

**2019 MASTERs Conference**

**23075 IoT6**

**Simplifying TCP/IP Applications with MPLAB® Harmony**

**Hands-On**

**Lab Manual**

***Instructors:***

***Martin Ruppert***

***Raji Shanmugasundaram***

***Niklas Larsson***

*Microchip Technology Inc.*



























































































































































































































LAB 2

# Introduction

You will be implementing an embedded network application to update and track status of an Vending Machine (VM).

The implementation is sectioned into two modules.

## TASK 1:

E70 HTTP- SERVER

/Vending Machine -1

Ethernet Switch

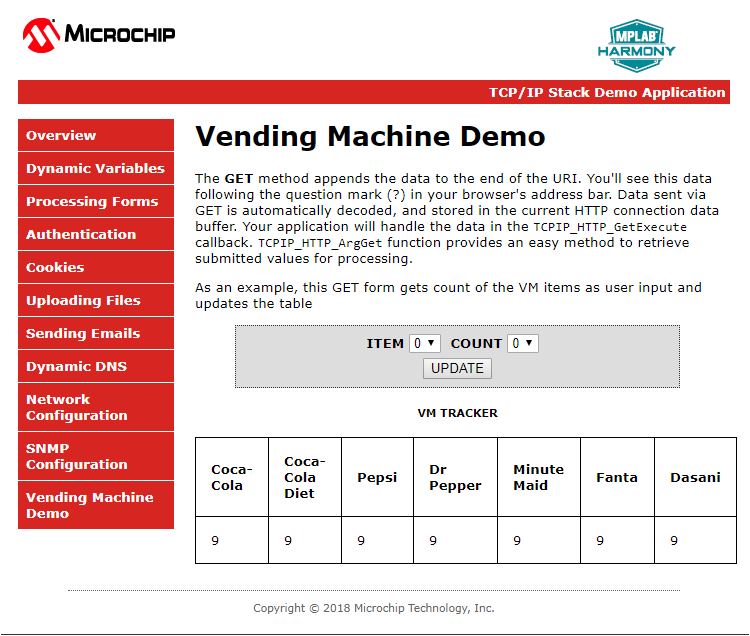
PC – HTTP-Client (1)

E70 HTTP-SERVER /Vending Machine-(n)

PC- HTTP-Client (n)

The task is to interface the up/down and select buttons and an OLED of a Vending Machine to the HTTP-Server. The HTTP-Server keeps track of the items in the Vending Machine.

The HTTP- Client (PC-web page) talks to the HTTP- server and displays the status of the Vending Machine. The HTTP-Client also sends out an update request to the HTTP- server using Ethernet Interface.



## TASK 2:

The task is to make all the VMs in the class network to talk to a server. When a Bay of a VM is empty it sends out a message to a common server.

Ethernet Switch

E70 HTTP-SERVER / Vending Machine (1)

SERVER

E70 HTTP- SERVER / Vending Machine (n)

An example Message:

“Message: 1 from Martin: The Pepsi Bay is empty”

The Message number (1) and the Host name (Martin) together forms a unique Message and makes it easy to differentiate your message from other VMs on the network server and the number helps to find the recent ones.

# Data Protocol

# TASK1

GET /[VM.htm?ITEM=2&COUNT=5](http://10.13.33.91/VM.htm?ITEM=2&COUNT=5) HTTP/1.1



**TCP HTTP Port 80**

Sends the request “**GET /MY\_Data.xml HTTP/1.1**” for every 500ms



PC HTTP-Client-1

**TCP HTTP Port 80**

HTTP/1.1 200 OK

Sends the MY\_Data.xml file to update the web page with current data



PC -HTTP-Client-1

**TCP HTTP Port 80**

PC HTTP-Client-1



VM-1

# 

|  |
| --- |
| VM-1 |

VM-1

# TASK 2



**BAY Empty Message**

**BAY Empty Message**

**BAY Empty Message**

VM-1

CLASS SERVER

VM-2

VM-3

# Application Implementation

The Vending Machine demo (Task 1 and 2) is implemented using “**SAM E70 Xplained Ultra Evaluation Kit** “and “**OLED1 Xplained Pro Extension Kit”.** The OLED Xplained prohas three buttons, LEDs and an OLED.

* The VM application uses Button 1(Down) and 2(UP) to scroll through the VM items and Button 3 to select an item from the Vending Machine.
* The LEDs above the button indicates a button press with a blink.
* The OLED is used to display the name and their corresponding number of the items in a Vending Machine.
* When select button(B3) is pressed the HTTP-Server will decrement the count of an item and updates the HTTP-Client(Web page)through ethernet interface.



E70 WEB SERVER

/Vending Machine

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

The Vending machine demo runs the below user applications in conjunction with MPLAB Harmony TCP/IP Stack:

* **OLED and Button Contoller** (MMI\_Tasks): Manages the operation of the OLED display , Buttons and LEDs.
* **Application Contoller** (app\_Tasks()): manages all high level network communications with client and server.

# 

TCPIP\_STACK\_Tasks ()

APP\_Tasks ()

SYS\_CMD\_TASKS()

SYS\_FS\_TASKS()

DRV\_MIIM\_TASKs ()

NET\_PRES\_Tasks ()

SYS\_FS\_TASKS()

NET\_PRES\_Tasks ()

# Objectives

In Lab 1, you configured a new MPLAB Harmony Project from scratch that included the TCP/IP Stack middleware and a LED Flasher. The project you created in Lab 1 will be used as the starting point for Lab 2.. The source code for the OLED and Button Controller are included into the project.

You will learn about several Harmony TCP/IP API functions by adding the necessary APIs into the application source code to manage the TCP Sockets and data exchanged with socket. You will also learn how the dynamic variables in a web page are handled by the server. Finally, you will get to use the Packet Sender software tool to perform isolated testing of the embedded application prior to connecting your VM implementation to the server on the network.

