms。（請參考步驟 4.3與步驟 4.4）

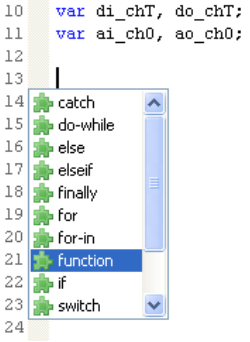
5.2.2 步驟 2 – 撰寫 script 程式碼

藉由 Auto-Completion 與關鍵字的快捷建立功能，可以幫助使用者在開發 Script 程式邏輯時，減少重複輸入相同的字元，且避免造成與文字輸入相關的錯誤。下面將針對檔案程式一一介紹，使用者可嘗試輸入程式內容。

- Controller1 程式簡介

如範例所見，我們將在 Controller1/_onStart.js 檔案內建立兩個新的函式，名稱各為 DO_AddValue 與 AO_AddValue。在 DO_AddValue 中會對 do_chT 這一個變數一直做加 1 的動作。當超過 65535 時，就將其值歸零重新開始計算。

在專案管理員視窗上選取_onStart.js，並按兩下開啟檔案。先將四個變數（di_chT、do_chT、ai_ch0、ao_ch0）以 var 做變數宣告，接著移到下一行開頭，按下快速鍵[Ctrl+K]啟動關鍵字輸入，在此選擇 function 關鍵字，將建立一個新的函式範本。



```
13 function () {
14
15 }
```

在游標駐點閃爍的位置中，輸入函式的名稱 DO_AddValue，若函式要宣告參數，則只需寫上參數名稱後以「,」分開即可，如：function name(arg1, arg2) {}。接下來加入邏輯陳述，做漸進加 1 的運算。

```
13 function DO_AddValue()
14 {
15     do_chT++;
16
17     if (do_chT > 65535)
18         do_chT = 0;
19     return;
20 }
```

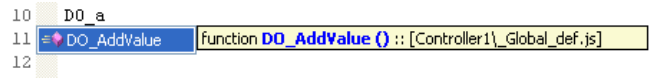
完成新函式之後請務必**儲存檔案**。儲存檔案後，DO_AddValue 函式名稱才可於編輯 Script 檔時加入 Auto-Completion 的輔助列表中。因此，編輯檔案時，若希望重複使用先前所編輯的函式，可先儲存檔案，即可由 Auto-Completion 功能提供的輔助列表輸入此函式。AO_AddValue 函式的宣告程序與上述函式相似。與 DO_AddValue 的差別只在每次增加 0.5，以 10 為最高上限。

Controller1/_onStart.js 完整的程式內容，請參考 Example 5-1。

Example 5-1. Controller1/_onStart.js

```
1  /**
2  * FileName: _onStart.js
3  * Author:
4  * Version:
5  * Description:
6  *
7  * Date: 2009/3/11
8  */
9
10
11 var di_chT, do_chT;
12 var ai_ch0, ao_ch0;
13
14 function DO_AddValue()
15 {
16     do_chT++;
17
18     if (do_chT > 65535)
19         do_chT = 0;
20     return;
21 }
22
23 function AO_AddValue()
24 {
25     ao_ch0 = ao_ch0 + 0.5;
26
27     if (ao_ch0 > 10.0)
28         ao_ch0 = 0;
29     return;
30 }
31
32 do_chT = 0;
33 di_chT = 0;
34 ai_ch0 = 0;
35 ao_ch0 = 0;
```

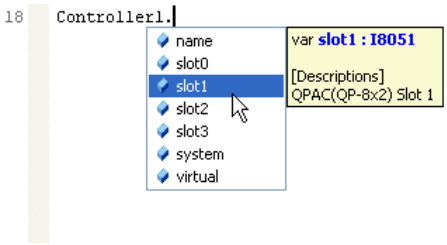
在 Controller1/DIO.js 中，我們將呼叫 DO_AddValue()函式對儲存 DO 值的變數做加一，與存取模組資料。開啟 DIO.js，在游標處輸入 DO 兩個字元，由於之前已將 Controller1/_onStart.js 編輯好並儲存。所以當



輸入這兩個字元時將觸發 Auto-Completion 的功能。當輸入了 DO_a 時便可在輔助列表中找到 DO_AddValue 函式。

為了讓開發者在開發系統時能有一個直觀且能符合實際硬體架構的操作概念，在 SmartQ 的開發環境中，定義了一個控制器硬體模組的資料模型，此資料模型以階層架構展現（請參考附錄 A.3）。在 Script Editor 撰寫 Script 時，必須參照此模型，按照階層依序定義要存取的控制器硬體模組資料。

以本個案為例，使用者輸入 “Con” 即可由輔助選單點選 Controller1 將其加入程式。由於 DO 模組在 Slot1，當我們在 Controller1 後面輸入「.」時，即可顯示在此 QPAC 上的模組介面（如：slot1）及插在介面上的模組名稱（如：I8051 模組）。選擇模組介面後，再次輸入「.」，將顯示模組上的 I/O 模式（如:DO）。如此反覆定義，直到最底層的 Channel 值，便可得到”Controller1.slot0.DO.chT”的字串。當 Script runtime 執行此一字串時，將可即時取得該硬體模組的所有 DO channel 值。



在此特別說明：chT 即是代表該 DO 模組所有 digital channel 個數的 bit 數組合值，故直接使用 chT 或單獨操作 channel 的意義是相似的。假設有一個 DO 模組的 channel 個數是 4 時，chT 則是 4 個 bit 的 2 進位值。當所有 channel 都為 true 時，chT 值則為 1111（10 進位的 15）。

Controller1/DIO.js 與 Controller1/AIO.js 的程式內容，請參考 Example 5-2 及 5-3。在 Example 5-2 第 12 行的程式中，以 do_chT 變數儲存 DO 模組將要輸出的資料，將此值設定至 DO 模組 chT 屬性，使 DO 模組依照儲存的 chT 值輸出訊號。再以 di_chT 來取得 DI 模組上得到的訊號值。並在遠端 QPAC Script Runtime 的介面上分別將此兩變數以 print 函式列印其值（請參考附錄 D）。5-3 的概念亦同。

Example 5-2. Controller1/DIO.js

```
/**
* FileName: DIO.js
* Author:
* Version:
* Description:
*
* Date: 2008/10/2
*/

DO_AddValue();

Controller1.slot0.DO.chT = do_chT;
print("DIO >>>> Controller1.slot0.DO.chT = " + do_chT);

di_chT = Controller1.slot1.DI.chT;
print("DIO ---- Controller1.slot1.DI.chT = " + di_chT);
```

Example 5-3. Controller1/AIO.js

```
/**
* FileName: AIO.js
* Author:
* Version:
* Description:
*
* Date: 2008/10/2
*/

AO_AddValue();

Controller1.slot2.AO.ch0 = ao_ch0;
print("AIO >>>> Controller1.slot2.AO.ch0 = " + ao_ch0);

ai_ch0 = Controller1.slot3.AI.ch0;
print("AIO ---- Controller1.slot3.AI.ch0 = " + ai_ch0);
```

Controller2 程式簡介

在 Controller2 的程式中，由於並無任何函式將提供_onStart.js 以外的 Script 檔案共用，在此只需利用 var 宣告兩個變數，用以儲存 2 的次方值。Controller2/_onStart.js 的程式如 Example 5-4。

Example 5-4. Controller2/_onStart.js

```
/**
* FileName: _onStart.js
* Author:
* Version:
* Description:
*
* Date: 2009/3/11
*/

var count = 1;
var ans = 0;
```

Example 5-5. Controller2/calculate.js

```
1  /**
2  * FileName: calculate.js
3  * Author:
4  * Version:
5  * Description:
6  *
7  * Date: 2009/3/4
8  */
9
10 if( count != 0 )
11    count = count % Math.pow(2, 16);
12
13 /*
14  * when the ans is the 16th powers of the number two,
15  * it must be set to one, but after mod operation it is zero.
16  */
17
18 //if (count == 0)
19 //    count = 1;
20
21 Controller2.slot0.DO.chT = count;
22 count = count * 2;
```

Controller2/calculate.js 此檔案將進行 2 的次方計算。在 Example 5-5 第 10 行的程式中，利用系統預設的 Math 物件函數 pow()取得次方值，pow()函數具兩個參數，分別為基底與冪數，函數將取得 2 的 16 次方值，因此設定 pow()函數的基底為 2、冪數為 16。再取得 count 變數值做 2 的 16 次方 Mod 運算後的餘數值。在第 21 行的程式中，將餘數值設定給 DO 模組的 chT，決定要輸出訊號的 channel。

Controller2/modes.js 依據 DI 模組收到的實體訊號，判斷哪個 digital channel 的值為真，用以決定列印的模式代號與 AO 模組在 channel0 上輸出的伏特值。由這兩個 Script 檔案將可瞭解如何操作模組上 channel 實體訊號的輸入與輸出。

