MICROCHIP

MASTERs Conference

1538-TCP2

Designing Embedded TCP/IP Monitor and Control

Lab Manual

Designing Embedded TCP/IP Monitor and Control

Lab Background/Flow, Solutions & Symbols:

Exercise Background and Flow:

Many embedded applications can be functionally enhanced if the designer includes the ability to access the system via the existing internet infrastructure. This concept was the focus of a webinar series, available on www.microchip.com/ethernet, that discusses adding Ethernet networking capability to a soda vending machine. That webinar series was the basis for developing this class and the 6 associated labs.

The user can choose one of 3 MCU platforms: 1) PIC18F, 2)PIC24F or 3)PIC32MX. Three MAC/PHY options are also available in the class. The PIC18F, via the PICDEM.net 2 board offers an integrated MAC/PHY or an external ENC28J60. The latter two offer one of three MAC/PHYs: ENC28J60 10Mbps, ENC624J600 10/100 Mbps, or MRF24WB0 wireless 1-2Mbps all via the Ethernet PICTail Plus modules This choice is facilitated in the source files via *#if defines* shown in HardwareProfile.h header file.

Lab1-Join the Network-starts by using a demo application from the Microchip TCPIP stack distribution then utilizes the *TCPIPConfig* utility to set the development board up for the balance of the labs. Completion of this lab is required as the board's assigned MAC address and user-specified board name is copied to the subsequent labs files via the *Replicate HWConfig.bat* routine *via the TCPIPConfig.h* file.

Lab 2 is an exercise in designing an business card image on a HTML/CSS web page. Since this is a practice exercise the files are erased during subsequent labs. However, the user is given modifiable vending machine web pages to use for Labs 3 thru 6. These labs steer the user thru the process of integrating the vending machine application code with the TCP/IP stack, removing blocking code for maximum performance, and customizing for monitoring and controlling remote apps.

All the files used in these exercises have been loaded into the C\RTC\COM4201\ directories along with the most recent version of the Microchip released TCP/IP stack. When newer stacks are web-downloaded, they are stored in a C:\Microchip Solutions folder during normal install. The applications used in this class are available in the most recent stack versions.

Solutions:

All but Lab 1 have solution projects available in their respective folders. However, for some labs just compiling the MCU solution is not adequate to complete the exercise. When using a solution workspace and it's associated completed firmware entries refer to the enclosed table for a starting point to complete the exercise.

Solution Steps	
Lab 3:	Step 8
Lab 4:	Step 6
Lab 5:	Step 14
Lab 6:	Step 3

Code Analysis-HardwareProfile.h

Each compiler, facilitated by the HardwareProfile.h file and partially shown below, can determine the Microchip development platform used based on the compiler and part selected within the MPLAB project. Unique versions of this file are needed to differentiate between the MAC/PHY choices in the class. This is handled by TCPIP HWConfig.bat in Lab 1.

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
```

Symbols

Helpful information for completing the lab2

BACKGROUND and PURPOSE of the Lab

PC configuration for the lab.

1 Procedural Commands to be executed

Sections of Ansi-C code

Sections of HTML/CSS code

Target results of the lab

Code Analysis and Conclusions of the lab

Cautions or warnings about frequent mistakes

Table of Contents

Lab Exercise 1

Join the Network

Purpose

Before a device can be connected to a network, it needs a unique host name and MAC address. This ensures that the device can communicate independently with other network nodes, and prevents duplicate addresses on the network.
In this first lab, the **TCPIPConfig Wizard** will be used to configure a unique address for your development hardware. These changes will be written to a configuration file named `TCPIPConfig.h` that is included in the project file, and will be compiled into the `COM4201 WiFi Demo App` project. To finish the lab, you will upload a set of example web pages and test the network connection.

Requirements

Software:	**TCPIPConfig Wizard from 1538 Lab Build Installer v1.536**
Environment:	MPLAB® IDE 8.60 or later or X, 1538 Lab Build Installer 1.536
C Compiler:	MPLAB C18 v3.38 or later, C30 v3.30 or later, or C32 v1.12 or later
H/W Tools:	PIC18F: PICDEM.net™ 2, or PIC24FFJ128GA010 PIM: Explorer 16 w/ 10Mbps, 10/100Mbps PICtail™ Plus or MRF24WB0 or PIC32MX360 or PIC32MX460 PIM: Explorer 16 & 10Mbps, 10/100Mbps PICtail™ Plus or MRF24WB0 AND MPLAB Real ICE™ , MPLAB ICD3 AND DHCP Enabled Router (wired or wireless)
Lab Files:	C:\Masters\1538\Lab1...

Objectives

- Configure the lab class hardware platform and ensure it is connected to the local LAN.
- Use the TCPIP Config Utility to make specify the hardware configuration.

Procedures

1 If you have a previous MPLAB project open, you must first close it by selecting from the menu:
File ▸ Close Workspace

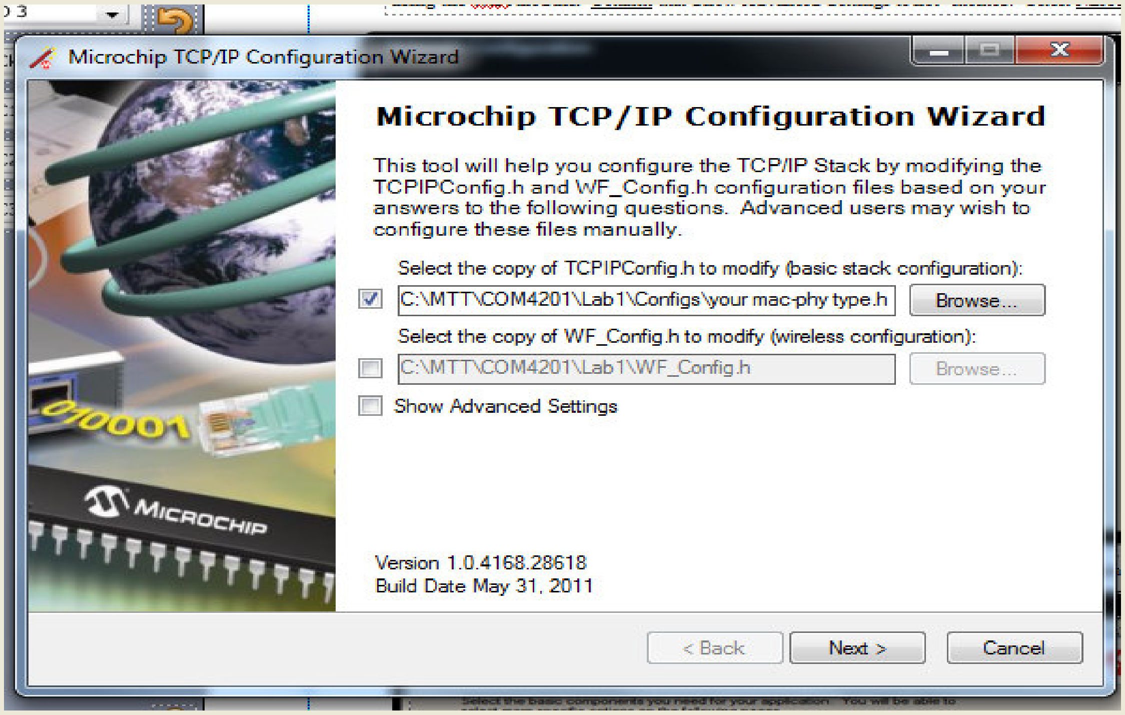
Confirm the following directory/folder is used during the utility processing:

Project Directory:
C:\Masters\1538\Lab1\*

2

Launch the **TCPIP Config Wizard** to modify the TCPIPConfig.h.
Start ▸ Programs ▸ Microchip ▸ 1538 ▸ TCPIPConfig Wizard
Browse to **ENSURE** the Lab1\Configs directory shown below. **Check the CONFIGURE WIRELESS SETTINGS if using the WiFi module.** Confirm that **Show Advanced Settings** is **not** checked. Select NEXT.

Network Configuration



Microchip TCP/IP Configuration Wizard

This tool will help you configure the TCP/IP Stack by modifying the TCPIPConfig.h and WF_Config.h configuration files based on your answers to the following questions. Advanced users may wish to configure these files manually.

Select the copy of TCPIPConfig.h to modify (basic stack configuration):
[✓] C:\MTT\COM4201\Lab1\Configs\your mac-phy type.h Browse...

Select the copy of WF_Config.h to modify (wireless configuration):
[ ] C:\MTT\COM4201\Lab1\WF_Config.h Browse...

[ ] Show Advanced Settings

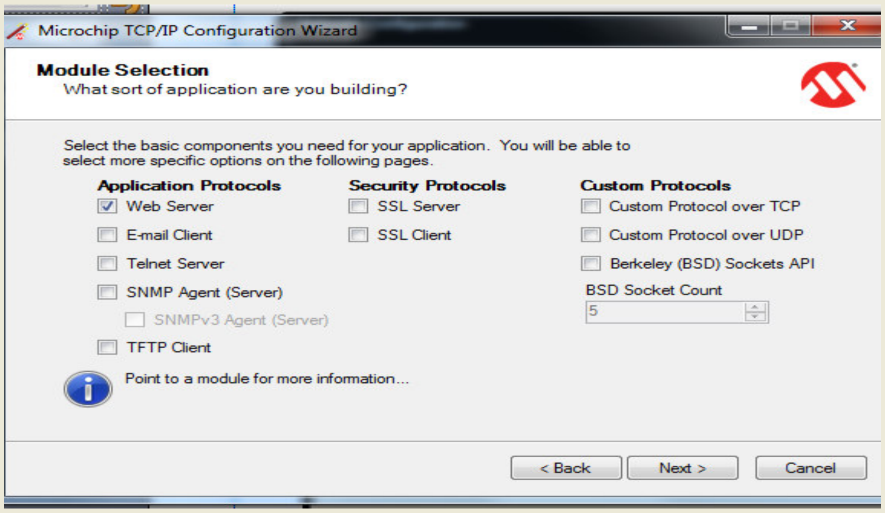
Version 1.0.4168.28618
Build Date May 31, 2011

< Back Next > Cancel

3

Ensure only the Web Server box is checked in the next two screens-Module Selection and Example Modules:

Module Selection



Module Selection
What sort of application are you building?

Select the basic components you need for your application. You will be able to select more specific options on the following pages.

Application Protocols
[✓] Web Server
[ ] E-mail Client
[ ] Telnet Server
[ ] SNMP Agent (Server)
[ ] SNMPv3 Agent (Server)
[ ] TFTP Client

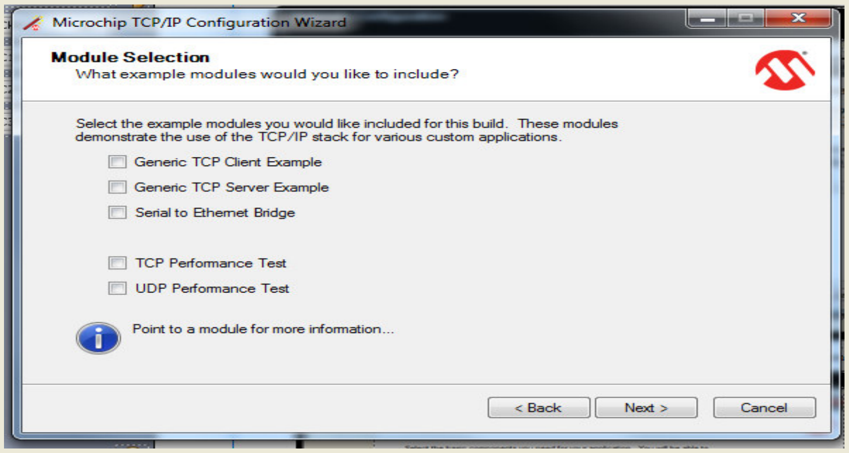
Security Protocols
[ ] SSL Server
[ ] SSL Client

Custom Protocols
[ ] Custom Protocol over TCP
[ ] Custom Protocol over UDP
[ ] Berkeley (BSD) Sockets API
BSD Socket Count: 5

Point to a module for more information...

< Back Next > Cancel

Example Modules



Microchip TCP/IP Configuration Wizard

Module Selection
What example modules would you like to include?

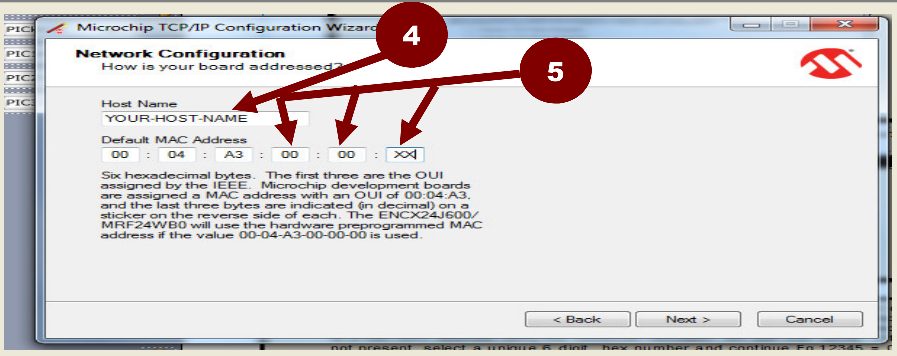
Select the example modules you would like included for this build. These modules demonstrate the use of the TCP/IP stack for various custom applications.

- [ ] Generic TCP Client Example
- [ ] Generic TCP Server Example
- [ ] Serial to Ethernet Bridge
- [ ] TCP Performance Test
- [ ] UDP Performance Test

Point to a module for more information...

4 **Define and enter** a unique host name for your board on the NETWORK CONFIGURATION screen shown below. The name must be 15 characters or less, and cannot contain \ / : * ? " ; |

Network Configuration



Microchip TCP/IP Configuration Wizard

Network Configuration
How is your board addressed?

Host Name
YOUR-HOST-NAME

Default MAC Address
00 : 04 : A3 : 00 : 00 : XX

Six hexadecimal bytes. The first three are the OUI assigned by the IEEE. Microchip development boards are assigned a MAC address with an OUI of 00:04:A3, and the last three bytes are indicated (in decimal) on a sticker on the reverse side of each. The ENCX24J600/MRF24WB0 will use the hardware preprogrammed MAC address if the value 00-04-A3-00-00-00 is used.

Microchip assigned Ethernet MAC address decimal numbers are labeled on the wired 10Mbps PICTail Plus module or on the back of the PICDEM.net 2 development board. When using either of these, step 5 will require you to convert the Ethernet decimal number to a hex MAC address using the Windows calculator, left pad with zeros. If the number is not present, select a unique 6 digit hex number and continue.Eg.12345 = 00:30:39 hex. 10/100Mbps PICTail and MRF24WB0 have preprogrammed MAC addresses, so click NEXT thru step 5.

5 If using PICDEM.net 2 or 10Mbps PicTail plus module, Enter the last 3 bytes of the MAC address field, in hex corresponding to the Ethernet serial decimal number printed on the label. If using the 10/100 PICtail Plus or the ZG2100, ensure that the last 3 hexadecimal digits are 00-00-00.

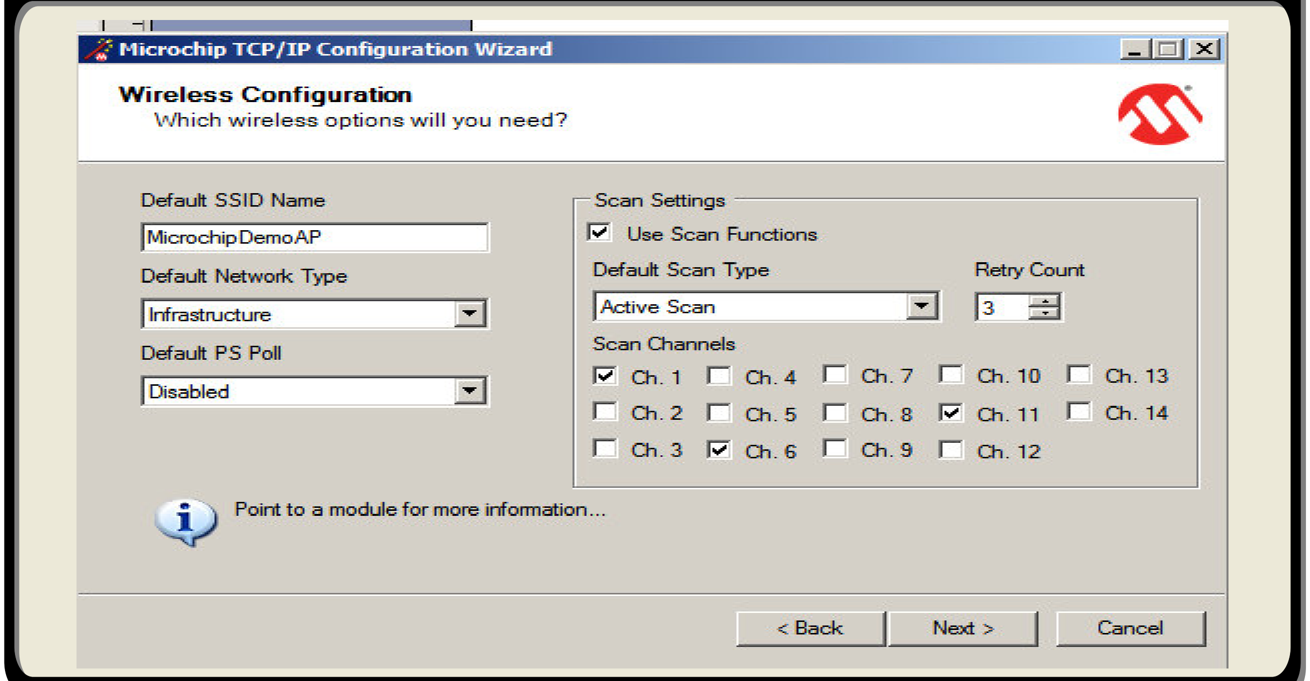
For wired configurations with the ENC28J60 or ENCx24J600

Click **Next** thru the following screens, and click **Finish** to complete the session.
Skip to STEP 6 to continue the configuration process.

For wireless configurations using MRF24WB0 PICtail Plus Module:

Check the appropriate boxes for the regional configurations in the 2 screens below:
Click **Next** thru the following screens (no Security, no Web page change, no MPFS image change), until
Click **Finish** to complete the utility session.
Continue to Step 6 to continue the configuration process.

Network Configuration



Microchip TCP/IP Configuration Wizard

Wireless Configuration
Which wireless options will you need?

Default SSID Name: MicrochipDemoAP
Default Network Type: Infrastructure
Default PS Poll: Disabled

Scan Settings
- [x] Use Scan Functions

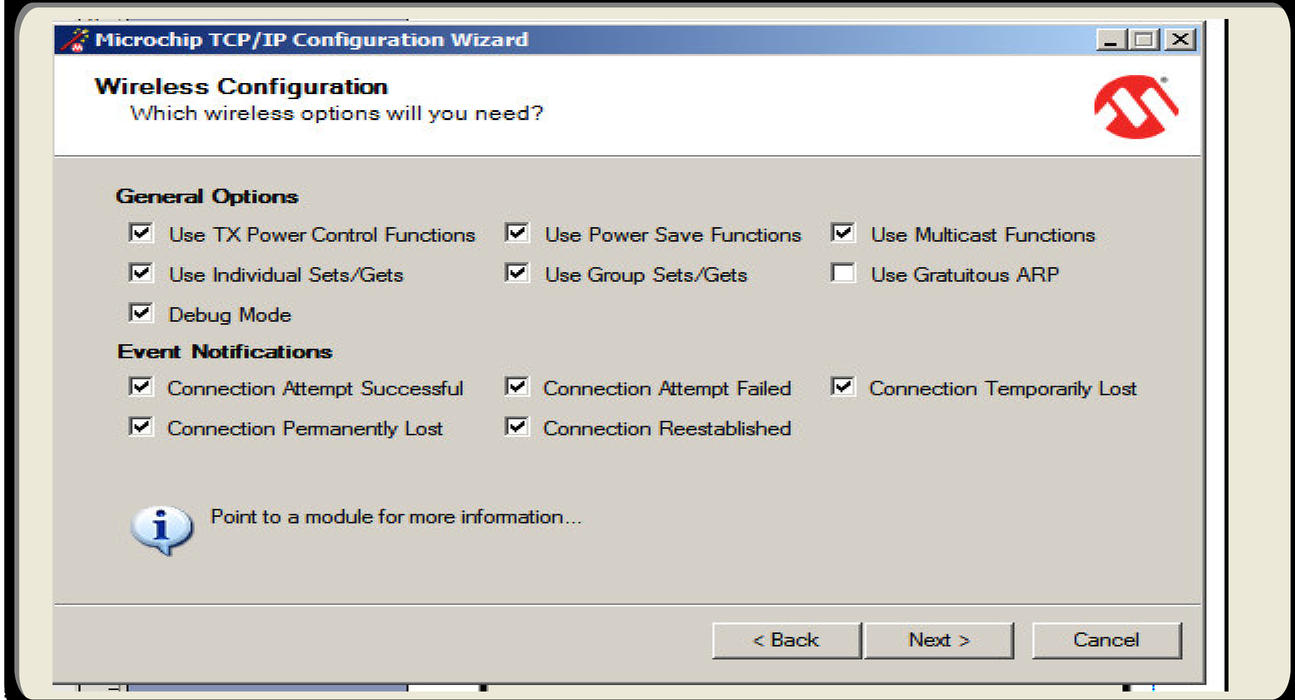
Default Scan Type: Active Scan
Retry Count: 3

Scan Channels

[x] Ch. 1	[ ] Ch. 4	[ ] Ch. 7	[ ] Ch. 10	[ ] Ch. 13
[ ] Ch. 2	[ ] Ch. 5	[ ] Ch. 8	[x] Ch. 11	[ ] Ch. 14
[ ] Ch. 3	[x] Ch. 6	[ ] Ch. 9	[ ] Ch. 12	

Point to a module for more information...

Network Configuration



Microchip TCP/IP Configuration Wizard

Wireless Configuration
Which wireless options will you need?

General Options
- [x] Use TX Power Control Functions
- [x] Use Power Save Functions
- [x] Use Multicast Functions
- [x] Use Individual Sets/Gets
- [x] Use Group Sets/Gets
- [ ] Use Gratuitous ARP
- [x] Debug Mode

Event Notifications
- [x] Connection Attempt Successful
- [x] Connection Attempt Failed
- [x] Connection Temporarily Lost
- [x] Connection Permanently Lost
- [x] Connection Reestablished

Point to a module for more information...

6 Execute the batch script to copy TCPIP configuration just completed to all of the lab and lab solution project folders for use later in the class.

Start ▸ Programs ▸ Microchip ▸ 1538 ▸ Replicate TCPIPCONFIG-HeaderFile.bat

C:\Masters\1538\Lab1\Lab1-C**-***-***

7 MPLAB File ▸ Open Project

`Lab1-C18-PICDN2-ETH97.X`	for the PICDEM.net 2 with internal MAC/PHY setup
`Lab1-C18-PICDN2-MRF24W.X`	for the PICDEM.net 2 with Wireless MAC/PHY setup
`Lab1-C30-EXP16-ENC28.X`	for the Explorer 16 with PIC24F/dsPIC33F PIMs with 10Mbps module
`Lab1-C30-EXP16-ENC624.X`	for the Explorer 16 with PIC24F/dsPIC33F PIMs with 10/100Mbps module
`Lab1-C30-EXP16-MRF24WB.X`	for the Explorer 16 with PIC24F/dsPIC33F PIMs with WIFI module
`Lab1-C32-EXP16-ENC28.X`	for the Explorer 16 with PIC32MX3xx/4xx PIMs with 10Mbps module
`Lab1-C32-EXP16-ENC624.X`	for the Explorer 16 with PIC32MX3xx/4xx PIMs with 10/100Mbps module
`Lab1-C32-EXP16-MRF24WB.X`	for the Explorer 16 with PIC32MX3xx/4xx PIMs with WIFI module

8 **SEE THE SECTION 2.1 OR 2.3 OF APPENDIX A FOR MPLABX BUILD AND/OR RUN OPTIONS.** Build and Program the firmware into the selected device in RELEASE mode.

BUILD TO CONFIRM NO SYNTAX ERRORS

SEE SECTION 1.8 IN APPENDIX A FOR CONFIGURING YOUR PROGRAMMER

BUILD, PROGRAM AND RUN IN THE TARGET SYSTEM

About Network Settings

The demo application stores network configuration settings in EEPROM, so settings stored in the project file are only loaded when EEPROM data is not present. Procedure #11 is only required when new network settings are configured for the project. THIS ONLY NEEDS TO BE COMPLETED AFTER PROGRAMMING IN LAB1.

9 Confirm a new IP address from the Router appears on the LCD, something besides 169.254...
Clear the on-board EEPROM by:

1. Hold the right-most pushbutton down (S5 on PICDEM.net 2, S4 on Explorer 16)
2. Press and release the /MCLR button (S1 on PICDEM.net 2, S1 on Explorer 16)
3. Continue holding the right-most button until several LEDs blink once.
4. Release the right-most button

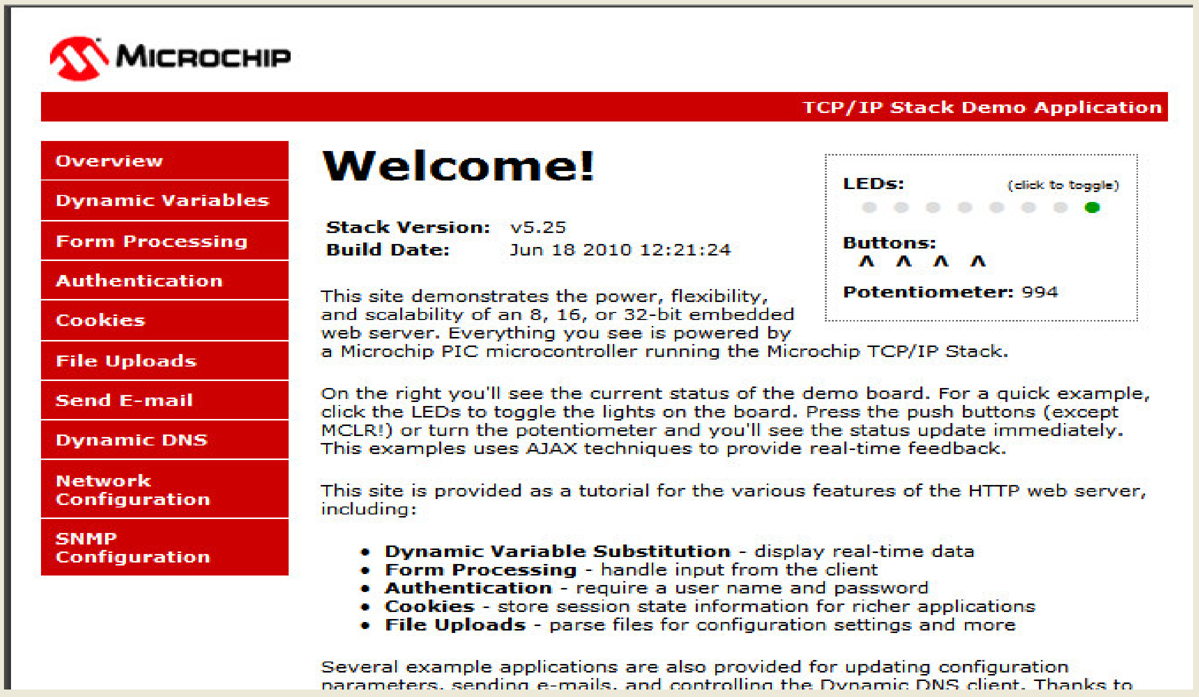
C:\Masters\1538\Lab1\MPFSImg2.bin

10 Open a web browser and **access** the webpage upload function by entering **http://*UNIQUE-NAME*/mpfsupload** into the address bar. Use the host name defined in Step 4.
Upload the **`MPFSImg2.bin`** bundle of web pages to the target system when prompted.
Click the link for **Site Main Page.** Navigate the demo website specifically in the upper right corner where you should be able to confirm the switch, potentiometer and LED operations.

Helpful Suggestions for unfound *hostname*

In some PC configurations, Firewall settings can block browsers from properly connected via the *hostname* specification used in Step 12 and follow-on labs. If this is experienced, it is recommended to disable the PC firewall software for completion of the lab exercises.
Also, multiple unsuccessful attempts to access the development boards by *hostname* can cause an unusable browser cache configuration. It is helpful to try one of the following two procedures prior to repeating the rebuild/reprogram sequence:
- Execute *nbtstat –R* in a DOS CMD window to purge the browser cache OR - Use the new IP address that was assigned by the router in-lieu of the *hostname in steps above*

Sample Web Page Display


TCP/IP Stack Demo Application

Welcome!

Stack Version: v5.25
Build Date: Jun 18 2010 12:21:24

This site demonstrates the power, flexibility, and scalability of an 8, 16, or 32-bit embedded web server. Everything you see is powered by a Microchip PIC microcontroller running the Microchip TCP/IP Stack.

On the right you'll see the current status of the demo board. For a quick example, click the LEDs to toggle the lights on the board. Press the push buttons (except MCLR!) or turn the potentiometer and you'll see the status update immediately. This examples uses AJAX techniques to provide real-time feedback.

This site is provided as a tutorial for the various features of the HTTP web server, including:

- **Dynamic Variable Substitution** - display real-time data
- **Form Processing** - handle input from the client
- **Authentication** - require a user name and password
- **Cookies** - store session state information for richer applications
- **File Uploads** - parse files for configuration settings and more

Several example applications are also provided for updating configuration parameters, sending e-mails, and controlling the Dynamic DNS client. Thanks to

Additional Features shown in this web page

This sample web page demonstrates many of the capabilities of the Microchip TCP/IP Stack. If you finish early, you can try out the various features. The LEDs on the board can be controlled via the status box at the upper right. (If you've selected the Explorer 16, the left-most LED shares a pin with a pushbutton, and so it will not display correctly and be uncontrollable.) The board reports back the status of its pushbuttons and potentiometer after they are changed on the target system.

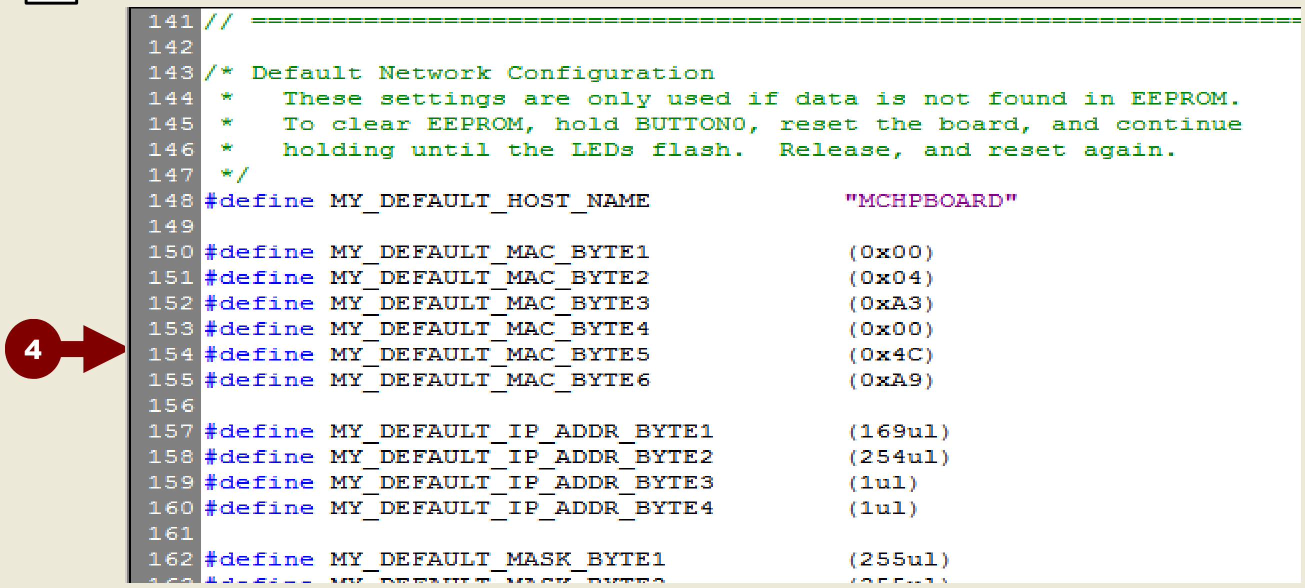
Results

You have just set up the TCPIP Demo App project on your development board and uploaded a sample set of web pages. When complete, you should see a page similar to the screen shown on the previous page: SAMPLE WEB PAGE DISPLAY.

Code Analysis

The TCP/IP Config Wizard utility was used to modify network software configuration settings via compiler macros (`#define`) stored in `TCPIPConfig.h`. This **utility** changes `the header file` in your project via designer input but also can be viewed and edited via a text editor. Open `TCPIPConfig.h` and find where the tool stored the updated host name and MAC address. You'll see many other settings as well, most of which can be configured by enabling advanced settings in the utility. Specific board configurations, as discussed earlier, are handled similarly in the `HardwareProfile.h` file.

TCPIPConfig.h



