WINC1500 MLA Getting Started Guide

Version 1.0

[This page left blank]

Table of Contents

Revision History

Version	Changes	Release Date
1.0	• Initial Release	26 JAN 2017

1 Introduction

This getting started guide describes the Microchip WINC1500 Wi-Fi Network Controller used for building state-of-the-art Internet of Things (IoT) applications. The guide explains hardware information and how to install MPLAB X (IDE), compile examples, and download WINC1500 firmware within the Microchip Libraries for Applications (MLA) framework. The following topics will be covered:

- How to get MPLABX IDE and install it
- How to download the Microchip MLA framework
- Target board information
- How to run the Wi-Fi scan demo project
- How to get log messages from the WINC1500
- How to download firmware and certificates

For detailed information on the WINC1500 MLA driver, API calls, and event handling please see the *WINC1500 MLA User's Guide*.

2 Prerequisites

2.1 Hardware Prerequisites

The WINC1500 MLA software projects give you a choice of three different hardware platforms.

Table 2-1: Hardware Platform Options

Development Platforms	MCU	Comment
Option 1: • Microchip **Explorer 16** Development Board (DM240001) • MPLAB ICD3 In-Circuit Debugger (DV164035) • 9V Power supply (AC002014) • DB9 Serial Cable	**PIC24FJ128GA310** Plug-in Module (MA240029)	• PIC24 MCU (16-bit) • Supports WINC1500 PICtail Plus
Option 2: • Microchip **Explorer 16/32** Development Board • (DM240001-2) • Microchip **PICtail Plus Expansion Board** (AC240100) • MPLAB ICD3 In-Circuit Debugger (DV164035) • Micro USB to USB (for serial communications and power)	**PIC24FJ128GA310** Plug-in Module (MA240029)	• PIC24 MCU (16-bit) • Supports WINC1500 PICtail Plus or mikroBUS Click board
Option 3: • **Explorer 8** Development Board (DM160228) • MPLAB ICD3 In-Circuit Debugger (DV164035) • Micro USB to USB (for serial communications and power)	**PIC18F87K22** Plug-in Module (MA180028)	• PIC18 MCU (8-bit) • Supports WINC1500 PICtail or mikroBUS Click board

2.2 Software Prerequisites

Software	Available at:
MPLAB X IDE	http://www.microchip.com/mplab/mplab-x-ide
MPLAB XC16 toolchain	http://www.microchip.com/mplab/compilers
MPLAB XC8 toolchain	http://www.microchip.com/mplab/compilers
WINC1500 MLA Software	http://www.microchip.com/mplab/microchip-libraries-for-applications

3 Getting Started with MPLAB® X IDE

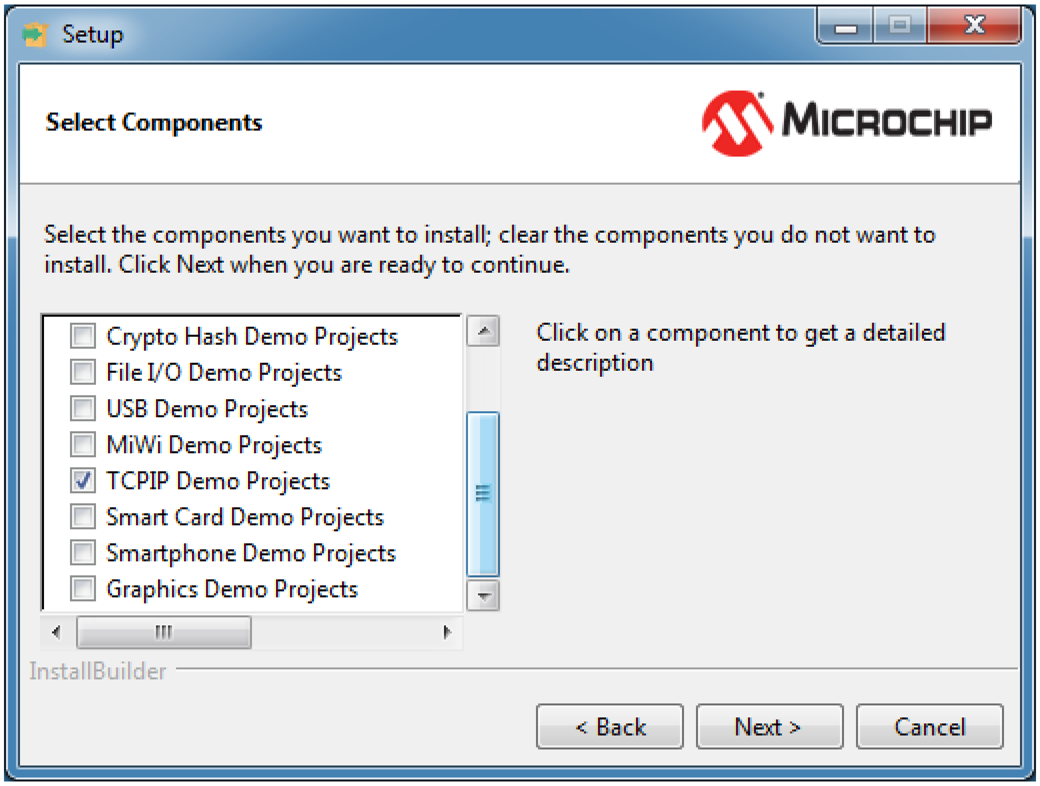
The MPLAB X IDE is a software program that is used to develop applications for Microchip microcontrollers. The MPLAB X IDE gives you a seamless and easy-to-use environment to write, build, and debug applications written in C/C++ or assembly code. MPLAB X IDE runs on Windows®, MAC OS®, or Linux®.

To get the necessary software perform the following steps:

1) Download and install MPLAB X IDE from http://www.microchip.com/mplab/mplab-x-ide. Choose the Windows, MAC OS X, or Linux version depending on the development platform being used.

2) Download the required toolchain – MPLAB XC8 for the PIC18 or MPLAB XC16 for the PIC24. The toolchain can be downloaded from http://www.microchip.com/mplab/compilers. Choose the Windows, MAC OS X, or Linux version depending on the development platform being used.

3) Download and install the MLA software framework, containing the WINC1500 projects, from http://www.microchip.com/mplab/microchip-libraries-for-applications. Choose the Windows, MAC OS X, or Linux version depending on the development platform being used. You can download the entire MLA if desired, but the only component needed for the WINC1500 projects are contained in the 'TCPIP Demo Projects'. Below is the 'Select Components' window that appears during the installation showing the required component checked:



Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [ ] Crypto Hash Demo Projects
- [ ] File I/O Demo Projects
- [ ] USB Demo Projects
- [ ] MiWi Demo Projects
- [✓] TCPIP Demo Projects
- [ ] Smart Card Demo Projects
- [ ] Smartphone Demo Projects
- [ ] Graphics Demo Projects

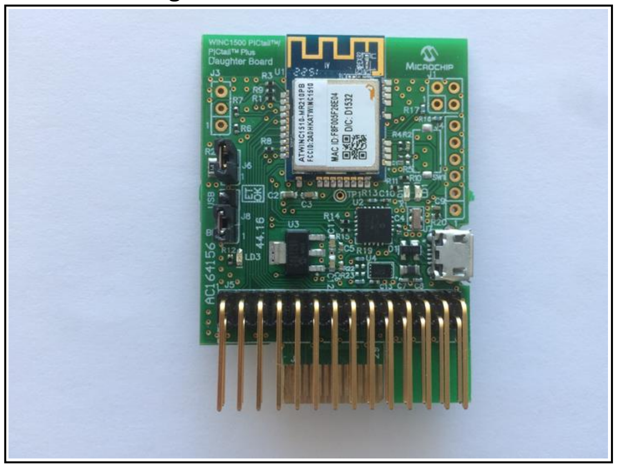
Click on a component to get a detailed description

The MPLABX, the toolchains, and MLA software are free of charge. The toolchains can be purchased if it is desired to take full advantage of the compiler optimization features.

4 Getting Started with WINC1500 PICtail

This section describes how to configure the three platform options for debugging. All three platforms use the WINC1500 PICtail, shown below.

