# 實驗指南

## 背景、流程及解決方案與符號

### 練習背景：

本課程共有6個實驗。客戶可選擇3個MCU平臺（內建MAC/PHY的PIC18；PIC24 或 PIC32——通過 PICTail Plus 模組外接 MAC/PHY——由此處所述的HardwareProfile.h實現）之一完成練習。實驗1：連接網路——利用一個應用程式來設置開發板，採用指定的MAC位址和客戶選擇的開發板來讓實驗得以順利進行。在進行後續實驗時需要確認網路是連通的。實驗2是一個關於設計HTML/CSS網頁的練習。但我們爲客戶直接進行實驗3至實驗6提供了備選網頁。最後4個實驗指導客戶完成了以下過程：將其應用與TCP/IP協議層結合，刪除了垃圾程式，進而達到最佳性能，以及爲監控遠端應用進行時用設計。

**專案目錄：C:\RTC\COM4201\Microchip Solutions\TCPIP Demo App**

### 練習流程：

完成實驗1是完成所有後續實驗所必需的，因爲要重複使用TCPIPConfig.h檔來設定硬體。實驗2是自包含的，無需保存，在後續實驗中將刪除這些檔。一旦打開了MPLAB中的專案，在實驗1至實驗6中便無需再關閉，因爲此項目是爲實現最後的監控應用而建立的。因此，無需重複打開專案和選擇除錯器這兩步驟。

這些練習所使用的所有檔案及Microchip發佈的v4.50版TCP/IP協議層已載入到C\RTC\COM4201\Microcchip Solutions\目錄中。從網站上下載此產品協定層並進行安裝時，該協議層將保存在C:\Microchip Solutions文件夾中。除實驗2外，練習期間對程式作出的所有修改都是在TCPIP Demo App目錄中進行的，其位置如上所述。開始實驗3時，將打開此目錄中的TCPIP Demo App專案，並根據設計構建的需要，進行修改。在這些練習中還有一些其他專案目錄。TCPIP Demo App和自動販賣機應用程式示範隨本產品協定層一起提供，以允許在將來實現增強功能。

## 解決方案：

所有實驗都附帶有解決方案檔案夾。要體驗各個實驗結果，請參考下面的解決方案步驟編號部分。實驗3至實驗6都配備有解決方案，可用作繼續後續實驗的起始點。**如果需要，可關閉現有專案，打開前面所述的相應解決方案專案，從此步開始將此專案作爲當前專案的構建基礎。解決方案包含完成的附加程式。**

載入解決方案專案後的起始步驟：

實驗1：第7步；實驗2：第9步：實驗3：第10步：實驗4：第6步；實驗5：第15步；實驗6：第7步

# 程式分析——HardwareProfile.h

無需修改檔即可支援您的硬體選擇。由HardwareProfile.h檔實現（下面列出了該檔的一部分），選定的編譯器將根據選擇的部件自動選擇Microchip開發平臺。但如果您選擇開發自己的硬體，將需要設置此檔。

## HardwareProfile.h

```
// Choose which hardware profile to compile for here.  See
// the hardware profiles below for meaning of various options.
//#define PICDEMNET2
//#define HPC_EXPLORER
//#define PICDEMZ
//#define PIC24FJ64GA004_PIM   // Explorer 16, but with the PIC24FJ64GA004 ...
//#define EXPLORER_16          // PIC24FJ128GA010, PIC24HJ256GP610, ...
//#define DSPICDEM11
//#define DSPICDEMNET1         // Not currently supported, wrong Ethernet ...
//#define DSPICDEMNET2         // Not currently supported, wrong Ethernet ...
//#define YOUR_BOARD

// If no hardware profiles are defined, assume that we are using
// a Microchip demo board and try to auto-select the correct profile
// based on processor selected in MPLAB
#if !defined(PICDEMNET2) && !defined(HPC_EXPLORER) && !defined(PICDEMZ) && ...
    #if defined(__18F97J60) || defined(_18F97J60)
          #define PICDEMNET2
    #elif defined(__18F67J60) || defined(_18F67J60)
          #define INTERNET_RADIO
    #elif defined(__18F8722) || defined(__18F87J10) || defined(_18F8722) ...
          #define HPC_EXPLORER
```

## 本實驗手冊使用下列符號引導客戶完成所有練習：


幫助完成實驗的輔助資訊


實驗練習的背景和目的


練習所需的硬體和軟體


1 要執行的步驟命令

## 要編輯或複製的原始檔案位置


用於編輯和分析的ANSI-C程式段或解決方案


用於編輯和分析的HTML/CSS程式段或解決方案


討論練習的目標結果


程式分析和練習的結論

# 實驗 1

## 連接網路

## 目的

要將元件連接到網路，必須爲其分配一個唯一的主機名稱和MAC位址，從而確保該元件能夠與網路中的其他節點進行獨立的通信，並防止網路中有重複的位址。

在這第一個實驗中，將使用**TCPIPConfig Wizard（TCPIP設定精靈）**爲開發硬體設定唯一的地址。這些更改將被寫入名爲`TCPIPConfig.h`的設定檔中，該檔包含在專案檔中，之後將被編譯到`TCPIP Demo App`專案中。要完成本實驗，需要上傳一系列示範網頁並檢測網路連接。

## 要求

| | |
|---|---|
| **軟體：** | **TCPIPConfig Wizard** |
| **環境 ：** | MPLAB® IDE 8.01和COM4201 Lab Build Installer 1.01 |
| **C編譯器：** | MPLAB C18 v3.16 或更高版本 /C30 v3.01或更高版本 /C32 v1.02 或更高版本 |
| **硬體工具：** | PICDEM.net™ 2 或 帶乙太網 PICtail™ Plus 及 PIC24FJ128GA010 或 PIC32MX360F512L PIM 的 Explorer 16開發板 |

## 步驟

1. 如果您已打開了一個MPLAB項目，則必需首先通過選擇下列功能表將其關閉：

   File ▸ Close Workspace

   **專案目錄：**
   **C:\RTC\COM4201\Microchip Solutions\TCPIP Demo App**

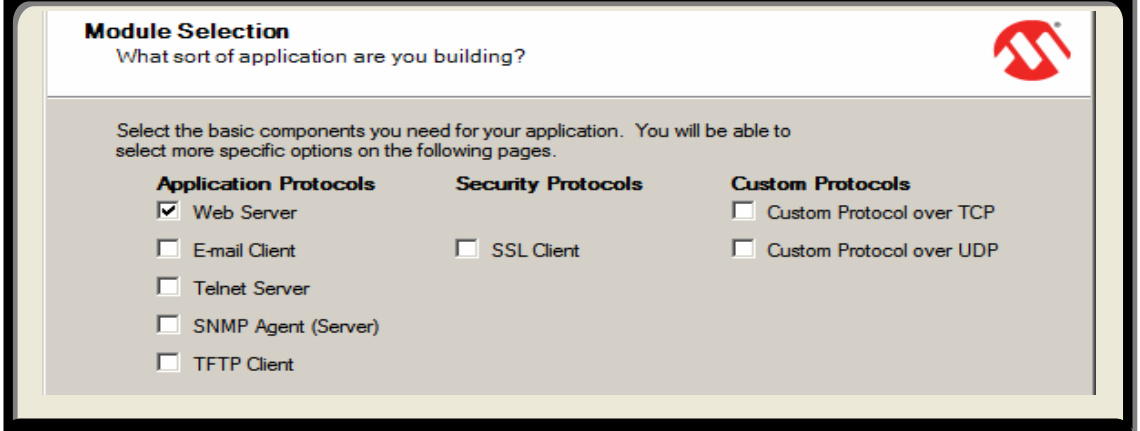
2. 啓動 **TCPIP Config Wizard** 來修改 TCPIPConfig.h。

   開始 } 所有程式 ▸ Microchip ▸ COM4201 ▸ TCPIPConfig Wizard
   該精靈也可在C:\RTC\COM4201\Microchip Solutions\Microchip\TCPIP Stack\Utilities中找到

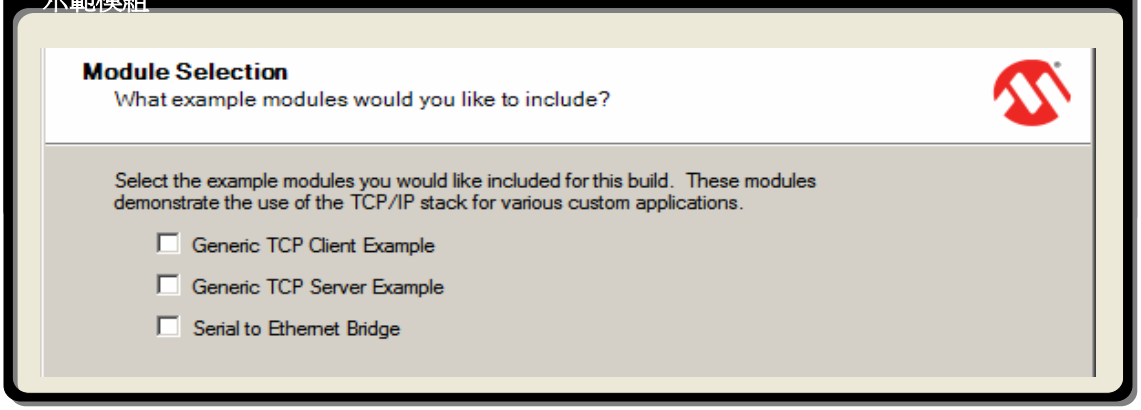
   瀏覽到上述檔案夾。確認**未點選中Show Advanced Settings**（顯示進階設定）。

3 點選 **Next**（下一步）完成以下兩個螢幕，確認下面指示的選項。

模組選擇




示範模組



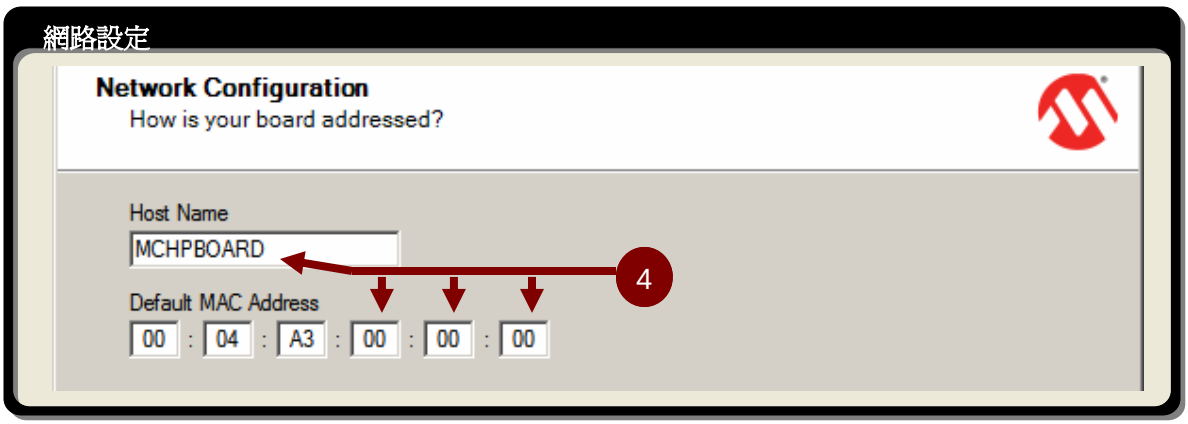

4 **將下列資訊輸入到 NETWORK CONFIGURATION（網路設定）螢幕，如下所示：**

在下個螢幕中使用最多15個字母、數位和/或連字元爲開發板**選擇**並**輸入**一個唯一的主機名。把這個名稱記錄到下面的橫線上備用。

**選定的主機名：** _____

以十六進位方式輸入MAC位址欄位的最後3個位元組，使其對應于開發板或PicTail plus的十進位乙太網序列號。

網路設定



乙太網序列號印製在開發板背面的貼紙上，或粘貼在 PICTail Plus 模組上。PICDEM.net 2 有兩個序列號，您可選擇使用其中任一個。如果沒有序列號，則請選擇一個唯一的6位十六進位數，並繼續。

要將乙太網序列號轉換爲十六進位的MAC位址，請使用Windows計算器將此十進位值轉換爲等效的十六進位值，並使用0填充。例如：將12345轉換爲十六進位值，得00:30:39。

