MPLAB Harmony Help

MPLAB Harmony Integrated Software Framework

Volume VII: Utilities

This volume describes the utilities that are available in MPLAB Harmony.

Description

MPLAB Harmony provides utilities to simplify the development process of middleware technologies, such as TCP/IP and Wi-Fi, as well as the Microchip Proprietary File System (MPFS2) and the Microchip Management Information Base (MIB) Compiler.

Microchip MIB Compiler

This section provides a description of the Microchip Management Information Base (MIB) Compiler.

Description

Microchip's SNMP Agent uses a custom script to describe the MIB. This script is designed to simplify the MIB definition and its integration with the main application. The actual MIB used by the SNMP Agent is a binary image created by the Microchip Management Information Base (MIB) to Binary Information Base (BIB) compiler named mib2bib.

Microchip MIB Script Commands

A Microchip MIB file is an ASCII text file consisting of multiple command lines. Each command line consists of a single command, starting with the dollar sign character ($), and one or more command parameters delimited with commas and enclosed in parentheses. Lines that do not start with a dollar sign are interpreted as comments and are not processed by the compiler. Commands must be written in a single line; they cannot span multiple lines.

The MIB script language includes a total of five commands, each having a specific syntax. Only one command, DeclareVar, is mandatory; the others are optional depending on the application and the types of information to be defined. In practice, at least one other command will be used in defining an MIB.

Notes:

1. Both the ASN.1 MIB file and the Microchip MIB script use the same `.mib` file extension; however, the files have distinctly different purposes. The ASN.1 MIB file is used by the MIB browser (NMS) to properly display context for your application.
2. The Microchip MIB script is compiled using mib2bib to create a BIB file. The BIB file is later converted using MPFS2 to store the MIB data for your application in internal Flash or EEPROM.

MIB Compiler

The mib2bib compiler converts the Microchip MIB script into a binary format compatible with the Microchip SNMP Agent. It accepts a Microchip MIB script in ASCII format and generates two output files: the binary information file, `snmp.bib`, and the C header file, `mib.h`. The binary file can be included in a Microchip File System (MPFS2) image.

The complete command line syntax for mib2bib is:

```
# java –jar mib2bib.jar [/?] [/h] [/q] <MIBFile> [/b=<OutputBIBDir>] [/I=<OutputIncDir]
```

where:

- `/?` - Displays command line help
- `/h` - Displays detail help for all script commands
- `/q` - Overwrites existing `snmp.bib` and `mib.h` files
- `<MIBFile>` - is the input MIB script file
- `<OutputBIBDir>` - is the output BIB directory where snmp.bib should be copied. If a directory is not specified, the current directory will be used
- `<OutputIncDir>` - is the output Inc directory where mib.h should be copied. If a directory is not specified, the current directory will be used

For example, the command, `Java –jar mib2bib.jar MySNMP.mib`, compiles the script, `MySNMP.mib`, and generates the `snmp.bib` and `mib.h` output files in the same directory.

Conversely, the command, `mib2bib /q MySNMP.mib /b=WebPages`, compiles the `MySNMP.mib` script file and overwrites the existing output files. It also specifies that the `snmp.mib` file is located in the subdirectory, `WebPages`. Because it is not specified, `mib.h` is assumed to be in the current directory.

If compilation is successful, mib2bib displays the statistics on the binary file, including the number of OIDs and the Agent ID, and the output file size.

The MIB compiler is a simple rule script compiler. While it can detect and report many types of parsing errors, it has these known limitations:

- All command lines must be written in single line
- All command parameters must immediately end with either a comma or a right parenthesis. For example, `$DeclareVar(myOID, ASCII_STRING …)`, will fail because the ASCII_STRING keyword is not immediately followed by a comma.
- All numerical data must be written in decimal

Example: Typical Output Display for a mib2bib Compilation

```
C:\microchip\harmony\v0_70b\utilities\mib2bib>java –jar mib2bib.jar snmp.mib
mib2bib v1.0.1 (Oct 14 2003)
Copyright (c) 2003 Microchip Technology Inc.
Input MIB File : C:\microchip\harmony\v0_70b\utilities\mib2bib\snmp.mib
Output BIB File: C:\microchip\harmony\v0_70b\utilities\mib2bib\snmp.bib
Output Inc File: C:\microchip\harmony\v0_70b\utilities\mib2bib\mib.h
BIB File Statistics:
          Total Static OIDs           : 9
          Total Static data bytes     : 57
          Total Dynamic OIDs          : 10
          (mib.h entries)
```

```
          Total Read-Only OIDs         : 3
          Total Read-Write OIDs        : 7
-----------------------------------------
Total OIDs                             : 19
Total Sequence OIDs                    : 4
Total AgentIDs                         : 1
=========================================
Total MIB bytes                        : 224
(snmp.bib size)
```

Note: For additional details about the Microchip MIB script, please refer to the Microchip application note *AN870* - "SNMP V2c Agent for Microchip TCP/IP Stack" (DS00870).

mib2bib.jar Run-time Error Codes

This topic lists the run-time error codes for `mib2bib.jar`.

Description

Run-time Error Codes

Error Code	Description	Reason
1000	Unexpected End-Of-File (EOF) found.	End-Of-File was reached before the end of command.
1001	Unexpected End-Of-Line (EOL) found.	End-Of-Line was reached before the end of command.
1002	Invalid escape sequence detected; only ‘,’, ‘\’, ‘(‘, or ’)’ may follow a ‘\’.	All occurrences of ‘,’, ‘(‘, ’)’, and ‘\’ must be preceded by a ‘\’.
1003	Unexpected empty command string received.	Command does not contain any parameter.
1004	Unexpected right parenthesis found.	Right parenthesis was found in place of a parameter.
1005	Invalid or empty command received.	Command does not contain sufficient parameters.
1006	Unexpected escape character received.	A ‘\’ character was detected before or after parameters were expected.
1007	Unknown command received.	N/A
1008	Invalid parameters: expected `$DeclareVar(oidName, dataType, oidType, accessType, oidType)`.	N/A
1009	Duplicate OID name found.	Specified OID name is already in use.
1010	Unknown data type received.	Data type keyword does not match one of the allowed keywords.
1011	Unknown OID type received.	OID type keyword does not match one of the allowed keywords.
1012	Empty OID string received.	N/A
1013	Invalid parameters: expected `$DynamicVar(oidName, id)`.	N/A
1014	OID name is not defined.	N/A
1015	Invalid OID ID received – must be between 0-1023 inclusive.	N/A
1016	Invalid parameters: expected `$SequenceVar(oidName,index)`.	N/A
1017	Invalid parameters: expected `$SequenceVar(oidName,index)`.	N/A
1018	Current OID already contains a static value.	This OID has already been declared static.
1019	Invalid number of index parameters received.	All SequenceVar must include only one index.
1020	OID of sequence type cannot contain static data.	All sequence OID variables must be dynamic.
1021	This is a duplicate OID or the root of this OID is not the same as previous OID(s), or this OID is a child of a previously defined OID.	All OID strings must contain the same root OID.
1022	Invalid index received; must be BYTE data value.	All sequence index OID must be of data type, BYTE.
1023	Invalid OID access type received; must be READONLY or READWRITE.	N/A
1024	Current OID is already assigned an ID value.	Current OID is already declared as dynamic.
1025	Duplicate dynamic ID found.	Current OID is already declared as dynamic with duplicate ID.
1026	No static value found for this OID.	Current OID was declared static, but does not contain any data.

1027	No index value found for this OID.	Current OID was declared as sequence, but does not contain any index.
1028	OID data scope (dynamic/static) is not defined.	Current OID was declared, but was not defined to be static or dynamic.
1029	Invalid data value found.	Data value for current OID does not match with its data type.
1030	Invalid parameters: expected `$AgentID(oidName, id)`.	N/A
1031	Only OID data type is allowed for this command.	AgentID command must use OID name of OID data type.
1032	This OID must contain static OID data.	AgentID command must use OID name of static data.
1033	This OID is already declared as an Agent ID.	Only one AgentID command is allowed.
1034	An Agent ID is already assigned.	Only one AgentID command is allowed.
1035	OID with READWRITE access cannot be static.	An OID was declared READWRITE and made static.
1036	OID of OID data type cannot be dynamic.	Current version does not support OID variable of data type, OID.
1037	This OID is already declared as dynamic.	N/A
1038	This OID is already declared as static.	N/A
1039	This OID does not contain the Internet root. The Internet root of ‘43’ must be used if this is an Internet MIB.	All internet OIDs must start with ‘43’. This is a warning only and will not stop script generation.
1040	The given value was truncated to fit in a specified data type.	An OID was declared as BYTE or WORD but the value given in StaticVar exceeded the data range.
1041	The given string exceeds a maximum length of 127.	All OCTET_STRING and ASCII_STRING must be less than 128.
1042	Invalid OID name detected; OID name must follow standard ‘C’ variable naming convention.	All OID names must follow ‘C’ naming convention as these names are used to create ‘define' statements in the `mib.h` file.
1043	Total number of dynamic OIDs exceeds 1023.	This version supports total dynamic OIDs of 1024 only. All dynamic OID IDs must range from 0-1023.

MPFS2 Utility

This section provides a description of the Microchip Proprietary File System (MPFS2) utility.

Introduction

This topic provides an overview of the MPFS2 Utility in MPLAB Harmony.

Description

The MPFS2 Utility packages web pages into a format for efficient storage in an embedded system. It is a graphical application for personal computers that can generate MPFS2 images for storage in external storage or internal Flash program memory.

When used to build MPFS2 images, the MPFS2 Utility also indexes the dynamic variables found. It uses this information to generate `http_print.h`, which ensures that the proper callback functions are invoked as necessary. It also stores this index information along with the file in the MPFS2 image, which alleviates the task of searching from the embedded device.