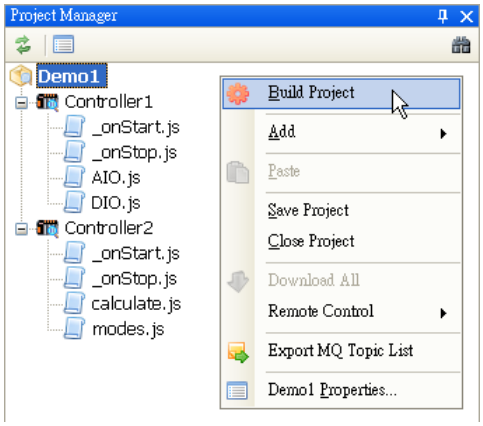
Example 5-6. Controller2/modes.js

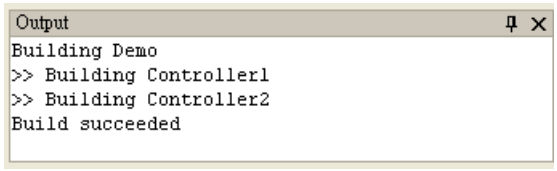
```
/**
* FileName: modes.js
* Author:
* Version:
* Description:
*
* Date: 2009/3/4
*/

if( Controller2.slot1.DI.ch0 )
  { print("mode 0");
    Controller2.slot2.AO.ch0 = -5;   }
else if( Controller2.slot1.DI.ch1 )
  { print("mode 1");
    Controller2.slot2.AO.ch0 = -4;   }
else if( Controller2.slot1.DI.ch2 )
  { print("mode 2");
    Controller2.slot2.AO.ch0 = -3;   }
else if( Controller2.slot1.DI.ch3 )
  { print("mode 3");
    Controller2.slot2.AO.ch0 = -2;   }
else if( Controller2.slot1.DI.ch4 )
  { print("mode 4");
    Controller2.slot2.AO.ch0 = -1;   }
else if( Controller2.slot1.DI.ch5 )
  { print("mode 5");
    Controller2.slot2.AO.ch0 = 0;    }
else if( Controller2.slot1.DI.ch6 )
  { print("mode 6");
    Controller2.slot2.AO.ch0 = 1;    }
else if( Controller2.slot1.DI.ch7 )
  { print("mode 7");
    Controller2.slot2.AO.ch0 = 2;    }
else if( Controller2.slot1.DI.ch8 )
  { print("mode 8");
    Controller2.slot2.AO.ch0 = 3;    }
else if ( Controller2.slot1.DI.ch9 )
  { print("mode 9");
    Controller2.slot2.AO.ch0 = 4;    }
else if( Controller2.slot1.DI.ch10 )
  { print("mode 10");
    Controller2.slot2.AO.ch0 = 5;    }
else if( Controller2.slot1.DI.ch11 )
  { print("mode 11");
    Controller2.slot2.AO.ch0 = 6;    }
else if( Controller2.slot1.DI.ch12 )
  { print("mode 12");
    Controller2.slot2.AO.ch0 = 7;    }
else if( Controller2.slot1.DI.ch13 )
  { print("mode 13");
    Controller2.slot2.AO.ch0 = 8;    }
else if( Controller2.slot1.DI.ch14 )
  { print("mode 14");
    Controller2.slot2.AO.ch0 = 9;    }
else if( Controller2.slot1.DI.ch15 )
  { print("mode 15");
    Controller2.slot2.AO.ch0 = 10;   }
else
  { print("all off" );                }
```

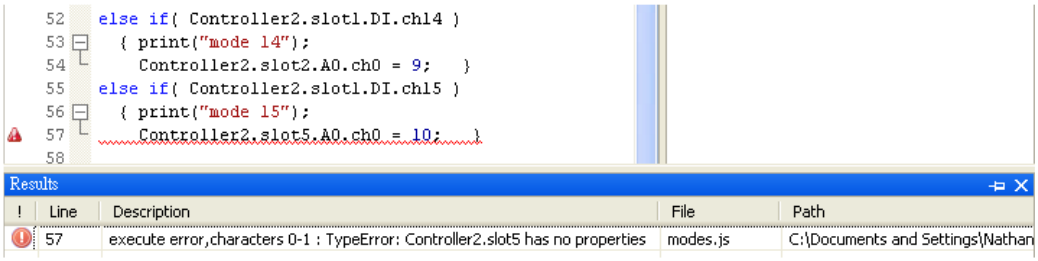
5.2.3 步驟 3 – 編譯專案

由於此案例的設定為兩台 QPAC 控制器各自獨立執行 Script，並未與 Flash HMI 介面通訊。因此無須確認相關的通訊設定，可直接進行編譯(請參考步驟 4.7)。在專案管理員視窗中，選擇專案的節點，點選並按滑鼠右鍵，顯示右鍵選單，由右鍵選單中按下[Build Project]。若程式編輯時無錯誤（如打字錯誤等），在輸出視窗上將出現編譯成功的訊息(Build succeeded)。

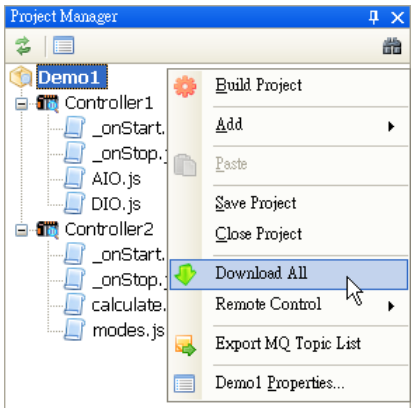




若出現輸入錯誤，如：將 Controller2/modes.js 中第 57 行誤植為”Controller2.slot5.AO.ch0 = 10”，進行編譯後將立即於結果視窗顯示錯誤訊息，第 57 行發生錯誤處將以紅色波浪線標示出。使用者可依照提示修改正確的值。



5.2.4 步驟 4 – 下載專案



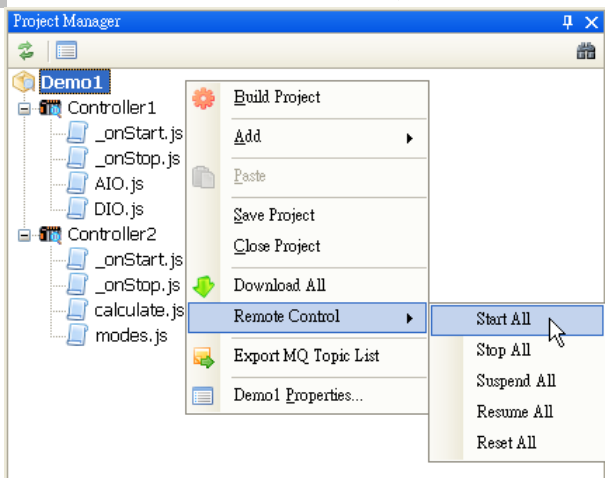
再將 Script 檔案下載至遠端的 QPAC 上，由於此次為初次下載(請參考步驟 4.8)，必須下載全部。在專案管理員視窗中，選擇專案的節點，點選並按滑鼠右鍵，顯示右鍵選單，從右鍵選單中按下[Download All]。

若下載成功，則在輸出視窗上將出現下載成功的訊息。若發現下載等候時間過長或是出現無法連線的錯誤訊息時，請先檢查網路線是否鬆脫、QPAC 是否已成功取得連線的 IP、QPAC 子專案的 IP 是否有設錯、QPAC 是否正常運作、QPAC 上的 Script Runtime 是否正常運作。

5.2.5 步驟 5 – 執行專案

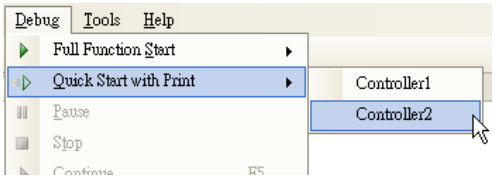
在專案節點的右鍵選單點選[Start All]，便可同時使兩台 QPAC 上的 Script Runtime 執行下載後的 Script 檔案。詳細操作步驟請參考步驟 4.9。此時使用者應可發現在 Controller2 上，DI 與 DO 模組的面板燈號依序輪流亮燈一次後便停止，並未按照預期的功能持續依序亮燈。

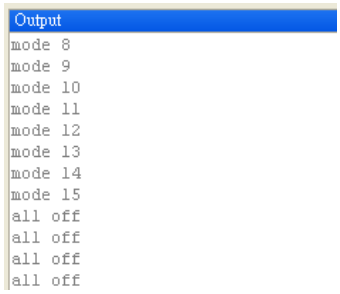
為了進入偵錯模式釐清錯誤發生處，需先停止正在執行 Script 的 Controller2。在 Controller2 節點的右鍵選單點選 Remote Control > Stop，停止程式執行，以便準備讓 Contorller2 進入 Debug 模式。



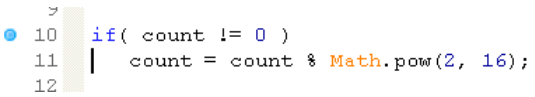
5.2.6 步驟 6 – 對專案進行執行中線上偵錯

為了瞭解可能的錯誤原因，我們先觀察 Controller2 所輸出的訊息。將滑鼠移到功能表上 **Debug > Quick Start with Print**，此時將出現專案下所有 QPAC 子專案名稱，選擇要進行偵錯的 Controller2。此時 Controller2/Modes.js 會將模式代號以 print()函式列印出來，在輸出視窗將依序列印模式代號，當列印完"mode 15"後，將反覆重複列印"all off"的訊息。再按下 tool bar 的停止按鈕 停止輸出，將執行狀態回復為編輯狀態。



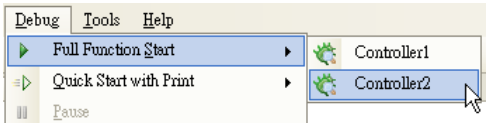


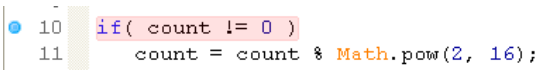
接著開啟 Controller2 專案下的兩個 Script 檔案 Modes.js 與 Calculate.js。由於 Modes.js 檔案為依照輸入訊號判斷輸出模式，發生錯誤的可能性較小，而設定燈號的程式碼主要位於 Controller2\Calculate.js 第 21 行，出錯的原因極有可能為 count 值未改變。