5. 點選**Next**完成隨後的螢幕，然後點選**Finish**（完成）即可完成本對話。

6. 執行批次處理腳本，以將此開發板設定複製到所有解決方案專案檔夾以繼續在本課程中使用。

   開始 } 所有程式 ▸ Microchip ▸ COM4201 ▸ Replicate TCPIPConfig.bat

C:\RTC\COM4201\Microchip Solutions\TCPIP Demo App*.mcp

7. 打開**MPLAB**，並載入適合您開發板設置的專案：

   Project ▸ Open

   **TCPIP Demo App-C18.mcp**用於PICDEM.net 2設置
   **TCPIP Demo App-C30.mcp**用於PIC24FJ128GA010 PIM的Explorer 16開發板
   **TCPIP Demo App-C32.mcp**用於PIC32MX360F512L PIM的Explorer 16開發板

對於第8步和第9步，您可選擇使用MPLAB中的圖示，如圖所示。

8. 編譯程式並選擇 **ICD2或 Real ICE** 作爲除錯工具：

   Project ▸ Make，然後選擇 Programmer ▸ Select Programmer ▸ MPLAB ICD 2或 REAL ICE

9. 燒錄選項 Programmer ▸ Program

## 關於網路設置

本範例應用程式將網路設定設置存儲在EEPROM中，因此只有當EEPROM中的資料遺失時，才會載入存儲在專案檔中的設置。僅當爲專案設定了新的網路設置時才需要執行下一步。由於MPLAB ICD2 和 Real ICE 除錯器控制主復位（/MCLR）接腳，因此僅當除錯器與目標系統斷開時此步驟才有效。此外，此步驟還要求元件工作在燒錄模式下。如果您已使用了除錯模式，則應切換到燒錄模式，並重新對元件進行新的燒錄動作。

10 通過斷開目標系統的燒錄器電纜將元件從重置模式中釋放。 確認路由器(Router)爲開發板分配了一個新的IP位址。

11 通過以下步驟清除EEPROM的內容：

1. 保持按下最右側的按鈕（在PICDEM.net 2中爲S5，在Explorer 16開發板中爲S4）
2. 按下然後釋放 MCLR按鈕 (Reset )（在PICDEM.net 2和Explorer 16開發板中均爲S1）
3. 繼續保持按下最右側的按鈕，直到有幾個LED閃爍爲止

**C:\RTC\COM4201\Microchip Solutions\TCPIP Demo App\MPFSImg2.bin**

12 打開網頁瀏覽器，通過在位址欄中輸入**http://*hostname*/mpfsupload**來**執行**網頁的上傳功能 。使用第4步中定義的主機名。

出現提示時，將 **`MPFSImg2.bin`**網頁資料上傳到目標系統。

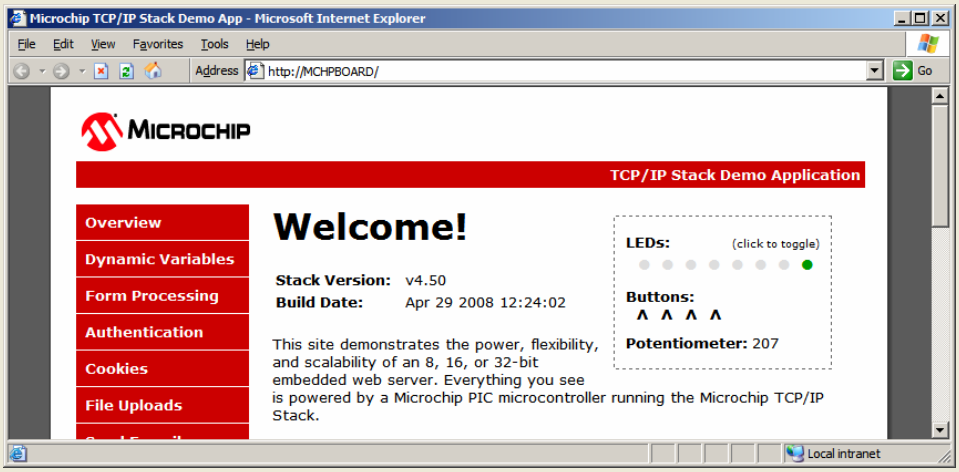
點選 **Site Main Page（站點主頁面）** 鏈結。瀏覽該示範網站，尤其要注意右上角，在這裏您可確認開關、電位器和LED的操作。

## 結果

已在開發板上設置了TCPIP Demo App專案，並上傳了一組示範網頁。在完成後，您應能看到類似於“示範網頁顯示”中所示螢幕的頁面。

本示範網頁可演示Microchip TCP/IP協定層的多種功能。如果您完成的較快，則可以嘗試其各種特性。可通過右上角的狀態框控制開發板上的LED。（如果您選擇了Explorer 16 開發板，最左側的LED將與一個按鈕共用同一接腳 ，因此其顯示可能不正常，且不可控制 。）當目標系統中按鈕和電位器的狀態發生變化後，開發板將返回它們的狀態。

示範網頁顯示




# 程式分析

TCP/IP Config Wizard應用程式用於通過巨集的定義（`#define`）修改網路軟體的設定配置，此巨集存儲在`TCPIPConfig.h`中。此**應用程式**通過設計人員輸入來修改專案中的頭檔，但也可通過文本編輯器進行查看和編輯。打開`TCPIPConfig.h`檔，並找到該工具存放更新後的主機名和MAC位址的位置。您還將在此檔中看到許多其他設置，其中多數可通過使能此應用程式中的進階設置來進行設定。正如先前討論過的那樣，硬體設定的處理方式類似於`HardwareProfile.h`文件。

## TCPIPConfig.h

```
141 // ============================================================================
142
143 /* Default Network Configuration
144  *    These settings are only used if data is not found in EEPROM.
145  *    To clear EEPROM, hold BUTTON0, reset the board, and continue
146  *    holding until the LEDs flash.  Release, and reset again.
147  */
148 #define MY_DEFAULT_HOST_NAME            "MCHPBOARD"
149
150 #define MY_DEFAULT_MAC_BYTE1            (0x00)
151 #define MY_DEFAULT_MAC_BYTE2            (0x04)
152 #define MY_DEFAULT_MAC_BYTE3            (0xA3)
153 #define MY_DEFAULT_MAC_BYTE4            (0x00)
154 #define MY_DEFAULT_MAC_BYTE5            (0x4C)
155 #define MY_DEFAULT_MAC_BYTE6            (0xA9)
156
157 #define MY_DEFAULT_IP_ADDR_BYTE1        (169ul)
158 #define MY_DEFAULT_IP_ADDR_BYTE2        (254ul)
159 #define MY_DEFAULT_IP_ADDR_BYTE3        (1ul)
160 #define MY_DEFAULT_IP_ADDR_BYTE4        (1ul)
161
162 #define MY_DEFAULT_MASK_BYTE1           (255ul)
```

4

# 結論

本實驗已確認了您的網路設置和開發板是正常的。已爲您的開發板分配了唯一的MAC位址以及主機名（可獨立于開發板的IP位址單獨瀏覽）。通過NetBIOS命名服務管理主機名，此服務僅用於本地子網（在本實驗中爲教室）。可通過DNS訪問子網外部的主機名，這將在以後討論。

LCD螢幕上顯示的IP位址是由本地路由器中的DHCP伺服器自動分配的。DHCP爲多數網路的標準，它取代了對靜態IP位址的硬體編碼。如果需要，可設定靜態IP位址。

# 實驗2

## 創建簡單的網頁

## 目的

當今的網頁都是混合使用HTML和CSS程式創建的。HTML是一種標記語言，這意味著其內容將被劃分爲某種結構，其中每個元素都具有特定的類型。CSS是一種樣式語言，用於指定以何種方式呈現HTML結構中的內容。搭配這兩種語言，可通過網頁瀏覽器解析這些檔以建立螢幕顯示。

在本實驗中，您將創建一個簡單的網頁，其中包含您的姓名、關於您的一段簡短描述和一些視覺元素。通過您個人電腦（PC）上的網頁瀏覽器檢測頁面。

Microchip嵌入式平臺的網頁已編譯爲Microchip檔系統（Microchip File System，MPFS）映射。此映射類似於壓縮檔，並包含以易於PIC元件處理的格式存儲的所有網頁程式。要完成本實驗，您需要使用**MPFS2應用程式**將您的網頁打包並上傳到開發板，從而允許其他參與者在他們的PC上也能查看此頁面。

如果您完成的較快，附加部分將教授您一些基本的CSS知識，來幫助您設置頁面的格式。

## 要求

| | |
|---|---|
| **軟體：** | **完成的實驗1和Crimson Editor** |
| **環境：** | COM4201 Lab Build Installer 1.01 |
| **C編譯器：** | 本實驗不需要 |
| **硬體工具：** | PICDEM.net™ 2 或 帶乙太網PICtail™ Plus 及 PIC24FJ128GA010 或 PIC32MX360F512L PIM 的 Explorer 16 開發板 |

**網頁目錄：**
**C:\RTC\COM4201\Microchip Solutions\TCPIP Demo App\WebPages2**

## 步驟

1. **刪除**位於WebPages2目錄中的所有檔，如下所示。這些檔是隨TCP/IP 協議層一起提供的，本課程中不需要， 但需要使用相同的檔夾來存放新頁面。

   Windows桌面 } 我的電腦 } 導航到 WebPages2 } Edit } Select All } Edit } Delete

2 啓動 Crimson Editor來 建立一個名爲 **index.htm** 的新檔，並將其保存到前面提及的 **WebPages2** 中。

開始 } 所有程式 } Crimson Editor } Crimson Editor

File } New

File } Save As } index.htm

3 插入所需的**<html>**、**<head>**和**<body>**結構標記，如下面的示範網頁所示。並**包含**相應的關閉標記

### 示範HTML網頁語法架構

```
<html>
  <head>
    <title>My First Web Page</title>
  </head>
  <body>
    <h1>My First Web Page</h1>
    <p>Compared to assembly, HTML is <b>really</b> easy!</p>
  </body>
</html>
```

4 在**<head>**標記中， 插入一個**<title>**標記，括起您的公司名。在**<body>**標記中插入一個**<h1>**標記，括起您的姓名。還應確保**包含**相應的關閉標記。

5 在**<h1>**標記下插入一個**<p>**標記，括起兩句關於您的簡短描述。

這在您的頁面上將顯示爲一段文本。

6 在**<p>**標記下插入另一個**<p>**標記。**列出**您名片上的資訊，例如姓名、頭銜、地址和電話號碼。

在HTML中，**<head>**段中的**<title>**標記顯示在瀏覽器標題欄中，而不是頁面內容。**<h1>**標記將在頁面中以大號粗體顯示文本。<p>標記將段落之間用空行隔開。

<br />標記強制換行，不會額外添加空行。<br />標記是自結束標記，不括起任何內容，僅指示視覺顯示元素。其他自結束標記示範包括圖像標記<img … />和表單輸入域<input … />。

7 插入公司網站（或自選網站）的鏈結，作爲“名片”資訊的末行。應使用<a>標記。在此標記中插入一個href（超文本引用）參數，用於指示鏈結目標。<a>和</a>標記之間括起想要顯示的文本（帶有藍色下劃線）。類似於：

```
<a href="http://www.microchip.com">www.microchip.com</a>
```

此鏈結應爲完全限定的URL，包含http://，這是因爲該鏈結要引用外部伺服器。對於本地鏈結，您只需列出相對於本頁面的路徑，例如：

```
<a href="page2.htm">Next Page</a>
```