Finally, when developing an application that uses external storage, the MPFS2 Utility can upload images to the external storage device using the upload functionality built into the HTTP web server or FTP server. The `http_print.idx` file is also generated by the utility, which keeps information of all the dynamic variable details.

The MPFS2 utility generates the `MPFS2SettingDetails.xml` file, which contains all of the run-time configured parameters.

Note: If there is any change in `http_print.idx` , the updated MPFS image will be generated. It is recommended to remove the `http_print.idx` file before generating a new MPFS image.

Building MPFS2 Images

This topic provides information for building MPFS2 images.

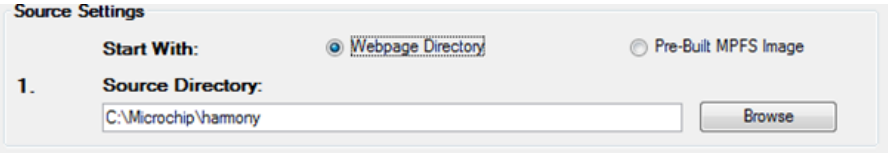
Description

The MPFS2 Utility has four steps, which are denoted on the left hand side of the dialog.

To build an MPFS image, select **Start With: Webpage Directory** in Step 1 and choose the directory in which the web pages are stored.

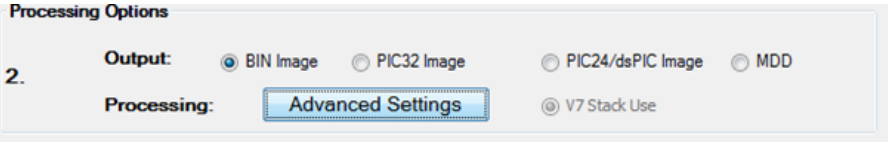
By default, the source directory path is `C:\Microchip\harmony`. Users need to provide the web page directory path from the `apps\tcpip\` demonstration project.

The Source Directory path is stored in `MPFS2SettingDetails.xml` for future access.



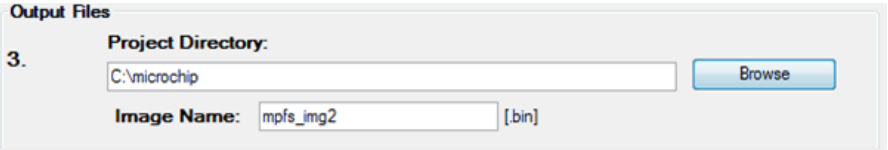
Step 2 selects the output format. If storing the web pages in external EEPROM or serial Flash, choose the **BIN Image** output format. If internal program memory will be used, select **PIC32 Image** for use with 32-bit devices, or **ASM30 Array** for 16-bit targets. To store the web pages on a device formatted with the FAT file system without compressing them into an MPFS image, select **MDD**. Refer to Advanced MPFS2 Settings for more information.

The configurable parameters of Output file format and Advance Settings are stored in `MPFS2SettingDetails.xml`.



Step 3 asks for the TCP/IP MPLAB X IDE project directory. The MPFS utility will write the image file to the project directory, and will also update the `http_print.h` file there if needed. Select the correct directory so that the right files are modified.

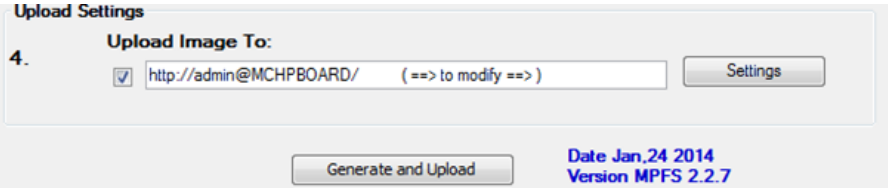
By default, the project directory path is one path behind the source directory. Both the Project Directory and Image Name are configurable parameters, which are stored in `MPFS2SettingDetails.xml`.



Step 4 controls the upload settings. When external EEPROM or serial Flash is used for storage, the option to upload the newly created image to the board is available. Check the box next to **Upload Image To** to enable this feature. The target host name (or IP address), upload protocol, and

upload path may need to be changed to the one chosen when the board was first configured. You may also need to modify the user name and password used to access the secured functionality in your application, like web page upload. Use the **Settings** button to edit these values (see MPFS2 Upload Settings for more information).

If internal program memory is being used, the image will be compiled in with the project and so direct uploads are not available. Make sure to include the output source file indicated in Step 3 as part of the project.



Once all the correct settings have been chosen, click the **Generate** button to create the image. If uploads are enabled, this will also attempt to upload the file to the device.

Uploading Prebuilt MPFS2 Images

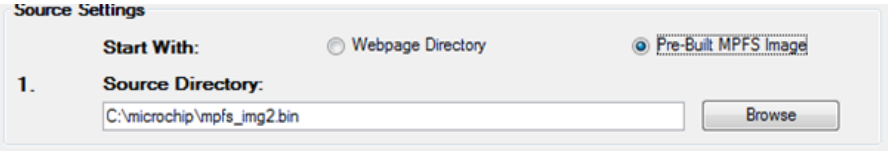
This topic provides information on uploading prebuilt MPFS2 images.

Description

There are two ways to upload a prebuilt image to external storage. The first involves uploading from the browser directly. The second is to use the MPFS2 Utility to upload the image. You can select HTTP or FTP uploading to match the protocol that your application uses.

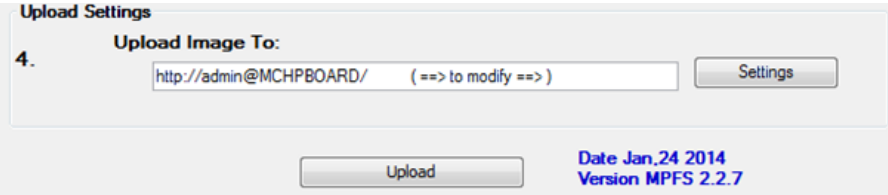
To use the MPFS2 Utility to upload an image, begin by selecting Start With: **Pre-Built MPFS Image** in Step 1. The Source: Directory path and Start With: radio button are configurable parameters, which are stored in `MPFS2SettingDetails.xml`.

Choose the image file to upload.



Steps 2 and 3 are not required for prebuilt images. Proceed directly to Step 4 and verify that the upload settings are correct. The target host name (or IP address), upload protocol, and upload path may need to be changed to the one chosen when the board was first configured. You may also need to modify the user name and password used to access the secured functionality in your application, like web page upload. Use the Settings button to edit these values (see MPFS2 Upload Settings for more information).

Once all the settings are correct, click **Upload**. The image will be uploaded to the board.

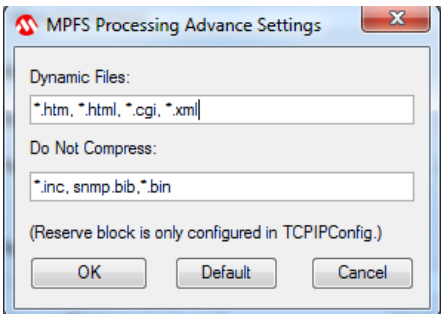


Advanced MPFS2 Settings

This topic provides information on advanced MPFS2 settings.

Description

The Advanced Settings dialog found in Step 2 provides greater control over how files are processed.



The **Dynamic Files** list indicates which file types to parse for dynamic variables. By default, all files with the extensions `.htm`, `.html`, `.cgi`, or `.xml` are parsed. If an application has dynamic variables in other file types, these types must be added to the list. This field must be a comma-separated list of extensions and file names.

Default Dynamic Files are `*.htm`, `*.html`, `*.cgi`, and `*.xml`.

The **Do Not Compress** field indicates which file types should never be compressed. Compressing files with GZIP saves both storage space and transmission time. However, this is only suitable for static content such as CSS or JavaScript. Any files with dynamic variables will automatically be excluded. In addition, any file that the PIC may need to process internally should be excluded. Files included via `~inc:filename~` should not be compressed, nor should any BIB file used for the SNMP module (if present). Additional file types can be added to this list if a custom application will be accessing the MPFS.

Default Do Not Compress Files are `*.inc`, `snmp.bib`, and `*.bin`.

The GZIP compressor will attempt to shrink all files. In some cases, especially with images, little or no compression is achieved. When this occurs the file is stored "as is" in the MPFS image.

MPFS2 Command Line Options

This topic provides a description of the MPFS2 command line options.

Description

To facilitate batch files and automation, the MPFS2 Utility also supports execution from the command line. The syntax is as follows:

```
MPFS2.jar [options] <SourceDir> <ProjectDir> <OutputFile>
```

The SourceDir, ProjectDir, and OutputFile options are required and should be enclosed in quotation marks. The OutputFile option will be relative to ProjectDir, and cannot be a full path name. The various option switches are described in the following table.