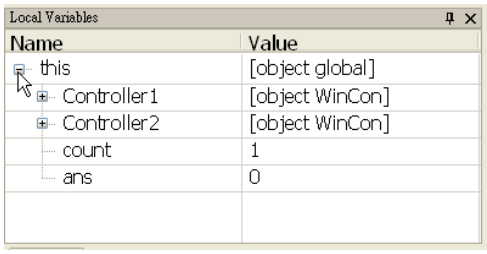
為了確認 count 值的變化，我們將游標移到 Controller2\Calculate.js 的第 10 行，按下[F9]設定中斷點，此時在第十行前將會出現藍色的中斷點。（請參考步驟 4.10.1）

在功能表上點選 **Debug > Full Function Start**，將出現專案下所有的 QPAC 子專案名稱，點選將要進行偵錯的 Controller2。





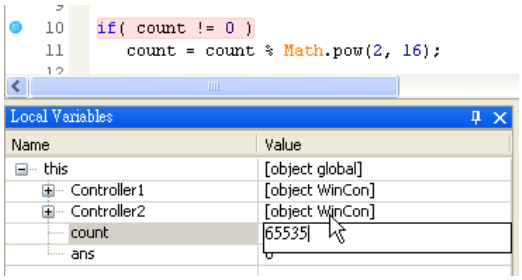
此時程式開始進入 Debug 模式，並中斷於 Calculate.js 的第 10 行，該行將以粉紅色背景反白標示。同時在區域變數視窗中，顯示出系統變數 this，按下加號⊞展開 this 的內容，可看到 count 變數的值此時為 1。（請參考步驟 4.10.2）



再按下 tool bar 上的繼續執行按鈕 繼續執行，再次執行至第 10 行中斷，此時 count 將變為 2。如此重複執行將可看到 count 值以 2 的次方漸次增長至 65536，此時改用跨步執行(Step over)使偵錯器一行一行依序執行，使用者將可確認每一行的執行結果。

使用者應可發現：當 count 為 65536 時，執行第 11 行的 mod 運算將歸為 0，此錯誤造成第 22 行進行乘法運算時，無法再成為 2 的次方。按照預設情況，此時 count 必須歸為 1，才能夠繼續做次方變化。

為了確認此一發現的問題，讓程式執行停在中斷點第 10 行上，並將滑鼠移到呈現展開變數狀態的區域變數視窗，將 count 該行反白，在對應的[Value]欄位上，點擊滑鼠左鍵。該對應的[Value]欄位則會呈現框格為黑邊的輸入狀態，將 count 值改成 65536 後，重新一步一步執行觀察 count 變化，確認確實為 count 為 0 所造成的錯誤。



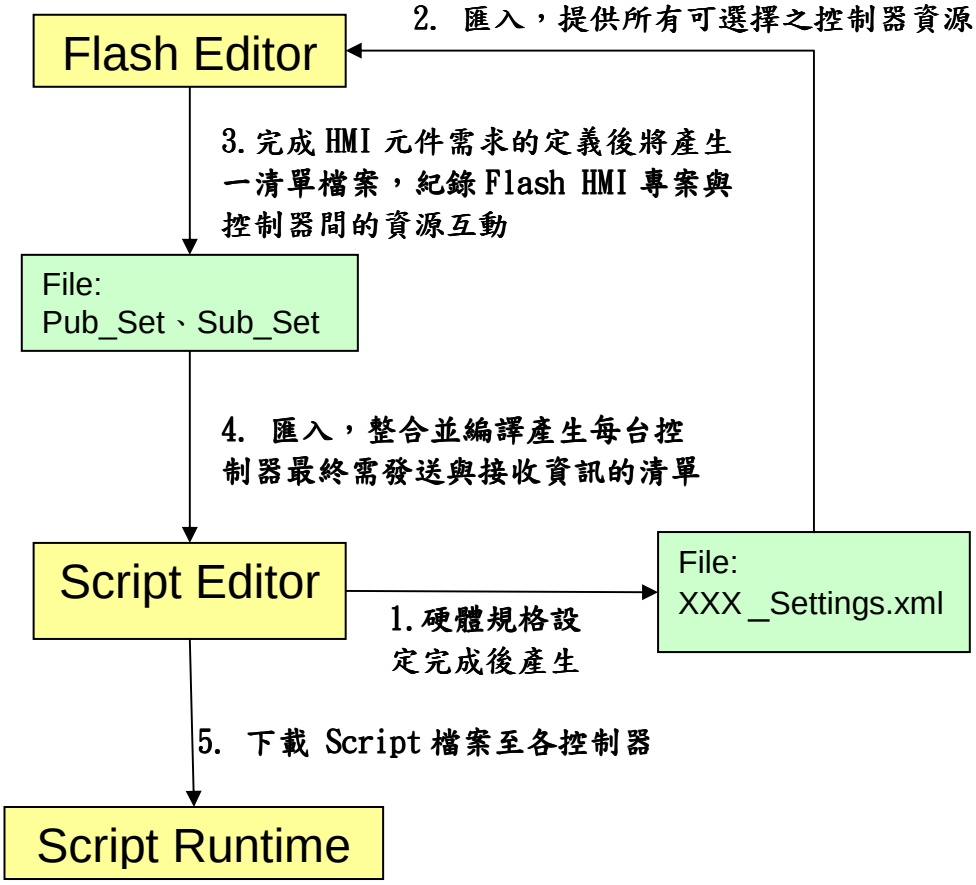
為了避免歸零的狀況再度發生，必須在 mod 運算後，所得的值為 0 時，就需將 count 值重設為 1。將 Controller2\calculate.js 的第 18、19 行的兩個反斜線(/)刪除，解除註解狀態。重新編譯下載執行後，便可看到 Controller2 的面板燈號一個接著一個重複亮起。

Appendix A
SmartQ 專案設計流程與重要觀念

A.1 Script Editor 與 Flash Editor 的資訊整合

SmartQ 主要設計概念為協助控制器做控制邏輯的運算及工控系統人機介面的設計與整合，因此 SmartQ 的 Script Editor 與 Flash Editor 必須藉由一定的流程，整合彼此的資訊需求，如此遠端的控制器與人機介面設備才可依照需求，互相發送與接收資訊。

整體使用流程分別如下圖所示：



1. SmartQ Script Editor 定義 QP-500 網路位址及硬體的規格與屬性，Script Editor 將產生與硬體相對應的資訊檔，名稱為 XXX _Settings.xml。（參考步驟 4.2與步驟 4.3）
2. 在 Flash Editor 建立 HMI 元件時，必須瞭解現階段控制器上可供使用的 I/O 模組資訊。故使用者在 Flash Editor 開發專案時，需

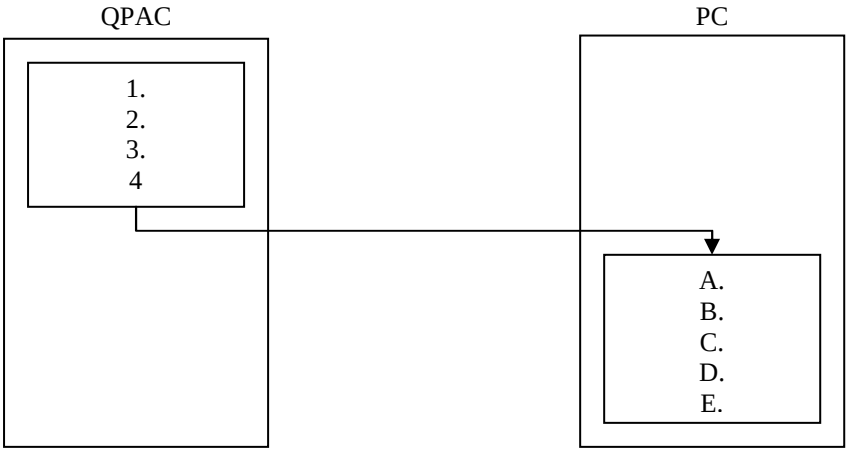
在專案設定中，設定 Topic 檔路徑，將此 XML 檔匯入。 （參考 SmartQ Flash HMI tools 入門手冊 6.2）

3. Flash Editor 在完成 HMI 元件的通訊設定後，將產生兩個分別紀錄接收與發送的資訊需求檔。
4. 此時再透過 Script Editor 專案管理員視窗，定義外部 Flash Editor 專案路徑，匯入 Flash Editor 所需要參考資訊需求。（參考步驟 4.6）
5. 最後將 Script Editor 的專案編譯後，以 Script Editor 將 Script 檔與相關資訊設定檔下載至遠端控制器上供 Script Runtime 執行。

A.2 硬體 I/O 模組組態設定

在控制器上有部份 I/O 模組必須藉由其他設定工具設定模組組態，並產生參考檔案，以便 Script Editor 依照設定產生對應的物件變數。以下將依照各種可能的狀況，介紹組態設定步驟與流程。