# Lab Procedure

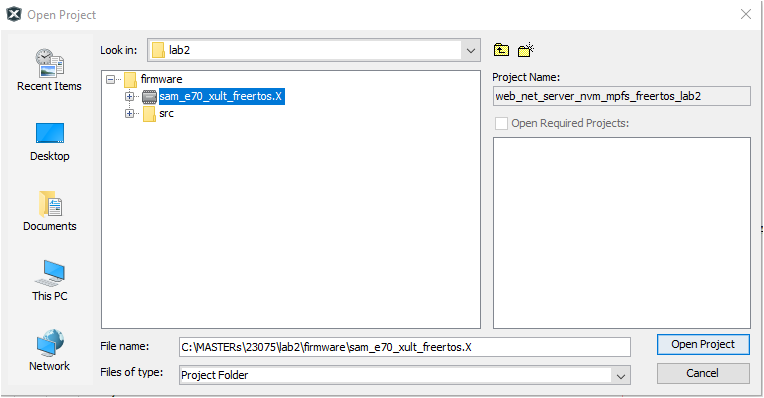
## Project Setup

* 1. Close Lab 1 project by choosing File⮞Close Project(lab1) in the main menu.

|  |
| --- |
|  |

* 1. The project for Lab 2 has already been setup in advance. The project is a working implementation of Lab 1 that has been renamed to web\_net\_server\_nvm\_mpfs\_lab2, and has a number of files added including mmi.c and modified app.c file are added to the project. To open the Lab 2 project, choose File⮞Open Project in the main menu.

|  |
| --- |
|  |

* 1. In the Open Project window, enter C:\MASTERs\21070\ web\_net\_server\_nvm\_mpfs\_lab2\firmware into the File name text box or navigate to the file .
  2. Click on sam\_e70\_xult\_freetos.X icon in the file list.
  3. Press Open Project.

|  |
| --- |
|  |

* 1. The source and header structure for the project is shown below.

|  |
| --- |
|  |

## MHC Code Generation

* 1. The project configurations are already set, and we are going to look on the configurations and generate the code. (steps 1.8 -1.22)
  2. Before the MHC Configuration tool can be used, you need to set lab2 as the main project, by right clicking on the web\_net\_server\_nvm\_mpfs\_lab2 folder under the Projects window, and choosing Set as Main Project in the popup menu.

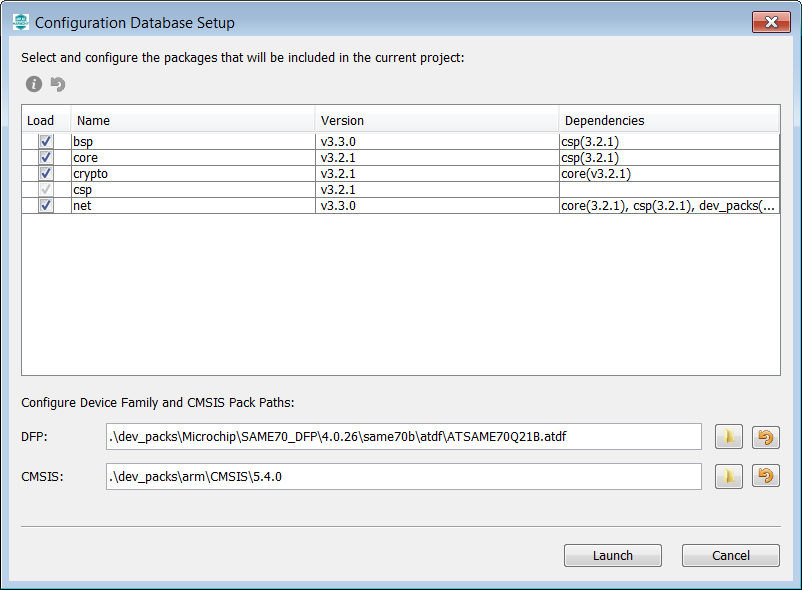
|  |  |
| --- | --- |
|  |  |

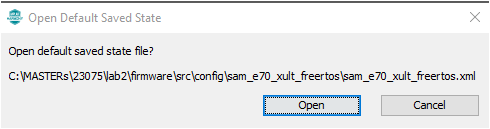
* 1. To launch MHC, choose Tools⮞Embedded⮞MPLAB Harmony 3 Configurator in the main menu.

|  |
| --- |
|  |

* 1. In the MPLAB Harmony Launcher dialog box, make sure the path states   
     C:\MASTERs\23075\h3 and click on Launch.

|  |
| --- |
|  |

* 1. In the nest window the H3 parts and their used Version Number are displayed. Click on Launch
  2. Open the saved state file



* 1. Explore and look on the configurations tree by clicking on the modules under Active Components (step 1.14 and 1.15- can be skipped and continue from 1.16).
  2. Click and check the “**Core”** option under Active Components. On right hand side the Configuration options are displayed, check the “Number of Applications “options under Configuration Options which is set to 2.Expand both Application 0 and 1 Configuration trees, and the Application Name is set to app (Application Interface) for Application 0, and mmi(Man Managed Interface for buttons, leds and OLED)for Application 1.
  3. Check on TCP/IP Stack under Active components and its configuration under Configuration options.