```
// ====================================================================

/* Default Network Configuration
 *   These settings are only used if data is not found in EEPROM.
 *   To clear EEPROM, hold BUTTON0, reset the board, and continue
 *   holding until the LEDs flash.  Release, and reset again.
 */
#define MY_DEFAULT_HOST_NAME            "MCHPBOARD"

#define MY_DEFAULT_MAC_BYTE1            (0x00)
#define MY_DEFAULT_MAC_BYTE2            (0x04)
#define MY_DEFAULT_MAC_BYTE3            (0xA3)
#define MY_DEFAULT_MAC_BYTE4            (0x00)
#define MY_DEFAULT_MAC_BYTE5            (0x4C)
#define MY_DEFAULT_MAC_BYTE6            (0xA9)

#define MY_DEFAULT_IP_ADDR_BYTE1        (169ul)
#define MY_DEFAULT_IP_ADDR_BYTE2        (254ul)
#define MY_DEFAULT_IP_ADDR_BYTE3        (1ul)
#define MY_DEFAULT_IP_ADDR_BYTE4        (1ul)

#define MY_DEFAULT_MASK_BYTE1           (255ul)
```

Conclusions

This lab has confirmed that your network setup and development board are operational. You have assigned a unique MAC address to your board, as well as a host name that can be accessed independently of the board's IP address. This host name operates through the NetBIOS Name Service, which is limited to the local subnet (in this case, your and your next door neighbor). Host name access outside of subnets is accomplished through DNS, which will be discussed later.
The IP address shown on the LCD screen has been automatically assigned by a DHCP server in a local router. DHCP is standard for most networks, and has replaced the need to hard code a static IP address. Static IP addresses can be configured when necessary.

THIS PAGE INTENTIONALLY LEFT BLANK

Lab Exercise 2

Using Wireshark for Protocol Debugging

Purpose

To obtain a working knowledge of using the Wireshark protocol sniffing software using the HTTP Get Message implementation from Lab 1 of this class.

Requirements

Software:	**Wireshark utility from 1538 Lab Build Installer v1.536**
Environment:	MPLAB® IDE 8.60 , X or later, 1538 Masters Install Script 1.536
C Compiler:	MPLAB C18 v3..38 or later, C30 v3.30 or later, or C32 v1.12 or later
H/W Tools:	PIC18F: PICDEM.net™ 2, OR
	PIC24FFJ128GA010 PIM: Explorer 16 w/ 10 Mbps, 10/100 Mbps PICtail Plus or MRF24WB0 OR
	PIC32MX360 or PIC32MX460 PIM: Explorer 16 & 10 Mbps, 10/100 Mbps PICtail Plus or MRF24WB0
	AND MPLAB REAL ICE™ In-Circuit Emulator, or MPLAB ICD 3
	AND DHCP Enabled Router (wired or wireless)
Lab Files:	C:\Masters\1538\Lab2...

Objectives

- Utilize Wireshark to test and debug communication traffic between Host and Client.
- Understand some TCPIP parameters such as: BOOTP, Payload content analysis, et. al.

Web Page Directory: C:\Master\1538\Lab2\*

Procedures

1 Confirm Lab 1 was complete, including executing the Replicate TCPIPCONFIG-HeaderFile.bat confirmi Ing your unique hardware. If you have a previous MPLAB IDE project open, you must first close it by selecting from the menu: File ▸ Close Workspace

2 Bundle and **upload** the completed page to the development board with MPFS utility.
The **MPFS2 Utility** generates the file `HTTPPrint.h`. This must be completed prior to compiling your MPLAB IDE project to include the dynamic variable callback functions already included in customHTTPapp.c.
Start ▸ Programs ▸ Microchip ▸ 1538 ▸ MPFS2
Confirm your 1538\Lab2\WebPages 2 folder and unique hostname is listed as the Device Address within the Upload Settings section. Click **Generate and Upload** to program your new pages to the development board. Confirm the utility successful completed.

3 MPLAB File ▸ Open Project

`Lab2-C18-PICDN2-ETH97.X` for the PICDEM.net 2 with internal MAC/PHY setup
`Lab2-C18-PICDN2-MRF24WB.X` for the PICDEM.net 2 with Wireless MAC/PHY setup
`Lab2-C30-EXP16-ENC28.X` for the Explorer 16 with PIC24F/dsPIC33F PIMs with 10Mbps module
`Lab2-C30-EXP16-ENC624.X` for the Explorer 16 with PIC24F/dsPIC33F PIMs with 10/100Mbps module
`Lab2-C30-EXP16-MRF24WB.X` for the Explorer 16 with PIC24F/dsPIC33F PIMs with WIFI module
`Lab2-C32-EXP16-ENC28.X` for the Explorer 16 with PIC32MX3xx/4xx PIMs with 10Mbps module
`Lab2-C32-EXP16-ENC624.X` for the Explorer 16 with PIC32MX3xx/4xx PIMs with 10/100Mbps module
`Lab2-C32-EXP16-MRF24WB.X` for the Explorer 16 with PIC32MX3xx/4xx PIMs with WIFI module

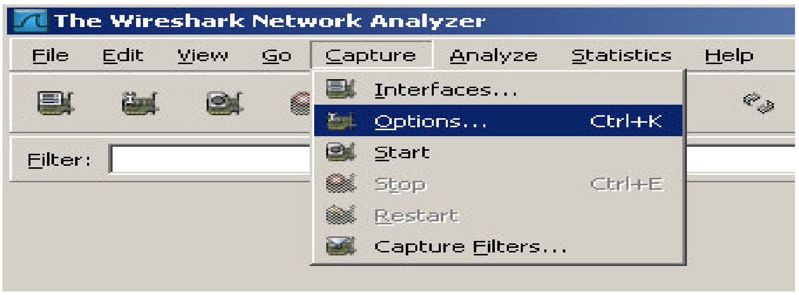
4 **SEE THE SECTION 2.1 OR 2.3 OF APPENDIX A FOR MPLABX BUILD AND/OR RUN OPTIONS.** Build and Program the firmware into the selected device in RELEASE mode.

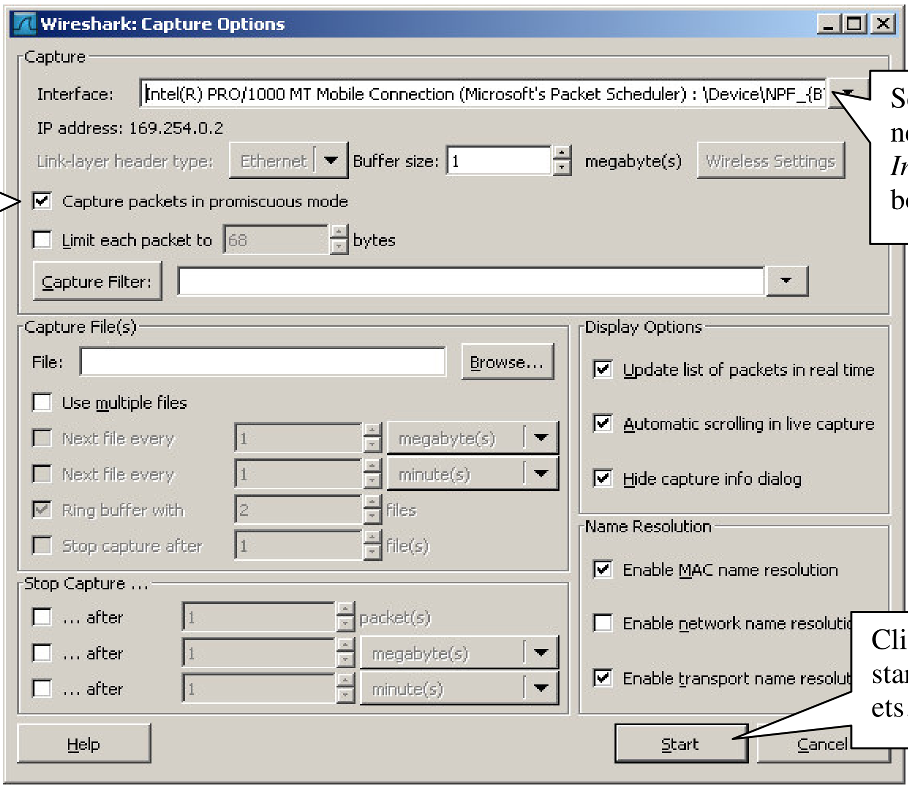
BUILD TO CONFIRM NO SYNTAX ERRORS

SEE SECTION 1.8 IN APPENDIX A FOR CONFIGURING YOUR PROGRAMMER

BUILD, PROGRAM AND RUN IN THE TARGET SYSTEM

5 Open Wireshark. Newer versions have a start up screen, and offer quick Network card selection and start of capture. Otherwise use the following to select the card and start a capture, as indicated in the next two screen shots:
1) Select CAPTURE/OPTIONS
2) Then select your PCs Ethernet card, Promiscuous mode, and Start a capture





Wireshark: Capture Options

Capture

Interface: Intel(R) PRO/1000 MT Mobile Connection (Microsoft's Packet Scheduler) : \Device\NPF_{B

IP address: 169.254.0.2

Link-layer header type: Ethernet | Buffer size: 1 megabyte(s) | Wireless Settings

- [✓] Capture packets in promiscuous mode
- [ ] Limit each packet to 68 bytes

Capture Filter:

Capture File(s)

File: | Browse...

- [ ] Use multiple files
- [ ] Next file every 1 megabyte(s)
- [ ] Next file every 1 minute(s)
- [✓] Ring buffer with 2 files
- [ ] Stop capture after 1 file(s)

Stop Capture ...

- [ ] ... after 1 packet(s)
- [ ] ... after 1 megabyte(s)
- [ ] ... after 1 minute(s)

Display Options

- [✓] Update list of packets in real time
- [✓] Automatic scrolling in live capture
- [✓] Hide capture info dialog

Name Resolution

- [✓] Enable MAC name resolution
- [ ] Enable network name resolution
- [✓] Enable transport name resolution

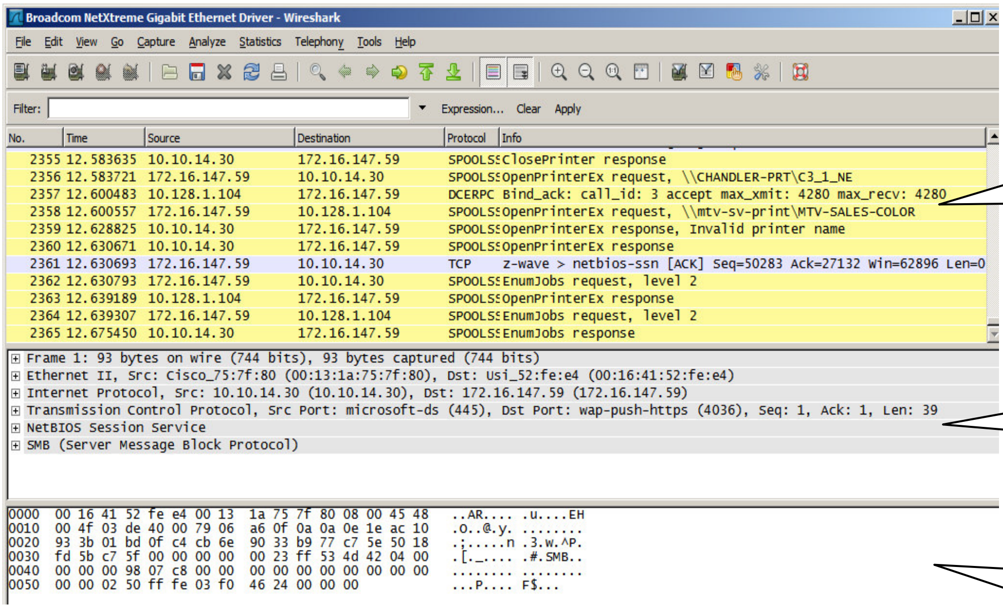
Help | Start | Cancel

Ensure that *Capture packets in promiscuous mode* is checked.

Select your PC's network card in the *Interface* drop down box.

Click *Start* button to start capturing packets…

6 STOP the capture after you see packets being captured in Wireshark. The figure below shows Wireshark with its three main panes: Packet List, Packet Details, and Packet Bytes. The Stop icon is shown below.



Broadcom NetXtreme Gigabit Ethernet Driver - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply

```
No.  Time      Source         Destination    Protocol Info
2355 12.583635 10.10.14.30    172.16.147.59  SPOOLSS ClosePrinter response
2356 12.583721 172.16.147.59  10.10.14.30    SPOOLSS OpenPrinterEx request, \\CHANDLER-PRT\C3_1_NE
2357 12.600483 10.128.1.104   172.16.147.59  DCERPC  Bind_ack: call_id: 3 accept max_xmit: 4280 max_recv: 4280
2358 12.600557 172.16.147.59  10.128.1.104   SPOOLSS OpenPrinterEx request, \\mtv-sv-print\MTV-SALES-COLOR
2359 12.628825 10.10.14.30    172.16.147.59  SPOOLSS OpenPrinterEx response, Invalid printer name
2360 12.630671 10.10.14.30    172.16.147.59  SPOOLSS OpenPrinterEx response
2361 12.630693 172.16.147.59  10.10.14.30    TCP     z-wave > netbios-ssn [ACK] Seq=50283 Ack=27132 Win=62896 Len=0
2362 12.630793 172.16.147.59  10.10.14.30    SPOOLSS EnumJobs request, level 2
2363 12.639189 10.128.1.104   172.16.147.59  SPOOLSS OpenPrinterEx response
2364 12.639307 172.16.147.59  10.128.1.104   SPOOLSS EnumJobs request, level 2
2365 12.675450 10.10.14.30    172.16.147.59  SPOOLSS EnumJobs response
```

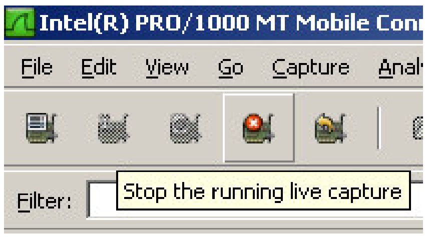
```
⊞ Frame 1: 93 bytes on wire (744 bits), 93 bytes captured (744 bits)
⊞ Ethernet II, Src: Cisco_75:7f:80 (00:13:1a:75:7f:80), Dst: Usi_52:fe:e4 (00:16:41:52:fe:e4)
⊞ Internet Protocol, Src: 10.10.14.30 (10.10.14.30), Dst: 172.16.147.59 (172.16.147.59)
⊞ Transmission Control Protocol, Src Port: microsoft-ds (445), Dst Port: wap-push-https (4036), Seq: 1, Ack: 1, Len: 39
⊞ NetBIOS Session Service
⊞ SMB (Server Message Block Protocol)
```

```
0000  00 16 41 52 fe e4 00 13  1a 75 7f 80 08 00 45 48   ..AR.... .u....EH
0010  00 4f 03 de 40 00 79 06  a6 0f 0a 0a 0e 1e ac 10   .O..@.y. ........
0020  93 3b 01 bd 0f c4 cb 6e  90 33 b9 77 c7 5e 50 18   .;.....n .3.w.^P.
0030  fd 5b c7 5f 00 00 00 00  00 23 ff 53 4d 42 04 00   .[._.... .#.SMB..
0040  00 00 00 98 07 c8 00 00  00 00 00 00 00 00 00 00   ........ ........
0050  00 00 02 50 ff fe 03 f0  46 24 00 00 00            ...P.... F$...
```

"Packet List" Pane

"Packet Details" Pane

"Packet Bytes" Pane

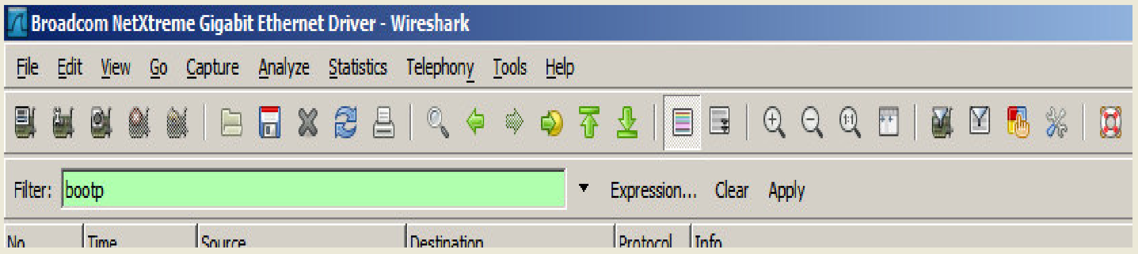


Intel(R) PRO/1000 MT Mobile Con
File Edit View Go Capture Anal
Filter:
Stop the running live capture

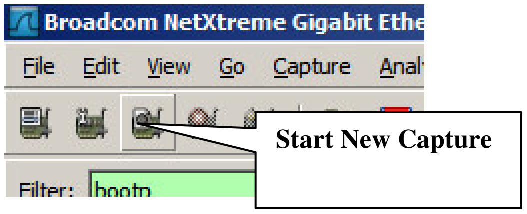
7

Enter a filter to look for DHCP protocols. Either use the expression builder or type directly into the filter bar.
Bootp is a protocol keyword for DHCP.
RESTART the capture again via the Start ICON shown below, disregard the request for saving.
PRESS the Master Reset button on the board. STOP the capture and investigate the bootp packets.

Wireshark Filter Bar


Broadcom NetXtreme Gigabit Ethernet Driver - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Expression... Clear Apply
No. Time Source Destination Protocol Info

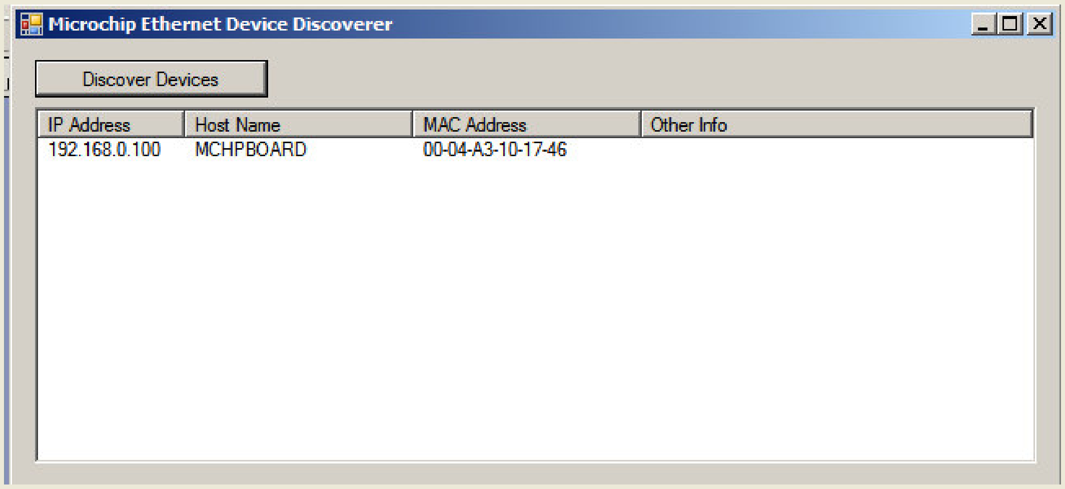
Filter: bootp


Broadcom NetXtreme Gigabit Ethe
File Edit View Go Capture Anal
Filter: bootp
Start New Capture

8

Use the Ethernet Discoverer executable from Microchip tools to confirm your dev tool's IP address.
Start ▸ Programs ▸ Programs ▸ Microchip ▸ 1538 ▸ Microchip Ethernet Discoverer

Network Discoverer

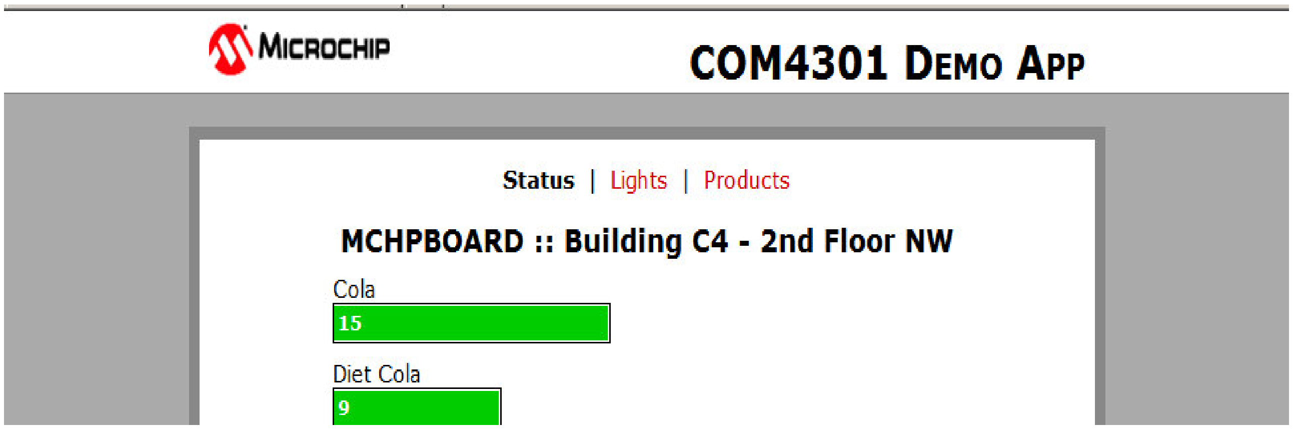


IP Address	Host Name	MAC Address	Other Info
192.168.0.100	MCHPBOARD	00-04-A3-10-17-46	

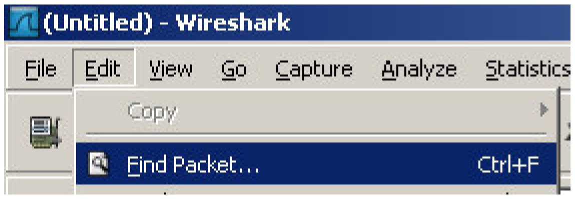
9

Change the filter again to only show your board's IP address.
Type it into the filter bar or use the Expression Builder.
a. To filter for all traffic with an IP address or destination source and destination use the following filter, ip.addr == <IP Address> with your board's IP address
b. If you find too many results, filter on HTTP as well : Example: ip.addr == <IP Address> && http
Notice the capture window is reduced to frames only associated with that IP address.

10 RESTART the capture. Open a web browser and point to **http://*UNIQUE-NAME*/** specified in Lab 1. Stay on the STATUS page which is default when receiving the index.htm page. STOP the Wireshark capture sequence. Observe the frames monitored by Wireshark. Note the TCP socket connection frames.

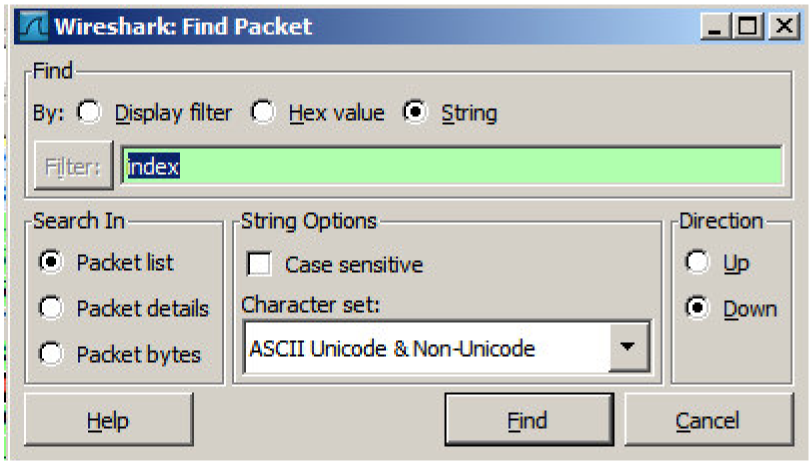

MICROCHIP
COM4301 DEMO APP
Status | Lights | Products
MCHPBOARD :: Building C4 - 2nd Floor NW
Cola
15
Diet Cola
9

11 View the capture in Wireshark utility to follow a TCP Stream for Payload and Socket processing.
Use the 'Find Packet…' command to display only the packets associated with the TCP transaction for the index.htm page loading. Select this from the menu > Edit > Find Packet, or Ctrl-F.
Notice the packet where the first instance of the string is found is highlighted, shown on the screen at the top of the next page.



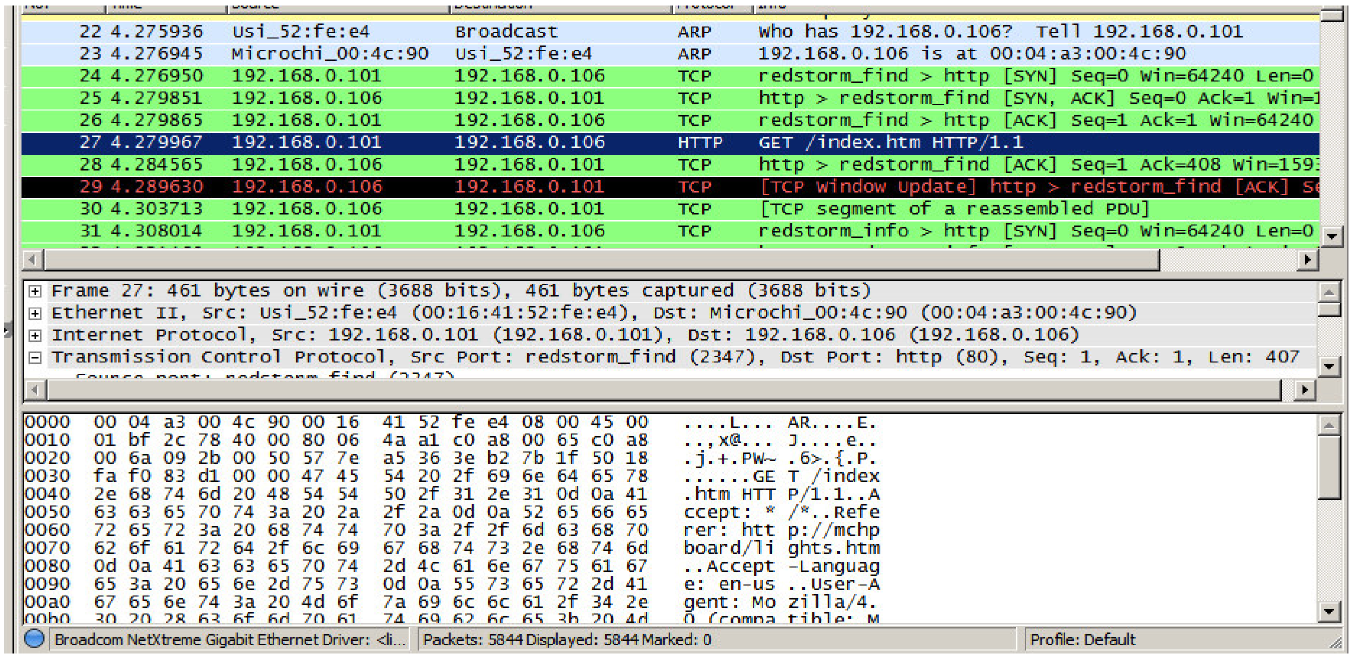
Helpful Suggestions for Wireshark Packet Finding

When a browser sends a request for a file from an HTTP server, it sends out the request in ASCII text. Therefore, the Find dialog box should use the 'String' search filter. Also, start with the packet finder searching in the PACKET LIST, as shown in the display below.


Wireshark: Find Packet

Find
By: Display filter Hex value [✓ String]
Filter: index

Search In: [✓ Packet list] Packet details Packet bytes
String Options: ☐ Case sensitive
Character set: ASCII Unicode & Non-Unicode
Direction: Up [✓ Down]