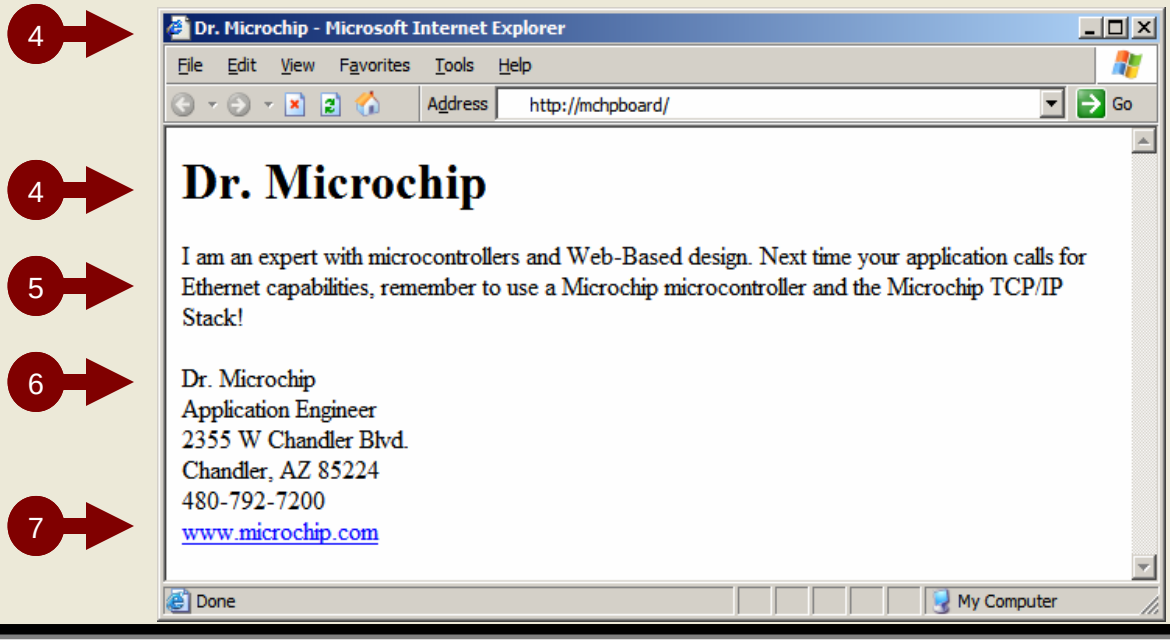
8 保存此檔，然後在您的網頁瀏覽器中打開完成的網頁，進行查看。

**Windows桌面 } 我的電腦 } 導航到WebPages2 } 雙擊 index.htm**

在本步驟中查看了網頁之後，您可以返回到第3步至第8步進行更改。保存之後，使用瀏覽器的 **Refresh**（刷新）按鈕查看所作的修改。

注意，由於此檔是用Crimson Editor保存的，因此其容量小於500位元組。請切記，絕不要使用可產生較大檔案（嵌入式存儲陣列的存儲能力難以滿足要求）的編輯器。

示範網頁



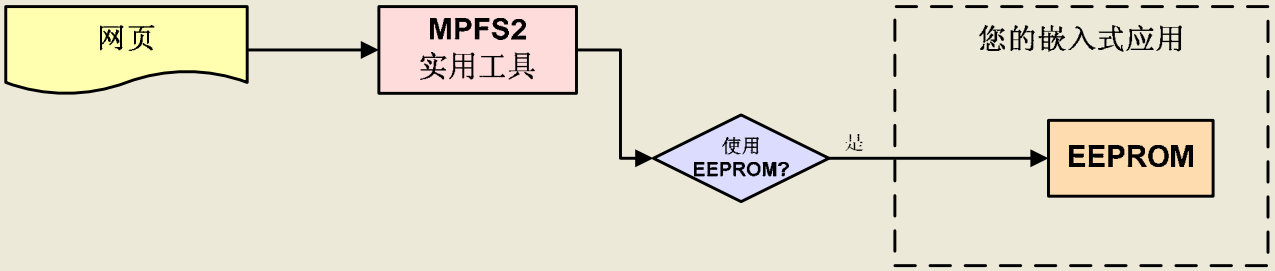

教室可能沒有接入Internet，因此添加到網頁中的鏈結可能無法正常訪問。

在接下來的幾步中，將使用MPFS2應用程式對您的網頁進行打包，並上傳到開發板。在此之前，應確保開發板仍在運行並已連接到網路。

9 打開 **MPFS2** 應用程式：

開始 } 所有程式 } Microchip } COM4201 } MPFS2

## 如何打包網頁




**原始檔案目錄:**
**C:\RTC\COM4201\Microchip Solutions\TCPIP Demo App\WebPages2**

10 確認選中了**Start With: Webpage Directory**（開始：網頁目錄），並正確輸入了上述原始檔案目錄。

11 確認選中了**Output: BIN Image**（輸出：BIN圖像）。

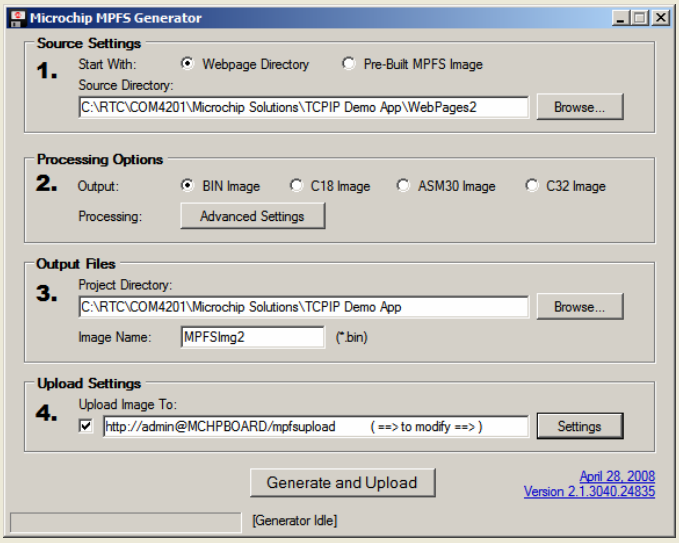
**專案目錄：**
**C:\RTC\COM4201\Microchip Solutions\TCPIP Demo App**

12 確認輸入了上述專案目錄。

13 確認選中了**Upload Image**（上傳圖像）。然後點選**Settings**（設置）按鈕，並進行以下更改和確認：

- 點選**Defaults**按鈕以恢復標準設置。
- 設置Device Address（元件位址）爲在實驗1中選擇的主機名。
- 點選**OK**（確定）保存更改。

設定後的MPFS2應用程式



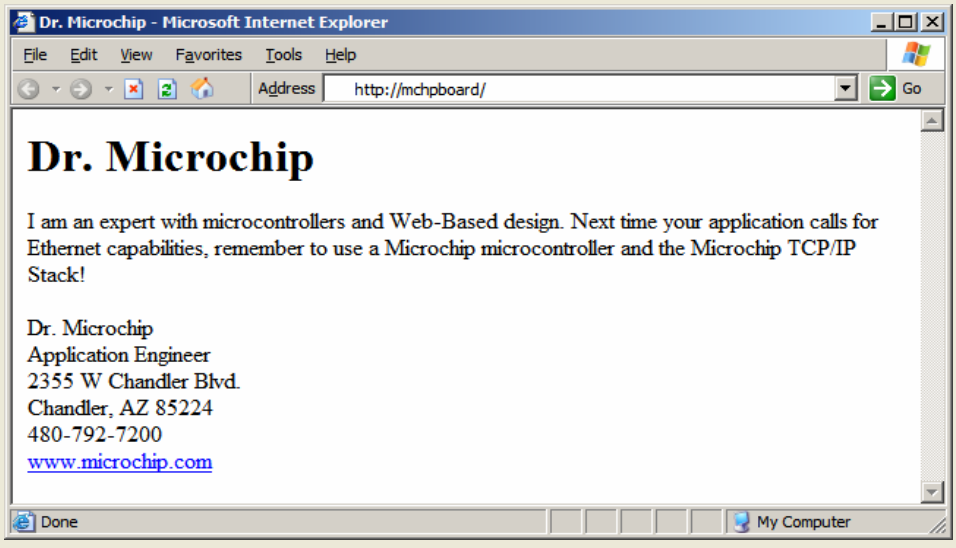

14 點選 **Generate and Upload**（產生並上傳）按鈕。
MPFS2應用程式將對網頁進行打包並上傳到開發板。

15 驗證現在是否可以使用實驗1中的位址訪問開發板上的新網頁。
確保斷開了除錯器，並確認路由器已爲開發板分配了IP地址。

16 在您的瀏覽器位址欄中通過輸入相鄰同學的主機名來訪問他們的開發板。
確認他們也可從他們的瀏覽器中訪問您的網頁。

# 結果

示範網頁



# 附加步驟

17 在**Crimson Editor**中返回到**index.htm**。 在**<head>**段中插入一個指向要建立的樣式表的鏈結：

```
<link href="style.css" rel="stylesheet" type="text/css" />
```

18 插入一個**class**參數到包含名片資訊的**<p>**開始標記中。（關閉標記不需要修改。）此時開始標記應類似於：

```
<p class="businesscard">
```

19 保存**最新修改的index.htm**。

**File } Save**

20 建立一個名爲**style.css**的新檔，將其與HTML頁面一起保存到**WebPages2**檔夾中。

**File } New File } Save As } style.css**

21 插入樣式段到此檔中，如下面“程式分析”部分所示。保存更改，然後**重複**第14步和第15步以驗證所作的更改是否可見。

顏色表示爲一個以十六進位值表示RGB三要素的組合，可隨意修改。與某一標記名相匹配的段將應用於此類型的所有標記。而帶句點分隔符號號後跟另一標記名的段將只應用於類型與第二個名稱相匹配的標記。

# 程式分析

## index.htm

```
1  <html>
2    <head>
3      <title>Dr. Microchip</title>
4      <link href="style.css" rel="stylesheet" type="text/css" />
5    </head>
6    <body>
7      <h1>Dr. Microchip</h1>
8      <p>I am an expert with microcontrollers and Web-Based design.  Next
9        time your application calls for Ethernet capabilities, remember
10       to use a Microchip microcontroller and the Microchip TCP/IP Stack!</p>
11     <p class="businesscard">
12       Dr. Microchip<br />
13       Application Engineer<br />
14       2355 W Chandler Blvd.<br />
15       Chandler, AZ 85224<br />
16       480-792-7200<br />
17       <a href="http://www.microchip.com">www.microchip.com</a></p>
18   </body>
19 </html>
```

HTML文檔遵循必需的語法架構，即使用<html>標記括起<head>和<body>標記。添加<title>標記是爲了在瀏覽器標題欄（以及搜索引擎、書簽和其他位置）中顯示頁面標題。

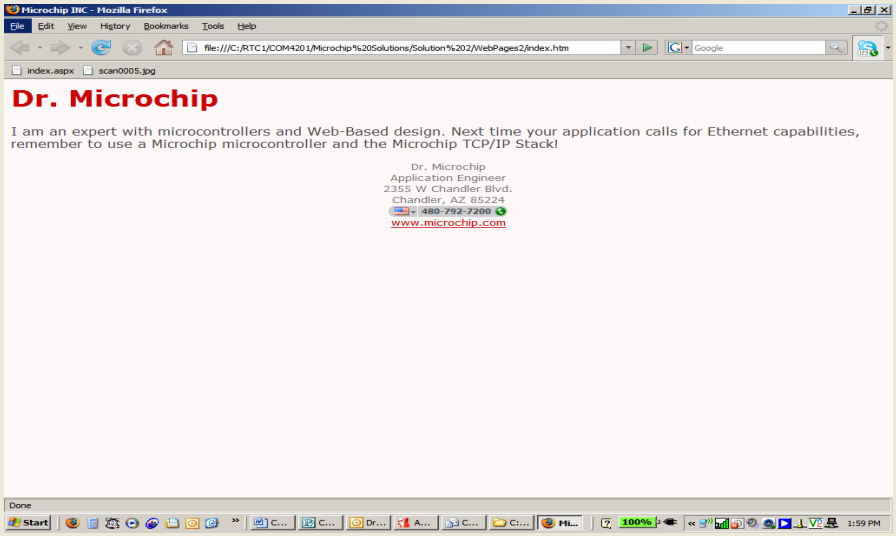
在主體部分，插入了一個標題，隨後爲兩段文字。第二段使用了換行標記以強制開始一個新行。每個標記（自關閉標記除外）都有一個相應的關閉標記來結束每段。

只有在您完成了此附加部分時，<link>標記才會出現在HTML中，第二個<p>標記中的class屬性也是如此。如果您確實完成了此附加部分，您也就創建了一個類似於下圖的style.css文件：

## style.css

```
1  body {
2    background: #fff8f8;
3    font-family: Verdana, Arial, sans-serif;
4    color: #444444;
5  }
6
7  a { color: #cc0000; }
8
9  h1 { color: #cc0000; }
10
11 p.businesscard {
12   color: #777777;
13   text-align: center;
14   font-size: 0.8em;
15 }
```