- i-8K 組態設定

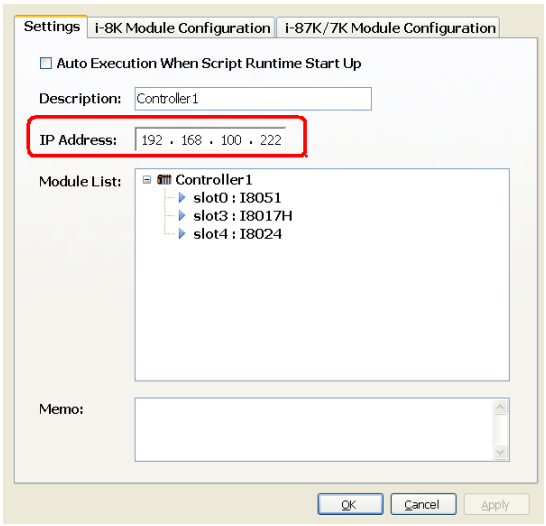


QPAC 端：

1. 插槽插上 i-8K 模組
2. 連結 Ethernet 網路
3. 開啟電源
4. 檢查是否已取得正確的 IP 位址

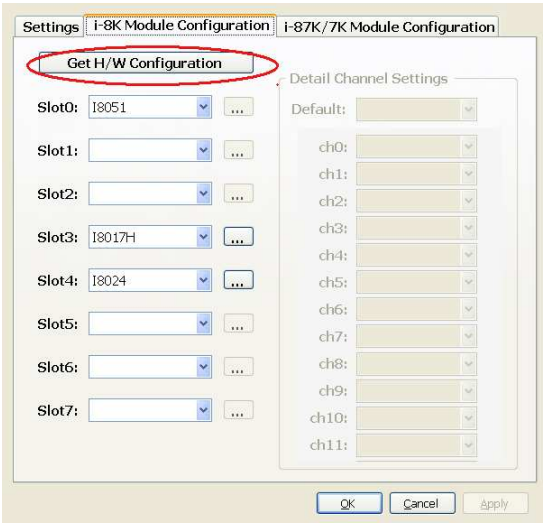
PC 端：

A. 執行 Script Editor 程式
B. 新增專案，新增 QPAC 控制器
C. 開啟 QPAC 屬性設定對話視窗，設定控制器 IP 位址

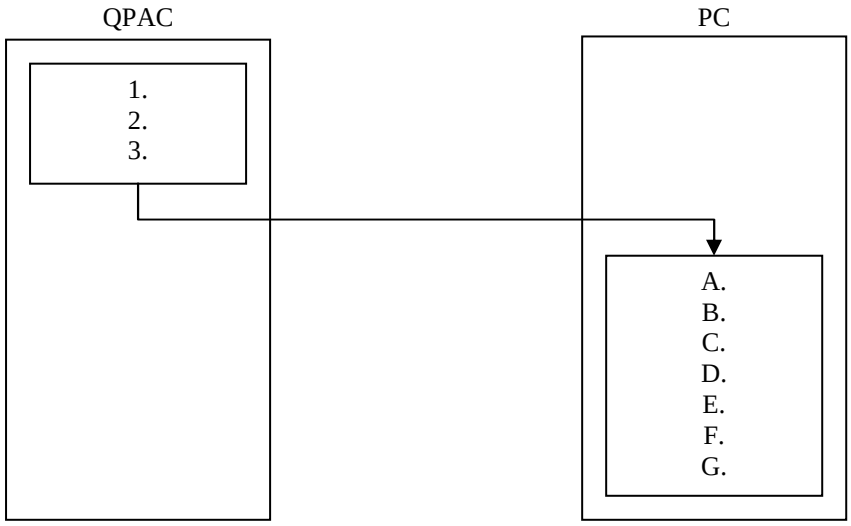


D. 切換[i-8K Module Configuration]頁籤

E. 按下[Get H/W Configuration]掃描取得控制器上的 i-8K 模組列表，或手動點選 I/O 模組

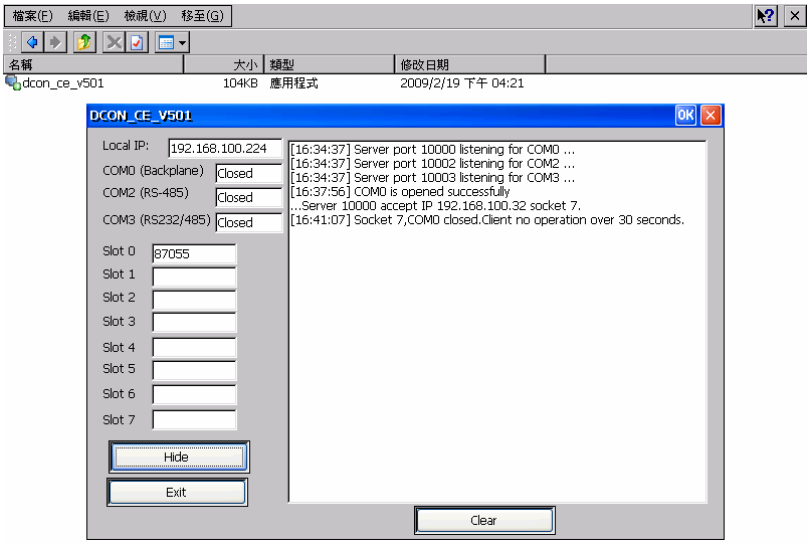


Slot(COM0)上的 i-87K 組態設定



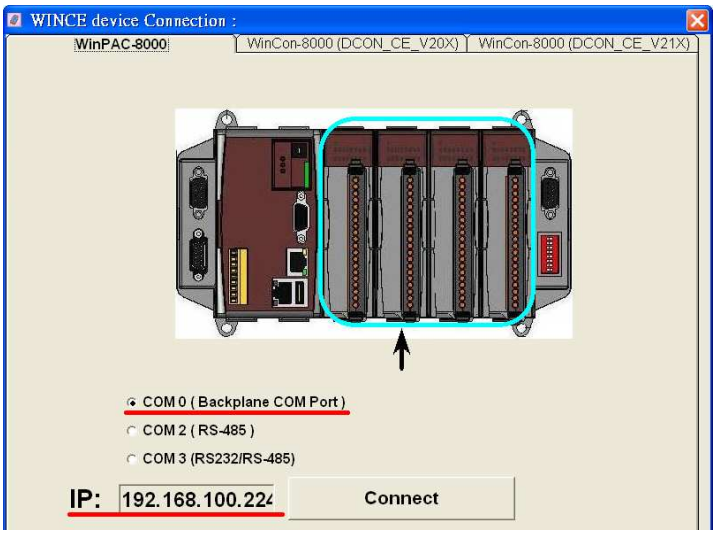
QPAC 端：

1. 插槽接上 i-87K 模組，連結 Ethernet 網路
2. 開啟電源，檢查是否已取得正確的 IP 位址
3. 執行 dcon_ce_v501.exe 等待 PC 端連結(檔案位於 System_Disk\tools\dcon_ce\下)

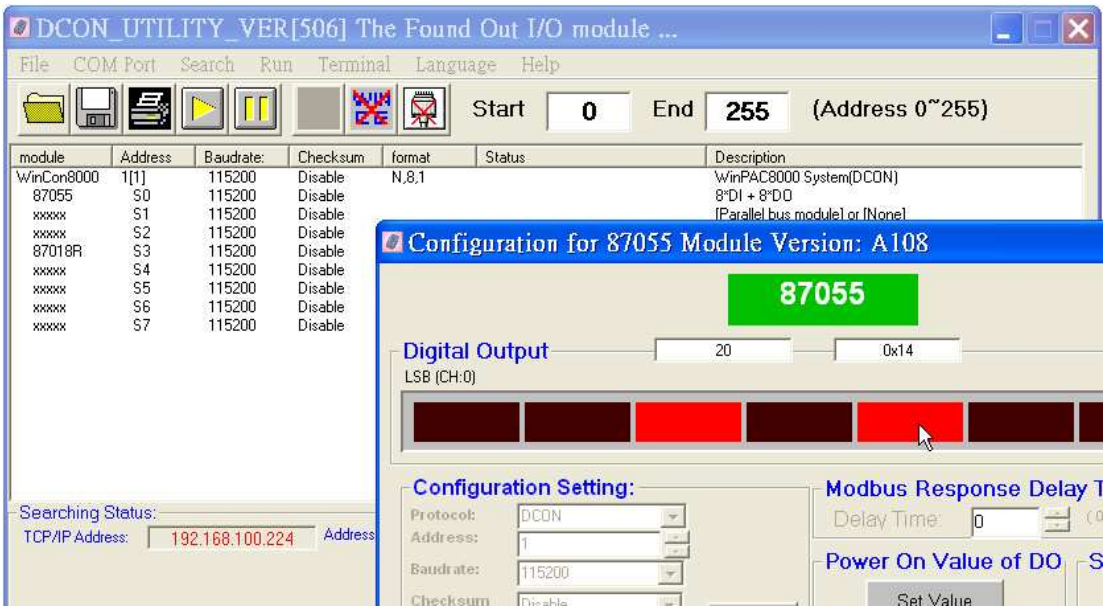


PC 端：

A. 安裝並執行 DCON Utility 5 程式
B. 按下[WinCE device connection]
C. 在[WinPAC-8000]頁籤內，設定 QPAC 之 IP 位址
D. 選擇透過 COM 0 方式掃描，按下[Connect]