|  |
| --- |
|  |







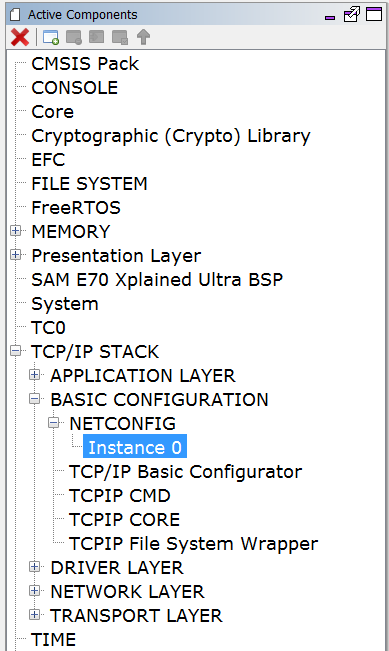


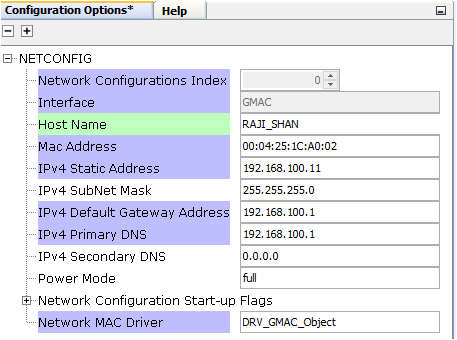


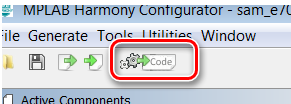
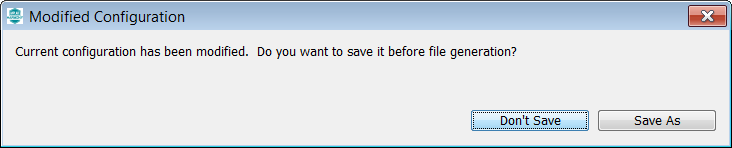
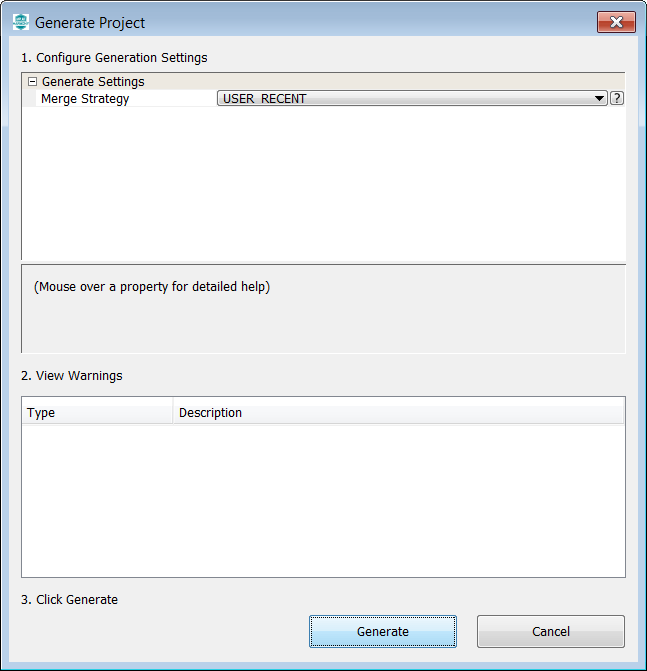


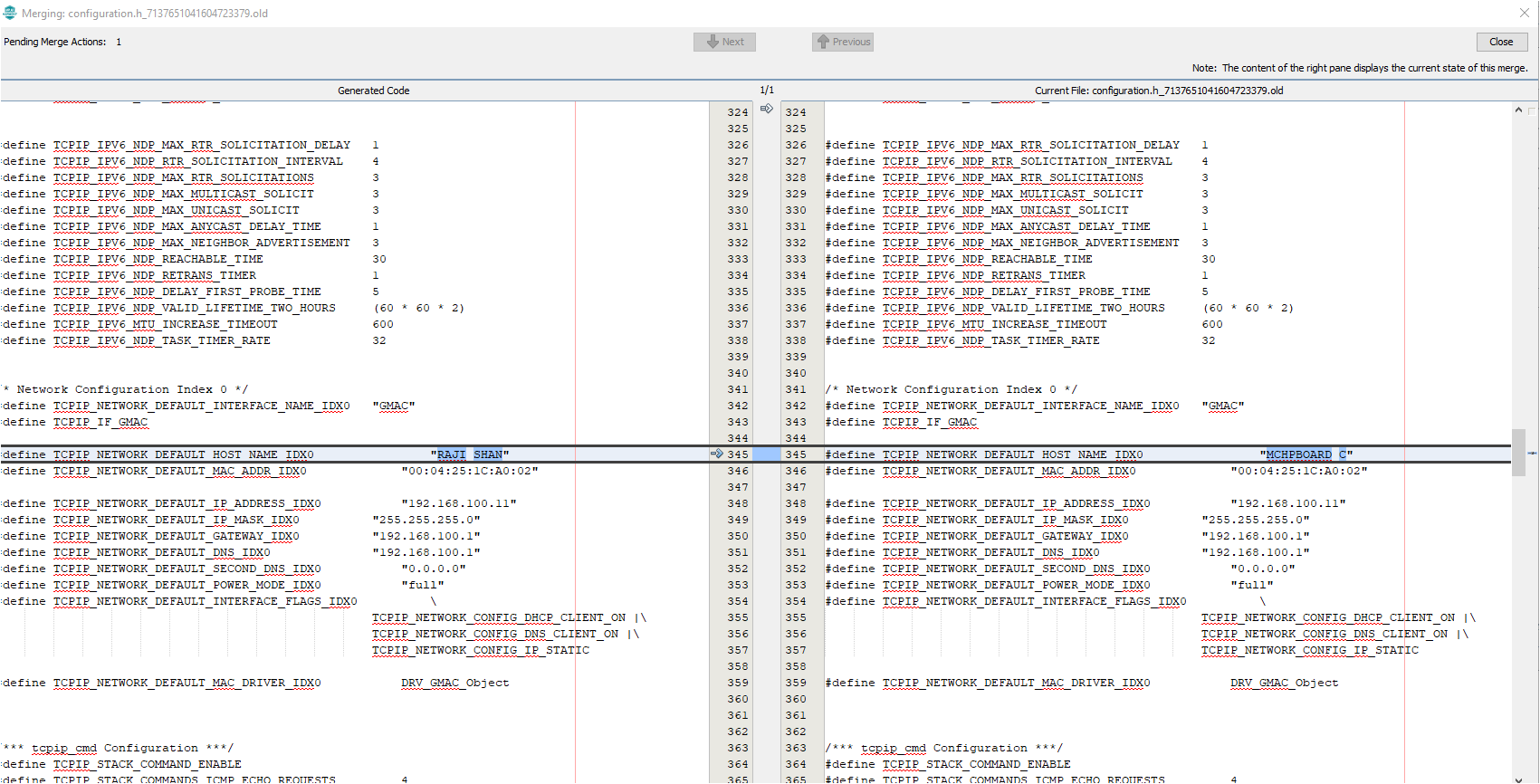


* 1. Select the **Instance 0** in the **TCP/IP stack** under **Active Components** to enter the HOST Name **(TCP/IP STACK->BASIC CONFIGURATION->Instance 0).** And change in the **Configuration Options** (displayed on the right side of MPlabx) the Host Name to your firstname\_lastname. The Host name can be identified in the Network.