最終網頁



# 結論

本實驗對HTML和CSS進行了極其簡短的概述。這兩種語言構成了當今網頁開發的基礎。在開始您自己的設計之前，您可能想瞭解HTML的其他功能，包括<form>標記、<img>、 、以及行內格式化標記<b>和<i>。

HTML和CSS的相關文檔均可在由萬維網聯盟在其網站www.w3c.org發佈的推薦書中找到。對於那些喜歡更深入研讀技術規範的同學，快速瀏覽您所喜愛的搜索引擎，將找到數以百計的教授HTML設計基礎的網站。

對於本課程的其餘部分，您僅需要能夠閱讀並理解一些HTML文檔的結構即可。我們將提供用於本課程的網頁，因此您無需從頭開始構建這些網頁。本練習的目的僅是讓您熟悉一下HTML，並學到一些足供以後閱讀程式的知識。

# 實驗 3

## 內建應用程式與協定層

## 目的

對於多數設計，您需要內建現有應用程式與協定層。 對於新的設計，理解協議層的基本設計原理和放置應用程式碼的位置，仍然很重要。

在本實驗中，您將採用前面提到的自動販賣機應用程式實例，並將其與協定層相內建。由於協定層的複雜性，將應用程式內建到其中與將其內建到應用程式中相比通常較爲簡單。本實驗將指導您完成這些步驟，完成後，自動販賣機應用程式和協定層便可同時運行了。（此時它們不能共用資料；後面將添加該功能。）

## 要求

| | |
|---|---|
| **軟體：** | **完成的實驗1** |
| **環境：** | MPLAB® IDE 8.01、COM4201 Lab Build Installer 1.01和Crimson Editor 3.70 |
| **C編譯器：** | MPLAB C18 v3.16或更高版本/C30 v3.01或更高版本/C32 v1.02或更高版本 |
| **硬體工具：** | PICDEM.net™ 2或帶乙太網PICtail™ Plus及PIC24FJ128GA010或<br>PIC32MX360F512L PIM的Explorer 16開發板<br>MPLAB Real ICE™（使用PIC32時需要）或MPLAB ICD 2 |

## 步驟

C:\RTC\COM4201\Microchip Solutions\TCPIP Demo App*.mcp

1. 打開**MPLAB**，並載入適合您開發板設置的專案。 將在本課程的其餘部分使用此專案。

   Project } Open

   **TCPIP Demo App-C18.mcp**用於PICDEM.net 2設置
   **TCPIP Demo App-C30.mcp**用於PIC24FJ128GA010 PIM 的Explorer 16
   **TCPIP Demo App-C32.mcp**用於PIC32MX360F512L PIM的Explorer 16

自動販賣機目錄和原始程式檔：

專案目錄：

2 將文件 VendingMachine.c 和 VendingMachine.h 從 Vending Machine 目錄複製到專案目錄中。可使用**Windows檔案管理員**或**我的電腦**。

3 使用**MPLAB**打開下列指定文件。**不要將這些檔添加到專案中**。將在以下步驟中根據需要複製相應部分。

File } Open } **VendingMachine.h、VendingMachine.c和MainDemo.c**

4 將“Required Headers”部分從VendingMachine.h中刪除。這些頭檔已包含在協定層應用程式的其他部分，此處不應重複。**保存**所作更改。File } Save

**VendingMachine.h**

```
50 #ifndef __VENDINGMACHINE_H
51 #define __VENDINGMACHINE_H
52
53
54 /*****************************************************************
55   Section:
56     Required Headers
57     REMOVE THIS SECTION
58   *****************************************************************
59 //#include "Compiler.h"
60 //#include "GenericTypeDefs.h"
61 //#include "HardwareProfile.h"
62 //#include "Delay.h"
63 //#include "LCDBlocking.h"
64
65
```

5 將 **#Include** VendingMachine.h 插入到 MainDemo.c 中，以便獲取對自動販賣機應用程式資料的存取。將此句直接放在用於包含TCPIP Stack/TCPIP.h的虛擬指令下。

**MainDemo.c**

```
83  */
84 #define THIS_IS_STACK_APPLICATION
85
86 // Include all headers for any enabled TCPIP Stack functions
87 #include "TCPIP Stack/TCPIP.h"
88 #include "VendingMachine.h"
```

6 將以下部分從VendingMachine.c 複製到 MainDemo.c：（見結果列表）

1. 將包括 smVend 列舉(enum)在內的 “Vending Machine Application Global Variables” （自動販賣機應用程式總體變數）部分直接插入到 #include “VendingMachine.h”虛擬指令（已在第5步中添加）下。
2. 將包括WriteLCDMenu()和WritePriceLCD()函數在內的 “Vending Machine LCD Functions”(自動販賣機LCD函數)部分插入到檔案尾端。
3. 將包括InitializeVend()函數在內的 “Vending Machine Initialization”(自動販賣機初始化)部分插入到檔案尾端。

VendingMachine.c

6

```
  Section:
    Vending Machine Application Global Variables
    COPY THIS SECTION
  ************************************************************************

VEND_ITEM    Products[MAX_PRODUCTS];      // All items in the machin
BYTE         machineDesc[33];             // Machine description str
BYTE         curItem;                     // Current product being d
BYTE         curCredit;                   // Current deposit credit,

static enum
{
    SM_IDLE = 0u,                // No events are pending
    SM_DEBOUNCE_DOWN,            // Button press detected...verify
    SM_ADD_COIN,                 // A "coin" was "deposited"
    SM_TRY_VEND,                 // Vend was requested
    SM_PREV,                     // Go back one product
    SM_NEXT,                     // Go forward one product
    SM_DISPLAY_WAIT,             // Waiting for a display message t
    SM_SHOW_MENU,                // Restore the menu to the screen
    SM_RELEASE_WAIT,             // Waiting for button to be releas
    SM_DEBOUNCE_UP               // Verify release of button
} smVend = SM_DISPLAY_WAIT;      // Application state machine
```

6

```
  Section:
    Vending Machine LCD Functions
    COPY THIS SECTION
  ************************************************************************
void WriteLCDMenu(void)
{// Update the LCD screen

    // Blank the LCD display
    LCDErase();

    // Show the name
    strcpy((char*)LCDText, (char*)Products[curItem].name);
    LCDText[strlen((char*)Products[curItem].name)] = ' ';

    // Show the price, or sold out status
```

```
                break;
            case 1:
                LCDText[position++] = '2';
                LCDText[position]   = '5';
                break;
            case 2:
                LCDText[position++] = '5';
                LCDText[position]   = '0';
                break;
            case 3:
                LCDText[position++] = '7';
                LCDText[position]   = '5';
                break;
    }
}
```

## VendingMachine.c

6

```
305 /*****************************************************************************
306   Section:
307     Vending Machine Initialization
308     COPY THIS SECTION
309   *****************************************************************************/
310 void InitializeVend(void)
311 {
312     // Vending machine specific defaults
313     strcpypgm2ram((char*)Products[0].name,  (ROM char*)"Cola");
314     strcpypgm2ram((char*)Products[1].name,  (ROM char*)"Diet Cola");
315     strcpypgm2ram((char*)Products[2].name,  (ROM char*)"Root Beer");
316     strcpypgm2ram((char*)Products[3].name,  (ROM char*)"Orange");
317     strcpypgm2ram((char*)Products[4].name,  (ROM char*)"Lemonade");
318     strcpypgm2ram((char*)Products[5].name,  (ROM char*)"Iced Tea");
319     strcpypgm2ram((char*)Products[6].name,  (ROM char*)"Water");
320     Products[0].price = 4;
321     Products[1].price = 4;
322     Products[2].price = 4;
323     Products[3].price = 4;
324     Products[4].price = 5;
325     Products[5].price = 7;
326     Products[6].price = 8;
327     Products[0].stock = 15;
328     Products[1].stock = 9;
329     Products[2].stock = 22;
330     Products[3].stock = 18;
331     Products[4].stock = 4;
332     Products[5].stock = 29;
333     Products[6].stock = 14;
334
335     strcpypgm2ram((char*)machineDesc,  (ROM char*)"Building C4 - 2nd F
336     machineDesc[32] = '\0';
337     curItem = 0;
338     curCredit = 0;
339 }
```

應用程式全局變數部分聲明了存儲狀態並定義了自動販賣機應用程式所使用的狀態機。LCD函數部分編寫了各種用於LCD顯示的功能表。最後，初始化部分設定了產品名、價格和設備啓動時的庫存。

## MainDemo.c- #include 和全局變數

6

```
 87 #include "TCPIP Stack/TCPIP.h"
 88 #include "VendingMachine.h"
 89
 90 /*****************************************************************************
 91   Section:
 92     Vending Machine Application Global Variables
 93     COPY THIS SECTION
 94   *****************************************************************************/
 95
 96 VEND_ITEM    Products[MAX_PRODUCTS];      // All items in the machin
 97 BYTE         machineDesc[33];             // Machine description str
 98 BYTE         curItem;                     // Current product being d
 99 BYTE         curCredit;                   // Current deposit credit,
100
101 static enum
102 {
103     SM_IDLE = 0u,                   // No events are pending
104     SM_DEBOUNCE_DOWN,               // Button press detected...verify
105     SM_ADD_COIN,                    // A "coin" was "deposited"
106     SM_TRY_VEND,                    // Vend was requested
107     SM_PREV,                        // Go back one product
108     SM_NEXT,                        // Go forward one product
109     SM_DISPLAY_WAIT,                // Waiting for a display message t
110     SM_SHOW_MENU,                   // Restore the menu to the screen
111     SM_RELEASE_WAIT,                // Waiting for button to be releas
112     SM_DEBOUNCE_UP                  // Verify release of button
113 } smVend = SM_DISPLAY_WAIT;         // Application state machine
```

## MainDemo.c（文件末尾）

```
854     SPIFlashWriteArray((BYTE*)&AppConfig, sizeof(AppConfig));
855
856     #endif
857 }
858 #endif
859
860 /*****************************************************************************
861   Section:
862     Vending Machine LCD Functions
863     COPY THIS SECTION
864   *****************************************************************************/
865 void WriteLCDMenu(void)
866 {// Update the LCD screen
867
868     // Blank the LCD display
869     LCDErase();
870
871     // Show the name
872     strcpy((char*)LCDText, (char*)Products[curItem].name);
873     LCDText[strlen((char*)Products[curItem].name)] = ' ';
```

6

```
319     strcpypgm2ram((char*)Products[6].name, (ROM char*)"Water");
320     Products[0].price = 4;
321     Products[1].price = 4;
322     Products[2].price = 4;
323     Products[3].price = 4;
324     Products[4].price = 5;
325     Products[5].price = 7;
326     Products[6].price = 8;
327     Products[0].stock = 15;
328     Products[1].stock = 9;
329     Products[2].stock = 22;
330     Products[3].stock = 18;
331     Products[4].stock = 4;
332     Products[5].stock = 29;
333     Products[6].stock = 14;
334
335     strcpypgm2ram((char*)machineDesc, (ROM char*)"Building C4 - 2nd Floor NW");
336     machineDesc[32] = '\0';
337     curItem = 0;
338     curCredit = 0;
339 }
```

6

7 在main()函數中插入對InitializeVend()的呼叫。此函數將在自動販賣機啓動過程中對其進行設定，且應在叫用InitAppConfig()之後立即對其進行呼叫。

## MainDemo.c-初始化部分

```
222     MPFSInit();
223     #endif
224
225     // Initialize Stack and application related NV variables into
226     InitAppConfig();
227     InitializeVend();
228
229     // Initiates board setup process if button is depressed
```

7

8 將狀態機從VendingMachine.c的main()中複製到MainDemo.c文件的ProcessIO()段。此函數用於實現自動販賣機執行操作的狀態機。應複製while無限迴圈的內容，而不是此迴圈本身。協定層將此函數作爲協同多工處理迴圈的一部分來重複呼叫。應使用此程式替換ProcessIO()的現有內容，因爲已不再需要演示程式的A/D轉換功能。

9 在ProcessIO()的頂部插入一段名爲displayTimeout的對static WORD值的**聲明**，並將其值**初始化**爲2。上一步中複製的狀態機根據此值在顯示狀態資訊時跟蹤時間。

