

# Microchip PIC32 Ethernet Reference Designs

Dr. Richard Wall – Professor

Department of Electrical and Computer Engineering

University of Idaho

Moscow, ID 83844-1023

Revised: December 20, 2012

[rwall@uidaho.edu](mailto:rwall@uidaho.edu)

## I. Ethernet Reference Designs:

### A. Networking and the Internet

The context of communications networks is an environment that facilitates the exchange of information between two or more independently operating elements.

#### 1. Properties of an Network

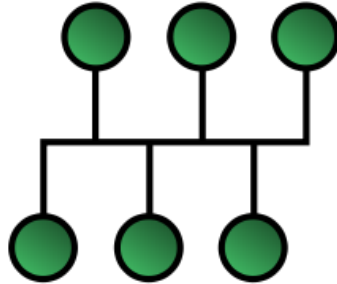
Networks have characteristics that are based upon achieving as many benefits at the lowest cost. Although the network characteristics are distinctly categorized and classified in the following discussion, most realized networks are hybridized combinations of characteristics and configurations.

##### a. Topologies

- i. **Bus network** is where all nodes are connected to a common medium along this medium. Bus networks were common during the era of [party line telephone](#) systems. (This was far before your time.) This was the layout used in the original [Ethernet](#), called [10BASE5](#) and [10BASE2](#). Multiple receivers may be connected to such a network in a linear, [multi-drop](#) configuration. Ethernet hubs (not switches or routers) in effect implement a bus network. Although this hardware is becoming more difficult to find, it is very useful when using [Wireshark](#) or similar network packet sniffing tools for network development and analysis.

From Figure 1, you can see that the communications medium is shared by all elements thus forming a communications bottle neck. This is scenario for both wire based and wireless, based communications. Examples of wire based are [I2C](#), [CAN](#), [SDLC/HDLC](#), [BPL](#), and [RS485](#). Examples of wireless communications are [underwater acoustic communications](#), [Wi-Fi](#), [ZigBee](#) (also see [MiWi](#) and [Xbee](#)).

It is imperative that a device addressing scheme be used along with some type of collision avoidance or collision arbitration to protect the integrity of the information. See [CSMA/CA and CSMA/CD](#). The major advantage to this topology is simplicity low cost, and device failure isolation. Disadvantages include lower bandwidth, collision avoidance and or mitigation hardware/software, and range (physical distance between noder).



**Figure 1. Organization of elements in a bus network.**

- ii. **Star network** or **hub based network** are one of the most common [computer network topologies](#) where the elements are organized as illustrated in Figure 2. In its simplest form, a star network consists of one central [switch](#), [hub](#) or computer, which acts as a conduit to transmit messages. This consists of a central node, to which all other nodes are connected; this central node provides a common connection point for all nodes through a hub. Star network that use classical hubs suffer many of the shortcomings of bus networks except for the range limitations.

[Switches are different than hubs](#). Hubs act a junction box where each connection input is connected to all connection outputs. Thus, a hub can only connect two elements at a time unless the communications is to be broadcasted to all elements. A switch can simultaneously connect multiple pairs of communicating elements.

The chief advantage of a star network configuration is greater bandwidth by allowing multiple device pairs to communicate through the hub element. The disadvantage is that the hub or switch represents a single point of failure.

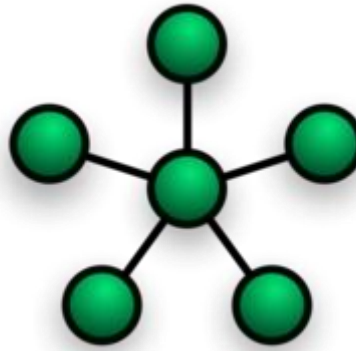


Figure 2. Organization of elements in a star network.

- iii. **Ring network** in which each node connects to exactly two other nodes, forming a single continuous pathway for signals through each node - a ring as shown in Figure 3. Data travels from node to node, with each node along the way handling every packet. Because a ring topology provides only one pathway between any two nodes, ring networks may be disrupted by the failure of a single link. A node failure or cable break might isolate every node attached to the ring.

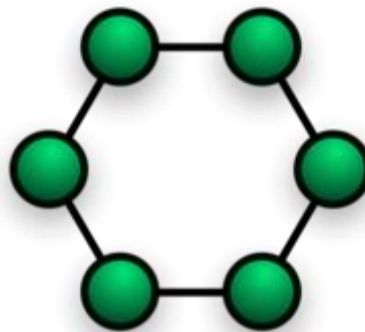


Figure 3. Organization of elements in a ring network.

- iv. **Token ring networks** refer to software that manages when elements can send communications on the network. The elements take turns using a “hall-pass” management scheme to avoid conflicts to gain access to a shared resource. A software token is systematically passed to a predetermined element. Elements are allowed to transmit information only when it possesses the token. During the time an element possesses the token, it can engage with bidirectional communications with any other element in the ringed network. Each element is restricted to the length of time it is allowed to hold the token before it must

relinquish it to the next element in the ring. If an element fails to pass the token in a specified time, the ring is broken and the ring must be rebuilt.

Token ring networks can be implemented on any physical network topology. The chief advantage for token ring networks is that the worst case communications delay is deterministic. One disadvantage is that it takes time to build or repair a broken token ring. The only way for an element to join an existing network is to destroy the token and break the ring. Even if an element has nothing to communicate, it still consumes network bandwidth.

- v. **Daisy Chain or Linear network** has a topology that can be represented by a broken ring network as illustrated in Figure 5. All elements have a two-way link between one computer and the next. However, this was expensive in the early days of computing, since each computer (except for the ones at each end) required two receivers and two transmitters. This topology can be used as an emergency fall back configuration for broken ring networks.

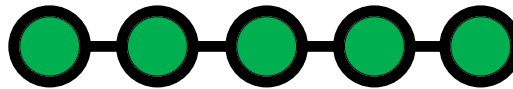


Figure 4. Organization of elements in a Daisy Chain network.

- vi. **Mesh network** is a type of networking where each node (element) must not only capture and disseminate its own data, but also serve as a *relay* for other nodes, that is, it must collaborate to propagate the data in the network. The connectivity of elements are illustrated in **Error! Reference source not found.** through Figure 8.

Cellular telephones and ZigBee are common technologies that employ mesh network. Advantages of mesh networks include dynamic reconfiguration for mobile elements, multiple information routing paths results in robust operation, and fault isolation. In some implementations, information is stored in a relaying element until communications can be reestablished.

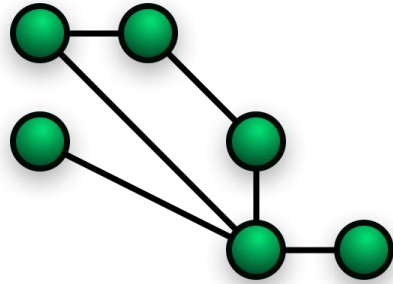


Figure 5. Organization of elements in a mesh network.