****

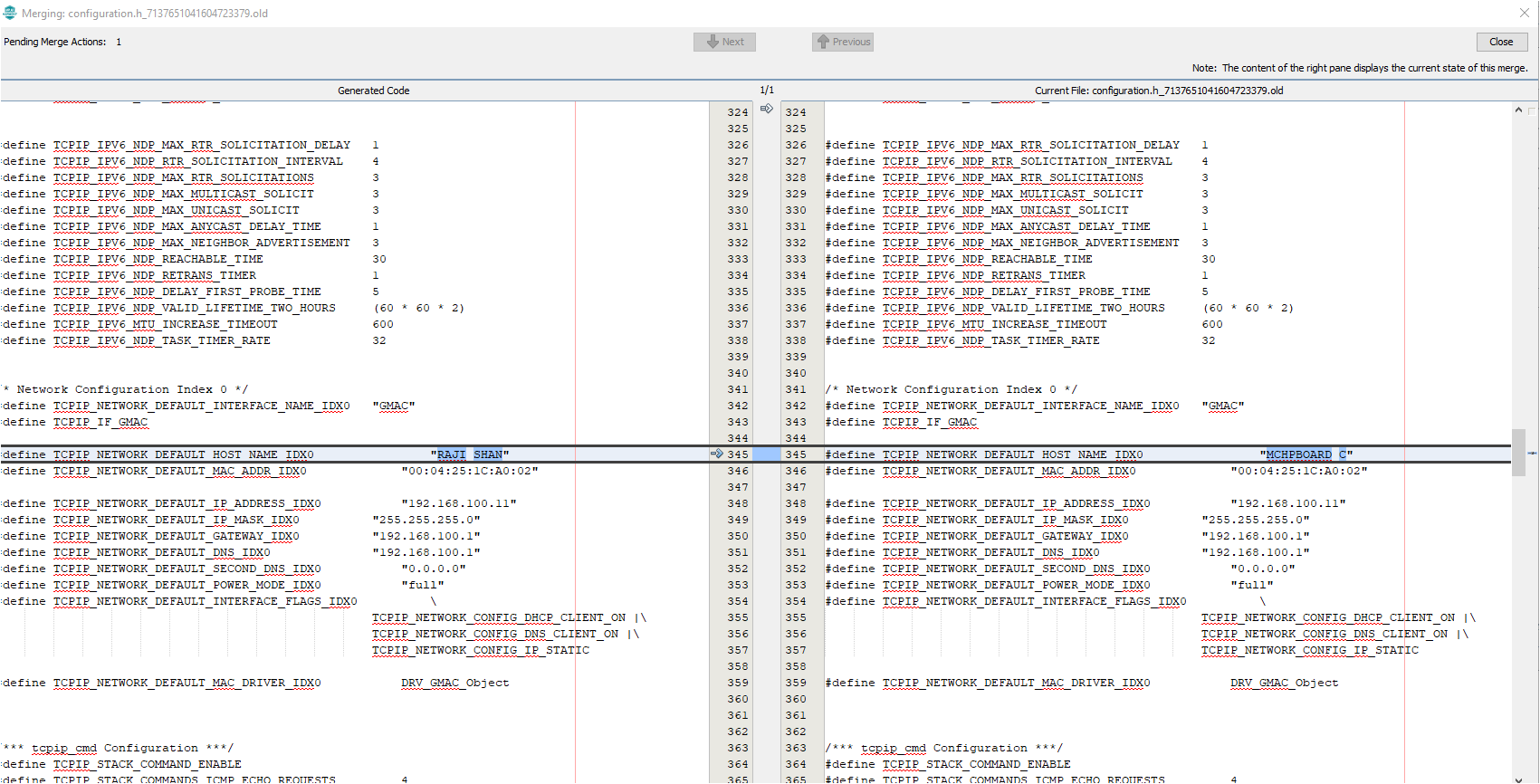


* 1. Select “Code” (Generate Code) which is under the menu tabs.  
     
  2. Select Don’t Save  
     
  3. Select Generate  
     
  4. Some Files will be changed, and the MHC is asking in a “diff” window, if the changes should be taken over. Don’t accept the changes for custom\_http\_net\_app.c file just click on close(step 2 in pic).Accept all changes for the other files by clicking on the Arrow in the middle and then close.



1

2



1

2

.

* 1. The whole process is displayed with a progress bar  
     











* 1. After the MHC has finished generating the project, go to the Projects Window and expand the Header Files and Source Files to see the source/header files for the *app and mmi* application file and TCP/IP stack files.
* The WEB page source code for the VM application is found under the folder web-pages->VM.htm
* The hex file for the web page is in mpfs\_net\_img.c which is generated using the utility MPFS- generator which comes along with Harmony net package.

|  |
| --- |
|  |





















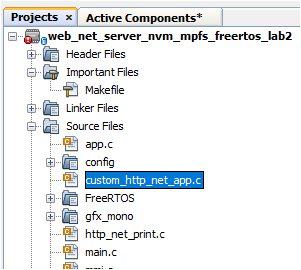


## Application Source and Header File SetupTASK1:

In the VM application when an VM ITEM update is initiated by the HTTP- client (WEB PAGE) it sends out a GET command that needs to be processed by the HTTP-Server. The get command is sent along with the argument of the VM “Item” to be updated and its “Count” which are processed by the HTTP-Server. In Harmony TCP/IP stack GET command is handled by the function TCPIP\_HTTP\_NET\_ConnectionGetExecute ().

Follow the procedure under to include the code that process the GET request from HTTP-CLient: -

* 1. Click and Open the file custom\_http\_net\_app.c. Search for the function TCPIP\_HTTP\_NET\_ConnectionGetExecute () (Approx. Line.No. 247) in which the below code snippets need to be inserted.



* 1. Find the TODO[1] and insert the below code starting from the line mentioned as ***“//<--Insert the solution for TODO[4] starting on this line”*** . Code can be inserted either by typing or copy and paste which is provided for your convenience at the end of the file (custom\_http\_net\_app.c).

ptr = TCPIP\_HTTP\_NET\_ArgGet(httpDataBuff, (const uint8\_t \*)"ITEM");

* 1. Find the TODO[2] and insert the code snippet starting from the line mentioned as ***“//<--Insert the solution for Item 2 starting on this line”*** . Code can be inserted either by typing or copy and paste which is provided for your convenience at the end of the file (custom\_http\_net\_app.c).

ptr = TCPIP\_HTTP\_NET\_ArgGet(httpDataBuff, (const uint8\_t \*)"COUNT");

VM\_Count[Update\_Item]=\*ptr;

## TASK 2:

When a Bay of the VM is empty it sends out the message to the common server. The sate machine to open a socket to send message and close socket is shown below

**Stack Initialized**

**NO**

**YES**

**File Mounted**

**No link found**

**If new link found**

**Stack not Initialized**

**If MAC address is not read**

**MAC address READ**

**If WEB file is not mounted**

**not mounted**

READ

MAC address

TCP/IP stack Int

**Link is established**

**Connection Established**

**If wait time out**

Send out the TCP message

“Bay is empty”