Figure 4-1: WINC1500 PICtail

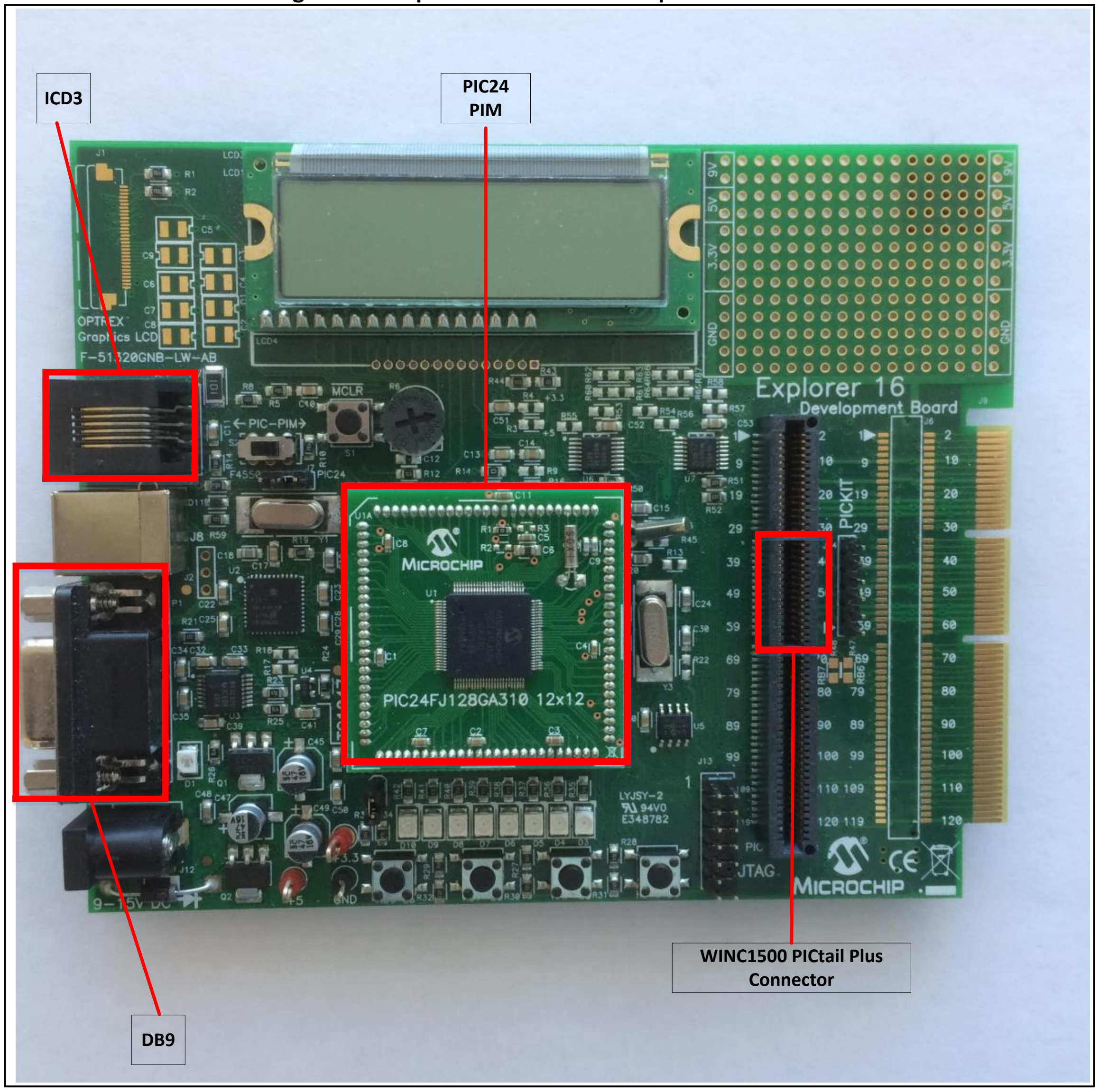


The module supports two connectors. The edge connector is the PICtail Plus, which is used for the Explorer 16 and Explorer 16/32 boards. The pin header is for a PICtail connector, and is used for the Explorer 8 board.

4.1 Explorer 16 with Development Platform

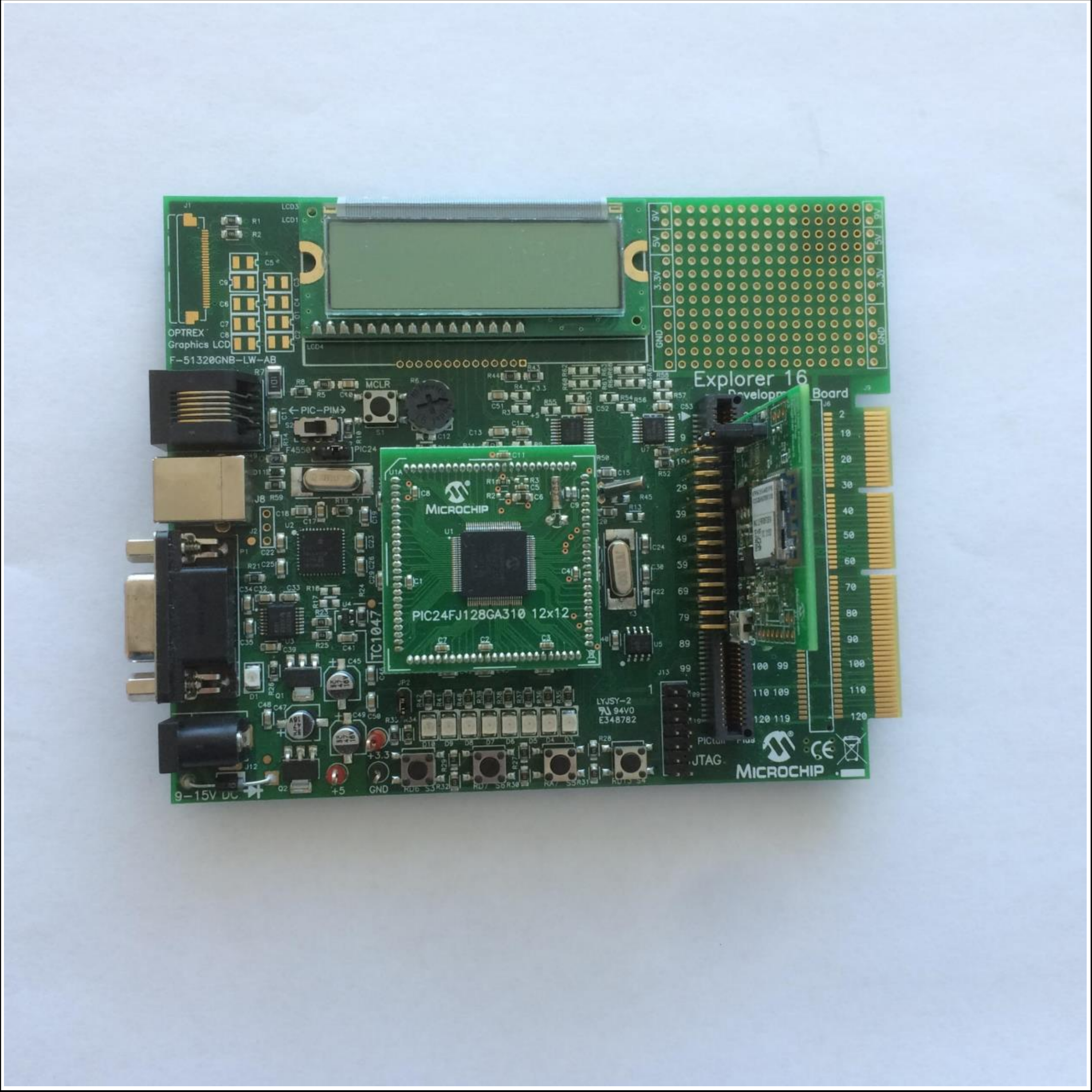
Below is a picture of the Explorer 16 board with needed features labeled. Note the WINC1500 PICtail module plugs into the second slot of the PICTail Plus connector.

Figure 4-2: Explorer 16 Board Development Platform



Below is a picture of the Explorer 16 board with the WINC1500 PICtail Plus module plugged into the PICtail Plus connector. Note the WINC1500 module faces towards the PIC24 PIM.

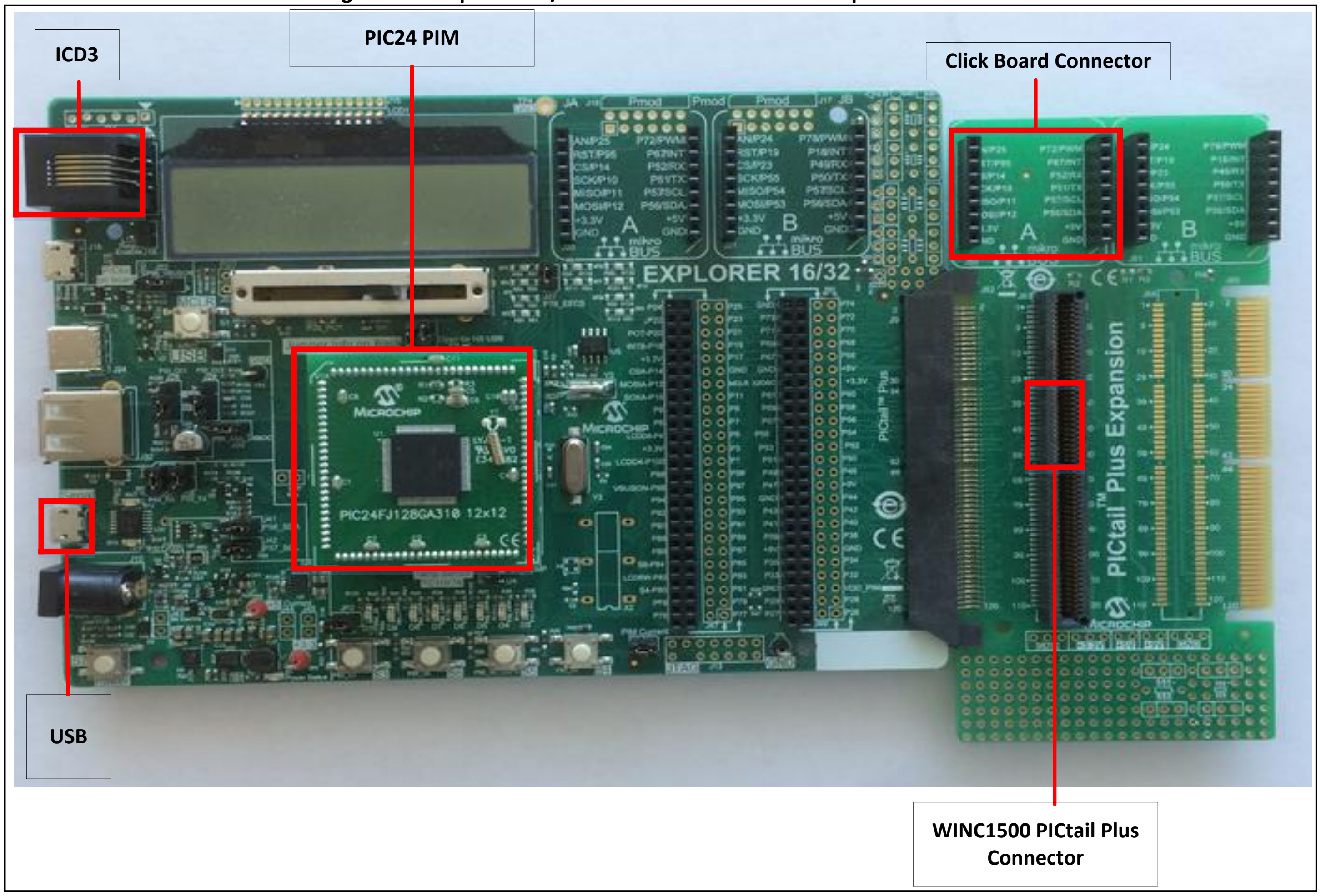
Figure 4-3: Explorer 16 Board with WINC1500 Module