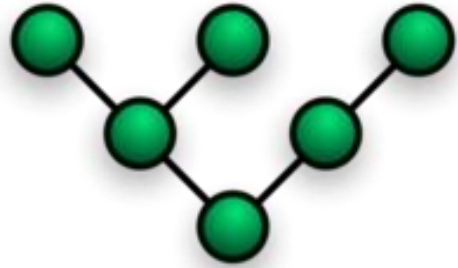


Figure 6. Organization of elements in a tree network.

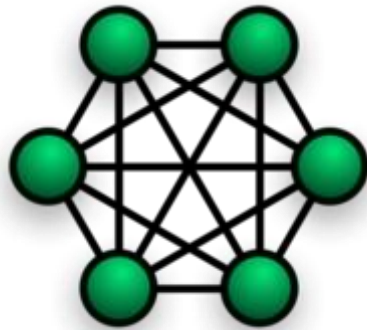


Figure 7. Organization of elements in a fully mesh network.



Figure 8. Organization of elements in a linear mesh network.

It is also possible to implement a mesh network using Wi-Fi by constructing a wireless distribution system ([WDS](#)). I developed one such system for the USDA Agricultural Research Station in Kimberly, Idaho that is logically configured as illustrated in Figure 9. The spatial organization of the equipment is shown in Figure 10.

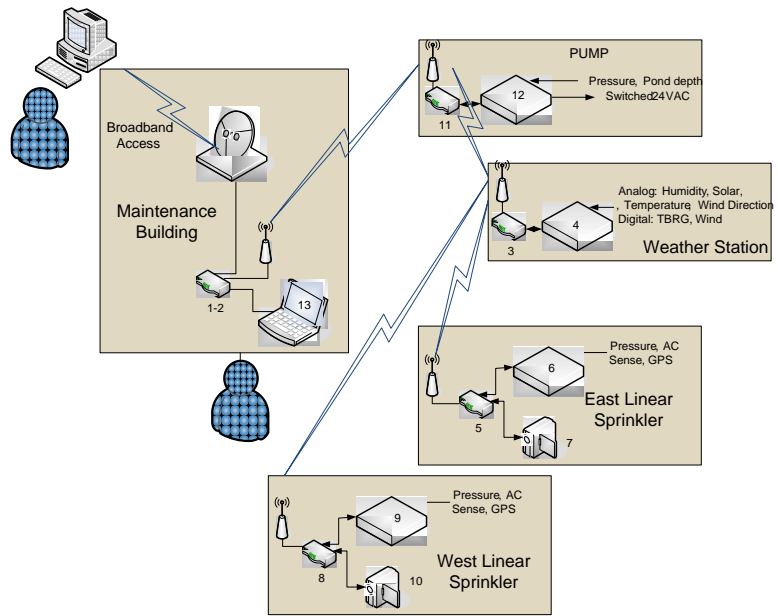


Figure 9. Kimberly South Farm Irrigation Control Architecture

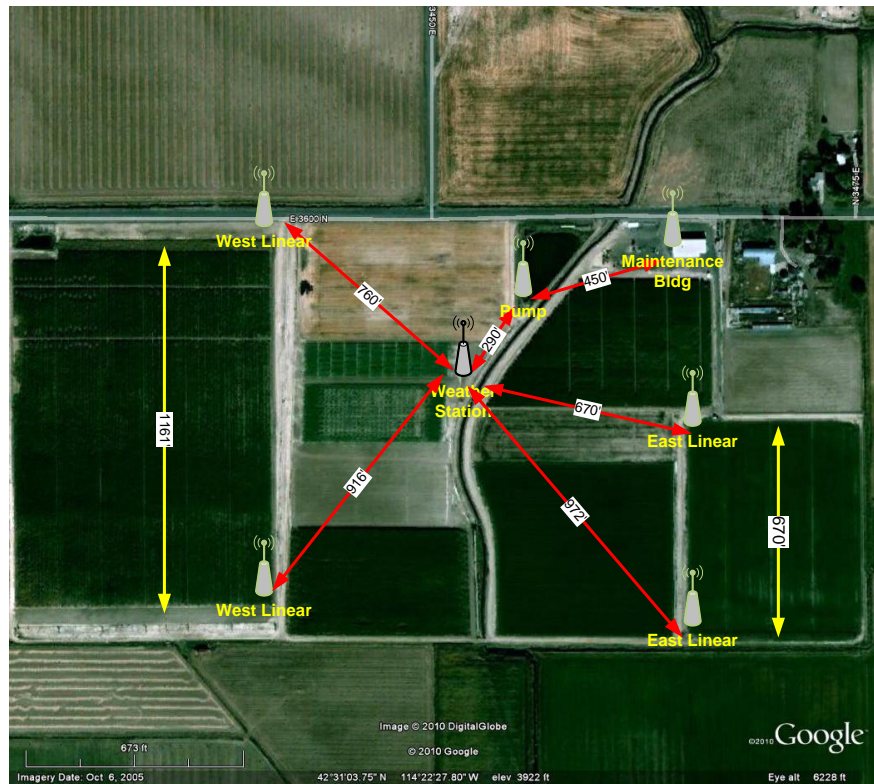


Figure 10. Physical layout of South Farm nodes

- b. **Network Scope** refers to the uses of the network. It involves the amount and type of information, the uses of the information, and the quality of service. For example, email was originally designed to distribute personal text based communications to one or more other individuals. With the advent of attachments to email, the scope now includes large data bases, graphics, and audio files. While the internet has the bandwidth to support the size of the data in a expected period of time, a scope of a network like I2C does not include large data transfers. How a network is to be used and range of uses defines the scope of the network.
- c. **Network Scale** is the size of a network in numbers of nodes that any single node can ultimately communicate within a degree of timely response. In some cases, it is the maximum number of nodes or elements that can be individually addresses. Another measure of the scale of a network is a measure of the amount of traffic as measured in packets per seconds a network can accommodate in a timely manner.
- d. **Network Range** in a spatial context is the physical distance between elements that a network can support. For self contained network topologies that only point to point communications, the physical constraints of the layer 1 media, transmitter power, and receiver sensitivity dictate the maximum distance between nodes.
- e. **Network Efficiency** encompasses many aspects of communications. The ratio of useful data as measured in specific quanta such as bytes to the total quanta required to deliver the information to the intended receiver.
- f. **Network Speed** is the time needed for information to travel from the sender to the receiver. To some extent defines the time a communications spends on the layer 1 media. In the internet environment, this includes the latency time needed for routing and handshaking for insurance of delivery. Typically, it is measures in bits per second or bytes per second.
- g. **Network Security** defines the capability to detect and manage information that has been intentionally or inadvertently modified or compromised.
- h. **Network Interoperability** is a property of a product or system, whose [interfaces](#) are completely understood, to work with other products or systems, present or future, without any restricted access or implementation.