```
22 4.275936  Usi_52:fe:e4       Broadcast       ARP   Who has 192.168.0.106?  Tell 192.168.0.101
23 4.276945  Microchi_00:4c:90  Usi_52:fe:e4    ARP   192.168.0.106 is at 00:04:a3:00:4c:90
24 4.276950  192.168.0.101      192.168.0.106   TCP   redstorm_find > http [SYN] Seq=0 Win=64240 Len=0
25 4.279851  192.168.0.106      192.168.0.101   TCP   http > redstorm_find [SYN, ACK] Seq=0 Ack=1 Win=1
26 4.279865  192.168.0.101      192.168.0.106   TCP   redstorm_find > http [ACK] Seq=1 Ack=1 Win=64240
27 4.279967  192.168.0.101      192.168.0.106   HTTP  GET /index.htm HTTP/1.1
28 4.284565  192.168.0.106      192.168.0.101   TCP   http > redstorm_find [ACK] Seq=1 Ack=408 Win=159
29 4.289630  192.168.0.106      192.168.0.101   TCP   [TCP Window Update] http > redstorm_find [ACK] Se
30 4.303713  192.168.0.106      192.168.0.101   TCP   [TCP segment of a reassembled PDU]
31 4.308014  192.168.0.101      192.168.0.106   TCP   redstorm_info > http [SYN] Seq=0 Win=64240 Len=0
```

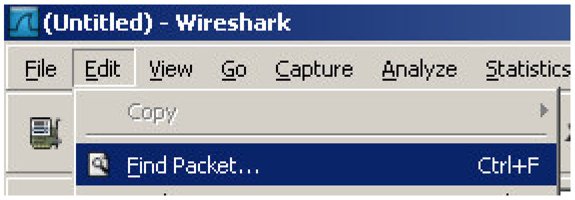
```
⊞ Frame 27: 461 bytes on wire (3688 bits), 461 bytes captured (3688 bits)
⊞ Ethernet II, Src: Usi_52:fe:e4 (00:16:41:52:fe:e4), Dst: Microchi_00:4c:90 (00:04:a3:00:4c:90)
⊞ Internet Protocol, Src: 192.168.0.101 (192.168.0.101), Dst: 192.168.0.106 (192.168.0.106)
⊟ Transmission Control Protocol, Src Port: redstorm_find (2347), Dst Port: http (80), Seq: 1, Ack: 1, Len: 407
```

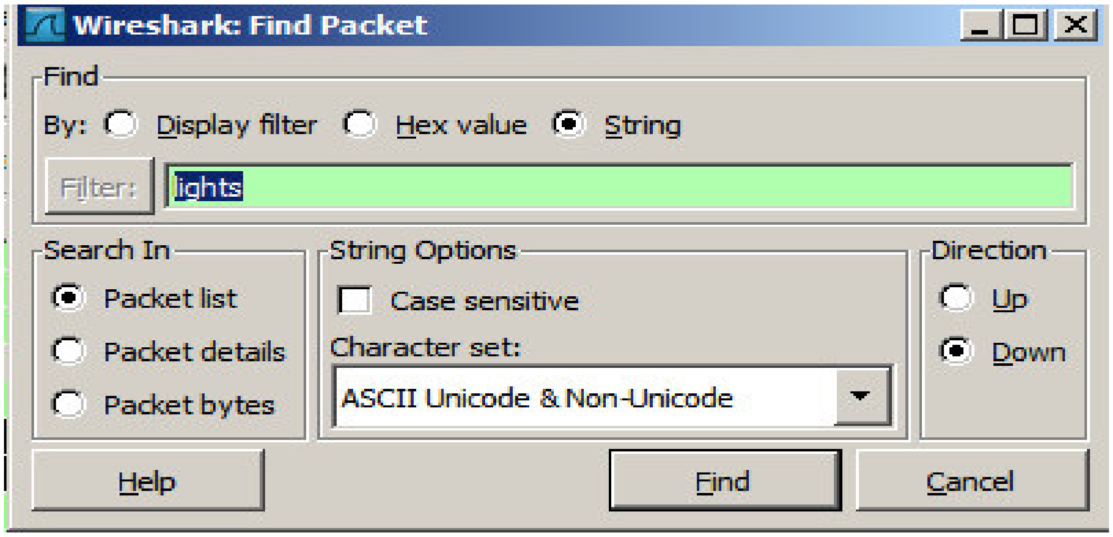
```
0000  00 04 a3 00 4c 90 00 16  41 52 fe e4 08 00 45 00   ....L... AR....E.
0010  01 bf 2c 78 40 00 80 06  4a a1 c0 a8 00 65 c0 a8   ..,x@... J....e..
0020  00 6a 09 2b 00 50 57 7e  a5 36 3e b2 7b 1f 50 18   .j.+.PW~ .6>.{.P.
0030  fa f0 83 d1 00 00 47 45  54 20 2f 69 6e 64 65 78   ......GE T /index
0040  2e 68 74 6d 20 48 54 54  50 2f 31 2e 31 0d 0a 41   .htm HTT P/1.1..A
0050  63 63 65 70 74 3a 20 2a  2f 2a 0d 0a 52 65 66 65   ccept: * /*..Refe
0060  72 65 72 3a 20 68 74 74  70 3a 2f 2f 6d 63 68 70   rer: htt p://mchp
0070  62 6f 61 72 64 2f 6c 69  67 68 74 73 2e 68 74 6d   board/li ghts.htm
0080  0d 0a 41 63 63 65 70 74  2d 4c 61 6e 67 75 61 67   ..Accept -Languag
0090  65 3a 20 65 6e 2d 75 73  0d 0a 55 73 65 72 2d 41   e: en-us ..User-A
00a0  67 65 6e 74 3a 20 4d 6f  7a 69 6c 6c 61 2f 34 2e   gent: Mo zilla/4.
```

Broadcom NetXtreme Gigabit Ethernet Driver: <li... | Packets: 5844 Displayed: 5844 Marked: 0 | Profile: Default

12 RESTART the capture.
Select the lights webpage heading in the browser at the top of the screen.
STOP the Wireshark capture sequence. Observe the frames monitored by Wireshark.

13 Now use the 'Find Packet…' command to display only the packets associated with the TCP transaction for the lights.htm page loading. Select this from the menu > Edit > Find Packet, or Ctrl-F.





Wireshark: Find Packet

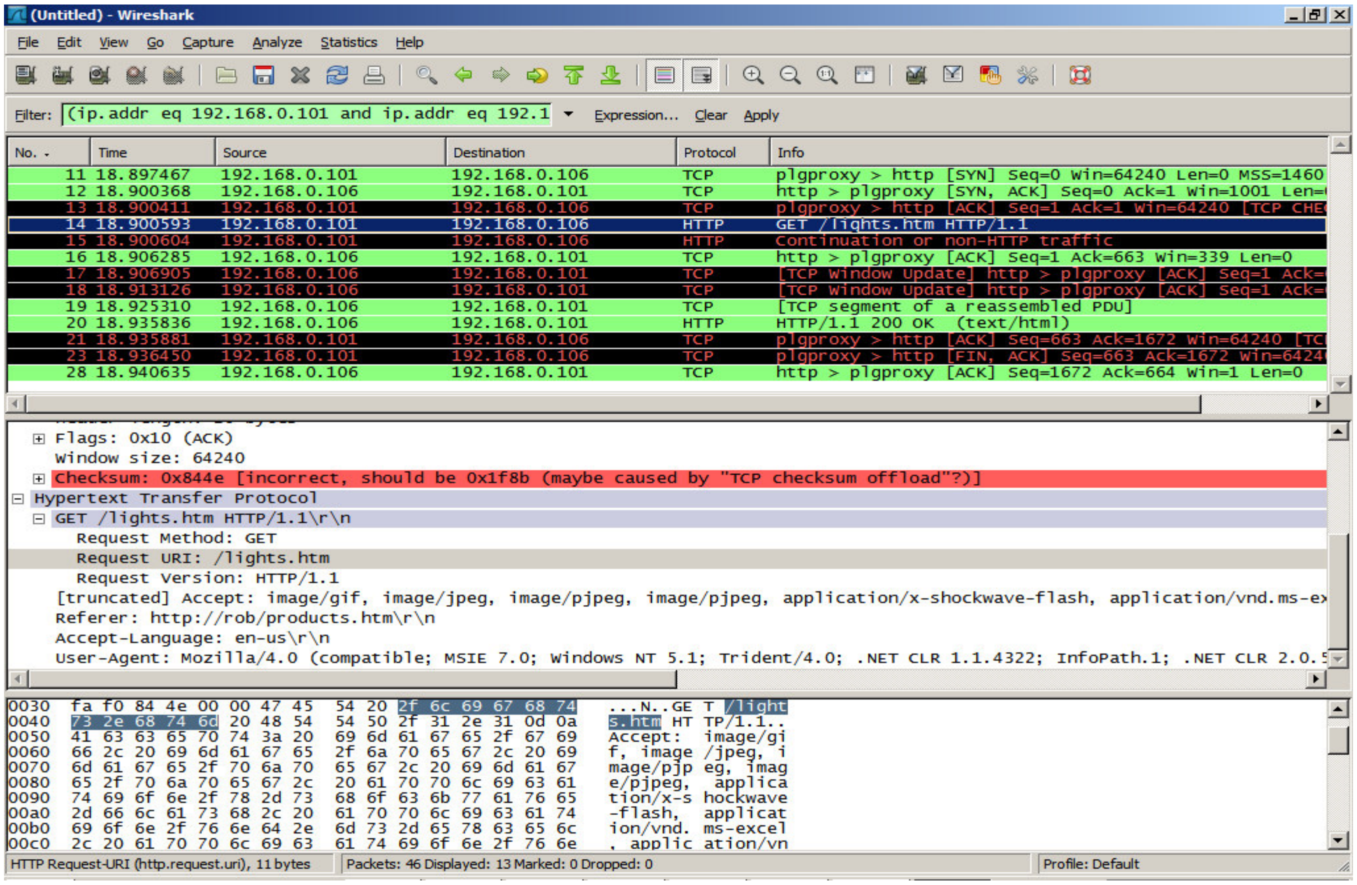
Find
By: ( ) Display filter ( ) Hex value [✓ String]
Filter: lights

Search In
[✓ Packet list]
( ) Packet details
( ) Packet bytes

String Options
☐ Case sensitive
Character set: ASCII Unicode & Non-Unicode

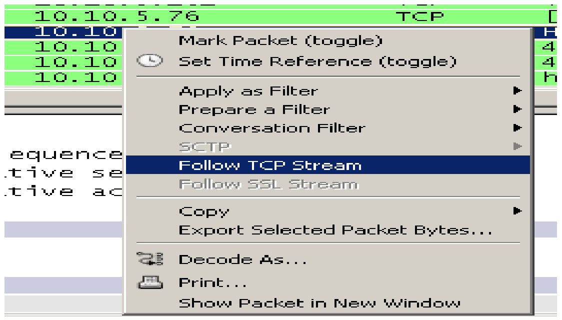
Direction
( ) Up
[✓ Down]

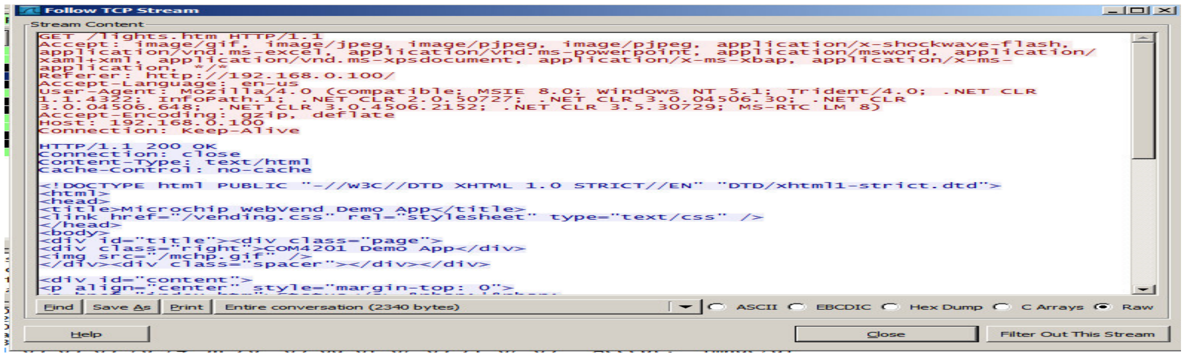
Help | Find | Cancel


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Expression... Clear Apply

```
Filter: (ip.addr eq 192.168.0.101 and ip.addr eq 192.1

No. -  Time        Source          Destination     Protocol Info
   11 18.897467   192.168.0.101   192.168.0.106   TCP      plgproxy > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460
   12 18.900368   192.168.0.106   192.168.0.101   TCP      http > plgproxy [SYN, ACK] Seq=0 Ack=1 Win=1001 Len=
   13 18.900411   192.168.0.101   192.168.0.106   TCP      plgproxy > http [ACK] Seq=1 Ack=1 Win=64240 [TCP CHE
   14 18.900593   192.168.0.101   192.168.0.106   HTTP     GET /lights.htm HTTP/1.1
   15 18.900604   192.168.0.101   192.168.0.106   HTTP     Continuation or non-HTTP traffic
   16 18.906285   192.168.0.106   192.168.0.101   TCP      http > plgproxy [ACK] Seq=1 Ack=663 Win=339 Len=0
   17 18.906905   192.168.0.106   192.168.0.101   TCP      [TCP Window Update] http > plgproxy [ACK] Seq=1 Ack=
   18 18.913126   192.168.0.106   192.168.0.101   TCP      [TCP Window Update] http > plgproxy [ACK] Seq=1 Ack=
   19 18.925310   192.168.0.106   192.168.0.101   TCP      [TCP segment of a reassembled PDU]
   20 18.935836   192.168.0.106   192.168.0.101   HTTP     HTTP/1.1 200 OK  (text/html)
   21 18.935881   192.168.0.101   192.168.0.106   TCP      plgproxy > http [ACK] Seq=663 Ack=1672 Win=64240 [TC
   23 18.936450   192.168.0.101   192.168.0.106   TCP      plgproxy > http [FIN, ACK] Seq=663 Ack=1672 Win=6424
   28 18.940635   192.168.0.106   192.168.0.101   TCP      http > plgproxy [ACK] Seq=1672 Ack=664 Win=1 Len=0

  ⊞ Flags: 0x10 (ACK)
    Window size: 64240
  ⊞ Checksum: 0x844e [incorrect, should be 0x1f8b (maybe caused by "TCP checksum offload"?)]
⊟ Hypertext Transfer Protocol
  ⊟ GET /lights.htm HTTP/1.1\r\n
      Request Method: GET
      Request URI: /lights.htm
      Request Version: HTTP/1.1
    [truncated] Accept: image/gif, image/jpeg, image/pjpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-ex
    Referer: http://rob/products.htm\r\n
    Accept-Language: en-us\r\n
    User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; InfoPath.1; .NET CLR 2.0.5

0030  fa f0 84 4e 00 00 47 45  54 20 2f 6c 69 67 68 74   ...N..GE T /light
0040  73 2e 68 74 6d 20 48 54  54 50 2f 31 2e 31 0d 0a   s.htm HT TP/1.1..
0050  41 63 63 65 70 74 3a 20  69 6d 61 67 65 2f 67 69   Accept:  image/gi
0060  66 2c 20 69 6d 61 67 65  2f 6a 70 65 67 2c 20 69   f, image /jpeg, i
0070  6d 61 67 65 2f 70 6a 70  65 67 2c 20 69 6d 61 67   mage/pjp eg, imag
0080  65 2f 70 6a 70 65 67 2c  20 61 70 70 6c 69 63 61   e/pjpeg,  applica
0090  74 69 6f 6e 2f 78 2d 73  68 6f 63 6b 77 61 76 65   tion/x-s hockwave
00a0  2d 66 6c 61 73 68 2c 20  61 70 70 6c 69 63 61 74   -flash,  applicat
00b0  69 6f 6e 2f 76 6e 64 2e  6d 73 2d 65 78 63 65 6c   ion/vnd. ms-excel
00c0  2c 20 61 70 70 6c 69 63  61 74 69 6f 6e 2f 76 6e   , applic ation/vn

HTTP Request-URI (http.request.uri), 11 bytes | Packets: 46 Displayed: 13 Marked: 0 Dropped: 0 | Profile: Default
```

14 Once the desired packet is found, you can right click on that packet in the packet list pane to get a context menu. Use the 'Follow TCP Stream' option. Close the screen shown below that opens automatically. Notice the filter contents change to only the requested details.





```
GET /lights.htm HTTP/1.1
Accept: image/gif, image/jpeg, image/pjpeg, image/pjpeg, application/x-shockwave-flash,
application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/
xaml+xml, application/vnd.ms-xpsdocument, application/x-ms-xbap, application/x-ms-
application, */*
Referer: http://192.168.0.100/
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR
1.1.4322; InfoPath.1; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30; .NET CLR
3.0.04506.648; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; MS-RTC LM 8)
Accept-Encoding: gzip, deflate
Host: 192.168.0.100
Connection: Keep-Alive

HTTP/1.1 200 OK
Connection: close
Content-Type: text/html
Cache-Control: no-cache

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 STRICT//EN" "DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Microchip WebVend Demo App</title>
<link href="/vending.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div id="title"><div class="page">
<div class="right">COM4201 Demo App</div>
<img src="/mchp.gif" />
</div><div class="spacer"></div></div>

