# Marvell
# Wireless Microcontroller

## Software Development Kit 3.2

### Quick Start Guide

Not Approved by Document Control.

August 24, 2015

CONFIDENTIAL

Document Classification: Proprietary Information

## Document Conventions

| | |
|---|---|
|  | **Note:** Provides related information or information of special importance. |
|   | **Caution:** Indicates potential damage to hardware or software, or loss of data. |
|  | **Warning:** Indicates a risk of personal injury. |

For more information, visit our website at: www.marvell.com

| Date | Revision | Changelog |
|---|---|---|
| Aug 19 2015 | SDKv3.2 | Update for release 3.2 |
| Apr 7 2015 | SDKv3.1 | Update for release 3.1, Rewrite the document flow |
| Feb 11 2015 | SDKv3.0 | Update for release 3.0 |
| Dec 14 2014 | SDKv3.0 R1 | • Update for release 3.0 Preview Release 1 |
| Oct 1 2014 | SDKv3.0 Alpha 2 | • Update most sections.<br>• RD board v2.0 is used instead of v1.0<br>• In addition to Cygwin, experimental MinGW support is now added for Windows platforms. |
| Jun 23 2014 | SDKv3.0 Alpha | • Update for release 3.0-Alpha Release |
| Jan 30 2014 | SDKv2.13 | • Update for release 2.13 |
| Dec 2 2013 | SDKv2.12 | • Update for release 2.12 |
| Aug 26 2013 | SDKv2.11 | • Update for release 2.11 |
| Apr 20 2013 | SDKv2.10 | • Update for release 2.10 |
| Feb 2 2013 | SDKv2.9 | • Update for release 2.9 |
| Nov 23 2012 | SDKv2.8 | • Update for release 2.8 |
| Oct 23 2012 | SDKv2.7 | • Update for release 2.7 |
| Sep 13 2012 | SDKv2.6 | • Update for release 2.6 |
| Jul 27 2012 | SDKv2.5 | • Update for release 2.5 |
| Jun 15 2012 | SDKv2.4 | • Update for release 2.4 |
| May 7 2012 | SDKv2.3 | • Update for release 2.3 |
| Apr 2 2012 | SDKv2.2 | • Update for release 2.2 |
| Mar 9 2012 | SDKv2.1 v4 | • Cleanups |
| Mar 6 2012 | SDKv2.1 Draft 3 | • Include details about the power manager |
| Feb 17 2012 | SDKv2.0 Draft 2 | • Remove 8688 support |
| Jan 12 2012 | SDKv2.0 Draft 1 | • Update for SDK 2.0 |
| Oct 13 2011 | SDKv1.3 Draft 2 | • Documentation fixes |
| Mar 17 2011 | SDKv1.2 Draft 1 | • First Draft |

# Table of Contents

# 1 Overview

Marvell's Wireless Microcontroller Software Development Kit (WMSDK) enables the development of custom firmware images that can be run on the processor built-into Marvell's 88MW300 Single Chip WLAN Microcontroller as well as the 88MC200 System-on-Chip (SoC).

This software/hardware platform can be used to realize complex applications that can run on low-cost and low-power consuming hardware.

The SDK and the corresponding documentation all the products phases from hardware design, software development to all the manufacturing and certification related requirements.

## 1.1 About This Document

This document is organized as follows:

1. **Overview**: The first few sections of the document provide an overview of the WMSDK.
    a. **Hardware**: Section 1.2 provides an overview of Marvell's hardware platforms that are supported by this SDK
    b. **Software**: Section 1.3 describes the various functionalities from the ground-up that are provided by this SDK
    c. **About This Release**: Section 2 talks about the new features that are part of this release
2. **Trying out**: Now that you have an overview of the WMSDK, let's get our hands on a board to try out the various features. Section 3 allows you to exercise these features using a development board that you received from Marvell
3. **Developing with the SDK**: Now it's time to start development with the SDK
    a. **Development Host Setup**: Please refer to the *Development Host Setup* document for instructions on setting up your development host.
    b. Developing Firmware: Section 4 provides the basic steps that are required for any development activities with the WMSDK

Please also refer to additional documentation on extranet for WMSDK features, manufacturing and certification.

## 1.2 Marvell's Hardware Platforms

The WMSDK supports two Marvell SoCs:

1. **88MC200**: This is Marvell's 32-bit microcontroller. This can be used in conjunction with Marvell's Wi-Fi/BLE chipsets like 88W8801/88W8782/88W8787/88W8777.
2. **88MW300**: This is a single-chip microcontroller with Microcontroller and Wi-Fi as part of the same SoC.

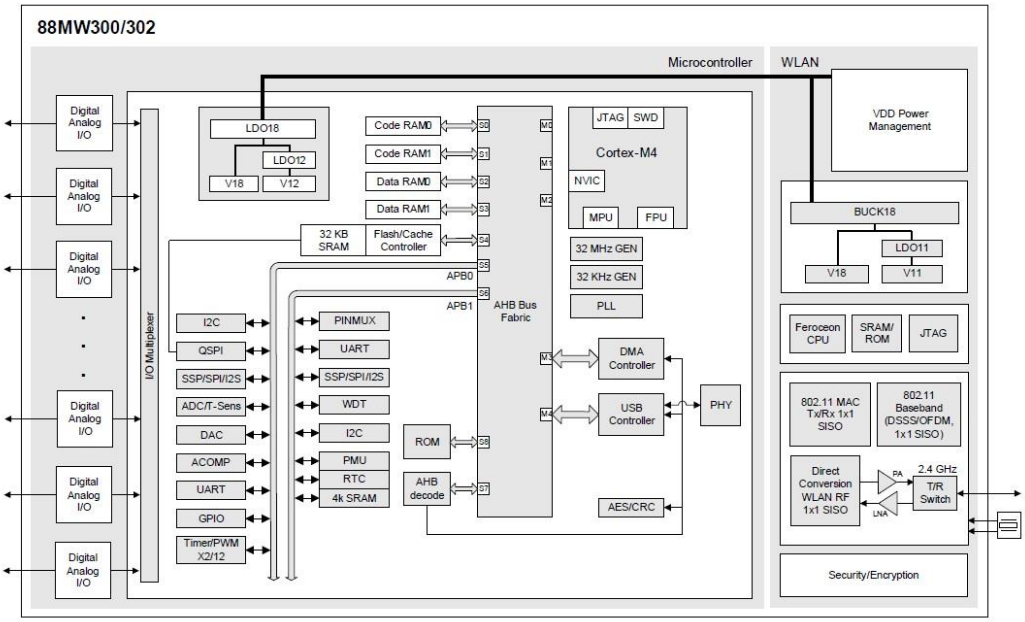
### 1.2.1 Single Chip Wireless Microcontroller (88MW300)

Marvell's 88MW300 is a low-cost, low-power, highly integrated SoC with an ARM Cortex-M4F based microcontroller and 88W8801 based Wi-Fi on chip to provide the Wireless Microcontroller single chip solution

Key Features on top of the 88MC200 are:

- Integrated on-chip Wi-Fi
- ARM Cortex-M4F Core with Floating Point and DSP instructions supported
- eXecute in Place (XiP) support with 32KB of flash controller cache
- 128KB of ROM
- USB OTG 2.0 High/Full Speed Support
- Option of 68/88 pin package