## B. Embedded Networking

- a. **Definition** – Embedded networking is multi device communications in a single system or between multiple systems that can be characterized as an embedded system. An embedded system is a [computer system](#) designed for specific control functions within a larger system, often with [real-time computing](#) constraints. It is *embedded* as part of a complete device often including hardware and mechanical parts.
- b. **Ethernet** - is a family of [computer networking](#) technologies for [local area networks](#) (LANs). Ethernet was commercially introduced in 1980 and standardized in 1985 as [IEEE 802.3](#). Ethernet has largely replaced competing wired LAN technologies.
- c. **Embedded Ethernet** – employs Ethernet networking communications in an embedded application. Typically, low priced microcontrollers are used in the design. Processors such as the PIC32MX7 have hardware integrated inside the processor package that supports Ethernet communications.
- d. **Ethernet Stack** – software that implements protocols commonly used over the internet. The internet layer provides for the delivery of the data gram to the intended receiver(s). Examples of internet transport protocols for which reference designs are provided are: TCP Transaction Controlled Protocol, UDP – User Data Protocol, SMTP – Simple Mail Transfer Protocol (email data grams) , ICMP – Internet Message Control Protocol (ping data grams). These protocols are stacked up as wrappers around the information to be sent from one element to one or more elements as shown in Figure 11.

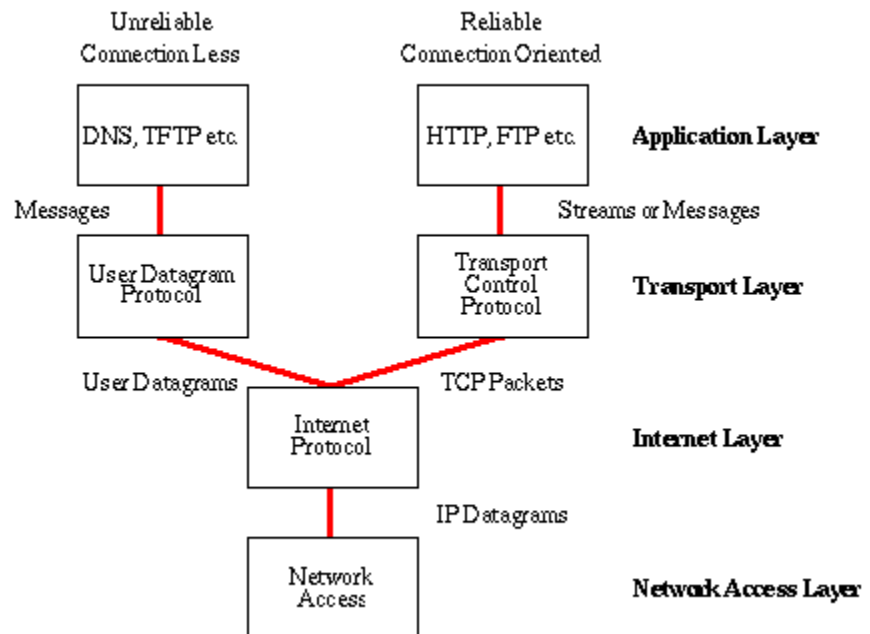


Figure 11. [Internet protocol](#) stack and relates the stack to the ISO OSI reference protocol stack model.



## C. Development Platform General Notes:

The purpose of this document is to provide instructions relating to and the functionality of reference design for the Microchip TCP/IP stack applications implemented on a PIC32MX7 processor. This guide assumes that the user has a general knowledge of internet and Ethernet concepts. It also assumes that the user is familiar with developing web pages using HTML and CSS. Web links to tutorials and application notes are provided at the end of this document.

## D. Hardware Customization Notes:

All reference designs have been tested on a [Digilent Cerebot MX7ck](#) processor board with a [2X16 character LCD](#). The reference designs provided have been either created by the author or are significantly modified versions of the tutorial projects that are provided with the [Microchip Applications Library](#) (MAL). Most if not all references to Microchip processors other than the Microchip PIC32 and C Compilers other than [Microchip C32](#) have been removed to avoid obfuscation by “#if define” segments of code that accommodate over Microchip processors. The specific Cerebot MX7ck hardware interface is defined in the configs/HardwareProfile.h file.

The PIC32 server IP addressed is set by specifying constants in the configs/TCPIP Eth795.h file that is associated with each Ethernet project as described below. The HTML code hosted on the PIC32 is converted to an MPSPImg2.c file using a Java application installed with the Microchip Applications Library package. The design references are designed to be run from a very specific directory structure. Additional details regarding these development activities are provided below.

The processor utilization for the reference designs requires approximately 32K bytes of both Flash and Ram. This will vary depending upon the memory required to store the HTML files. The processor core frequency is set for 80 MHz and the peripheral bus clock is configured for 10 MHz.

## E. PIC32 WEB Page Tutorial Guide

### 1. Resources

#### a. MPLAB 8.88

[http://www.microchip.com/stellent/idcplg?IdcService=SS\\_GET\\_PAGE&nodeId=1406&dDocName=en019469&part=SW007002](http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002)

#### b. mplabc32\_v2\_02a\_windows\_installer

[http://www.microchip.com/stellent/idcplg?IdcService=SS\\_GET\\_PAGE&nodeId=2615&dDocName=en532454&redirects=c32](http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2615&dDocName=en532454&redirects=c32)

#### c. Microchip Applications Library v2012\_10\_15 Windows

[http://www.microchip.com/stellent/idcplg?IdcService=SS\\_GET\\_PAGE&nodeId=2680&dDocName=en547784](http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2680&dDocName=en547784)

- d. Unzip the reference design project code into the directory PIC DEV
- e. Download and unzip into the directory at the same level as Project X  
[http://www.ece.uidaho.edu/ee/classes/ECE443/code/RD15\\_UI\\_WebVend.zip](http://www.ece.uidaho.edu/ee/classes/ECE443/code/RD15_UI_WebVend.zip)

2. **The PIC32 HTTP Development Directory Structure:** The directory structure for a Microchip TCP/IP application is shown in Figure 12. Figure 13 shows the directory for reference design RD18. The files and subdirectory shown in Figure 13 are typical of projects involving Microchip TCP/IP stack applications.