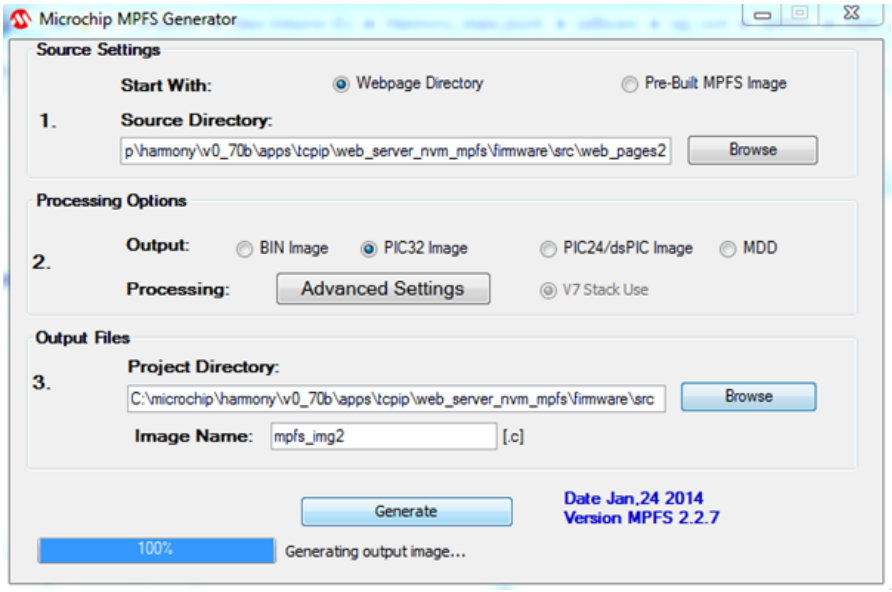
Switch	Short	Description
/BIN	/b	Output a BIN image (Default)
/PIC32	/c	Output a XC32 image
/ASM16	/s	Output an ASM16 image
/mpfs2	/2	Use the MPFS2 format (Default)
/html "..."	/h "..."	File types to be parsed for dynamic variables (Default: "`*.htm`, `*.html`, `*.cgi`, `*.xml`")
/xgzip "..."	/z "..."	File types to be excluded from GZIP compression (Default: "`*.bib`, `*.inc`, `*.bin`")

SourceDir, ProjectDir, and OutputFile are required and should be enclosed in quotes.

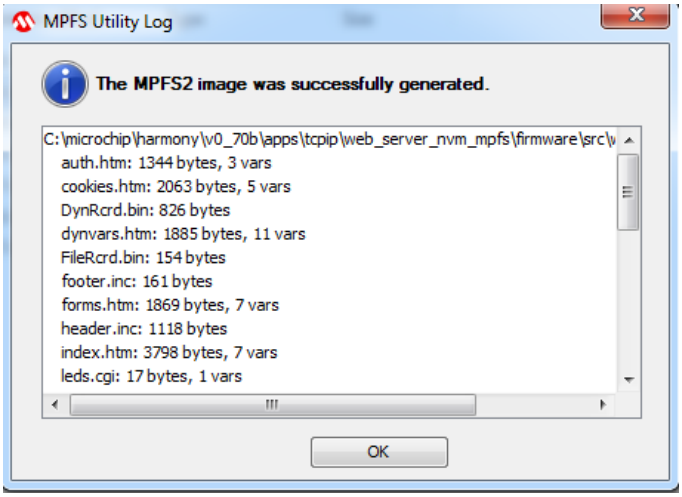
OutputFile is placed relative to ProjectDir and cannot be a full path name.

For unified stack (/v) should be used with PIC32 output image format.

PIC32 Output Option Example:



Successful MPFS2 Image Generation Example:



MPFS2 Utility Output Log Details

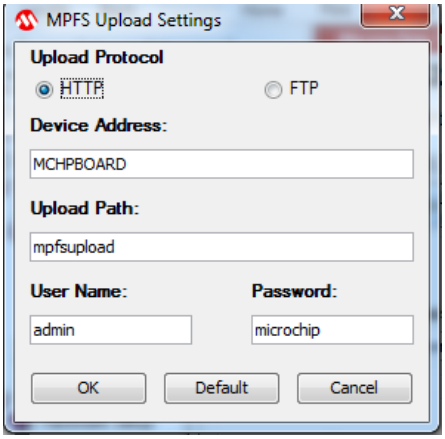
```
C:\microchip\harmony\v0_70b\apps\tcpip\web_server_nvm_mpfs\firmware\src\web_pages2 :
auth.htm: 1344 bytes, 3 vars
cookies.htm: 2063 bytes, 5 vars
DynRcrd.bin: 826 bytes
dynvars.htm: 1885 bytes, 11 vars
FileRcrd.bin: 154 bytes
footer.inc: 161 bytes
forms.htm: 1869 bytes, 7 vars
header.inc: 1118 bytes
index.htm: 3798 bytes, 7 vars
leds.cgi: 17 bytes, 1 vars
mchp.css: 915 bytes (gzipped by 72%)
mchp.gif: 1250 bytes (gzipped by 2%)
mchp.js: 1474 bytes (gzipped by 61%)
snmp.bib: 493 bytes
status.xml: 183 bytes, 7 vars
temp.cpf: 236 bytes (gzipped by 31%)
upload.htm: 943 bytes, 4 vars
C:\microchip\harmony\v0_70b\apps\tcpip\web_server_nvm_mpfs\firmware\src\web_pages2\snmp :
snmp/snmpconfig.htm: 1341 bytes, 9 vars
C:\microchip\harmony\v0_70b\apps\tcpip\web_server_nvm_mpfs\firmware\src\web_pages2\protect :
protect/config.htm: 1888 bytes, 12 vars
protect/index.htm: 963 bytes, 3 vars
protect/reboot.cgi: 8 bytes, 1 vars
protect/reboot.htm: 1779 bytes, 4 vars
C:\microchip\harmony\v0_70b\apps\tcpip\web_server_nvm_mpfs\firmware\src\web_pages2\email :
email/index.htm: 1876 bytes, 6 vars
C:\microchip\harmony\v0_70b\apps\tcpip\web_server_nvm_mpfs\firmware\src\web_pages2\dyndns :
dyndns/index.htm: 1461 bytes, 12 vars
GENERATED MPFS2 IMAGE: 28932 bytes
```

MPFS2 Upload Settings

This topic provides a description of the MPFS2 upload settings.

Description

The Upload Settings dialog found in Step 4 of Building MPFS2 Images provides configuration to upload the binary file to the external EEPROM or serial Flash using either the HTTP or FTP protocol.



The Device Address is either a board name or an IP address. Please not that only IPv4 is supported.

By default, the Upload Path is "mpfsupload", the MPFS upload User Name is "admin", and the Password is "Microchip".

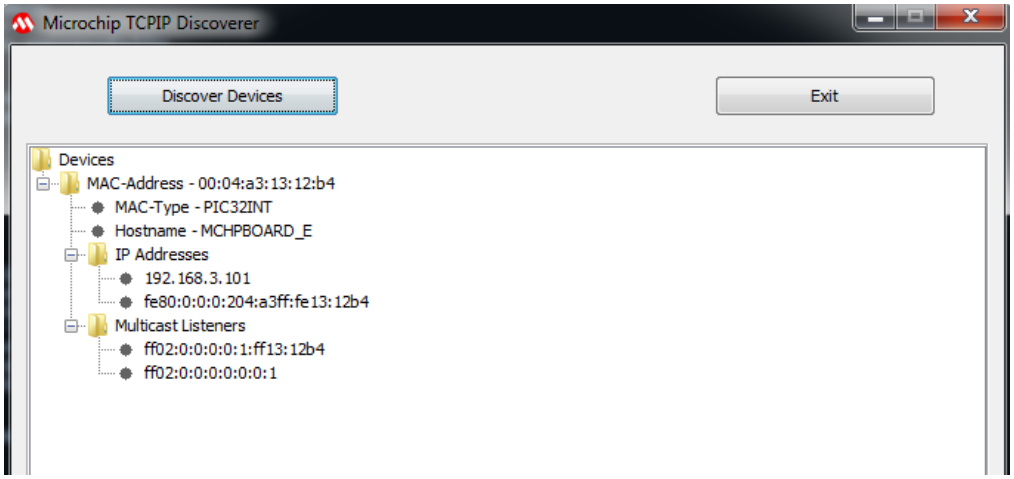
TCP/IP Discoverer Utility

This section provides an overview of the TCP/IP Discoverer utility.

Description

The TCP/IP Discoverer personal computer project (formerly known as the Embedded Ethernet Device Discoverer) will aid in embedded product device discovery (with the Announce protocol) and will demonstrate how to write personal computer applications to communicate to embedded devices.

Clicking **Discover Devices** (see the following figure), causes the application to transmit a broadcast UDP packet containing the message, "Discovery: Who is out there?" on the local network to port 30303. If any embedded devices with the Announce protocol enabled are connected to the network, they will respond with a UDP packet containing their MAC-Address, MAC-Type (interface type), Hostname (NBNS), IP Addresses (IPv4/IPv6), and Multicast Listeners (IPv6).



Each part of this UDP packet is displayed as a set of hierarchical data (JTree Class). "Devices" is represented as a root node and each MAC-Address of the UDP packet is represented as the child node and other payload information of the UDP data are child nodes to the MAC-Address node. The "Truncated" string is added as a child node to the MAC-Address node, if the Announce UDP packet payload does not have sufficient space to contain the interface details.

If TCPIP_STACK_USE_HTTP2_SERVER is enabled, double clicking MAC-Address node, http://ipv4address client, will communicate with the Web Server module of the device.

The Java source code for this application is also included. This source code should provide a rough idea of how to write a personal computer-based application to communicate with your embedded devices.

WiFi Utilities

This section provides information on the firmware update utility.

WINC1500 Firmware Update Guide

This topic describes updating the WINC1500 Firmware.

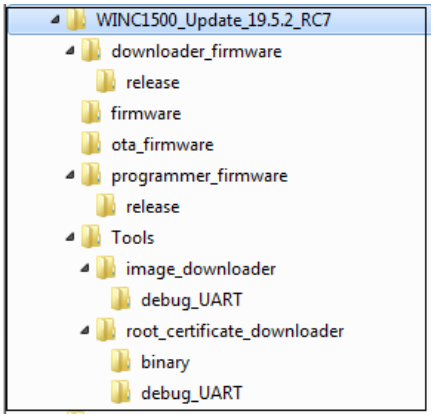
Introduction

This document applies to MPLAB Harmony version 2.04 and later, and provides firmware upgrade instructions for the WINC1500 Wi-Fi module.

Description

The WINC1500 Firmware Update utility installation file is available in your installation of the Windows version of MPLAB Harmony v2.04 and later in the following folder: `<install-dir>\utilities\wi-fi\winc1500_fw_19.5.2_rc7_setup.exe`.