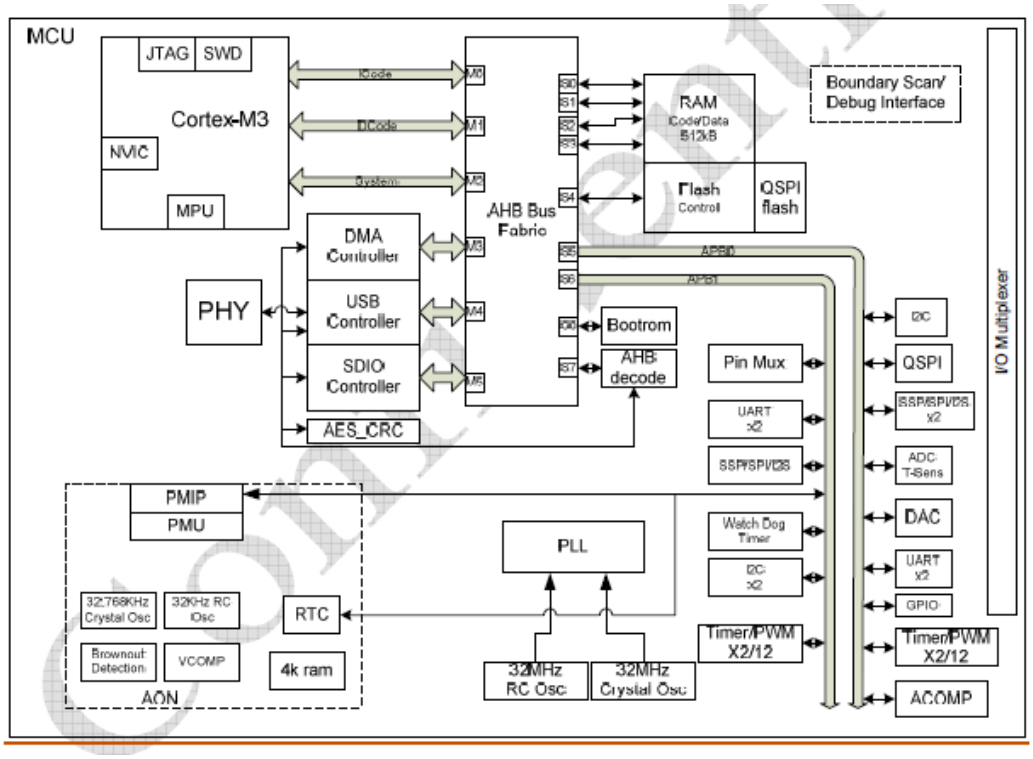
**Figure 1: 88MW300 Block Diagram**




## 1.2.2 Wireless Microcontroller (88MC200)

Marvell's 88MC200 is a low-cost, low-power, highly integrated SoC that works in conjunction with the Marvell 88W8782 / 88W8787 /W8801 SD Wi-Fi chip to provide the Wireless Microcontroller solution.

**Figure 2: 88MC200 Block Diagram**




## 1.2.3 Supported Development Boards

Marvell has created multiple boards that can be used for development. The WMSDK supports the following development boards:

1. 88MW30x Reference Design (RD) Board (AB-88MW30X V2.0)
    a. This is the 88MW30X Reference Design
2. 88MC200 + SD8801 Development Board v2 (RD-88MC200-W8801-V2)
    a. This board consists a 88MC200 + 88W8801 Wi-Fi module
3. Lark-mini Development Board v3 (DB-88MC200-A1-LK20-01)
    a. This board consists of a 88MC200 + sd8787 Wi-Fi module
4. 88MC200 Development Board v2 (DB-88MC200-A0-DEV-02)
    a. This board consists only of a 88MC200. It provides an SD slot. Any SD Wi-Fi chipset can be used with this development board.

The WMSDK may work on legacy boards, but that has not been tested. If you have a legacy board, please work with your Marvell representative to procure the supported boards.

You can identify the board that you have, by looking at the following identification strings:

**Table 1: Board Identification Table**

| Board Name | Identification String | Location on the Board |
|---|---|---|

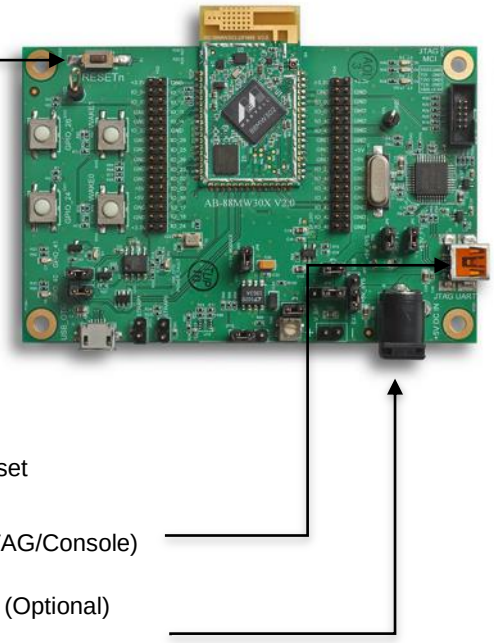
| Board Name | Identification String | Location on the Board |
|---|---|---|
| *88MW30x RD Board* | AB-88MW30X V2.0 | Next to the module on the board |
| *88MC200 + SD8801 Development Board v2* | AB-88MC200-W8801-V1 | Bottom plate of the board |
| *Lark-mini Development Board v3* | DB-88MC200-A1-LK20-01 | Next to the Marvell logo |
| *88MC200 Development Board v2* | 88MC200 Development Board V2.0 | Below the LCD screen in the right corner |

## 1.2.4 Board Setup

1. All boards can be powered on using a USB to mini-USB cable.
    a. The same cable provides console access to the board. A virtual tty/com port is registered with the Development Host that can be used to access the console.
    b. The same cable also provides JTAG access to the board. This can be used for loading/unloading firmware images into the RAM or flash of the board, or for debugging purposes.

**Figure 4: 88MC200 Development Board**

**Figure 4: 88MW302 Development Board**




## 1.2.5 Supported Modules

The WMSDK supports the following modules from module vendors:

- Azurewave AW-CU288, AW-CU277, AW-CU282
- Hi-Flying HF-LPB200

Appropriate build configuration files have been provided with the SDK to ensure that the builds can be generated for the development boards of your choice.Marvell's Software Development Kit
The WMSDK includes a rich set of production-ready software components that enable rapid development of functionality rich, autonomous, IP-networked, microcontroller applications. It also includes a complete set of tools to support development.

# 1.3 Core Run-Time Software

At its core, the WMSDK software combines the 'wireless' and 'microcontroller' feature sets.

### 1.3.1.1 Microcontroller

The WMSDK can be used to build firmware (or applications from the perspective of the WMSDK) images that execute on the microcontroller. The firmware image is stored on the flash of the microcontroller (or externally connected flash). On boot-up, the boot-loader loads the firmware and this begins execution. At its core the following features are supported on the microcontroller:

- **OS:** The FreeRTOS operating system. The OS provides support for OS-level primitives such as scheduling and thread management, semaphores and mutexes, message queues and timers. All the APIs of the Operating System are exposed through an OS Abstraction Layer, facilitating ease of using another OS of the customer's choice.
- **Networking Stack:** The lwIP network stack that provides TCP/IP (IPv4 and IPv6) networking with BSD Socket level API
- **Drivers:** Drivers for the various interfaces of the microcontroller like UART, I2C, SPI, USB Host, USB Client, ACOMP, ADC/DAC, GPT, Flash, watchdog and SDIO
- **Power Manager Framework:** The microcontroller has various levels of power conservation. The microcontroller is executing when it is in PM0 mode. The microcontroller can be in either of the Power Save levels, PM1, PM2, PM3 and PM4 based on the functionality that has to be retained. The Power Manager Framework allows applications to make the best use of these power-save levels by automatically detecting idle times (*tickless idle* kernel) and managing the power save entry-exit in these idle durations. Applications can control and tune this framework as per their use-cases based using the APIs that are available.

  All the threads and timers in the WMSDK are optimized such that the microcontroller can stay in the low power modes as long as possible without affecting the functionality.
- **Footprint:** Various configuration options are available for extracting maximum utilization of the RAM and flash footprints.
  - **eXecute in Place (XiP) (MW300 only):** The code can be executed directly from QSPI Flash. A cache is provided to increase execution speed. Note that some components like interrupt service routines, flash access function, etc. need to be in RAM. The linker script and boot-loader together take care of putting the appropriate components in RAM during bootup. Application developers do not have to worry about it.
  - **Overlays:** This feature allows application to selectively load sections of code that are independent and mutually exclusive in the same section of RAM. This provides better RAM utilization by not having to load any content that is not immediately required. This feature is generally useful only when XIP cannot be used.
  - **Compression:** The Wi-Fi firmware can be stored in the flash in a compressed format. The 'xz' compression algorithm is supported which provides maximum compression. This provides better flash utilization.