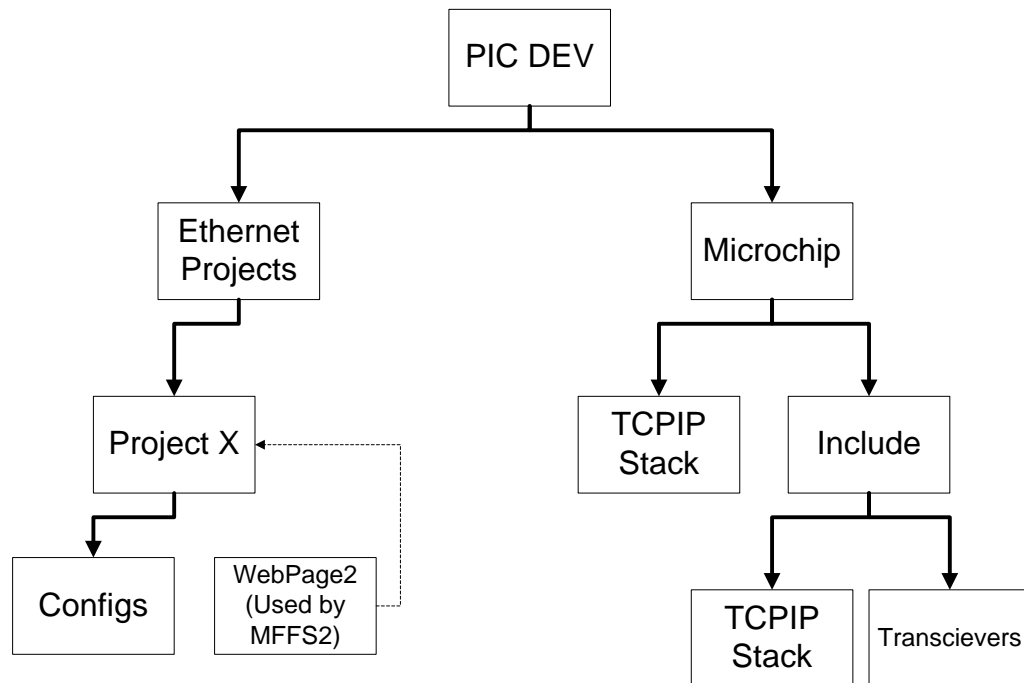


Figure 12. TCPIP development file structure

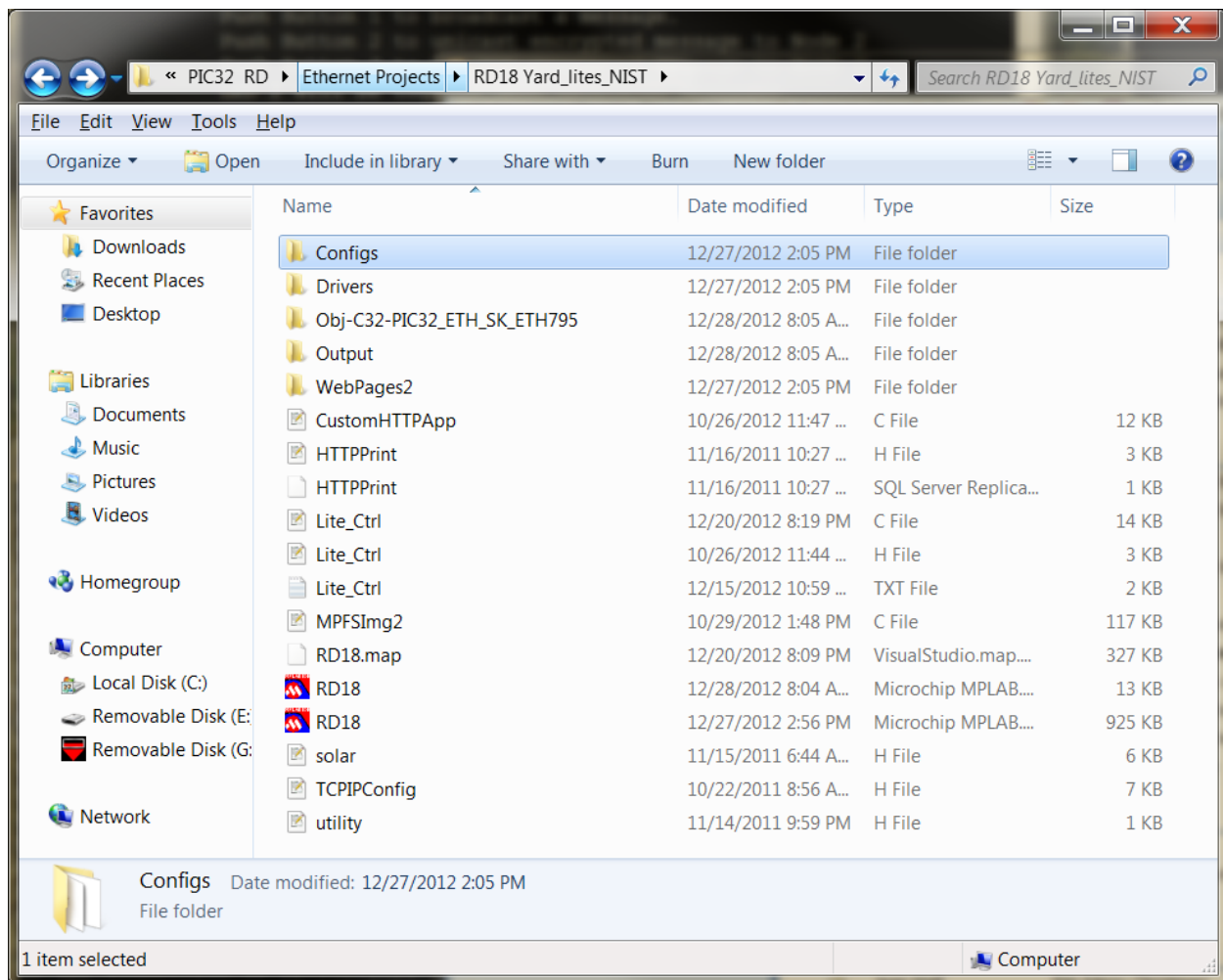


Figure 13. Directory for RD18

3. **Project Build Options:** The project must be setup as shown in **Error! Reference source not found.** through **Error! Reference source not found.** use the following settings:
  - a. Search path Included files is set for:
    - i. The local directory - "."
    - ii. The hardware specific configuration header files - ".\configs"
    - iii. The MAL support files - "..\..\microchip\Include"
  - b. Intermediate directory set for ".\Obj-C32-PIC32-ETH\_SK\_ETH795".
  - c. MPLAB PIC32 C Compiler Preprocessor Macro -  
CFG\_INCLUDE\_PIC32\_ETH\_SK\_ETH795
  - d. MPLAB PIC32 Assembler Preprocessor Macro -  
CFG\_INCLUDE\_PIC32\_ETH\_SK\_ETH795