Run the `winc1500_fw_update_19_5_2_RC7.exe` executable and follow the installation prompts. Once the installation has completed the following folders and files are available:



Notes:

1. The firmware binary file, `m2m_aio_3a0.bin`, for *over the serial port* upgrade is available in the folder: `.\WINC1500_Update_19.5.2_RC7\firmware`. The batch file, `download_all_mla_harmony.bat`, to perform the upgrade is found in: `.\WINC1500_Update_19.5.2_RC7`.
2. The firmware binary, `m2m_ota_3a0.bin`, for *Over-The-Air (OTA)* upgrade is available in the folder: `.\WINC1500_Update_19.5.2_RC7\ \ota_firmware`. This binary should be uploaded to the OTA HTTP or HTTPS Web server prior to performing the OTA firmware upgrade on the device.

Prerequisites

Provides information on the hardware and software prerequisites.

Description

The following general development hardware is required:

- Windows 7 computer
- MPLAB ICD 3 In-circuit Debugger
- WINC1500 PICtail/PICtail Plus Daughter Board module or WINC1500 WiFi7click module with one of the following BSPs selected:
  - pic32mx795_pim+e16
  - pic32mz_ef_sk+Starter Kit I/O Expansion Board + pic32mz Starter Kit Adapter Board
  - pic32mx_eth_sk+Starter Kit I/O Expansion Board
  - pic32mx795_pim+e16_32
- USB/Serial cable
- AC power adapter

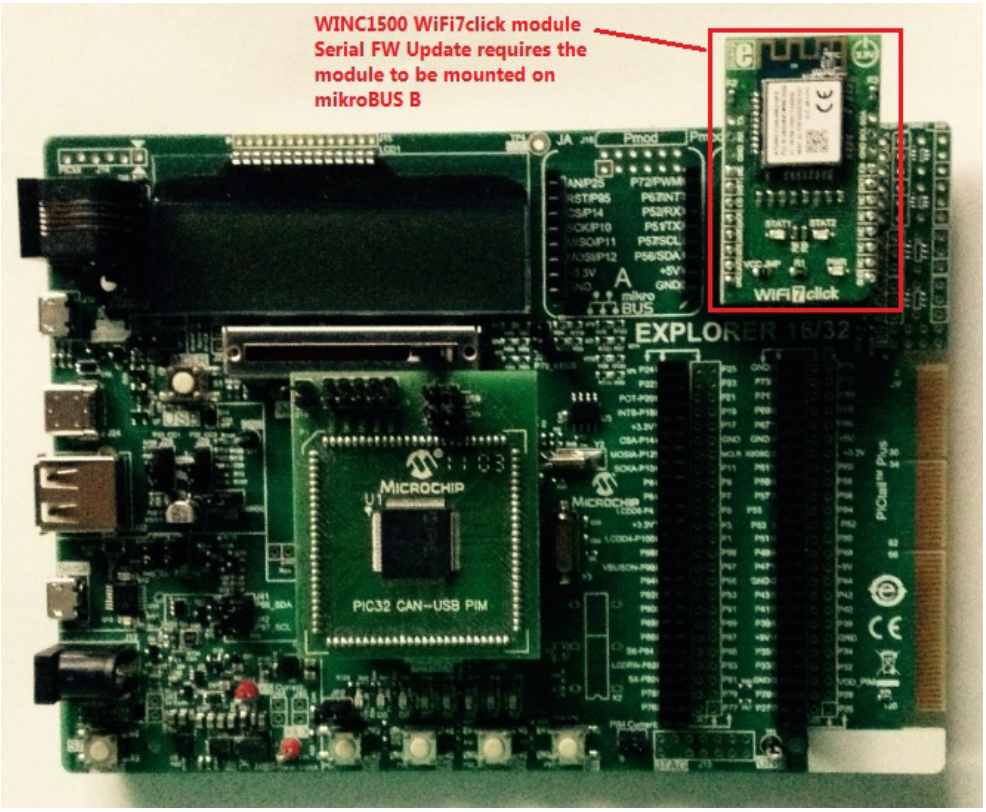
The following figures shows the hardware configuration using a Windows computer.

WINC1500 PICtail module FW Serial Update Using the pic32mx795_pim+e16

WINC1500 PICtail Module FW Serial Update Using the pic32mz_ef_sk + Starter Kit I/O Expansion Board + pic32mz Starter Kit Adapter Board

WINC1500 PICtail Module FW Serial Update Using the pic32mx_eth_sk + Starter Kit I/O Expansion Board

WINC1500 WiFi 7 Click Module FW Serial Update Using the pic32mx795_pim+e16_32



Hardware Prerequisites

Provides information on the hardware prerequisites.

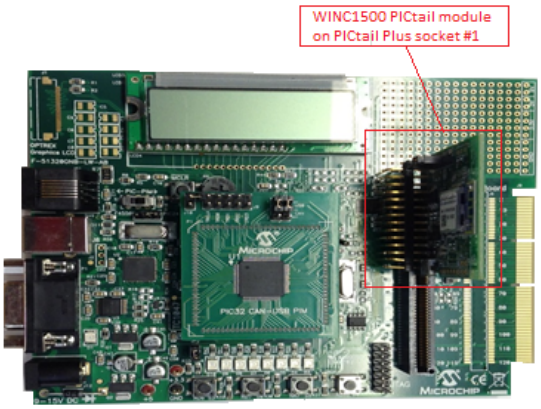
Description

The WINC1500 Firmware Upgrade software projects provide you with a choice of two different hardware platforms. The following table lists and describes the three options.

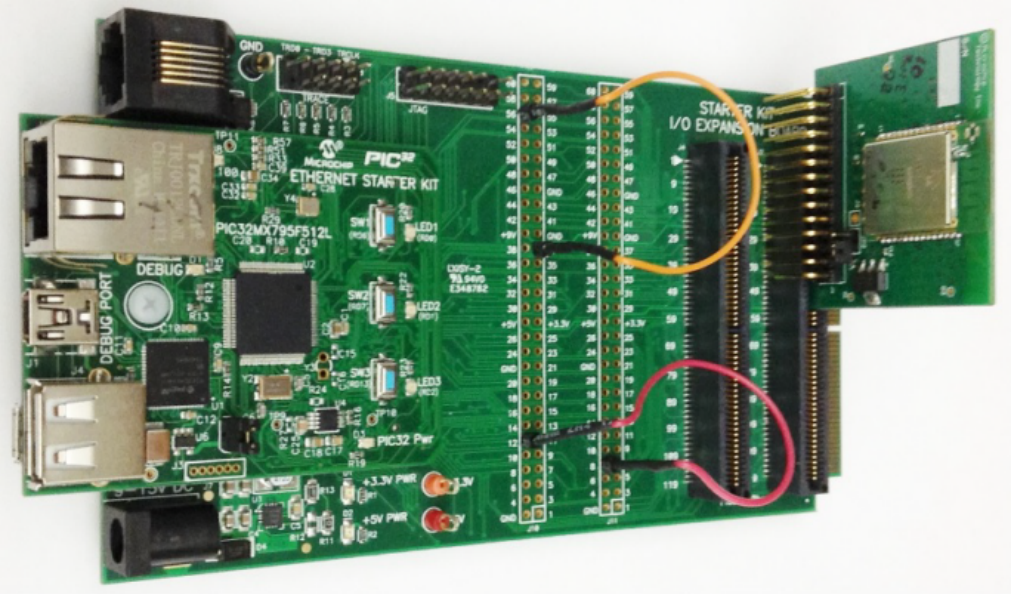
Hardware Platform Options

Development Platforms	MCU	WINC1500	Comments
Option 1: • • Microchip Explorer 16 Development Board (DM240001) • • MPLAB ICD3 In-Circuit Debugger (DV164035) • • 9V Power supply (AC002014) • • DB9 Serial cable or USB/Serial cable	PIC32MX USB/CAN Plug-in Module (MA320003)	WINC1500 PICtail Plus module	Hardware Platform support in MPLAB Harmony v2.04 and later. This setup is required for WINC1500 Firmware Upgrade over Serial Port.
Option 2: • PIC32 Ethernet Starter Kit (DM320004) • Starter Kit I/O Expansion Board (DM320002) • MPLAB ICD3 In-Circuit Debugger (DV164035) • Micro USB to USB (for serial and power)		WINC1500 PICtail Plus module	Hardware Platform support in MPLAB Harmony v2.04 and later
Option 3: • PIC32MZ Embedded Connectivity with Floating Point Unit (EF) (DM320007) • Starter Kit I/O Expansion Board (DM320002) • PIC32MZ Starter Kit Adaptor Board (AC320006) • MPLAB ICD3 In-Circuit Debugger (DV164035) • Micro USB to USB (for serial and power)		WINC1500 PICtail Plus module	Hardware Platform support in MPLAB Harmony v2.04 and later
Option 4: • Microchip Explorer 16/32 Development Board (DM240001-2) • MPLAB ICD3 In-Circuit Debugger (DV164035) • Micro USB to USB (for serial and power)	PIC32MX795F512L USB-CAN PIM (MA320003)	WiFi 7 Click Module	Hardware Platform support in MPLAB Harmony v2.0.4 and later. This setup is required for the WINC1500 WiFi 7 Click Module Firmware Upgrade over Serial Port.