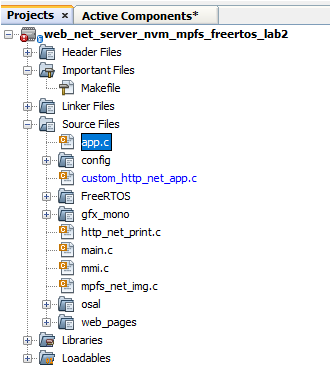
Close connection

Set-up TCP client

Display the IP address

In this section, you will gain some experience with the use of Harmony TCP/IP API functions. The app.c source is missing lines of code. All missing code specifically relates to management of the setting up the TCP Client such as opening or closing the socket, checking if the socket is connected, checking if data is available, and writing data to the socket. Your task is to read the description for each missing item, select the appropriate Harmony TCP/IP API, and fill in the missing line of code.

* 1. The first step to do is get the **common server’s IP address**. Get the address from the presenter and make a note.
  2. Click and Open the file app.c. and find the function APP\_Tasks (). (Approx. Line.No. 200) in which the below code snippets are inserted.



* 1. Find the TODO[3] in which common SERVER IP address is entered. Replace the x with the common server IP address ().

For example: if the server IP address is 192.168.0.108 replace

\* x1 with 192

\* x2 with 168

\* x3 with 0

\* x4 with 108

* 1. Find TODO[4] and add the below snippet to Open a socket for TCP\_Client . Code can be inserted either by typing or copy and paste the code which is provided for your convenience at the end of the file (app.c).

appData.socket = TCPIP\_TCP\_ClientOpen(IP\_ADDRESS\_TYPE\_IPV4,

appData.port,

(IP\_MULTI\_ADDRESS\*) & addr);

* 1. Find TODO[5] and add the below snippet to check the TCP connection is established 0.

Insert the code in the line mentioned as ***“//<--Insert the solution for TODO[5] starting on this line”***

Code can be inserted either by typing or copy and paste the code which is provided for your convenience at the end of the file (app.c).

if (!TCPIP\_TCP\_IsConnected(appData.socket))

* 1. Find TODO[6] and add the below snippet to check the TCP connection is established .

Insert the code in the line mentioned as ***“//<--Insert the solution for TODO[6] starting on this line”***

Code can be inserted either by typing or copy and paste the code which is provided for your convenience at the end of the file (app.c).

if (TCPIP\_TCP\_PutIsReady(appData.socket) == 0)

* 1. Find TODO[7] and add the below snippet to send the message to the common server once a TCP connection at port 80 is established .

Insert the code in the line mentioned as ***“//<--Insert the solution for TODO[7] starting on this line”***

Code can be inserted either by typing or copy and paste the code which is provided for your convenience at the end of the file (app.c).

sprintf(buffer, "MSG:%d from %s : %s is empty", (int) MessageCounter++, (char \*) TCPIP\_HOSTS\_CONFIGURATION[0].macAddr, (char \*) VM\_Items[bay\_index - 1]);

SYS\_CONSOLE\_PRINT("Sending message: %s\r\n", buffer);

TCPIP\_TCP\_ArrayPut(appData.socket, (uint8\_t\*) buffer, strlen(buffer));

* 1. Find TODO[8] and add the below snippet to close the TCP Client socket.

Insert the code in the line mentioned as ***“//<--Insert the solution for TODO[8] starting on this line”***

Code can be inserted either by typing or copy and paste the code which is provided for your convenience at the end of the file (app.c).

TCPIP\_TCP\_Close(appData.socket);

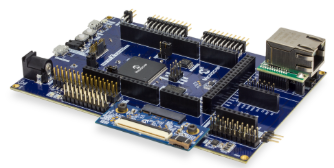






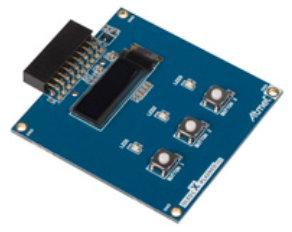
### Hardware Setup

This hardware setup is common to both TASK1 and TASK2. After adding the required code for the application, make the hardware connections to programme the device and to see the output

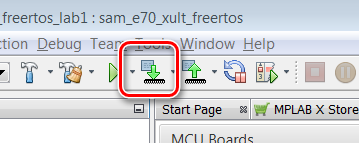


CLASS ETHERNET NETWORK



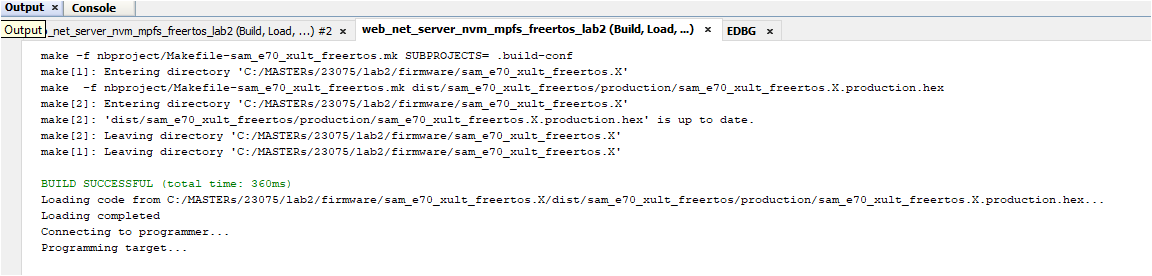


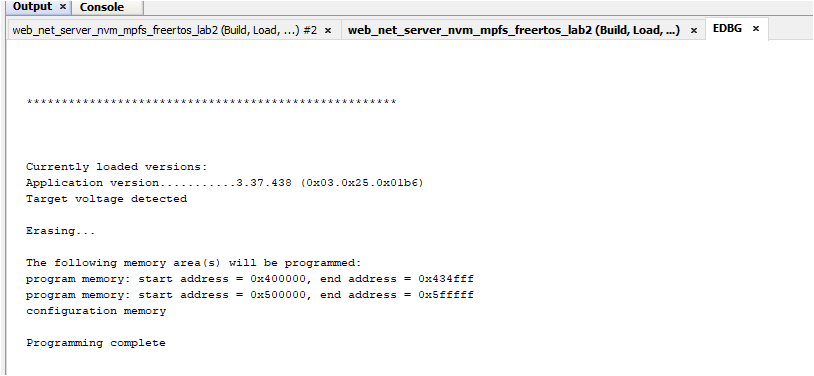
|  |
| --- |
| * 1. Network Connection   Cable: CAT5 Ethernet Cable supplied with Starter Kit  Connection: RJ45 Jack from the class network to PCB Top RJ45 Jack(make sure the jumper below the Ethernet PHY module is taken off) |
| * 1. Programming and Console Connection   CableUSB Male A to USB Male B Micro cable  Connection: USB Debug Port on PCB Top to Laptop USB Port |
| * 1. OLED1 Xplained Pro Connection   OLED1 X PLAINED PRO  Connection: Connect the OLED x PLAINED PRO to the EXT 2 of the board (Extension close to Ethernet PHY)   * 1. PC   PC needs to be connected to the Class network or  It can be connected to the wireless network |