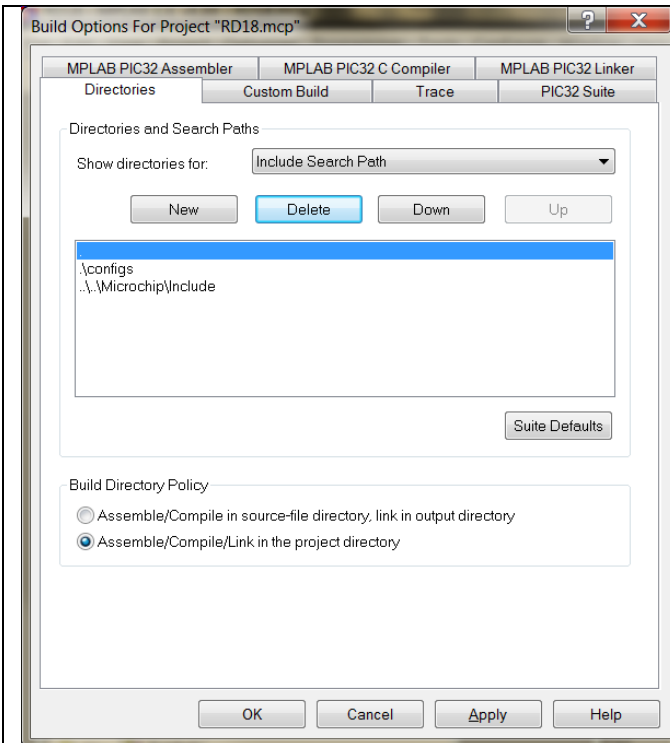


Figure 14. Include search path

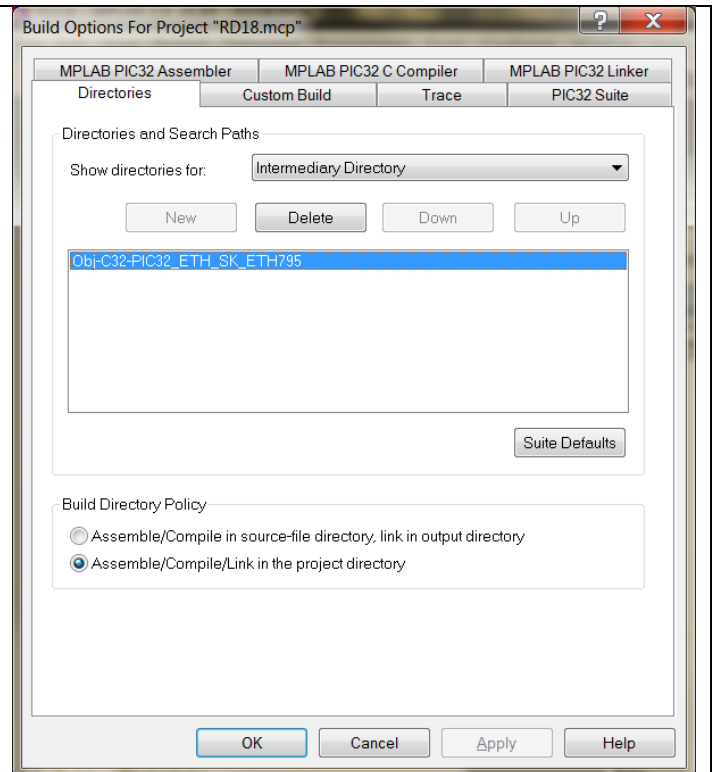


Figure 15. Intermediate Directory path

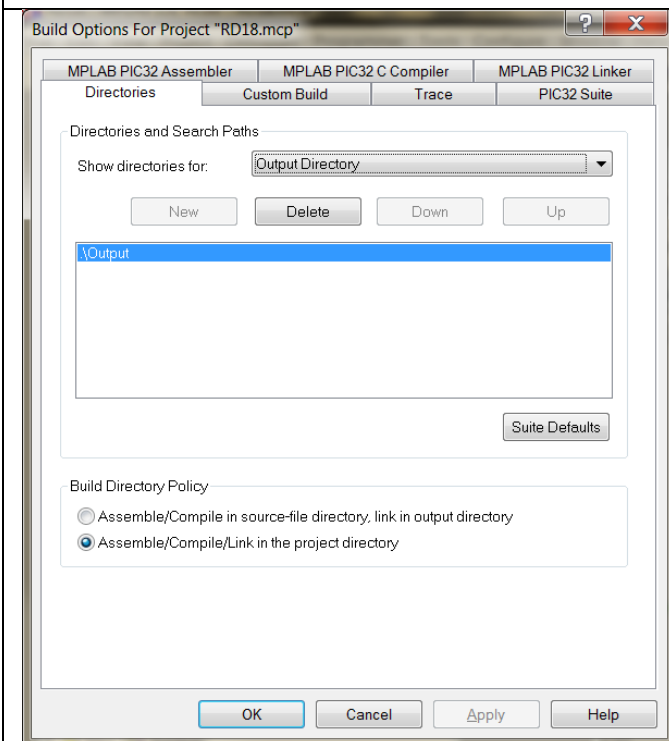


Figure 16. Output Directory

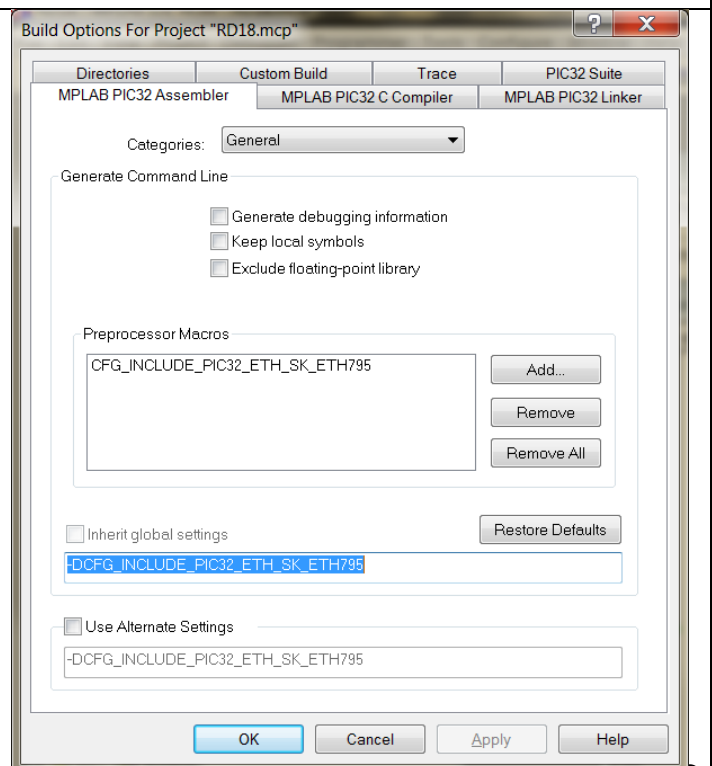
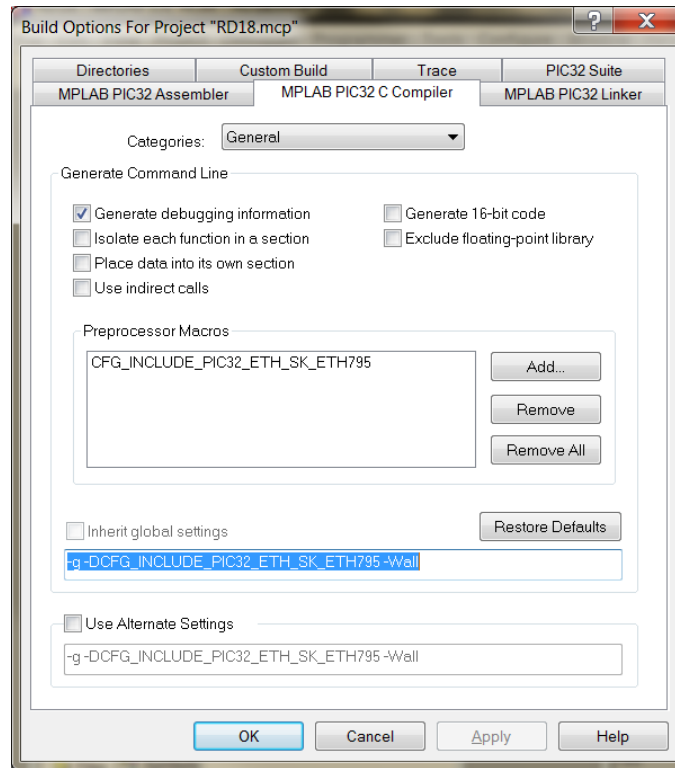
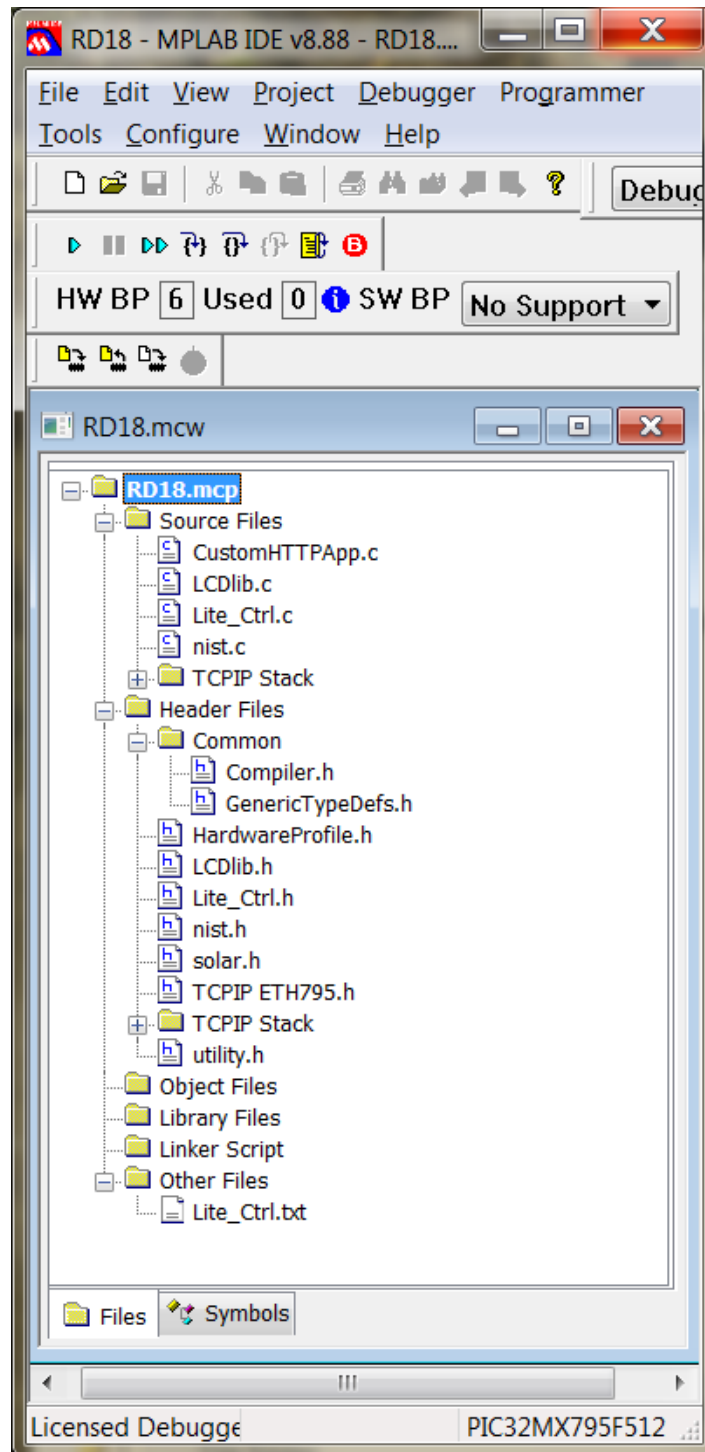


Figure 17. MPLAB PIC32 Assembler Preprocessor Macros



**Figure 18. MPLAB PIC32 Compiler Preprocessor Macros**

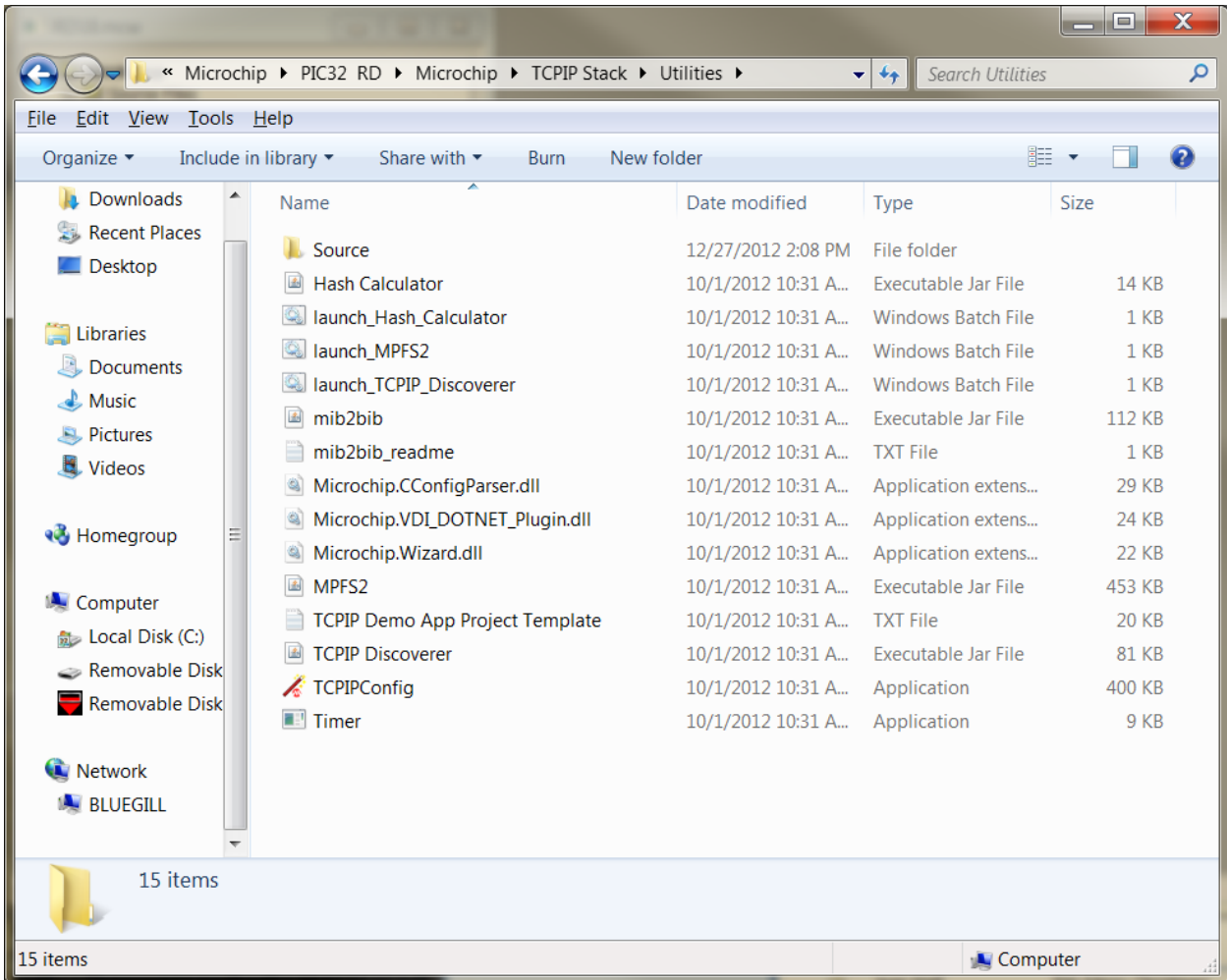
The project window for Microchip TCP/IP stacks applications is shown in Figure 19. The folder labeled “Other Files” are not processed when building a project. Also note that the folder organization does not represent the directory structure where the files are actually located on your computers drive. The structure shown on the project window is strictly for convenience. However, all files listed will be searched to determine if they are required.