- **Footprint-conscious architecture:** The modules of the WMSDK are so structured that only the functions and data structures that are actually used are pulled in to create the firmware image.
- **Tools:** The WMSDK provides some tools for analyzing footprint usage.
- **Debug/Release Configurations:** The WMSDK provides pre-generated configuration options for creating debug and release images. The release images are much smaller in size because it excludes all the debug messages and other developer-friendly functionality.

#### 1.3.1.2 Wi-Fi

The WMSDK supports Marvell Wi-Fi chipsets, 88MW300, 88W8782, 88W8787 and W8801. The wireless driver within the WMSDK works in conjunction with the wireless firmware for these chipsets to provide the Wi-Fi functionality. The Wi-Fi firmware is stored on the flash of the development kit and is loaded into Wi-Fi chipset during bootup.

The Wi-Fi firmware binaries are available in the *wifi-firmware/* directory of the wmsdk bundle, which are flashed using the flashprog utility.

The WMSDK Wi-Fi driver supports the following features:

- **Wi-Fi Client/Station:** In this mode, the Wi-Fi chips acts as a *station* and connects to other *Access Points* in its vicinity. In the *station* mode, following features are supported:
  - 802.11 b/g
  - 802.11 n
    - 5 GHz band support (on select Wi-Fi chipsets)
    - Receive support for A-MPDU, A-MSDU
  - Security
    - Open and shared-key authentication
    - WEP Data encryption
    - WPA-PSK and WPA2-PSK
    - WPA2 Enterprise with EAP-TLS
  - WPS – Wi-Fi Protected Setup (PIN and PBC methods)
  - 802.11d (Additional Regulatory Domains)
- **Access Point:** In this mode, the Wi-Fi chip acts as an Access Point (named as micro-AP mode), so that other Wi-Fi clients/stations can connect to it. In the micro-AP mode, following features are supported:
  - 802.11b/g
  - Security:
    - No Security
    - WPA2-PSK
  - 802.11d (Additional Regulatory Domains) and Region Configuration
- **Wi-Fi P2P:** In this mode, the Wi-Fi chip uses the Wi-Fi-Direct/P2P feature to establish a peer-to-peer link with other Wi-Fi-Direct/P2P enabled devices.
- **Wi-Fi Connection Manager:** The Wi-Fi Connection Manager enables applications to easily connect to existing Wi-Fi networks, or start their own Wi-Fi network (using micro-AP) without requiring detailed 802.11 protocol knowledge.

- **Wi-Fi Power Management:** Support for three different Wi-Fi power-save states: IEEEPS, DeepSleep and PDN (Power Down) is available. While being used as a Wi-Fi station, the Power Manager Framework can also intelligently detect and put the Wi-Fi chipset in IEEEPS modes whenever it identifies idle durations.
- **Wake-on-WLAN/Host Wakeup:** The power management modes of both the microcontroller and the Wi-Fi chipset/section operate seamlessly in conjunction with each other. Both the host-microcontroller/section and the Wi-Fi chipset/section can simultaneously enter Power Save modes and they can be configured in such a way that the Wi-Fi chip/section wakes up the host microcontroller/section when certain kinds of packets are received from the network This allows the host to remain in power down mode for extended times.

The core run-time software components together with the integrated driver support enable developers to focus on application-specific functionality. This relieves the developers from focusing on low-level programming for the SoC components, developing or integrating operating system software, TCP/IP stack, or Wi-Fi features.

#### 1.3.1.3 BLE (Bluetooth Low Energy)

The 88W8777 chipset provides both Wi-Fi and BLE functionality. The WMSDK provides the BLE stack that is required to operate the chip.

The Wi-Fi firmware that is used for the 8777 chipset provides firmware functionality for both the Wi-Fi as well as BLE connectivity. The Wi-Fi firmware binaries are available in the *wifi-firmware/* directory of the wmsdk bundle.

The WMSDK BLE stack supports the following features:

1. **Co-Existence with Wi-Fi**: Both Wi-Fi and BLE on the chipset can simultaneously operate without interfering with each other.
2. **GAP Roles**:
    a. Central
    b. Peripheral
    c. Observer
    d. Broadcaster
3. **GATT Roles**: Client and Server
4. L2CAP dynamic channel
5. **Security Manager Protocol (SMP)**: Legacy and Secure Connection

The BLE stack is provided with multiple feature packs. This allows you to choose the appropriate feature pack for your usecase. The BLE stack can be built as follows:

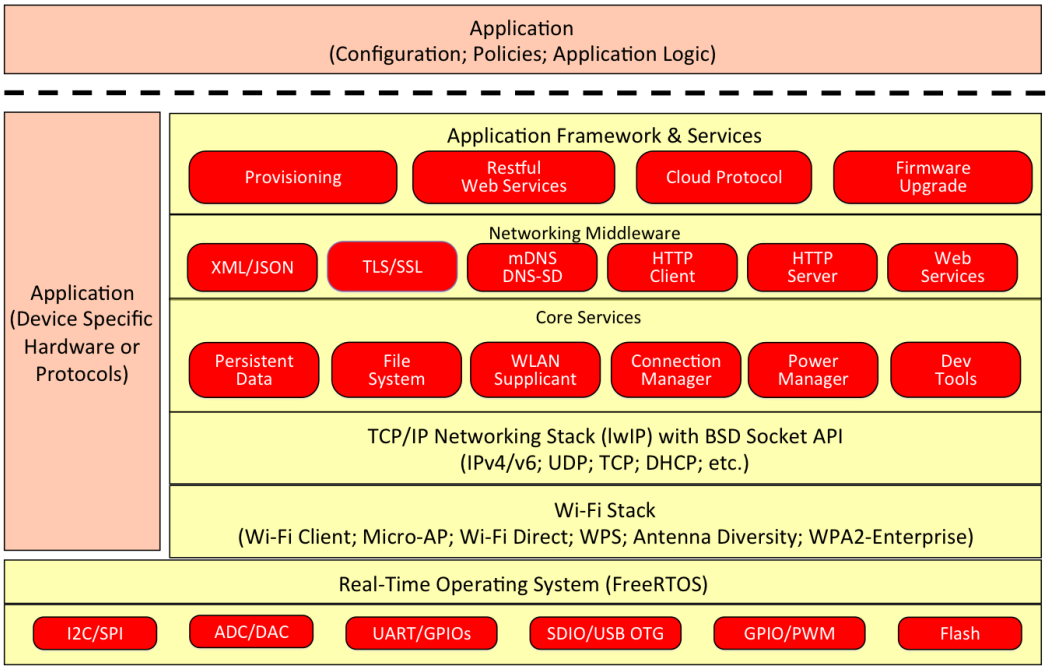
```
$ cd wmsdk_bundle-x.y.z
```

```
$ make EXT_BLESTACK_PATH=/path/to/mrvl_lestack-
w.x.y.z/mrvl_lestack_feature_packs/mc200/ble-fpX/mrvl_lestack/
```

### 1.3.2 SDK Features

The SDK also provides a rich set of middleware components to enable development of connected applications that integrate seamlessly with the Wi-Fi installations, Internet and Web infrastructure. The following is the Software Block Diagram of the WMSDK.

## Figure 5: WMSDK Software Block Diagram




### 1.3.2.1 Provisioning

When the WMSDK device acts as a Wi-Fi station interface, it connects to a Wi-Fi network in the vicinity. In case of devices targeting the home users, this could be the user's home Wi-Fi network. Provisioning is the process performed by the end user. At the end of this process the WMSDK based devices are configured with the end-customer's Wi-Fi network settings. The WMSDK supports all the following network provisioning methods.