4.2 Explorer 16/32 Development Platform

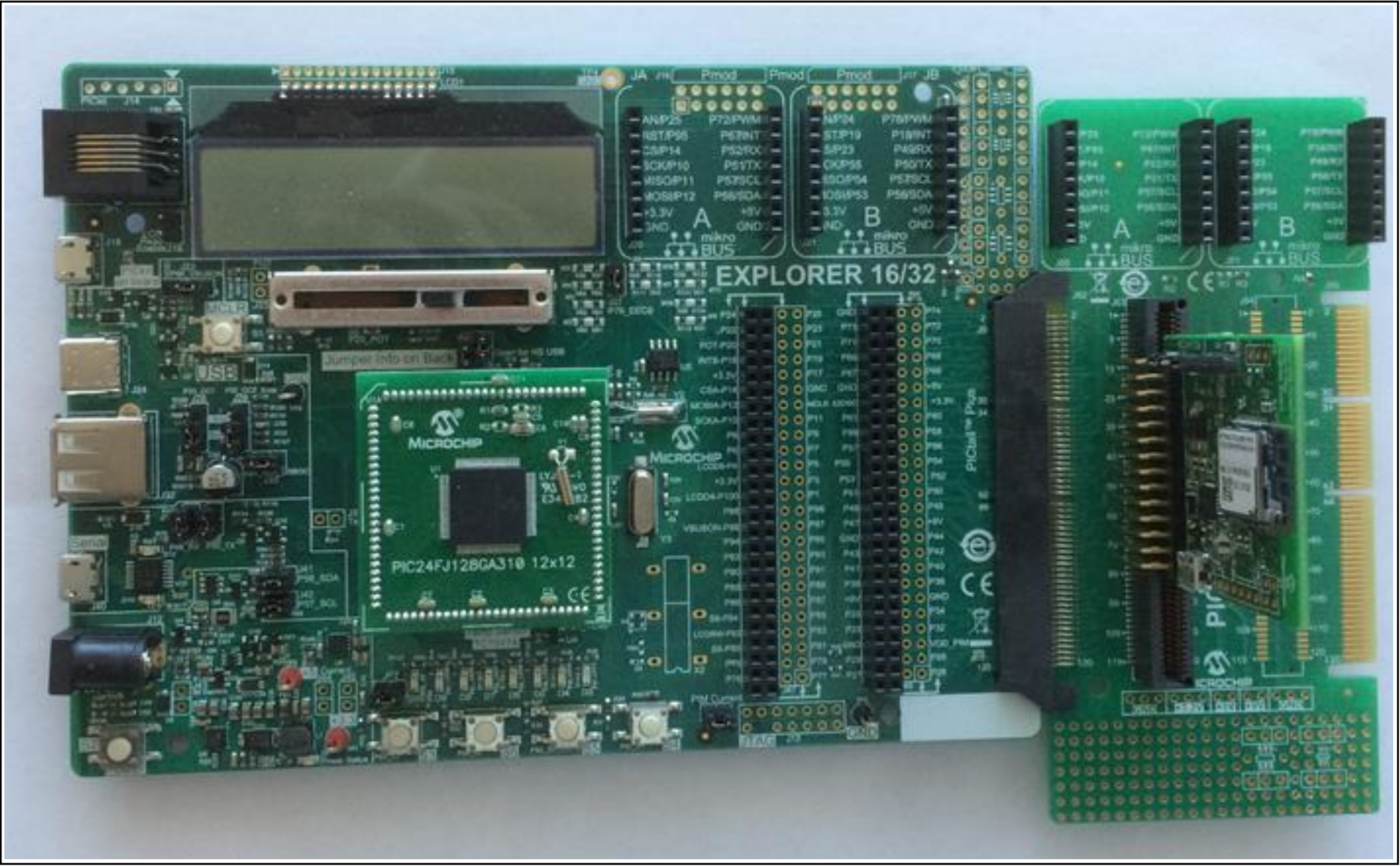
Below is a picture of the Explorer 16/32 board with attached PICtail Plus Expansion board plugged in (on the right side). Note the WINC1500 PICtail module plugs into the second slot of the PICTail Plus connector.

Figure 4-4: Explorer 16/32 Board with PICTail Plus Expansion Board



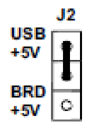
Below is a picture of the Explorer 16/32 and PICtail Plus Expansion board with the WINC1500 module plugged in.

Figure 4-5: Explorer 16/32 and PICtail Expansion with WINC1500

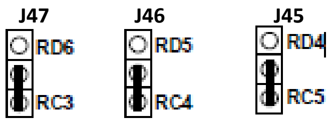


4.3 Explorer 8 Development Platform

In order to run this board using USB power and have a serial connection, be sure to configure Jumper J2 as shown below. If the jumper is in the other position you will need a 9V supply plugged into J1.

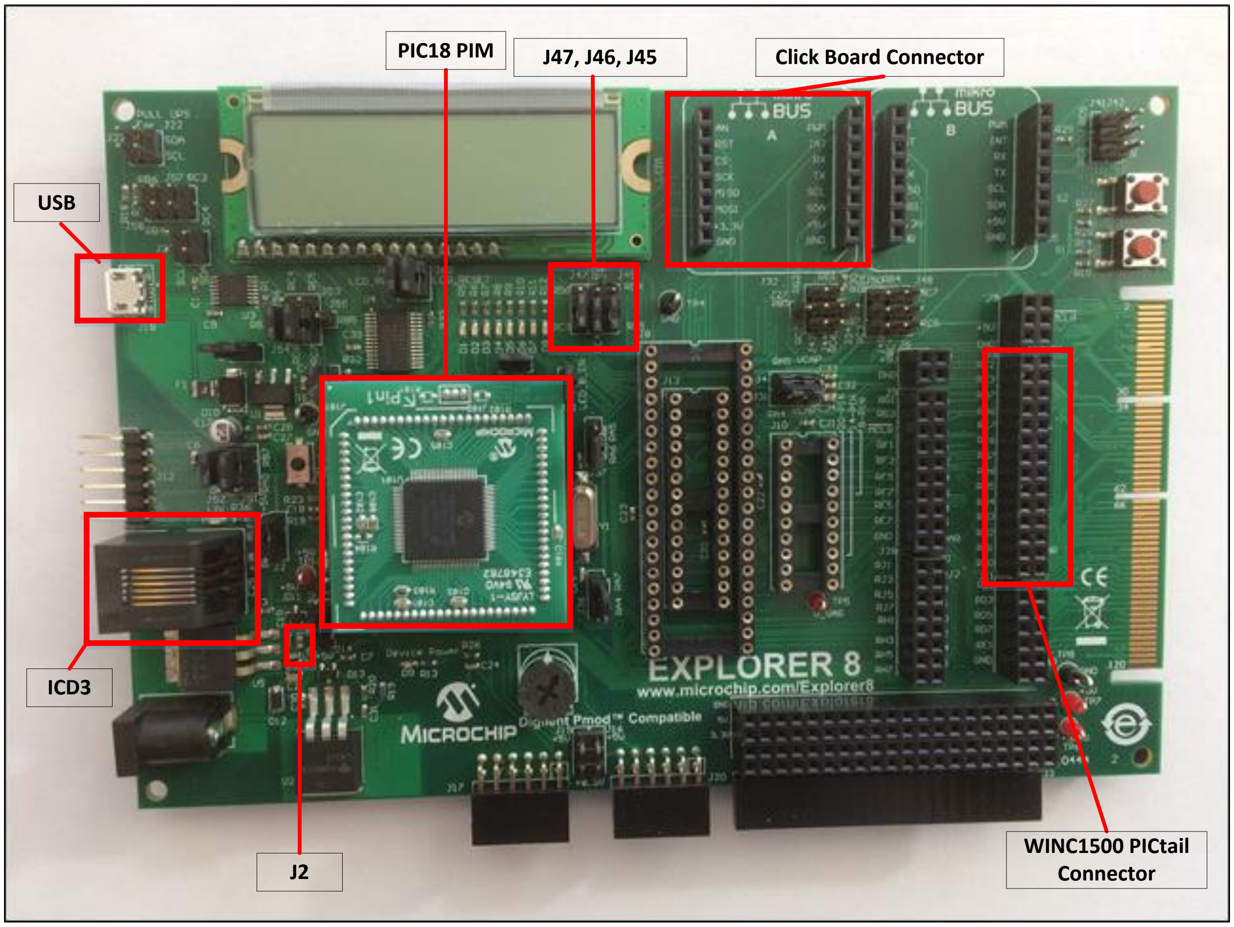


In addition, three other jumpers are required to connect Port pins RC3, RC4, and RC5 to the MCU. These jumpers are not installed by default.



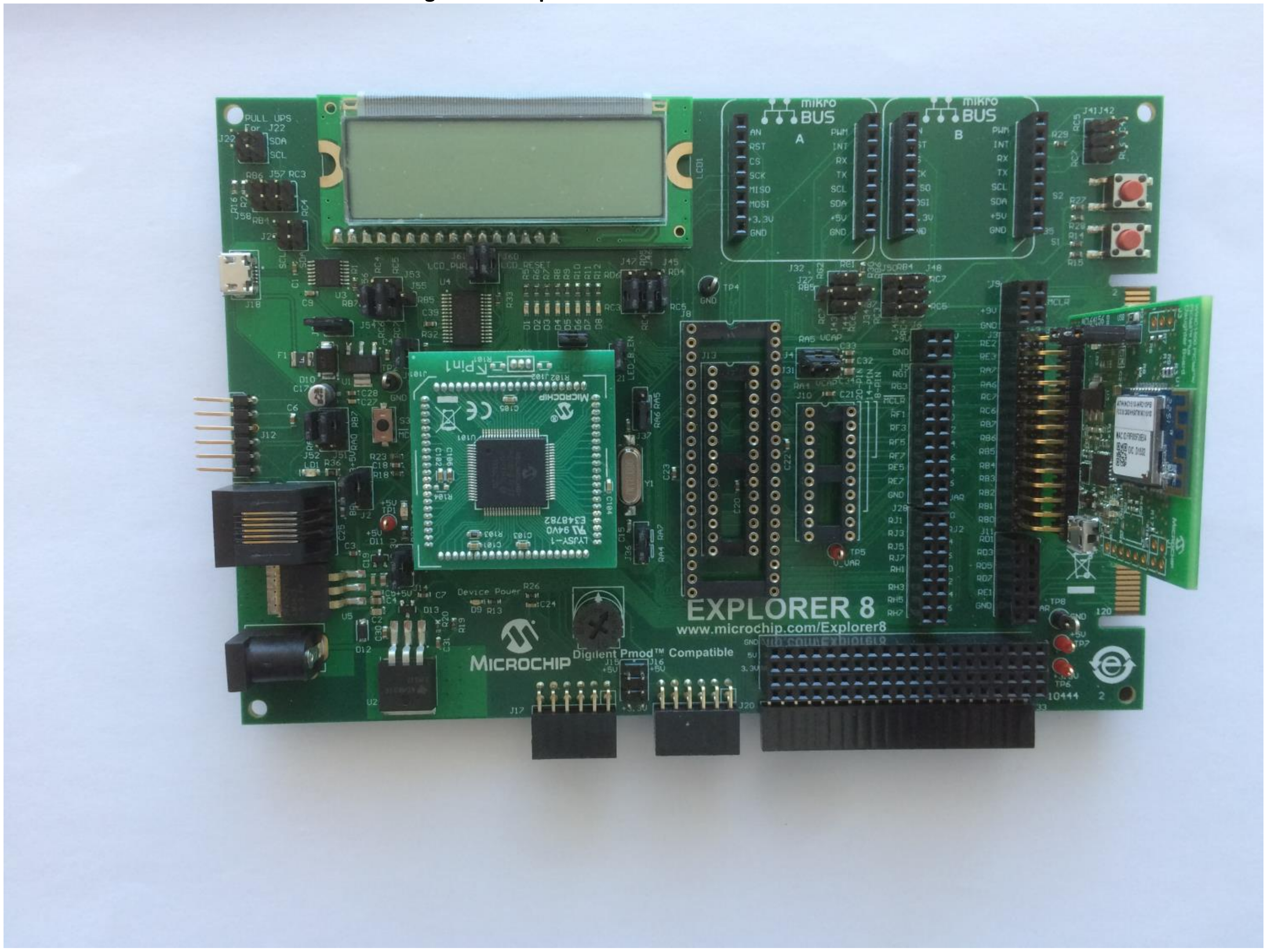
Below is a picture of the Explorer 8 board showing the PIC18F87K22 PIM plugged into the PIM connector. Note the orientation of the PIM (appears upside down, but is correct). The WINC1500 PICtail is plugged into the PICtail connector with the WINC1500 module facing towards the PIM. The board comes with a PIC16 plugged into J8 – be sure to remove it.