**Figure 19. MPLAB Project window for RD18**

- 4. Converting Web page HTML files for PIC32 Server applications**
  - a. Create the HTML file for your application in the project WebPage2 subdirectory.

- b. Remove all files not associated with the web page in the WebPage2 directory such as backup files created by text editors. All files in the WebPage2 directory will be converted to a C file. Extraneous files will just consume PIC flash memory.
- c. Open file PIC DEV/Microchip/TCIP Stack/Utilities/MPFS2.jar in the directory shown in Figure 20.



**Figure 20. Folder location for MPFS2 application**

- d. Follow the steps on the window for the MPFS2 application to see the MPFS2 application shown in Figure 21. Complete the three steps shown following the following procedure.
  - i. For step 1: Set directory for../IP\_PACKAGE/Projects//Project X/WebPage2
  - ii. For step 2: Select "PIC18/PIC32 Image
  - iii. For step 3: Set directory for../IP\_PACKAGE/Projects//Project X – Set Image name for MPFSImg2
  - iv. Click on Generate – This generates the file MPFSImg2.c that must be included in your project file list

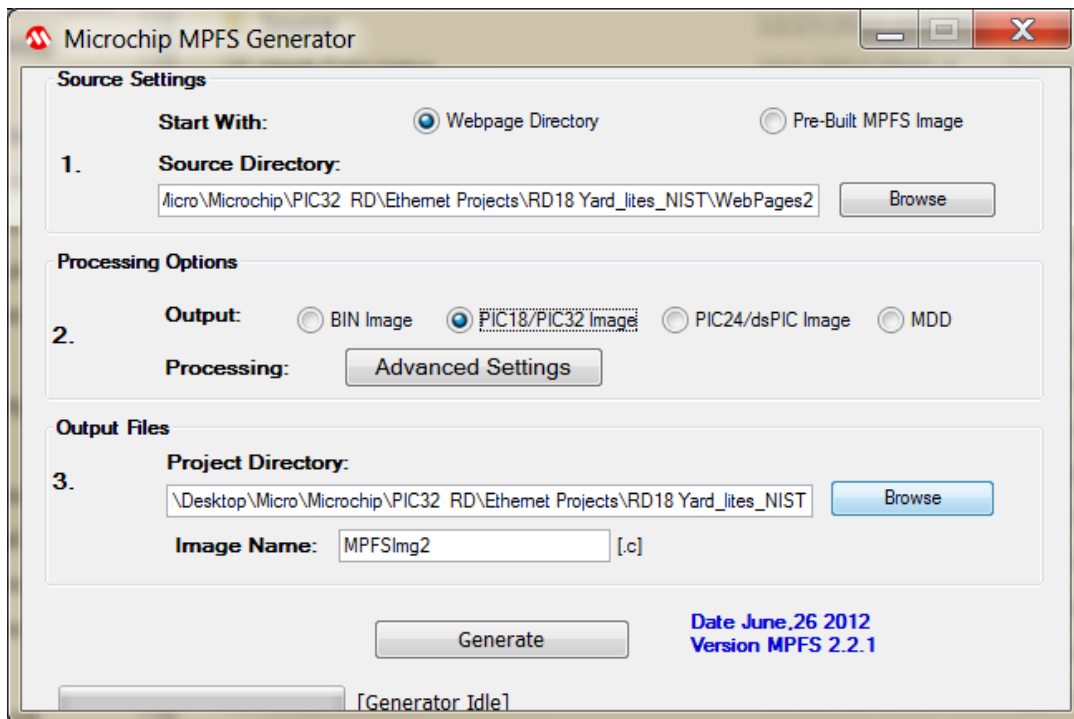


Figure 21. MPFS2 Application window

- v. The report generated by MPFS2 should look like Figure 22.

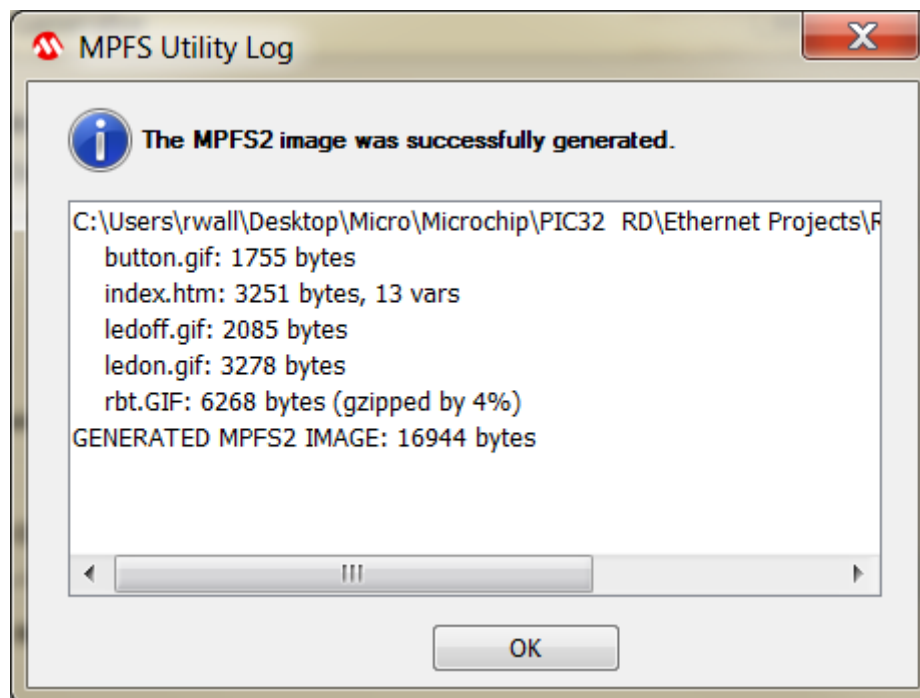


Figure 22. MPFS2 generation results report.



5. **Customizing project for IP Address:** Modification to TCPIP ETH795.h are required.

- a. Application Level Module Selection must include:
  - i. Required - STACK\_USE\_HTTP2\_SERVER
  - ii. Suggested - STACK\_USE\_ICMP\_SERVER
  - iii. Suggested - STACK\_USE\_ANNOUNCE
- b. Transport Layer Options must include: STACK\_USE\_TCP and
- c. Network addressing Options:
  - i. #define MY\_DEFAULT\_HOST\_NAME "PIC32 MX7ck" // less than 16 bytes

ii. Defaults to PIC32 MAC address in silicon:

```
#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      (0x00)
#define MY_DEFAULT_MAC_BYTE6      (0x00)
```

iii. Set the IP Address as required for your specific network environment.