* 1. In the main window of MPLABX, click on the “Make and Program Device” button  
     

|  |
| --- |
|  |

* 1. After successful build, the SAME70 is programmed automatically. Build and Programming are monitored on the output window at the bottom of the MPLAB X IDE screen.





* 1. Immediately after programming, you can see the text on the OLED screen in series like the screens below.

Use the Buttons 1 and 2 (Previous and Next item) on the OLED1 Xplained pro to scroll through the items of the Vending Machine.

Welcome to

23075 IoT6

Vending

Machine

B1-UP, B2-Down

B3-Select

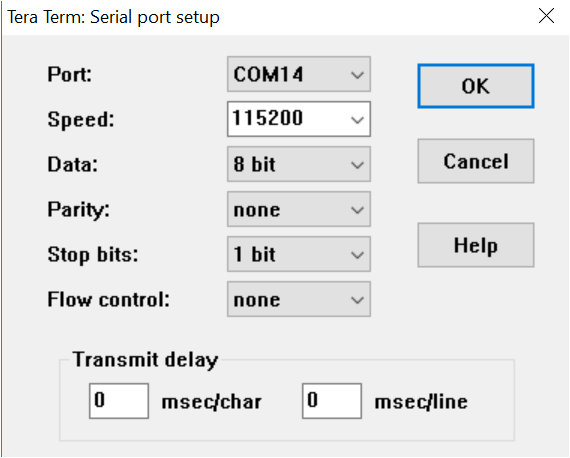
1.Coco-Cola

9

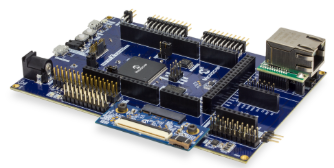
* 1. Now open a Tera Term to see the status of the HTTP server and its IP address.

If Tera Term is still open from Lab 1, click on the window title bar to make it active. If Tera Term needs to be re-opened and configured, refer the below steps.

Open Terra Term Terminal Program  and select under” **Setup->Serial Port”** (in this case a COM14, but could be a different COM Port number in your case) set the speed to 115200.Match the below set-up and click “OK”



* 1. Reset the Board and let the Firmware restart.