## MainDemo.c-ProcessIO() 程式段

```
403 // Processes A/D data from the potentiometer
404 static void ProcessIO(void)
405 {
406     static WORD displayTimeout = 2;
407
408     // Main state machine
409     switch(smVend)
410     {
411         case SM_IDLE:
412             // Wait for a button press
413             if(BUTTON0_IO == 0 || BUTTON1_IO == 0 || BUTTON2_IO ==
414                 smVend = SM_DEBOUNCE_DOWN;
415             DelayMs(1);
416             break;
417
418         case SM_DEBOUNCE_DOWN:
```

```
506         case SM_RELEASE_WAIT:
507             // Wait for all buttons to be released
508             if(BUTTON0_IO == 1 && BUTTON1_IO == 1 && BUTTON2_IO ==
509                 smVend = SM_DEBOUNCE_UP;
510             DelayMs(1);
511             break;
512
513         case SM_DEBOUNCE_UP:
514             // Make sure all buttons were released
515             DelayMs(1);
516             if(BUTTON0_IO == 1 && BUTTON1_IO == 1 && BUTTON2_IO ==
517                 smVend = SM_IDLE;
518             else
519                 smVend = SM_RELEASE_WAIT;
520             break;
521     }
522 }
```

10 編譯專案的新程式，並將其燒錄到開發板中。 斷打開除錯器，並確認路由器爲開發板分配了新的IP位址。

11 驗證自動販賣機是否按最初演示中討論的那樣工作。LCD顯示幕上無法預料的輸出將在下一步中進行校正。

驗證在實驗2中訪問的網頁是否仍然可用。

12 註解掉主程序迴圈中對`PingDemo()`的呼叫以禁止此輸出。

當按下最右側的按鈕時，ping demo 將產生“Ping timed out”（Ping 超時）消息。

註解掉對`DisplayIPValue()`的呼叫，因爲它會將IP位址寫入LCD。

這些刪除的部分將會**取消**將外部資料輸出到LCD的操作。

**MainDemo.c**

```
328         #if defined(STACK_USE_ICMP_CLIENT)
329         //PingDemo();
330         #endif
331
332         #if defined(STACK_USE_SNMP_SERVER) && !defined(SNMP_TRAP_DISABLED)
333         SNMPTrapDemo();
334         #endif
335
336         ProcessIO();
337
338         // If the DHCP lease has changed recently, write the new
339         // IP address to the LCD display, UART, and Announce service
340         if(DHCPBindCount != myDHCPBindCount)
341         {
342             myDHCPBindCount = DHCPBindCount;
343
344             #if defined(STACK_USE_UART)
345                 putrsUART((ROM char*)"\r\nNew IP Address: ");
346             #endif
347
348             //DisplayIPValue(AppConfig.MyIPAddr);
```

12

12

13 編譯專案的新程式、選擇編程器，並將程式燒錄到開發板中。
斷開除錯器，並確認路由器爲開發板分配了新的IP位址。
驗證不期望的消息是否已不再出現在LCD螢幕上。


## 結果

如果自動販賣機仍在正常工作，且LCD資料是正確的，則表明已成功內建了這兩個應用程式。在這兩個函數尚未進行交互時， 執行基本功能。

## 結論

雖然並未增加新功能，但現在自動販賣機已非常接近於可通過網路操作的設備。與協議層的內建是最有挑戰性的任務，這是因爲此操作需要（多數情況下）從一個新的角度來考慮應用程式。仍然存在一些內建問題，這將在下一實驗中進行研究。

## 程式分析

自動販賣機應用程式作爲狀態機來實現，這對於許多基於單片機的簡單系統很常見。由於每種狀態執行時間極短，因此將應用程式內建到協定層相對簡單一些。但是，對於較複雜的應用，則需要考慮通用模型和程式放置。

通常，應在呼叫InitAppConfig()之後再呼叫初始化程式。此程式將在設備啓動時執行一次。應用程式碼應放在主協定層迴圈末或`ProcessIO()`函數中。

# 實驗4

## 刪除阻斷程式

## 目的

網路應用程式必須能處理大量的背景程式，以保持網路連接處於活動狀態。要完成這些對時間要求嚴格的事件，Microchip TCP/IP協議層應以協同多工方式工作，無需使用RTOS和中斷，進而減少了開銷。但這意味著，您的應用程式中不能有任何可能會阻斷處理器進程的程式。

在本實驗中，您將看到阻斷迴圈對TCP/IP協議層造成的影響，還將學習如何採取措施以避免應用程式中的此類迴圈。

## 要求

| | |
|---|---|
| **軟體：** | **完成的實驗3或從解決方案3入手** |
| **環境：** | MPLAB® IDE 8.01、COM4201 Lab Build Installer 1.01和Crimson Editor 3.70 |
| **C編譯器：** | MPLAB C18 v3.16或更高版本/C30 v3.01或更高版本/C32 v1.02或更高版本 |
| **硬體工具：** | PICDEM.net™ 2或帶乙太網PICtail™ Plus及PIC24FJ128GA010或<br>PIC32MX360F512L PIM的Explorer 16開發板<br>MPLAB Real ICE™（使用PIC32時需要）或MPLAB ICD 2 |

## 步驟

1. 根據下面的描述，確認閃爍LED的工作狀態。

   通過運行實驗3中的應用程式，注意開發板上LED的閃爍狀態。當有消息顯示在LCD上時，此LED燈將發生什麼情況?（在您將超過5.00美元的現金插入到自動販賣機時，經常會看到這樣的情況。）

   LED 穩定閃爍表示正在頻繁呼叫TCP/IP協議層 。當此狀態燈不閃爍時，表示協議層未運行，且無法回應進入的網路通信。現有自動販賣機程式使用阻斷迴圈來對顯示消息進行計時。需要校正此程式。

2. 對於前面所選的**PIC18、PIC24或PIC32**，在**MPLAB**中找到MainDemo.c文件中的ProcessIO()。

## 關於定時迴圈

實現延時(Delay)的常見方法是讓處理器完成一些無意義的任務，例如計數到10,000或呼叫其中一個在內部實現此功能的Delay()函數。但是，這種程式結構將降低處理器效能並會消耗MCU資源，而這些MCU資源可用於處理進入的乙太網資料包或其他網路功能。應避免這種類型的迴圈。

轉而使用所提供的Tick模組。此Tick模組是由中斷驅動的，基於硬體時鐘。它穩定而精確，可用於通過比較當前時間計數值與超時值實現非阻斷延時。由於它是基於硬體的，所以也適合用作即時時鐘。

3 找到SM_DISPLAY_WAIT 語句中的阻斷迴圈。此迴圈通過執行無意義的任務消耗處理時間。使用標記將其註解掉。

插入一個TICK比較值以確定是否經過了足夠的時間。如果發生超時，則執行狀態機。否則，僅執行 break，返回到主協定層應用程式。該協議層稍後將再次呼叫自動販賣機執行比較操作。使用如下示範。

**MainDemo.c**

3 →

```
493         case SM_DISPLAY_WAIT:
494             // Wait for the timout to occur before continuing
495             //for(displayTimeout *= 1000; displayTimeout > 0; displayTimeout--)
496                 //DelayMs(1);
497             if ((LONG) TickGet() - displayTimeout) > (LONG)0)
498             smVend = SM_SHOW_MENU;
499             break;
500
501         case SM_SHOW_MENU:
```

4 在ProcessIO()的頂部更改此值的聲明，給此變數分配TICK類型，而不是WORD類型。還應更改其初始值，以指示2秒，而不僅僅是一個數字2。使用宏TICK_SECOND來計算此值。您的程式應類似於下例。

**MainDemo.c**

4 →

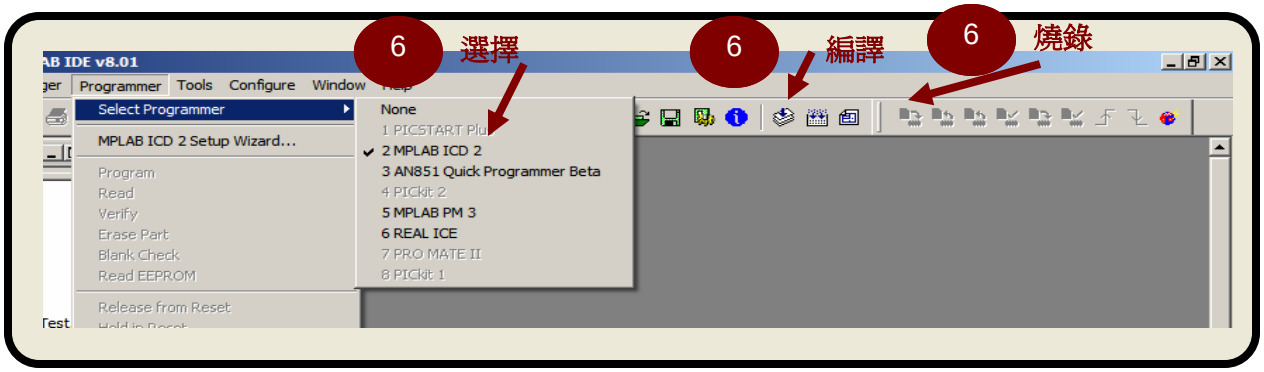
```
403 // Processes A/D data from the potentiometer
404 static void ProcessIO(void)
405 {
406     static TICK displayTimeout = 2*TICK_SECOND;
407
408     // Main state machine
409     switch(smVend)
410     {
```

5 **更改**下面四個實例，其中displayTimeout被設置爲使用一個相對TICK時間，而不是絕對延時。使用一個基於TickGet()和TICK_SECOND的值來替代數字賦值，如下所示。

## MainDemo.c

```
444                 LCDUpdate();
445                 displayTimeout = TickGet() + 2*TICK_SECOND;
446                 smVend = SM_DISPLAY_WAIT;
447             }
448             break;
449
450         case SM_TRY_VEND:
451             // Try to vend a product
452             if(Products[curItem].stock == 0)
453             {// Product is sold out
454                 strcpypgm2ram((char*)LCDText, (ROM char*)"    SOLD OUT                    ");
455                 LCDUpdate();
456                 displayTimeout = TickGet() + 2*TICK_SECOND;
457                 smVend = SM_DISPLAY_WAIT;
458             }
459             else if(Products[curItem].price > curCredit)
460             {
461                 strcpypgm2ram((char*)LCDText, (ROM char*)"Price:          Credit:         ");
462                 WritePriceLCD(Products[curItem].price, 8);
463                 WritePriceLCD(curCredit, 24);
464                 LCDUpdate();
465                 displayTimeout = TickGet() + 2*TICK_SECOND;
466                 smVend = SM_DISPLAY_WAIT;
467             }
468             else
469             {
470                 strcpypgm2ram((char*)LCDText, (ROM char*)"   vending...                ");
471                 curCredit -= Products[curItem].price;
472                 Products[curItem].stock--;
473                 LCDUpdate();
474                 displayTimeout = TickGet() + 1*TICK_SECOND;
475                 smVend = SM_DISPLAY_WAIT;
```

6 編譯專案的新程式、選擇編程器，並將程式燒錄到開發板中。
斷開除錯器，並確認路由器爲開發板分配了新的IP位址。



7 **驗證**自動販賣機是否仍在能正常工作。

1. 確保仍能通過瀏覽器訪問實驗2中的網頁。
2. 驗證即使在有狀態消息顯示在螢幕上的情況下，狀態LED是否仍能繼續閃爍 。
   i. 在以不足的現金嘗試購買時進行測試。
   ii. 在投入的現金超過5.00美元需要找零時進行測試。
   iii. 在售出某個產品時進行測試。
   iv. 在某個產品脫銷時進行測試。

# 結果

本實驗順利完成之後，狀態LED將能在LCD上有消息顯示時不間斷地持續閃爍。這表明應用程式沒有阻斷處理器運行，且協議層能及時完成其任務。

# 程式分析

本實驗所提供的示範實現了一個無阻斷迴圈的計時器。在超時階段內使用單一的狀態，而在發生超時後設置各種機器狀態 。

超時時間存儲在`displayTimeout`中，聲明爲`TICK`值。此超時值通過將當前時間（通過`TickGet()`獲得）與`TICK_SECOND`的若干倍相加計算得到。這種方法適用於測量從幾微秒到幾小時的時間。（其配套方法`TickGetDiv256()`和`TickGetDiv64K()`分別用於獲得幾周或幾年的時間。）`TICK_SECOND`的值是通過`GetInstructionClock()`值計算得到的，計算過程在`HardwareProfile.h`中指定。