1. **Marvell Provisioning:** In this provisioning mechanism the WMSDK devices initially boot up in micro AP mode.. Customers then connect to this micro-AP and access the web UI served by the on device HTTP server. This webUI guides the customers through the simple process of configuring WMSDK device for the target Wi-Fi network. As an alternative to the webUI, the provisioning can also be done using native Android/iPhone apps. Fully functional reference applications for Android and iOS are shipped along with the WMSDK bundle.
2. **EZConnect Provisioning:** In this provisioning mechanism, a native smart phone app transparently interacts with the WMSDK device. The app broadcasts the credentials of the target Wi-Fi network in a secure and encrypted manner. The WMSDK device sniffs these packets from the air and extracts the network configuration information. Fully functional reference applications for Android and iOS are shipped along with the WMSDK bundle.
3. **WAC (Wireless Accessory Configuration):** This is Apple's standard for performing the network provisioning of the devices. Note that this requires Apple's MFi authentication chip to be interfaced with the microcontroller. Support for WAC is in-built in OS-X and iOS. Thus, as against the other provisioning mechanisms, no other phone/desktop application need be installed for performing the WAC provisioning. Note that this support is currently only available in the HAP SDK.
4. **WPS (Wi-Fi Protected Setup):** This is the easiest provisioning mechanism and published as a standard by the Wi-Fi Alliance. In its simplest form, this includes the customer pressing a button on their home AP, and another button on the WMSDK device.

The device and the AP then perform the negotiation, and follow a defined process. The end result of the process is that the device gets the configuration of the target Wi-Fi network. While this method is the easiest, not all APs may support this mechanism because of some known security issues. This method can be used along with the *Marvell Provisioning* method defined below to address this problem.

#### 1.3.2.2 HTTP/HTTPS Web Server

The WMSDK includes a HTTP Web Server. This server has been designed to work with all the commonly available browsers on most platforms. The web server has the following features:

1. **Serve Files:** The webserver can serve HTML/JS files to connected clients. These files can be stored on the flash of the device. Application developers can create web apps that can provide the end-users with meaningful workflows. Such web applications are useful since they allow the end-user to start interact with the device without having them to install any application on their systems. A collection of files from the given folder is bundled up to create an FTFS image. This FTFS image is then flashed along with the MCU firmware. Note that the FTFS is read-only file system. No files can be added/updated.
2. **Caching Support:** WMSDK web server includes partial support of RFC 7234 standard for HTTP caching. This enables developers to use this feature to speed up page loading times for the end user.
3. **Web Server Gateway Interface:** The applications can write custom WSGI handlers associated with specific URIs. Applications register with the web server to get a callback for specific paths and HTTP methods GET, POST, PUT and DELETE. This allows applications to expose rich functionality through web APIs. This can be used to create RESTful web services for e.g.

   The WSGI handlers are provided with the entire environment in which the HTTP request is received along with any data that was part of the HTTP request. Utility functions are available to be used by WSGI handlers for quick development of data processing functionality, be it in the form or URL-encoded parameters, HTTP form data or POST data in the body of the request. Functions are also available for the WSGI handlers to generate the different HTTP response codes, encodings and data formats.
4. **HTTP Secure (https):** With appropriate configuration, the HTTP server also can serve content on a TLS connection i.e. https over port 443.
5. **HTTP Persistent Connections and Pipelining:** Multiple simultaneous open connections are supported and appropriate context is provided for each open socket connection,

#### 1.3.2.3 Filesystem (FTFS)

The Flash Table File System (FTFS) is a read-only flash filesystem used to store static content to be served by the web server. Other applications can also use FTFS to store permanent configuration data The FTFS exposes standard file handling APIs viz. fopen, fread, ftell, fclose, etc.

#### 1.3.2.4 Secure Boot

The 88MW300's BootROM can load images that are encrypted (AES-CCM) and/or signed (RSA). The SDK support a trusted chain of booting such that:

a. The BootROM can decrypt and/or verify the boot2 boot-loader (AES-CCM-256 / RSA-2048)
b. The boot-loader can decrypt and/or verify the microcontroller firmware (AES-CTR-128 / RSA-2048)
c. The Microcontroller firmware can store encrypted values into the PSM (AES-CTR-128)
d. Additionally, support is also available for locking out JTAG and disabling other boot options

#### 1.3.2.5 HTTP/HTTPS Web Client

The HTTP web client module allows applications to perform HTTP operations like GET, POST, etc. on any webserver. Following are the list of features in this module:

- HTTPS
- Transparent generation of HTTP request headers and parsing the response.
- Support partial support of Keep Alive mechanism allowing persistent connections.
- Interleaved operation for faster data transfer in cases of persistent connections.
- Transparent handling of chunked encoded data for GET operations.
- Chunked write

#### 1.3.2.6 WebSockets

The HTTP Web Client module, mentioned above, enables a device to communicate with cloud servers. The HTTP protocol being is a request-response protocol, may not be suitable for certain kind of applications that require instantaneous notifications on either end. The WMSDK also has implementation of the WebSockets protocol. The WebSockets protocol enables full-duplex communication of data frames over the HTTP protocol.

The sample applications also include a sample cloud server that is implemented using Web Sockets.

#### 1.3.2.7 Bonjour/multicast Domain Name System (mDNS)

The WMSDK supports service discovery and announcement using mDNS. The mDNS module can be used as a responder, or a querier or both. Bonjour being an implementation of Zeroconf network, this is very useful to setup a network with minimal configuration. Dynamic ecosystems of heterogeneous devices can be built using this platform.

- **Responder:** As a responder, this module can be used to announce services that this WMSDK device offers over the network. Other entities on the network can discover who offers certain services. The responder also responds to hostname lookup requests. This enables end-users to access devices using hostnames instead of IP addresses.
- **Querier:** As a querier, this module can be used to lookup services on the network that the WMSDK device is interested in. Once services are discovered these services can be accessed to build an ecosystem of multiple devices working together.

#### 1.3.2.8 Over The Air (OTA) Upgrades

The WMSDK includes field-validated support for firmware and filesystem (FTFS) upgrades. The upgrades can be performed over the local network (HTTP POST) or remotely through the cloud (HTTP GET). Using a fail-safe active-passive partition mechanism ensures that the WMSDK devices continue to stay up and functional even across failure events such as network loss or others.

The boot2 boot-loader ensures that it always boots into the latest upgraded firmware on the system. In case it detects any errors with this firmware, it falls back to boot from the older firmware partition.

Similarly, the MCU firmware uses the latest and the compatible filesystem and Wi-Fi firmware images from flash.

### 1.3.2.9 Persistent Storage Manager (PSM)

The PSM module provides a way for applications to read or store persistently name-value pairs on flash. The persistent information could be configuration related, like the target Wi-Fi network's settings, or any other state of the WMSDK device. Following are the features:

- One-level deep namespace for variable names.
- Binary values are supported.
- Specially designed to be run over flashes. It uses a special algorithm to avoid excessive erasing of flash sectors thus significantly prolonging the usable life of the flash device.
- Power resiliency. It uses reserved sectors to ensure that ever if power fails during write operation, existing variables are safe from corruption. This handling is transparent to user of PSM.
- Read only mode of operation. This allows user to store key value pairs permanently. This is useful to store factory data that is preserved across reset-to-factory scenarios.
- Multiple instances of PSM module can be run simultaneously to allow multiple separate PSM storage areas on the same device.

To provide the power resiliency feature, the PSM reserves some flash sectors for recovery. Thus, PSM requires at least two (2) sectors to work unless run in read-only mode. The number of reserved sectors among the given sectors is managed automatically.

Note that the above features are part of newer version of PSM (PSMv2). The older version of PSM, which was part of earlier releases has been replaced with PSMv2. The API compatibility layer is present to map from older API's to newer API's. The full features of PSMv2 can be used only if newer API's are used.

The older PSM can be manually enabled if necessary for any reason. It is not recommended as it has high probability of failing in cases of power failures.

### 1.3.2.10 DHCP Server

When the WMSDK device hosts its own network using micro-AP, the DHCP server is responsible for dynamically assigning IP Addresses to the Wi-Fi clients that connect to this micro-AP.

Features:

- DHCP dynamic allocation.
- DHCP caches mappings and preferentially assigns same IP address a client previously had during renewal.
- No persistent storage of mappings. Mapping cache is refreshed after reboot of device.

### 1.3.2.11 JSON / XML

The WMSDK also includes modules for parsing as well as generating data in the JSON (JavaScript Object Notation) or XML (Extensible Markup Language) formats.