<div id="content">
<p align="center" style="margin-top: 0">
```

Results

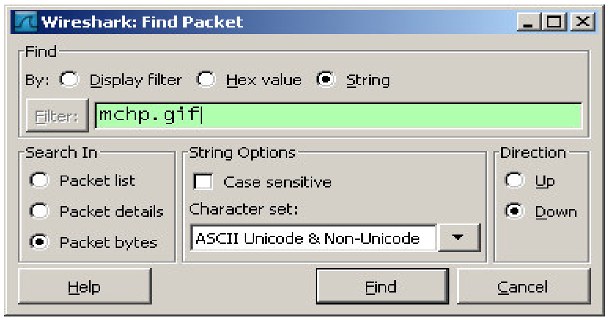
The Wireshark utility has been used to monitor the communication from a PC browser to the MCU acting as a webpage server. Specific IP addresses were filtered for, and a TCP communication stream was observed.

Analysis

Wireshark can be used to investigate the communication traffic by inserting a PC into the network. The utility can be used to see all traffic, or can be configured to filter traffic for a defined subset of the communication partners. Filters can also be used to track specific protocols and therefore help the designer confirm their firmware is behaving as expected.

15 Bonus Procedure

Figure out how to display only the packets associated with the TCP associated with downloading the mchp.gif file inside this design. Use the 'Find' command by selecting:
> Edit > Find Packets…


Wireshark: Find Packet

Find
By: Display filter, Hex value, [✓ String]
Filter: mchp.gif

Search In: Packet list, Packet details, [✓ Packet bytes]

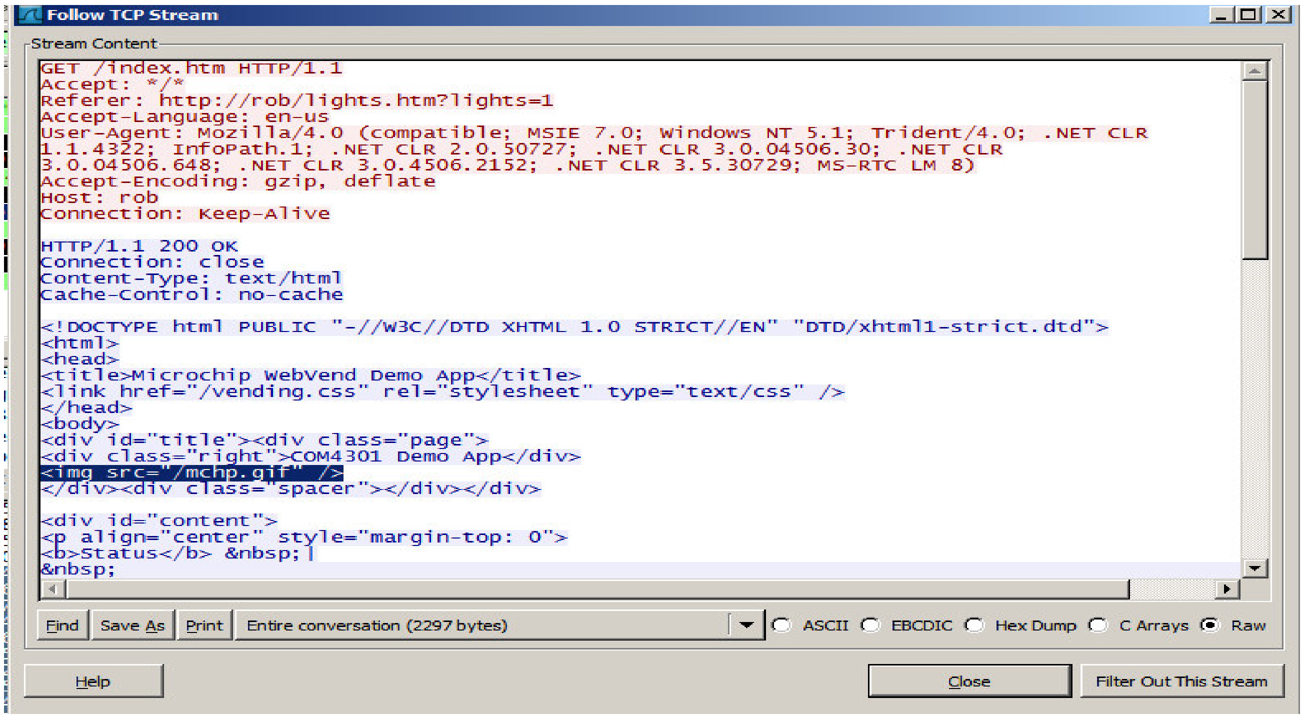
String Options: Case sensitive ☐
Character set: ASCII Unicode & Non-Unicode

Direction: Up, [✓ Down]

16 After finding the packet, ALTER THE TIME COLUMN PARAMETERS to suit your need for the timing calculation. Then USE FOLLOW THE TCP STREAM FEATURE. Application transmit performance can be calculated by: (**mchp.gif** file size)/(time elapsed for entire TCP transaction).

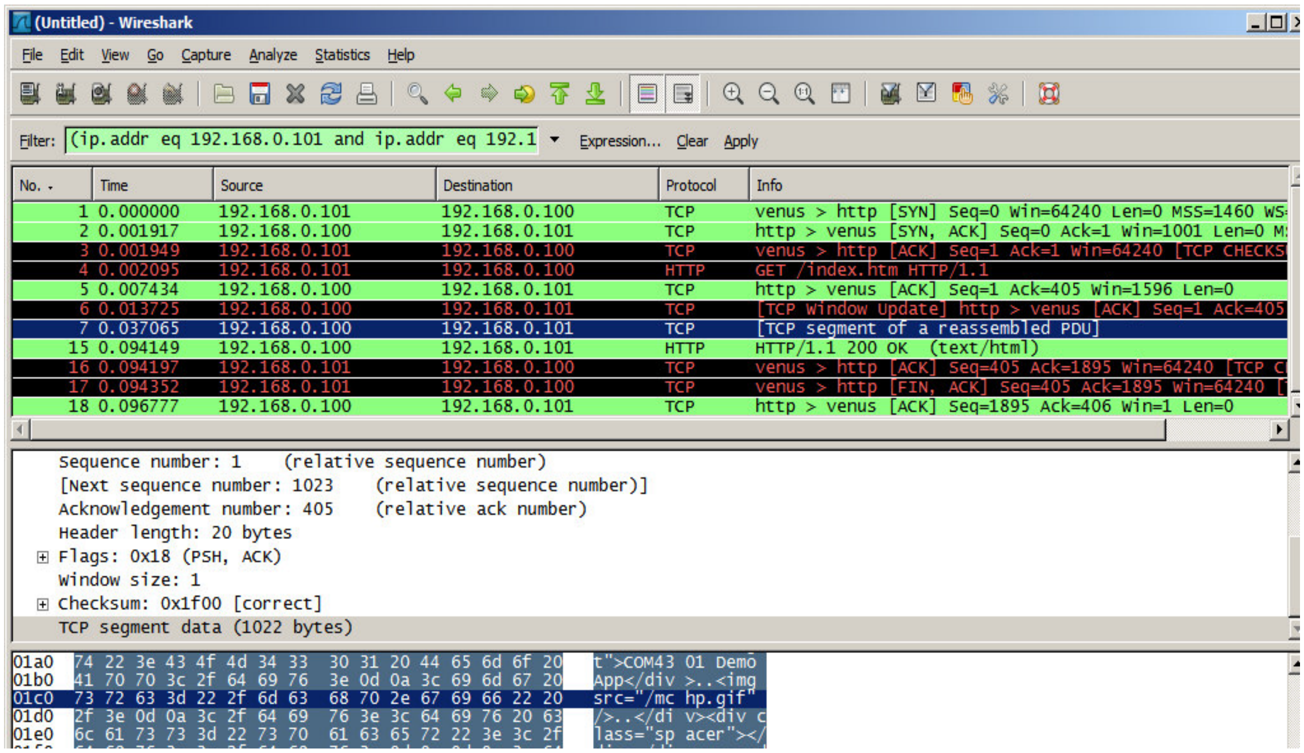
Wireshark Menu Altering

The *Time* column in Wireshark can be configured to display several different values (time since beginning of capture, time since last packet, etc.). Remember that right clicking on a packet in the packet pane will bring up a useful menu.



```
GET /index.htm HTTP/1.1
Accept: */*
Referer: http://rob/lights.htm?lights=1
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR
1.1.4322; InfoPath.1; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30; .NET CLR
3.0.04506.648; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; MS-RTC LM 8)
Accept-Encoding: gzip, deflate
Host: rob
Connection: Keep-Alive

HTTP/1.1 200 OK
Connection: close
Content-Type: text/html
Cache-Control: no-cache

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 STRICT//EN" "DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Microchip WebVend Demo App</title>
<link href="/vending.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div id="title"><div class="page">
<div class="right">COM4301 Demo App</div>
<img src="/mchp.gif" />
</div><div class="spacer"></div></div>

<div id="content">
<p align="center" style="margin-top: 0">
<b>Status</b> &nbsp;
&nbsp;
```

Entire conversation (2297 bytes) ASCII EBCDIC Hex Dump C Arrays [✓ Raw]


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Expression... Clear Apply

Filter: (ip.addr eq 192.168.0.101 and ip.addr eq 192.1

```
No. . Time      Source          Destination     Protocol  Info
    1 0.000000  192.168.0.101   192.168.0.100   TCP       venus > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS
    2 0.001917  192.168.0.100   192.168.0.101   TCP       http > venus [SYN, ACK] Seq=0 Ack=1 Win=1001 Len=0 M
    3 0.001949  192.168.0.101   192.168.0.100   TCP       venus > http [ACK] Seq=1 Ack=1 Win=64240 [TCP CHECKS
    4 0.002095  192.168.0.101   192.168.0.100   HTTP      GET /index.htm HTTP/1.1
    5 0.007434  192.168.0.100   192.168.0.101   TCP       http > venus [ACK] Seq=1 Ack=405 Win=1596 Len=0
    6 0.013725  192.168.0.100   192.168.0.101   TCP       [TCP Window Update] http > venus [ACK] Seq=1 Ack=405
    7 0.037065  192.168.0.100   192.168.0.101   TCP       [TCP segment of a reassembled PDU]
   15 0.094149  192.168.0.100   192.168.0.101   HTTP      HTTP/1.1 200 OK  (text/html)
   16 0.094197  192.168.0.101   192.168.0.100   TCP       venus > http [ACK] Seq=405 Ack=1895 Win=64240 [TCP C
   17 0.094352  192.168.0.101   192.168.0.100   TCP       venus > http [FIN, ACK] Seq=405 Ack=1895 Win=64240 [
   18 0.096777  192.168.0.100   192.168.0.101   TCP       http > venus [ACK] Seq=1895 Ack=406 Win=1 Len=0
```

```
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 1023    (relative sequence number)]
    Acknowledgement number: 405    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x18 (PSH, ACK)
    Window size: 1
  ⊞ Checksum: 0x1f00 [correct]
    TCP segment data (1022 bytes)
```

```
01a0  74 22 3e 43 4f 4d 34 33  30 31 20 44 65 6d 6f 20   t">COM43 01 Demo
01b0  41 70 70 3c 2f 64 69 76  3e 0d 0a 3c 69 6d 67 20   App</div >..<img
01c0  73 72 63 3d 22 2f 6d 63  68 70 2e 67 69 66 22 20   src="/mc hp.gif"
01d0  2f 3e 0d 0a 3c 2f 64 69  76 3e 3c 64 69 76 20 63   />..</di v><div c
01e0  6c 61 73 73 3d 22 73 70  61 63 65 72 22 3e 3c 2f   lass="sp acer"></
```

Bonus Code Analysis

Wireshark can be used for performance analysis by using the time column for calculations. Application transmit performance can be calculated by: (**application subset** file size)*(time elapsed for entire TCP transaction)/total application file size.

Lab Exercise 3

Integrating an Application with the Stack

Purpose

For many designs, you will be integrating an existing application with the stack. For new designs, it is still important to understand the fundamental design of the stack and where to place your application's code.

In this lab, you will take the example vending machine application demonstrated earlier and integrate it with the stack. Due to the complexity of the stack, it is generally easier to integrate your application into the stack rather than the other way around. This lab will walk you through those steps, and upon completion the vending machine and the stack will be operating separately within the device. The following labs will focus on them working cooperatively.

Requirements

Software:	**Completed Lab 1 to support network settings**
Environment:	MPLAB® IDE 8.60, X, or later, 1538 Masters Installer 1.536
C Compiler:	MPLAB C18 v3.38 or later, C30 v3.30 or later, or C32 v1.12 or later
H/W Tools:	PIC18F: PICDEM.net™ 2, or PIC24FFJ128GA010 PIM: Explorer 16 w/ 10Mbps, 10/100Mbps PICtail™ Plus or MRF24WB0 or PIC32MX360 or PIC32MX460 PIM: Explorer 16 & 10Mbps, 10/100Mbps PICtail™ Plus or MRF24WB0 AND MPLAB Real ICE™ , or MPLAB ICD 3 AND DHCP Enabled Router (wired or wireless)
Lab Files:	C:\Masters\1538\Lab3...

Objectives

- Experience the procedural steps for combing an existing firmware application with the TCP/IP stack
- Practice identifying the desirable locations for the firmware application within the TCPIP stack.

C:\Masters\1538\Lab3*

C:\Masters\1538\Vending Machine\VendingMachine.*

The Vending Machine application is implemented in the Vending Machine Project Directory above. The core of the application is implemented in two source files.

Procedures

1 MPLAB File ▶ Open Project

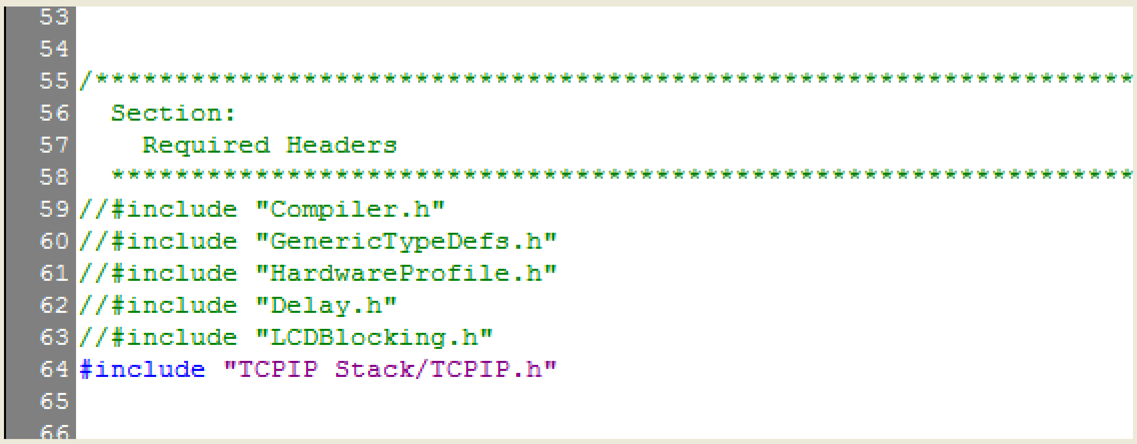
Lab3-C18-PICDN2-ETH97 for the PICDEM.net 2 with internal MAC/PHY setup
Lab3-C18-PICDN2-MRF24WB for the PICDEM.net 2 with Wireless MAC/PHY setup
Lab3-C30-EXP16-ENC28 for the Explorer 16 with PIC24F/dsPIC33F PIMs with 10Mbps module
Lab3-C30-EXP16-ENC624 for the Explorer 16 with PIC24F/dsPIC33F PIMs with 10/100Mbps module
Lab3-C30-EXP16-MRF24WB for the Explorer 16 with PIC24F/dsPIC33F PIMs with WIFI module
Lab3-C32-EXP16-ENC28 for the Explorer 16 with PIC32MX3xx/4xx PIMs with 10Mbps module
Lab3-C32-EXP16-ENC624 for the Explorer 16 with PIC32MX3xx/4xx PIMs with 10/100Mbps module
Lab3-C32-EXP16-MRF24WB for the Explorer 16 with PIC32MX3xx/4xx PIMs with WIFI module

2 Copy files `VendingMachine.c` and `VendingMachine.h` from the `Vending Machine` directory to the Lab 3 project directory. Use **Windows Explorer** or **My Computer**.

3 Add the two files you copied to your MPLAB project. You may need to select "All Source And Header Files" from the dropdown list for "Files of type" in order to see both the `.c` and `.h` file. **Project ▶ Add Files ▶** `VendingMachine.c` and `VendingMachine.h.`

4 Open `VendingMachine.h and` Comment out the section "Required Headers". Also Insert #include "TCPIP Stack/TCPIP.h" just below as shown. These headers and insertion integrate this file with TCPIP stack files. **Save** the changes you make. File ▶ Save

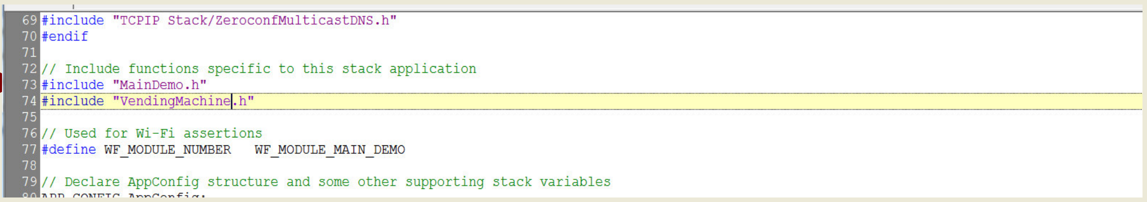
VendingMachine.h



```


/**********************************************************************
  Section:
    Required Headers
  ********************************************************************
//#include "Compiler.h"
//#include "GenericTypeDefs.h"
//#include "HardwareProfile.h"
//#include "Delay.h"
//#include "LCDBlocking.h"
#include "TCPIP Stack/TCPIP.h"


```

5 Open MainDemo.c and Insert #Include VendingMachine.h in so as to gain access to the vending machine application data. Place this directly below the directive for MainDemo.h. line.

MainDemo.c

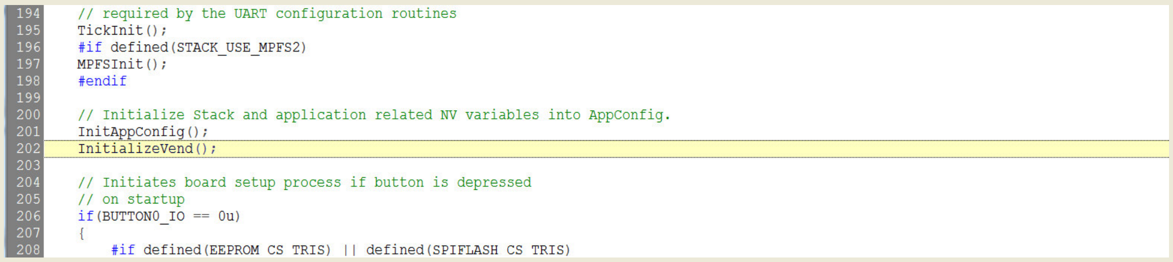


```
#include "TCPIP Stack/ZeroconfMulticastDNS.h"
#endif

// Include functions specific to this stack application
#include "MainDemo.h"
#include "VendingMachine.h"

// Used for Wi-Fi assertions
#define WF_MODULE_NUMBER   WF_MODULE_MAIN_DEMO

// Declare AppConfig structure and some other supporting stack variables
```

6 Insert a call to the vending machine's initialization function `InitializeVend()` in `main()`. This should be placed immediately following the call to `InitAppConfig()` on line 262.

MainDemo.c

6

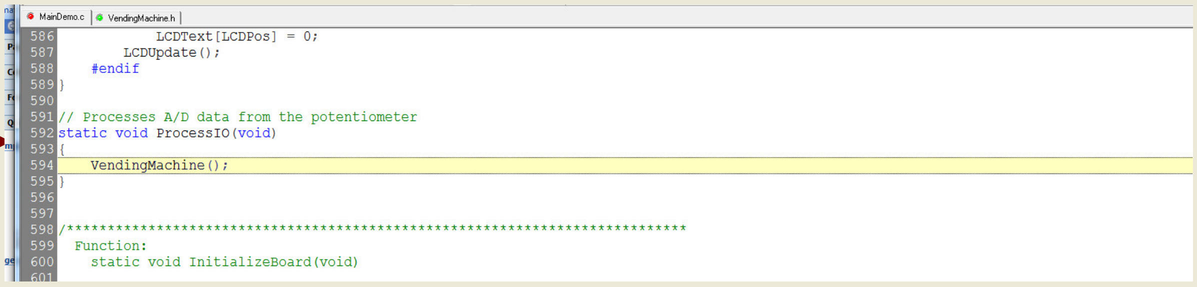


```
    // required by the UART configuration routines
    TickInit();
    #if defined(STACK_USE_MPFS2)
    MPFSInit();
    #endif

    // Initialize Stack and application related NV variables into AppConfig.
    InitAppConfig();
    InitializeVend();

    // Initiates board setup process if button is depressed
    // on startup
    if(BUTTON0_IO == 0u)
    {
        #if defined(EEPROM CS TRIS) || defined(SPIFLASH CS TRIS)
```

7 Replace the contents of `ProcessIO()` from line 644. Instead of performing an A/D conversion, call `VendingMachine()` to perform any necessary vending machine tasks.

MainDemo.c

7

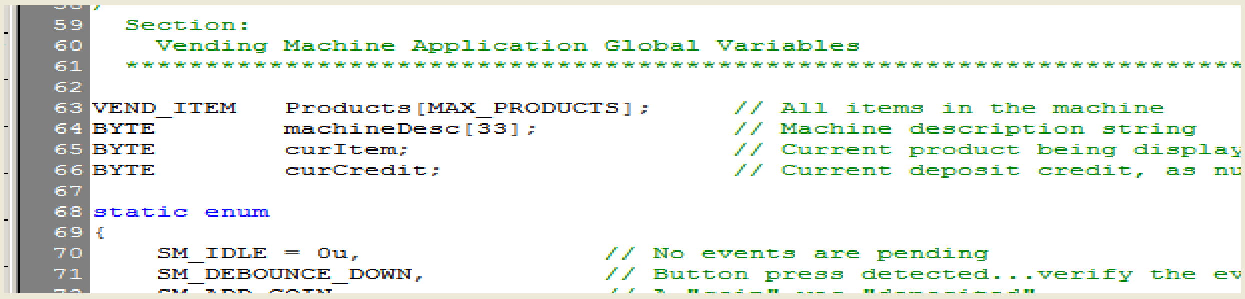


MainDemo.c | VendingMachine.h

```
            LCDText[LCDPos] = 0;
        LCDUpdate();
    #endif
}

// Processes A/D data from the potentiometer
static void ProcessIO(void)
{
    VendingMachine();
}


/*****************************************************************************
  Function:
    static void InitializeBoard(void)
```

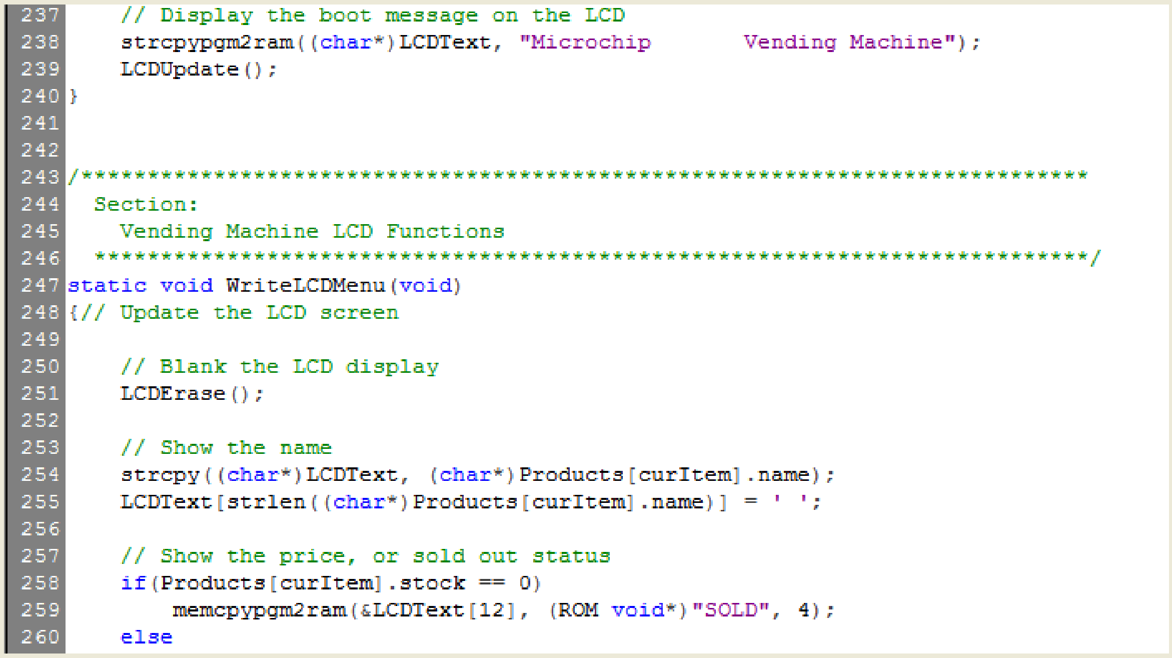
The Application Global Variables within Vending Machine.c declare the storage locations and define the state machine used by for the vending machine application. The LCD Functions section writes the various menus to the LCD display. Finally, the Initialization section configures the product names, prices, and stocks when the device boots.

VendingMachine.c-Global Variables



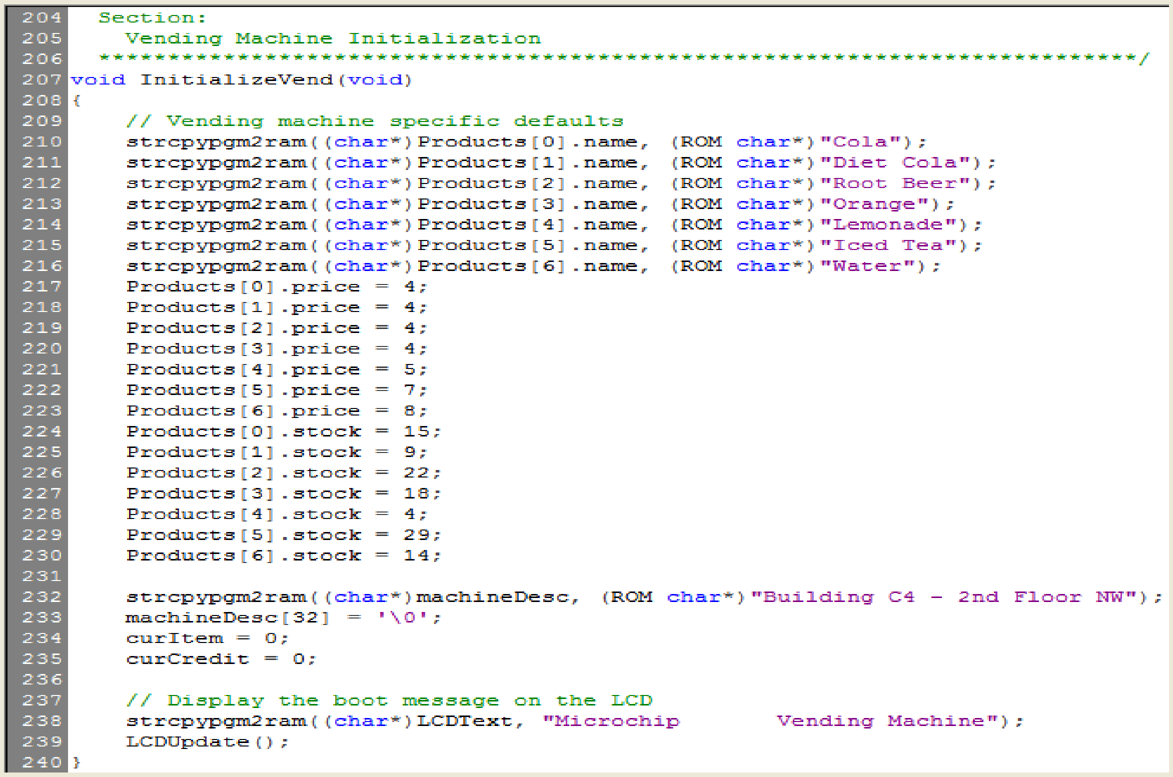
```
  Section:
    Vending Machine Application Global Variables
  ***********************************************************************

VEND_ITEM    Products[MAX_PRODUCTS];       // All items in the machine
BYTE         machineDesc[33];              // Machine description string
BYTE         curItem;                      // Current product being display
BYTE         curCredit;                    // Current deposit credit, as nu

static enum
{
    SM_IDLE = 0u,                    // No events are pending
    SM_DEBOUNCE_DOWN,                // Button press detected...verify the ev
```

VendingMachine.c: LCD writes



```
    // Display the boot message on the LCD
    strcpypgm2ram((char*)LCDText, "Microchip       Vending Machine");
    LCDUpdate();
}


/*********************************************************************
  Section:
    Vending Machine LCD Functions
  *********************************************************************/
static 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
    if(Products[curItem].stock == 0)
        memcpypgm2ram(&LCDText[12], (ROM void*)"SOLD", 4);
    else
```

VendingMachine.c: Initialization



```
  Section:
    Vending Machine Initialization
  *********************************************************************/
void InitializeVend(void)
{
    // Vending machine specific defaults
    strcpypgm2ram((char*)Products[0].name, (ROM char*)"Cola");
    strcpypgm2ram((char*)Products[1].name, (ROM char*)"Diet Cola");
    strcpypgm2ram((char*)Products[2].name, (ROM char*)"Root Beer");
    strcpypgm2ram((char*)Products[3].name, (ROM char*)"Orange");
    strcpypgm2ram((char*)Products[4].name, (ROM char*)"Lemonade");
    strcpypgm2ram((char*)Products[5].name, (ROM char*)"Iced Tea");
    strcpypgm2ram((char*)Products[6].name, (ROM char*)"Water");
    Products[0].price = 4;
    Products[1].price = 4;
    Products[2].price = 4;
    Products[3].price = 4;
    Products[4].price = 5;
    Products[5].price = 7;
    Products[6].price = 8;
    Products[0].stock = 15;
    Products[1].stock = 9;
    Products[2].stock = 22;
    Products[3].stock = 18;
    Products[4].stock = 4;
    Products[5].stock = 29;
    Products[6].stock = 14;

    strcpypgm2ram((char*)machineDesc, (ROM char*)"Building C4 - 2nd Floor NW");
    machineDesc[32] = '\0';
    curItem = 0;
    curCredit = 0;