`TICK`值存儲爲32位元無符號整數。當加上一個時間增量（如`2*TICK_SECOND`）時，可能會導致溢位。類似的，當執行減法時可能會導致借位。當將`TICK`值強制轉換爲`LONG`時， `TICK`值的減法通過恢復符號位元對借位的情況進行了補償，從而允許在所有情況下進行有效比較。

# 附加步驟

8 程式中仍保留有對`DelayMs()`的幾次呼叫，以提供按鈕彈跳所需時間。由於此時協定層迴圈在每個狀態間執行，試研究是否可在不影響操作的情況下移除這些呼叫。具體取決於您的平臺，因爲較快的處理器，其迴圈速度可能太快而無法避開這些瞬態過程。

# 結論

現在已完全內建了自動販賣機，但兩個任務不能進行交互。由於此操作已經完成，您現在可爲該設備添加網路特性了。

在上面兩個實驗中學到的原理可應用於所有協定層應用程式：

1. 將大的任務分爲多個狀態，允許協定層頻繁執行。
2. 使用Tick模組而非阻斷迴圈爲事件定時。

協議層的協同多工方法很靈活，且消耗的處理器時間與應用程式所提供的時間相同。如果需要，通過應用程式可將協定層操作中斷幾百毫秒甚至幾秒。在不需要時（不應經常發生），多數網路應用協定將不會超時，除非有幾秒未進行通訊。但此時，應用程式將會有其接收緩衝區溢出和遺失資料的危險。一般來說，應用程式應嘗試以至少10或20 ms的間隔運行主協議層迴圈，若間隔時間縮短至1-2 ms，則可實現最佳性能。

# 實驗5

## 基於網頁的監視

## 目的

此自動販賣機首先要實現的網路特性是遠端確定其庫存的能力。 已決定基於網頁進行監視，那麼下一步便需要將自動販賣機資料鏈結到網頁。

已爲此項目提供了示範網頁。但僅是靜態網頁，沒有將自動販賣機資料填充到網頁的機制。在本實驗中， 將編寫第一個頁面來顯示自動販賣機報告的產品名和剩餘庫存量。

## 要求

| | |
|---|---|
| **軟體：** | **完成的實驗4或從解決方案4入手** |
| **環境：** | MPLAB® IDE 8.01、COM4201 Lab Build Installer 1.01和Crimson Editor 3.70 |
| **C編譯器：** | MPLAB C18 v3.16或更高版本/C30 v3.01或更高版本/C32 v1.02或更高版本 |
| **硬體工具：** | PICDEM.net™ 2或帶乙太網PICtail™ Plus及PIC24FJ128GA010或 PIC32MX360F512L PIM的Explorer 16開發板<br>MPLAB Real ICE™（使用PIC32時需要）或MPLAB ICD 2 |

**示範網頁（原始檔）：**
**C:\RTC\COM4201\Sample Web Pages**

**網頁目錄（目標）：**
**C:\RTC\COM4201\Microchip Solutions\TCPIP Demo App\WebPages2**

## 步驟

1. 刪除在實驗2中使用的`WebPages2`檔夾中的所有檔案。接下來的實驗將使用預先編寫好的網頁，因此將不再需要這些檔了。
   使用從下面指出的`Sample Web Pages`檔夾中複製的網頁替換這些檔。

2. 在**Crimson Editor**中，打開`WebPages2`目錄下的`index.htm`。

   開始 } 所有程式 } Crimson Editor } Crimson Editor

3 使用名爲**~hostname~**的動態變數替換 **HOSTNAME** 靜態文本。

將 **Machine Location / Description**文本替換爲**~machineDesc~**。

**保存**對HTML程式所作的更改。

**index.htm**

```
3  <b>Status</b>  | 
4  <a href="lights.htm">Lights</a>  | 
5  <a href="products.htm">Products</a>
6  </p>
7
8  <div id="location">Machine ~hostname~ :: ~machineDesc~ </div>
9
10 <div id="bargraph">
11
12     <div class="productname">~name(0)~</div>
```

3

4 使用MPFS應用程式將修改後的頁面打包並**上傳**到開發板。

前面講過，使用**MPFS2應用程式**產生`HTTPPrint.h`，此處此檔必須先編譯到MPLAB專案中，然後才能與第5步中的動態變數回調函數編譯到一起。

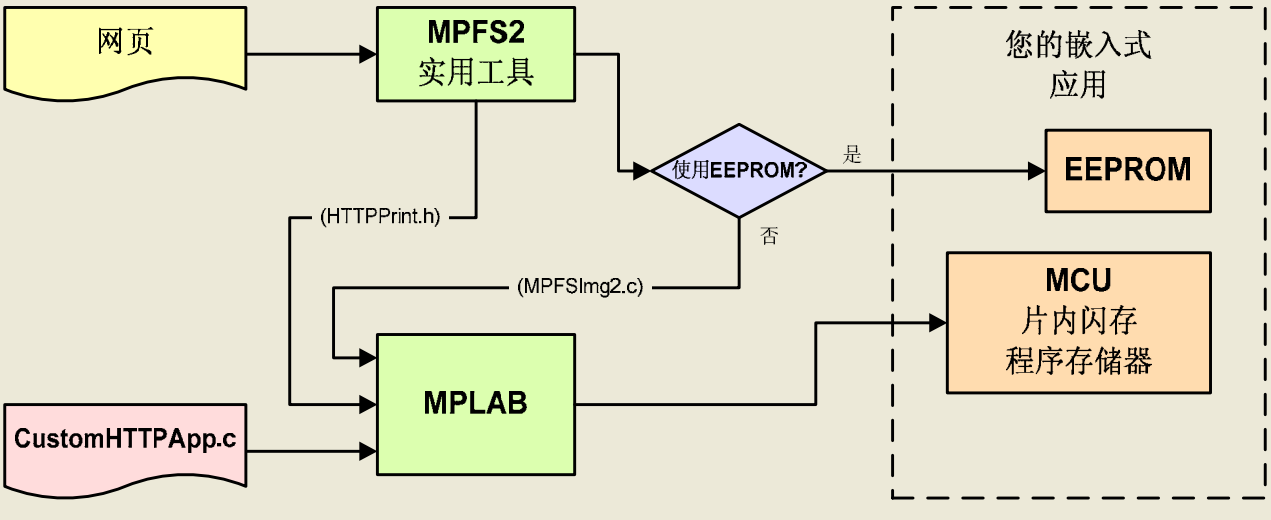
開始 } 所有程式 } Microchip } COM4201 } MPFS2

無需對此應用程式設定進行任何修改。如果上傳失敗，請驗證此設置是否與實驗2第9步至第14步的設定相匹配。

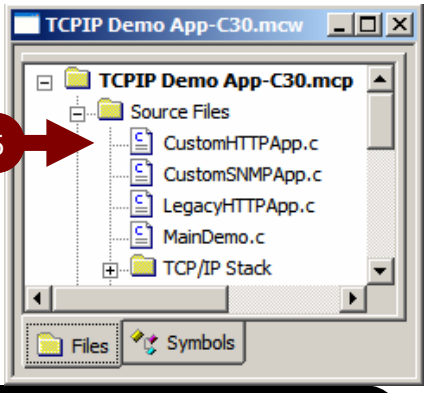
點選**Generate and Upload**將新頁面上傳到開發板。

## 如何打包網頁

前面講過：必須先執行MPFS2應用程式，然後才能在MPLAB中編譯項目。



5 在MPLAB專案視窗中打開CustomHTTPApp.c。




6 刪除“Dynamic Variable Callback Functions”（動態變數回調函數）部分中除HTTPPrint_version()和HTTPPrint_builddate()以外的所有HTTPPrint_*函數（這些回調函數用於演示應用程式，其中的大多數已不再需要。）。

## CustomHTTPApp.c（刪除未使用部分）

```
1331
1332 void HTTPPrint_version(void)
1333 {
1334     TCPPutROMString(sktHTTP, (ROM void*)VERSION);
1335 }
1336
1337
1338
1339
1340 ROM BYTE HTML_UP_ARROW[] = "up";
1341 ROM BYTE HTML_DOWN_ARROW[] = "dn";
1342
6 1343 void HTTPPrint_btn(WORD num)
1344 {
1345     // Determine which button
1346     switch(num)
1347     {
1348         case 0:
1349             num = BUTTON0_IO;
```

```
1766 void HTTPPrint_status_ok(void)
1767 {
1768     if(lastSuccess)
1769         TCPPutROMString(sktHTTP, (ROM BYTE*)"block");
1770     else
1771         TCPPutROMString(sktHTTP, (ROM BYTE*)"none");
1772     lastSuccess = FALSE;
1773 }
1774
1775 void HTTPPrint_status_fail(void)
1776 {
1777     if(lastFailure)
1778         TCPPutROMString(sktHTTP, (ROM BYTE*)"block");
1779     else
1780         TCPPutROMString(sktHTTP, (ROM BYTE*)"none");
1781     lastFailure = FALSE;
6 1782 }
1783
1784 #endif
```

7 在檔案的這一部分插入一個名爲HTTPPrint_hostname()的回調函數。此函數無任何參數且無返回值。在此函數中，使用TCPPutString()將RAM單元中的主機名AppConfig.NetBIOSName輸出到TCP套接字sktHTTP。更多關於回調的資訊，請參見下面標記有相關主題的資訊框。

## CustomHTTPApp.c

```
1326    if(TCPIsPutReady(sktHTTP) < strlenpgm((ROM char*)__DATE__" "_
1327        return;
1328
1329    curHTTP.callbackPos = 0x00;
1330    TCPPutROMString(sktHTTP, (ROM void*)__DATE__" "__TIME__);
1331}
1332
1333void HTTPPrint_version(void)
1334{
1335    TCPPutROMString(sktHTTP, (ROM void*)VERSION);
1336}
1337
1338void HTTPPrint_hostname(void)
1339{
1340    TCPPutString(sktHTTP, AppConfig.NetBIOSName);
1341}
```

7

## 關於回調函數

回調函數是一個事件驅動編程的概念。對於HTTP2網路服務器，當在HTML程式中遇到動態變數時將發生某一事件。由回調函數處理的回應，將文本或資料輸出到網路瀏覽器中。

無需擔心如何或何時呼叫此回調函數。HTTP2伺服器與由MPFS2應用程式生成的HTTPPrint.h檔共同管理此操作。

8 在檔的這一部分添加第二個名爲HTTPPrint_machineDesc()的回調函數 。此函數也沒有參數且無返回值。 此函數要稍微複雜一些。

首先，在此檔的頂部插入一個#include VendingMachine.h條目。此條目提供了對machineDesc字串的引用。

其次，插入一個if迴圈，此迴圈使用TCPIsPutReady()來確定緩衝空間大小。machineDesc字串最長可爲32個位元組，這將大於HTTP2網路服務器保證可用的16位元組。因此，HTTPPrint呼叫必須遵循確保TCP緩衝區中至少有32個空閒位元組的原則。如果緩衝區過小，則在curHTTP.callbackPos中設置一個非零標誌並返回，在有更多空間時，將請求再次呼叫。

## CustomHTTPApp.c

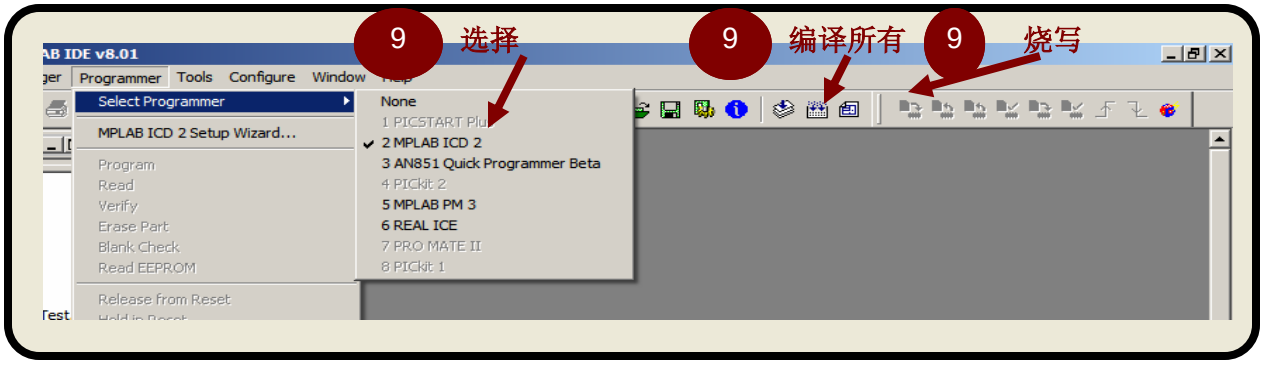
8

```
1335        TCPPutROMString(sktHTTP, (ROM void*)VERSION);
1336    }
1337
1338    void HTTPPrint_hostname(void)
1339    {
1340        TCPPutString(sktHTTP, AppConfig.NetBIOSName);
1341    }
1342
1343    void HTTPPrint_machineDesc(void)
1344    {
1345        if(TCPIsPutReady(sktHTTP) < 32)
1346        {
1347            curHTTP.callbackPos = 0x01;
1348            return;
1349        }
1350
1351        curHTTP.callbackPos = 0x00;
1352        TCPPutString(sktHTTP, machineDesc);
1353    }
```