### 1.3.2.12 Transport Layer Security (TLS & SSL)

WMSDK supports Transport Layer Security (TLS & SSL) by integrating the CyaSSL library. This enables support for all of the SSL/TLS family protocols, viz, SSLv2, SSLv3, TLSv1.0 TLSv1.1 and TLSv1.2. Following are some of the features present:

- Key Exchanges: RSA, DH, EDH and NTRU
- Data Encryption/Decryption: DES, 3DES, AES, ARC4, RABBIT, HC-128
- Message Authentication Code: MD-2/4/5, SHA-1/2/256/512, RIPEMD-169

Additionally, X.509 v3 certificate support is included. The support for all of the above is available in client as well as server mode.

Note that WMSDK release comes along with binary only commercial release of CyaSSL. Libraries and header files are provided.

Please refer to README from CyaSSL folder for more information.

#### 1.3.2.12.1 Feature Packs

The TLS libraries are included as multiple feature packs within the WMSDK. Since the footprint impact is high, customers can choose the appropriate feature-footprint tradeoffs out of these feature packs. For example the FP3 feature pack provides a high-security low-footprint TLS library. Details about the exact set of features within the feature packs are available at: *cyassl-feature-packs/feature-packs-matrix.xls* within the WMSDK tarball.

All the feature packs are available in the *cyassl-feature-packs/* directory. For selecting a feature pack, pass the appropriate feature pack option to the build command. For example:

```
$ cd wmsdk_bundle-x.y.z
$ make CYASSL_FEATURE_PACK=fp0
```

OR

```
$ make CYASSL_FEATURE_PACK=fp0_debug
```

If you prefer a debug feature pack, apart from passing the appropriate option, please also set the following build configuration: *Development and Debugging → Enable Debugging Output → Transport Layer Security* and the build configuration: *Modules → TLS → Enable CyaSSL Debug*

### 1.3.2.13 Application Framework

The SDK is equipped with a feature-rich application framework that simplifies application development. The application framework builds on top of the SDK core-runtime and the middleware modules, and implements the state machine that addresses the most common use-cases required by most applications. The application developers can choose the scenario best suited for their model, thus enabling application developers to focus on the core components specific to their application.

### 1.3.2.14 Command Line Interface

The command line module allows applications to add own commands to aid development process. Numerous existing commands allows the developer to control and observe existing modules like WLAN connection manager, PSM, FTFS, etc. The user and extend this by adding own commands using the command line module. Please refer to cli.h documentation from WMSDK reference module and sample applications for example usage.

## 1.3.3 Sample Applications

The WMSDK bundle contains a rich set of sample applications that can be used to either understand the WMSDK features, or as a starting point to build production applications from.

## 1.3.4 Development Tools

To work with the SDK, you will require at a minimum, an ARM compiler toolchain. Please refer to the *Developer Host Setup* document for details. Linux, Windows and Mac host environments are officially supported.

Additional tools needed for development are all bundled with the SDK. These include:

- A build framework to build all of the SDK components that are delivered in source form, and to build your own application software and custom firmware images.

- OpenOCD – a software based JTAG solution.
- Tools to read and write from Serial Flash and EEPROM. These include: writing firmware images into flash, writing file-system images into flash, burning the boot2 boot-loader, reading contents of the flash, etc.

The SDK also includes facilities to support debugging and interactive development. These include:

- A serial console facility over UART.
- A network console facility that can be accessed using telnet clients.
- A simple command-shell with support for tabbing, back-space, etc., and an extensible command-line interface.
- A set of commands exported by various run-time components that allows the users to interact with those components to achieve various tasks. Applications can also add their own command line programs.

Application development using the SDK can be performed by the help of commonly used IDEs like Eclipse. Please refer to the Application Note for building with Eclipse. Developers can also work with their own existing code editors and use command line to build and debug the applications. The WMSDK build system uses standard GNU Makefiles with very few external dependencies. Applications can be debugged using the GNU debugger, gdb.

# 2 What's New in this Release

## 2.1 Feature Enhancements

1. **Evrythng Cloud**: This release includes support for the Evrythng cloud (http://evrythng.com/). This allows secure TLS based communication with the Evrythng cloud server for remote access.
2. **Eclipse Integration**: This release does a better integration with Eclipse out-of-the-box. This Quick Start Guide documents steps for quickly setting up Eclipse on your development host for getting started. The following can now be easily done from Eclipse with minimum setup:
    a. Edit-Compile SDK and applications
    b. Programming the flash
    c. Support for IDE-based debugging
3. **BLE**: This release moves the BLE package out of the WMSDK bundle directory, into a separate tarball. A build command-line option, EXT_BLESTACK_PATH, can be passed to the WMSDK build, pointing to the location where the BLE stack is untarred.

    Additionally, BLE is now split into multiple feature pack that you can choose as per your use. For example,

    ```
    $ cd wmsdk_bundle-x.y.z
    ```

    ```
    $ make EXT_BLESTACK_PATH=/path/to/mrvl_lestack-
    w.x.y.z/mrvl_lestack_feature_packs/mc200/ble-fp0/mrvl_lestack/
    ```
4. **Driver Enhancements:**
    a. Support for the Cortex-**M4** instructions (DSP/Floating point) is now included, by incorporating CMSIS support. Sample applications have been added to demonstrate the behavior.
    b. SSP Driver now supports a wider frequency range
    c. Support for XIP execution from SPI flash (instead of just the QSPI) is included
    d. Support for UART Hardware Flow Control
    e. Ability to put Wi-Fi in power-down and still run MCU at 200MHz off the RC32M
5. **Tools:**
    a. OpenOCD has been updated to OpenOCD v0.9.0
    b. The directories tools/mc200/OpenOCD and tools/mw300/OpenOCD have been removed. Instead a single directory tools/OpenOCD has been created. The flashprog and ramload tools are placed in this directory and can be executed for both the platforms
6. **Others**
    a. Support for compiling assembly language files (.s) in sample applications has been included
    b. Footprint optimizations have been performed throughout the WMSDK to reduce the footprint, particular in the 'debug' version
    c. The leftover free space in SRAM1 can now also be added to the system heap. This lets the application developer use most of the available SRAM, without having to play tricks with the linker script.

A wide range of sample applications has been included. The focus of these is to create ready-to-copy recipes that can be used to quickly implement the features you want to try out.

## 2.2 Release Notes

This release includes bug-fixes, cleanups and enhancements through the various modules of the SDK.

A brief list of new additions is as follows.

### 2.2.1 Enhancements

1) Support for Evrythng Cloud
2) Better integration with Eclipse IDE
3) BLE
   a) Feature Packs for various configurations
   b) BLE commands included in the FCC app
   c) BLE package is moved out of the SDK, as a separate tarball
4) MW300 Drivers
   a) Support for Cortex-"M4" features DSP and Floating point using CMSIS
   b) SSP driver now supports wider frequency range
   c) Support for SPI flash
   d) UART Hardware Flow Control support
   e) Ability to put Wi-Fi in Power-Down and run MCU at 200MHz off the RC32M
5) Tools
   a) OpenOCD has been updated to OpenOCD v0.9.0
   b) The directory structure tools/{mc200,mw300}/OpenOCD has been changed to tools/OpenOCD
6) Others
   a) Support for compiling Assembly language files (.s) in sample applications
   b) Footprint optimizations for ‘debug’ images throughout all modules
   c) Support for adding free space left in SRAM1 to the system heap
   d) A wide range of new sample applications that facilitate simple recipes for specific items

### 2.2.2 API Changes

1) app_network_mgr_configure_uap renamed to app_get_current_uap_network
2) wlan_get_country added to get configured country code
3) wlan_initialize_uap_network API added
4) New APIs fpu_init and fpu_deinit for initializing the FPU

### 2.2.3 Bugfixes

1) I2C1 and I2C2 module frequency configuration is missing
2) I2C doesn’t work reliably in conjunction with PM3 enter-exit
3) Incorrect Audio clock initialization in the SSP driver
4) Memory leak in I2C open/close sequences

5) work-queue should add latest job to the tail instead of adding to head
6) Flash controller's cache should be flushed after flash write/erase operations
7) SSP driver write operation using DMA hangs sometimes
8) ADC configuration issue
9) VCO setting for PLL was incorrect
10) Listen interval configuration isn't notified to the AP in IEEEPS
11) mdns: Increase no. of conflicts permissible from 99 to 999
12) Memory leak in work-queue
13) Compilation fails when used with gcc-arm-none-eabi-4_9-2015q2-toolchain

### 2.2.4 Wi-Fi Firmware Changelog

MW30x:

- Fix the issue for firmware not loading on a few boards

8801:

- Fix a bug with RSN replay counter for host-based supplicant

# 3 Trying Out a Sample Demo

The development kit comes pre-flashed with a sample application called the Wireless Microcontroller Demo (wm_demo). The wm_demo provides a complete end-to-end functionality typically desired by Wi-Fi based devices.