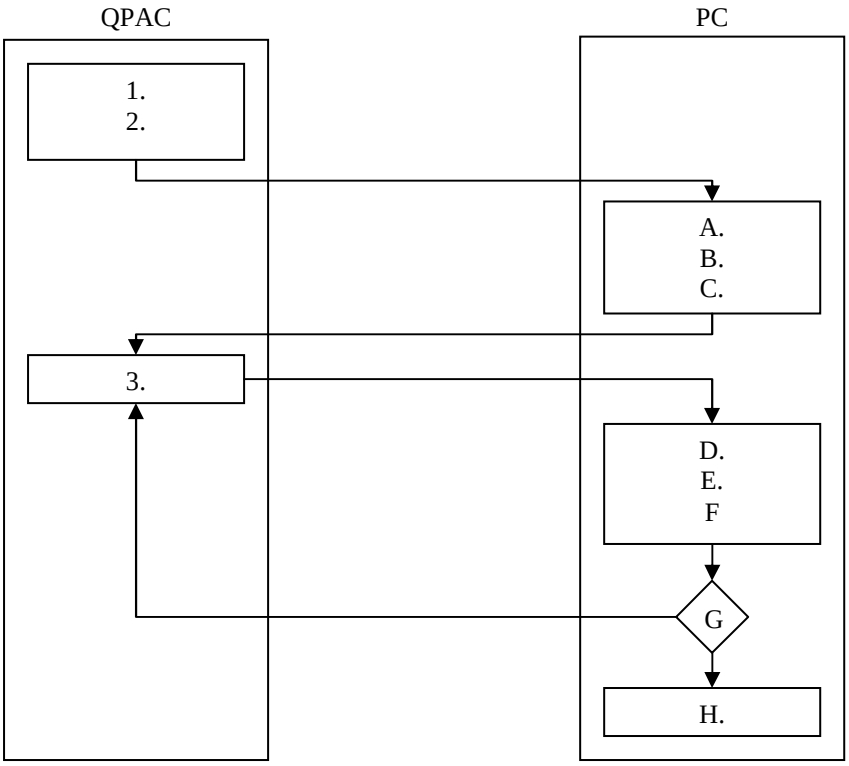


E. 若成功將列出找到的 i-87K 模組
F. 可直接點擊模組，設定初始值



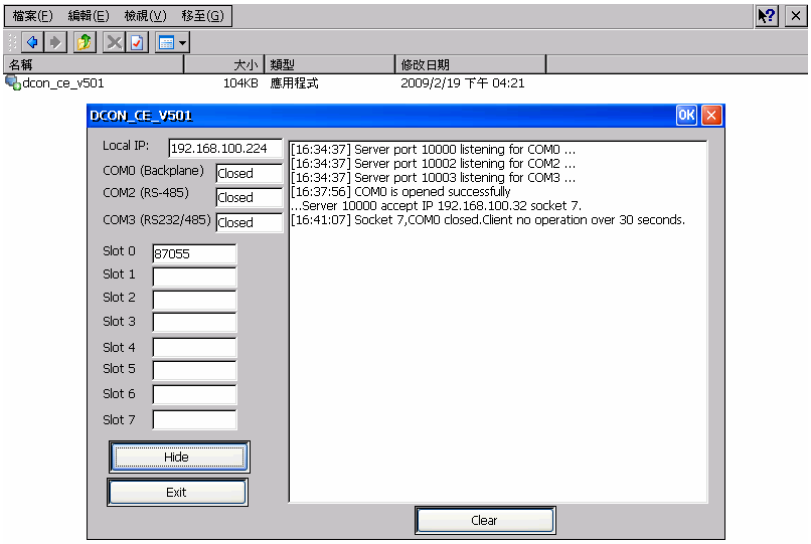
G. 將此結果儲存為 xxx.map 檔，可供 Script Editor 設定 i-87K COM0 模組時使用

COM2 上的 i-87K/7K 系列組態設定



QPAC 端：

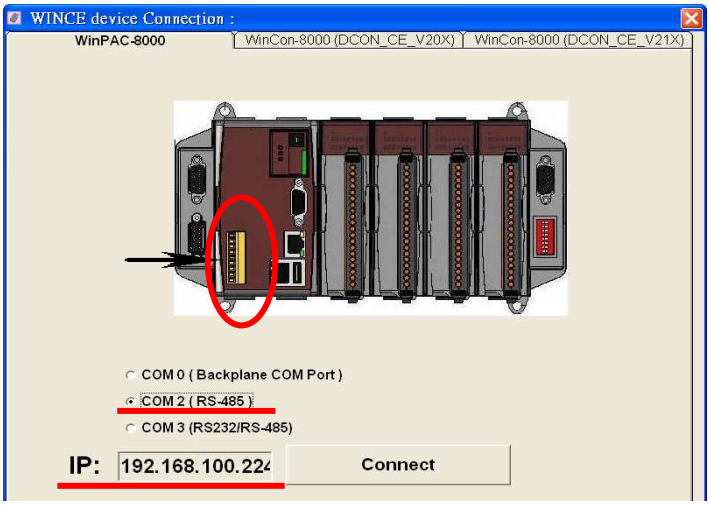
1. 連結 Ethernet 網路，開啟電源，檢查是否已取得正確的 IP 位址
2. 執行 dcon_ce_v501.exe，等待 PC 端連結



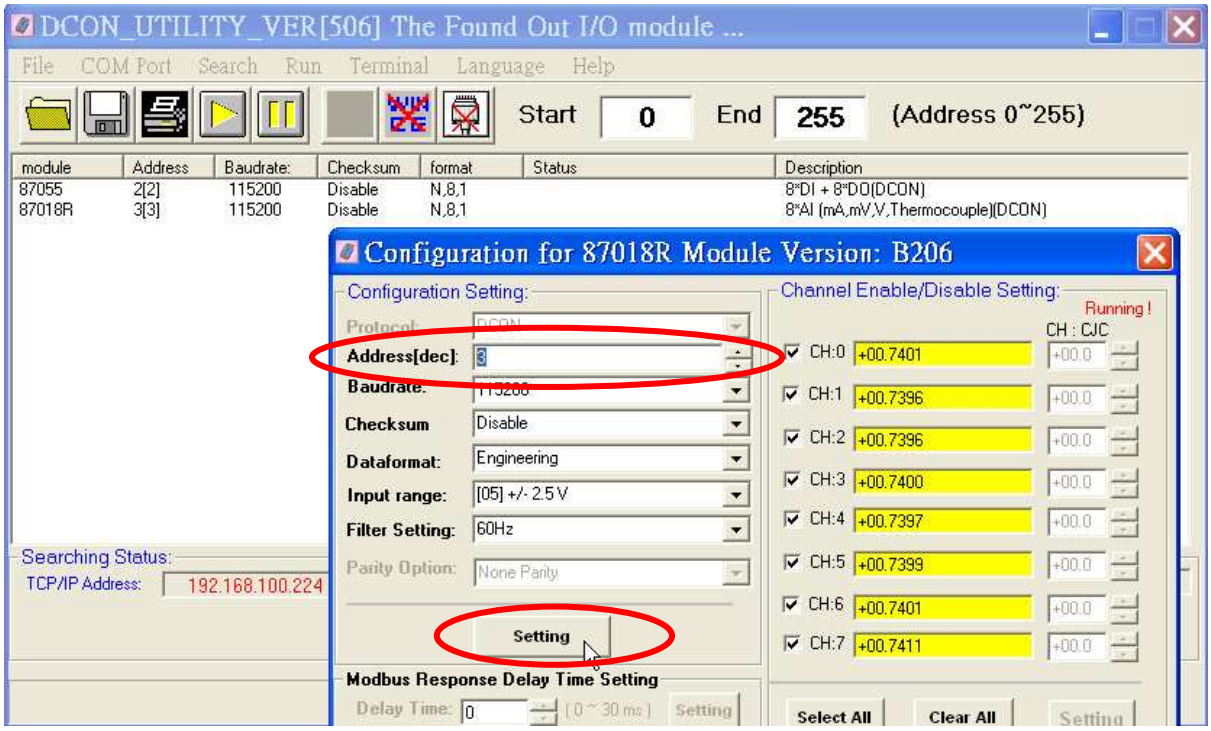
3. 透過 COM2，連接 i-87K4 擴充單位，在其上接 i-87K 模組或透過 COM2 連接 i-7K 模組(一次僅新增一個)

PC 端：

A. 執行 DCON Utility 5 程式
B. 按下[WinCE device connection]
C. 在[WinPAC-8000]頁籤內，設定 QPAC 之 IP 位址
D. 選擇透過 COM 2 方式掃描，按下[Connect]