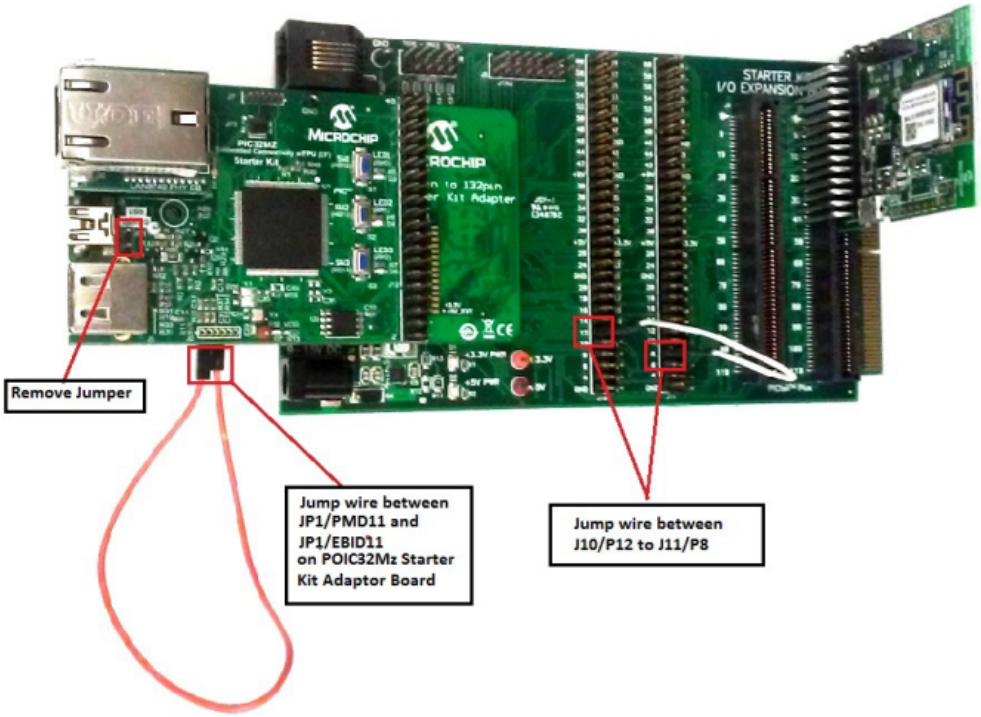
Hardware Platform Option 1 With the WINC1500 PICtail Module



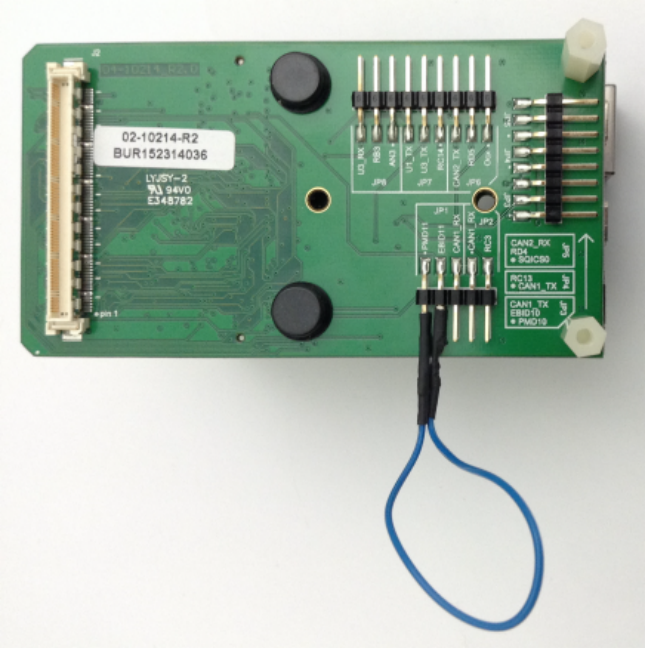
Hardware Platform Option 2 With the WINC1500 PICtail Module



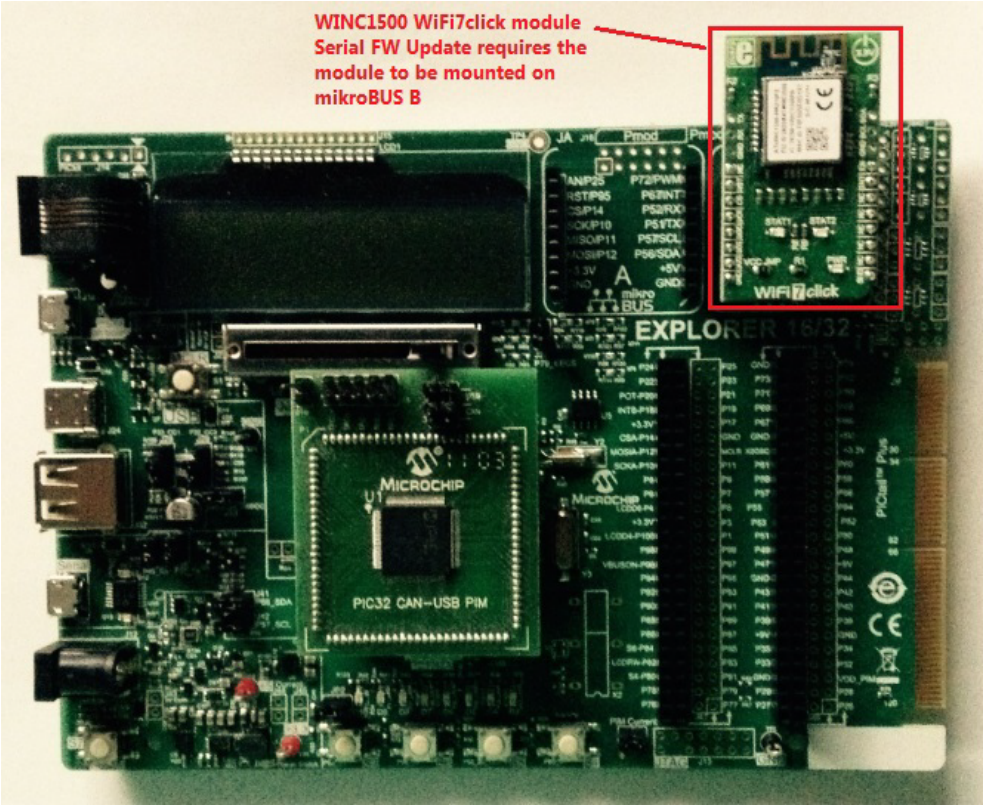
Hardware Platform Option 3 With the WINC1500 PICtail Module



PIC32MZ Starter Kit Adaptor Board - Jumper between JP1/PMD11 and JP1/EBID11



Hardware Platform Option 4 With the WINC1500 WiFi 7 Click Module



Software Prerequisities for Windows

Provides information on the software prerequisites.

Description

- MPLAB X IDE v4.00 or later
- MPLAB XC32 v1.43 or later
- WINC1500 Wi-Fi Driver Library in MPLAB Harmony v2.04, or later
- WINC1500_Update_19.5.2_RC7 Utility

- The utility installation file, `winc1500_fw_19.5.2_rc7_setup.exe`, is located in the `<install-dir>/utilities/wi-fi` folder of your MPLAB Harmony installation (v2.04 or later)

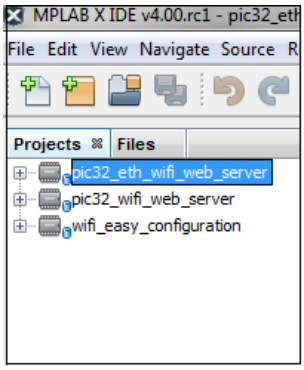
Firmware and Root Certificate Upgrade Over Serial Port

This section discusses how to upgrade the firmware revision and the root certificate files of the WINC1500 PICtail module.

