SmartQ 系統開發範例說明

[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、散佈等方式取得部份或全部軟體或相關的複製品。

目錄：

1. 前置準備

本文件的主要目的在於以實際範例介紹 SmartQ 開發工具(如: Script Editor 及 Flash HMI Editor)的整合使用，以各種模擬個案的開發步驟及流程，做實際的示範操作，有助於使用者快速熟悉此套工具。本文件所演示的各個模擬個案，著重於整個 SmartQ 系統的開發程序及流程，對於單一工具的使用細節並未詳述。使用者可在各工具的使用手冊中取得詳細的相關資訊。此外，本文中有關硬體 I/O 模組的設定，也可參照相關的硬體手冊，將有助於瞭解本說明中的個案應用。SmartQ 相關使用手冊如下：

- SmartQ 系統概述
- SmartQ Script Editor 入門手冊
- SmartQ Flash HMI Tools 入門手冊

1.1 軟體開發工具需求

SmartQ 開發工具包括 SmartQ Script Editor 及 SmartQ HMI Editor。其中 SmartQ Script Editor 為用於各控制器工作邏輯的開發，SmartQ HMI Editor 為控制系統人機介面的設計。SmartQ 所提供的開發工具操作平台為 Windows OS 環境，故建議使用者操作時，電腦作業系統需為 Windows XP 以上的等級。由於 SmartQ 的開發工具各有不同的軟體環境需求，請使用者在安裝之前，參閱各開發工具入門手冊的描述，建置所需的軟體環境。

1.2 硬體平台需求

SmartQ 在系統開發上，各工具的角色主要為 Script Editor 負責撰寫控制器工作邏輯的 Script 檔案；Script Runtime 負責執行 Script 檔案內容；Flash HMI Editor 負責人機介面的設計；Flash HMI Player 負責人機介面的執行。故使用者的基本硬體需求為一台個人電腦並安裝 Script Editor 及 Flash HMI Editor；而在 Script 的執行部份，則需要已安裝 Script Runtime 的 QPAC-8x2 控制器支援。由於 SmartQ 開發工具各有不同的硬體環境需求，請使用者在安裝之前，參閱各工具的使用手冊，建置所需的硬體環境。而控制器上所需要的 I/O 模組，可以以各案例中所條列的硬體設備為參考依據。

2. 開發範例說明

本章節將以一套簡單的 I/O 系統示範如何藉由 SmartQ Script Editor 與 SmartQ Flash HMI Editor，開發出一系列由簡入繁的案例。藉由範例的介紹讓使用者快速熟悉 SmartQ 開發工具。

2.1 範例 1：多台控制器的獨立控制

2.1.1 範例描述

此專案範例位於光碟目錄：Example\Demo1，請將此專案完整複製於 C:\ICPDAS\SmartQ\Example\Demo1 目錄下，再開啟 Script Editor **先將 Controller2/calculate.js 的第 18、19 行的單行註解符號（\\）刪除**。

本案例使用兩個控制器，每一個控制器上各有一組 DI/DO 與 AI/AO 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。

範例所使用的設備如下：

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		安裝 SmartQ Script Editor 工具軟體
電源供應器	DP-665	+10V ~ +30VDC

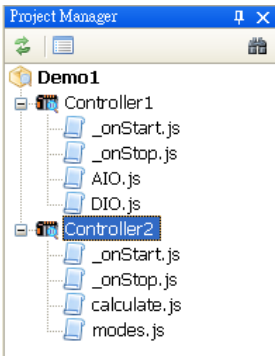
本專案範例中兩個控制器彼此的運作是各自獨立，專案內每一個控制器所執行的工作內容與目的如下表：

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

2.1.2 範例實作

2.1.2.1步驟 1 – 建立新專案



由於此案例中使用兩個控制器，故必須在專案管理員視窗中於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 Script Editor 入門手冊步驟 4.3 與步驟 4.4）

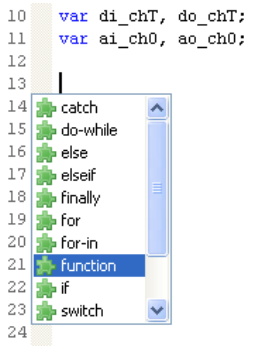
2.1.2.2 步驟 2 – 撰寫 Script 程式碼

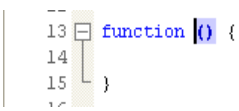
藉由 Auto-Completion 與關鍵字的快捷建立功能，可以幫助使用者在開發 Script 程式邏輯時，減少重複輸入相同的字元，且避免造成與文字輸入相關的錯誤。下面將針對檔案程式一一介紹，使用者可嘗試輸入程式內容。

- Controller1 程式簡介

如範例所示，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 中將對 do_chT 這一個變數持續做加 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。





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  }
```

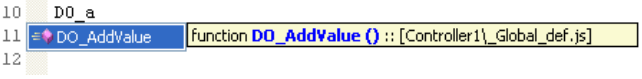
完成新函式之後請務必**儲存檔案**。儲存檔案後，DO_AddValue 函式名稱才可於編輯 Script 檔時加入 Auto-Completion 的輔助列表中。因此，編輯檔案時，若希望重複使用先前所編輯的函式，可先儲存檔案，即可由 Auto-Completion 功能提供的輔助列表輸入此函式。AO_AddValue 函式的宣告程序與上述函式相似。與 DO_AddValue 的差別只在每次增加 0.5，以 10 為最高上限。

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

Example 1-1. Controller1/_onStart.js

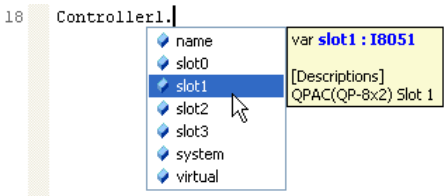
```
1  /**
2  * FileName: _onStart.js
3  * Author:
4  * Version:
5  * Description:
6  *
7  * Date: 2009/3/11
8  */
9
10
11 var di_chT, do_chT;
12 var ai_ch0, ao_ch0;
13
14 function DO_AddValue()
15 {
16     do_chT++;
17
18     if (do_chT > 65535)
19         do_chT = 0;
20     return;
21 }
22
23 function 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，由於事先已完成 Controller1/_onStart.js 編輯並儲存檔案。當輸入這兩個字元時將觸發 Auto-Completion 的功能，顯示輔助列表。輸入 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 Script Editor 入門手冊附錄 A.3）。在 Script Editor 撰寫 Script 時，必須參照此模型，按照階層依序定義即將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。



Controller1/DIO.js 與 Controller1/AIO.js 的程式內容，請參考 Example 1-2 及 1-3。在 Example 1-2 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 Script Editor 入門手冊附錄 D）。 Example 1-3 的概念亦同。

Example 1-2. Controller1/DIO.js

```
1  /**
2  * FileName: DIO.js
3  * Author:
4  * Version:
5  * Description:
6  *
7  * Date: 2008/10/2
8  */
9
10 DO_AddValue();
11
12 Controller1.slot0.DO.chT = do_chT;
13 print("DIO >>>> Controller1.slot0.DO.chT = " + do_chT);
14
15 di_chT = Controller1.slot1.DI.chT;
16 print("DIO ---- Controller1.slot1.DI.chT = " + di_chT);
17
```

Example 1-3. 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，由於其他 Script 檔案並未使用任何函數，因此在_onStart.js 的程式不需要宣告函式只需以 var 宣告兩個變數，儲存 2 的次方值。Controller2/_onStart.js 的程式如 Example 1-4。

Example 1-4. Controller2/_onStart.js

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

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

Example 1-5. Controller2/calculate.js

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

if( count != 0 )
   count = count % Math.pow(2, 16);

/*
 * when the ans is the 16th powers of the number two,
 * it must be set to one, but after mod operation it is zero.
 */

if (count == 0)
    count = 1;

Controller2.slot0.DO.chT = count;
count = count * 2;
```

Controller2/calculate.js 此檔案將進行 2 的次方計算。**注意，若第 18、19 行有單行註解符號（\\），請先將其刪除。**在 Example 1-5 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。

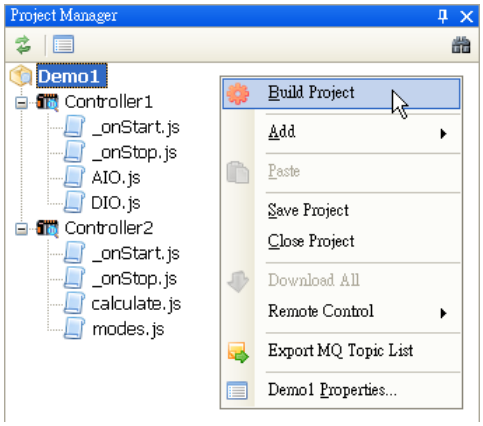
Controller2/modes.js 依據 DI 模組收到的實體訊號，判斷哪個 digital channel 的值為真，用以決定列印的模式代號與 AO 模組在 channel0 上輸出的伏特值。由以上這兩個 Script 檔案將可瞭解如何操作模組上 channel 實體訊號的輸入與輸出。

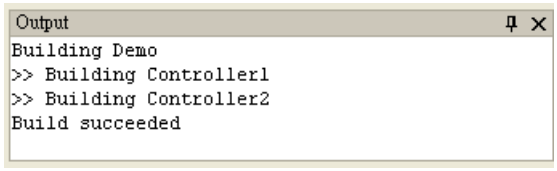
Example 1-6. Controller2/modes.js