It also provides access to many of the capabilities of the SDK and allows exploration of these capabilities via the command shell and the CLIs available via the shell. The following sections describe how to interact with this demo project.

Please power on your development board; by connecting the board to your development host with a USB cable as described in Section 1.2.4.

## 3.1 Connecting with micro-AP

As with any Wi-Fi enabled device, the first thing that a user would have to do is to configure it to connect to the home Wi-Fi network. By default, the wm_demo starts off in this mode, called the provisioning mode. The WMSDK supports multiple different mechanisms for performing such network provisioning. The wm_demo, by default, uses the uAP provisioning mechanism.

Whenever a new device (or, a device that needs to be reconfigured) boots up, it will boot up in a provisioning mode. In the provisioning mode, the device will start its own Wi-Fi network (micro-AP). A user can connect to this network using any Wi-Fi enabled laptop or a smart phone. The Wireless Microcontroller device not only supports connection at the WLAN level (layer-2), but also at IP layer (layer 3) and above by serving DHCP addresses. The device then uses the web-server interface to serve a web-application that guides the user through the provisioning procedure. If all goes well, at the end of the procedure, the device will be properly configured and can then automatically connect to the home Wi-Fi network.

Please connect to this micro-AP network from a Wi-Fi enabled device. The network name for this network is *wm_demo-wxyz.* The Passphrase for this network is ‘marvellwm’.

## 3.2 Provisioning with Web-App

The wm_demo is equipped with a Web-App to take you through the provisioning process.

Once you are connected to the micro-AP network, as mentioned above, you can access the webpages on the Wireless Microcontroller by typing the following URL in your browser *http://192.168.10.1*.

The web pages for the provisioning mode assist you to perform the configuration of this Wireless Microcontroller. This typically involves selecting a wireless network and providing the passphrase to connect to this network. Once the configurations are performed, the Wireless Microcontroller connects to this home Wi-Fi network.

## 3.3 Provisioning with WPS

Alternatively, if you have a WPS-enabled Access Point, you can also use WPS to provision the Wireless Microcontroller. In order to do this, press-and-hold the push-button *SW2,* on the Development board. Now press the WPS push-button on your router/Access Point (Some Access Points may not support WPS, or may have WPS disabled, by default). This will start the WPS session. Once the WPS session is successful, the development kit will have fetched and stored the wireless settings and connected to the Access Point.

The following table shows the button that should be pressed for initiating the WPS session on each board:

| Target Board | WPS Button on the board |
|---|---|
| 88MW300 RD Board V1.0 | SW2 |
| 88MC200 + SD8801 Development Board V1 | KEY-IO_04 |
| 88MC200 Development Board V2 | KEY-IO28 |
| Lark-mini Development Board V3 | KEY-IO2 |

## 3.4 mDNS/DNS-SD

The Wireless Microcontroller is now connected to the customer's home Wi-Fi network. How does the customer now communicate with the device?

The wm_demo project advertises services using the multicast-DNS/DNS Service Discovery (Bonjour) protocol. This enables service discovery on the local wireless network using mDNS/DNS-SD clients.

Support for this protocol also enables name resolution on the local network. In the wm_demo project all the devices have a hostname of the form *wmdemo-wxyz.local*. Thus instead of typing the IP Address in the browser, the webpages of this demo can be accessed by typing this hostname in the browser: *http://wmdemo-wxyz.local.*. Support for this name resolution is available in the iOS, Android, Linux and Windows operating systems.

## 3.5 Reset to Provisioning

The device can be reset back into the provisioning mode using either of the following options:

1. Click on the *Reset to Provisioning* button in the Web UI
2. Press-and release WAKEUP0 key, to wake the device from Power save, followed by press and-release board_button_2() (Refer to corresponding board file for the pushbutton location)

Once back in provisioning mode the device will host its own micro AP network as explained above.

# 4 Developing with the SDK

## 4.1 Installing the SDK

All the components of the SDK are delivered in the form of a WMSDK bundle. The WMSDK bundle is delivered as compressed tarball. To use the bundle, do the following:

```
# tar zxvf wmsdk_bundle-<version>.tar.gz
```

This will uncompress and untar the contents into a directory named `wmsdk_bundle-<version>`, which will have all of the required components.

**Note** Windows users, please avoid using other programs for uncompressing the WMSDK bundle. These programs do not faithfully uncompress the permissions of all files within the bundle. This could cause issues while using the WMSDK for development.

### 4.1.1 Directory Structure and Contents

The WMSDK bundle directory is organized as follows:

**Table 2: WMSDK Bundle Directory Contents**

| Sub-Directory | Description |
|---|---|
| `boot2 /` | This directory contains the sources of the boot2 boot-loader. The boot-loader is a small program that is executed after BootROM. The boot-loader selects the correct MCU firmware and passes control to it. |
| `sample_apps/` | This directory contains some sample applications that can demonstrate the various features of the WMSDK and can be used as a good starting point to develop your application. |
| `wifi-firmware/` | This directory contains the Wi-Fi firmware images for the Wi-Fi chipsets that are supported by the WMSDK. Please select correct firmware depending on your board. Each firmware is available in two formats compressed and uncompressed. |
| `wmsdk/` | This directory contains the sources of the WMSDK. |
| `phone_apps/` | This directory contains sources for reference Android and iOS applications that are bundled with the SDK. These apps showcase the network provisioning feature. |

## 4.2 Development with Eclipse

**Note** Please make sure that you follow the steps described in the Development Host Setup Guide for Eclipse Setup before performing any of the steps in this section.

### 4.2.1 Preparing the SDK

Depending on the development host machine, please extract either *wmsdk_bundle-x.y.z/settings-windows.tar.gz* or *wmsdk_bundle-x.y.z/settings-mac-linux.tar.gz* file in place.

```
# cd wmsdk_bundle-x.y.z/
# tar -xzf settings-windows.tar.gz
```

If you wish to use any of the existing sample application as a starting point, please copy that application from sample_apps/ directory into the root of the SDK. We assume wm_demo as a starting application for example here

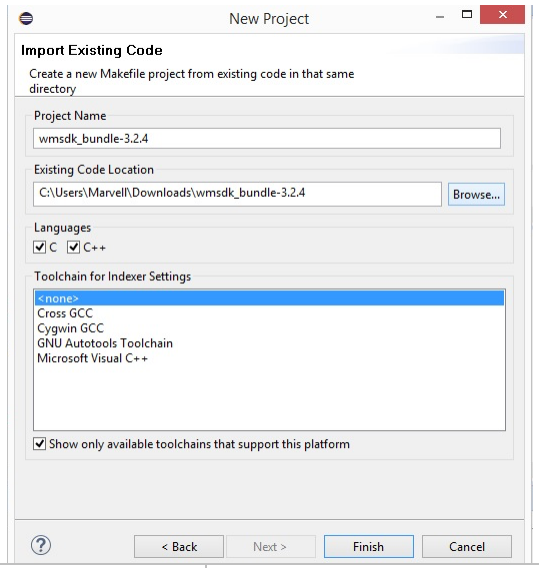
```
# cp -a sample_apps/wlan/wm_demo ./my_app
```

### 4.2.2 Creating Eclipse Project for WMSDK

Start Eclipse. Go to File -> New -> “Makefile Project with Existing Code”. Click on “Browse” button and choose the wmsdk_bundle-x.y.z folder. Select “None” option in the “Toolchain for Indexer Settings”. Click on “Finish” button.