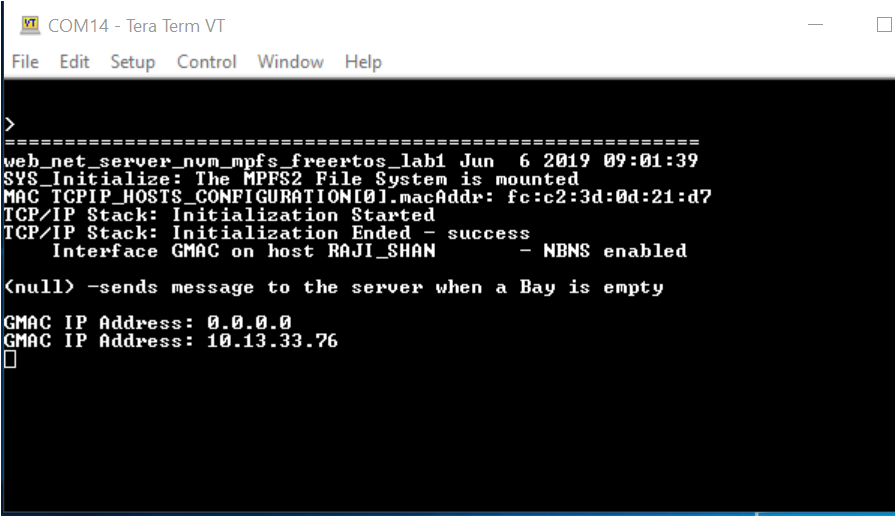


RESET SW401

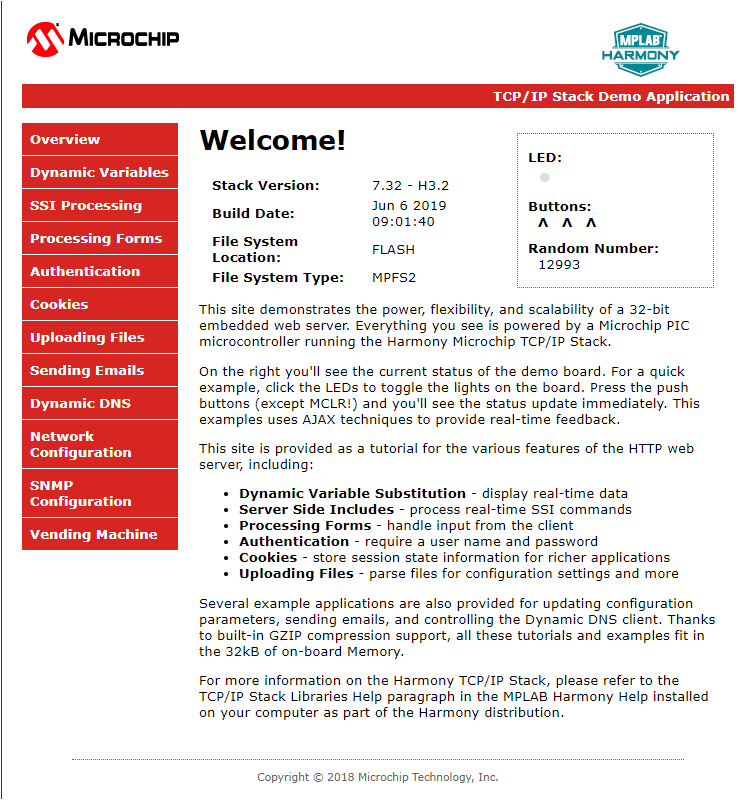
|  |
| --- |
|  |

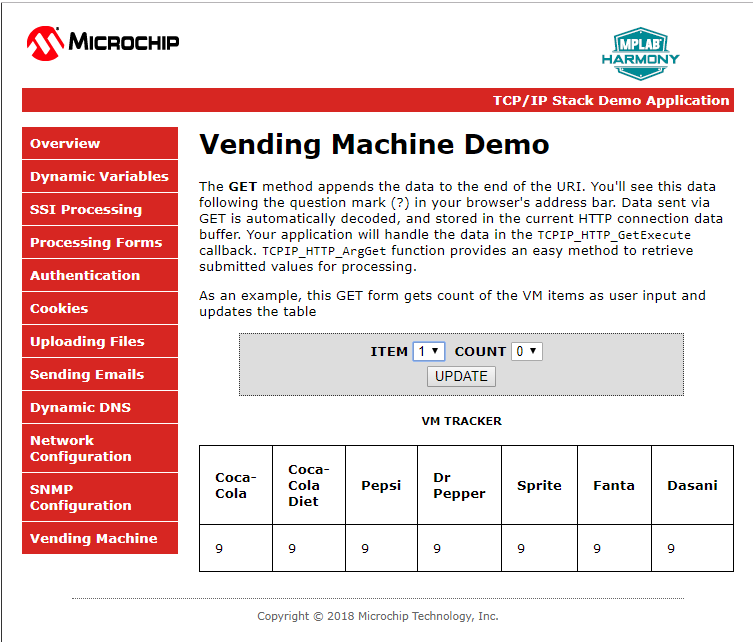
* 1. After some small time, the IP Address should change from 0.0.0.0 to a valid address. Take time to go through the Initialization messages, MAC address, Host name and the IP address of the HTTP- server displayed on the tera term terminal.

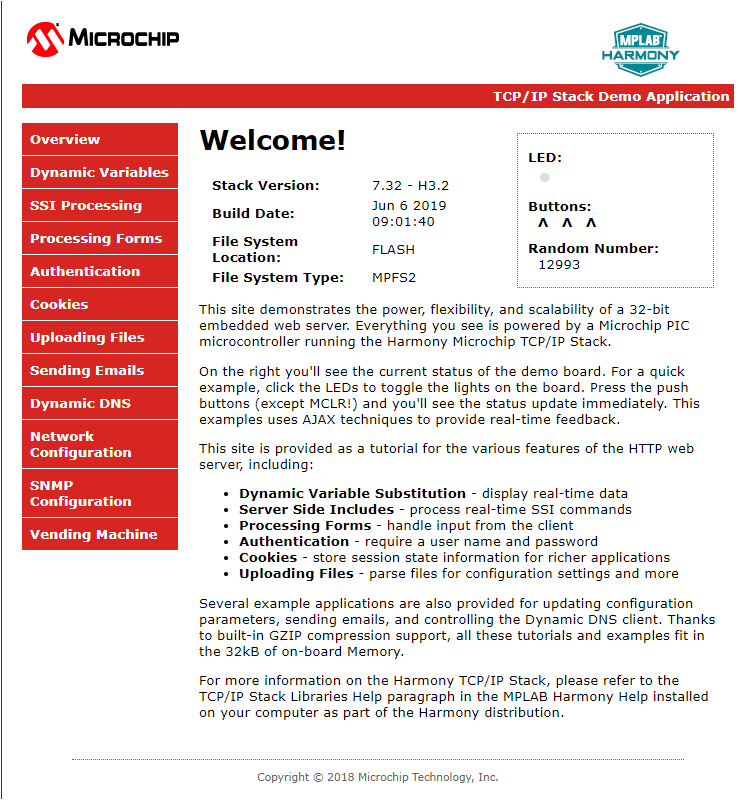
|  |
| --- |
|  |



* 1. Open a web browser and enter the HTTP- Server IP address that you got on the tera term in the previous step. This opens a web page (HTTP-Client)



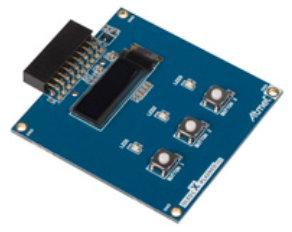
* 1. Click on the Vending Machine tab to see the page for the Vending Machine Demo.



|  |
| --- |
|  |

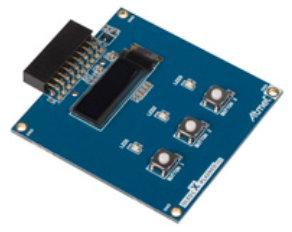
|  |
| --- |
| * 1. Use the Button 3 on the OLED to pick an item from the VM. The count of the corresponding item on the OLED scren will be decremented by 1 and is reflected on the Web page as well as on the OLED screen. |

* 1. Check on the VM update. Pick an item and the count to be changed on the WEB page from the **ITEM** and **COUNT** drop down menu. Select update. When an update is selected the WEB page will send a GET command along with the ITEM number and COUNT to the HTTP-server. The HTTP-server decodes the GET command and displays it on the VM diaply (OLED).