```
/**
 * 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" );                }
```

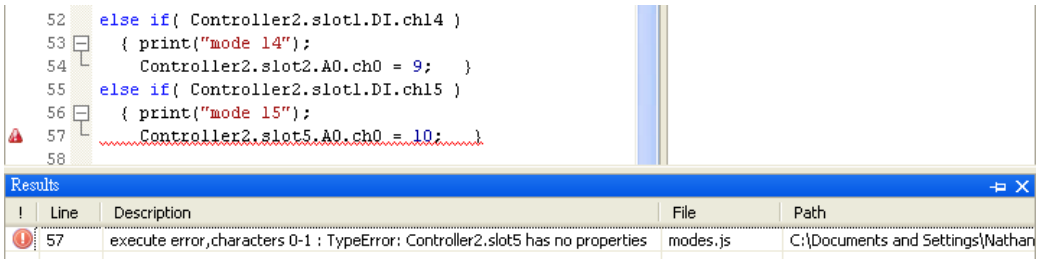
2.1.2.3 步驟 3 – 編譯專案

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 介面做通訊。因此無須確認相關的通訊設定，可直接進行編譯(請參考 Script Editor 入門手冊步驟 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)。

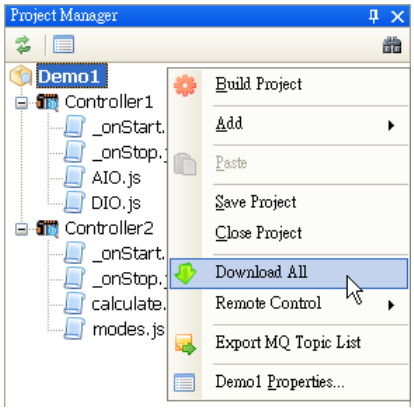




若出現輸入錯誤，如：將 Controller2/modes.js 中第 57 行誤植為”Controller2.slot5.AO.ch0 = 10”，進行編譯後將立即於結果視窗顯示錯誤訊息，第 57 行發生錯誤處將以紅色波浪線標示。使用者可依照提示修改正確的值。



2.1.2.4 步驟 4 – 下載專案



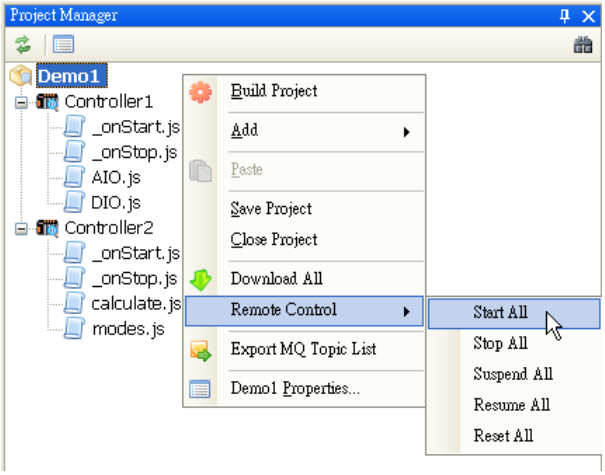
再將 Script 檔案下載至遠端的 QPAC 上，由於此次為初次下載(請參考 Script Editor 入門手冊步驟 4.8)，必須下載全部。在專案管理員視窗中，選擇專案的節點，點選並按滑鼠右鍵，顯示右鍵選單，從右鍵選單中按下[Download All]。

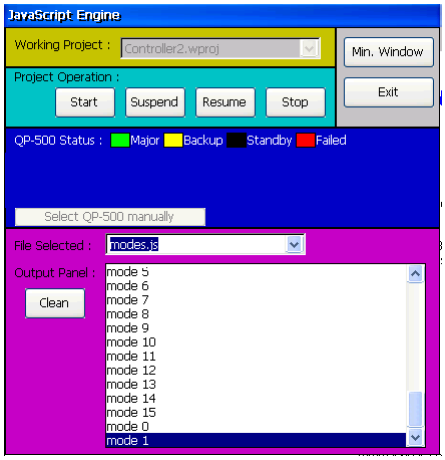
若下載成功，則輸出視窗上將出現下載成功的訊息。若發現下載等候時間過長或是出現無法連線的錯誤訊息時，請先檢查網路線是否鬆脫、QPAC 是否已成功取得連線的 IP、QPAC 子專案的 IP 是否有設錯、QPAC 是否正常運作、QPAC 上的 Script Runtime 是否正常運作。

2.1.2.5 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 runtime 執行下載後的 Script 檔案。詳細操作步驟請參考 Script Editor 入門手冊步驟 4.9。

此時使用者應可發現在 Controller2 上，DI 與 DO 模組的面板燈號亮過一輪後，再繼續重新依序亮起。這與我們預期的功能相符合。而 Controller2 的列印資訊，則可透過控制器上的 Script Runtime 介面檢視列印資訊。例如：modes.js 的資訊列印，將持續列印 mode 模式號碼。





2.2 範例 2：多台控制器間的 Master/Slave 控制

2.2.1 範例描述

此專案範例位於光碟目錄：Example\Demo2，請將此專案完整複製於 C:\ICPDAS\SmartQ\Example\Demo2 目錄下，再開啟 Script Editor。

本案例使用兩個控制器，每一個控制器上各有一組 DI/DO 與 AI/AO 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表:

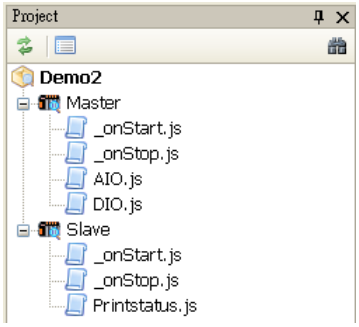
設備	型號	說　明
SmartQ Broker	QP-500	SmartQ Broker 執行設備
PAC	QP-842	QPAC 系列可程式自動控制器, 4 插槽
DO 模組	I-8056W	DO x 16 通道, 安裝在 QP-842 的插槽 0
DI 模組	I-8051W	DI x 16 通道, 安裝在 QP-842 的插槽 1
AO 模組	I-8024W	AO x 4 通道, 安裝在 QP-842 的插槽 2
AI 模組	I-8017HW	AI x 8 通道, 安裝在 QP-842 的插槽 3
PC 或 NB		用於安裝 SmartQ Script Editor 工具軟體
電源供應器	DP-665	+10V ~ +30VDC

本案例中兩個控制器彼此的運作並非各自獨立，而是以 Master 和 Slave 的模式運作。Master 控制器除了控制本身的 I/O 模組外，也對 Slave 控制器的 I/O 模組下達命令，要求 Slave 隨 Master 執行相同的動作。在 Slave 這一方則是純粹的接受來自 Master 的命令並列印出本身的 I/O 模組狀態。每一個控制器專案所執行的工作內容與目的如下表：

QPAC 名稱	Master
工作內容	DO 或 DI 其輸出或接收的資料為累加的狀況，從 0 加到 65535，讓模組上的燈號能以一個 2 進位模式來亮燈。 AO 或 AI 其輸出或接收的資料亦為累加的狀況，由 0v 加到 10v，每次遞增 0.5v。 **遞送相同命令給 Slave**（與前一個案例差異處）
程式目的	瞭解如何對遠端的控制器下達命令
QPAC 名稱	Slave
工作內容	只列印出控制器上的模組狀態
程式目的	觀察控制器本身模組狀態

2.2.2 範例實作

2.2.2.1 步驟 1 – 建立新專案



在此案例中必需有兩個的控制器，故必須在專案管理員視窗中於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，Master 將增加兩個 Script 檔案，Slave 將增加一個 Script 檔案。所有的 Script 檔案屬性皆設定為 Timer：500ms。為了讓兩個控制器也能溝通，必須在專案根節點 Demo2 的屬性頁設定 Qp-500 IP 位址。

在此需注意：**若控制器需交換資訊，有幾台控制器要加入通訊，便需在專案管理員視窗上宣告同樣數量的 QPAC 子專案**。如此 Script Editor 才能得知有其他的控制器加入通訊，必須定義控制器具有的變數與函式，並加入 Auto-Completion 提供的輔助選單，供使用者選取。以此個案為例，若 Slave 子專案不需執行任何 Script 檔，仍必須新增一個 Slave QPAC 子專案，在編輯 Master 子專案下的 Script 檔時，Auto-Completion 功能才可將 Slave 加入輔助列表中供使用者選擇。

2.2.2.2 步驟 2 – 撰寫 Script 程式碼

- Master 程式簡介

如範例所見，我們將在 Master/_onStart.js 檔案內建立兩個新的函式：DO_AddValue 與 AO_AddValue。在 DO_AddValue 中將對變數 do_chT 持續做加 1 的動作。當超過 65535 時，就將其值歸零重新開始計算。AO_AddValue 此函式的作法亦類似。與 DO_AddValue 的差別只在每次增加 0.5，以 10 為最高上限。這些動作與上一個案例相同。

Master/_onStart.js 的完整程式內容，請參考 Example 2-1。

Example 2-1. Master/_onStart.js

```
1  /**
2  * FileName: _onStart.js
3  * Author:
4  * Version:
5  * Description:
6  *
7  * Date: 2009/3/20
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;
```

Example 2-2. Master/DIO.js

```
1  /**
2  * FileName: DIO.js
3  * Author:
4  * Version:
5  * Description:
6  *
7  * Date: 2008/10/2
8  */
9
10 DO_AddValue();
11
12 Master.slot0.DO.chT = do_chT;
13 Slave.slot0.DO.chT = do_chT;
14 print("DIO >>>> Master.slot0.DO.chT = " + do_chT);
15
16 di_chT = Master.slot1.DI.chT;
17 print("DIO ---- Master.slot1.DI.chT = " + di_chT);
```

Master/DIO.js 與 Master/AIO.js 的完整程式內容，請參考 Example 2-2 及 2-3。在 Example 2-2 程式中，DO_AddValue()函式以 do_chT 變數儲存要 DO 模組輸出的資料，並在第 12 行將 do_chT 變數值指定給 DO

模組，使其輸出訊號。再透過第 16 行的變數 di_chT 取得 DI 模組上得到的訊號值。並分別將此兩變數列印出。Example 2-3 的概念亦同。

若與上一個案例的 DIO.js 與 AIO.js 比較，主要差別位於兩個檔案的第 13 行，此行程式碼為分別對遠端的 slave 下指令，使遠端的 Slave 做相同的動作。

Example 2-3. Master /AIO.js

```
1  /**
2   * FileName: AIO.js
3   * Author:
4   * Version:
5   * Description:
6   *
7   * Date: 2008/10/2
8   */
9
10  AO_AddValue();
11
12  Master.slot2.AO.ch0 = ao_ch0;
13  Slave.slot2.AO.ch0 = ao_ch0;
14  print("AIO >>>> Master.slot2.AO.ch0 = " + ao_ch0);
15
16  ai_ch0 = Master.slot3.AI.ch0;
17  print("AIO ---- Master.slot3.AI.ch0 = " + ai_ch0);
```

Slave 程式簡介

由於 Slave 僅接收命令做動作，並不需在本地端主動執行命令操作 I/O 模組。故只需在 Printstatus.js 中執行列印模組資訊的程式，確認是否已執行 Master 的命令，隨 Master 控制器更改模組狀態。

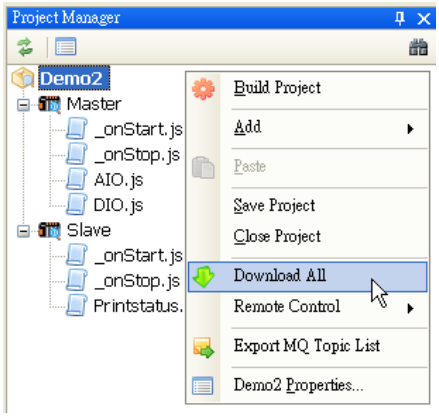
Example 2-4. Slave/Printstatus.js

```
1  /**
2   * FileName: Printstatus.js
3   * Author:
4   * Version:
5   * Description:
6   *
7   * Date: 2009/3/23
8   */
9
10  var DIOresult;
11  var AIOresult;
12  DIOresult = Slave.slot1.DI.chT;
13  print("DIO ---- Slave.slot1.DI.chT = " + DIOresult);
14
15  AIOresult = Slave.slot3.AI.ch0;
16  print("AIO ---- Slave.slot3.AI.ch0 = " + AIOresult);
```

2.2.2.3步驟 3 – 編譯專案

此步驟與第一個個案操作步驟相同，請使用者參考本範例說明章節步驟 2.1.2.4。請務必注意 QP-500 的 IP 位址是否已設定。若未設定，控制器將無法互相溝通資訊。

2.2.2.4步驟 4 – 下載專案



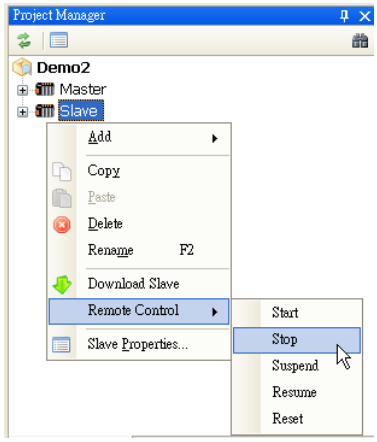
再將 Script 檔案下載至遠端的 QPAC 上，在此個案情境下，QPAC 將互相通訊，必須全部下載。因此在專案管理員視窗中，選擇專案的節點，點選並按滑鼠右鍵，顯示右鍵選單，由右鍵選單中點選[Download All]。

若下載成功，將在輸出視窗上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。

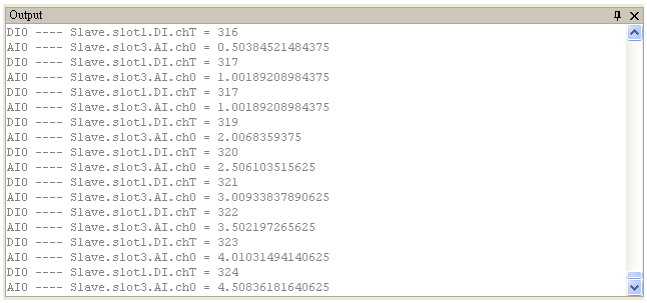
為了確保 QPAC 彼此間通訊無礙，每當使用者修改檔案並完成編譯後，務必以[Download all]的模式下載檔案，更新所有 QPAC 上的組態。

2.2.2.5 步驟 5 – 執行專案

此步驟與第一個個案操作步驟相同，請使用者參考本範例說明章節2.1.2.5。此時使用者應該可以發現在 Slave 控制器上，DI 與 DO 模組的面板燈號跟著 Master 控制器上的 DI/DO 模組一樣的操作方式，Slave 本身並未進行亮燈的操作，僅透過 Master 控制器上的 Script 命令，遠端遙控 Slave 的運作。



若兩台控制器同時執行 Script 程式，Slave 上的變數值變化可由遠端 Slave 控制器上的 Script Runtime 使用者介面觀察。也可以點選 Script Editor 上的 Slave 專案節點並按滑鼠右鍵，顯示右鍵選單，按下 **Remote Control > Stop**，暫時停止 Slave 的運作。在 Script Editor 功能表(Menu Bar)上，點選 **Debug > Quick Start with Print > Slave** 即可將 Slave 控制器上的資訊傳到 Script



Editor，此時 Slave/Printstatus.js 執行內容將列印於輸出視窗上，透過此方式確認輸入 DI 模組的值。

2.3 範例 3：多台控制器間的分散式控制

2.3.1 範例描述

此專案範例位於光碟目錄：Example\Demo3，請將此專案完整複製於 C:\ICPDAS\SmartQ\Example\Demo3 目錄下，再開啟 Script Editor。

本案例使用兩個控制器，每一個控制器上各有一組 DI/DO 與 AI/AO 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:

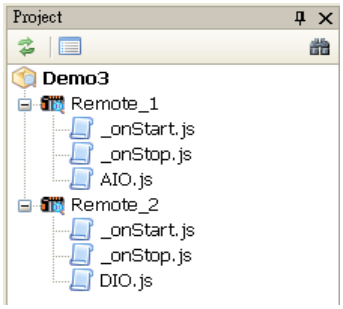
設備	型號	說　明
SmartQ Broker	QP-500	SmartQ Broker 執行設備
PAC	QP-842	QPAC 系列可程式自動控制器, 4 插槽
DO 模組	I-8056W	DO x 16 通道, 安裝在 QP-842 的插槽 0
DI 模組	I-8051W	DI x 16 通道, 安裝在 QP-842 的插槽 1
AO 模組	I-8024W	AO x 4 通道, 安裝在 QP-842 的插槽 2
AI 模組	I-8017HW	AI x 8 通道, 安裝在 QP-842 的插槽 3
PC 或 NB		用於安裝 SmartQ Script Editor 工具軟體
電源供應器	DP-665	+10V ~ +30VDC

本案例中兩個控制器彼此的運作並非各自獨立，而是以即時分散式的模式運作。Remote_1 控制器除了設定本身的 AI/AO 外，也對 Remote_2 控制器的 AI/AO 下達相同命令。Remote_2 控制器除了設定本身的 DI/DO 外，也對 Remote_1 控制器的 DI/DO 下達相同命令。每一個控制器專案所執行的工作內容與目的如下表：

QPAC 名稱	Remote_1
工作內容	AO 或 AI 其輸出或接收的資料為累加的值，由 0v 加到 10v，每次遞增 0.5v。 **遞送相同命令給 Remote_2 的 AO**（與前一個案例差異處）
程式目的	瞭解如何對遠端的控制器下達命令、 如何呼叫需要參數的函式
QPAC 名稱	Remote_2
工作內容	DO 或 DI 其輸出或接收的資料為累加的狀況，從 0 加到 65535，讓模組上的燈號能以一個 2 進位模式來亮燈。 **遞送相同命令給 Remote_1 的 DO**（與前一個案例差異處）
程式目的	瞭解如何對遠端的控制器下達命令

2.3.2 範例實作

2.3.2.1 步驟 1 – 建立新專案



由於此案例中有兩個的控制器，故必須在專案管理員視窗中於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，將再各自增加一個 Script 檔案。而所有的 Script 檔案屬性皆設定為 Timer：500ms。為了讓兩個控制器也能溝通，必須在根節點 Demo3 的屬性頁設定 QP-500 的 IP 位址。

在此需注意：**若控制器需交換資訊，有幾台控制器要加入通訊，便需在專案管理員視窗上宣告同樣數量的 QPAC 子專案**。如此 Script Editor 才能得知有其他的控制器加入通訊，必須定義控制器具有的變數與函式，並加入 Auto-Completion 提供的輔助選單，供使用者選取。

2.3.2.2 步驟 2 – 撰寫 Script 程式碼

- Remote_1 程式簡介

如範例所見，我們在 Remote_1/_onStart.js 檔案內建立兩個新的函式，名稱各為 AO_AddValue()與 round()。在 AO_AddValue()函式中將對 ao_ch0 這一個變數持續做加 0.5 的動作。當超過 10 時，便將其值歸零重新開始計算。為了使輸出美觀，round()函式將類比資料輸出時，以四捨五入取到小數點第二位。

藉由 round()函式可以學習如何在函式中傳遞參數與回傳值。round()函式中宣告了兩個參數 value 和 decimalplace，value 用以接收要做運算的小數，decimalplace 則用來界定透過函數處理後要輸出的小數點後數字位數，並透過 return 關鍵字將回傳值傳回。

Master/_onStart.js 的完整程式內容，請參考 Example 3-1。

Example 3-1. Remote_1/_onStart.js

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

Remote_1/AIO.js 的程式詳細內容，請參考 Example 3-2。在程式 Example 3-2 中，AO_AddValue()函數以 ao_ch0 變數儲存要 AO 模組輸出的資料，第 12 行將 ao_ch0 變數值指定給 AO 模組，使其輸出訊號。再透過 ai_ch0 取得 DI 模組上得到的訊號值。並分別將此兩變數列印出。

若與上一個案例的 AIO.js 比較，大致上的行為都類似。但此兩個個案主要差別在於上一個案例 Master 控制器以 DIO.js 負責處理本身 DIO 模組。而本案例 Remot_1 控制器的 DIO 模組，則透過接收 Remote_2 的命令運作，並對 Remot_2 下命令。

故 Remote_1/AIO.js 的第 13 行及第 19 至 20 行的程式碼有所差異。第 13 行程式碼對遠端的 Remote_2 下指令，使遠端 Remote_2 的 AI/AO 模組執行相同的動作。第 19 至 20 行則是用以確認 Remote_1 是否成功接收由遠端 Remote_2 所傳遞的命令，並更新本身 DI/DO 模組的狀態。

Example 3-2. Remote_1/AIO.js