9 編譯所有此專案的新程式、選擇編程器，並將程式燒錄到開發板中。

由於添加了動態變數，因此使用Build All，而非Make。（對於其他更改，使用較快的Make選項仍然是安全的。）

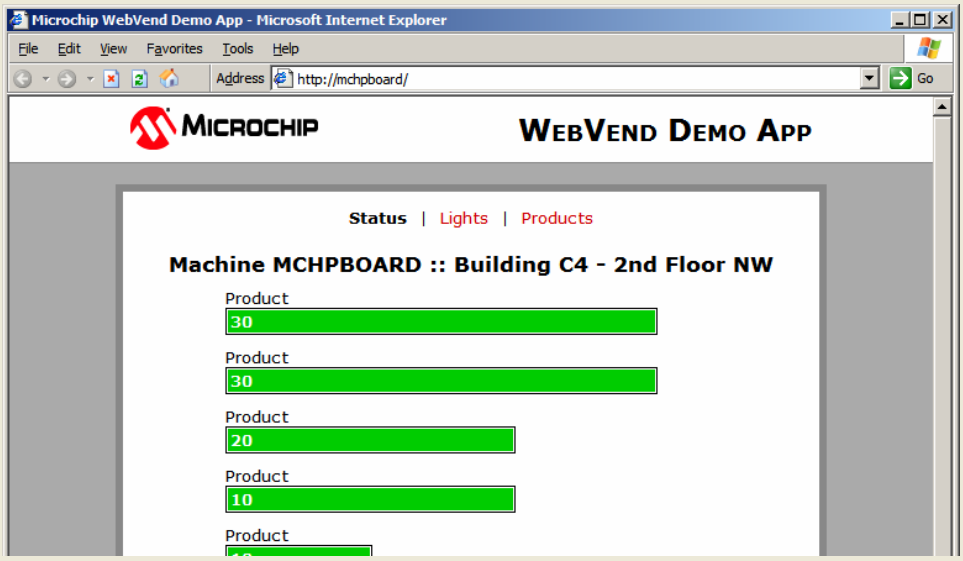
斷開除錯器，並確認路由器爲開發板分配了新的IP位址。




10 驗證自動販賣機是否按最初演示中討論的那樣工作。

驗證此網頁是否已用動態變數替換了靜態值，尤其是主機名和位置。位置字串在MainDemo.c中設置，如果需要可在該檔中進行修改。

示範網頁




網頁目錄（目標）：
C:\RTC\COM4201\Microchip Solutions\TCPIP Demo App\WebPages2

11 返回到**Crimson Editor**中的`index.htm`。

使用具有單一參數的動態變數替換靜態字串“Product” 的每個實例。呼叫變數`~name(i)~`，其中i表示索引0到6。這便爲每個產品名添加了動態變數。

12 使用動態變數替換庫存量的每個靜態實例，命名此變數爲`~stock(i)~`，其索引仍爲0到6。每個條形圖均有兩個位置列示庫存：第一個作爲文字顯示在條形圖內，而第二個用於控制條形圖的顯示長度。確保用動態變數替換了這兩個位置，但保留`em`識別字。（`em`是標準列印尺寸。）

**確保替換了產品和庫存條目所有6個實例。**

**保存**對HTML檔所作的更改。

## index.htm

```
 8 <div id="location">Machine ~hostname~ :: ~machineDesc~ </div>
 9
10 <div id="bargraph">
11
12     <div class="productname">~name(0)~</div>
13     <div class="bar" style="width: ~stock(0)~em">
14       <div class="~status(0)~">~stock(0)~</div>
15     </div>
16
17     <div class="productname">~name(1)~</div>
18     <div class="bar" style="width: ~stock(1)~em">
19       <div class="~status(1)~">~stock(1)~</div>
20     </div>
21
22     <div class="productname">~name(2)~</div>
```

13 類似於第4步執行**MPFS2應用程式**，並點選**Generate and Upload**來燒錄新網頁。

14 在CustomHTTPApp.c中插入另一個回調函數HTTPPrint_name()，用於顯示所請求的產品名。此函數接受一個WORD參數。

產品資料存儲在Products陣列中。如下所示，將WORD參數作爲此陣列的索引，使用TCPPutString輸出相關的name元素。

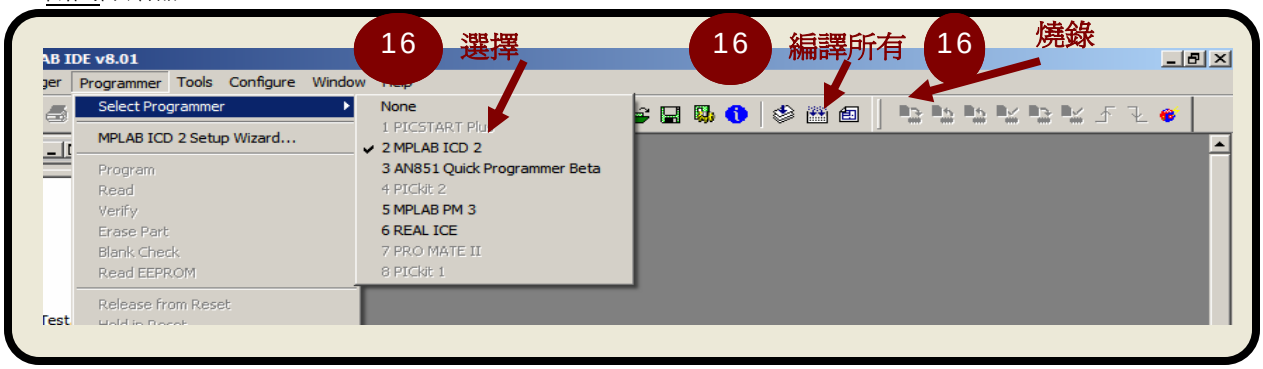
15 插入另一個回調函數HTTPPrint_stock()。此函數也接受一個WORD參數。使用uitoa(WORD, BYTE*)函數將庫存元素從二進位值轉換爲字串，然後將此輸出寫入套接字。

## CustomHTTPApp.c

```
1350
1351     curHTTP.callbackPos = 0x00;
1352     TCPPutString(sktHTTP, machineDesc);
1353 }
1354
1355 void HTTPPrint_name(WORD item)
1356 {
1357     TCPPutString(sktHTTP, Products[item].name);
1358 }
1359
1360 void HTTPPrint_stock(WORD item)
1361 {
1362     BYTE buf[4];
1363
1364     uitoa(Products[item].stock, buf);
1365     TCPPutString(sktHTTP, buf);
```

16 編譯所有此專案的新程式、選擇編程器，並將程式燒錄到開發板中。由於添加了動態變數，因此使用Build All，而非Make。

斷開除錯器。



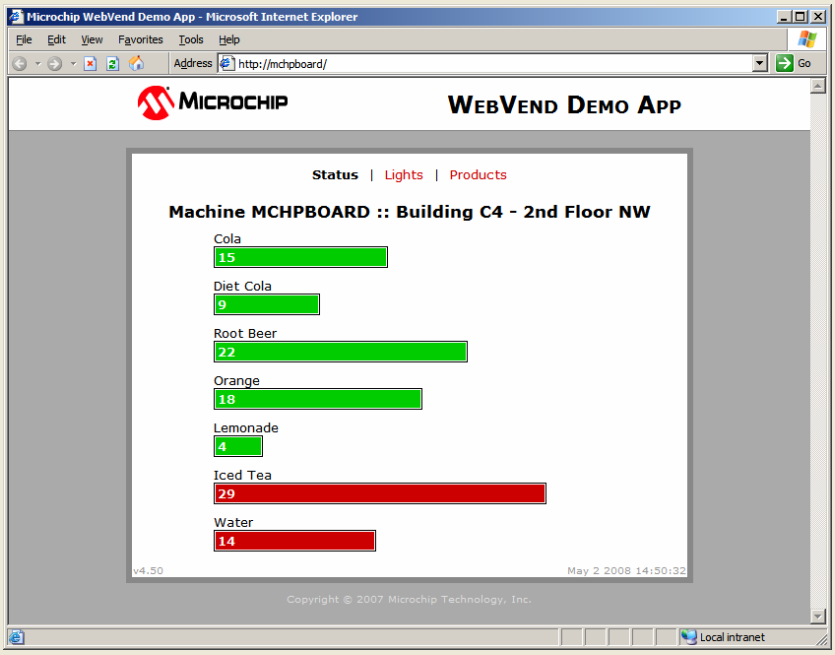

17 使用瀏覽器訪問開發板網頁。

驗證條形圖顯示現在是否已使用來自開發板的動態資料進行了填充。售出一些飲料之後，點選瀏覽器中的**Refresh**按鈕，以驗證庫存數和條形圖長度是否發生變化。

## 結果

完成實驗後，使用瀏覽器查看將出現類似於下圖的螢幕。從此機器中購買產品後，刷新此瀏覽器頁面將指示新的庫存量。自動更新顯示在技術上是可行的，但超出了本課程的範圍。

**完成的示範网页**



# 程式分析

在HTML檔中，插入了兩類動態回調。在本實驗的第一部分，添加了標準文本輸出。這些回調不接收參數，僅需呼叫`TCPPutString()`將文本寫入網頁。

其中一個回調（用於機器描述字串）可能需要輸出多於16個位元組。 要阻止可能的緩衝區溢出，回調必須管理其輸出狀態。由於輸出相對較短，此函數僅檢查是否有足夠的空間可用。如果沒有，它將在`curHTTP.callbackPos`中設置一個標誌後返回，這將指示伺服器必須再次呼叫此回調以完成輸出。這對於少於約50位元組的字串是可接受的解決方案。對於較長的輸出， 可寫入盡可能多的位元組，並使用`callbackPos`變數跟蹤輸出位置，如演講幻燈片中所討論的那樣。完成之後，將`callbackPos`變數恢復到0以指示回調完成。

還爲每種產品的名稱和庫存量添加了動態變數。這些變數會將參數傳遞給函數。所有參數都作爲`WORD`值傳遞，回調函數將此值用作`Products`陣列的索引。這些函數的輸出用於輸出文本及控制顯示元素，如條形圖顯示中的條形長度。

# 結論

在本實驗中，爲自動販賣機添加了基於網頁的監視。實現了幾個動態變數示範，展示了此靈活特性的各種功能。僅通過幾行程式，自動販賣機現在便可通過清晰的條形圖顯示輸出其當前狀態。

# 附加步驟

動態變數也可用於控制視覺元素。注意，底部的兩個條形圖是紅色的，而其他條形圖是綠色的。我們想要實現這樣的顏色控制，即在任何產品的庫存很低時能自動顯示爲紅色。

18 通過Crimson Editor，在index.htm檔中使用稱爲~status(i)~ 的動態變數替換文本的class屬性low和ok。

19 執行**MPFS2應用程式**以打包經過修改的網頁，並將其上傳到開發板。此應用程式還將更新HTTPPrint.h的副本以鏈結新的動態變數。

20 在CustomHTTPApp.c中插入回調函數HTTPPrint_status()。 此回調函數在庫存低於8時應輸出字串 “low” ，而在其他情況下輸出“ok” 。