Normally, only the settings for MY\_DEFAULT\_IP\_ADDR\_BYTEx will need to be changed but the rest of the setting should be checked. The specific IP interface for the two locations is set using lines 143 to 166 or 172 to 195.

```
#define MY_DEFAULT_IP_ADDR_BYTE1      (192ul)
#define MY_DEFAULT_IP_ADDR_BYTE2      (168ul)
#define MY_DEFAULT_IP_ADDR_BYTE3      (0ul)
#define MY_DEFAULT_IP_ADDR_BYTE4      (100ul)

#define MY_DEFAULT_MASK_BYTE1          (255ul)
#define MY_DEFAULT_MASK_BYTE2          (255ul)
#define MY_DEFAULT_MASK_BYTE3          (255ul)
#define MY_DEFAULT_MASK_BYTE4          (0ul)

#define MY_DEFAULT_GATE_BYTE1          (192ul)
#define MY_DEFAULT_GATE_BYTE2          (168ul)
#define MY_DEFAULT_GATE_BYTE3          (0ul)
#define MY_DEFAULT_GATE_BYTE4          (1ul)

#define MY_DEFAULT_PRIMARY_DNS_BYTE1   (192ul)
#define MY_DEFAULT_PRIMARY_DNS_BYTE2   (168ul)
#define MY_DEFAULT_PRIMARY_DNS_BYTE3   (0ul)
#define MY_DEFAULT_PRIMARY_DNS_BYTE4   (1ul)

#define MY_DEFAULT_SECONDARY_DNS_BYTE1 (0ul)
#define MY_DEFAULT_SECONDARY_DNS_BYTE2 (0ul)
#define MY_DEFAULT_SECONDARY_DNS_BYTE3 (0ul)
#define MY_DEFAULT_SECONDARY_DNS_BYTE4 (0ul)
```

## F. Reference Design Descriptions

1. **RD11-RD13 Lesson 1-3:** These reference designs are customized versions of the WebVend tutorials that are found at the three tutorials listed in the Tutorials and Applications Notes section provided at the end of this document. The console (serial port connection to a PC terminal program) is not used for these examples although the code can be modified to do so. The LCDblocking function supplied with the MAL is non reentrant. Additional information of the reference designs that describe the operations specific to the Cerebot MX7ck processor board are described in the VendingMachine.txt file that is accessible through the Projects window.
2. **RD11Lesson 0** – An example of a static web page. There are no dynamic variables or control features are implemented. There also no redirection to other web pages
3. **RD12Lesson 1** – An example a dynamic web page. The parameters modified by the application are updated on the web page but only after a page refresh is requested.
4. **RD13 Lesson 2** – An example a dynamic web page with control. It is now possible to redirect to two other web pages: one that controls the on/off state of the LEDs on the Cerebot board and the other to set the price and name of the product. The status web page is updated with redirecting from the products page to the status page. The status web page must be updated to reflect any produce quantity count changes through the application program.
5. **RD14 Lesson 3** - This is essentially the same example as RD13 example except that the Diet Cola option is enhance with a mouse-over feature.
6. **RD15 UI WebVend** - This project is the basis for the assignment discussed in RD 16.
7. **RD16 HW4** - The solution for a class assignment to modify RD 15
8. **RD17 Yard Lites Basic** – is an example of web based control for a outdoor landscape lighting control system. This design was initially implemented on a Rabbit Semiconductor series 3000 in 2004 and has been running ever since at my home. This demonstration illustrates how to build your own (BYO) web control application. See RD18. There is no mechanism to automatically update the RTCC.

9. **RD18 Yard Lites NIST** - implements both server and client operations. The client operation is used to connect to a NIST time service to set the PIC32 real-time clock and calendar (RTCC). The NIST time service is contacted each morning at approximately 2:00 a.m. to correct the RTCC. The NIST time service also allows adjustments for daylight savings time. The sunrise and sunset times are established by reading a table that was specifically set for Moscow Idaho.

Starting with this project, the LCDblocking LCD driver based on bit-banging supplied with the MAL is no longer used. The LCDlib driver is my own LCD driver that uses the Peripheral Master Port (PMP). This results in higher speed LCD operation. However, this interface requires 11 processor IO pins. An alternate LCD interface is introduced in RD20 that uses only 7 processor IO pins.

Modifications to TCPIP ETH795 must include:

- i. `#define STACK_USE_UART`
- ii. `#define STACK_USE_SNTP_CLIENT`
- iii. `#define STACK_CLIENT_MODE`

10. **RD19 P2Solution** – A project solution for an enhance sprinkler / lighting control system.

All features described in RD18 are carried over to this reference design. This design implements password protection and uses the EEPORM (see i2clib files) to store operating parameters which are updated whenever there is a change sent from the web interface. Additional information is provided by the PDF file under other files.

11. **RD20a UDP\_MASTER** – An enhanced version of RD19 that serves as a master node that sends UDP packets to the slave light control node. The LCDblocking function that is provided with the MAL is replaces with my on LCD functions. Console communications is added using UART1. There are significant changes to the TCPIP ETH797.5 to support the additional features. Console is used

12. **RD20b UDP\_Slave** – This is the slave node that works with RD20a to receive the UDP packets and control a slave lighting system. This node does not serve a web page. It is an example of machine-machine internet control.

13. **RD20c UDP email** – This is a modified version of the master node that works with RD20b to send UDP packets to the slave node for controlling the lighting system. This node serves a web page identical to RD20a. Pressing BTN2 and BTN3 simultaneous causes this node to send a email as specified in the file SMTPclient. A valid email server and recipient must be specified as indicated in the SMTPclient.c code.

Application Level Modules selected must include `STACK_USE_SMTP_CLIENT` and `STACK_USE_HTTP2_SERVER`

## G. Application Notes and Tutorials

1. [HTML Tutorial – W3Schools](#)
2. [CSS Tutorial – W3Schools](#)
3. [TCP/IP Tutorial – W3Schools](#)
4. [JavaScript Tutorial – W3Schools](#)
5. The Microchip TCP/IP Stack [AN833](#)
6. Microchip TCP/IP Stack with BSD Socket API [AN1108](#)
7. [Microchip TCP/IP Stack Help .pdf](#) - installed with MAL
8. [Microchip TCP/IP Stack Overview](#)
9. [TCP/IP Networking Part 1: Web Based Status Monitoring](#) - [Video](#) (Lesson 1)
10. [TCP/IP Networking Part 2: Web Based Status Monitoring](#) - [Video](#) (Lesson 2)
11. [TCP/IP Networking Part 3: Web Based Status Monitoring](#) - [Video](#) (Lesson 3)
12. TCP/IP Stack Demo - <http://www.microchipweb.fr/>
13. Decoding HTTP see <http://en.wikipedia.org/wiki/Percent-encoding>