Figure 4-6: Explorer 8 Development Board



Below is the Explorer 8 board with the WINC1500 module plugged into the PICtail connector.

Figure 4-7: Explorer 8 with WINC1500 Module



5 Getting Started with WINC1500 IoT Demos

This section introduces WINC1500 IoT demos and how to run them. The steps are the same, whether running the PIC24 or PIC18 demos.

5.1 Directory Structure

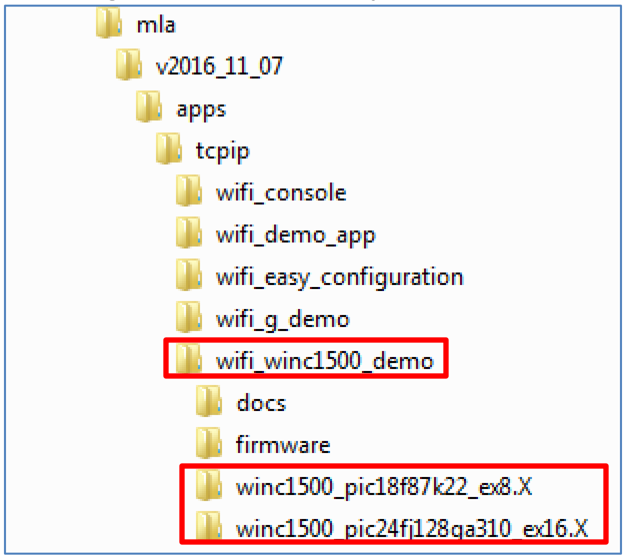
This section discusses where to find WINC1500 project and driver files after installation.

5.1.1 WINC1500 Project Directories

The two sample projects for the WINC1500 are located under the wifi_winc1500_demo directory as shown below. There are two MPLABX sample project directories:

- winc1500_pic18f87k22_ex8.X for the PIC18 MCU running on the Explorer 8 board.
- winc1500_pic24fj128ga310_ex16 for the PIC24 MCU running on either the Explorer 16 or the Explorer 16/32 boards

Figure 5-1: WINC1500 Project Directories

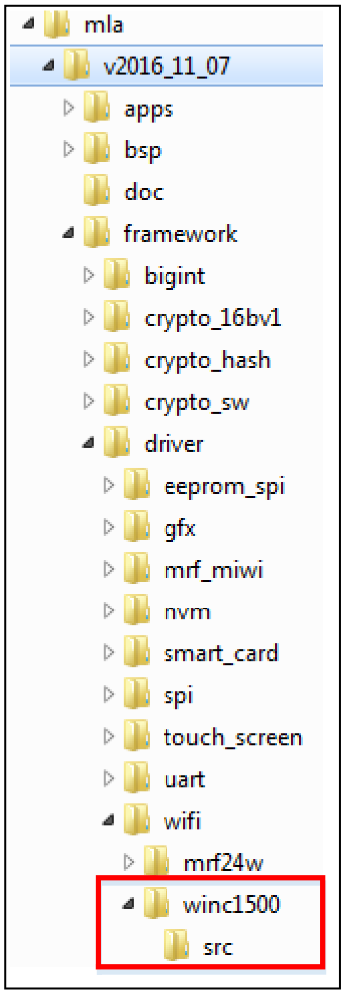


```
mla
  v2016_11_07
    apps
      tcpip
        wifi_console
        wifi_demo_app
        wifi_easy_configuration
        wifi_g_demo
        wifi_winc1500_demo
          docs
          firmware
          winc1500_pic18f87k22_ex8.X
          winc1500_pic24fj128ga310_ex16.X
```

5.1.2 WINC1500 Driver Directory

The WINC1500 driver files are located under the winc1500 sub-directory as shown below.

Figure 5-2: WINC1500 Driver Directory



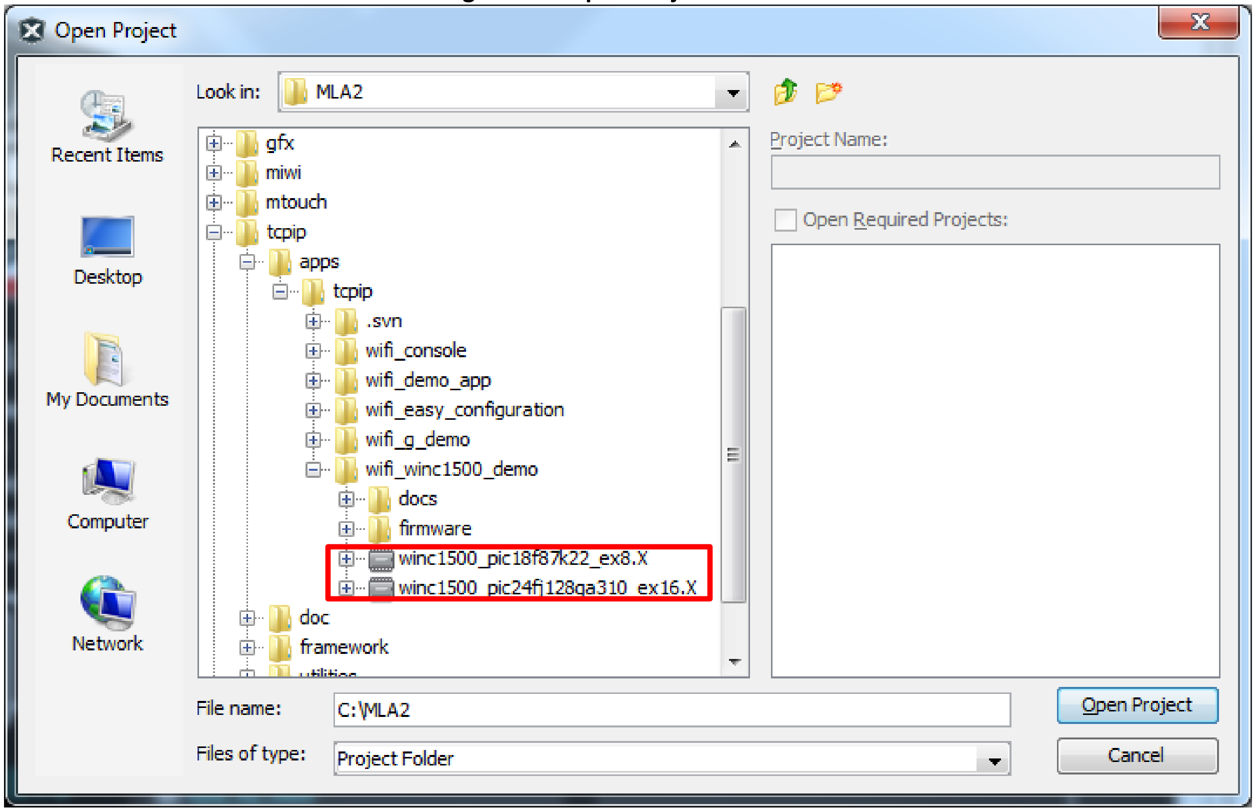
```
mla
  v2016_11_07
    apps
    bsp
    doc
    framework
      bigint
      crypto_16bv1
      crypto_hash
      crypto_sw
      driver
        eeprom_spi
        gfx
        mrf_miwi
        nvm
        smart_card
        spi
        touch_screen
        uart
        wifi
          mrf24w
          winc1500
            src
```

5.2 WINC1500 Demos

Open MPLAB X IDE and select one of the projects (PIC18 or PIC24). From the MPLA X menu select 'File', then 'Open Project'. In the window browse down to the projects and select the desired project.

For the purpose of this discussion, the PIC18 project will be used. The following instructions with work the PIC24 project as well.