21 編譯所有此專案的新程式、選擇編程器，並將程式燒錄到開發板中。
由於添加了動態變數，因此使用Build All，而非Make。

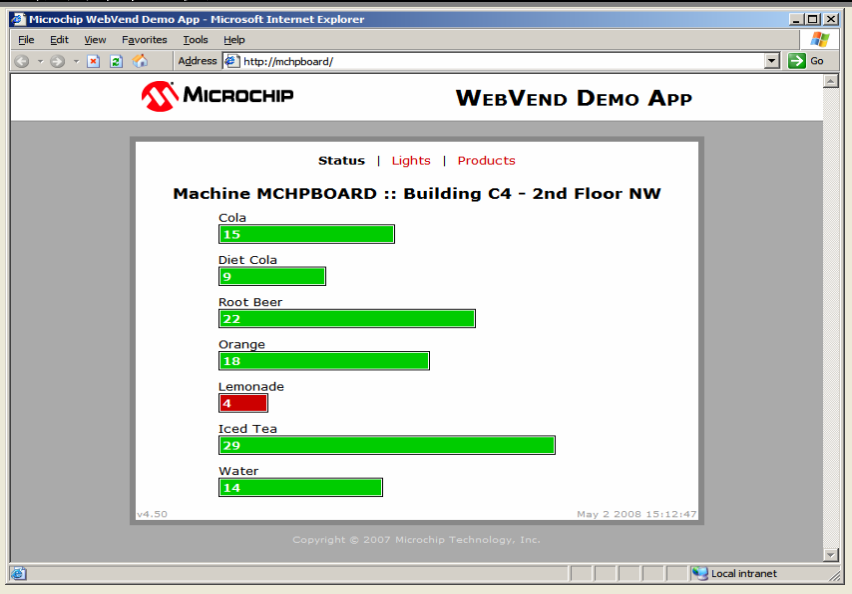
斷開除錯器。 驗證在選擇某產品出售之後，產品的數量和顏色是否發生了相應的變化。

## CustomHTTPApp.c

20

```
1367
1368 void HTTPPrint_status(WORD item)
1369 {
1370     if(Products[item].stock < 8)
1371         TCPPutROMString(sktHTTP, (ROM BYTE*)"low");
1372     else
1373         TCPPutROMString(sktHTTP, (ROM BYTE*)"ok");
1374 }
1375
1376 #endif
```

## 完成附加步驟之后的示範网页



# 實驗6

## 通過GET方法實現基於網頁的控制

## 目的

現在可對自動販賣機進行遠端監視了，再添加一些控制功能將會更加完善。 正如所討論的那樣，可通過兩種不同的HTTP方法實現：GET和POST。

在本實驗中，您將學到如何使用GET方法向元件傳送少量的資料。在此過程中，將添加一個控制自動販賣機上指示燈的介面。 由於當前開發平臺只具有LED，將使用其中之一進行模擬。將在以後的課程中討論通過POST 方法進行的控制。

## 要求

**軟體：** 完成的實驗 5或從解決方案5入手

**環境：** MPLAB® IDE 8.01、COM4201 Lab Build Installer v1.01和Crimson Editor 3.70

**C編譯器** MPLAB C18 v3.16 或更高版本/C30 v3.01或更高版本/ C32 v1.02 或更高版本

**硬體工具：** PICDEM.net™ 2或帶乙太網PICtail™ Plus及PIC24FJ128GA010或 PIC32MX360F512L PIM的Explorer 16開發板
MPLAB Real ICE™（使用PIC32時需要）或MPLAB ICD 2

**網頁目錄：**
**C:\RTC\COM4201\Microchip Solutions\TCPIP Demo App\WebPages2**

## 步驟

1. 通過Crimson Editor打開`lights.htm`。此檔包含將要用於控制LED的HTML表單。

   開始 } 所有程式 } Crimson Editor } Crimson Editor

2. 找到`<form>`標記，注意，`method`屬性設置爲`get`。此表單包含兩個輸入按鈕，它們均命名爲lights。

   找到單選按鈕輸入。注意這兩個按鈕均命名爲`lights`。 提交此表單之後，lights參數的值將被設置爲與選中的單選按鈕相等。

## index.htm

2

```
16
17 <!-- These two fields create the On/Off radio selectors -->
18 <input type="radio" name="lights" value="1" /> On
19 <input type="radio" name="lights" value="0" /> Off
20
```

3 通過MPLAB打開`CustomHTTPApp.c`。通過GET提交的表單在`HTTPExecuteGet()`中進行處理。**找到**此函數。

注：此函數的第一步是確定要訪問的檔案名（即，所提交的表單）。 通過呼叫`MPFSGetFilename()`來處理此操作，然後使用`memcmp`變數來檢查檔案名。

`memcmppgm2ram()`將RAM中的陣列與程式記憶體中的陣列進行比較。它接受兩個指標和一個長度參數。如果這兩個陣列相匹配，則將返回0，這便是將比較操作取反的原因。

4 更新第一次比較以檢查`lights.htm`。注意還應將長度參數更新爲`10`。

5 如果名稱相匹配，應用程式現在應查找一個名爲“`lights`”的參數。如果呼叫`HTTPGetROMArg()`返回的指標不爲`NULL`且匹配ASCII “`1`”，那麼應拉高接腳`LED4_IO`。否則，該接腳應被拉低。

網頁表單返回的參數始終表示爲字串。如果返回數位值，這些數位值必須由它們的字串表示形式解析得到。

6 如下頁所示，插入表單處理功能。
刪除`HTTPExecuteGet()`中的其餘程式，包括用於`cookies.htm`和`leds.cgi`的處理程式。

這些表單是在演示應用程式中實現的，而項目網頁中已不存在此表單。**確保**此函數仍然返回`HTTP_IO_DONE`，這將指示HTTP2伺服器此函數已完成，不應再呼叫。

7 編譯專案並**燒錄**開發板。
斷開除錯器。

8 訪問您的開發板網頁，點選**Lights**鏈結。
驗證能否通過網頁控制開發板上的某個LED。

## CustomHTTPApp.c

```
162     HTTP_IO_RESULT HTTPExecuteGet(void)
163
164   Internal:
165     See documentation in the TCP/IP Stack API or HTTP2.h for details.
166   ********************************************************************
167 HTTP_IO_RESULT HTTPExecuteGet(void)
168 {
169     BYTE *ptr;
170     BYTE filename[20];
171
172     // Load the file name
173     // Make sure BYTE filename[] above is large enough for your longest name
174     MPFSGetFilename(curHTTP.file, filename, 20);
175
176     // If its the forms.htm page
177     if(!memcmppgm2ram(filename, "lights.htm", 10))
178     {
179         ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"lights");
180         if(ptr)
181             LED4_IO = (*ptr == '1');
182     }
183
184     return HTTP_IO_DONE;
185 }
186
```

4

5

6

## 結果

正確的實現允許您通過網頁控制LED的狀態。

## 程式分析

如前面提到的那樣，通過GET方法接收到的資料存儲在`curHTTP.data`中。此資料可通過`HTTPGetROMArg()`函數找到，該函數將返回指向此值的指標或NULL（如果未找到請求的參數）。然後此指針將與期望值進行比較，以確定相應的操作。

由於`HTTPExecuteGet()`處理所有GET表單，故此函數必須通過檢查檔案名對所提交的表單進行解析。`MPFSGetFilename()`函數確定伺服器所訪問的檔案名，並將其存回臨時字串，用於比較。

## 附加步驟

大多數網頁都預選擇了當前值，從而爲客戶提供更爲直觀的介面。可通過動態變數爲您的表單實現此功能。

9

再次在**Crimson Editor** 中打開`lights.htm`。將在此檔中添加一個動態變數，以根據LED是點亮還是熄滅來預選擇正確的單選按鈕。

10 如果某個單選按鈕具有checked屬性，那麼將選中此按鈕。在每個單選按鈕中**插入**一個動態變數 ，該變數允許應用程式動態插入此屬性。

**index.htm**

```
16
17 <!-- These two fields create the On/Off radio selectors -->
18 <input type="radio" name="lights" value="1" ~light_chk(1)~ /> On
19 <input type="radio" name="lights" value="0" ~light_chk(0)~ /> Off
20
```

11 **保存**對HTML程式所作的更改，並使用**MPFS2應用程式**將新頁面**上傳**到開發板。

12 通過MPLAB將對HTTPPrint_light_chk()的呼叫插入到檔CustomHTTPApp.c中。

此函數接受一個WORD參數。

在此函數中，插入一個檢查：檢查所接收的參數是否與LED4_IO的當前狀態相匹配。如果匹配，則使用TCPPutROMString()將字串“checked”寫到頁面中。

**CustomHTTPApp.c**

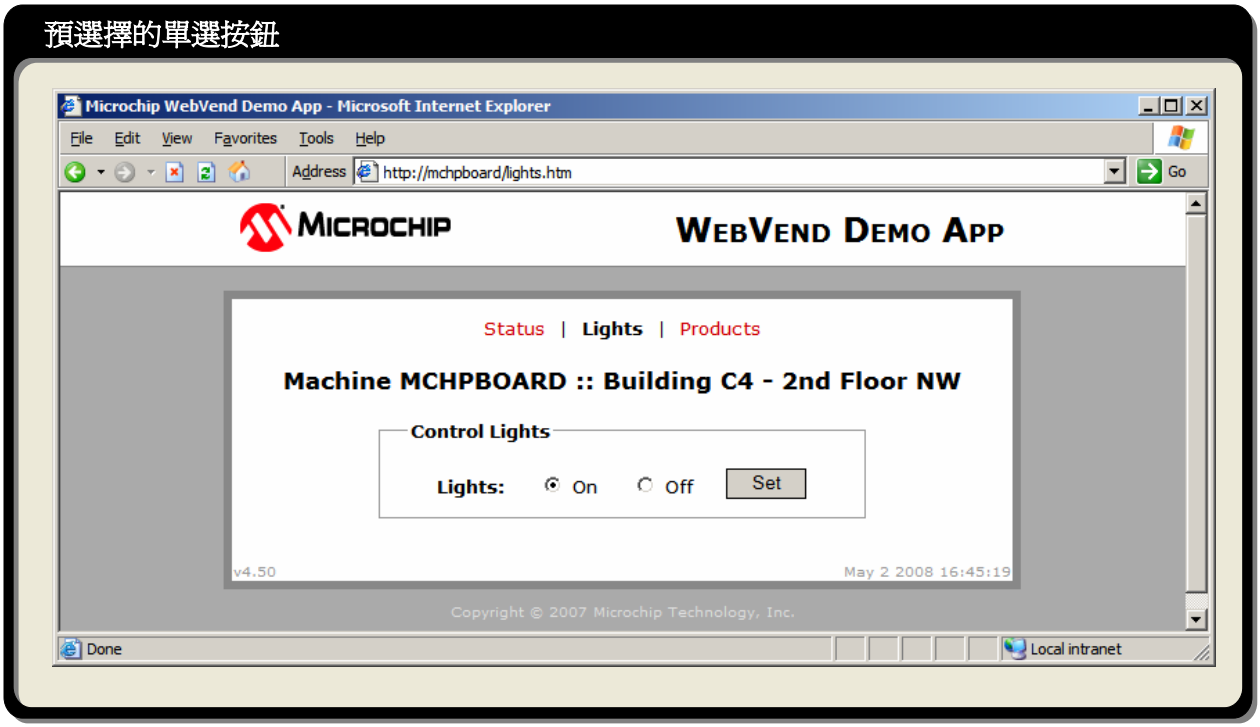
```
1317
1318 void HTTPPrint_light_chk(WORD state)
1319 {
1320     if(state == LED4_IO)
1321         TCPPutROMString(sktHTTP, (ROM BYTE*)"checked");
1322 }
1323
1324 #endif
1325
```

13 編譯所有經過更改的檔、選擇編程器，並將程式燒錄到開發板中。
斷開除錯器。
驗證在載入頁面時，表單中是否已預選擇了LED的當前狀態。

## 附加步驟結果

在本部分中，爲單選按鈕添加了動態變數。此動態變數將輸出“checked”或無任何輸出，具體取決於LED的狀態。通過輸出“checked”，伺服器可控制表單輸入域中的默認狀態。在載入頁面時，此網頁已預選擇了兩個單選按鈕中的一個，如下頁所示。

預選擇的單選按鈕




# 結論

本實驗爲設備添加了一個簡單的表單處理。對於多數應用，需要處理的只有GET表單。由於資料已存儲在記憶體中，從而使得這些表單非常易於處理。但請切記，GET表單輸入資料的上限爲大約100個位元組。

以後的課程將涉及POST表單，
此表單允許提交資料的範圍從幾位元組到幾百萬位元組。如需瞭解更多 資訊，
請參加關於本主題的網上研討會 。