Once the Eclipse project is created, navigate to Project -> Properties menu. Then select “C/C++ Build -> Environment”. Then add following variables.

| Variable Name | Example Value | Comment |
|---|---|---|
| DEFCONFIG | mw300_defconfig | This variable holds name of the SDK configuration file. The SDK configuration files can be found in *wmsdk/build/config* directory in the SDK |

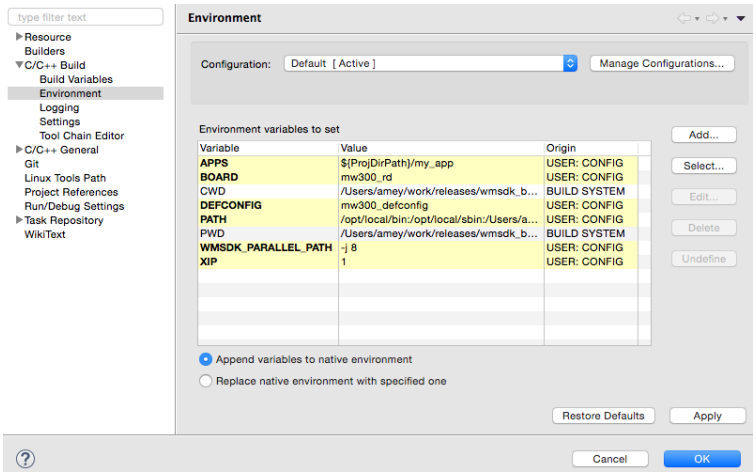



| | | |
|---|---|---|
| BOARD | mw300_rd | This variable holds name of the board for which the boot2 and applications are compiled for. All the supported boards can be found in wmsdk/src/boards directory. Note that the value does not have “.c” extension in them. |
| XIP | 0 | This variable is only applicable for MW300 based systems. This variable indicates if the application is compiled with XIP (execute-in-place) support or |

| Variable Name | Example Value | Comment |
|---|---|---|
| | | not. |
| APPS | ${ProjDirPath}/my_app | This variable indicates which application is to be compiled. If you have copied your own application into root of the SDK, please use ${ProjDirPath}/ prefix. You can also specify a specific application inside sample_apps as “hello_world”. Not setting this variable will compile all the sample applications. |
| WMSDK_PARALLEL_MAKE | -j 4 | This is an optional flag that allows parallel make to compile the SDK faster. |

## 4.2.6 Building SDK and Applications

The SDK and application can be built using “Project -> Build Project” option. Upon successful build completion, the application binary will be generated in the my_app/bin folder.

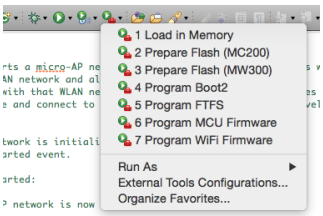



## 4.2.7 Programming the Board

WMSDK release provides pre-configured “External Tools” in the Eclipse IDE that can take care of the board programming.

### 4.2.7.1 Preparing Flash

This tool provides a way to erase the entire flash of the board and program the partition layout. There are two separate options available for MC200 and MW300 based boards. Please connect the board to the development host and select appropriate option.



#### 4.2.7.2 Program Boot2

The board needs to be programmed second level boot-loader when the flash is erased. The compiled boot2 image is found in the SDK release at bin/<board name> directory. Please note that you should select the boot2.bin in the Project Explorer pane and then click on “Program Boot2” external tool to program the boot2 binary on the board.

#### 4.2.7.3 Program WiFi Firmware

For WiFi application, WiFi firmware needs to be programmed in the board. Each WMSDK release has pre-built WiFi firmware images available for different Marvell WiFi chips. These firmware can be found in wifi-firmware/ directory in the SDK. Please select appropriate WiFi firmware (bin or bin.xz) in the Project Explorer pane and then then click on “Program WiFi Firmware” external tool to program the WiFi firmware binary on the board.

#### 4.2.7.4 Program MCU Firmware

The compiled application firmware image (.bin) is found in bin/<board name> directory. Please note that you should select the <application>.bin in the Project Explorer pane and then click on “Program MCU Firmware” external tool to program the MCU firmware on the board.

#### 4.2.7.5 Program FTFS

If the application uses the FTFS, the compiled FTFS image can be found along side of the application firmware binary. This FTFS image should be selected in the Project Explorer pane and then click on “Program FTFS” external tool to program the FTFS on the board.

### 4.2.8 Loading and Executing Application from Memory

“Load in Memory” external tool provides ability to load the application in the board memory and starts execution. Please select the compiled application executable (axf file) in the Project Explorer pane and then click on “Load in Memory” external tool to load and execute from memory.

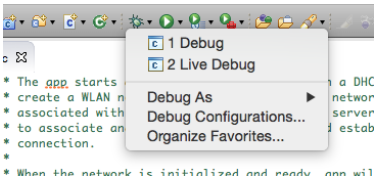
### 4.2.9 Debugging

Debug and “Live Debug” options are provided for debugging of the application on the board.

Debug option can be used for non-XIP (XIP=0) images in case of MW300 based board. For MC200, the application should not be using overlays in order to debug with “Debug” option. The application is loaded in the memory and executed by the debug option.

Live Debug option can be used in all the situations where the board already has a running application and debugger needs to be connected without resetting the board. This option also works with XIP applications and overlays enabled applications.

In order to use any of these two debug options, please select the application executable (axf file) in the Project explorer pane, and then click on the appropriate debug option. All the standard Eclipse debugging features can be used then.



# 4.3 Development with Command-line

## 4.3.1 Building the SDK

The following steps build the WMSDK and the applications:

**Note** The WMSDK bundle should be extracted to a path that does not contain spaces.

```
# cd wmsdk_bundle-x.y.z/
# make
```

**Note** The generated application images are stored in the *bin/*directory within every application.

Whenever any application is built, the build generates two images:

1. wm_demo.axf
2. wm_demo.bin

The axf file can be used to directly load the image into the SRAM of the microcontroller. This is a fast operation and is useful for iterative development of the firmware. The bin file can be used to flash to the microcontroller. This then gets executed on the next bootup.

### 4.3.1.1 Customizing the build for your board

All the typical board-specific changes have been abstracted by the SDK into a board file. By default, the accessories are built for the *mw302_rd* board. Additionally, support is available to build for other boards. All the board files are available in the *wmsdk/src/boards/* directory.

If you have made any customizations for your board that affect the SDK, you should create your own board file accordingly. The board files that are available in the SDK serve as a good starting point to write your own board file. You may refer to the board.h file documentation in the *WMSDK Reference Manual* for more details about each API that should be implemented.

A make variable 'BOARD=' should be passed to the build to indicate which board to build the applications for.

```
# cd wmsdk_bundle-x.y.z/
# make BOARD=aw-cu288
```

## 4.3.2 Executing Firmware Images from Memory

**NOTE:** This is not possible for code running in XiP mode. By default, the generated images are non-XIP.

On building the firmware, the SDK generates an *`application-name.axf`* file that can directly be loaded into the memory for execution. This is very helpful during iterative development.

**Note**: other components like Wi-Fi firmware image that are used by the axf file being loaded should already be present in the flash.

You can ramload images into the microcontroller as shown below:

```
# cd wmsdk_bundle-x.y.z/wmsdk/tools/OpenOCD
```

```
# sudo ./ramload.sh /path/to/hello_world.axf
```

**Note**

OpenOCD does not work if the device is in any of the power save states: PM2, PM3 or PM4, This is because power to the JTAG pins is turned off. For OpenOCD to work, please ensure that the device comes out of these PS states, either by pressing the wakeup key or the reset key.

If your application immediately puts the device into PS modes, you may have to press-and-hold the wakeup key while you execute the openocd utilities like *ramload* and *flashprog*.