E. 若成功即會列出找到的 i-87K/i-7K 模組
F. 直接點擊模組，設定 ID address(不可重複)

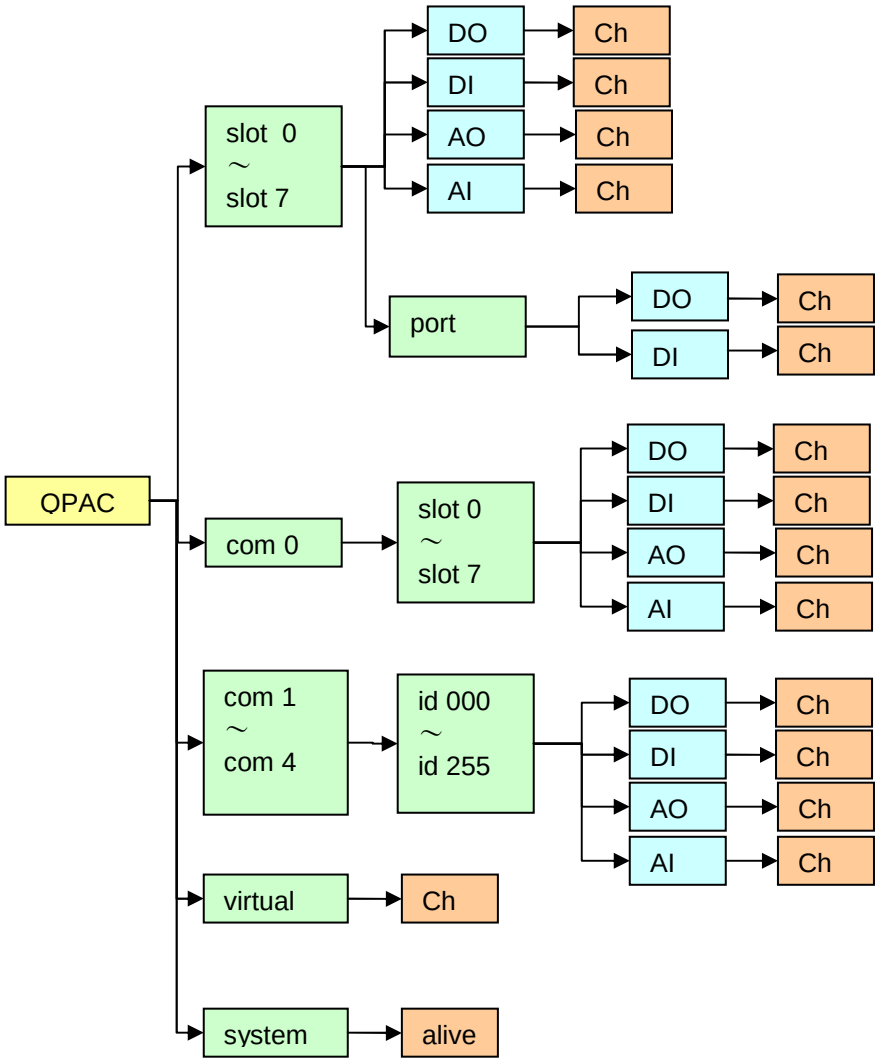


G. 之後再新增一個模組，再重複掃描一次，同上步驟

H. 最後，將結果儲存為 xxx.map 檔，可供 Script Editor 之 COM1/2/3/4/模組設定之用

A.3 階層式控制器物件資料模型

為了使 Script Runtime 執行 Script 檔案時，直觀地應用控制器上所提供的 I/O 模組資訊；並且使控制器與 Flash HMI 交換硬體模組的控制訊號。SmartQ 開發工具針對控制器及 I/O 模組硬體結構，定義了 Script 所能操作並且用來與 Flash HMI 溝通的階層式控制器物件資料模型。撰寫 Script 時，可以參考此資料模型，層層定義存取硬體資訊；也可以參考此資料模型定義 Flash HMI 與控制器溝通時所需要設定的資料通道。以 QPAC 為例，整個階層概略如下圖所示：



A.4階層式控制器物件資料模型使用範例

- QPAC 控制器 I/O 模組實體資料通道
  - Mapping to i-8K/i-87K/i7000 modules (DI/DO/AI/AO)
    - QPAC1.slot1.DO.chT = 255 (Total channel value)
    - QPAC1.slot7.port0.DO6.chT = 32 (for I-8172, FRnet modules)
    - QPAC1.slot7.port1.DI8.chT
    - QPAC1.com0.slot2.DI.ch0
    - QPAC1.com0.slot2.CNT.ch2 (for DI Counter function)
    - QPAC1.com2.id006.LATCH.hi (for DI Latch function)
    - QPAC1.com3.id255.ALARM.clear (for AI Alarm function)

- QPAC 控制器虛擬資料通道
  - 變數型態可為數字或字串
  - 一台 QPAC 控制器可提供 20 個虛擬 channel 供使用者撰寫 Script 檔案用

- QPAC1.virtual.ch0 = 1.234
- QPAC1.virtual.ch0 = “Hello world”

- QPAC 控制器系統資料通道
  - 提供 QPAC 控制器是否已連線的資訊(read-only)
    - QPAC1.system.alive (0：此控制器未連線，1：此控制器已連線)

Appendix B Script 語法簡介

SmartQ Script Editor 所依循的為 ***JavaScript*** 語法，JavaScript 為目前世界上最被廣泛使用且容易上手的 Script，尤其在網頁程式的開發上更是如此，因此我們選用 JavaScript 作為 SmartQ Script Editor 在開發 Script 檔案及 Script Runtime 執行 Script 檔案時的依據。由於 JavaScript 的被廣泛的應用，使用者可輕易取得相關的資源及範例作為參考。以下章節將簡要說明 JavaScript 語法，使用者也可由網路取得更多的資訊。

B.1 概要

任何語言最基本的概念為型態 (type)。Script 程式可以改變「值」(value)，而這些值各自有其歸屬的型態。Script 所支援的型態有：

- Number（數字）
- String（字串）
- Boolean （布林）
- Function （函式）
- Array （陣列）
- Null（空）
- Undefined（未定義）

B.2 數字

Script 數字為「雙精確度 64 位元格式 IEEE 754 值」("double-precision 64-bit format IEEE 754 values")，再加上 Script 在設計上並無所謂的整數。這可能造成在計算帶有小數的數字時會產生一些如下的狀況：

```
0.1 + 0.2 = 0.30000000000000004
0.1 + 1.1 = 1.2000000000000002
```

Script 支援標準的數字運算子，包含加法、減法、取餘數等算術。此外尚有一個內建物件：Math（數學），可用以處理較為進階的數學函數和常數：

```
Math.sin(3.5);
d = Math.PI * r * r;
```

將特殊的值 NaN（「Not a Number」，「非數字」的簡稱）做任何數學運算，結果仍為 NaN：

```
> NaN + 5
NaN
```

內建的 isNaN() 函式可判斷一個值是否為 NaN：

```
> isNaN(NaN)
true
```

特殊的值 Infinity（無限）以及 -Infinity（負無限）：

```
> 1 / 0
Infinity
> -1 / 0
-Infinity
```

B.3 字串

Script 中的字串為一個有序排列的 Unicode 字元，每個字元皆以一個 16 位元表達的數字作為索引。一個單一字元可以以長度為 1 的字串表達。要得知一個字串的長度，可以透過該字串的 length（長度）屬性取得：

```
> "hello".length
5
```

字串也是物件，本身即具備一些方法 (method)：

```
> "hello".charAt(0) //位置 0 的字元
h
> "hello, world".replace("hello", "goodbye") //把 hello 換成 goodbye
goodbye, world
> "hello".toUpperCase() //轉成大寫
HELLO
```

B.4 其他類型

在撰寫 Script 時，需特別注意下列兩者的分別：null（空），屬於「object」類型的物件，用以表示無數值；以及 undefined（無定義），屬於「undefined」類型的物件，用以表示未初始化的值，也就是說，尚未指定數值。你可以宣告一個變數但不指定其值，該變數的型態便是 undefined。

Script 有布林 (boolean) 型態，用以表達邏輯的 true（真）與 false（假）而此兩數字皆為關鍵字。根據下列規則，任何值都可以被轉換為布林值：

1. false、0、空字串 ("")、NaN、null、以及 undefined 皆為 false
2. 所有其他的值皆為 true

可以呼叫 Boolean() 函式進行強制轉換：

```
> Boolean("")
false
> Boolean(234)
true
```

實際上，較少藉由 Boolean()函式強制轉換為布林值，因為若 Script 遇到需要接收布林值的情況，如 if 陳述式（見下面 B.7 結構控制），便會執行布林型態自動轉換。Script 可支援布林運算，如 &&（邏輯「*與*」，英稱 *and*）、||（邏輯「*或*」，英稱 *or*）、以及 !（邏輯「*非*」，英稱 *not*），請見下面章節介紹。

B.5變數

在 Script 語法中，要宣告新變數，使用的是 var 關鍵字：

```
var a;
var name = "simon";
```

若宣告一個變數但不指定任何值，其型態便為 undefined（未定義）。

B.6運算子

Script 的數字運算子有 「+」、「-」、「*」、「/」、以及 「%」 - 最後一個符號「%」為取餘數的運算子（英稱 *mod*），此外，用以指定值的運算子為 「=」。另外尚有複合指定陳述式，如 「+=」 以及 「-=」，用以延伸及簡化 「x = x *運算子* y」。

```
x += 5
x = x + 5
```

「++」 和 「--」分別為增加或減少數值。這些運算子可以放在變數的開頭或結尾。
「+」運算子也能把字串連接 (concatenate) 起來:

```
> "hello" + " world"
hello world
```

若你把一字串加到一個數字（或其他型態數值如：字串、布林值等），所有的相加物件將先轉為字串再做字串的串連。因此，把一個空字串加進一個物件便可輕鬆轉換字串，不失為一個轉換字串的好方法。

```
> "3" + 4 + 5
345
> 3 + 4 + "5"
75              ==>先運算 3+4，再將運算式轉換成 7 + "5"
> "" + 4
4               ==>轉換為字串"4"
```

Script 中的比較運算子為：「<」、「>」、「<=」、以及 「>=」。這些運算子可使用於字串或數字型態的運算元。等值比較 (equality) 具兩種運算子，分別為：「＝＝」與「＝＝＝」。雙等號運算子（等於）將進行型態強制轉換，若比較的資料型態不一樣，將產生意想不到的情形：

```
> "dog" == "dog"
true
> 1 == true
true            ==>數字 1 強制轉換成布林值 true，接著進行等值比較運算
```

要避免型態強制轉換，要用三等號運算子（絕對等於）：

```
> 1 === true
False         ==>數字 1 不強制轉換成布林值 true，運算則因型態不同為 false
> true === true
true
```

此外尚有 != （不等於）以及 !== （絕對不等於）運算子。

B.7控制結構

Script 與其他同屬 C 家族的程式語言有類似的控制結構。條件陳述式主要以 if 及 else 來表示，或組合使用 else if：

```
var name = "cat";
if (name == "dog") {
  name += "!";
} else if (name == "cat") {
  name += "!!";
} else {
  name = "!" + name;
}
name == "cat!!"
```

Script 也有 while 迴圈以及 do-while 迴圈。前者適合做基本的迴圈，若至少執行一次或先執行再判斷，適合使用後者：

```
while (true) {
  // 無限迴圈！
}

do {
  // 無限迴圈！
} while (true)
```