```
1  /**
2  * FileName: AI0.js
3  * Author:
4  * Version:
5  * Description:
6  *
7  * Date: 2008/10/2
8  */
9
10 AO_AddValue();
11
12 Remote_1.slot2.A0.ch0 = ao_ch0;
13 Remote_2.slot2.A0.ch0 = ao_ch0;
14 print("remote_1>>>> Remote_1.slot2.A0.ch0 = " + round(ao_ch0,2));
15
16 ai_ch0 = Remote_1.slot3.AI.ch0;
17 print("remote_1>>>> Remote_1.slot3.AI.ch0 = " + round(ai_ch0, 2));
18
19 do_chT = Remote_1.slot0.D0.chT;
20 di_chT = Remote_1.slot1.DI.chT;
21 print("remote_1>>>> Remote_1.slot0.D0.ch0 = " + do_chT);
22 print("remote_1>>>> Remote_1.slot1.DI.ch0 = " + di_chT);
```

Remote_2 程式簡介

Remote_2 只需執行 DIO 模組的操作，因此在 Remote_2/_onStart.js 檔案內建立 2 個新的函式，名稱為 D0_AddValue()及 round()。

在 DO_AddValue()中將對 do_chT 這一個變數持續做加 1 的動作。當超過 65535 時，就將其值歸零重新開始計算。而 round()函式則與 Example 3-1 目的相同。Remote_2/_onStart.js 完整的程式內容，請參考 Example 3-3。

在 Remote_2/DIO.js 中，我們將呼叫 DO_AddValue()函式對儲存 DO 值的變數做加一，並存取模組資料。Remote_2/DIO.js 的完整程式內容，請參考 Example 3-4。在 Remote_2/DIO.js 程式中，Do_AddValue()函式以 do_chT 變數儲存要 DO 模組輸出的資料，第 12 行程式碼將 do_chT 便數值指定給 DO 模組，使其輸出訊號。再透過 di_chT 取得 DI 模組上得到的訊號值。並分別將此兩項變數列印出來。

若與上一個案例的 DIO.js 比較，其主要差別在於檔案的第 13 行及第 19 至 20 行的程式碼有所差異。第 13 行程式碼對遠端的 Remote_1 下指令，使遠端 Remote_1 的 DIO 模組執行相同的動作。第 19 至 20 行則是用以確認 Remote_2 是否成功接收由遠端的 Remote_1 所傳遞的命令，並更新本身 AI/AO 模組的狀態。

Example 3-3. Remote_2/_onStart.js

```
/**
* FileName: _onStart.js
* Author:
* Version:
* Description:
*
* Date: 2009/3/25
*/


var di_chT, do_chT;
var ai_ch0, ao_ch0;

function D0_AddValue()
{
    do_chT++;

    if (do_chT > 65535)
        do_chT = 0;
    return;
}

function round(value,decimalplace)
{
    return Math.round(value * Math.pow(10,decimalplace)) / Math.pow(10,decimalplace);
}

do_chT = 0;
di_chT = 0;
ai_ch0 = 0;
ao_ch0 = 0;
```

Example 3-4. The Remote_2/DIO.js example

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

D0_AddValue();

Remote_2.slot0.D0.chT = do_chT;
Remote_1.slot0.D0.chT = do_chT;
print("Remote_2---- Remote_2.slot0.D0.chT = " + do_chT);

di_chT = Remote_2.slot1.DI.chT;
print("Remote_2---- Remote_2.slot1.DI.chT = " + di_chT);

ao_ch0 = Remote_2.slot2.A0.ch0;
ai_ch0 = Remote_2.slot3.AI.ch0;
print("remote_2---- Remote_2.slot2.A0.ch0 = " + round(ao_ch0, 2));
print("remote_2---- Remote_2.slot3.AI.ch0 = " + round(ai_ch0, 2));
```

2.3.2.3 步驟 3 – 編譯專案

此步驟與第一個個案操作步驟相同，請參考本範例說明章節2.1.2.3。請注意 QP-500 的 IP 位址是否已完成設定。若未完成設定，控制器將無法互相通訊。

2.3.2.4 步驟 4 – 下載專案

完成編譯專案後，便可將 Script 檔案下載至遠端的 QPAC 上。在此個案中， 需要 QPAC 彼此溝通，請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]。

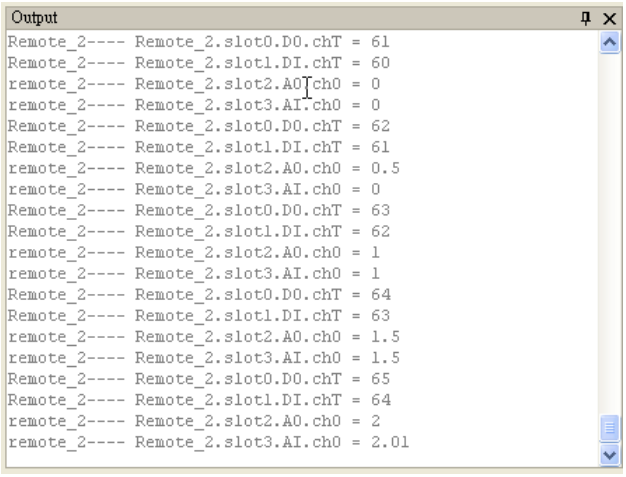
若下載成功，則在輸出視窗上將可看到下載成功的訊息。若發現下載等候時間過長或是有無法連線的錯誤訊息時，請先檢查網路線是否鬆脫、QPAC 是否已成功取得連線的 IP、QPAC 子專案的 IP 是否有設錯、QPAC 是否正常運作、QPAC 上的 Script Runtime 是否正常運作。

為了確保 QPAC 彼此間通訊無礙，每當使用者修改完檔案並完成編譯後，務必以[Download all]的模式下更新 QPAC 上的組態。

2.3.2.5 步驟 5 – 執行專案

此步驟與第一個個案操作步驟相同，請使用者參考本範例說明章節2.1.2.5。此時使用者應可發現在 Remote_1 控制器上，DI 與 DO 模組的面板燈號隨著 Remote_2 控制器上的 DI/O 模組的操作而變化，Remote_1 本身並未執行亮燈的操作，僅透過 Remote_2 上的 Script 命令，遠端遙控 Remote_1 的運作。

若兩台控制器正同時執行時 Script 程式，可進行以下步驟觀察 Remote_2 上的 AI 值：點選 Remote_2 專案節點並按滑鼠右鍵，顯示右鍵選單，按下 **Remote Control > Stop**，暫時停止 Remote_2 的運作。在 Script Editor 功能表(Menu Bar)上，點選 **Debug > Quick Start with Print > Remote_2**。則 Remote_2/DIO.js 執行內容將列印於輸出視窗上，透過此方式確認 Remote_2 是否成功接收 AI/AO 模組的輸入值。



2.4 範例 4: HMI 與控制器間的互動

2.4.1 範例描述

此專案範例位於光碟目錄：Example\Demo4，請將此專案完整複製於 C:\ICPDAS\SmartQ\Example\Demo4 目錄下，再開啟 Script Editor。Flash HMI 專案目錄則為 Demo4\HMIDemo。

本案例使用兩個控制器，每一個控制器上各有一組 DI/DO 與 AI/AO 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
SmartQ Broker	QP-500	SmartQ Broker 執行設備
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 及 SmartQ Flash HMI Editor 工具軟體
電源供應器	DP-665	+10V ~ +30VDC

除了硬體控制器外，本案例另外採用 SmartQ HMI Editor 使用手冊第六章的範例做為本案例的人機操作介面，模擬 HMI 如何與兩台控制器的居家樓房工控系統進行互動，利用 HMI 操作介面執行遠端控制並且取得遠端控制資訊顯示於操作介面上，關於此 HMI 介面的設計流程，請參考 SmartQ HMI Editor 使用手冊第六章。

本案例中兩個控制器彼此的運作是各自獨立，每一個控制器專案所執行的工作內容與目的如下表：

QPAC 名稱	Control_1
工作內容	控制房屋左半區域的大門、走廊、大廳、客廳的燈控與風扇控制
程式目的	實作大門按鈕與客廳按鈕的互斥與風扇運轉限制條件
QPAC 名稱	Control_2
工作內容	控制房屋右半區域的廚房、書房、主臥的燈控與風扇控制
程式目的	實作瓦斯使用的控制與用量計算