Figure 5-3: Open Project Window



Look in: MLA2

```
gfx
miwi
mtouch
tcpip
  apps
    tcpip
      .svn
      wifi_console
      wifi_demo_app
      wifi_easy_configuration
      wifi_g_demo
      wifi_winc1500_demo
        docs
        firmware
        winc1500_pic18f87k22_ex8.X
        winc1500_pic24fj128ga310_ex16.X
  doc
  framework
```

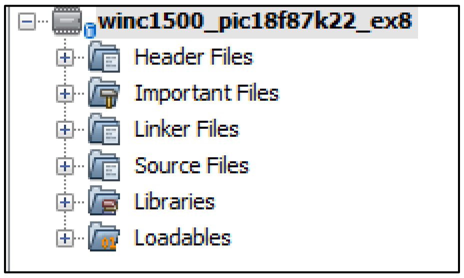
File name: C:\MLA2

Files of type: Project Folder

5.2.1 Select Demo to Run

After the project is opened, the project window should show the following:

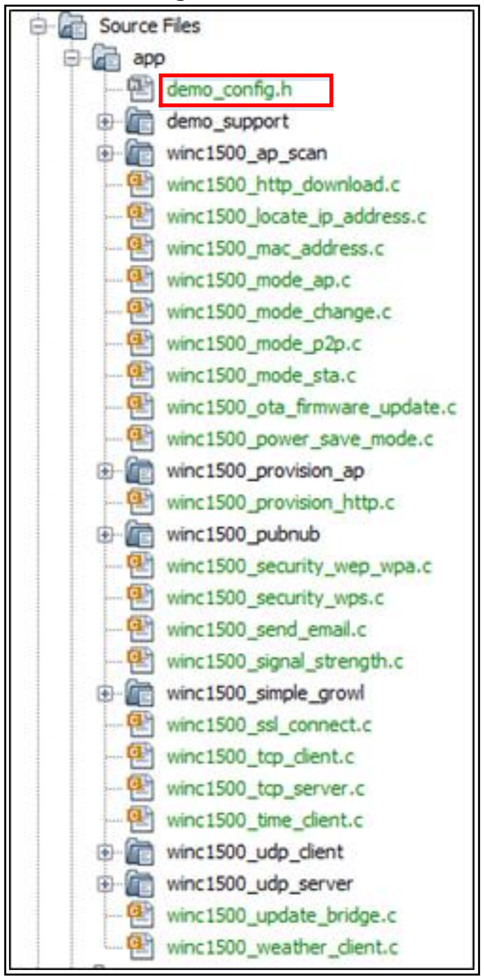
Figure 5-4: Project Window



```
winc1500_pic18f87k22_ex8
  Header Files
  Important Files
  Linker Files
  Source Files
  Libraries
  Loadables
```

Right-click on the project name tab and select ‘Set as Main Project’. This will cause the project name to display in a bold font. Expand the ‘Source Files’ tab, then expand the ‘app’ tab, and you should see all the demos that can be run by the project. Only one demo can be selected at a time. The active demo is selected in the demo_config.h file.

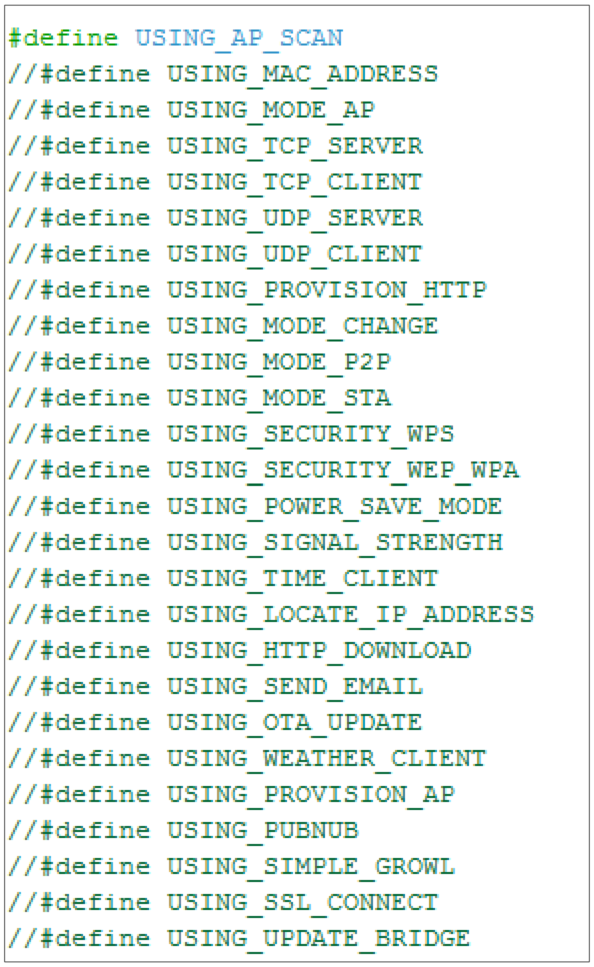
Figure 5-5: Demo Files



```
Source Files
  app
    demo_config.h
    demo_support
    winc1500_ap_scan
    winc1500_http_download.c
    winc1500_locate_ip_address.c
    winc1500_mac_address.c
    winc1500_mode_ap.c
    winc1500_mode_change.c
    winc1500_mode_p2p.c
    winc1500_mode_sta.c
    winc1500_ota_firmware_update.c
    winc1500_power_save_mode.c
    winc1500_provision_ap
    winc1500_provision_http.c
    winc1500_pubnub
    winc1500_security_wep_wpa.c
    winc1500_security_wps.c
    winc1500_send_email.c
    winc1500_signal_strength.c
    winc1500_simple_growl
    winc1500_ssl_connect.c
    winc1500_tcp_client.c
    winc1500_tcp_server.c
    winc1500_time_client.c
    winc1500_udp_client
    winc1500_udp_server
    winc1500_update_bridge.c
    winc1500_weather_client.c
```

Double-click on demo_config.h to open it. There is a block of defines, all of which are commented out except one – that is the demo that will be built and run. Shown below are the demo defines – the AP Scan Demo is uncommented and when the project is built this is the demo that will be built and run. To run any other demo, simply ensure the desired demo is uncommented, and all other demos are commented out. Then build the project and run the demo.

Figure 5-6: Demo-selection defines

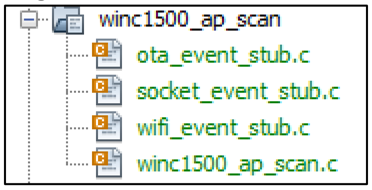


```
#define USING_AP_SCAN
//#define USING_MAC_ADDRESS
//#define USING_MODE_AP
//#define USING_TCP_SERVER
//#define USING_TCP_CLIENT
//#define USING_UDP_SERVER
//#define USING_UDP_CLIENT
//#define USING_PROVISION_HTTP
//#define USING_MODE_CHANGE
//#define USING_MODE_P2P
//#define USING_MODE_STA
//#define USING_SECURITY_WPS
//#define USING_SECURITY_WEP_WPA
//#define USING_POWER_SAVE_MODE
//#define USING_SIGNAL_STRENGTH
//#define USING_TIME_CLIENT
//#define USING_LOCATE_IP_ADDRESS
//#define USING_HTTP_DOWNLOAD
//#define USING_SEND_EMAIL
//#define USING_OTA_UPDATE
//#define USING_WEATHER_CLIENT
//#define USING_PROVISION_AP
//#define USING_PUBNUB
//#define USING_SIMPLE_GROWL
//#define USING_SSL_CONNECT
//#define USING_UPDATE_BRIDGE
```

5.2.2 Configure Demo by Modifying Configuration Defines

Let's continue with the AP scan demo. If you expand the 'winc1500_ap_scan' tab in the project window you will see this demo consists of 4 files.

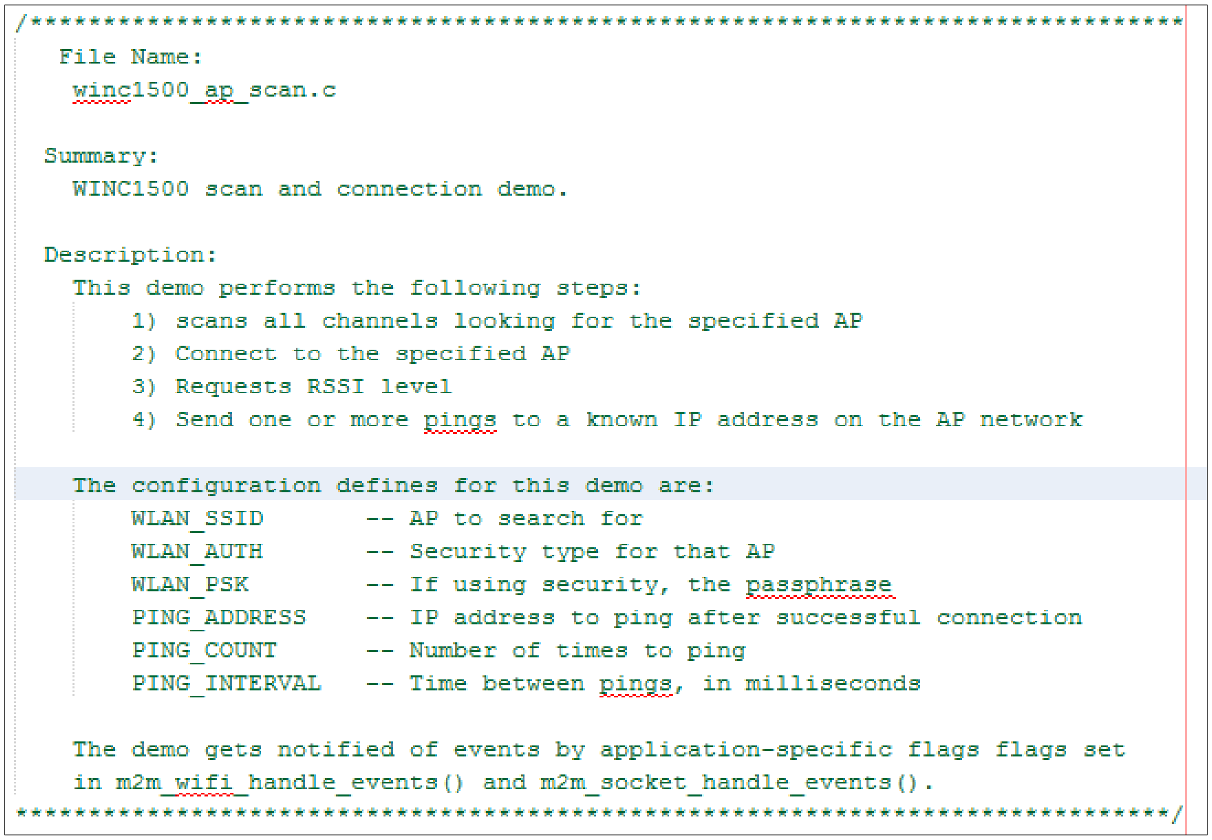
Figure 5-7: AP Scan Demo Files



```
winc1500_ap_scan
    ota_event_stub.c
    socket_event_stub.c
    wifi_event_stub.c
    winc1500_ap_scan.c
```

The main file is winc1500_ap_scan.c. Open that file for a description of what the demo does. The file header, reproduced here, gives a description of what the demo does and how to configure the demo. The other files handle the various events that can be generated, many of which are not used in this demo.