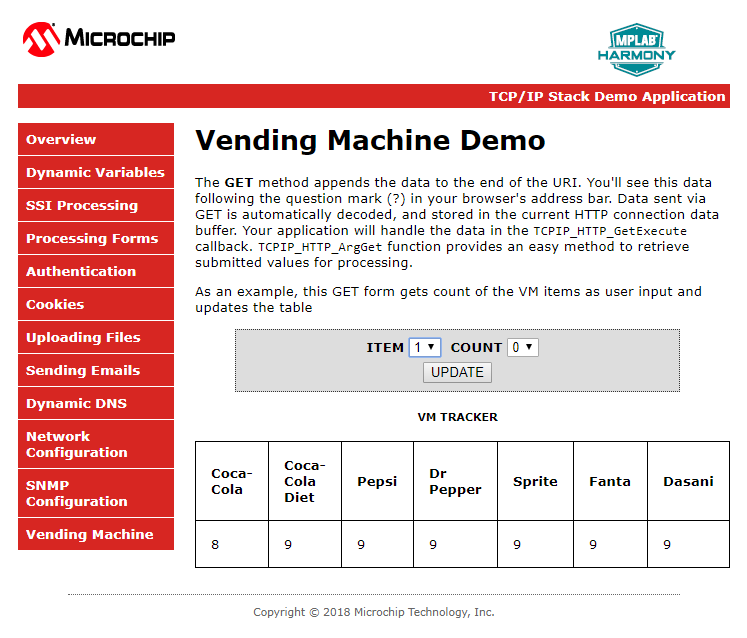
1.Coco-Cola

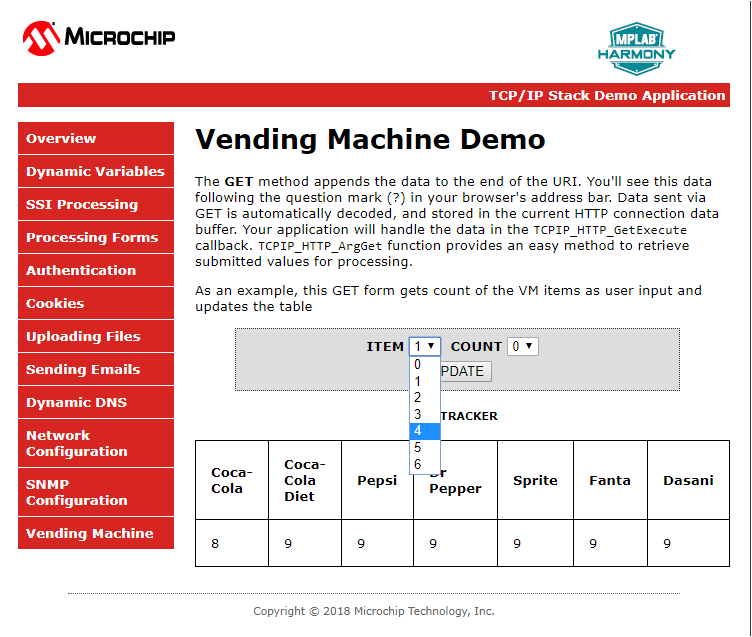
9

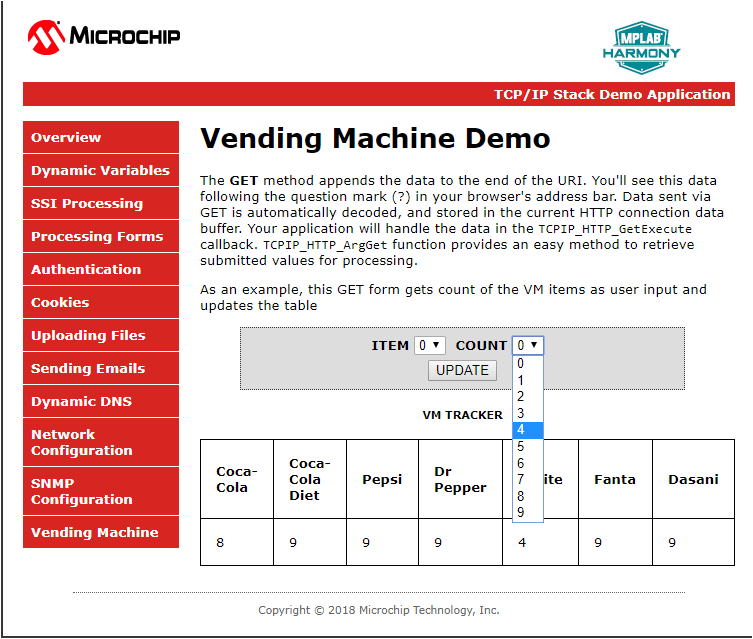


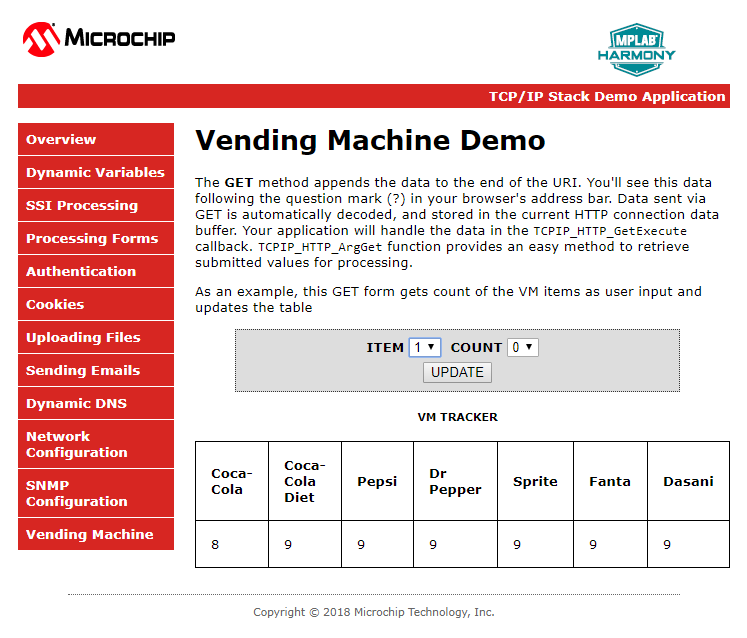
1.Coco-Cola

8



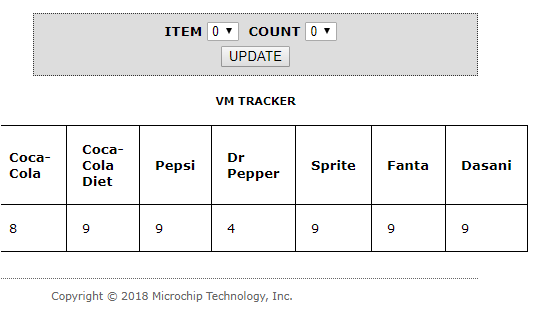






Example:

On the web page ITEM 4 and its count will be changed to 4 and, also on the OLED.



4

4. Dr Pepper

You have successfully completed TASK 1 and let’s move on to TASK 2.



* 1. To test Task 2, one or more Bay of the VM machine must be Zero. Press the select Button 3 and make one or more item to zero count. After few second you can see the messages ” Sending VM status “ , “Starting Connection” and “TCP Socket Connected” and the “BAY Empty message” and TCP client closed on the tera term.
  2. Look at the presenter’s screen to identify your message.

MSG:338 from fc:c2:3d:0d:21:d7 : 2.Diet-Coke is empty

Message No. MAC address which is unique for each VM Bay empty message

































































































































































































































































































































