    // Display the boot message on the LCD
    strcpypgm2ram((char*)LCDText, "Microchip       Vending Machine");
    LCDUpdate();
}
```

8 **SEE THE SECTION 2.1 OR 2.3 OF APPENDIX A FOR MPLABX BUILD AND/OR RUN OPTIONS.** Build and Program the firmware into the selected device in RELEASE mode.

BUILD TO CONFIRM NO SYNTAX ERRORS

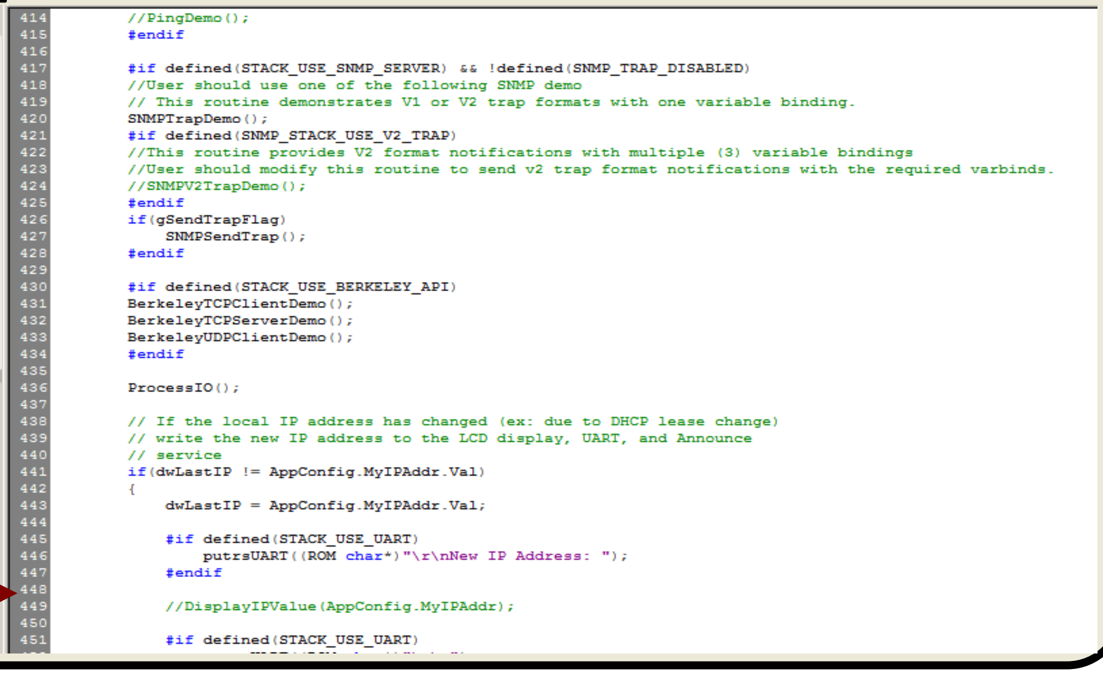
SEE SECTION 1.8 IN APPENDIX A FOR CONFIGURING YOUR PROGRAMMER

BUILD, PROGRAM AND RUN IN THE TARGET SYSTEM

9 Verify that the vending machine operates as discussed in the initial demo. Unexpected output on the LCD display will be corrected in the next step.
Verify that the web pages accessed during Lab 2 are still available.

10 In some versions of the stack, annoying messages are received on line 2 of the LCD display. If that occurs:
Comment out the call to `PingDemo()` in the main program loop to disable this output.
The "Ping timed out" message is generated by the ping demo when the right-most button is pressed.
Comment out the call to `DisplayIPValue()as it writes` the IP address to the LCD .
These deletions will **eliminate** the output of extraneous data to the LCD.

MainDemo.c


10
10

```
        //PingDemo();
        #endif

        #if defined(STACK_USE_SNMP_SERVER) && !defined(SNMP_TRAP_DISABLED)
        //User should use one of the following SNMP demo
        // This routine demonstrates V1 or V2 trap formats with one variable binding.
        SNMPTrapDemo();
        #if defined(SNMP_STACK_USE_V2_TRAP)
        //This routine provides V2 format notifications with multiple (3) variable bindings
        //User should modify this routine to send v2 trap format notifications with the required varbinds.
        //SNMPV2TrapDemo();
        #endif
        if(gSendTrapFlag)
            SNMPSendTrap();
        #endif

        #if defined(STACK_USE_BERKELEY_API)
        BerkeleyTCPClientDemo();
        BerkeleyTCPServerDemo();
        BerkeleyUDPClientDemo();
        #endif

        ProcessIO();

        // If the local IP address has changed (ex: due to DHCP lease change)
        // write the new IP address to the LCD display, UART, and Announce
        // service
        if(dwLastIP != AppConfig.MyIPAddr.Val)
        {
            dwLastIP = AppConfig.MyIPAddr.Val;

            #if defined(STACK_USE_UART)
                putrsUART((ROM char*)"\r\nNew IP Address: ");
            #endif

            //DisplayIPValue(AppConfig.MyIPAddr);

            #if defined(STACK_USE_UART)
```

11 **SEE THE SECTION 2.1 OR 2.3 OF APPENDIX A FOR MPLABX BUILD AND/OR RUN OPTIONS.** Build and Program the firmware into the selected device in RELEASE mode.

BUILD TO CONFIRM NO SYNTAX ERRORS

SEE SECTION 1.8 IN APPENDIX A FOR CONFIGURING YOUR PROGRAMMER

BUILD, PROGRAM AND RUN IN THE TARGET SYSTEM

12 Confirm the Router assigns an new IP Address.
Verify that the undesired messages no longer appear on the LCD screen.

Results

Your project now has a vending machine operating simultaneously with the TCP/IP Stack. While the two functions do not interact with each other yet, the basic functionality is working

Code Analysis

The vending machine application was implemented as a state machine, which is common for many simple microcontroller-based systems. Since each state executes for only a brief period, it was relatively easy to integrate into the stack. However, the general model and code placement applies for more complex applications as well.
In general, initialization code should be called after the InitAppConfig() call. This code will be executed once when your device boots. Application code should be placed either at the end of the main stack loop or in the `ProcessIO()` function.

Conclusions

Although no new functionality has been gained, the vending machine is now much closer to being network-enabled. Integration with the stack is the most challenging task because it requires (in many cases) a new way of thinking about the application. A few integration issues still remain, which will be explored in the next lab.

Lab Exercise 4

Remove Blocking Code

Purpose

Networked applications must handle a significant amount of background processing in order to keep the network connection alive. To accomplish these time-sensitive events, the Microchip TCP/IP Stack operates in a co-operative multi-ta sking fashion. This eliminates the need for an RTOS, as well as the need for interrupts and overhead.
However, it means that your application must be implemented without any code that may block the processor. In this lab, you will witness how a blocking loop can affect the TCP/IP Stack, and will learn how to take steps to avoid such loops in your application.

Requirements

Environment:	MPLAB® IDE 8.60, X, or later, 1538 Masters Installer 1.536
C Compiler:	MPLAB C18 v3.38 or later, C30 v3.30 or later, or C32 v1.12 or later
H/W Tools:	PIC18F: PICDEM.net™ 2, or
	PIC24FFJ128GA010 PIM: Explorer 16 w/ 10Mbps, 10/100Mbps PICtail™ Plus or MRF24WB0 or
	PIC32MX360 or PIC32MX460 PIM: Explorer 16 & 10Mbps, 10/100Mbps PICtail™ Plus or MRF24WB0
	AND MPLAB Real ICE™ , or MPLAB ICD 3
	AND DHCP Enabled Router (wired or wireless)
Lab Files:	C:\Masters\1538\Lab4..

Objectives

- Identify non-conforming application code that can disrupt the required operational flow of the TCP/IP stack
- Practice making application code modifications to ensure compatible coexistence of two code sets

Procedures

0 Confirm the operation of the blinking LED per the description below:.
With your application from Lab 3 running, notice the blinking status LED on your development board. What happens to that light when a message change is displayed on the LCD? (The effect is most visible when you attempt to insert more than $5.00 of credit into the machine.) Stable blinking operation of this LED signifies that the TCP/IP Stack is being called frequently. When the status light is not blinking, the stack is not running and cannot respond to incoming network traffic. The existing vending machine code uses a blocking loop to time the display message. This needs to be corrected.

C:\Masters\1538\Lab4*

1 MPLAB File ▸ Open Project

Lab4-C18-PICDN2-ETH97.X for the PICDEM.net 2 with internal MAC/PHY setup
Lab4-C18-PICDN2-MRF24WB.X for the PICDEM.net 2 with Wireless MAC/PHY setup
Lab4-C30-EXP16-ENC28.X for the Explorer 16 with PIC24F/dsPIC33F PIMs with 10Mbps module
Lab4-C30-EXP16-ENC624.X for the Explorer 16 with PIC24F/dsPIC33F PIMs with 10/100Mbps module
Lab4-C30-EXP16-MRF24WB.X for the Explorer 16 with PIC24F/dsPIC33F PIMs with WIFI module
Lab4-C32-EXP16-ENC28.X for the Explorer 16 with PIC32MX3xx/4xx PIMs with 10Mbps module
Lab4-C32-EXP16-ENC624.X for the Explorer 16 with PIC32MX3xx/4xx PIMs with 10/100Mbps module
Lab4-C32-EXP16-MRF24WB.X for the Explorer 16 with PIC32MX3xx/4xx PIMs with WIFI module

2 Open the vending machine source file `VendingMachine.c`. Locate the blocking FOR loop in the `SM_DISPLAY_WAIT` state inside of `VendingMachine()`. This loop consumes processing cycles by performing a meaningless task. Mark this loop as commented out as shown in the example below.

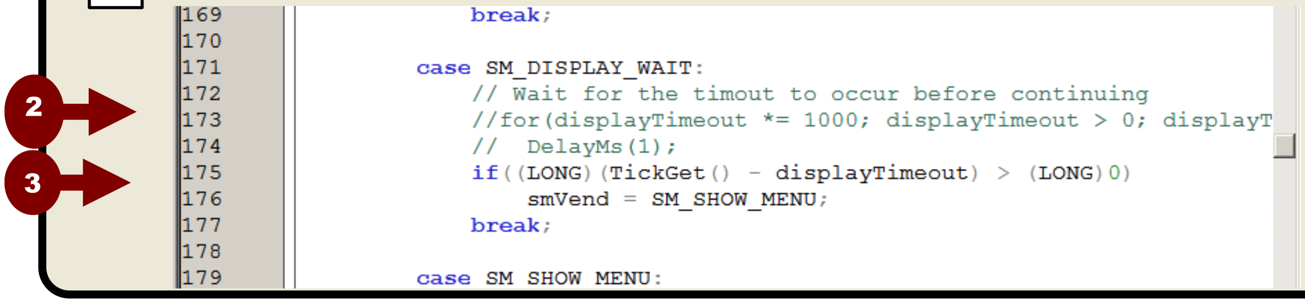
About Timing Loops

A common method for implementing delays is to cause the processor to complete some meaningless task, such as counting to 10,000, or to call one of the `Delay()` functions that does this internally. However, code constructs such as these block the processor and waste MCU resources that could otherwise be spent handling incoming Ethernet packets or other network functions. Loops of this type should be avoided.

Instead, use the provided Tick module. The Tick module is interrupt-driven, and is based on the hardware clock. It is stable and accurate, and can be used to implement non-blocking delays by comparing current times to timeouts. Since it is hardware-based, it is also suitable for use as a Real Time Clock.

3 Insert a comparison of `TICK` values to determine if enough time has elapsed. If the timeout has occurred, advance the state machine. Otherwise, just `break` and return to the main stack application. The stack will call the vending machine to perform the comparison again later. Use the example shown below.

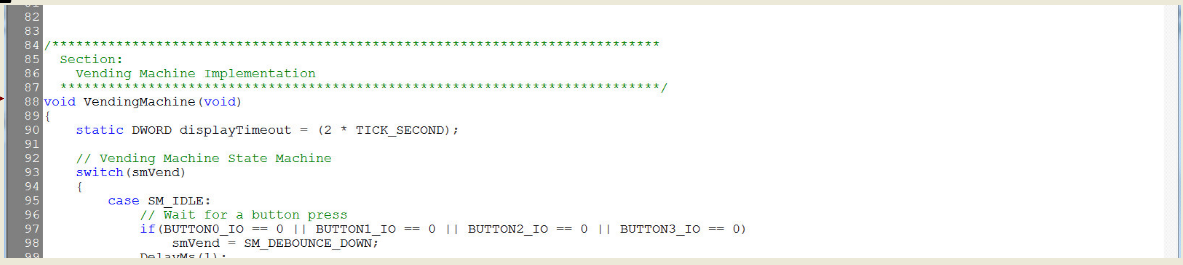
VendingMachine.c



```
            break;

        case SM_DISPLAY_WAIT:
            // Wait for the timout to occur before continuing
            //for(displayTimeout *= 1000; displayTimeout > 0; displayT
            //   DelayMs(1);
            if((LONG)(TickGet() - displayTimeout) > (LONG)0)
                smVend = SM_SHOW_MENU;
            break;

        case SM_SHOW_MENU:
```

4 Change the declaration of this value at the top of `VendingMachine()` to its initialization to indicate two seconds rather than just the number `2`. Use the macro `TICK_SECOND` to calculate this value. Your code should match the sample below.

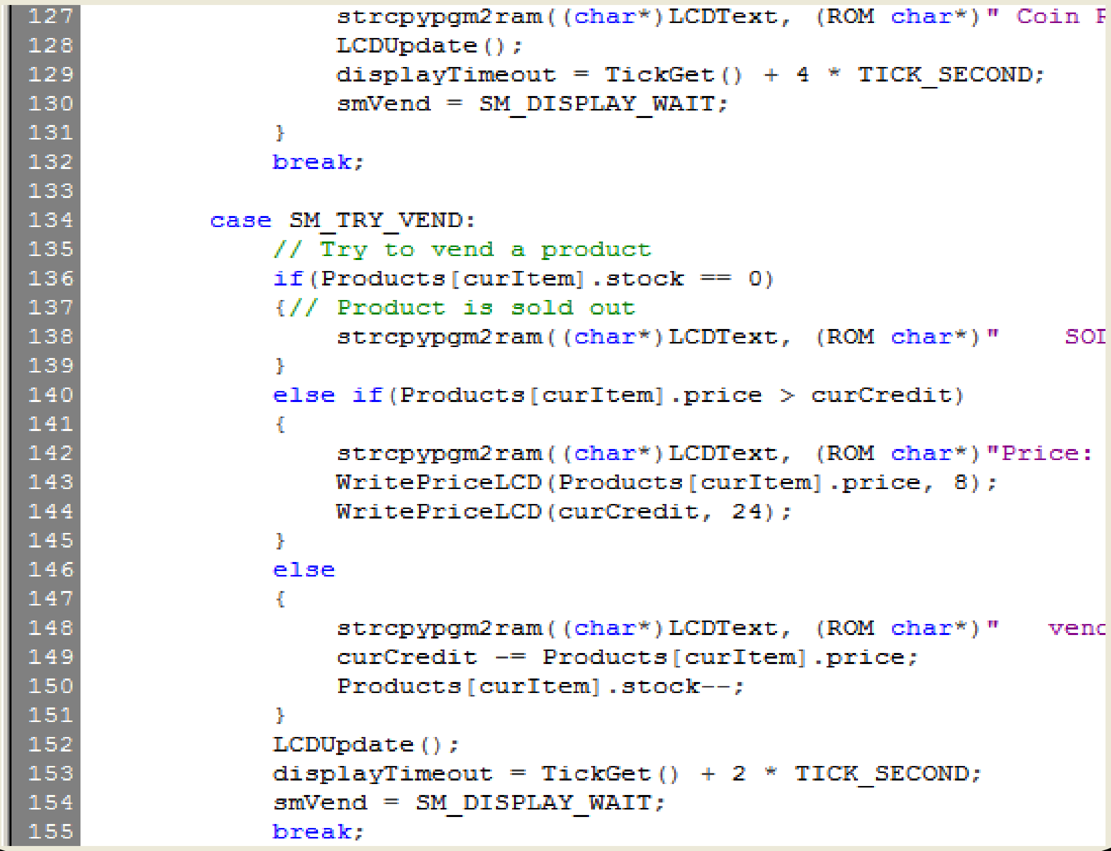
VendingMachine.c



```
/***************************************************************************
  Section:
    Vending Machine Implementation
  ***************************************************************************/
void VendingMachine(void)
{
    static DWORD displayTimeout = (2 * TICK_SECOND);

    // Vending Machine State Machine
    switch(smVend)
    {
        case SM_IDLE:
            // Wait for a button press
            if(BUTTON0_IO == 0 || BUTTON1_IO == 0 || BUTTON2_IO == 0 || BUTTON3_IO == 0)
                smVend = SM_DEBOUNCE_DOWN;
            DelayMs(1);
```

5 Change the two instances where `displayTimeout` is set. Use a relative time based on `TickGet()` and `TICK_SECOND` as indicated below. The variable `displayTimeout` is now allocated as a `TICK` and configured to expire on a relative timeout value. The last step in this conversion is to replace the code the assigns absolute delays to `displayTimeout` with an assignment relative to the current `TICK` value.

VendingMachine.c



```
                strcpypgm2ram((char*)LCDText, (ROM char*)" Coin P
                LCDUpdate();
                displayTimeout = TickGet() + 4 * TICK_SECOND;
                smVend = SM_DISPLAY_WAIT;
            }
            break;

        case SM_TRY_VEND:
            // Try to vend a product
            if(Products[curItem].stock == 0)
            {// Product is sold out
                strcpypgm2ram((char*)LCDText, (ROM char*)"     SOI
            }
            else if(Products[curItem].price > curCredit)
            {
                strcpypgm2ram((char*)LCDText, (ROM char*)"Price:
                WritePriceLCD(Products[curItem].price, 8);
                WritePriceLCD(curCredit, 24);
            }
            else
            {
                strcpypgm2ram((char*)LCDText, (ROM char*)"   venc
                curCredit -= Products[curItem].price;
                Products[curItem].stock--;
            }
            LCDUpdate();
            displayTimeout = TickGet() + 2 * TICK_SECOND;
            smVend = SM_DISPLAY_WAIT;
            break;
```

6

SEE THE SECTION 2.1 OR 2.3 OF APPENDIX A FOR MPLABX BUILD AND/OR RUN OPTIONS. Build and Program the firmware into the selected device in RELEASE mode.

BUILD TO CONFIRM NO SYNTAX ERRORS

SEE SECTION 1.8 IN APPENDIX A FOR CONFIGURING YOUR PROGRAMMER

BUILD, PROGRAM AND RUN IN THE TARGET SYSTEM

7

Verify that the vending machine still functions.

1. Ensure that your web pages from Lab 2 are still accessible via a browser.
2. Verify that the status LED continues to blink, even when a status message is displayed on the screen.
   - i. Test when vending is attempted with insufficient credit.
   - ii. Test when a coin is returned for depositing over $5.00.
   - iii. Test when a product is vended.
   - iv. Test when a product is sold out.

To use the tick module, replace the blocking call by checking the current time using TickGet. Then add an additional state to wait for the timeout. Transition to this state and return to the main loop. When the loop returns to your application, compare the stored value with the current time. You can use simple subtraction, since Ticks are stored as signed integers and will automatically overflow. The TICK_SECOND macro is based on the instruction frequency, and can be used for comparisons against elapsed time. You can also multiply or divide this value to obtain various resolutions. Tick is an unsigned value. Type-casting to LONG forces the result to be a signed value, which will automatically take care of overflow. Assume a TICK is only one byte. Say TickGet is equal to 253, and TICK_SECOND/10 is actually 10. That means that “doneAt” will be set to 7 after overflow. When TickGet returns 255, you’ll have 255 – 7. That’s 248, or 1111 1000b. Cast that to a LONG and the first becomes a sign bit. That binary value equates to -8, and so -8 < 0 and your timeout has not yet happened. When TickGet rolls over and returns 3, you’ll have 3 – 7. You’ll get -4 after the cast, and your timeout will fail. Once TickGet has incremented 10 times you’ll be at 8. You’ll have 8 – 7, which will be greater than zero and your timeout will succeed. This works in many other cases, but you may have to pull out some old binary arithmetic rules to prove it to yourself.

Results

When the lab is completed correctly, the status LED will continue to blink without interruption while messages are being displayed on the LCD. This signifies that application is not blocking the processor and that the stack can complete its tasks in a timely fashion.

Code Analysis

The examples provided in this lab implement a timer without blocking loops. A single state is used for the timeout stage, and the various machine states can set when a timeout will occur. Timeouts are stored in `displayTimeout`, which is declared as a `TICK` value. The timeout value is calculated as the current time (obtained via `TickGet()`), plus some multiple of `TICK_SECOND`. This method is suitable for measuring time differences from a few microseconds to a few hours. (It's companion methods `TickGetDiv256()` and `TickGetDiv64K()` are valid for a few weeks or several years, respectively.) The value of `TICK_SECOND` is calculated from the value of `GetInstructionClock()` as specified in `HardwareProfile.h`. `TICK` values are stored as 32-bit unsigned integers. When a time increment is added (such as `2*TICK_SECOND`), this could result in overflow. Similarly underflow may result when the subtraction is performed. The subtraction of `TICK` values, when cast to a `LONG`, compensates for this by reestablishing the sign bit. This allows for valid comparisons in all cases.

Bonus Procedure

8 Several calls to `DelayMs()` remain in the code to help de-bounce the pushbuttons. Since the stack loop now executes between each state, see if any of these can be removed without affecting operation. This will be dependent on your platform, as the faster processors may still loop too quickly to avoid the transient periods.

Conclusion

The vending machine is now completely integrated, and neither task will interfere with the other. With this complete, you are now ready to being adding network features to the device.
The principles learned in these past two labs apply to all stack applications:

1. Break long tasks into states, allowing the stack to execute frequently.
2. Use the Tick module instead of blocking loops for timed events.

The co-operative multi-tasking methodology of the stack is flexible, and will consume as little or as much processor time as the application supplies. If necessary, stack operation can be interrupted by the application for a few hundred milliseconds up to even a few seconds. While this is not desirable (and should not be a frequent occurrence), most network application protocols will not time out until several seconds have elapsed without communication. However, the application will risk overrunning its RX buffer and losing data during this time. As a general rule, applications should attempt to run the main stack loop at least every 10 or 20ms, with 1-2ms being the target for best performance.

Lab Exercise 5

Web-based Monitoring with Dynamic Variables

Purpose

The first network feature to be implemented for this vending machine is the ability to determine its stock remotely. Having already decided upon a web page for monitoring, the next step is to link the vending machine data into a web page. Example web pages have already been provided for this project. However, they are just static pages with no mechanism to populate vending machine data into the page. In this lab, you will program the first page to display the product name and remaining stock level as reported by the vending machine.

Requirements

Environment:	MPLAB® IDE 8.60 , X, or later, MPFS Utility, 1538 Masters Installer 1.536
C Compiler:	MPLAB C18 v3.38 or later, C30 v3.30 or later, or C32 v1.12 or later
H/W Tools:	PIC18F: PICDEM.net™ 2, or
	PIC24FFJ128GA010 PIM: Explorer 16 w/ 10Mbps, 10/100Mbps PICtail™ Plus or MRF24WB0 or
	PIC32MX360 or PIC32MX460 PIM: Explorer 16 & 10Mbps, 10/100Mbps PICtail™ Plus or MRF24WB0
	AND MPLAB Real ICE™ , MPLAB ICD3 or MPLAB ICD 2
	AND DHCP Enabled Router (wired or wireless)
Lab Files:	C:\Masters\1538\Lab5....

Objectives

- Implement dynamic replacement of static variables within HTML files for web server applications
- Utilize Callback procedure methodology between the HTTP2 module and customer application firmware

Web Page Directory:
C:\Masters\1538\Lab5\WebPages2

Procedures

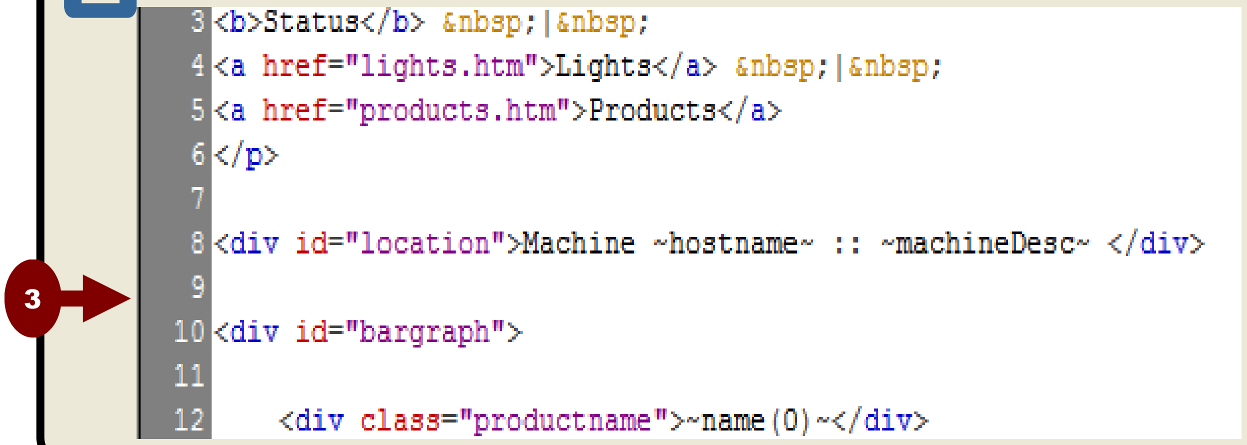
1 Confirm design files in the `WebPages2` folder in Lab 5 match the `Sample Web Pages` in C:\Masters\1538 folder
.

2 Open `index.htm` from the `WebPages2` directory in **Crimson Editor**.
Start ▸ Programs ▸ Crimson Editor ▸ Crimson Editor

3

Replace **HOSTNAME** static text with a dynamic variable called **~hostname~**.
Replace **Machine Location / Description** text to **~machineDesc~**.
Save your changes to the HTML code.

index.htm



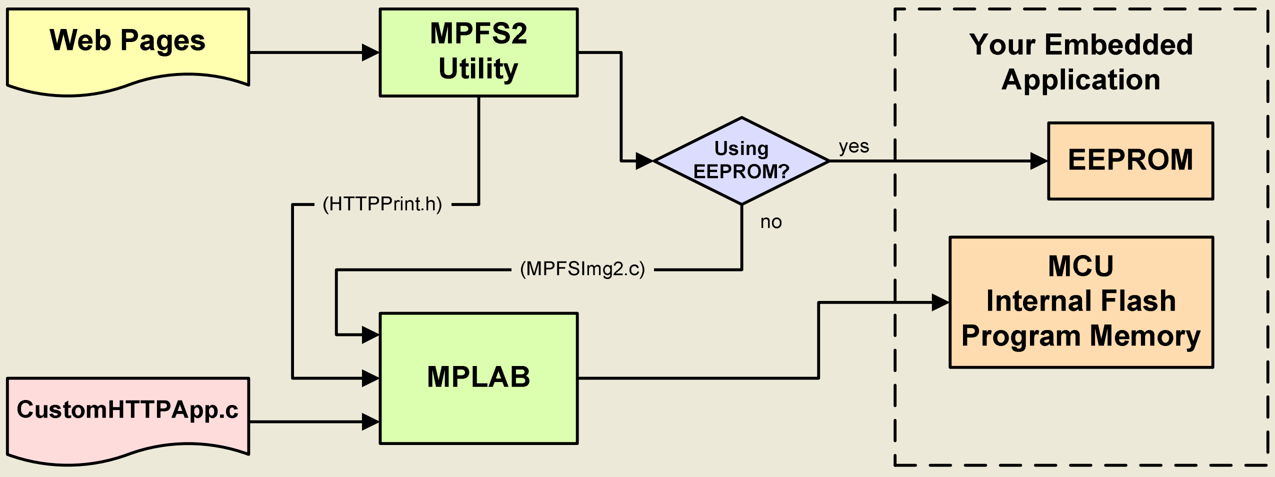
```
<b>Status</b> &nbsp;|&nbsp;
<a href="lights.htm">Lights</a> &nbsp;|&nbsp;
<a href="products.htm">Products</a>
</p>