Script 的 for 迴圈跟 C 和 Java 的一樣，可根據判斷式進行迴圈內容的重複運作。

```
for (var i = 0; i < 5; i++) {
  // 會執行 5 次
}
```

「&&」 以及 「||」 運算子為「短路邏輯」(short-circuit logic)，也就是說，第二個運算元是否會被執行將視第一個運算元而定。當存取一個物件的屬性前檢查物件是否為空 (null) 時，此運算非常有用。如下面的例子，若變數 o 為空(null)時，&&的第一個運算元將會強制轉型為 false，所以第二個運算元將不被執行運算，直接傳回 false 並指定給變數 name：

```
var name = o && o.length;
```

Script 也有三元運算子 (tertiary operator)，可以用以書寫單行的條件陳述式，如下面的例子，若 age 大於 18 將回傳 true；若小於或等於 18 則回傳 false：

```
var allowed = (age > 18) ? true : false;
```

switch 陳述式可以根據一個數字或字串做不同決定：

```
switch(action) {
    case 'draw':
        drawit(); //開始畫
        break; //中斷
    case 'eat':
        eatit(); //開始吃
        break; //中斷
    default: //預設
        donothing(); //不做任何事
}
```

break （中斷）陳述式為中斷程式，若未加 break，執行的程式碼將繼續往下執行下一個 case。一般來說，在 switch case 時不加 break 相當少見。若真要如此，最好加上註解說明，以方便除錯：

```
switch(a) {
    case 1: //繼續往下執行 case 2 內容
    case 2:
        eatit(); //開始吃
        break; //中斷跳出 switch
    default: //預設
        donothing(); //不做任何事
}
```

default 子句 (clause) 是選擇性的，用來處理未被列舉出來的 case 情境，如上面的例子，除了 a 等於 1 或 2 這兩個被特別列舉出的情況，其他情況都會執行 default 子句內

的程式碼。而且使用時可以在 switch 部分或 case 部分放表達式 (expression)，如下例的 1+3 和 2+2 兩個表達式，兩者之間的比較使用的會是 === 運算子：

```
switch(1 + 3):
    case 2 + 2:
        yay(); //耶！
        break; //中斷
    default: //預設
        neverhappens(); //根本不會發生
}
```

B.8 陣列

Script 的陣列其為一特殊的物件，具有「length」（長度）的屬性。此屬性數值為陣列最高索引數加上一，因為陣列的索引是從 0 開始算起。建立陣列的舊方法如下：

```
> var a = new Array();
> a[0] = "dog";
> a[1] = "cat";
> a[2] = "bird";
> a.length
3
```

使用陣列實體語法較為簡潔：

```
> var a = ["dog", "cat", "bird"];
> a.length
3
```

注意－－array.length 不一定為陣列中具有值的項目個數。比方說：

```
> var a = ["dog", "cat", "bird"];
> a[100] = "fox";
> a.length
101
```

以上範例的執行結果顯示：陣列的 length 就是最高索引數加一。
若查詢一個不存在的陣列索引，得到結果將為 undefined：

```
> typeof(a[110])
undefined
```

利用上述，便可在陣列上執行迴圈，如下例：

```
for (var i = 0; i < a.length; i++) {
    //處理 a[i]
}
```

然而這樣效率不佳，因為每迴圈一次便查詢一次 length 屬性。較好的做法為：

```
for (var i = 0, len = a.length; i < len; i++) {
    //處理 a[i]
}
```

若要在陣列結尾再新增項目，最安全的方法為：

```
a[a.length] = item;                    //同 a.push(item);
```

由於 a.length 必為最高索引數加一，可以假設所指定的必定是位在陣列結尾後的未用空間。
陣列內建有數種方法 (method)：

```
a.concat(item, ..), a.pop(), a.push(item, ..), a.slice(start, end),
a.sort(cmpfn), a.splice(start, delcount, [item]..), a.unshift([item]..)
```

- concat 結合，將傳回加入了新項目的新陣列
- pop 將移除最後一個項目並將其傳回
- push 將在結尾加入一個或多個項目（如同前面所提的 ar[ar.length] 方法）
- slice 將傳回副陣列
- sort 進行排序，可選擇性的接受「比較性函數」(comparison function)
- splice 讓你透過刪除陣列中一整個連續的項目個數並以其他項目來代替被刪除的連續項目
- unshift 在開頭加入一個或多個項目

B.9 函式

如同物件，函式 (function) 為 Script 的核心元件。最基本的函式相當簡單：

```
function add(x, y) {
    var total = x + y;
    return total;
}
>add(3, 4)
7
```

在此範例中顯示：一個 Script 函式可以接受 0 或多個具名 (named) 參數。函式內文 (body) 中的陳述式行數並無限制，且可以宣告於函式內部的區域 (local) 變數。return 陳述式可以在任何時候傳回一值並終止函式。如果沒有 return 陳述式（或者僅 return，卻無數值），Script 便會傳回 undefined。
具名參數比較像是做為參考，但非強制性的一定要設值。你可以呼叫一個函式但不提供其要求的參數，因此傳入的便是 undefined。以上面的 add()函數為例，若沒有傳入參數值，如下面例子，參數 X 與 Y 將被強制轉型為 undefined，則得到的函式回傳值將因兩個運算元都是 undefined，無法進行加法運算，傳回 NaN。

```
> add()
NaN // undefined 不能進行加法
```

Appendix C 關鍵字(Keywords)

Script 有許多關鍵字，這些關鍵字在 Script 語法中都有特定的意義。因此，Script 檔案中自行定義的函式、變數、迴圈標籤或常數名稱，不宜使用這些關鍵字。以下為三類關鍵字：保留關鍵字、擴充保留關鍵字、其他要避免的識別字，這三類關鍵字建議避免使用。

此外，變數或函式的命名規則為第一個字母必須是 ASCII 字元(大小寫都可以)、底線字元(_)。接下來的字元可以是任何字母、數字、底線字元。注意，在 Script 中文字的大小寫是有分別的。

C.1 保留關鍵字

Break	do	if	switch	typeof
Case	else	in	this	var
Catch	false	instanceof	throw	void
continue	finally	new	true	while
Default	for	null	try	with
Delete	function			

C.2 擴充保留關鍵字

Abstract	double	goto	native	static
Boolean	enum	implements	package	super
Byte	export	import	private	synchronized
Char	extends	int	protected	throws
Class	final	interface	public	transient
Const	float	long	short	volatile
debugger				

C.3 其他要避免的識別字

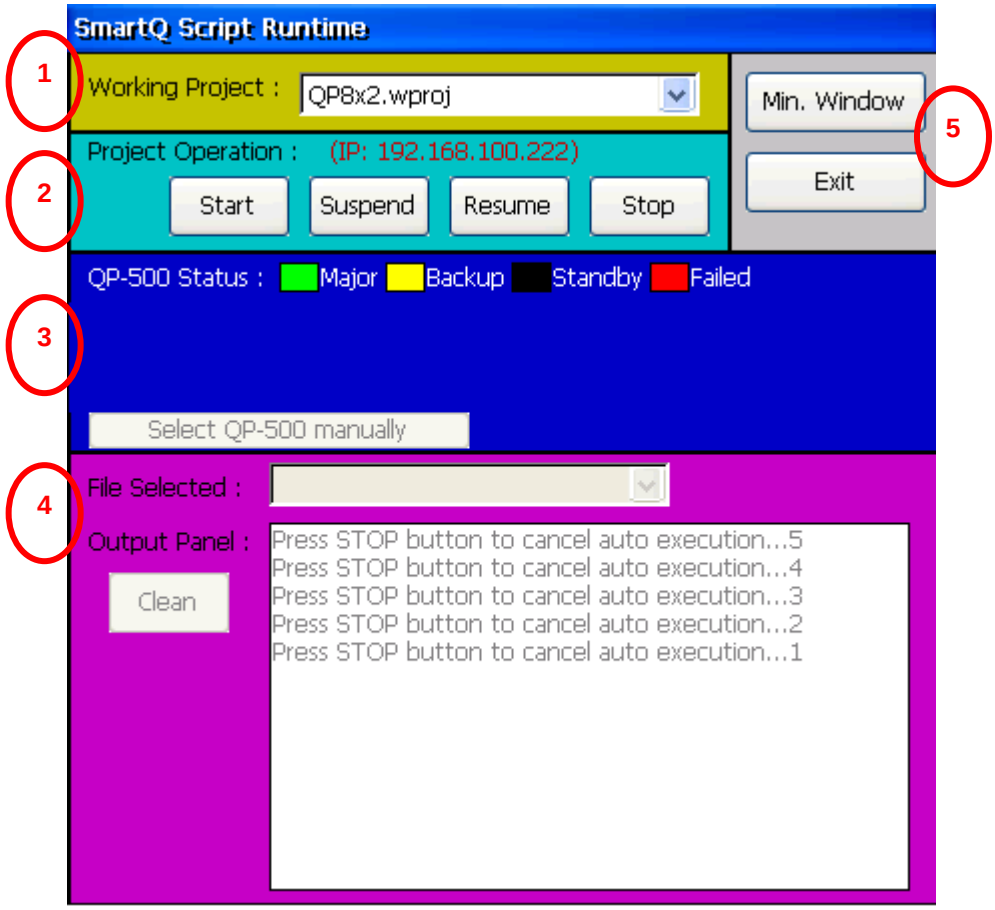
arguments	encodeURI	Infinity	Object	String
Array	Error	isFinite	parseFloat	SyntaxError
Boolean	escape	isNaN	parseInt	TypeError
Date	eval	Math	RangeError	undefined
decodeURI	EvalError	NaN	ReferenceError	unescape
decodeURIComponent	Function	Number	RegExp	URIError
WinCon	AI_8	AI_16	AO_4	DI_4
DI_8	DI_16	DI_32	DO_4	DO_6
DO_8	DO_16	DO_32	COM0	COMn
FRnet0	FRnet1	I7011	I7012	I7013
I7014	I7016	I7017	I7018	I7021
I7022	I7024	I7041	I7042	I7043
I7044	I7045	I7050	I7051	I7052
I7053	I7055	I7058	I7060	I7063
I7065	I7066	I7067	I8017H	I8017HS