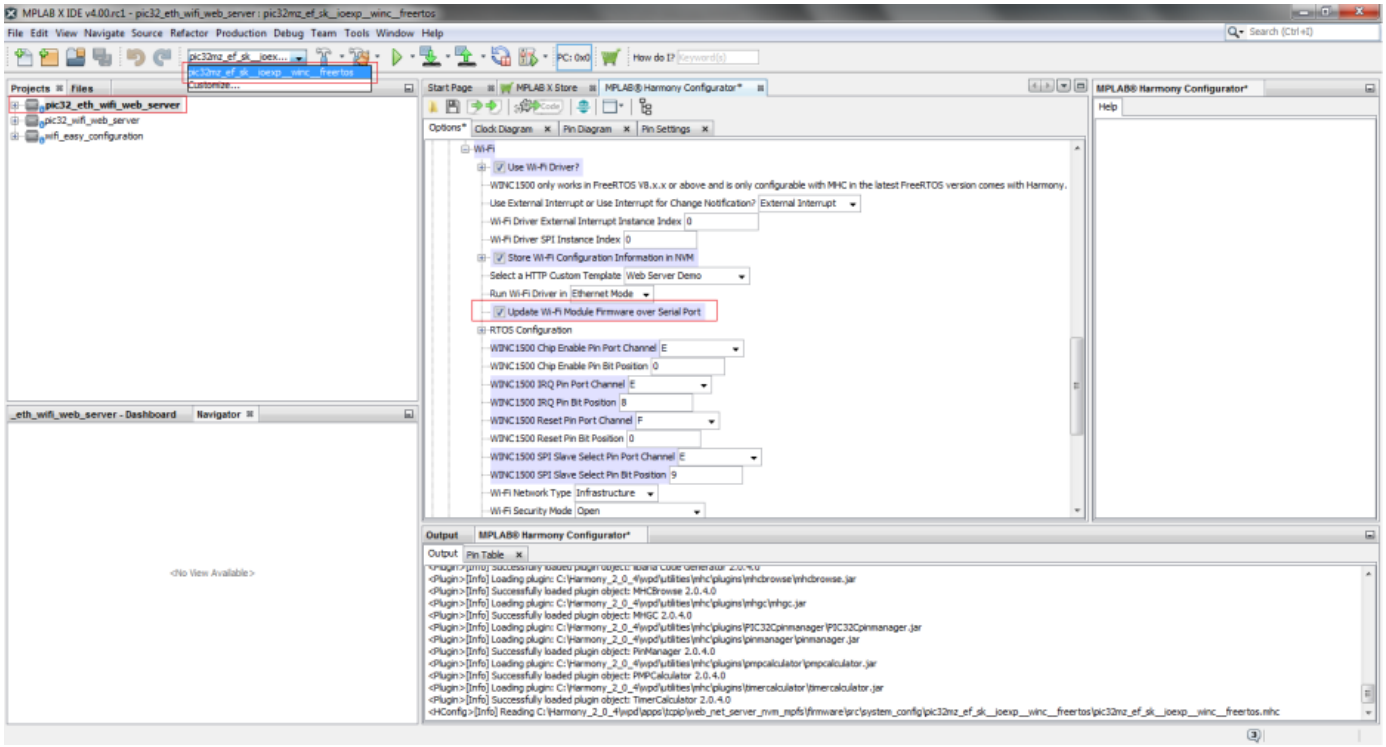
Description

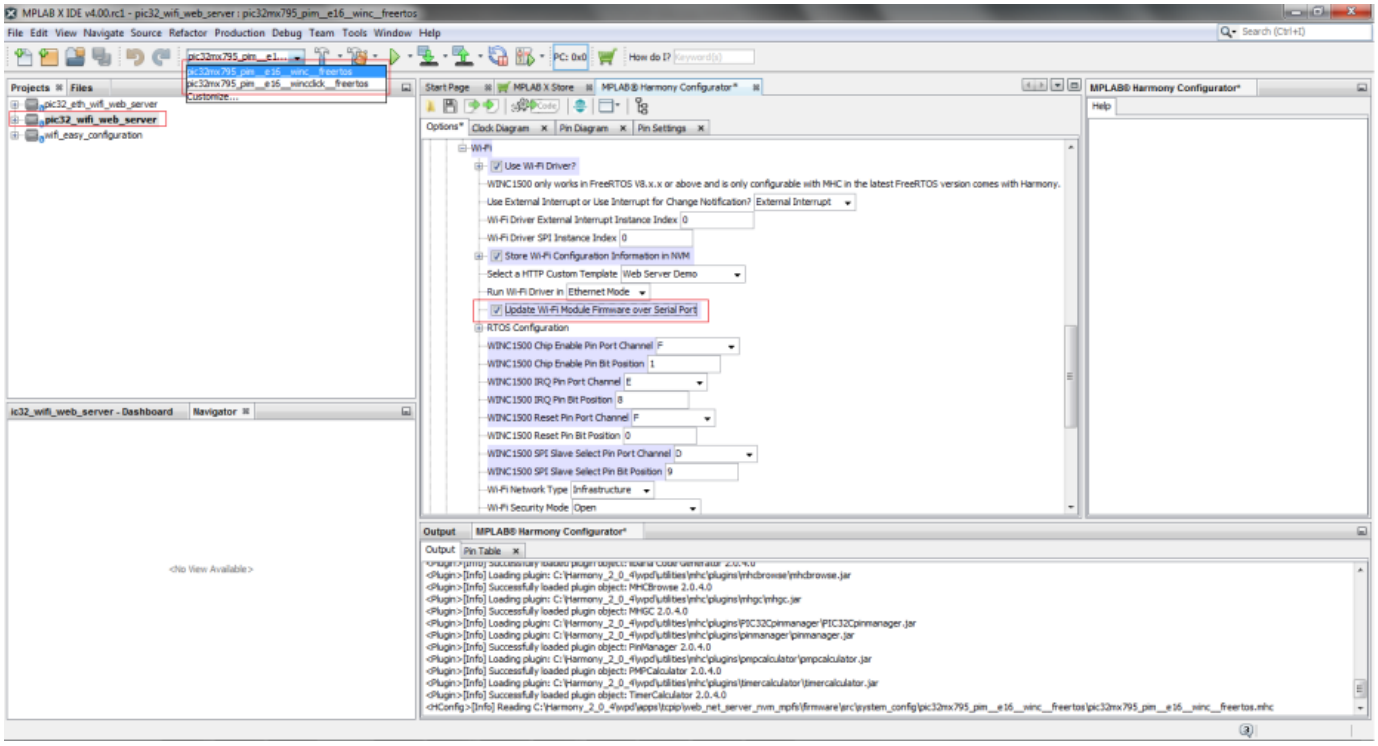
The upgrade process requires the following setup:

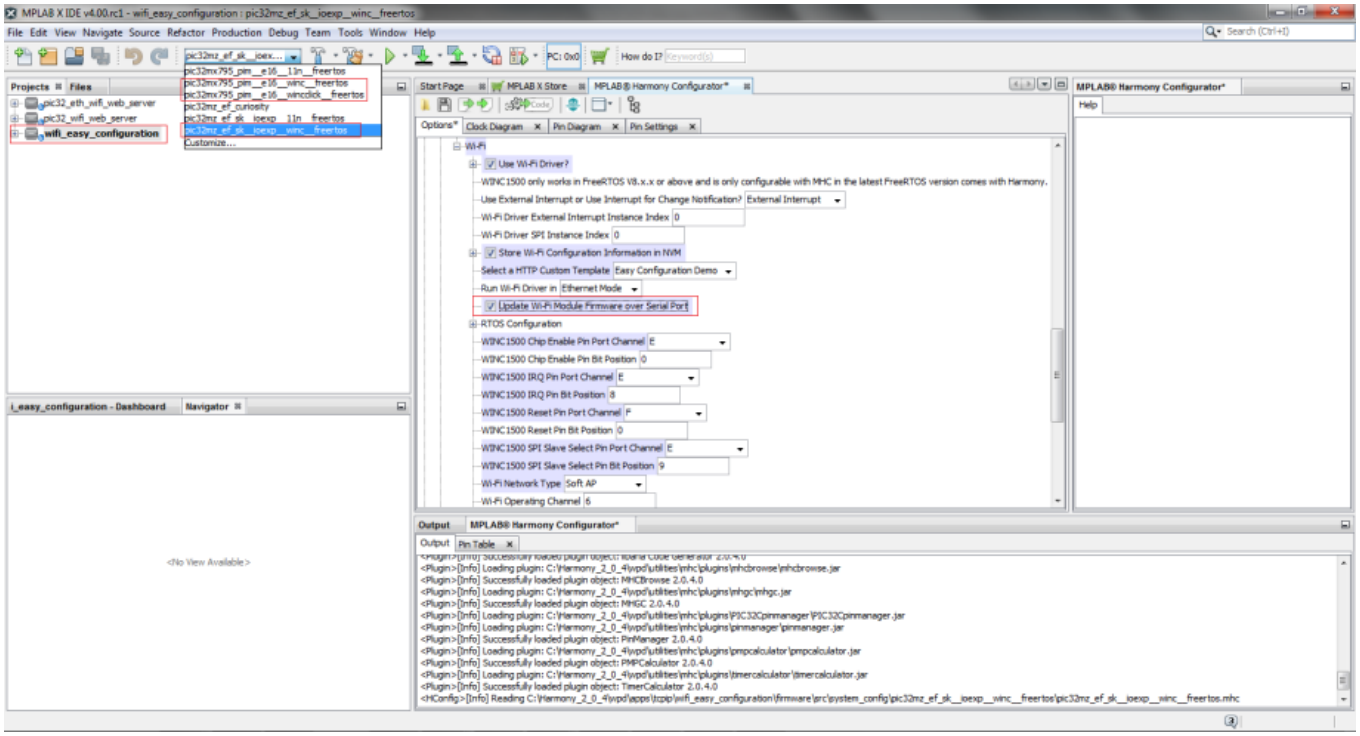
1. Hardware Platform with WINC1500 PICtail module (see Hardware Prerequisites).
2. Install the Software Prerequisites for Windows on a Windows computer (see Software Prerequisites for Windows).
3. On the Windows computer run MPLAB X IDE and open and of the MPLAB Harmony Wi-Fi projects shown in the following four figures to work with the hardware development board in use for firmware serial update, and in the MHC "Enable the Serial Update" option in the Wi-Fi driver.



4. Set the active project and select the project configuration, and in MHC within Wi-Fi Driver, select the "Enable the Serial Update" option.







4. Build and load the “project demonstration onto the MPLAB Harmony device.
5. To continue, there must be a UART connection between the board and a Windows computer. If you are using the Explorer 16 Development Board, this connection will be made using a DB9 cable. Do not open a terminal program on the PC, as the PC firmware update utility communicates directly with the board. The update utility automatically scans for the UART port connected to the board.

6. After completing all of the previous steps, run the batch file, `download_all_mla_harmony.bat`, on the PC. The batch file can be launched by double-clicking on the file within Windows Explorer, or by running the batch file from within a Command window. The output should be similar to what is shown in the Appendix.

OTA Firmware Upgrade

Provides information on updating the firmware over-the-air.

Upgrade Using HTTP Web Server

Provides information on updating the firmware OTA using a HTTP Web server.

Description

This example demonstrates how to update the firmware of the WINC1500 PICtail via OTA. The update downloads the WINC1500 OTA firmware binary file, `m2m_ota_3a0.bin`, from OTA Download Web server to your MPLAB Harmony device. You can then upload a new firmware image to the OTA Download Web server.

Note: The out-of-box the example demonstration supports OTA firmware binary upgrade with its console command, `ota <address of http server>`. No additional system configuration option is needed.