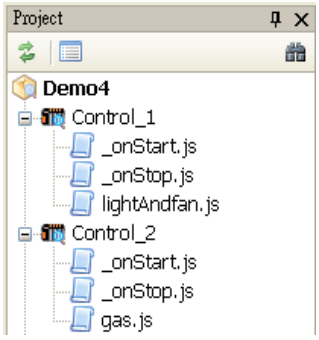
本案例透過人機操作介面與控制器希望呈現的個案情境描述如下：當住戶開啟大門進入屋內，走廊燈與大廳燈將自動開啟。而為了節省能源，大廳的風扇僅在室內溫度達到二個大刻度以上時才開始運轉。住戶在客廳時，大廳燈與走廊燈將自動關閉。住戶外出時，若忘記關閉客廳燈，將自動關閉客廳燈。為避免瓦斯中毒，當住戶在廚房烹飪時，將自動啟動廚房風扇由外引入流動空氣。

依照情境，此控制系統的功能規劃如下：

1. 大門開關與大廳燈及走廊燈連動。
2. 客廳開關與客廳燈連動。
3. 大門開關與客廳開關為互斥，同時只有一個是呈現 on 的狀態。
4. 客廳風扇將依照風扇開關與室內溫度的狀態做為啟動的依據。當風扇開關呈現 ON 狀態且室內溫度大於 2 個大刻度時，風扇才能啟動。
5. 書臥室開關與書房燈及主臥燈連動。
6. 當瓦斯閥打開時，廚房風扇將開啟轉動，運轉後瓦斯爐火才可使用,而瓦斯的用量將累計顯示在瓦斯用量表上，若關閉瓦斯閥關閉，廚房風扇便停止轉動。
7. 繪製趨勢圖顯示瓦斯閥開關與瓦斯用量的歷史紀錄。

2.4.2 範例實作

2.4.2.1 步驟 1 – 建立新專案



由於此案例中有兩個控制器，故必須在專案管理員視窗中於專案目錄下針對這兩個控制器各自建立一個 QPAC 子專案。專案結構如右圖。每一個控制器下除了預設的_onStart.js 和_onStop.js 外，將再各自增加一個 Script 檔案，而每個 Script 檔案屬性皆設定為 Timer：500ms。（請參考 Script Editor 入門手冊步驟 4.3 與步驟 4.4）

2.4.2.2 步驟 2 – 撰寫 script 程式碼

- Control_1 程式簡介

如範例所示，在 Control_1/_onStart.js 的第 22、23 行程式碼中將溫度控制一開始便設定為第五個大刻度(第 23 行)，其他則為變數宣告。Control_1/_onStart.js 完整的程式內容，請參考 Example 4-1。

Example 4-1. Control_1/_onStart.js

```
1  /**
2  * FileName: _onStart.js
3  * Author:
4  * Version:
5  * Description:
6  *
7  * Date: 2009/3/30
8  */
9
10 var door_button;
11 var door_lighter;
12 var livingroom_button;
13 var livingroom_lighter;
14 var hallfan_button;
15 var hallfan_animation;
16 var outside_thermometer;
17 var inside_thermometer;
18 var old_door_button;
19 var old_livingroom_button;
20
21 //讓程式一開始就設定室內溫度在第五個大刻度
22 outside_thermometer = 5;
23 Control_1.slot2.AO.ch0 = outside_thermometer;
```

當 Script Runtime 接收到 Stop 指令時，將執行_onStop.js，將所有的模組上的值歸零。

Example 4-2. Controller1/_onStop.js

```
/**
* FileName: _onStop.js
* Author:
* Version:
* Description:
*
* Date: 2009/3/30
*/


Control_1.slot0.DO.chT = 0;
Control_1.slot2.AO.ch0 = 0;
Control_1.slot2.AO.ch1 = 0;
```

Example 4-3. Control_1/lightAndfan.js

```
/**
* FileName: lightAndfan.js
* Author:
* Version:
* Description:
*
* Date: 2009/3/31
*/
old_door_button = door_button;
door_button = Control_1.slot0.DO.ch0;
old_livingroom_button = livingroom_button;
livingroom_button = Control_1.slot0.DO.ch1;
hallfan_button = Control_1.slot0.DO.ch2;

door_lighter = Control_1.slot1.DI.ch0;
livingroom_lighter = Control_1.slot1.DI.ch1;
hallfan_animation = Control_1.slot1.DI.ch2;
inside_thermometer = Control_1.slot3.AI.ch0;
//客廳開關與大門開關互斥設定
if ( door_button == 1 && livingroom_lighter ==1 )
   {
    //進客廳
    if ( old_door_button == 1 && old_livingroom_button == 0)
    {
        door_button = 0;
        Control_1.slot0.DO.ch0 = door_button;
    }
    //出門時
    if ( old_door_button == 0 && old_livingroom_button == 1)
    {
        livingroom_button = 0;
        Control_1.slot0.DO.ch1 = livingroom_button;
    }
   }

//設定客廳風扇只有在風扇開關on且室內溫度在兩個大刻度以上時才可以開啓
if ( hallfan_button == 1 )
   {
    if ( inside_thermometer < 2 )
    {
        hallfan_button = 0;
        Control_1.slot0.DO.ch2 = hallfan_button;
    }
   }
```

Control_1/lightAndfan.js 完整的程式內容，請參考 Example 4-3。在 Example 4-3 程式中，第 9 至 18 行為透過變數記錄每一個輸入輸出 channel 的值。此外，為了幫助狀態判別，在第 9 至 12 行分別記錄了大門與客廳開關的新舊兩個狀態值。

第 20 至 34 行則用以判斷：當大門與客廳新狀態（或現下狀態）皆為開啟時，將依照這兩個開關的舊狀態（或上一個狀態）判斷哪一個開關一直維持在開啟狀態；哪一個開關是稍後才轉變為開啟狀態。若開關新舊值的狀態皆為開啟，便將其轉換為關閉狀態，使大門與客廳開關最多只有一個可為開啟的狀態，以實作第一個功能需求。

第 37 至 44 行為執行第二項功能需求，當室內溫度小於兩個大刻度時將停止風扇的運轉。

Control_2 程式簡介

在 Control_2 的_onStart.js 程式中，瓦斯爐火大小歸零以表示瓦斯爐火初始狀態為關閉，而瓦斯用量因尚未使用也呈現為零。Control_2/_onStart.js 完整的程式內容請參考 Example 4-4。在 Example 4-5 的程式中，將在專案停止運行時，將控制器上的輸出與輸入模組狀態歸零，以確認專案停止運作。

Example 4-4. Control_2/_onStart.js

```
1  /**
2  * FileName: _onStart.js
3  * Author:
4  * Version:
5  * Description:
6  *
7  * Date: 2009/4/1
8  */
9
10 var gas_valve;
11 var bedroom_button;
12
13 var kitchenfan_animation;
14 var bedroom_lighter;
15
16 var gasstove_comsumption;
17 var gasgage_amount;
18
19 gasstove_comsumption = 0;
20 Control_2.slot2.AO.ch0 = gasstove_comsumption;
21
22 gasgage_amount = 0;
23 Control_2.slot2.AO.ch1 = gasgage_amount;
```

Example 4-5. Control_2/_onStop.js

```
/**
* FileName: _onStop.js
* Author:
* Version:
* Description:
*
* Date: 2009/4/1
*/


Control_2.slot0.DO.chT = 0;
Control_2.slot2.AO.ch0 = 0;
Control_2.slot2.AO.ch1 = 0;
gasgage_amount = 0;
```

Control_2/gas.js 主要用以執行當瓦斯閥狀態為開啟時所要完成的工作。瓦斯閥開啟時，執行第 15 和 20 行的程式，將每單位時間內的瓦斯用量加總，並顯示於人機控制介面的瓦斯用量表以及趨勢圖上。當瓦斯閥關閉時，則執行第 23 和 24 行的程式，關閉瓦斯爐火。

HMI Editor 所開發的操作介面必須與遠端控制器傳遞訊息，因此必須為 HMI Editor 專案中的元件建立通訊屬性設定，並在 Script Editor 專案上匯入 HMI Editor 專案所產生的通訊設定(以專案路徑匯入)以整合控制器與 HMI 間的通訊。

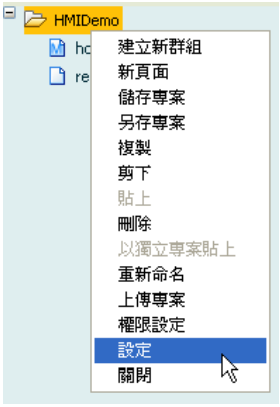
Example 4-6. Control_2/gas.js

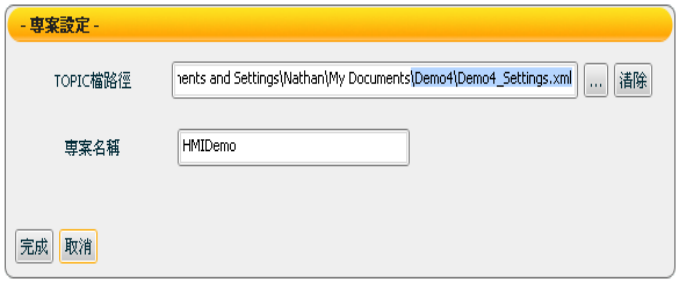
```
/**
* FileName: gas.js
* Author:
* Version:
* Description:
*
* Date: 2009/4/3
*/

gas_valve = Control_2.slot0.DO.ch0;
gasstove_comsumption = Control_2.slot2.AO.ch0;

if ( gas_valve == 1)
{
    gasgage_amount = gasgage_amount + (gasstove_comsumption / 200);
    if ( gasgage_amount > 10 || gasgage_amount == 10 )
       {gasgage_amount = 0; }

    Control_2.slot2.AO.ch1 = gasgage_amount;
}
else
{
    gasstove_comsumption = 0;
    Control_2.slot2.AO.ch0 = gasstove_comsumption;
}
```

HMI Editor 專案中的元件通訊屬性設定

使用者可參考 SmartQ Flash HMI tools 入門手冊 3.3.13 章節，在 HMI 專案上，按下滑鼠右鍵，顯示右鍵選單。點選右鍵選單上的設定項

目，開啟專案設定。在專案設定視窗中按下[...]後，選擇 Script Editor 專案路徑下的硬體對應資訊檔，名稱為 XXX_Settings.xml，此檔將在 SmartQ Script Editor 完成所有的 QPAC 子專案屬性設定後產生。以本個案為例，設定檔的實體檔案路徑為 Demo4\Demo4_Settings.xml 的設定檔。匯入此設定檔後，HMI Editor 將獲得現階段控制器上所有可供使用的 I/O 模組資訊，HMI 上的元件便可進行通訊屬性設定。

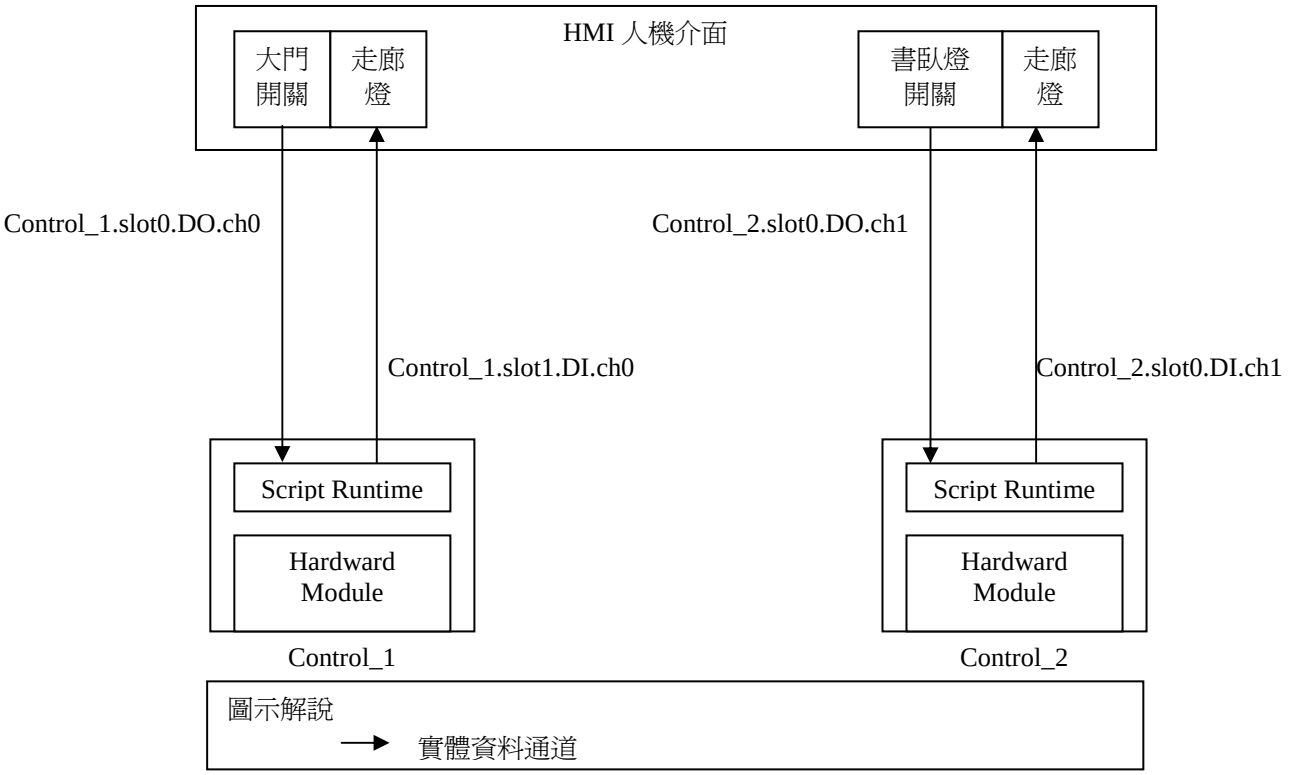




下一步驟將針對個別 HMI 元件設定接收與傳送資料的通道（開啟元件屬性設定中的通訊設定頁），設定的步驟可參考 SmartQ Flash HMI tools 入門手冊 3.5.1 章節中的元件通訊設定。下表列出各元件的通訊設定詳細內容。**所有元件設定完後請務必儲存專案，才能完整更新專案的設定。**