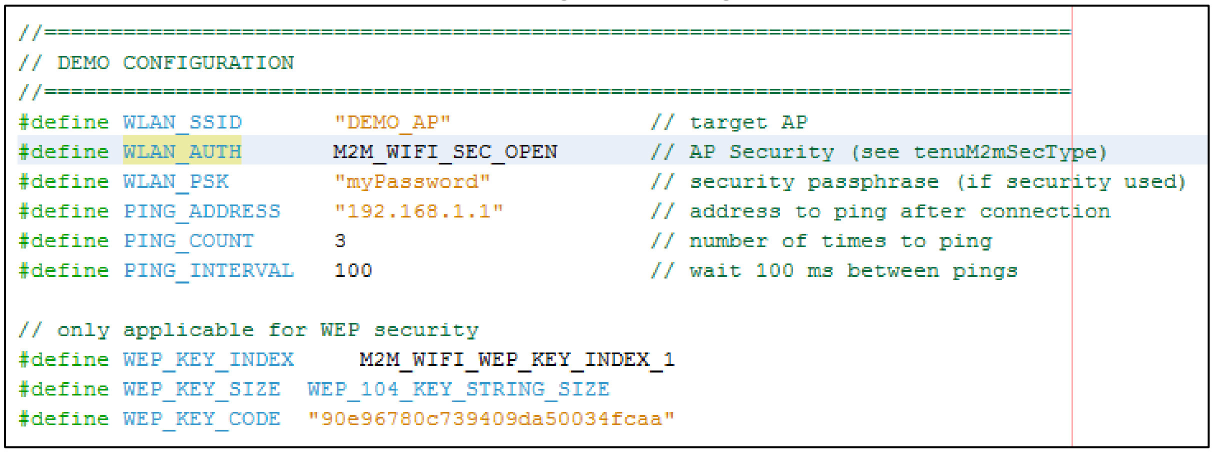
Figure 5-8: File Header Comment for winc1500_ap_scan.c



```
/*******************************************************************************
   File Name:
    winc1500_ap_scan.c

  Summary:
    WINC1500 scan and connection demo.

  Description:
    This demo performs the following steps:
        1) scans all channels looking for the specified AP
        2) Connect to the specified AP
        3) Requests RSSI level
        4) Send one or more pings to a known IP address on the AP network

    The configuration defines for this demo are:
        WLAN_SSID        -- AP to search for
        WLAN_AUTH        -- Security type for that AP
        WLAN_PSK         -- If using security, the passphrase
        PING_ADDRESS     -- IP address to ping after successful connection
        PING_COUNT       -- Number of times to ping
        PING_INTERVAL    -- Time between pings, in milliseconds

    The demo gets notified of events by application-specific flags flags set
    in m2m_wifi_handle_events() and m2m_socket_handle_events().
*******************************************************************************/
```

Further down in the file is the following block of #define's that need to be modified to run the demo in your environment.

Figure 5-9: Configuration Defines



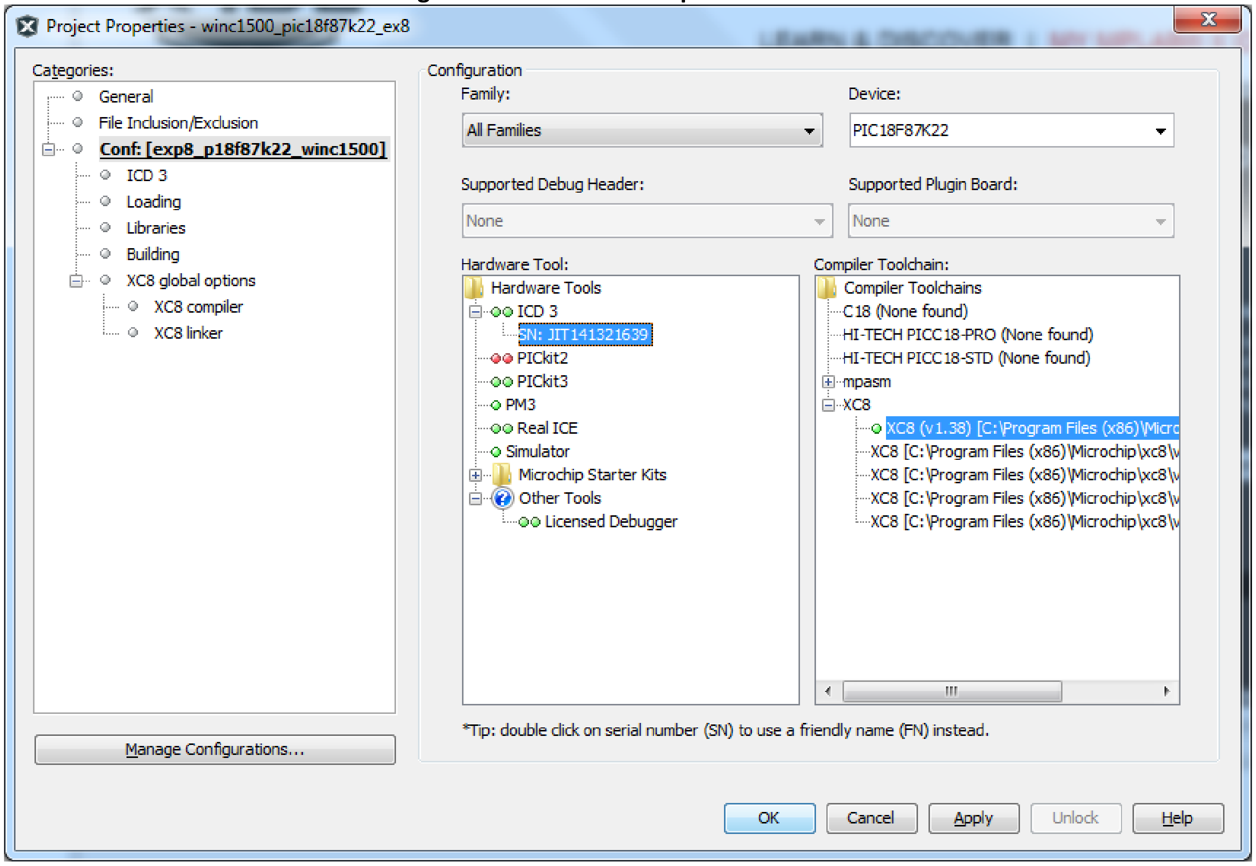
```
//==============================================================================
// DEMO CONFIGURATION
//==============================================================================
#define WLAN_SSID        "DEMO_AP"              // target AP
#define WLAN_AUTH        M2M_WIFI_SEC_OPEN      // AP Security (see tenuM2mSecType)
#define WLAN_PSK         "myPassword"           // security passphrase (if security used)
#define PING_ADDRESS     "192.168.1.1"          // address to ping after connection
#define PING_COUNT       3                      // number of times to ping
#define PING_INTERVAL    100                    // wait 100 ms between pings

// only applicable for WEP security
#define WEP_KEY_INDEX      M2M_WIFI_WEP_KEY_INDEX_1
#define WEP_KEY_SIZE   WEP_104_KEY_STRING_SIZE
#define WEP_KEY_CODE   "90e96780c739409da50034fcaa"
```

Update the above define's as needed for your environment.

5.2.3 Build Demo

Before building the project, ensure the compiler and ICD3 are present and selected. Right-click the project name tab in the project window and select properties. The ICD3 must be plugged into the PC in order for its serial number to appear as shown below. Select the ICD3 for this project by clicking on its Serial Number. Select the XC8 compiler in the same way. In this example, the PC has multiple versions of the XC8 compiler, which is why there is more than one choice. Note that the XC8 v1.38 is selected. Click OK to save changes and close the Properties window.

Figure 5-10: MPLAB X Properties Window



Project Properties - winc1500_pic18f87k22_ex8

Categories:
- General
- File Inclusion/Exclusion
- Conf: [exp8_p18f87k22_winc1500]
  - ICD 3
  - Loading
  - Libraries
  - Building
  - XC8 global options
    - XC8 compiler
    - XC8 linker

Configuration

Family: All Families

Device: PIC18F87K22

Supported Debug Header: None

Supported Plugin Board: None

Hardware Tool:
- Hardware Tools
  - ICD 3
    - SN: JIT141321639
  - PICkit2
  - PICkit3
  - PM3
  - Real ICE
  - Simulator
  - Microchip Starter Kits
  - Other Tools
    - Licensed Debugger

Compiler Toolchain:
- Compiler Toolchains
  - C18 (None found)
  - HI-TECH PICC18-PRO (None found)
  - HI-TECH PICC18-STD (None found)
  - mpasm
  - XC8
    - XC8 (v1.38) [C:\Program Files (x86)\Micro
    - XC8 [C:\Program Files (x86)\Microchip\xc8\v
    - XC8 [C:\Program Files (x86)\Microchip\xc8\v
    - XC8 [C:\Program Files (x86)\Microchip\xc8\v
    - XC8 [C:\Program Files (x86)\Microchip\xc8\v

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

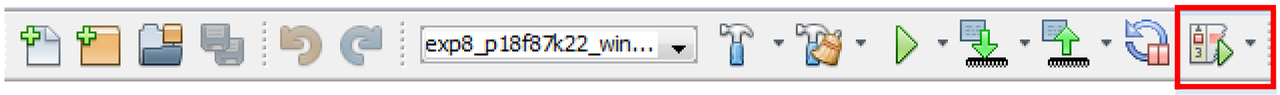
Now it is time to ensure the project builds. To do that, right click on the project tab in the project window and select 'Clean and Build'. The build results appear in the Output Window.