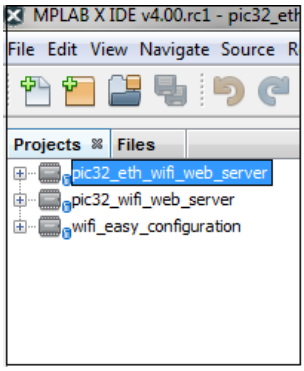
Examples:

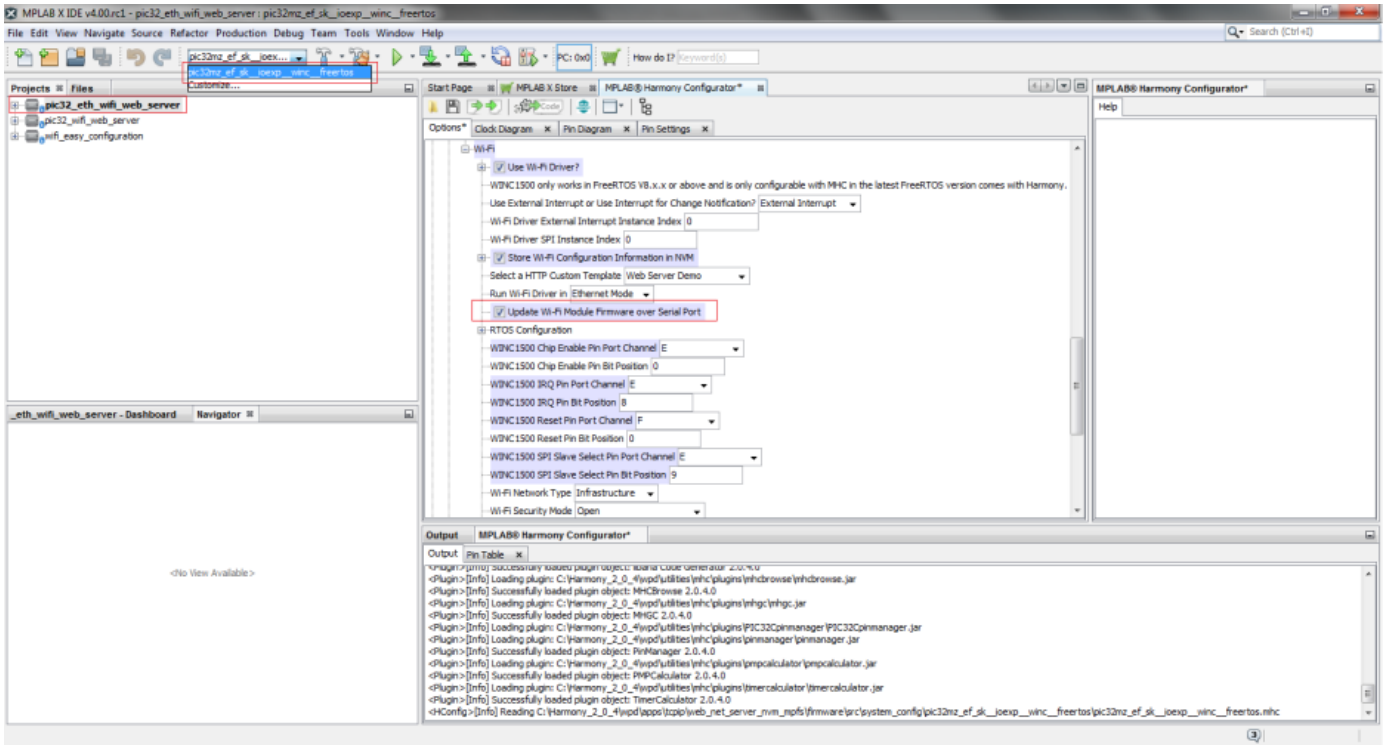
ota http://192.168.1.107/m2m_ota_3a0.bin

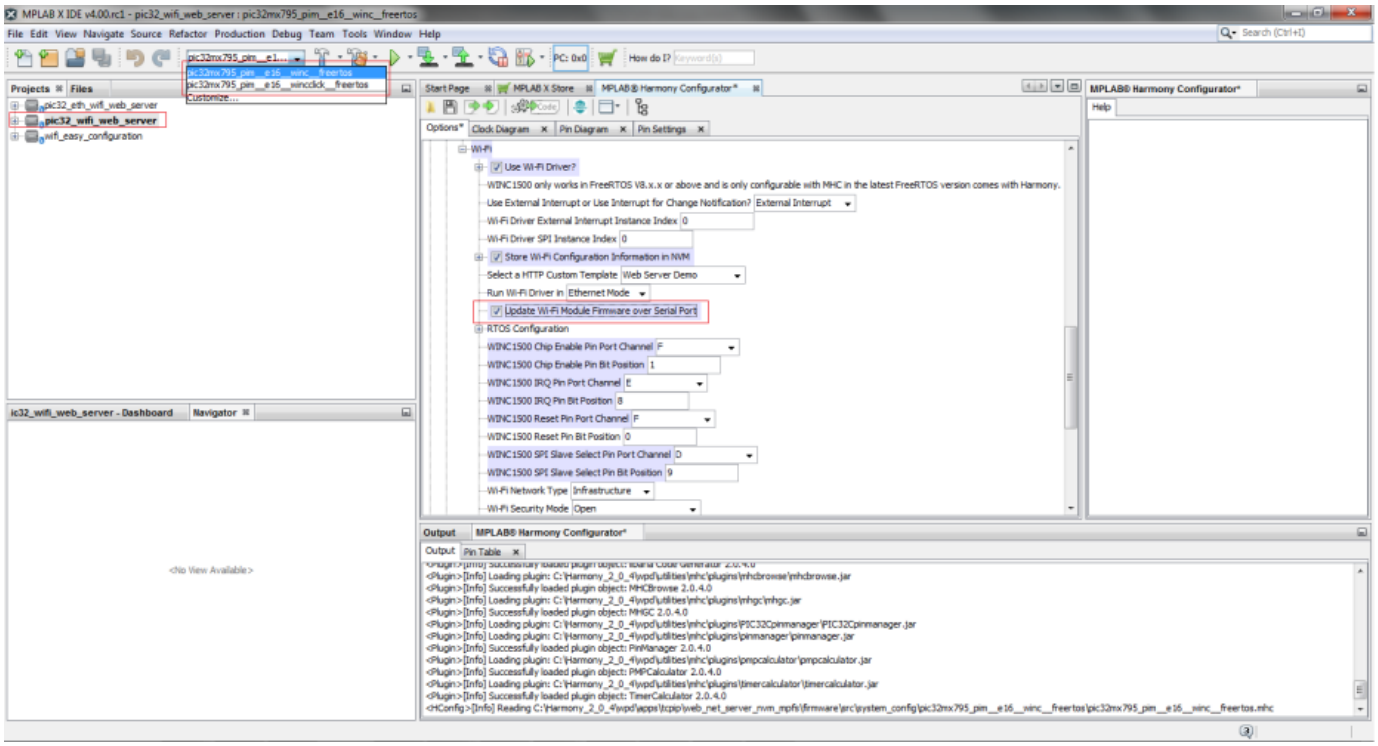
or

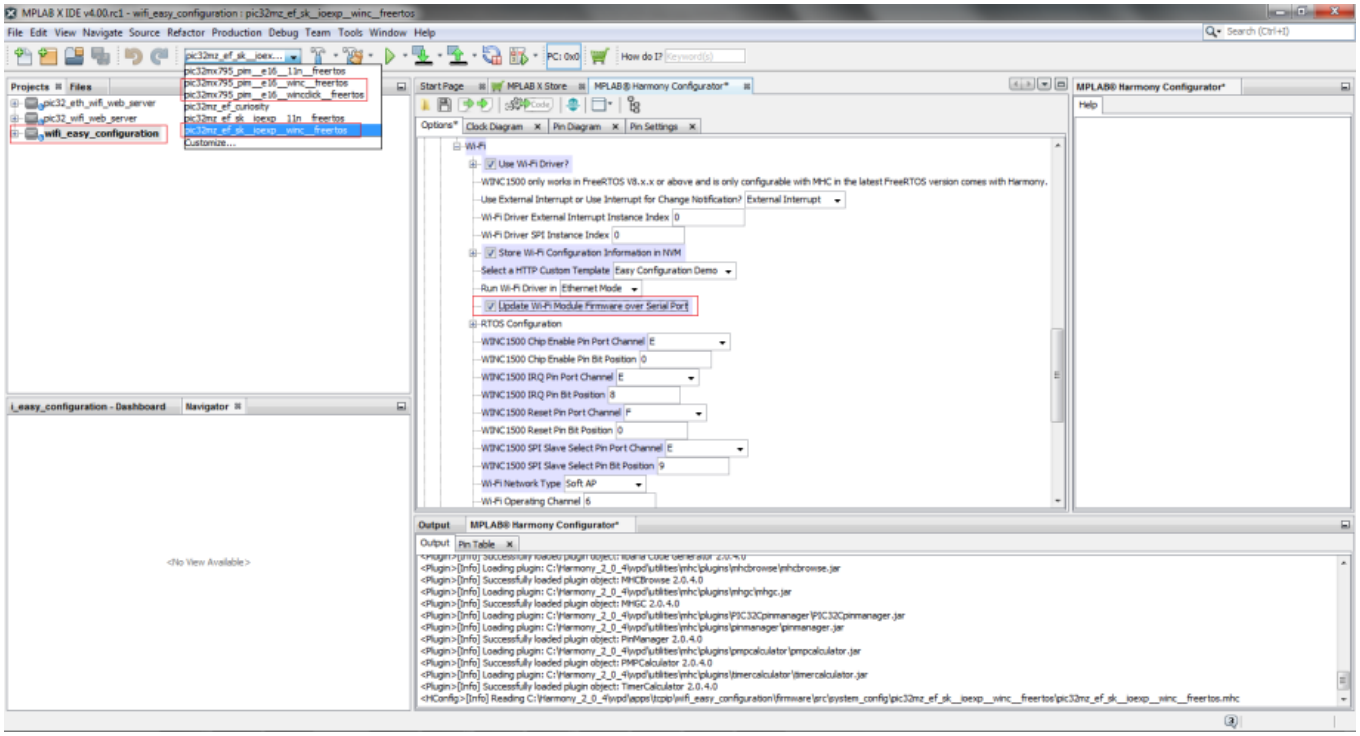
ota https://192.168.1.107/m2m_ota_3a0.bin

1. Start MPLAB X IDE and open the MPLAB Harmony project, by selecting one of the projects from the following figures, and then build and run to install the demonstration to the target hardware development board. Once the demonstration is running, go to the serial console and enter the OTA command to start the HTTP Firmware Update for WINC1500.