元件名稱	資料通道型態(發送、接收)	資料通道名稱
大門開關	發送、接收	Control_1.slot0.DO.ch0
客廳開關	發送、接收	Control_1.slot0.DO.ch1
風扇開關	發送、接收	Control_1.slot0.DO.ch2
走廊燈	接收	Control_1.slot1.DI.ch0
大廳燈	接收	Control_1.slot1.DI.ch0
客廳燈	接收	Control_1.slot1.DI.ch1
客廳風扇	接收	Control_1.slot1.DI.ch2
室內溫度控制(模擬)	發送、接收	Control_1.slot2.AO.ch0
室內溫度	接收	Control_1.slot3.AI.ch0
瓦斯閥	發送、接收	Control_2.slot0.DO.ch0
書臥燈開關	發送、接收	Control_2.slot0.DO.ch1
廚房風扇	接收	Control_2.slot1.DI.ch0
主臥燈	接收	Control_2.slot1.DI.ch1
書房燈	接收	Control_2.slot1.DI.ch1
瓦斯爐火	發送、接收	Control_2.slot2.AO.ch0

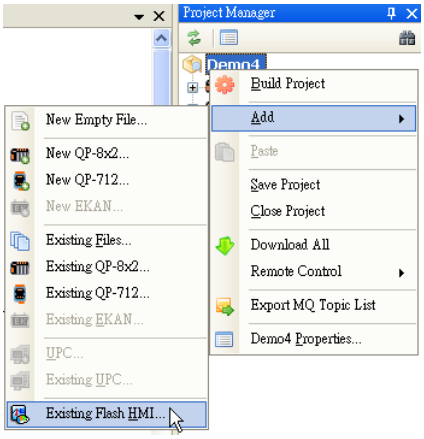
瓦斯用量表	接收	Control_2.slot3.AI.ch1
瓦斯記錄趨勢表－畫筆 1	接收	Control_2.slot3.AI.ch1
瓦斯記錄趨勢表－畫筆 2	接收	Control_2.slot0.DO.ch0

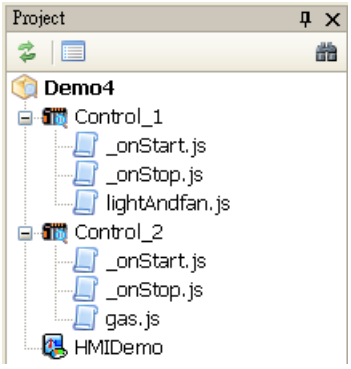


藉由 Script 與 HMI 元件的設定將可達成上圖的溝通環境，使用者可藉此個案了解如何以 SmartQ 開發工具建構通訊環境，以及如何建立硬體與 HMI 人機介面的通訊。

指定與 Flash HMI 通訊對照的專案路徑

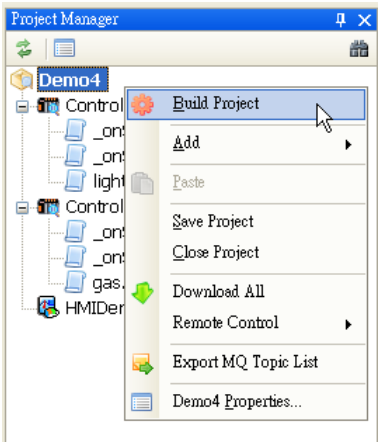
此部份操作可參考 SmartQ Script Editor 入門手冊第 4.6 章節，在 Script Editor 的專案管理員視窗中，選擇專案的節點，點選並按滑鼠右鍵，顯示右鍵選單。從右鍵選單中按下 Add > Existing Flash HMI…，加入 Flash HMI 專案檔案夾路徑，如此 Script 執行時便能與 Flash HMI 通





訊。此時在專案管理員視窗中將可看到新增的 Flash HMI 專案節點。

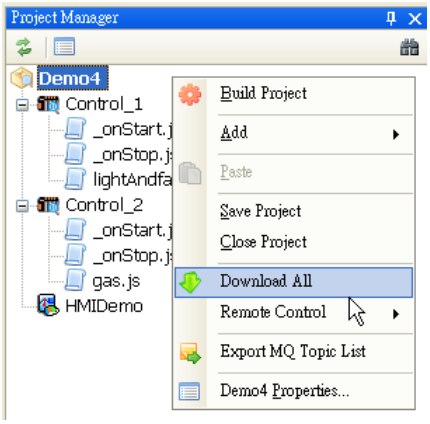
2.4.2.3 步驟 3 – 編譯專案



在此案例中，兩台 QPAC 控制器，除將各自獨立執行 Script 外，也將與 Flash HMI 介面通訊。因此在編譯專案前請先確認專案管理員視窗中已設有 Flash HMI 專案節點，且設定的 Flash HMI 專案路徑正確。Flash HMI Editor 所建立的專案也已存檔。

確認以上的設定無誤後，便可進行編譯程序(請參考 Script Editor 入門手冊步驟 4.8)。直接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），輸出視窗將出現編譯成功的訊息(Build succeeded)。若編譯成功將自動檢查 Flash HMI 專案所設定 QP-500 的 IP 位址與資料通道是否為有效的設定，若正確無誤將顯示 ok 的訊息。

2.4.2.4 步驟 4 – 下載專案

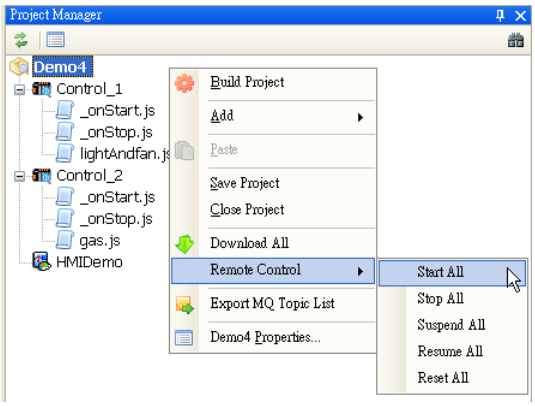


完成編譯專案後，便可將 Script 檔案下載至遠端的 QPAC 上，由於此次為初次下載(請參考 Script Editor 入門手冊步驟 4.9)，必須下載全部。直接在專案管理員視窗中，選擇專案的節點，點選並按滑鼠右鍵，顯示右鍵選單，由右鍵選單中按下[Download All]。

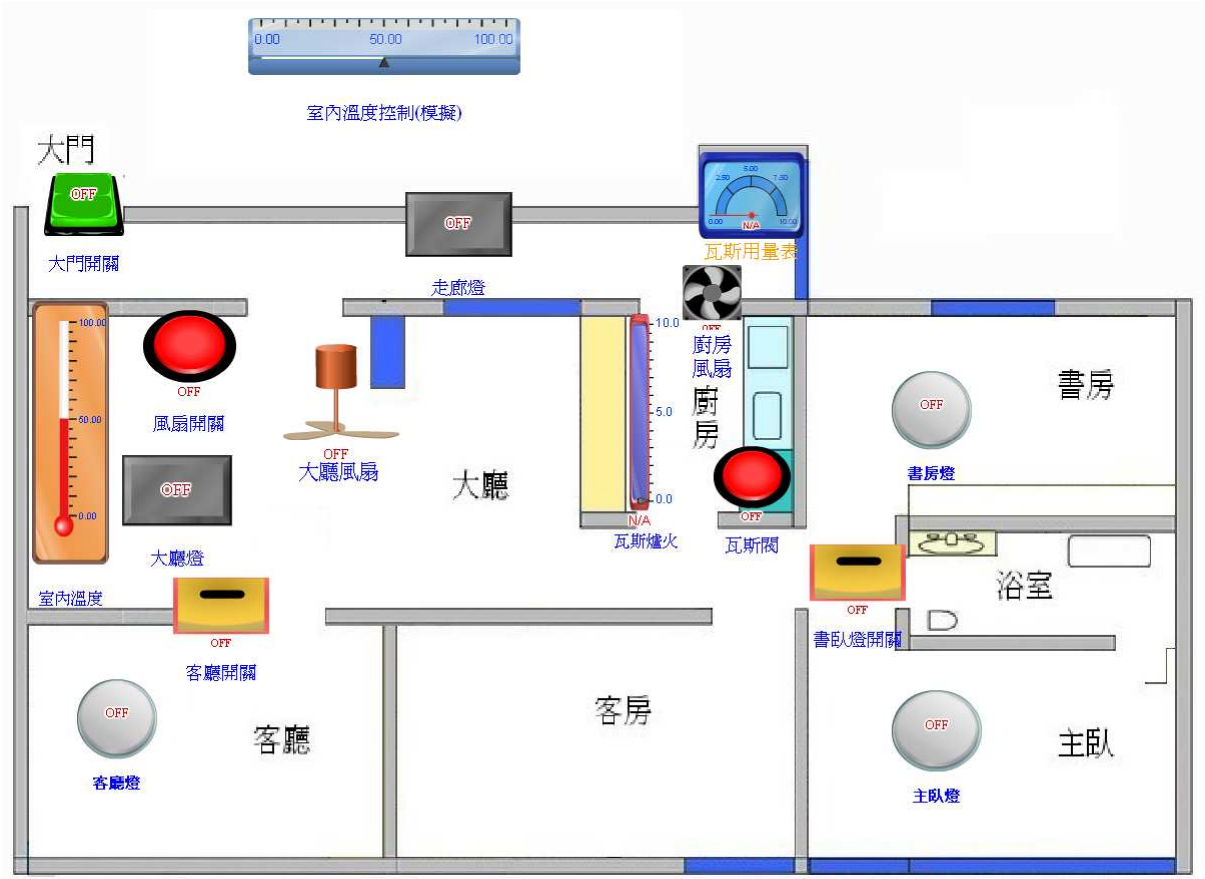
若下載成功，則在輸出視窗上將可看到下載成功的訊息。若發現下載等候時間過長或是出現無法連線的錯誤訊息時，請先檢查網路線是否鬆脫、QPAC 是否已成功取得連線的 IP、QPAC 子專案的 IP 是否有設錯、QPAC 是否正常運作、QPAC 上的 Script Runtime 是否正常運作。

2.4.2.5 步驟 5 – 執行專案

在執行專案前，**請先確認遠端的 QP-500 已經成功運作執行**，如此才能確保專案執行控制器與 HMI 操作介面可交換資訊，使控制器與 HMI 操作介面產生互動。



開啟 Script Editor 專案節點的右鍵選單，由右鍵選單中按下[Start All]，便可同時使兩台 QPAC 上的 Runtime 執行下載後的 Script 檔案。詳細的資訊請參考 Script Editor 入門手冊步驟 4.10。