<div id="location">Machine ~hostname~ :: ~machineDesc~ </div>

<div id="bargraph">

    <div class="productname">~name(0)~</div>
```

4

Bundle and **upload** this modified page to the development board with MPFS utility.
Recall that the **MPFS2 Utility** generates `HTTPPrint.h`, which must be compiled into your MPLAB project prior to compiling with your dynamic variable callback functions in step 5 herein.
Start ▸ Programs ▸ Microchip ▸ 1538 ▸ MPFS2
No modifications to the utility configuration should be necessary. Click **Generate and Upload** to program your new pages to the development board.

How Web Pages are Bundled

Recall: The MPFS2 utility must be executed prior to compiling your project in MPLAB.



C:\Masters\1538\Lab5\*

5 MPLAB File ▶ Open Project

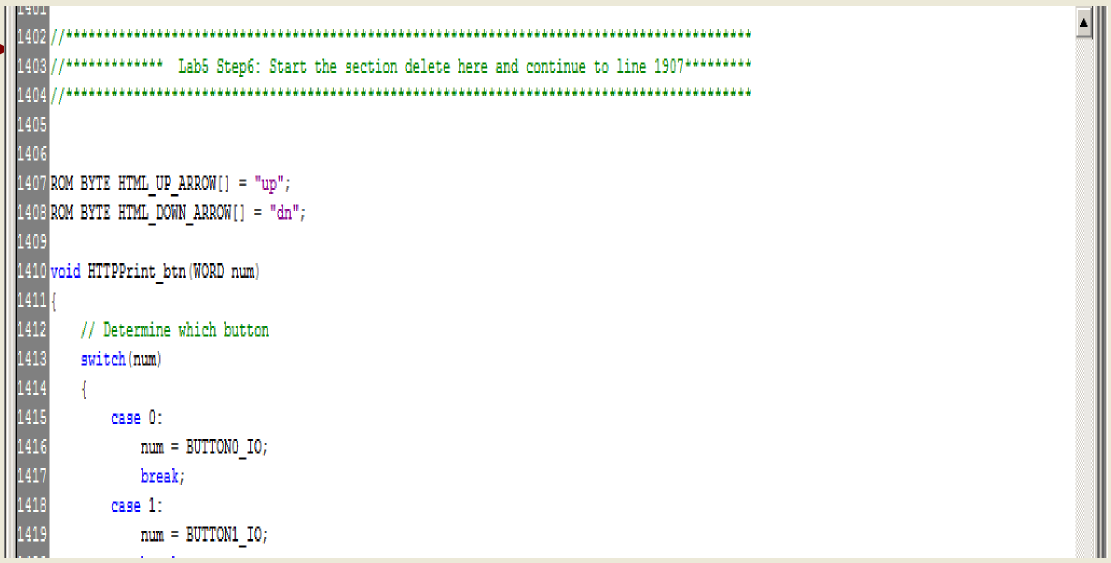
Lab5-C18-PICDN2-ETH97.X for the PICDEM.net 2 with internal MAC/PHY setup
Lab5-C18-PICDN2-MRF24WB.X for the PICDEM.net 2 with Wireless MAC/PHY setup
Lab5-C30-EXP16-ENC28.X for the Explorer 16 with PIC24F/dsPIC33F PIMs with 10Mbps module
Lab5-C30-EXP16-ENC624.X for the Explorer 16 with PIC24F/dsPIC33F PIMs with 10/100Mbps module
Lab5-C30-EXP16-MRF24WB.X for the Explorer 16 with PIC24F/dsPIC33F PIMs with WIFI module
Lab5-C32-EXP16-ENC28.X for the Explorer 16 with PIC32MX3xx/4xx PIMs with 10Mbps module
Lab5-C32-EXP16-ENC624.X for the Explorer 16 with PIC32MX3xx/4xx PIMs with 10/100Mbps module
Lab5-C32-EXP16-MRF24WB.X for the Explorer 16 with PIC32MX3xx/4xx PIMs with WIFI module

Open `CustomHTTPApp.c` within MPLAB's project window.

6 Remove all the `HTTPPrint_*` functions within the "Dynamic Variable Callback Functions" section (These callbacks are for the demo application, most of which are no longer needed.) except for `HTTPPrint_version()` and `HTTPPrint_builddate()`.

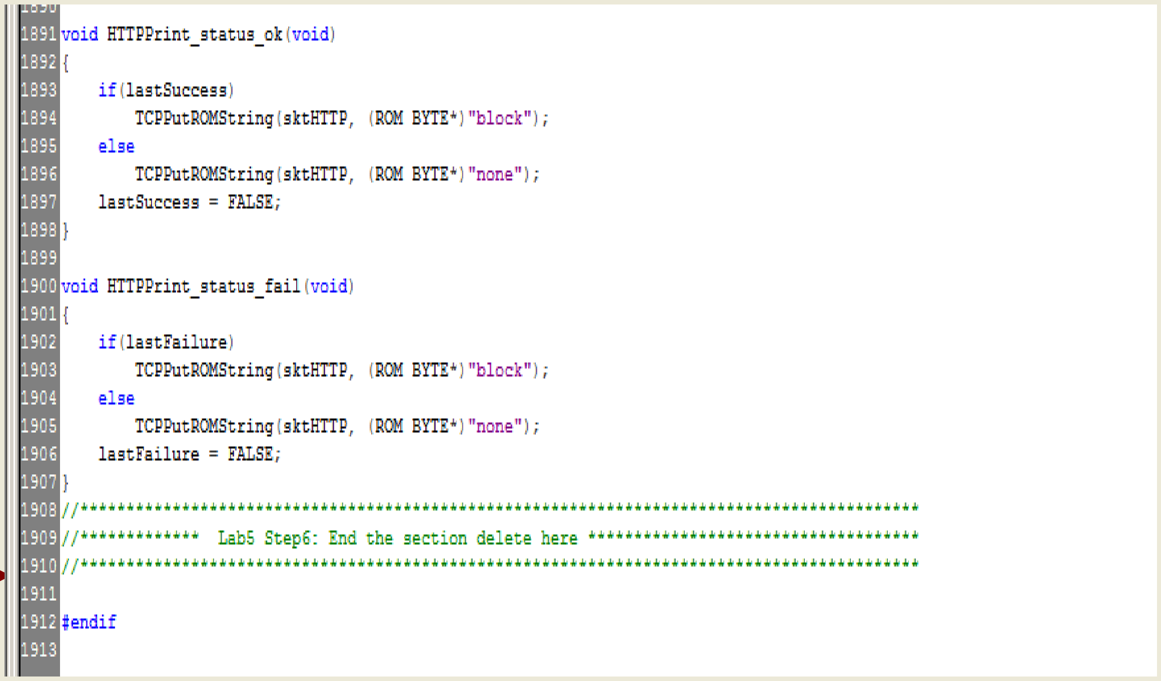
CustomHTTPApp.c (delete unused sections)

Continue deleting thru the lines on the next page



```
1402 //****************************************************************************************
1403 //************  Lab5 Step6: Start the section delete here and continue to line 1907*********
1404 //****************************************************************************************
1405
1406
1407 ROM BYTE HTML_UP_ARROW[] = "up";
1408 ROM BYTE HTML_DOWN_ARROW[] = "dn";
1409
1410 void HTTPPrint_btn(WORD num)
1411 {
1412     // Determine which button
1413     switch(num)
1414     {
1415         case 0:
1416             num = BUTTON0_IO;
1417             break;
1418         case 1:
1419             num = BUTTON1_IO;
```

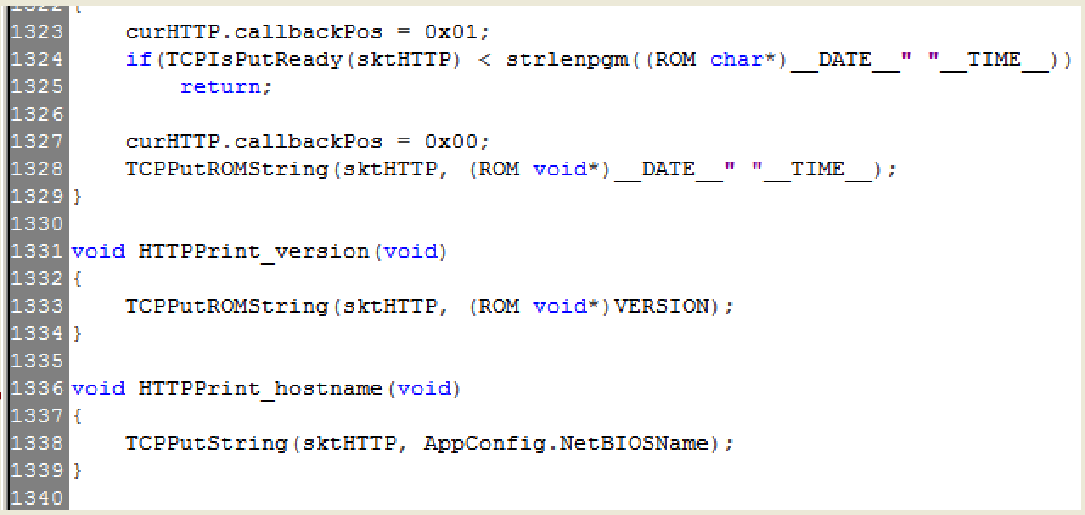
CustomHTTPApp.c (delete unused sections)



```c
void HTTPPrint_status_ok(void)
{
    if(lastSuccess)
        TCPPutROMString(sktHTTP, (ROM BYTE*)"block");
    else
        TCPPutROMString(sktHTTP, (ROM BYTE*)"none");
    lastSuccess = FALSE;
}

void HTTPPrint_status_fail(void)
{
    if(lastFailure)
        TCPPutROMString(sktHTTP, (ROM BYTE*)"block");
    else
        TCPPutROMString(sktHTTP, (ROM BYTE*)"none");
    lastFailure = FALSE;
}
//***************************************************************************************
//*************  Lab5 Step6: End the section delete here **********************************
//***************************************************************************************

#endif
```

7 Insert a callback function named `HTTPPrint_hostname()` to this section of the file. This function has no parameters and returns no value. Within the function, use `TCPPutString()` to print the hostname from its RAM location, `AppConfig.NetBIOSName`, to the TCP socket, `sktHTTP`. For more information about callbacks, refer to the info box labeled on the topic below..

CustomHTTPApp.c



```c
    curHTTP.callbackPos = 0x01;
    if(TCPIsPutReady(sktHTTP) < strlenpgm((ROM char*)__DATE__" "__TIME__))
        return;

    curHTTP.callbackPos = 0x00;
    TCPPutROMString(sktHTTP, (ROM void*)__DATE__" "__TIME__);
}

void HTTPPrint_version(void)
{
    TCPPutROMString(sktHTTP, (ROM void*)VERSION);
}

void HTTPPrint_hostname(void)
{
    TCPPutString(sktHTTP, AppConfig.NetBIOSName);
}
```

About Callback Functions

Callback functions are a concept of event-driven programming. For the HTTP2 web server, an event occurs when a dynamic variable is encountered in your HTML code. The response, handled by your callback function, is to print text or data to the web browser.

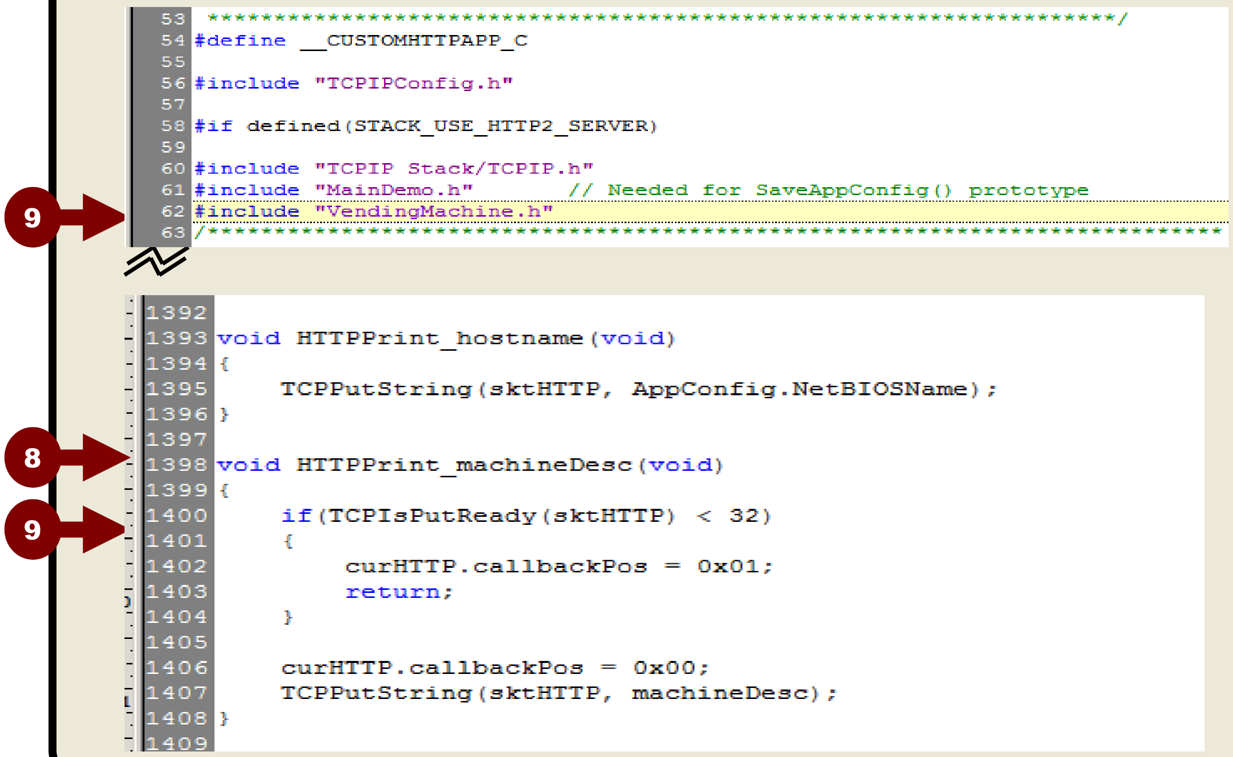
You do not need to worry about how or when the callback is invoked. The HTTP2 server, along with the `HTTPPrint.h` file generated by the MPFS2 Utility, take care of this for you.

8 Add a second callback function named `HTTPPrint_machineDesc()` to this section of the file. This function also has no parameters and returns no value. Adding this function is slightly more involved.

9 StepA, Insert a #include `VendingMachine.h` entry at the top of the file right after the existing #include MainDemo.h entry. This provides a reference to the `machineDesc` string.

StepB, Insert an if loop that uses `TCPIsPutReady() to confirm buffer space.` The machineDesc string may be up to 32 bytes long, which is longer than the 16 bytes the HTTP2 web server guarantees are available. So, the HTTPPrint call must follow the ensuring of at least 32 bytes are free in the TCP buffer. If the buffer is too small, set a non-zero flag in `curHTTP.callbackPos` and return, which will request to be called again when more space is available.

CustomHTTPApp.c



```
 ***********************************************************************/
#define __CUSTOMHTTPAPP_C

#include "TCPIPConfig.h"

#if defined(STACK_USE_HTTP2_SERVER)

#include "TCPIP Stack/TCPIP.h"
#include "MainDemo.h"        // Needed for SaveAppConfig() prototype
#include "VendingMachine.h"
/*********************************************************************************
```

```

void HTTPPrint_hostname(void)
{
    TCPPutString(sktHTTP, AppConfig.NetBIOSName);
}

void HTTPPrint_machineDesc(void)
{
    if(TCPIsPutReady(sktHTTP) < 32)
    {
        curHTTP.callbackPos = 0x01;
        return;
    }

    curHTTP.callbackPos = 0x00;
    TCPPutString(sktHTTP, machineDesc);
}

```

10

SEE THE SECTION 2.1 OR 2.3 OF APPENDIX A FOR MPLABX BUILD AND/OR RUN OPTIONS. Build and Program the firmware into the selected device in RELEASE mode.

BUILD TO CONFIRM NO SYNTAX ERRORS

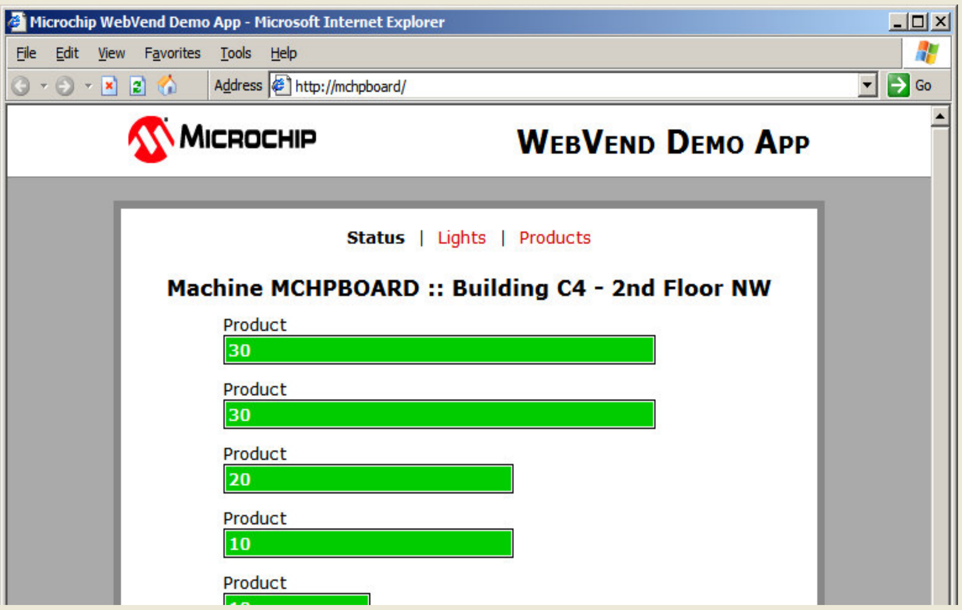
SEE SECTION 1.8 IN APPENDIX A FOR CONFIGURING YOUR PROGRAMMER

BUILD, PROGRAM AND RUN IN THE TARGET SYSTEM

11

Confirm the Router assigns an new IP Address
Verify that the vending machine still operates as discussed in the initial demo.
Verify that the web pages have the dynamic variables have replaced the static values, specifically hostname and location. The string for the location is set in `VendingMachine.c`, and can be modified there if desired.

Sample Web Page


Microchip WebVend Demo App - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://mchpboard/ Go
MICROCHIP
WEBVEND DEMO APP
Status | Lights | Products

Machine MCHPBOARD :: Building C4 - 2nd Floor NW

Product
30

Product
30

Product
20

Product
10

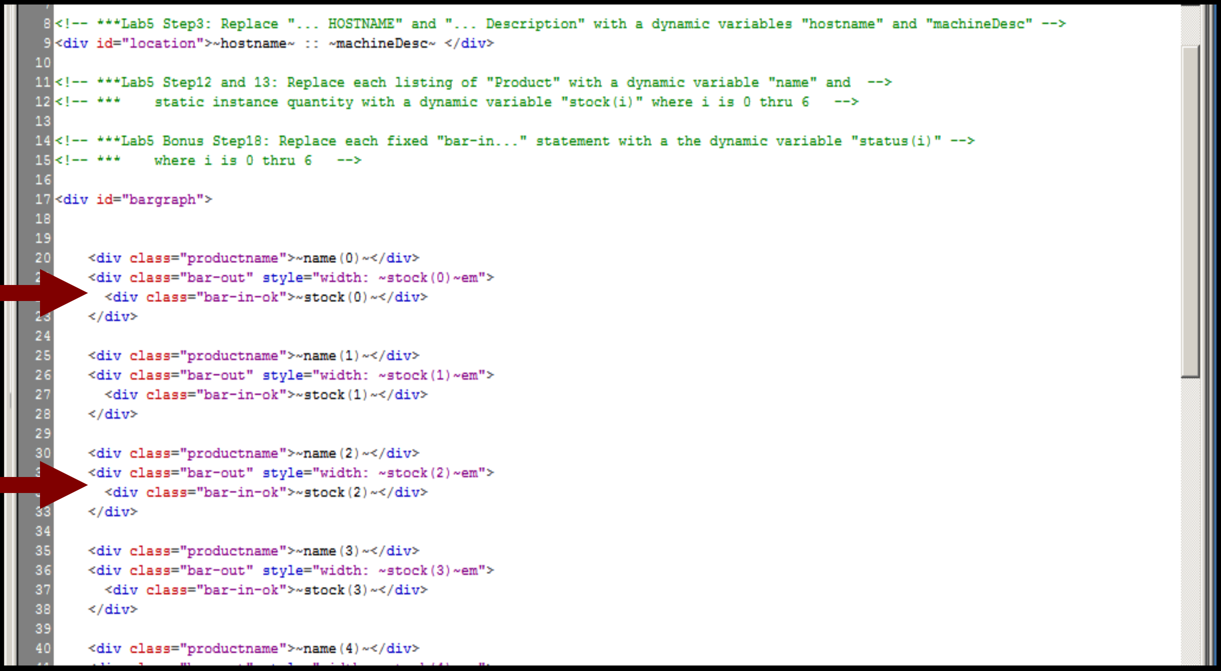
Product

Web Page Directory (Destination):
C:\Masters\1538Lab5\WebPages2

12 **Return** to `index.htm` in **Crimson Editor**.
Replace each instance of the static string "Product" with a dynamic variable with a single parameter. Call this variable `~name(i)~`, where i represents an index from 0 to 6. This adds dynamic variables for each of the product names.

13 Replace each static instance of stock quantities with a dynamic variable. Name this variable `~stock(i)~`, again with indices from 0 to 6. The stock is listed in two places for each bar. The first prints as text inside the bar, while the second controls the display width of the bar. Be sure to replace both locations, but leave the `em` designation. (The `em` is a standard print measurement.)
Ensure you replace all 7 instances for the product and stock entries.
Save your changes to the HTM file.

Index.htm

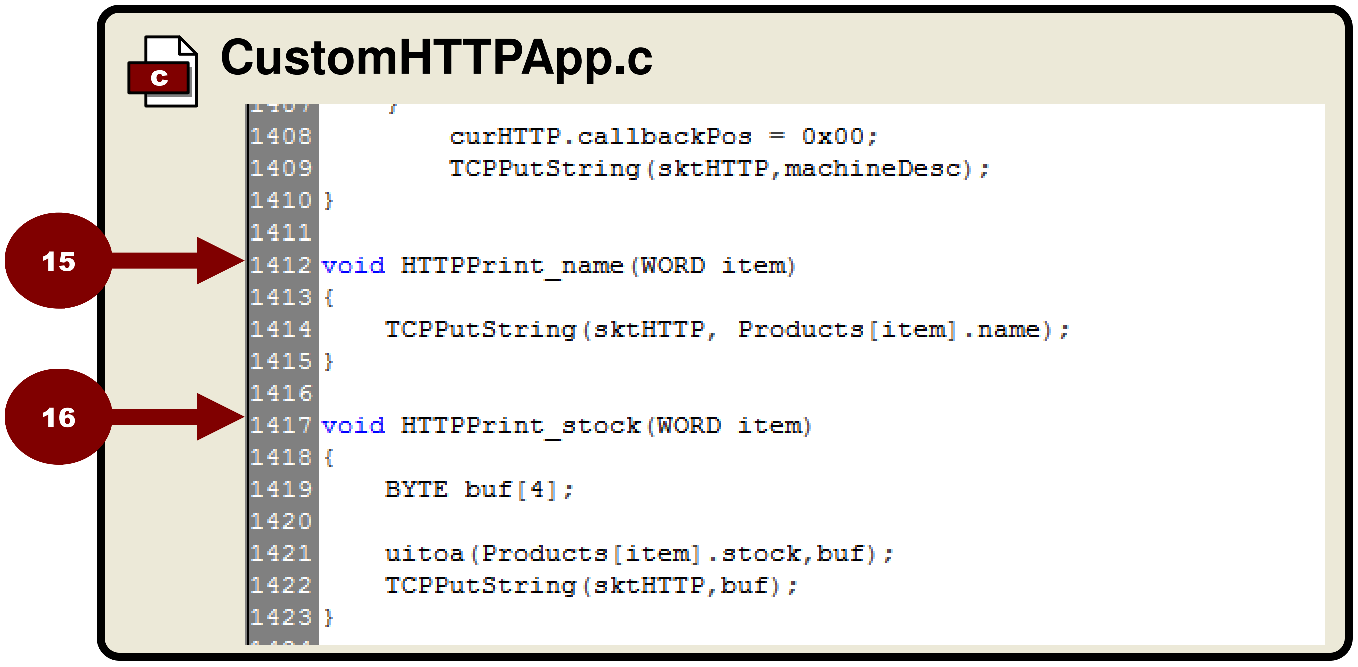


```
<!-- ***Lab5 Step3: Replace "... HOSTNAME" and "... Description" with a dynamic variables "hostname" and "machineDesc" -->
<div id="location">~hostname~ :: ~machineDesc~ </div>

<!-- ***Lab5 Step12 and 13: Replace each listing of "Product" with a dynamic variable "name" and  -->
<!-- ***    static instance quantity with a dynamic variable "stock(i)" where i is 0 thru 6   -->

<!-- ***Lab5 Bonus Step18: Replace each fixed "bar-in..." statement with a the dynamic variable "status(i)" -->
<!-- ***    where i is 0 thru 6   -->

<div id="bargraph">


    <div class="productname">~name(0)~</div>
    <div class="bar-out" style="width: ~stock(0)~em">
      <div class="bar-in-ok">~stock(0)~</div>
    </div>

    <div class="productname">~name(1)~</div>
    <div class="bar-out" style="width: ~stock(1)~em">
      <div class="bar-in-ok">~stock(1)~</div>
    </div>

    <div class="productname">~name(2)~</div>
    <div class="bar-out" style="width: ~stock(2)~em">
      <div class="bar-in-ok">~stock(2)~</div>
    </div>

    <div class="productname">~name(3)~</div>
    <div class="bar-out" style="width: ~stock(3)~em">
      <div class="bar-in-ok">~stock(3)~</div>
    </div>

    <div class="productname">~name(4)~</div>
```

14 Execute **MPFS2 Utility** similar to step 4 and click **Generate and Upload** to program your new web pages. . (IF USING THE SOLUTION WORKSPACE, skip to Step 16 to continue)

15 Insert another callback function HTTPPrint_name() into CustomHTTPApp.c. to display the name of the requested product. This function accepts a single WORD parameter.
Product data is stored in the Products array. Use the parameter as an index to this array, and print out the associated name element using TCPPutString, as shown below.

16 Insert **another** callback function for HTTPPrint_stock(). This function also accepts a single WORD parameter. Use the uitoa(WORD, BYTE*) function to convert the stock element from a binary value to a string, then write the output to the socket.



CustomHTTPApp.c