MPLAB X IDE v4.00.rc1 - pic32_eth_wifi_web_server : pic32mz_ef_sk_ioexp_winc_freertos
pic32_eth_wifi_web_server
pic32_wifi_web_server
wifi_easy_configuration
MPLAB® Harmony Configurator*
Use Wi-Fi Driver?
Update Wi-Fi Module Firmware over Serial Port
Output
MPLAB® Harmony Configurator*


MPLAB X IDE v4.00.rc1 - pic32_wifi_web_server : pic32mx795_pim_e16_winc_freertos
pic32_eth_wifi_web_server
pic32_wifi_web_server
wifi_easy_configuration
MPLAB® Harmony Configurator*
Use Wi-Fi Driver?
Update Wi-Fi Module Firmware over Serial Port
Output
MPLAB® Harmony Configurator*



2. From the serial terminal console, issue the command, `ota http://xxx.xxx.xxx.xxx/m2m_ota_3a0.bin` (where `xxx.xxx.xxx.xxx` is the IP address of the HTTP/HTTPS Web server) to start the OTA firmware update.

```
TCP/IP Stack: Initialization Started
WINC1500: Initializing...
SYS_Initialize: The MPFS2 File System is mounted
====================================
*** Wi-Fi TCP/IP Web Server Demo ***
====================================
Chip ID 1503a0
DriverVerInfo: 0x13521352
WINC1500 Firmware Data:
Firmware Ver: 19.5.2 SVN Rev 14274
Firmware Built at Jan 26 2017 Time 22:13:34
Firmware Min Driver Ver: 19.3.0
Driver Ver: 19.5.2 SVN Rev 14200
Driver SVN URL branches/WIFIIOT-1660_19_5_2_RC6
Driver Built at Feb 10 2017 Time 13:53:00
Module MAC: F8:F0:05:F2:6C:BB
POWER SAVE Disabled
Start Wi-Fi Connection...
TCP/IP Stack: Initialization Ended - success
  Interface WINC1500 on host MCHPBOARD_W     - NBNS enabled
>Wi-Fi Connected
WINC1500 IPv4 Address: 192.168.1.5
>ota http://192.168.1.2/m2m_ota_3a0.bin
echoing target URL: http://192.168.1.2/m2m_ota_3a0_RC7_Final_OTA.bin
>Wi-Fi Disconnected
WINC1500: NVM operation succeeded
WINC1500: De-initializing...
WINC1500: Initializing...
Chip ID 1503a0
DriverVerInfo: 0x13521352
WINC1500 Firmware Data:
Firmware Ver: 19.5.2 SVN Rev 14274
Firmware Built at Jan 26 2017 Time 22:13:34
Firmware Min Driver Ver: 19.3.0
Driver Ver: 19.5.2 SVN Rev 14200
Driver SVN URL branches/WIFIIOT-1660_19_5_2_RC7
Driver Built at Feb 10 2017 Time 13:53:00
```

```
Module MAC: F8:F0:05:F2:6C:BB
POWER SAVE Disabled
Start Wi-Fi Connection...
Wi-Fi Connected
Wi-Fi IP is 192.168.1.5
OtaUpdateCb: UpdateStatusType = 1, UpdateStatus = 0
OtaUpdateCb: m2m_ota_switch_firmware starts
OtaUpdateCb: UpdateStatusType = 2, UpdateStatus = 0
OtaUpdateCb: OTA complete, please reset your board
```

Upgrade Using HTTPS Web Server

Provides information on updating the firmware OTA using a secure HTTPS Web server.

Description

This OTA Firmware upgrade process uses a SSL connection and requires preinstallation of the root certificate of the HTTPS Web server on the WINC1500 module. Refer to Firmware and Root Certificate Upgrade Over Serial Port for installing the root certificate. The firmware upgrade process is the same as in Upgrade Using HTTP Web Server, with the OTA command in Step 3 replaced with: `ota https://192.168.1.2//m2m_ota_3a0.bin`.

Appendix

Provides example output from the firmware update utility over the serial port.

Description

Example Output From the Firmware Update Utility Using a Serial Bridge

```
Mode UART
Downloading Image...
****************************************
*  >Programmer for WINC1500 SPI Flash<     *
*      Owner:  Microchip Corporation       *
****************************************
SVN REV 13880 SVN BR trunk
Built at Jan  9 2017    10:21:03
Firmware Path (3A0) ../../../firmware/m2m_aio_3a0.bin
>>Initialize programmer.
Detecting ports...
(APP)(INFO)WINC1500 Serial Bridge Found
Avail port COM1
1 of ports found
Chip id 1503a0
>Waiting for chip permission...
OK.

----- NOW Programming Firmware Image Version -----
Firmware ver   : 19.5.2 Svnrev 13922
Min driver ver : 19.3.0
Firmware Build Nov 10 2016 Time 21:23:14

----- Previous Firmware Image Version -----
Firmware ver   : 19.5.2 Svnrev 13922
Min driver ver : 19.3.0
Firmware Build Nov 10 2016 Time 21:23:14

Flash ID 1440ef
(APP)(INFO)Flash Size 8 Mb
>Start erasing...
Done
#Erase time = 0.858000 sec
>Start programming...
Done
#Programming time = 90.216000 sec

(APP)(INFO)----------- BEGIN EFUSE DUMP ----------------
(APP)(INFO)(Efuse)Ver = 0,bank idx = 0,used = 1,invalid = 0
(APP)(INFO)(Efuse)Valid = 1,MAC = f8:f0:05:f2:6e:04
(APP)(INFO)(Efuse)Valid = 0,PATxGainCorr = 00
```

```
(APP)(INFO)(Efuse)Valid = 1,FreqOffset = 0101
(APP)(INFO)------------- END EFUSE DUMP ----------------
Creating look up table for PLL with xo_offset = 4.0156.
>Start erasing...
Done
#Erase time = 0.047000 sec
>Start programming...
Done
#Programming time = 0.905000 sec

done

>>Image downloaded successfully.
(APP)(INFO)----------- BEGIN EFUSE DUMP ----------------
(APP)(INFO)(Efuse)Ver = 0,bank idx = 0,used = 1,invalid = 0
(APP)(INFO)(Efuse)Valid = 1,MAC = f8:f0:05:f2:6e:04
(APP)(INFO)(Efuse)Valid = 0,PATxGainCorr = 00
(APP)(INFO)(Efuse)Valid = 1,FreqOffset = 0101
(APP)(INFO)------------- END EFUSE DUMP ----------------

>>This task finished after 92.65 sec
Downloading WINC TLS Certificate Materials...
****************************************
*   WINC1500 TLS Certificate Flash Tool  *
****************************************
Detecting ports...
(APP)(INFO)WINC1500 Serial Bridge Found
Avail port COM1
1 of ports found
Chip id 1503a0
>Waiting for chip permission...
OK.
TLS Certificate Store Update Success on Flash
****************************************
*   WINC1500 TLS Certificate Flash Tool  *
****************************************
Detecting ports...
(APP)(INFO)WINC1500 Serial Bridge Found
Avail port COM1
1 of ports found
Chip id 1503a0
>Waiting for chip permission...
OK.
Detecting ports...
(APP)(INFO)WINC1500 Serial Bridge Found
Avail port COM1
1 of ports found
Chip id 1503a0
>Waiting for chip permission...
OK.
TLS Certificate Store Update Success on Flash
Downloading root certificates...
************************************************
* > WINC1500 Root Certificate Flash Downloader < *
************************************************
SVN REV 13880 SVN BR trunk
Built at Jan  4 2017    10:07:54
Detecting ports...
(APP)(INFO)WINC1500 Serial Bridge Found
Avail port COM1
1 of ports found
Chip id 1503a0
>Waiting for chip permission...
OK.

>>>Found Certificate:
>>>     AtmelEccCA
>Writing the certificate to SPI flash...
Done
```

```
>>>Found Certificate:
>>>     Baltimore CyberTrust Root
>Writing the certificate to SPI flash...
Done

>>>Found Certificate:
>>>     DigiCert SHA2 High Assurance Server CA
>Writing the certificate to SPI flash...
Done
>>>Found Certificate:
>>>     Entrust Root Certification Authority
>Writing the certificate to SPI flash...
Done

>>>Found Certificate:
>>>     GlobalSign Root CA
>Writing the certificate to SPI flash...
Done

>>>Found Certificate:
>>>
>Writing the certificate to SPI flash...
Done

>>>Found Certificate:
>>>     Sample Matrix RSA-2048 Certificate Authority
>Writing the certificate to SPI flash...
Done

>>>Found Certificate:
>>>     AddTrust External CA Root
>Writing the certificate to SPI flash...
Done

>>>Found Certificate:
>>>
>Writing the certificate to SPI flash...
Done

>>>Found Certificate:
>>>     GeoTrust Global CA
>Writing the certificate to SPI flash...
Done

>>>Found Certificate:
>>>     QuoVadis Root CA 2
>Writing the certificate to SPI flash...
Done

>>>Found Certificate:
>>>     VeriSign Class 3 Public Primary Certification Authority - G5
>Writing the certificate to SPI flash...
Done

>>>Found Certificate:
>>>     WINCRootCA
>Writing the certificate to SPI flash...
Done
All certificates have been downloaded
OK
    ##########################################################################
    ##                                                                      ##
    ##                 ########     ##      ######   ######               ##
    ##                 ##     ##   ## ##    ##    ## ##    ##              ##
    ##                 ##     ##  ##   ##   ##       ##                    ##
    ##                 ########  ##     ##  ######   ######               ##
    ##                 ##        ########        ##       ##              ##
    ##                 ##        ##     ## ##    ## ##    ##              ##
```

```
    ##                ##        ##     ##  ######   ######             ##
    ##                                                                  ##
    ######################################################################
Downloading ends successfully
Press any key to continue . . .
```

Index