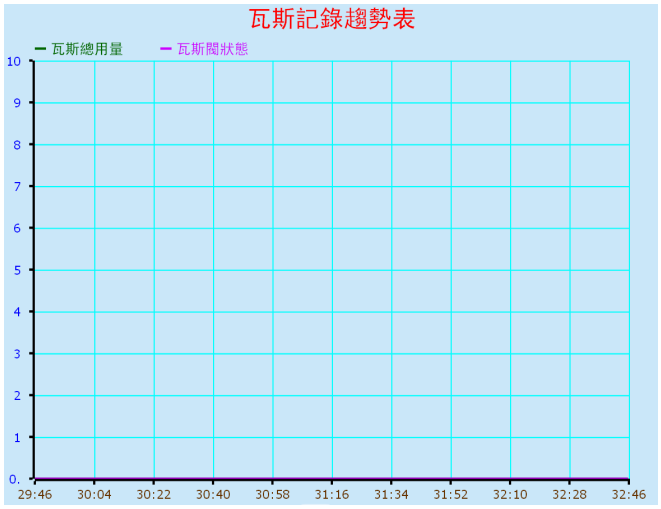


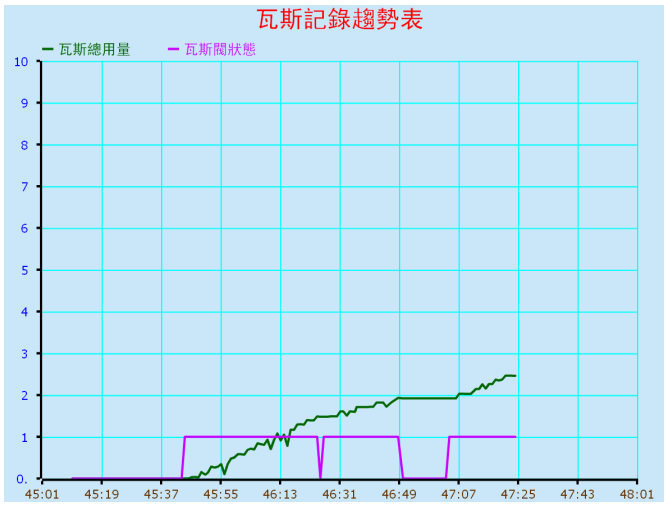
SmartQ Flash HMI Player 可以播放 HMI 控制介面，呈現執行的結果， SmartQ Flash HMI Player 的操作請參考 SmartQ Flash HMI tools 入門手冊第 4 章。載入後的初始頁面如上圖，由於 Control_1 與 Control_2 執行_onStart.js 初始化的指令，室內溫度將設定於第 5 個大刻度，瓦斯爐

火為未使用的狀態，瓦斯用量為零。_onStart.js 可在 Script Runtime 開始執行其他 Script 前，先執行_onStart.js 的程式內容（僅執行一次）。_onStart.js 的程式指令可被視為系統初始事件發生時，必須先行完成的操作。

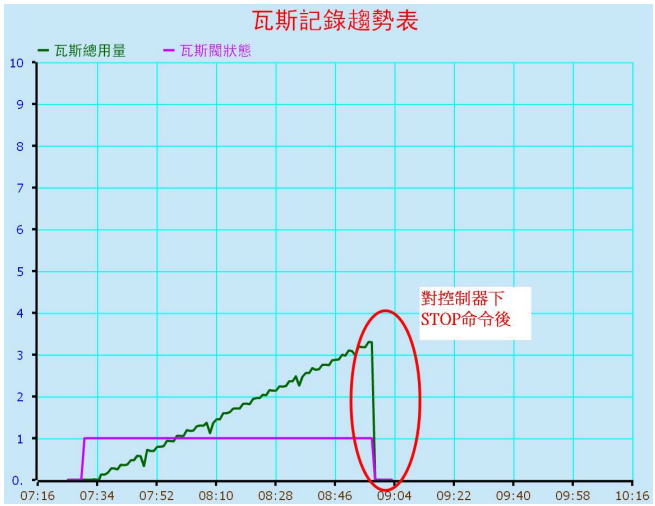
可在瓦斯記錄趨勢表的趨勢圖確認瓦斯用量呈現歸零狀態，且尚未紀錄任何瓦斯用量。開啟瓦斯閥使其為啟動(ON)的狀態，將瓦斯爐火設定為10，等待片刻後再關閉瓦斯閥，重複此動作兩三次，觀察趨勢圖的變化。

由 Script Editor 專案節點的右鍵選單按下[Stop All]後，控制器將執行





_onStop.js 的指令。此時原本正在閃動的燈號或轉動的風扇都將停止運作，呈現關閉的狀態。趨勢表中的兩條曲線：瓦斯總用量及瓦斯閥狀態曲線，也將歸零。藉此示範可知在_onStop.js 中可設定專案停止執行的事件發生前，所有控制器最後必須執行的命令。



2.5 範例 5: HMI 與控制器間事件驅動

2.5.1 範例描述

此專案範例位於光碟目錄：Example\Demo5，請將此專案完整複製到 C:\ICPDAS\SmartQ\Example\Demo5 目錄下，再開啟 Script Editor。Flash HMI 專案目錄則為 Demo5\HMIDemo。

本案例使用兩個控制器，每一個控制器上各有一組 DI/DO 與 AI/AO 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
SmartQ Broker	QP-500	SmartQ Broker 執行設備
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 W-8448 的插槽 0
DI 模組	I-8051W	DI x 16 通道, 安裝在 W-8448 的插槽 1
AO 模組	I-8024W	AO x 4 通道, 安裝在 W-8448 的插槽 2
AI 模組	I-8017HW	AI x 8 通道, 安裝在 W-8448 的插槽 3
PC 或 NB		用於安裝 SmartQ Script Editor 及 SmartQ Flash HMI Editor 工具軟體
電源供應器	DP-665	+10V ~ +30VDC

本案例將上一個案例的人機操作介面做一個小小的增修：增加了瓦斯含量狀態的文字訊息並修改部分元件的通訊設定。其目的同樣是用來模擬如何與一個具備兩台控制器的居家樓房工控系統作互動，利用 HMI 操作介面執行遠端控制並且將取得的遠端控制資訊顯示於操作介面上。

本案例中兩個控制器的運作與上一個案例相異之處為彼此將互動通訊，各個控制器專案所執行的工作內容與目的如下表：

QPAC 名稱	Control_1
工作內容	控制房屋左半區域的大門、走廊、大廳、客廳的燈控與風扇控制
程式目的	實作大門按鈕與客廳按鈕的互斥與風扇運轉限制條件
QPAC 名稱	Control_2
工作內容	控制房屋右半區域的廚房、書房、主臥的燈控與風扇控制
程式目的	實作大門按鈕與書臥室按鈕的互斥、瓦斯使用的控制與用量計算

本案例的情境為前個案情境的延伸，個案情境描述如下：當住戶開啟大門時，走廊燈與大廳燈將自動開啟。而為了節省能源，大廳的風扇將只在室內溫度達到二個大刻度以上時，才能夠運轉。住戶在客廳或書臥室時，大廳燈與走廊燈也將自動關閉。住戶外出時，若忘記關閉客廳燈或書臥室燈，將自動關閉客廳燈或書臥室燈。為避免瓦斯中毒，當住戶在廚房

烹飪時，將自動啟動廚房風扇引入流動空氣。若瓦斯超過使用配額時，則不再提供瓦斯，並將顯示瓦斯已用完的訊息。

依照情境要求，此控制系統的功能規劃如下：

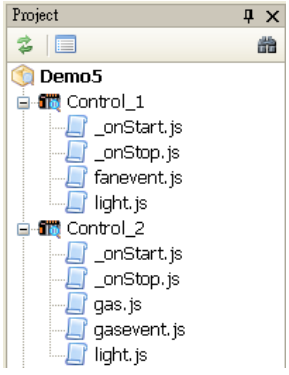
1. 大門開關與大廳燈及走廊燈連動。客廳開關與客廳燈連動。大門開關與客廳開關必須互斥，同時只有一個可呈現 on 的狀態。
2. 客廳風扇將依照風扇開關開與室內溫度的狀態做為啟動的依據。當風扇開關呈現 ON 狀態且室內溫度大於 2 個大刻度時，風扇開關才能作用啟動風扇。
3. 書臥室開關分別與書房燈及主臥燈連動。書臥室開關與大門開關必須互斥，同時只有一個可呈現 on 的狀態。
4. 當瓦斯閥打開時，瓦斯爐火才可使用，廚房風扇將開啟轉動；瓦斯的用量將累計顯示在瓦斯用量表上，若關閉瓦斯閥開關，廚房風扇便停止轉動。
5. 瓦斯用量超過配額時(以 9.9 個單位為上限)，將停止供應瓦斯，關閉瓦斯閥開關，並將瓦斯已用完的訊息顯示於瓦斯用量表旁的文字顯示元件。
6. 繪製即時趨勢圖顯示室內溫度與目前瓦斯用量的紀錄。
7. 依照使用者的權限顯示不同的操作介面。

此案例應用 SmartQ 系統開發工具的功能有：

1. HMI 操作介面的權限管理
2. 以 virtual 資料通道通訊
3. 以事件觸發執行 Script 檔案
4. 使 Script Runtime 啟動後自動執行專案

2.5.2 範例實作

2.5.2.1 步驟 1 – 建立新專案

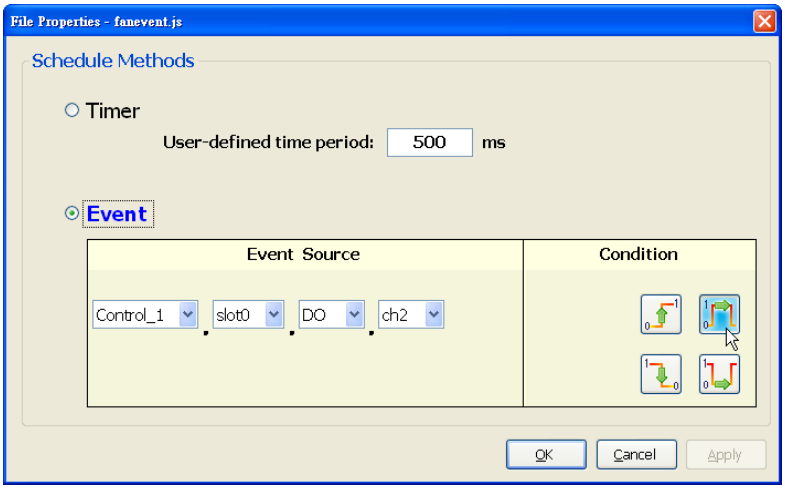


- Script Editor 的檔案結構

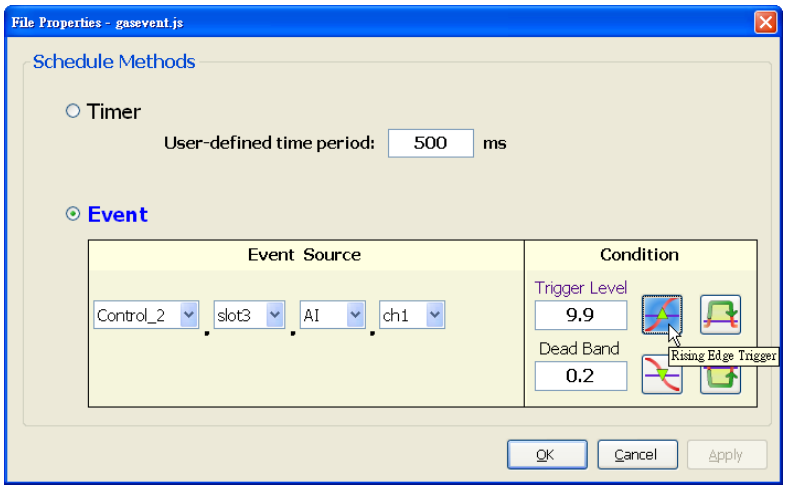
由於此案例中有兩個控制器，因此必須在專案管理員視窗中專案目錄下針對這兩個控制器各自建立一個 QPAC 子專案。專案結構如右圖。（請參考 Script Editor 入門手冊步驟 5.3 與步驟 5.4）

Control_1 控制器下除了預設的 _onStart.js 和 _onStop.js 外，將增加 2 個 Script 檔案，light.js 設定為 Timer：500ms，而 fanevent.js 則設定為事件觸發，觸發事件來源設定為 Control_1.slot0.Do.Ch2，也就是風扇開關，觸發的形態為數位訊號的

Level Trigger(above)。如此當風扇開關開啟(on)時，將觸發事件執行此 script 檔，直到風扇開關轉變為關閉(off)狀態才停止。



Control_2 控制器下除了預設的_onStart.js 和_onStop.js 外，將增加三個 Script 檔案：gas.js 與 light.js 設定為 Timer：500ms；gasevent.js 則設定為事件觸發，觸發事件來源設定為 Control_2.slot3.AI.Ch1，也就是瓦斯用量表，觸發的形態為類比訊號的 Rising Edge Trigger，Trigger Level 為 9.9，Dead Band 設定為 0.2。如此當瓦斯用量表的數字超過 9.9 時，便將觸發事件執行 Script。



Flash HMI Editor 的檔案結構

HMI 操作介面的權限管理，必須於 Flash HMI Editor 進行權限設定，請參考 SmartQ Flash HMI Tools 入門手冊的 3.3.12 章節，點選專案，按右鍵，選擇「權限設定」即可設定專案權限。在群組清單中，新增 manager 與 viewer 兩個群組。每一個群組的權限，由高至低分別為 manager：0、viewer：2、guest：4。並在使用者清單中除了預設的 guest 外另增兩名使用者，分別為 powerusr、viewusr。各群組下各設定

一位使用者，例如：manager 群組下有 powerusr、viewer 群組下則是 viewusr、guest 群組下則為 guest。

Flash HMI Editor 的檔案結構如右圖，如同第四個範例，將建立兩個頁面，每個頁面都將設定登入權限。點選每一個頁面，按下滑鼠右鍵，啟動右鍵選單，選擇設定項目，顯示頁面設定，設定每一頁面的頁面權限。將”Control 頁面”的權限設定為 2，“Record 頁面”的權限也設定為 2。