```c
	}
		curHTTP.callbackPos = 0x00;
		TCPPutString(sktHTTP,machineDesc);
}

void HTTPPrint_name(WORD item)
{
	TCPPutString(sktHTTP, Products[item].name);
}

void HTTPPrint_stock(WORD item)
{
	BYTE buf[4];

	uitoa(Products[item].stock,buf);
	TCPPutString(sktHTTP,buf);
}
```

17 **SEE THE SECTION 2.1 OR 2.3 OF APPENDIX A FOR MPLABX BUILD AND/OR RUN OPTIONS.** Build and Program the firmware into the selected device in RELEASE mode.

BUILD TO CONFIRM NO SYNTAX ERRORS

SEE SECTION 1.8 IN APPENDIX A FOR CONFIGURING YOUR PROGRAMMER

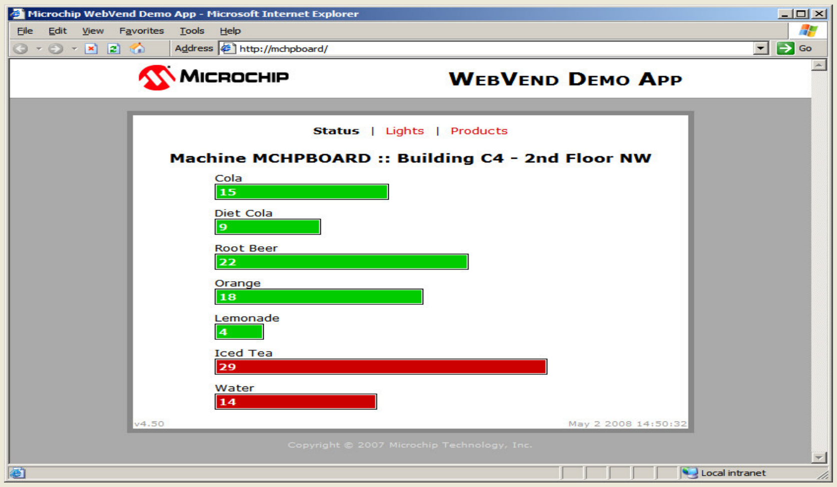
BUILD, PROGRAM AND RUN IN THE TARGET SYSTEM

18 Access your board's web page using a web browser.
Verify that the bar graph display is now populated with dynamic data from the board. Vend a few drinks and click **Refresh** in your browser to verify that the stock numbers and bar lengths change.

Results

A completed implementation will look similar to the following when viewed in a web browser. As products are purchased from the machine, a refresh of the browser page will indicate the new stock levels. An automatically updating display, while technically feasible, is outside the scope of this class.

Sample Completed Web Page


Microchip WebVend Demo App - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://mchpboard/
Go
MICROCHIP
WEBVEND DEMO APP
Status | Lights | Products
Machine MCHPBOARD :: Building C4 - 2nd Floor NW
Cola
15
Diet Cola
9
Root Beer
22
Orange
18
Lemonade
4
Iced Tea
29
Water
14
v4.50
May 2 2008 14:50:32
Copyright © 2007 Microchip Technology, Inc.
Local intranet

Code Analysis

In the HTML file, two types of dynamic callbacks were inserted. In the first portion of the lab, standard text outputs were added. These callbacks receive no parameters, and only needed to call `TCPPutString()` to write text into the web page.

One of these callbacks (for the machine description string) may need to output more than 16 bytes. To prevent potential buffer overruns, this callback had to manage its output state. Since the output was still relatively short, the function just checks if enough space is available. If not, it returns after setting a flag in `curHTTP.callbackPos`, which indicates to the server that this callback must be invoked again to complete its output. This is an acceptable solution for strings less than about 50 bytes. For longer outputs, write as many bytes as possible and use the `callbackPos` variable to track to position of the output, as discussed in the lecture slides. Once complete, restore the `callbackPos` variable to zero to indicate that the callback is finished. Dynamic variables for the names and stock levels of each product were also added. These variables had parameters passed to the functions. All parameters are passed as `WORD` values, which the callback function uses as an index to the `Products` array. The outputs of these functions were used for both printing text and controlling display elements, such as the length of the bars in the graph display.

Conclusions

In this lab, web-based monitoring was been added to the vending machine. Several examples of dynamic variables were implemented, demonstrating a wide range of capabilities for this flexible feature. With just a few lines of code, the vending machine can now output its current status in a clean bar graph display.

Bonus Procedure

Dynamic variables can also be used to control visual elements. Notice that one of the bottom two bars are red, while the other bars are green. We'd like to control this color so that any products where stock is running low are automatically highlighted in red.

18 Replace the text class attributes low and ok with a dynamic variable called `~status(i)~` within the index.htm file via Crimson Editor.

19 Execute **MPFS2 Utility** to bundle your changes to the webpage and upload them to the development board. The utility will also update your copy of `HTTPPrint.h` to link in your new dynamic variables.

20 Insert a callback for `HTTPPrint_status()` in `CustomHTTPApp.c`. This callback function should print the string "`low`" if stock is less than `8`, and "`ok`" otherwise.

21 **SEE THE SECTION 2.1 OR 2.3 OF APPENDIX A FOR MPLABX BUILD AND/OR RUN OPTIONS.** Build and Program the firmware into the selected device in RELEASE mode.

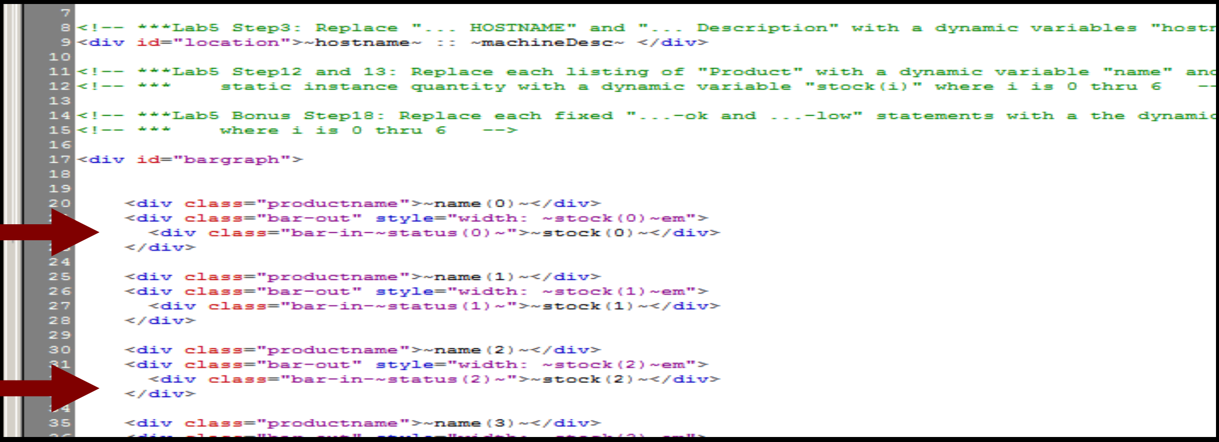
BUILD TO CONFIRM NO SYNTAX ERRORS

SEE SECTION 1.8 IN APPENDIX A FOR CONFIGURING YOUR PROGRAMMER

BUILD, PROGRAM AND RUN IN THE TARGET SYSTEM

22 Verify after selective vending, the quantities and colors change appropriately.

BONUS index.htm



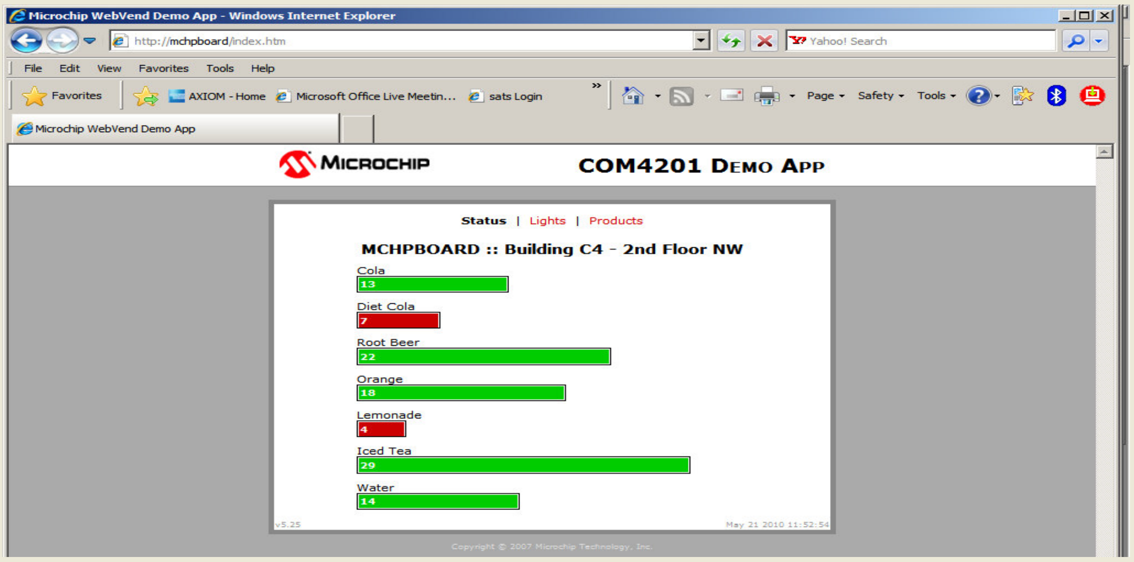
```
<!-- ***Lab5 Step3: Replace "... HOSTNAME" and "... Description" with a dynamic variables "hostr
<div id="location">~hostname~ :: ~machineDesc~ </div>

<!-- ***Lab5 Step12 and 13: Replace each listing of "Product" with a dynamic variable "name" and
<!-- ***     static instance quantity with a dynamic variable "stock(i)" where i is 0 thru 6    --

<!-- ***Lab5 Bonus Step18: Replace each fixed "...-ok and ...-low" statements with a the dynamic
<!-- ***     where i is 0 thru 6    -->

<div id="bargraph">


    <div class="productname">~name(0)~</div>
    <div class="bar-out" style="width: ~stock(0)~em">
      <div class="bar-in-~status(0)~">~stock(0)~</div>
    </div>

    <div class="productname">~name(1)~</div>
    <div class="bar-out" style="width: ~stock(1)~em">
      <div class="bar-in-~status(1)~">~stock(1)~</div>
    </div>

    <div class="productname">~name(2)~</div>
    <div class="bar-out" style="width: ~stock(2)~em">
      <div class="bar-in-~status(2)~">~stock(2)~</div>
    </div>

    <div class="productname">~name(3)~</div>
```

CustomHTTPApp.c

20

```
void HTTPPrint_status(WORD item)
{
    if(Products[item].stock < 8)
        TCPPutROMString(sktHTTP, (ROM BYTE*)"low");
    else
        TCPPutROMString(sktHTTP, (ROM BYTE*)"ok");
}

#endif
```

Sample Web Page after Bonus Procedure


Microchip WebVend Demo App - Windows Internet Explorer
http://mchpboard/index.htm
Yahoo! Search
File Edit View Favorites Tools Help
Favorites
AXIOM - Home
Microsoft Office Live Meetin...
sats Login
Page
Safety
Tools
Microchip WebVend Demo App
MICROCHIP

COM4201 DEMO APP

Status | Lights | Products

MCHPBOARD :: Building C4 - 2nd Floor NW

Cola
13

Diet Cola
7

Root Beer
22

Orange
18

Lemonade
4

Iced Tea
29

Water
14

v5.25

May 21 2010 11:52:54

Copyright © 2007 Microchip Technology, Inc.

Lab Exercise 6

Web-based Control via the GET Method

Purpose

Now that the vending machine can be monitored remotely, it would be nice to add some control features. As discussed, this can be accomplished through two different HTTP methods: GET and POST.
In this lab, you will learn how to use the GET method to pass short amounts of data to the device. In the process, an interface for controlling the lights on the vending machine will be added. Since the current development platform only has LEDs, one of these will be used for simulation. Control via the POST method will be discussed in a future class.

Requirements

Environment:	MPLAB® IDE 8.60 , X, or later, MPFS Utility, Masters 1538 Installer 1.536
C Compiler:	MPLAB C18 v3.38 or later, C30 v3.30 or later, or C32 v1.12 or later
H/W Tools:	PIC18F: PICDEM.net™ 2, or PIC24FFJ128GA010 PIM: Explorer 16 w/ 10Mbps, 10/100Mbps PICtail™ Plus or MRF24WB0 or PIC32MX360 or PIC32MX460 PIM: Explorer 16 & 10Mbps, 10/100Mbps PICtail™ Plus or MRF24WB0 AND MPLAB Real ICE™ , or MPLAB ICD 3 AND DHCP Enabled Router (wired or wireless)
Lab Files:	C:\Masters\1538\Lab6...

Objectives

- Study the procedural steps associated with adding HTML forms to a web page to implement control
- Practice with including the necessary Callback code within an application to issue HTML Get based control.

Web Page Directory:
C:\Masters\1538\Lab6\WebPages2

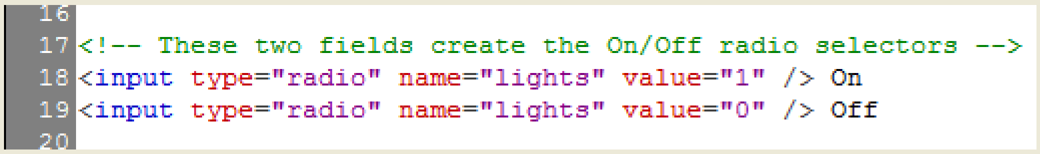
Procedures

1. Open `lights.htm`, from the folder listed above with the Crimson Editor.
This file contains the HTML form that will be used to CONTROL the LEDs.
Start ▸ Programs ▸ Crimson Editor ▸ Crimson Editor

2. Find the `<form>` tag, and notice that the `method` attribute is set to `get`. The form contains two input buttons, both of which are named lights.
Find the radio button inputs. Notice that both are named `lights`. When the form is submitted, the value of the lights parameter will be set equal to whichever radio button is selected.

lights.htm

2



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

3 Execute **MPFS2 Utility** similar to Lab5 step 4 and click **Generate and Upload** to program your new web pages.
(IF USING THE SOLUTION WORKSPACE, skip to Step 8 to continue)

4 MPLAB File ▸ Open Project

Lab6-C18-PICDN2-ETH97.X	for the PICDEM.net 2 with internal MAC/PHY setup
Lab6-C18-PICDN2-MRF24WB.X	for the PICDEM.net 2 with Wireless MAC/PHY setup
Lab6-C30-EXP16-ENC28.X	for the Explorer 16 with PIC24F/dsPIC33F PIMs with 10Mbps module
Lab6-C30-EXP16-ENC624.X	for the Explorer 16 with PIC24F/dsPIC33F PIMs with 10/100Mbps module
Lab6-C30-EXP16-MRF24WB.X	for the Explorer 16 with PIC24F/dsPIC33F PIMs with WIFI module
Lab6-C32-EXP16-ENC28.X	for the Explorer 16 with PIC32MX3xx/4xx PIMs with 10Mbps module
Lab6-C32-EXP16-ENC624.X	for the Explorer 16 with PIC32MX3xx/4xx PIMs with 10/100Mbps module
Lab6-C32-EXP16-MRF24WB.X	for the Explorer 16 with PIC32MX3xx/4xx PIMs with WIFI module

5 Open `CustomHTTPApp.c` within MPLAB's project window. Forms submitted via GET are processed in `HTTPExecuteGet()`. **Find** this function.
Note: The first step in this function is to determine which file name is being accessed (and therefore which form has been submitted). The call to `MPFSGetFilename()` handles this, and then a `memcmp` variant is used to check the filename.

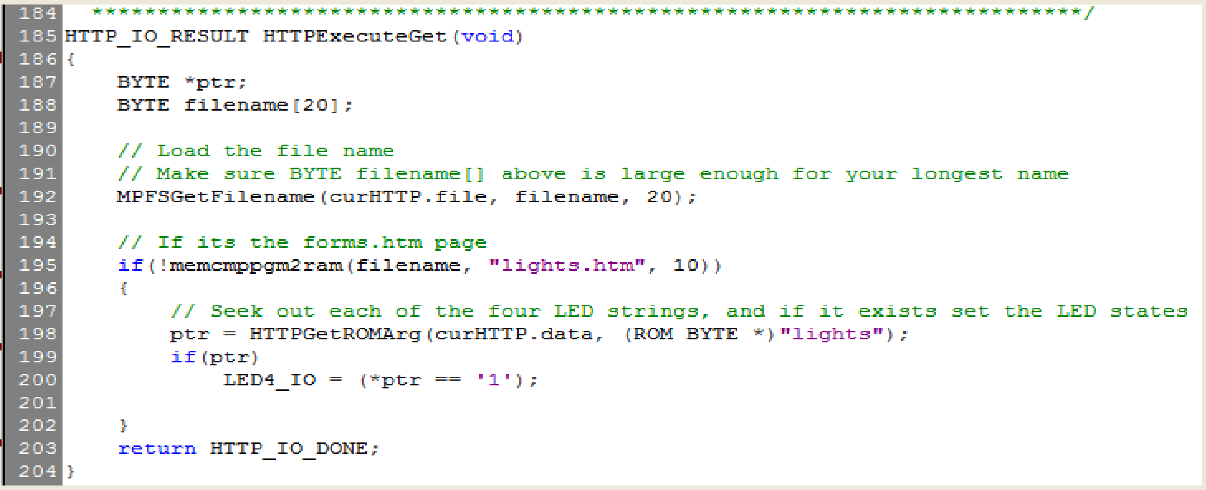
`memcmppgm2ram()` compares an array in RAM to an array in program memory. It accepts the two pointers and a length parameter. If the two arrays match, it returns zero, which is why the comparison is inverted.

6 Update the first comparison to check for `lights.htm`. Remember to update the length parameter to 10 as well. If the name matches, the application should now look for a parameter called `"lights"`. If the pointer returned by the call to `HTTPGetROMArg()` is not `NULL` and matches ASCII `'1'`, the pin `LED4_IO` should be turned on. Otherwise, the pin should be set to off.

Parameters returned from web forms are always represented as strings. If you return numeric values, they must be parsed from their string representations.

7 Insert the form processing functionality as shown on the following page. Remove any remaining code in `HTTPExecuteGet()`, which will include the processors for `cookies.htm` and `leds.cgi`.
These forms were implemented in the demo application, but no longer exist in your project's web pages. **Ensure** that the function still returns `HTTP_IO_DONE`. This indicates to the HTTP2 server that the function is complete and should not be called again.

CustomHTTPApp.c



```
 ***************************************************************************/
HTTP_IO_RESULT HTTPExecuteGet(void)
{
    BYTE *ptr;
    BYTE filename[20];

    // Load the file name
    // Make sure BYTE filename[] above is large enough for your longest name
    MPFSGetFilename(curHTTP.file, filename, 20);

    // If its the forms.htm page
    if(!memcmppgm2ram(filename, "lights.htm", 10))
    {
        // Seek out each of the four LED strings, and if it exists set the LED states
        ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"lights");
        if(ptr)
            LED4_IO = (*ptr == '1');

    }
    return HTTP_IO_DONE;
}
```

8 **SEE THE SECTION 2.1 OR 2.3 OF APPENDIX A FOR MPLABX BUILD AND/OR RUN OPTIONS.** Build and Program the firmware into the selected device in RELEASE mode.

BUILD TO CONFIRM NO SYNTAX ERRORS

SEE SECTION 1.8 IN APPENDIX A FOR CONFIGURING YOUR PROGRAMMER

BUILD, PROGRAM AND RUN IN THE TARGET SYSTEM

9 Access your board's web page and click on the link for **Lights**.
Verify that one of the LEDs on the board can be controlled using the web form.

Results

A correct implementation will allow you to control the state of one LEDs using a web form.

Code Analysis

As mentioned earlier, data received via the GET method is stored in `curHTTP.data`. This data is located using `HTTPGetROMArg()`, which returns a pointer to the value, or NULL when the requested parameter was not found. This pointer is then compared against expected values to determine the appropriate action.

Since HTTPExecuteGet() handles all GET forms, the function must resolve the form that was submitted by checking the file name. The MPFSGetFilename() function determines the name of the file being accessed by the server and stores it back to the temporary string for comparison.

Bonus Procedure

Most web forms pre-select the current value to provide a more intuitive interface to the user. This can be accomplished for your form using dynamic variables.

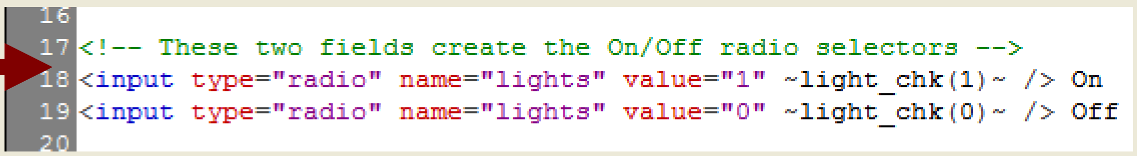
10 Open lights.htm in **Crimson Editor** again. You will add a dynamic variable here to pre-select the correct radio button depending on whether the LED is on or off. A radio button will be selected if it has a checked attribute.

11 **Insert** a dynamic variable in each radio button, which will allow your application to dynamically insert this attribute.
Save your changes to the HTML code

12 Execute **MPFS2 Utility** similar to step 3 and click **Generate and Upload** to program your new web pages.
. (IF USING THE SOLUTION WORKSPACE, skip to Step 15 to continue)

lights.htm

11



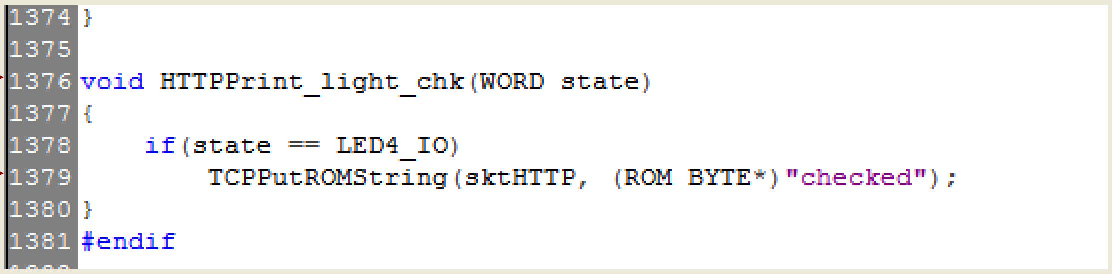
```

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

```

13 Insert the callback function HTTPPrint_light_chk() via MPLAB into the file CustomHTTPApp.c. This function will accept a single WORD parameter.

CustomHTTPApp.c

13

14