5.2.4 Run the Demo

Before running the demo, the development platform needs to powered and connected:

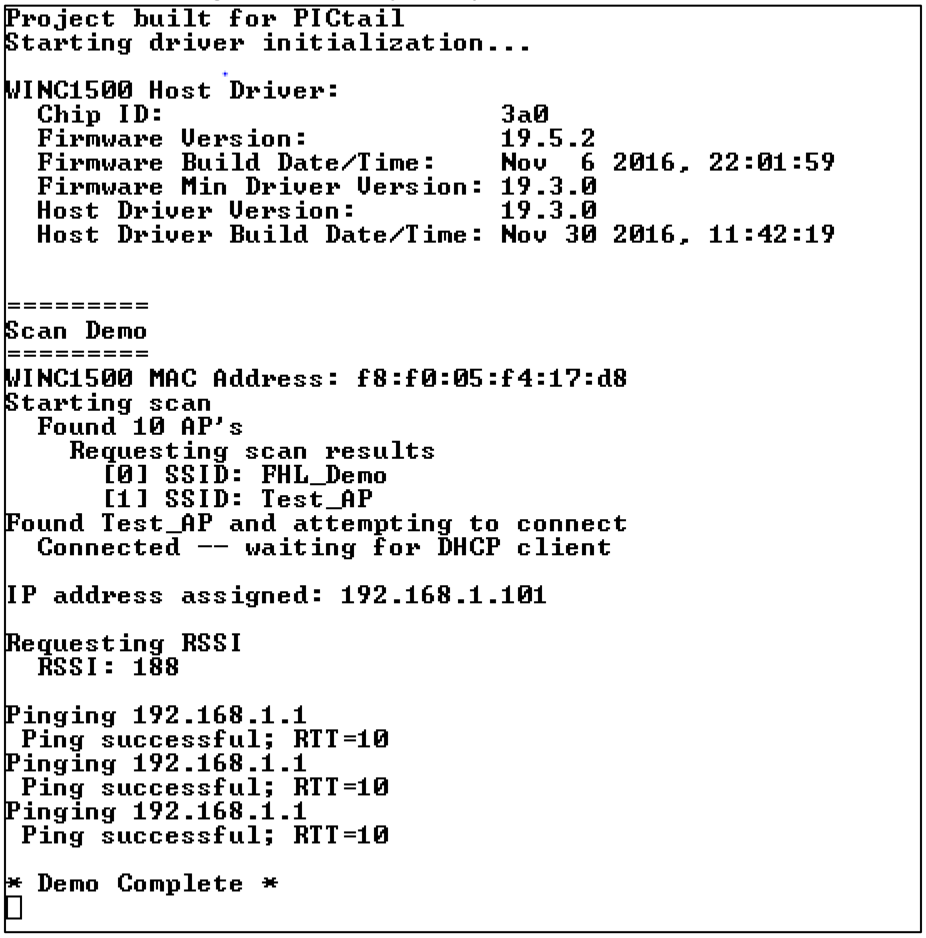
1) Plug the ICD3 into the board connector
2) Plug the WINC1500 PICtail into the board (see *Hardware Prerequisites*)
3) Connect the DB9 (or USB serial) from the board to the PC
4) Power up the board (via USB or 9V power supply)
5) Bring up a terminal window on the PC for the associated COM port and configure 115200 Baud, 8 data bits, 1 stop bit, no parity, no flow control.

After preparing the development platform, from MPLABX, run the demo. To run the demo in debug mode, simply click the 'Debug Main Project' icon on the MPLAB X toolbar (shown in red below).



The project will build again, download to the board and run. A successful run will show the output similar to Figure 5-11 below.

Figure 5-11: Example Output from AP Scan Demo

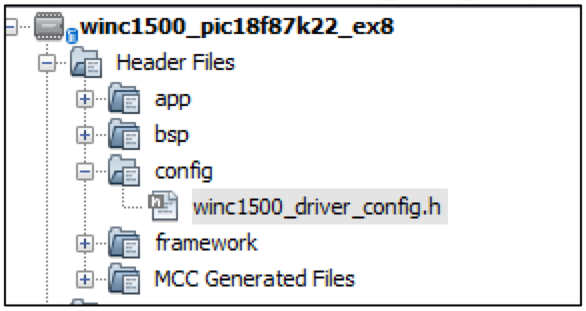


```
Project built for PICtail
Starting driver initialization...

WINC1500 Host Driver:
  Chip ID:                      3a0
  Firmware Version:             19.5.2
  Firmware Build Date/Time:     Nov  6 2016, 22:01:59
  Firmware Min Driver Version: 19.3.0
  Host Driver Version:          19.3.0
  Host Driver Build Date/Time: Nov 30 2016, 11:42:19


=========
Scan Demo
=========
WINC1500 MAC Address: f8:f0:05:f4:17:d8
Starting scan
  Found 10 AP's
    Requesting scan results
      [0] SSID: FHL_Demo
      [1] SSID: Test_AP
Found Test_AP and attempting to connect
  Connected -- waiting for DHCP client

IP address assigned: 192.168.1.101

Requesting RSSI
  RSSI: 188

Pinging 192.168.1.1
 Ping successful; RTT=10
Pinging 192.168.1.1
 Ping successful; RTT=10
Pinging 192.168.1.1
 Ping successful; RTT=10

* Demo Complete *
```

6 WINC1500 Driver Configuration

The WINC1500 Driver can be configured in a variety of ways. This is controlled by the **winc1500_driver_config.h** file, located at the following project location.

Figure 6-1: Location of winc1500_driver_config.h



Below is the list of configuration defines in this file.

Table 6-1: Driver Configuration Defines

Configuration Define	Description
`USING_PICTAIL` or `USING_CLICK_BOARD`	One of these defines must be enabled, the other must be commented out. Currently, only the PICtail is available, so USING_PICTAIL must be enabled.
`HOST_MCU_BIG_ENDIAN`	Comment out if host MCU is little-endian, else uncomment
`M2M_POINTER_SIZE_IN_BYTES`	Set this define to the size (number of bytes) in a pointer type. This can be determined by executing the code: int x = sizeof(int *);
`M2M_ENABLE_ERROR_CHECKING`	Enable to allow error checking; comment out to disable error checking
`M2M_ENABLE_PRNG`	If not using PRNG API functions then comment out to save code space
`M2M_ENABLE_SOFT_AP_MODE`	If not using Soft AP then comment out to save code space
`M2M_ENABLE_WPS`	If not using WPS then comment out to save code space
`M2M_WIFI_ENABLE_P2P`	If not using P2P then comment out to save code space
`M2M_ENABLE_HTTP_PROVISION_MODE`	If not using HTTP provisioning then comment out to save code space
`M2M_ENABLE_SCAN_MODE`	If not using scanning then comment out to save code space
`M2M_ENABLE_SPI_FLASH`	If not using host firmware update utility then comment out to save code space
`M2M_DISABLE_FIRMWARE_LOG`	If you want to see debug output from the WINC1500 module then comment out this define. To save power and increase WINC1500 efficiency then leave this define in place (not commented out). See *How to Debug WINC1500 Wi-Fi Firmware*.

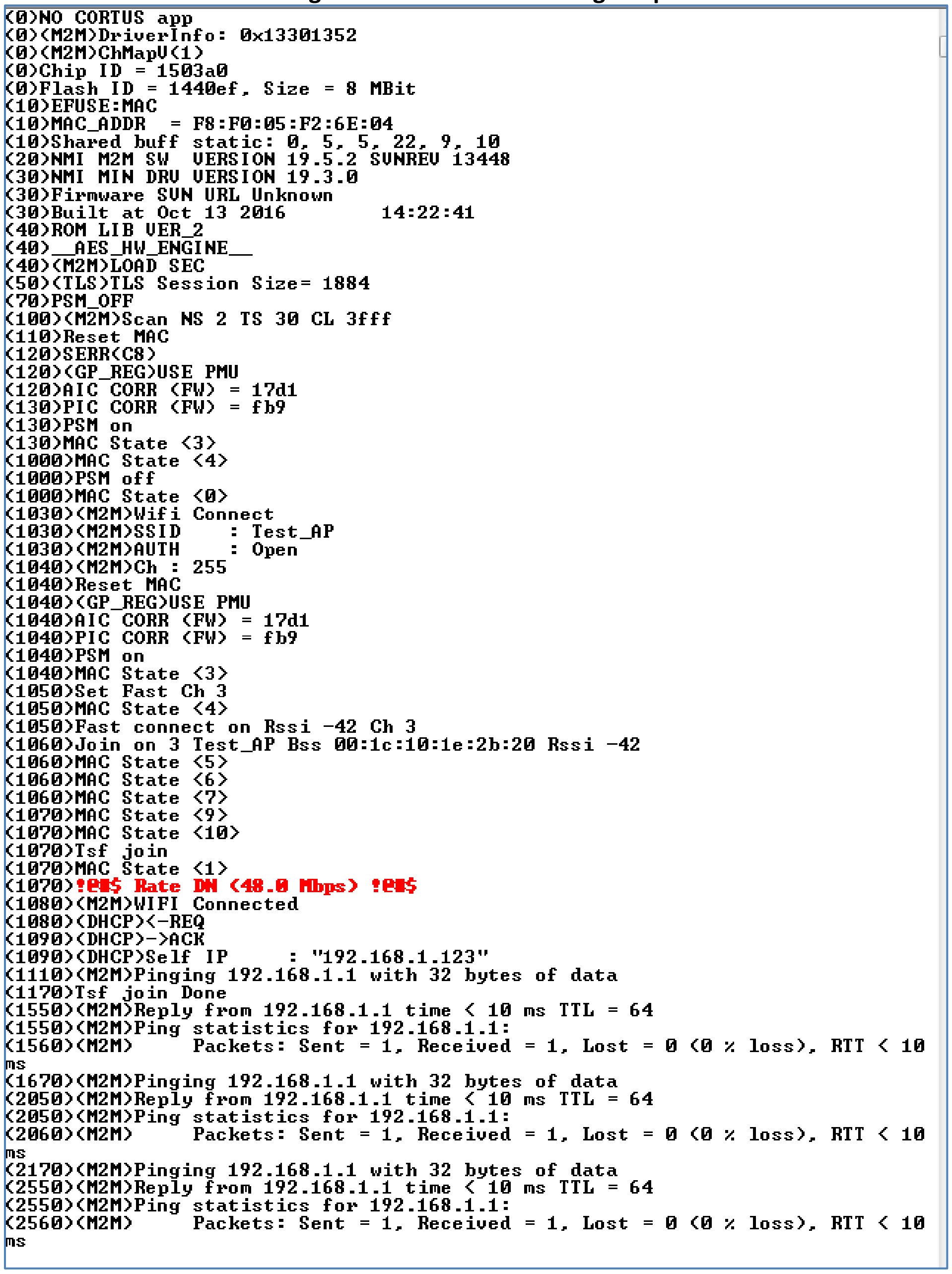
7 How to Debug WINC1500 Wi-Fi Firmware

The WINC1500 module provides a USB UART interface for debugging. You can connect the WINC1500 module to your PC using a USB connector to get firmware debug information. The UART terminal window should be configured to 115200 baud, 8 data bits, 1 stop bit, no parity, no flow control. Connect the module USB cable after the module is powered up; then bring up a terminal program.