“Control 頁面”與”Record 頁面”的內容與前案例相似，其差異處為：”Control 頁面”在瓦斯用量表旁新增一個靜態的 Text 類元件，元件樣式如右圖，請參考 Flash HMI tool 入門指南 7.8.2 章節新增 Text 類元件上的說明。屬性將設定為：

- 設定 x:714，y:210，寬:106，高:87
- 在屬性頁上【一般設定】->【基本設定】->「大小」設為 20、「文字內容」設為”gas status”

完成後，"Control 頁面"的人機控制介面設計內容如下圖：

而畫面上各元件的通訊設定如下面列表，粗斜體字型的文字為與前案例相異的設定，可詳細比較其差異之處：

元件名稱	資料通道型態(發送、接收)	資料通道名稱
大門開關	發送、接收	***Control_1.virtual.ch0***
客廳開關	發送、接收	***Control_1.virtual.ch1***
風扇開關	發送、接收	Control_1.slot0.DO.ch2
走廊燈	接收	Control_1.slot1.DI.ch0
大廳燈	接收	Control_1.slot1.DI.ch0
客廳燈	接收	Control_1.slot1.DI.ch1
大廳風扇	接收	Control_1.slot1.DI.ch2
室內溫度控制(模擬)	發送、接收	Control_1.slot2.AO.ch0
室內溫度	接收	Control_1.slot3.AI.ch0
瓦斯閥	發送、接收	Control_2.slot0.DO.ch0
書臥燈開關	發送、接收	***Control_2.virtual.ch0***
廚房風扇	接收	Control_2.slot1.DI.ch0
主臥燈	接收	***Control_2.virtual.ch1***
書房燈	接收	***Control_2.virtual.ch1***
瓦斯狀態文字	***接收***	***Control_2.virtual.ch2***

（瓦斯用量表旁）		
瓦斯爐火	發送、接收	Control_2.slot2.AO.ch0
瓦斯用量表	接收	Control_2.slot3.AI.ch1
瓦斯記錄趨勢表－畫筆 1	接收	Control_2.slot3.AI.ch1
瓦斯記錄趨勢表－畫筆 2	接收	Control_2.slot0.DO.ch0

資料通道設定的步驟請參考 SmartQ Flash HMI tools 入門手冊 3.5.1 章節及前個案的步驟 2 中的「HMI Editor 專案中的元件通訊屬性設定」。

2.5.2.2 步驟 2 – 撰寫 script 程式碼

以下將一一介紹檔案程式，使用者可自行嘗試輸入程式內容。

- Control_1 程式簡介

Example 5-1. Control_1/_onStart.js

```
/**
* FileName: _onStart.js
* Author:
* Version:
* Description:
*
* Date: 2009/3/30
*/

var door_button;
var door_lighter;
var livingroom_button;
var livingroom_lighter;
var hallfan_button;
var hallfan_animation;
var outside_thermometer;
var inside_thermometer;
var old_door_button;
var old_livingroom_button;

var bedroom_button;
var old_bedroom_button;

//讓程式一開始就設定室內溫度在第五個大刻度
outside_thermometer = 5;
Control_1.slot2.AO.ch0 = outside_thermometer;

//讓大門開關、客廳開關、風扇開關呈現爲關閉狀態
door_button = 0;
Control_1.virtual.ch0 = door_button;

livingroom_button = 0;
Control_1.virtual.ch1 = livingroom_button;

hallfan_button = 0;
Control_1.slot0.DO.ch2 = hallfan_button;
```

如範例所示，我們將在 Control_1/_onStart.js 檔案內將溫度控制初始設定設為第五個大刻度(第 23 行)，並且將大門開關、客廳開關、風扇開關設為關閉的狀態（第 29 至 36 行），其他則單純為變數宣告。Control_1/_onStart.js 完整的程式內容，請參考 Example 5-1。

當 Script Runtime 接收 Stop 指令時，將執行_onStop.js 使所有的模組上的值歸零。

Example 5-2. Controller1/_onStop.js

```
1  /**
2  * FileName: _onStop.js
3  * Author:
4  * Version:
5  * Description:
6  *
7  * Date: 2009/3/30
8  */
9
10
11 Control_1.slot0.DO.chT = 0;
12 Control_1.slot2.AO.ch0 = 0;
13 Control_1.slot2.AO.ch1 = 0;
14
15 Control_1.virtual.ch0 = 0;
16 Control_1.virtual.ch1 = 0;
17 Control_1.slot0.DO.ch2 = 0;
```

Control_1/fanevent.js 完整的程式內容，請參考 Example 5-3。由於 fanevent.js 為透過風扇開關開啟的事件驅動執行，故不必如前個案的 Example 4-3 先透過 if 判斷式確認風扇開關是否開啟，只需如 Example 5-3 第 16 行程式碼；當事件觸發執行時，判斷室內溫度是否小於兩個大刻度，若是則停止風扇的運轉。

Example 5-3. Control_1/fanevent.js

```
1  /**
2  * FileName: lightEven.js
3  * Author:
4  * Version:
5  * Description:
6  *
7  * Date: 2009/4/15
8  */
9
10 hallfan_button = Control_1.slot0.DO.ch2;
11 hallfan_animation = Control_1.slot1.DI.ch2;
12 inside_thermometer = Control_1.slot3.AI.ch0;
13
14 //設定客廳風扇只有在風扇開關on且室內溫度在兩個大刻度以上時才可以開啓
15
16  if ( inside_thermometer < 2 )
17  {
18      hallfan_button = 0;
19      Control_1.slot0.DO.ch2 = hallfan_button;
20  }
```

Control_1/light.js 完整的程式內容，請參考 Example 5-4。在 Example 5-4 程式中，第 10 至 19 行程式碼主要透過變數記錄大門開關、客廳開關、書臥燈開關，其中為了幫助狀態判別，在第 10 至 16 行程式碼分別記錄大門與客廳開關的新舊兩個狀態值。由於大門開關與客廳開關為透過 virtual 資料通道發送訊息給 Control_1 控制器，並未設定硬體資料通道，故需要透過第 12 與 16 行程式碼將各個由 virtual 資料通道所接收的資料設定至硬體資料通道，使控制器能改變 DO 模組的狀態。以外，因需

Example 5-4. Control_1/light.js

```
1  /**
2  * FileName: light.js
3  * Author:
4  * Version:
5  * Description:
6  *
7  * Date: 2009/3/31
8  */
9
10 old_door_button = door_button;
11 door_button = Control_1.virtual.ch0;
12 Control_1.slot0.DO.ch0 = door_button;
13
14 old_livingroom_button = livingroom_button;
15 livingroom_button = Control_1.virtual.ch1;
16 Control_1.slot0.DO.ch1 = livingroom_button;
17
18 old_bedroom_button = bedroom_button;
19 bedroom_button = Control_2.virtual.ch0;
20
21 door_lighter = Control_1.slot1.DI.ch0;
22 livingroom_lighter = Control_1.slot1.DI.ch1;
23
24 //客廳開關與大門開關互斥設定
25 if ( door_button == 1 )
26    {
27     if ( livingroom_button == 1 || bedroom_button == 1 )
28        {
29         //進客廳
30         if ( old_door_button == 1 && old_livingroom_button == 0)
31          {
32             door_button = 0;
33             Control_1.virtual.ch0 = door_button;
34             Control_1.slot0.DO.ch0 = door_button;
35          }
36
37         //出門時
38         if ( old_door_button == 0 )
39           {
40              if ( old_livingroom_button == 1 ||  old_bedroom_button ==1 )
41              {
42                  livingroom_button = 0;
43                  bedroom_button = 0;
44                  Control_1.virtual.ch1 = livingroom_button;
45                  Control_1.slot0.DO.ch1 = livingroom_button;
46                  Control_2.virtual.ch0 = bedroom_button;
47               }
48           }
49       }
50    }
51
```

加入與書臥燈開關的互斥操作，在第 18 至 19 行程式碼將取得 Control_2 控制器上代表書臥燈開關的 virtual 資料通道的資訊。

第 25 至 50 行程式碼則為控制大門開關與客廳開關或書臥燈開關互斥的指令。當大門與客廳開關或大門與書臥燈開關的狀態都為開啟時，將依照開關舊的狀態來判斷哪一個開關較晚變更為開啟狀態，並將已經開啟的開關更改為關閉狀態，使大門與客廳開關或大門與書臥燈僅有一個可呈現開啟的狀態。第 46 行程式碼使大門燈開啟時，Control_1 可透過 virtual 資料通道，命令 Control_2 上的書臥燈開關關閉，並改變 HMI 上的書臥燈開關狀態。藉此程式，使用者將可瞭解控制器如何透過 virtual 資料通道控制遠端控制器的運作。

Control_2 程式簡介

在 Control_2/_onStart.js 的程式中，以第 23、24 行將瓦斯爐火初始狀態設為關閉，瓦斯用量因為尚未使用也將歸零，如第 26、27 行程式碼所示。Control_2 子專案中，書臥燈開關、書房燈、主臥燈以及瓦斯狀態文字都將透過 virtual 資料通道通訊，故在第 29 至 34 行程式碼，將分別執行初始化的操作。由第 33、34 行程式碼可得知 virtual 資料通道可用來傳送數值的資料(整數或小數)以及文字字串的資料。Control_2/_onStart.js 完整的程式內容請參考 Example 5-5。在 Example 5-6 的程式中，將在專

Example 5-5. Control_2/_onStart.js

```
1  /**
2  * FileName: _onStart.js
3  * Author:
4  * Version:
5  * Description:
6  *
7  * Date: 2009/4/1
8  */
9
10 var gas_valve;
11 var bedroom_button;
12
13 var kitchenfan_animation;
14 var old_bedroom_button;
15 var bedroom_lighter;
16
17 var gasstove_comsumption;
18 var gasgage_amount;
19
20 var old_door_button;
21 var door_button;
22
23 gasstove_comsumption = 0;
24 Control_2.slot2.AO.ch0 = gasstove_comsumption;
25
26 gasgage_amount = 0;
27 Control_2.slot2.AO.ch1 = gasgage_amount;
28
29 bedroom_button = 0;
30 Control_2.virtual.ch0 = bedroom_button;
31
32 bedroom_lighter = 0;
33 Control_2.virtual.ch1 = bedroom_lighter;
34 Control_2.virtual.ch2 = "gas unempty";
```

案停止運行時，將控制器上的輸出與輸入模組狀態歸零，以確認專案停止運作。

Example 5-6. Control_2/_onStop.js

```
/**
* FileName: _onStop.js
* Author:
* Version:
* Description:
*
* Date: 2009/4/1
*/


Control_2.slot0.DO.chT = 0;
Control_2.slot2.AO.ch0 = 0;
Control_2.slot2.AO.ch1 = 0;
gasgage_amount = 0;

Control_2.virtual.ch0 = 0;
Control_2.virtual.ch1 = 0;
Control_2.virtual.ch2 = "gas unempty";
```

Example 5-7. Control_2/gas.js

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

gas_valve = Control_2.slot0.DO.ch0;
gasstove_comsumption = Control_2.slot2.AO.ch0;


if ( gas_valve == 1)
{
    gasgage_amount = gasgage_amount + (gasstove_comsumption / 200);

    if ( gasgage_amount > 10 || gasgage_amount == 10 )
    {
        gasgage_amount = 0;
        gas_valve = 0;
        Control_2.slot0.DO.ch0 = gas_valve;
    }
    Control_2.slot2.AO.ch1 = gasgage_amount;
}
else
{
    gasstove_comsumption = 0;
    Control_2.slot2.AO.ch0 = gasstove_comsumption;
}
```

Control_2/gas.js 主要用來判斷當瓦斯閥狀態為開啟與關閉時所要執行的工作。瓦斯閥開啟時，將執行第 15 至 26 行的程式，將每單位時間內

的瓦斯用量加總，並顯示於人機控制介面的瓦斯用量表以及趨勢圖上。第 18 至 23 行程式碼則是當瓦斯總用量超過最大用量時，便將瓦斯閥關閉不再提供瓦斯。瓦斯閥關閉時，便執行第 28、29 行程式碼關閉瓦斯爐火。

Example 5-8 實作的功能為：當瓦斯用量超過額度，將顯示訊息提醒使用者瓦斯已用罄。當瓦斯用量表超過 9.9 個單位時，自動觸發執行 gasevent.js 中的程式碼，修改瓦斯狀態文字並顯示瓦斯已用完的訊息。

Example 5-8. Control_2/gasevent.js