I8024	I8037	I8040	I8041	I8042
I8048	I8050	I8051	I8052	I8053
I8054	I8055	I8056	I8057	I8058
I8060	I8063	I8064	I8065	I8066
I8068	I8069	I8077	I8172	I87013
I87017	I87018	I87022	I87024	I87026
I87040	I87041	I87051	I87052	I87053
I87054	I87055	I87057	I87058	I87063
I87064	I87065	I87066	I87068	I87069
PTZ	System	virtual	cAI	cAI_2
cAI_4	cAI_8	cALARM	cAO	cAO_2
cAO_4	cCNT	cCNT_14	cCNT_16	cCNT_32
cCNT_4	cCNT_7	cCNT_8	cDI	cDI_14
cDI_16	cDI_32	cDI_4	cDI_7	cDI_7
cDI_8	cDO_13	cDO_16	cDO_2	cDO_3
cDO_32	cDO_4	cDO_5	cDO_7	cDO_8
cLATCH	fDI16	fDI18	fDI8h	fDO16
fDO8	uPAC			

Appendix D Script Runtime 操作介面解說

Script Runtime 在啟動之後，預設將最小化至系統列（system tray）中，如下圖所示。在紅框中的 icon 上雙擊後，便可開啟 Script Runtime 的操作介面。

以下將依照操作介面上的區塊分布，介紹各個元件的功能和使用方式。



D.1 專案選取介面（Project select interface）

此區塊為一個下拉式選單，使用者可選擇所要執行的專案。若開始執行所選擇的專案，此下拉式選單將呈現鎖定無法選擇的狀態，直到使用者停止執行後才能重新再次選擇欲執行的專案。

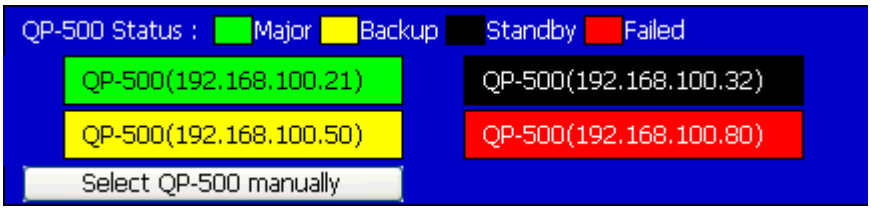
D.2 專案運作介面（Project operation interface）

此區塊將顯示此 Script Runtime 所在裝置的 IP，並具有四個功能按鈕：Start，Suspend，Resume，Stop。其功能分別如下：

- **Start**：啟動執行所選擇的專案。當專案被啟動之後，Start 按鈕將呈現 disable 的狀態以防專案再次被啟動。需待停止執行，或按下 Stop button 後，才可再度啟動。
- **Suspend**：暫停所執行的專案。
- **Resume**：回復並繼續執行先前所暫停的專案。
- **Stop**：停止執行所選擇的專案。停止專案需要一段時間與 QP-500 斷線，並進行一些參數的初始化，在這段時間內 Start 按鈕仍將處於 disable 的狀態，直到 Stop 的程序完成。

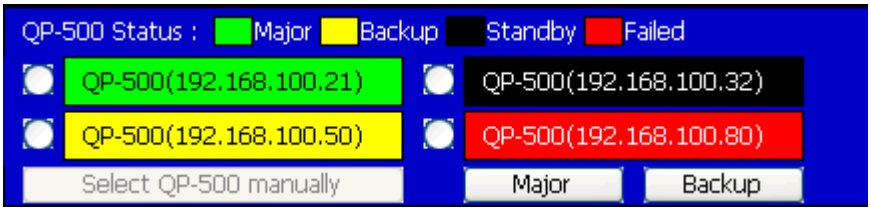
D.3 QP-500 狀態介面（QP-500 status interface）

此區塊用以顯示專案中所有通訊的 QP-500 連線狀況。在執行專案之後，此區塊將顯示使用者在 Scripr Editor 中編輯專案時所輸入的 QP-500，並以顏色表示該 QP-500 的狀態。介面如下圖所示：



上圖為一個包含四個 QP-500 的範例。Script Runtime 在執行時，將依照使用者在 Script Editor 中輸入 QP-500 的順序，決定備援(Redundant)功能啟動時，挑選 QP-500 的先後順序。當所有的 QP-500 都可以順利連線時，第一順位的 QP-500 將被定位為 major QP-500 (以淺綠色表示)，所有資料的傳輸都將透過 major QP-500 來進行。而第二順位的 QP-500 將成為 backup QP-500 (以黃色表示)，作為備援的第一順位。當 major QP-500 斷線時，backup QP-500 將在第一時間成為 major QP-500，以維持 SmartQ 系統中各控制器之間的連線。而其他連線正常的 QP-500 (以黑色表示)，則將依照順位再挑出下一個 backup QP-500。

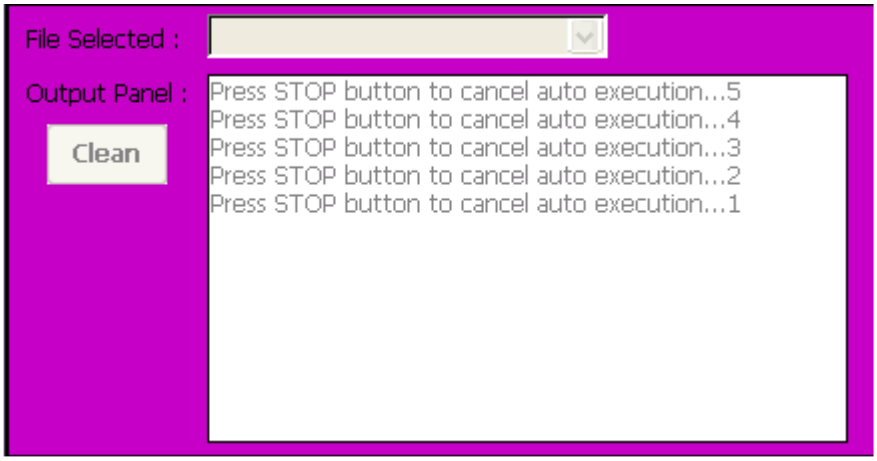
除了 Script Runtime 內建的備援規則外，也可由使用者以手動方式選擇 major 和 backup QP-500。點選介面上的”Select QP-500 manually”按鈕後，介面將呈現如下圖：



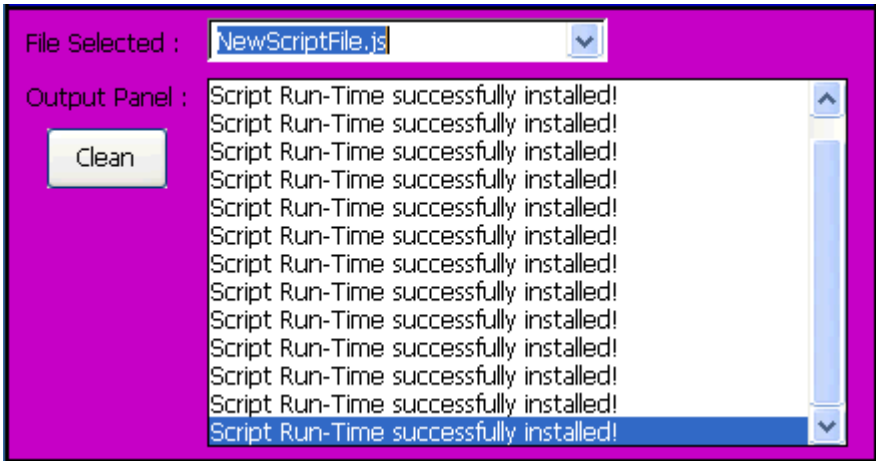
按下所要選擇的 QP-500 前方的 radio button，再按下 major 或是 backup 按鈕，便可將所選的 QP-500 變更為 major 或是 backup 。

D.4 除錯資訊介面（Debug Interface）

此區塊可提供使用者檢視 debug message 的功能。Script Runtime 剛開始啟動時，將自動搜尋是否設有自動執行的專案，若設有自動執行的專案，將在五秒後自動執行該專案。使用者可在這五秒內按下 Stop button 來停止 Script Runtime 自動執行專案。此倒數五秒鐘的訊息將呈現於 debug interface 中，如下圖所示。



當使用者在 Script Editor 中編輯 Script 時，可在各個 Script 中使用 print 函式，將某些變數值印出以幫助檢查 Script 的執行是否正確。在 File selected 的下拉式選單中可以選擇正在執行中的專案內所有 Script 檔案。若所選擇的 Script 中內有 print 函式，將在下方的 Output panel 列印出執行結果，如下圖所示：



此外，左方的 Clean 按鈕可用以清除 Output panel 的內容。

D.5 程式控制介面（Control interface）

此區塊具有兩個按鈕：

- Min. Window：將 Script Runtime 操作介面最小化至系統列(system tray)當中。
- Exit：關閉 Script Runtime。