```
}

void HTTPPrint_light_chk(WORD state)
{
    if(state == LED4_IO)
        TCPPutROMString(sktHTTP, (ROM BYTE*)"checked");
}
#endif
```

14 Inside this function, Insert a check if the parameter received matches the current state of `LED4_IO`. If it does, use `TCPPutROMString()` to write the string `"checked"` to the page.

15 **SEE THE SECTION 2.1 OR 2.3 OF APPENDIX A FOR MPLABX BUILD AND/OR RUN OPTIONS.** Build and Program the firmware into the selected device in RELEASE mode.

BUILD TO CONFIRM NO SYNTAX ERRORS

SEE SECTION 1.8 IN APPENDIX A FOR CONFIGURING YOUR PROGRAMMER

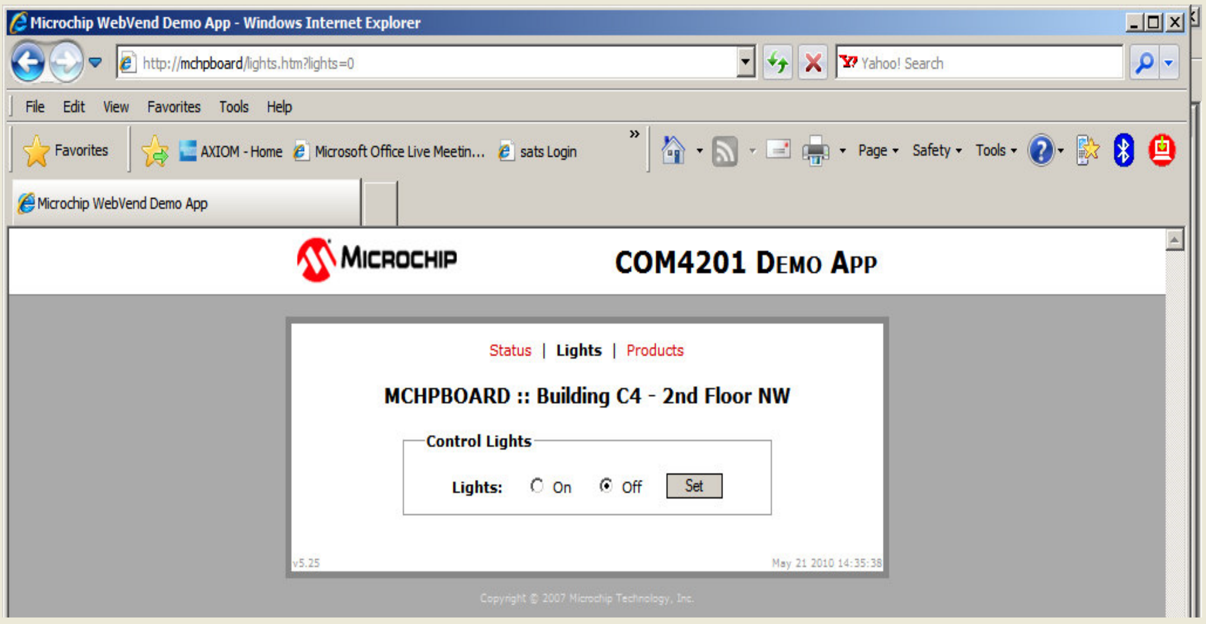
BUILD, PROGRAM AND RUN IN THE TARGET SYSTEM

16 Verify that the current state of the LED is now pre-selected on your form when the page loads.

Bonus Procedure Results:

In this section, a dynamic variable was added to the radio buttons. This dynamic variable either prints "checked" or nothing, depending on the state of the LED. By printing "checked", the server can control the default state of

Pre-Selected Radio Button


Microchip WebVend Demo App - Windows Internet Explorer
http://mchpboard/lights.htm?lights=0
Yahoo! Search
File Edit View Favorites Tools Help
Favorites
AXIOM - Home
Microsoft Office Live Meetin...
sats Login
Page
Safety
Tools
Microchip WebVend Demo App
MICROCHIP
COM4201 DEMO APP
Status | Lights | Products

MCHPBOARD :: Building C4 - 2nd Floor NW

Control Lights

Lights: ( ) On [✓ Off] Set

v5.25 May 21 2010 14:35:38

Copyright © 2007 Microchip Technology, Inc.

Appendix A

MPLAB® X IDE Quick Reference Guide

Table of Contents

Section 1.7
How to close a project

1 There are two methods you can use to close a project:
Method 1:
Right click on the top node of the project in the project tree (the chip icon) and select **Close** from the popup menu (about 2/3 of the way down).
Method 2:
From the main menu, select **File ▸ Close Project (*project name*)**
where ***project name*** is the name of the project you wish to close—there may be multiple similar menu items if you have more than one project open in the IDE.

Section 1.8
How to modify project settings

1 There are three ways to access a project's settings:
Method 1:
Right click on the top node (chip icon) of a project in the project tree and select **Properties** at the very bottom of the long popup menu.

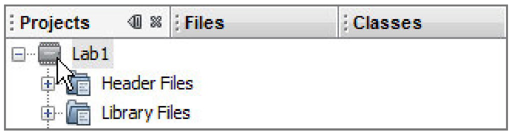


Figure 1.8.1
The top node of a project in the project tree

Method 2:
From the main menu select **File ▸ Project Properties (*project name*)**
Method 3:
If the **Project Environment** window is open (bottom left corner by default), you can click on the "wrench and bolt" icon in its left margin.

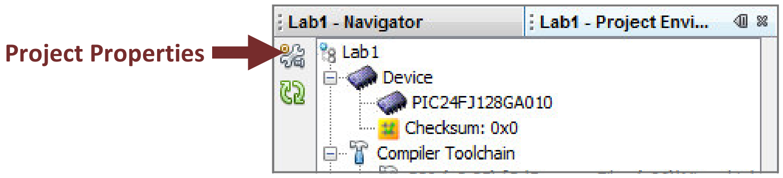
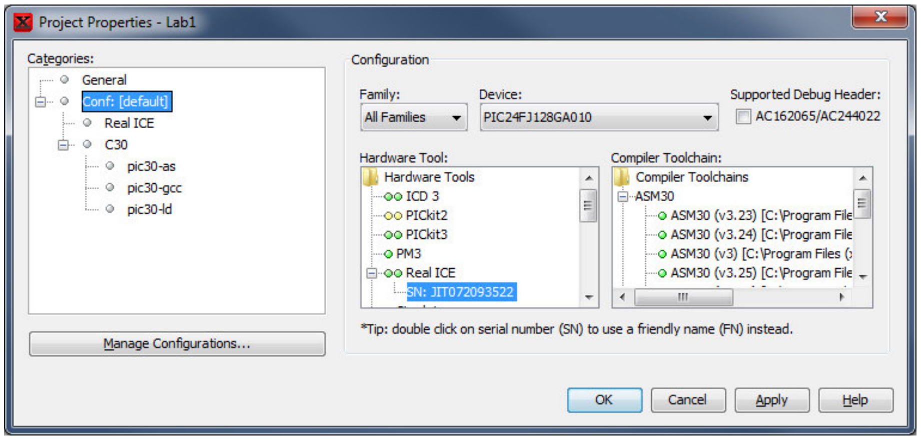


Figure 1.8.2
Project Properties button in the Project Environment window.

2 From here you can select a different device, debug tool or build tool and you can modify any of their settings. When choosing a new tool, click the Apply button to make it show up in the tree on the left side.



Family: All Families
Device: PIC24FJ128GA010
Supported Debug Header: AC162065/AC244022
Hardware Tool: SN: JIT072093522
Compiler Toolchain:

*Tip: double click on serial number (SN) to use a friendly name (FN) instead.

2. Building Projects

Section 2.1

How to build a project

There are several ways to build a project in MPLAB X depending on what you intend to do with the results. This method is only used to see the results of a build or to produce a release mode hex file.

There are two different types of build you can do:
Build:
This will build only the files in your project that have changed since the last build or it will build everything if nothing has been built previously. It will generally be faster to use this type of build, especially for larger projects.
Clean and Build:
This will remove any intermediate files generated by the previous build and will build every file in your project regardless of whether or not it has changed since the last build to ensure a full, clean build.

Information

The **Build** and **Clean and Build** functions are not intended for use before **Run Project**, **Debug Project** or **Make and Program Target**. All three of those functions automatically do a build before performing further steps. No harm will be done by using **Build** or **Clean and Build** first, but it will be a duplication of effort and will waste time.

1 There are two ways to access these two build types:
From the Main Toolbar:

Build	***Make*** in MPLAB IDE 8
Clean and Build	***Build All*** in MPLAB IDE 8

Alternatively, you can right click on the top node of a project (chip icon) in the project tree and select either **Build** or **Clean and Build** from the popup menu.

Section 2.2

How to build and run a project with a debugger

When you want to build a project for the purpose of programming a target to run with a debugger like the MPLAB® ICD 3 or REAL ICE, this is the method to use.

Information

It is not necessary to do a **Build** or **Clean and Build** before doing a **Debug Project** because a build will be done automatically.

1 There are three ways to build and run your code through a debugger:
Method 1:
Click on the **Debug Project** button on the main toolbar
Method 2:
Right click on the top node of the project (chip icon) in the project tree and select **Debug** from the popup menu.
Method 3:
From the main menu, select **Debug ▸ Debug Project (*project name*)**

This will perform the following tasks automatically:
a. Build (make) project in debug mode
b. Program target (unless using simulator)
c. Run code on target

Section 2.3

How to build and run a project without a debugger

Information

It is not necessary to do a **Build** or **Clean and Build** before doing a **Run Project** because a build will be done automatically.

When you want to build a project for the purpose of programming a target to run without a debugger, this is the method to use.

1 There are three ways to build and run your code on a target:

Method 1:
Click on the **Run Project** button on the main toolbar ▷

Method 2:
Right click on the top node of the project (chip icon) in the project tree and select **Run** from the popup menu.

Method 3:
From the main menu, select **Run ▸ Run Project (*project name*)**

This will perform the following tasks automatically:

a. Build (make) project in release mode
b. Program target
c. Run code on target

Additional Exercise Creating a Simple Web

Purpose

Modern web pages are created using a combination of HTML and CSS code. HTML is a markup language, meaning that the content is divided into a structure, with each element given a specific type. CSS is a style language which specifies how to present the content found in the HTML structure. Together, these files are parsed by a web browser to create the on-screen display.
In this lab, you will create a simple web page with your name, a brief paragraph about yourself, and a few visual elements. You will test your page using the web browser on your PC.
Web pages for Microchip's embedded platforms are compiled into an Microchip File System (MPFS) image. This image is similar to a zip file, and includes all of your web page code in a format that is easy for the PIC to serve. To finish the lab, you will bundle and upload your web page to your development board using the **MPFS2 Utility**, which will allow other attendees to view the page on their PC.
If you finish early, a bonus section will teach some basic CSS to help you format your page.

Requirements

Software:	**MPFS utility from Masters 1538 Installer v1.536, Crimson Editor**
Environment:	MPLAB® IDE 8.60 ,X, or later, Masters 1538 Installer 1.536
C Compiler:	Not Required for this Lab
H/W Tools:	PIC18F: PICDEM.net™ 2, or PIC24FFJ128GA010 PIM: Explorer 16 w/ 10Mbps, 10/100Mbps PICtail™ Plus or MRF24WB0 or PIC32MX360 or PIC32MX460 PIM: Explorer 16 & 10Mbps, 10/100Mbps PICtail™ Plus or MRF24WB0 AND MPLAB Real ICE™ , or MPLAB ICD 3 AND DHCP Enabled Router (wired or wireless)
Lab Files:	C:\Masters\1538\AddlLab\...

Objectives

- Practice HTML and CSS style sheet programming
- Utilize the MPFS utility to package web page information and transfer to dev tools via TCP/IP

Web Page Directory: C:\Masters\1538\AddlLab\WebPages2

Procedures

1 Confirm all the files located in `WebPages2` directory shown above are deleted These are shipped with the TCP/IP Stack, but are not needed for this class. However the same folder location will be used for new pages.

2 Launch the **Crimson Editor** to **create** a new file named **index.htm** and save it in the **WebPages2** mentioned previously. Start ▸ Programs ▸ Crimson Editor ▸ Crimson Editor File ▸ New File ▸ Save As ▸ **index.htm** in the folder C:\Masters\1538\AddlLab\Webpages2\

The default extension for HTML code in Crimson Editor is `*.html`. The default file name for the HTTP2 server is `index.htm`. Make sure your extension only has three letters!

3 Insert the required **<html>**, **<head>**, and **<body>** structure tags as shown in the sample web page below. **Include** the appropriate closing tags **</html>**, **</head>**, and **</body>** as well.

Sample HTML Page Structure

```
<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 Inside the **<head>** tag, insert a **<title>** tag enclosing your company name. Insert a **<h1>** tag inside the **<body>** tag, enclosing your personal name. Be sure to **include** appropriate closing tags as well.

5 Below the **<h1>** tag, insert a **<p>** tag enclosing two brief sentences about yourself. This will appear as a paragraph of text on your page.

6 Below the **<p>** tag, insert another **<p>** tag. **List** information from your business card such as your name, title, address, and phone number. Insert a **<br />** tag to add a line break after each element of data.

In HTML, a **<title>** tag in the **<head>** section displays in the browser's title bar, but not on the page itself. The **<h1>** tag will display as big bold text in your page. `<p>` tags are displayed with a blank line of space between them.

The `<br />` tag forces a line break without this extra line of space. The `<br />` tag is a self-closing tag. It does not enclose content, but merely indicates a visual display element. Other examples of self-closing tags include image tags `<img … />` and form input fields `<input … />`.

7 Insert a link to your company's website (or a site of your choosing) as the last line in the "business card" information. Use an `<a>` tag. Insert an `href` (hyper-text reference) parameter to the tag indicating the link's destination. Enclose the text you wish to display (underlined in blue) in between the `<a>` and `</a>` tags. It should look something like:

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

This should be a fully qualified URL, including `http://`, because the link references an external server. For local links you would list only the relative path to the page, such as:

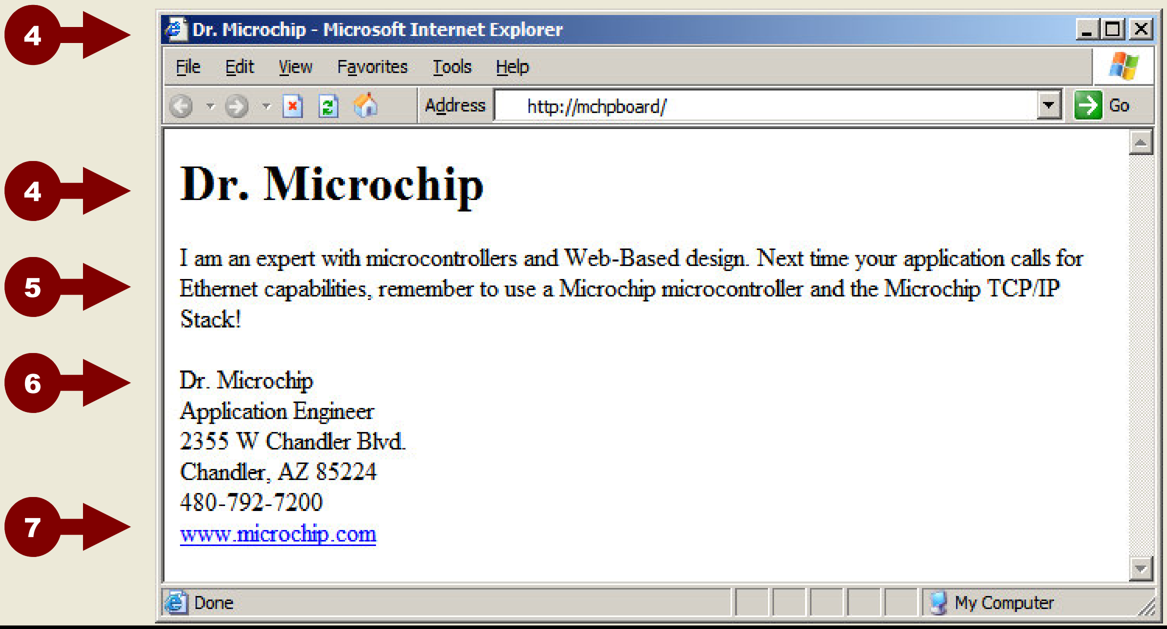
```
<a href="page2.htm">Next Page</a>
```

8 Save the file, then view your completed web page by opening it in your web browser.
Windows Desktop ▸ My Computer ▸ Navigate to `WebPages2` ▸ Double Click `index.htm`

As you view the web page in the step, you can return to steps 3 through 8 to make changes. After saving, use the browser's **Refresh** button to see the modifications.

Note the size of the file as it is saved from Crimson Editor, less than 500 bytes. It is important to avoid using editors that generate large files that become difficult to fit in embedded memory arrays.

Sample Web Page


4
4
5
6
7
Dr. Microchip - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://mchpboard/
Go
Done
My Computer

Dr. Microchip

I am an expert with microcontrollers and Web-Based design. Next time your application calls for Ethernet capabilities, remember to use a Microchip microcontroller and the Microchip TCP/IP Stack!

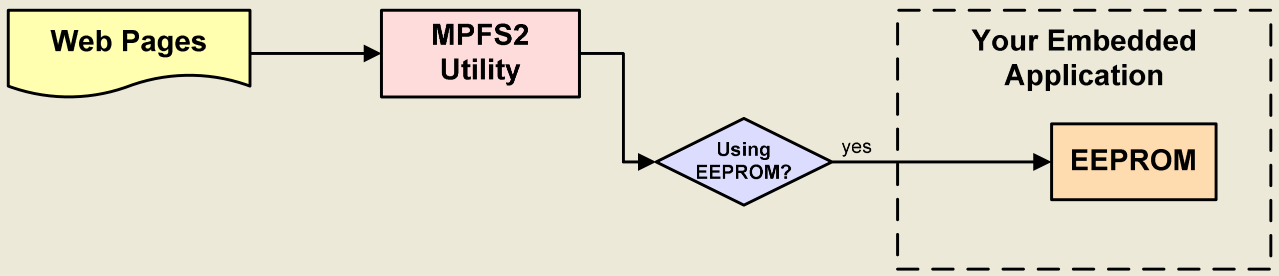
Dr. Microchip
Application Engineer
2355 W Chandler Blvd.
Chandler, AZ 85224
480-792-7200
www.microchip.com

Your classroom may not have Internet access, so the link you added to your web page may not work properly.

In the next few steps, you will use the MPFS2 Utility to bundle your web page and upload it to your development board. Before beginning, ensure that your development board is still running by confirming the new IP address on the LCD, and confirm the index.htm is saved.

9 **Open** the **MPFS2 Utility**, so on the next page of this manual.
Start ▸ Programs ▸ Microchip ▸ 1538 ▸ MPFS2

How Web Pages are Bundled



Source Directory:
C:\Masters\1538\AddlLab\WebPages2\

10 Confirm that **Start With: Webpage Directory** is selected and that the correct source directory shown above is entered:

11 Confirm that **Output: BIN Image** is selected.

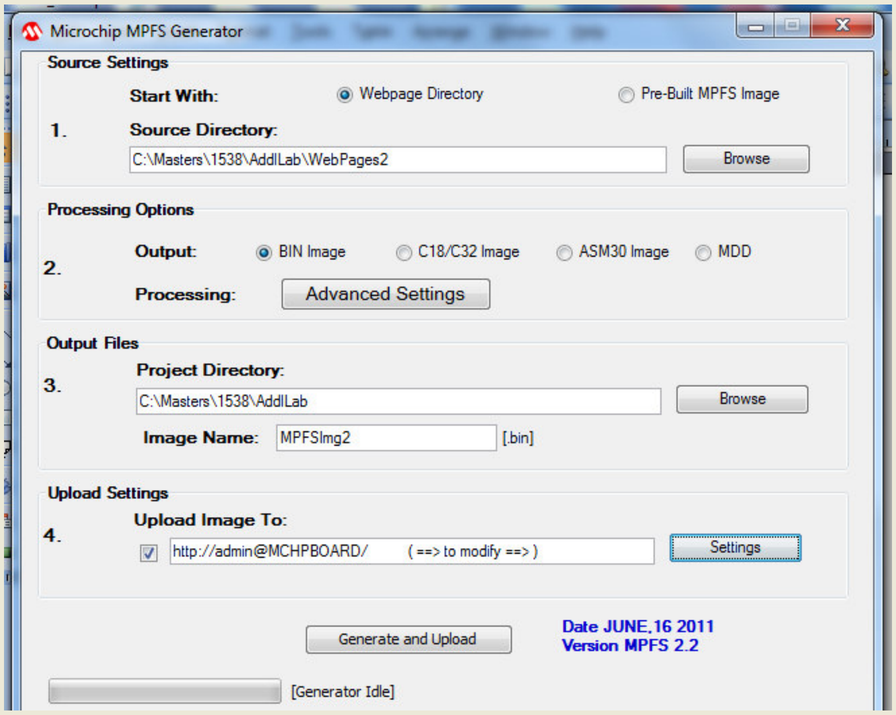
Project Directory:
C:\Masters\1538\AddlLab\

12 Confirm that the OUTPUT FILES project directory shown above is entered:

13 Confirm that **Upload Image** is selected. Then click the **Settings** button and make the following changes and confirmations:

- Click the **Defaults** button to restore the standard settings.
- Set the Device Address to the host name you chose in Lab 1.
- Click **OK** to save your changes.

MPFS2 Utility After Configuration



Microchip MPFS Generator

Source Settings

Start With: [● Webpage Directory] ○ Pre-Built MPFS Image

1. Source Directory: C:\Masters\1538\AddlLab\WebPages2 — Browse

Processing Options

2. Output: [● BIN Image] ○ C18/C32 Image ○ ASM30 Image ○ MDD

Processing: Advanced Settings

Output Files

3. Project Directory: C:\Masters\1538\AddlLab — Browse

Image Name: MPFSImg2 [.bin]

Upload Settings

4. Upload Image To: [✓] http://admin@MCHPBOARD/ (==> to modify ==>) — Settings

Generate and Upload

Date JUNE,16 2011
Version MPFS 2.2

[Generator Idle]

The next step may cause the Windows Firewall to prompt you about allowing the MPFS Generator to access the network. Be sure to click **Unblock** if this occurs.

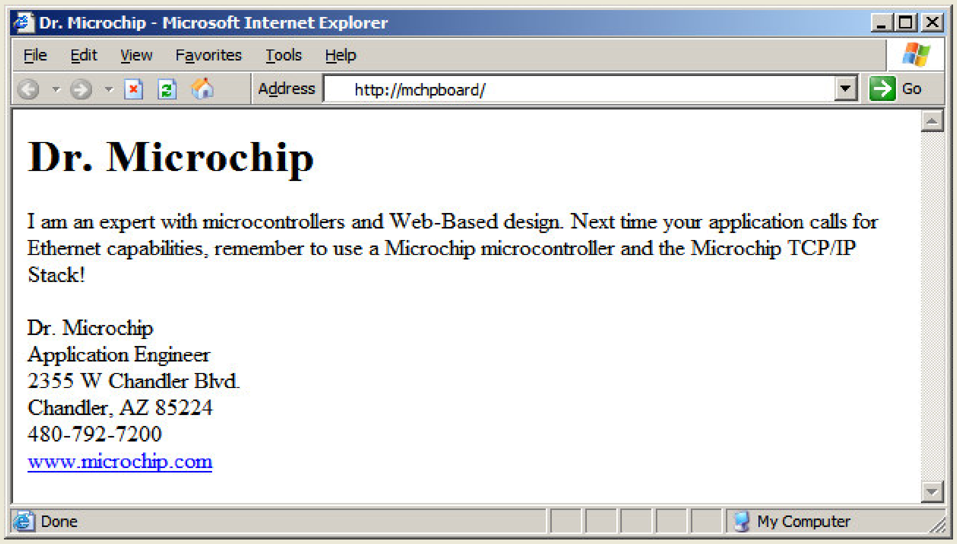
14 Click the **Generate and Upload** button. The MPFS2 Utility will package your web page and upload it to the development board.

15 Verify that you can now access your new web page from your board using the same address as you used in Lab 1. Ensure the ICD2 debugger is disconnected and Confirm the Router has assigned an IP Address.

16 Access your neighbor's board using their host name in your browser's address bar. Confirm that they can also access your web page from their browser.

Results

Sample Web Page


Dr. Microchip - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://mchpboard/
Go
Done
My Computer

Dr. Microchip

I am an expert with microcontrollers and Web-Based design. Next time your application calls for Ethernet capabilities, remember to use a Microchip microcontroller and the Microchip TCP/IP Stack!

Dr. Microchip
Application Engineer
2355 W Chandler Blvd.
Chandler, AZ 85224
480-792-7200
www.microchip.com

Bonus Procedure

17 Return to **index.htm** in **Crimson Editor**. Insert a link to the style sheet you will create within the **<head>** section:

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

18 **Insert** a **class** parameter to the opening **<p>** tag containing your business card information. (The closing tag does not need modification.) The opening tag should now look like:

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

19 Save **the newly modified index.htm**,
File ▶ Save

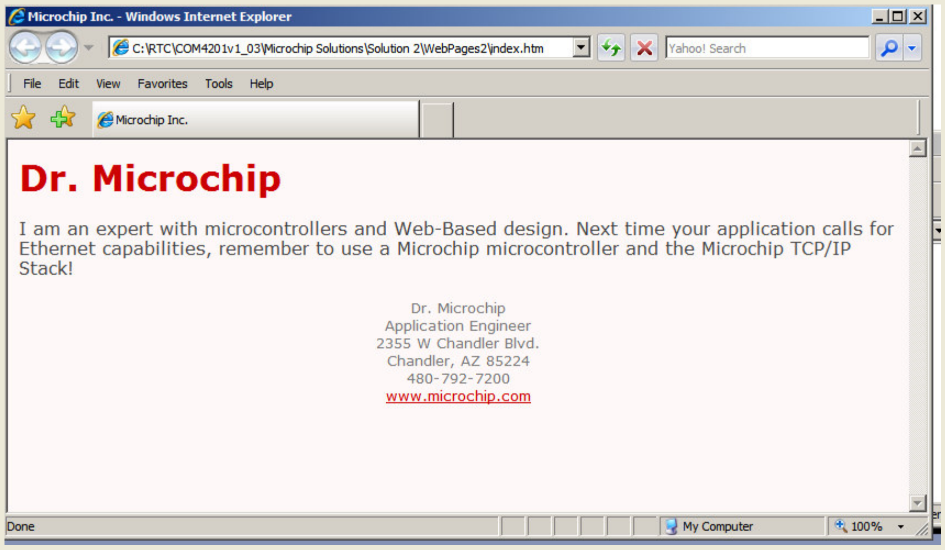
20 Create a new file named **style.css** and save it in the **WebPages2** folder with your HTML page.
File ▶ New File ▶ Save As ▶ style.css

21 Insert style sections to the file as shown in the code analysis section on the following page. Save your changes, then **repeat** steps 14 and 15 to verify that your changes are visible.

Colors are represented as hex RGB triplets, and you may modify them as you like. Sections matching the name of a tag apply to all tags of that type. Sections with a period separator followed by another name apply only to tags with a class matching the second name.

Bonus Results

Final Web Page


Microchip Inc. - Windows Internet Explorer
C:\RTC\COM4201v1_03\Microchip Solutions\Solution 2\WebPages2\index.htm
Yahoo! Search
File Edit View Favorites Tools Help
Microchip Inc.
Done
My Computer
100%

Dr. Microchip

I am an expert with microcontrollers and Web-Based design. Next time your application calls for Ethernet capabilities, remember to use a Microchip microcontroller and the Microchip TCP/IP Stack!

Dr. Microchip
Application Engineer
2355 W Chandler Blvd.
Chandler, AZ 85224
480-792-7200
www.microchip.com

Code Analysis

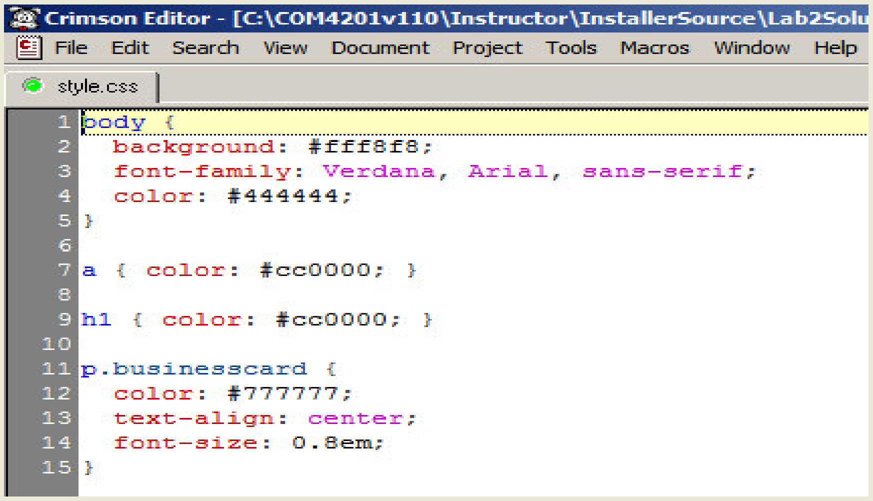
index.htm

```
<html>
  <head>
    <title>Dr. Microchip</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Dr. Microchip</h1>
    <p>I am an expert with microcontrollers and Web-Based design.  Next
      time your application calls for Ethernet capabilities, remember
      to use a Microchip microcontroller and the Microchip TCP/IP Stack!</p>
    <p class="businesscard">
      Dr. Microchip<br />
      Application Engineer<br />
      2355 W Chandler Blvd.<br />
      Chandler, AZ 85224<br />
      480-792-7200<br />
      <a href="http://www.microchip.com">www.microchip.com</a></p>
  </body>
</html>
```

The HTML document follows the required structure with an `<html>` tag enclosing `<head>` and `<body>` tags. The `<title>` tag is added to give the page a title in the browser's title bar (as well as in search engines, bookmarks, and other locations).

Within the body section, a header is inserted, followed by two paragraphs. The second paragraph uses the line break tag to force new lines. Each tag (except for self-closing tags) has an associated closing tag terminating each section. The `<link>` tag will only exist in your HTML if you completed the bonus section, as will the `class` attribute in the second `<p>` tag. If you did finish this section, you will also have a `style.css` file that looks similar to the following:

style.css


Crimson Editor - [C:\COM4201v110\Instructor\InstallerSource\Lab2Solu
File Edit Search View Document Project Tools Macros Window Help
style.css

```
body {
  background: #fff8f8;
  font-family: Verdana, Arial, sans-serif;
  color: #444444;
}

a { color: #cc0000; }

h1 { color: #cc0000; }

p.businesscard {
  color: #777777;
  text-align: center;
  font-size: 0.8em;
}
```

Conclusion

This lab has presented an extremely brief overview of HTML and CSS. These languages form the foundation for modern web page development. Before starting your own designs, you'll want to read about the other capabilities of HTML, including `<form>` tags, `<img>`, `<div>`, `<span>`, and in-line formatting with `<b>` and `<i>`.
HTML and CSS are well documented in recommendations posted by the World Wide Web Consortium at their website, www.w3c.org. For those who prefer tutorials to dense technical specifications, a quick visit to your favorite search engine will turn up hundreds of sites teaching the fundamentals of HTML design.
For the remainder of the class, you will only need to be able to read and understand the structure of a few HTML documents. The web pages used for this class will be provided, so you will not need to build them from scratch. This purpose of this exercise is only to familiarize yourself with HTML and learn enough to read code later.