```
1  /**
2   * FileName: gasevent.js
3   * Author:
4   * Version:
5   * Description:
6   *
7   * Date: 2009/4/16
8   */
9
10  Control_2.virtual.ch2 = "gas empty";
```

Example 5-8. Control_2/light.js

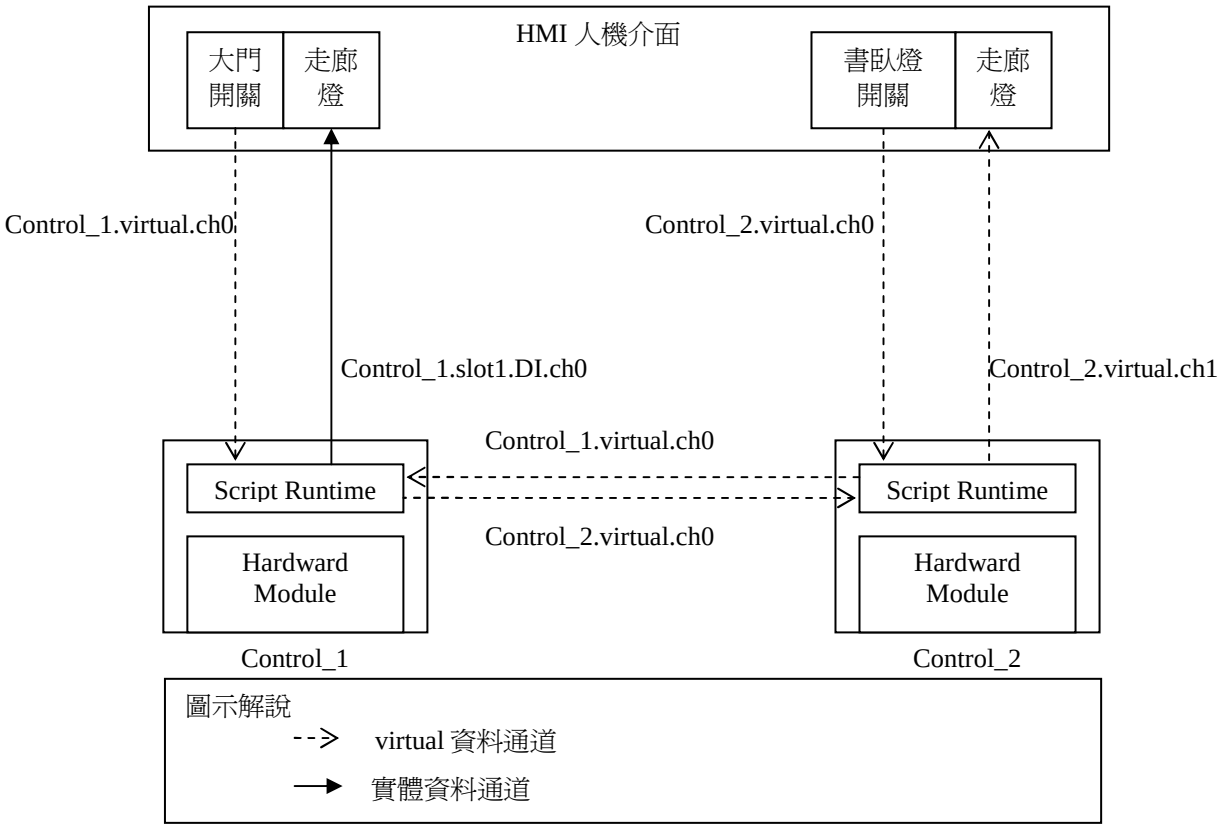
```
1  /**
2   * FileName: light.js
3   * Author:
4   * Version:
5   * Description:
6   *
7   * Date: 2009/4/14
8   */
9
10  old_bedroom_button = bedroom_button;
11  bedroom_button = Control_2.virtual.ch0;
12  Control_2.slot0.DO.ch1 = bedroom_button;
13  bedroom_lighter = Control_2.slot1.DI.ch1
14  Control_2.virtual.ch1 = bedroom_lighter;
15
16  old_door_button = door_button;
17  door_button = Control_1.virtual.ch0;
18
19  //書臥室開關與大門開關互斥設定
20  if ( door_button == 1 && bedroom_button ==1 )
21     {
22       //進書臥室
23       if ( old_door_button == 1 && old_bedroom_button == 0)
24       {
25           door_button = 0;
26           Control_1.virtual.ch0 = door_button;
27       }
28
29     }
```

light.js 為執行大門開關與書臥燈開關必須擇一開啟，無法同時開啟的互斥情境。第 10、11 行以及第 16、17 行程式碼將分別記錄書臥燈開關與大門開關的新與舊狀態。第 12 至 14 行程式碼為執行將書臥燈開關的狀態寫入對應的 DO 模組硬體資料通道，改變硬體 DO 模組上的訊號。並

由 DI 模組硬體資料通道讀取：因應書臥燈開關的狀態，書房燈與主臥燈所需更改的狀態。再透過 Control_2 的 virtual 資料通道，通知遠端的 HMI 燈號顯示元件；並使燈號顯示元件依照資訊更改燈號狀態。使用者可以由第 11、14、17 行程式碼瞭解如何取得與設定本機端與遠端控制器的 virtual 資料通道。

第 20 至 29 行程式碼則依照大門開關與書臥燈開關新舊狀態判斷；當書臥燈開關轉換為開啟時，是否大門開關也呈現開啟狀態？若為肯定，則將大門開關設為關閉狀態。

藉由 Control_1/light.js 與 Cotnrol_2/light.js 的設定可達成下圖的溝通環境，使用者可藉此個案了解如何以 SmartQ 開發工具建構通訊環境，以及如何建立硬體與 virtual 資料通道間的通訊。

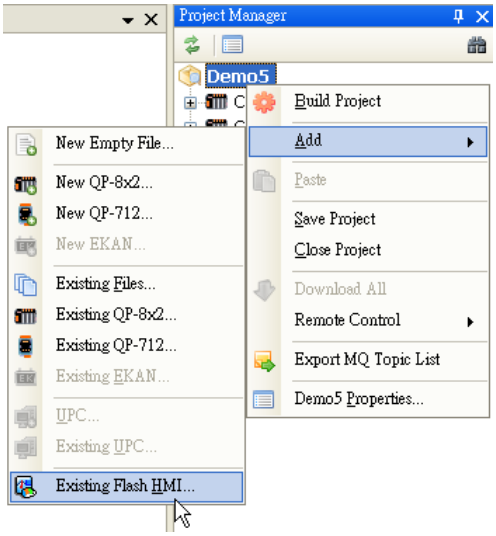


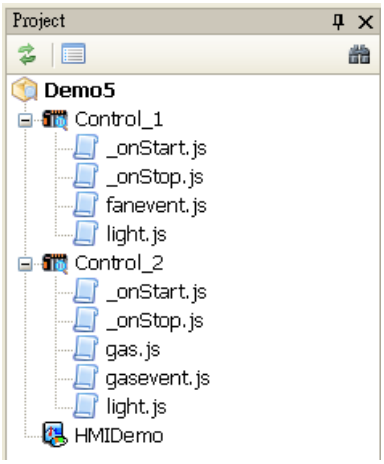
接下來的步驟大致與前個案相似；為了使 HMI Editor 所開發的操作介面與遠端控制器通訊，必須如前述建立 HMI Editor 的專案與元件通訊設定。除此之外，也必須在 Script Editor 專案上設定 HMI Editor 專案的專案路徑以便匯入通訊設定。

指定與 Flash HMI 通訊對照的專案路徑

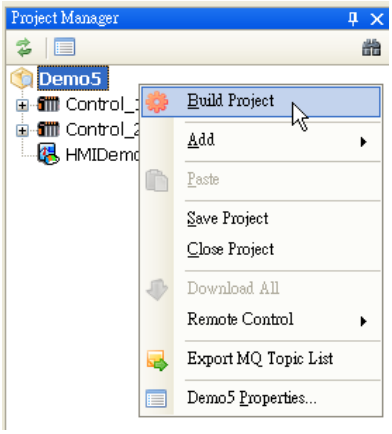
此部份操作請參考 SmartQ Script Editor 入門手冊第 4.6 章節，在 Script Editor 的專案管理員視窗中，選擇專案的節點，點選並按滑鼠右鍵，顯示右鍵選單。由右鍵選單中按下 Add > Flash HMI，加入 Flash

HMI 專案檔案夾路徑，Script 執行時便能與 Flash HMI 互相溝通。此時在專案管理員視窗中將可看到新增的 Flash HMI 專案節點。



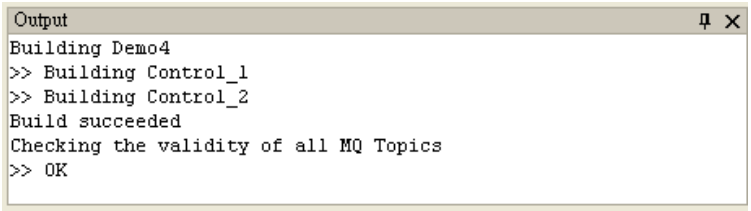


2.5.2.3 步驟 3 – 編譯專案



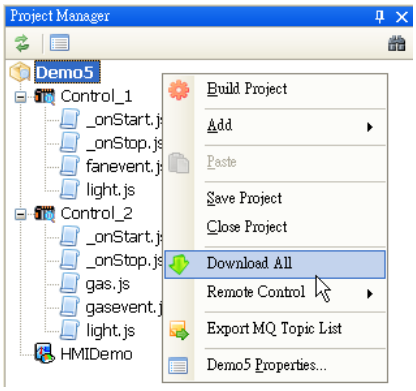
在此案例中，兩台 QPAC 控制器，除將各自獨立執行 Script 外，也將與 Flash HMI 介面通訊。因此每次編譯專案前，請務必確認專案管理員視窗上已設有 Flash HMI 專案節點、設定的 Flash HMI 專案路徑正確、且 Flash HMI Editor 所建立的專案也已存檔。

確認以上設定無誤後，便可進行編譯程序(請參考 Script Editor 入門手冊步驟 5.8)。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），將可在輸出視窗上看到編譯成功的訊息(Build succeeded)。編譯成功後將自動檢查 Flash HMI 專案所設定 QP-500 的 IP 位址與資料通道是否為有效的設定，若正確無誤將顯示 ok 的訊息。



2.5.2.4 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 Script Editor 入門手冊步驟 5.9)，必須下載全部；直接在專案管理員視



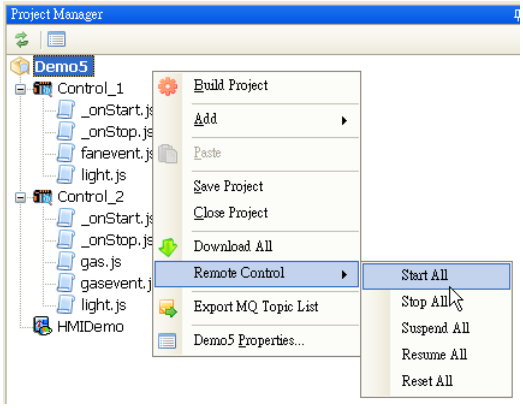
窗中，選擇專案的節點，點選並按滑鼠右鍵，顯示右鍵選單，從右鍵選單中按下[Download All]。

若下載成功，則在輸出視窗上將可看到下載成功的訊息。若發現下載等候時間過長或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。

2.5.2.5 步驟 5 – 執行專案

在執行專案前，**請確認遠端的 QP-500 已經成功運作執行**，如此才能確保專案執行後可交換控制器與 HMI 操作介面的資訊；控制器與 HMI 操作介面可互動。

再開啟 Script Editor 專案節點的右鍵選單，由右鍵選單中按下[Start All]，便可同時讓兩台 QPAC 上的 Runtime 執行下載後的 Script 檔案。詳細的資訊請參考 Script Editor 入門手冊步驟 5.10。



SmartQ Flash HMI Player 可以播放 HMI 控制介面，呈現執行的結果，請參考 SmartQ Flash HMI tools 入門手冊的第 5 章。載入後的初始頁面，可嘗試以權限不同的使用者登入（依照先前在 Flash HMI 工具所設的權限設定），觀察人機介面所呈現的差異。如：使用者可勾選「訪客登入」以 guest 帳號登入，由於 guest 帳號在此個案中設定為最低權限，所有的頁面權限設定皆高於 guest 帳號所設的權限，因此登入後將出現無瀏覽權限的訊息。若使用者以 viewusr 帳號登入，則可瀏覽 "Control 頁面" 和"Record 頁面"，但無法操作介面。若以 powerusr 帳號登入，便可操作介面，透過 HMI 人機介面來達成與遠端控制器互動的工作。

若所設計的 Script 以及 HMI 人機介面執行後確認無誤，便可在 Script Editor 上 QPAC 的專案屬性設定勾選 Settings 頁籤上的[Auto Execution When Script Runtime Start Up]，使控制器啟動時便開始執行 Script。設定完成後，重新完成專案編譯並下載至控制器更新檔案，再重新開啟 QPAC，此時，若使用者重新登入 HMI 人機介面便可發現控制器上的

Script 已在執行的狀態，使用者只需透過控制類元件即可以與遠端控制器互動。