Notes: In order to get debug output, the `M2M_DISABLE_FIRMWARE_LOG` define in winc1500_driver_config.h must be commented out. Also, be aware that the debug output from the module consumes time and will affect performance. It should not normally be enabled.

Below is the debug output when running the AP scan demo.

Figure 7-1: WINC1500 Debug Output



```
(0)NO CORTUS app
(0)(M2M)DriverInfo: 0x13301352
(0)(M2M)ChMapV(1)
(0)Chip ID = 1503a0
(0)Flash ID = 1440ef, Size = 8 MBit
(10)EFUSE:MAC
(10)MAC_ADDR  = F8:F0:05:F2:6E:04
(10)Shared buff static: 0, 5, 5, 22, 9, 10
(20)NMI M2M SW  VERSION 19.5.2 SVNREV 13448
(30)NMI MIN DRV VERSION 19.3.0
(30)Firmware SVN URL Unknown
(30)Built at Oct 13 2016        14:22:41
(40)ROM LIB VER_2
(40)__AES_HW_ENGINE__
(40)(M2M)LOAD SEC
(50)(TLS)TLS Session Size= 1884
(70)PSM_OFF
(100)(M2M)Scan NS 2 TS 30 CL 3fff
(110)Reset MAC
(120)SERR(C8)
(120)(GP_REG)USE PMU
(120)AIC CORR (FW) = 17d1
(130)PIC CORR (FW) = fb9
(130)PSM on
(130)MAC State <3>
(1000)MAC State <4>
(1000)PSM off
(1000)MAC State <0>
(1030)(M2M)Wifi Connect
(1030)(M2M)SSID     : Test_AP
(1030)(M2M)AUTH     : Open
(1040)(M2M)Ch : 255
(1040)Reset MAC
(1040)(GP_REG)USE PMU
(1040)AIC CORR (FW) = 17d1
(1040)PIC CORR (FW) = fb9
(1040)PSM on
(1040)MAC State <3>
(1050)Set Fast Ch 3
(1050)MAC State <4>
(1050)Fast connect on Rssi -42 Ch 3
(1060)Join on 3 Test_AP Bss 00:1c:10:1e:2b:20 Rssi -42
(1060)MAC State <5>
(1060)MAC State <6>
(1060)MAC State <7>
(1070)MAC State <9>
(1070)MAC State <10>
(1070)Tsf join
(1070)MAC State <1>
(1070)!@#$ Rate DN (48.0 Mbps) !@#$
(1080)(M2M)WIFI Connected
(1080)(DHCP)<-REQ
(1090)(DHCP)->ACK
(1090)(DHCP)Self IP      : "192.168.1.123"
(1110)(M2M)Pinging 192.168.1.1 with 32 bytes of data
(1170)Tsf join Done
(1550)(M2M)Reply from 192.168.1.1 time < 10 ms TTL = 64
(1550)(M2M)Ping statistics for 192.168.1.1:
(1560)(M2M)      Packets: Sent = 1, Received = 1, Lost = 0 (0 % loss), RTT < 10
ms
(1670)(M2M)Pinging 192.168.1.1 with 32 bytes of data
(2050)(M2M)Reply from 192.168.1.1 time < 10 ms TTL = 64
(2050)(M2M)Ping statistics for 192.168.1.1:
(2060)(M2M)      Packets: Sent = 1, Received = 1, Lost = 0 (0 % loss), RTT < 10
ms
(2170)(M2M)Pinging 192.168.1.1 with 32 bytes of data
(2550)(M2M)Reply from 192.168.1.1 time < 10 ms TTL = 64
(2550)(M2M)Ping statistics for 192.168.1.1:
(2560)(M2M)      Packets: Sent = 1, Received = 1, Lost = 0 (0 % loss), RTT < 10
ms
```

8 How to Download New WINC1500 Firmware

8.1 Download Firmware

This section discusses how to upgrade the firmware revision of the WINC1500 PICtail (or Click) module. This process updates the default certificate files as well. The upgrade process requires the following setup:

1) WINC1500 PICtail plugged into either an Explorer 16, Explorer 16/32, or Explorer 8 board

2) The host firmware must be running the winc1500_fw_update_over_serial demo (selected by uncommenting `#define USING_FW_UPDATE_OVER_SERIAL` in demo_config.h)

3) There must be a UART connection between the board and a Windows PC. If using the Explorer 16 this will be a DB9 cable. If using the Explorer 16/32 or Explorer 8 board this will be a USB cable. Refer to *Hardware Prerequisites* (Section 2.1) and *Getting Started with WINC1500 PICtail* (Section 4). Do not open a terminal program on the PC – the PC firmw are update utility communicates directly with the board. The update utility automatically scans for the UART port connected to the board.

4) The firmware upgrade utility must be installed on the PC. This utility is provided by Microchip when there is a new release.

5) After the above 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 Explorer, or by running the batch file from within a Cmd window. The output should be similar to what is shown in Section 10.1.

8.2 Download Certificate File

As mention above, the firmware upgrade also updates the certificate files. The certificate files that are written to the WINC1500 can be customized. There are two ways to do this. The following discussion presumes the download utility has been installed at `C:\winc1500`.

8.2.1 Method 1

In Method 1, you simply add any desired certificate files to the \binary directory and perform the steps described in Section 8.1. All the current certificate files in the FLASH will be erased, and the cert files in the \binary directory will automatically be written to WINC1500 FLASH. Presuming the download utility was installed at C:\winc1500, the \binary directory is at:

```
C:\winc1500\Tools\root_certificate_downloader\binary
```

8.2.2 Method 2

Method 2 allows you to update only the certificate files and not alter any other images on the WINC1500 FLASH. Perform the following steps:

1) As in Method 1, copy any desired certificate files to the \binary directory. For this description, presume there are a total of four certificate files to write to WINC1500 FLASH – cert1.cer, cert2.cer, cert3.cer, and cert4.cer. These four files should be copied to the \binary directory as described in Method 1.

2) Open a Cmd window and change into the **`C:\winc1500\Tools\root_certificate_downloader\Debug_UART`** directory.
3) Run `root_certificate_downloader.exe`. Two examples are shown below. In the first example the four certificates are added to the existing certificates already in the FLASH.

Figure 8-1: Retain Existing Certificates and Add Additional Certificates

```
root_certificate_downloader.exe -n 4 ..\binary\cert1.cer ..\binary\cert2.cer
..\binary\cert3.cer ..\binary\cert4.cer -no_wait -port 0
```

Figure 8-2: Erase Existing Certificates and then Add Certificates

```
root_certificate_downloader.exe -n 4 ..\binary\cert1.cer ..\binary\cert2.cer
..\binary\cert3.cer ..\binary\cert4.cer -no_wait -port 0 -e
```

Notes:

1) Change the `-n` parameter to match the number of files. In the above examples there are four files, hence `-n 4`. The desired files follow the `-n` parameter
2) You can add a `-e` parameter at the end of the command line; if present, all certificates in the FLASH are first erased, and then the certificates in the command line are written. If the `-e` is absent then the certificates in the command line are added to those already in the FLASH.

9 Conclusion

This document explained the essential elements of using the WINC1500 PICtail on three different Microchip development boards and how to run one of the demos. The following topics have been covered:

- How to get and install MPLAB X IDE, the toolchain for PIC18 and PIC24, and the MLA framework
- Target board information
- How to run the Wi-Fi AP Scan demo
- How to find and modify various configuration parameters
- How to update firmware and certificates
- How to get debug output from the WINC1500 module

10 Appendix

10.1 Example Output from Firmware Update Utility

Figure 10-1: Example Output from Firmware Update Utility

```
Downloading Image...
*******************************************
*  >Programmer for WINC1500 SPI Flash<     *
*      Owner:  Atmel Corporation           *
*******************************************
SVN REV 14274 SVN BR branches/WIFIIOT-1660_19_5_2_RC7
Built at Jan 27 2017    14:24:14
Firmware Path (3A0) ../../../firmware/m2m_aio_3a0.bin
>>Initialize programmer.
Detecting ports...
(APP)(INFO)WINC1500 Serial Bridge Found
Avail port COM51
1 of ports found
Chip id 1503a0
>Waiting for chip permission...
OK.

----- NOW Programming Firmware Image Version -----
Firmware ver   : 19.5.2 Svnrev 14274
Min driver ver : 19.3.0
Firmware Build Jan 26 2017 Time 22:13:34

----- Previous Firmware Image Version -----
Firmware ver   : 19.5.2 Svnrev 14274
Min driver ver : 19.3.0
Firmware Build Jan 26 2017 Time 22:13:34

Flash ID 1440ef
(APP)(INFO)Flash Size 8 Mb
>Start erasing...
Done
#Erase time = 0.890000 sec
>Start programming...
Done
#Programming time = 105.238000 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:f2:66:80
(APP)(INFO)(Efuse)Valid = 0,PATxGainCorr = 00
(APP)(INFO)(Efuse)Valid = 1,FreqOffset = 010c
(APP)(INFO)------------- END EFUSE DUMP ----------------
Creating look up table for PLL with xo_offset = 4.1875.
>Start erasing...
Done
#Erase time = 0.047000 sec
>Start programming...
Done
#Programming time = 1.061000 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:f2:66:80
```

```
(APP)(INFO)(Efuse)Valid = 0,PATxGainCorr = 00
(APP)(INFO)(Efuse)Valid = 1,FreqOffset = 010c
(APP)(INFO)------------- END EFUSE DUMP ----------------

>>This task finished after 108.00 sec
Downloading root certificates...
**************************************************
* > WINC1500 Root Certificate Flash Downloader < *
**************************************************
SVN REV 14274 SVN BR branches/WIFIIOT-1660_19_5_2_RC7
Built at Jan 27 2017    14:24:16
Detecting ports...
(APP)(INFO)WINC1500 Serial Bridge Found
Avail port COM51
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 . . .
```