On successful load of the image into the RAM, you will see messages like the following on the screen on your development host.

```
which: no arm-none-eabi-readelf in (/sbin:/bin:/usr/sbin:/usr/bin)
Open On-Chip Debugger 0.7.0 (2013-11-20-20:46)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.sourceforge.net/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'jtag'
adapter speed: 3000 kHz
adapter_nsrst_delay: 100
jtag_ntrst_delay: 100
cortex_m3 reset_config sysresetreq
sh_load
Info : clock speed 3000 kHz
Info : JTAG tap: mw300.cpu tap/device found: 0x4ba00477 (mfg: 0x23b,
part:
0xba00, ver: 0x4)
Info : mw300.cpu: hardware has 6 breakpoints, 4 watchpoints
Info : JTAG tap: mw300.cpu tap/device found: 0x4ba00477 (mfg: 0x23b,
part:
0xba00, ver: 0x4)
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00007f14 msp: 0x20001000
requesting target halt and executing a soft reset
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00007f14 msp: 0x20001000
294980 bytes written at address 0x00100000
4440 bytes written at address 0x20000040
downloaded 299420 bytes in 2.113963s (138.320 KiB/s)
verified 299420 bytes in 2.715310s (107.687 KiB/s)
```

```
shutdown command invoked
```

If you have a serial console connected to the device, then messages from the application being executed will show up on the serial console.

### 4.3.3 Flashing the Firmware Image

On building the accessory firmware, an *application-name.bin* file, which can be burnt into the flash is generated. If the system is set up correctly, the firmware image will be executed automatically when the system boots up.

Firmware images are burnt into the flash using the `flashprog.sh` utility that is bundled with the SDK. The utility is available in *hap_sdk_bundle-x.y.z/hap_sdk/tools/OpenOCD* directory and is executed via OpenOCD.

This can be executed as:

```
# cd wmsdk_bundle-x.y.z/wmsdk/tools/OpenOCD
# ./flashprog.sh --mcufw /path/to/hello_world.bin
```

When this executes, you will see the output from openocd like below:

```
which: no arm-none-eabi-readelf in (/sbin:/bin:/usr/sbin:/usr/bin)
Open On-Chip Debugger 0.7.0 (2013-11-20-20:46)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.sourceforge.net/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'jtag'
adapter speed: 3000 kHz
adapter_nsrst_delay: 100
jtag_ntrst_delay: 100
cortex_m3 reset_config sysresetreq
sh_load
Info : clock speed 3000 kHz
Info : JTAG tap: mw300.cpu tap/device found: 0x4ba00477 (mfg: 0x23b,
part:
0xba00, ver: 0x4)
Info : mw300.cpu: hardware has 6 breakpoints, 4 watchpoints
Info : JTAG tap: mw300.cpu tap/device found: 0x4ba00477 (mfg: 0x23b,
part:
0xba00, ver: 0x4)
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00007f14 msp: 0x20001000
requesting target halt and executing a soft reset
target state: halted
target halted due to debug-request, current mode: Thread
```

```
xPSR: 0x01000000 pc: 0x00007f14 msp: 0x20001000
28216 bytes written at address 0x00100000
downloaded 28216 bytes in 0.196813s (140.004 KiB/s)
verified 28216 bytes in 0.361205s (76.285 KiB/s)
semihosting is enabled

Flashprog version: 2.0.6
Writing "mcufw" @0x62000 (primary).....done
Please press CTRL+C to exit.
Exiting.
```

## 4.4 Sample Applications

The Wireless Microcontroller SDK includes a number of sample application projects that illustrate the various features of the SDK and the evaluation kit. These sample applications can be run without requiring the user to set up a working compiler and debugger toolchain and facilitate an easy review and evaluation of some of the capabilities provided by the SDK.

The sample applications also illustrate how various common tasks can be accomplished and how the API's provided by the SDK can be used. Reading through the code of the sample applications is a good way to get familiar with the programming model and APIs. Once you are ready to start development, these applications then provide a quick start and you can begin with simple customizations to these applications to get familiar with the development tools and environment, and with the API's. You may want to refer to the *WMSDK Reference Manual* that documents all the exported APIs of the WMSDK.

Finally, many of the sample applications are set up to accomplish a common use-case task, and the code from these applications can be used as the basis for developing your own custom application. The sample applications are available as a part of the SDK bundle.

### 4.4.1 Hello World

The Hello World project is a simple application to illustrate how to integrate and build a custom application firmware image. The following sections highlight some of the key aspects of an application project. The code for the application is stored in a single C code file (main.c).

#### 4.4.1.1 Initialization

The entry point for an application project is the `main()` function that gets called during system initialization. The application is expected to do any basic application-specific initialization. Typically, this involves initializing any optional system components, and creating any application threads.

In the Hello World project, the application initialization keeps printing a message on the serial console; infinitely in the context of the system initialization thread itself.

```
int main(void)
{
        int count = 0;

        /* Initialize console on uart0 */
        wmstdio_init(UART0_ID, 0);
```

```
        wmprintf("Hello World application Started\r\n");

        while (1) {
                count++;
                wmprintf("Hello World: iteration %d\r\n", count);

                /* Sleep  5 seconds */
                os_thread_sleep(os_msec_to_ticks(5000));
        }
        return 0;
}
```

**Note** The SDK uses the FreeRTOS Operating System underneath. If you are familiar with the FreeRTOS primitives, you may directly use them instead of using the OS abstraction layer primitives as shown in the sample applications.

#### 4.4.1.2 Application Output

When the Hello World firmware image is executed, you will see the following messages on the serial console.

```
Hello World application Started
Hello World: iteration 1
Hello World: iteration 2
Hello World: iteration 3
Hello World: iteration 4
Hello World: iteration 5
```

## 4.5 Build Configurations

To configure various options pertaining to the SDK build, you may run the following:

```
# make config
```

This will present various configuration options that can be tweaked in the build. Linux users may also use the more user interactive *make menuconfig* command.

There may be scenarios when you may want to only change a single option instead of going through the entire list of options presented by the *make config* command. In such situations you can follow this procedure:

```
# vi .config
# (update the option that you wish to change)
# make oldconfig
```

This will rebuild the build configuration but it will only query you for the options that you modified in the *.config* file.

## 4.6 Code Documentation

Now that you have started development with the WMSDK, it would be good to look at in-depth documentation about the various modules, and the corresponding APIs provided by the WMSDK. Please refer to the *WMSDK Reference Manual* for this documentation.

# 5 Command Shell

The Wireless Microcontroller SDK includes a command-shell that allows the users to interact with the built-in modules using a command-line interface (CLI).

The CLI provides an extensible set of commands that can be invoked via the command shell. The various components provided with the SDK provide commands that can be used from within the command shell. These are described in the sections below.

**Note**

For footprint reasons, not all the commands mentioned below are activated in the default WMSDK build. You may have to enable cli functions of some modules for activating them.

## 5.1 General Commands

### 5.1.1 help

The help command displays the list of commands that are registered with the command shell.

#### Syntax:

```
# help
```

#### Sample Output:

```
#help
system-conf
time-get
time-set <year><month><day><hour><min><sec> (all numeric)
time-get-posix
time-set-posix <sec_since_epoch>
pm-reboot
pm-mcu-cfg <enable><mode><threshold>
pm-mcu-state <state> [timeout in milliseconds]
sysinfo (see sysinfo -h for details)
memdump (memdump.[b|h|w] <address><# of objects>)
memwrite (memwrite.[b|h|w] <address><value> [count])
psm-register <module><partition-key>
psm-get <module><variable>
psm-set <module><variable><value>
psm-erase
psm-dump <partition_no>
```

```
psm-get-free-space <module>
ftfs-ls
ftfs-cat [-u] <filename>
ftfs-hexdump <filename>
rfprint <http_url>
updatefw <http_url>
updatewififw <http_url>
updatefs <fs-name><http_url>
wlan-scan
wlan-add <profile_name> ssid <ssid> bssid...
wlan-remove <profile_name>
wlan-list
wlan-connect [profile_name]
wlan-start-network <profile_name>
wlan-stop-network
wlan-disconnect
wlan-stat
wlan-info
wlan-address
wlan-mac
wlan-set-regioncode
wlan-get-regioncode
wlan-get-uap-channel
pm-ieeeps-hs-cfg (see pm-ieeeps-hs-cfg -h for details)
pm-wifi-deepsleep-enter (see pm-wifi-deepsleep-enter -h for details)
pm-wifi-deepsleep-exit
pm-wifi-configure-listen-interval <time in milliseconds>
pm-wifi-uap-inactivity-ps-enter <min time in microseconds><max time in
microseconds>
pm-wifi-uap-inactivity-ps-exit
wlan-gethostbyname <hostname>
pm-wifi-pdn-enter (see pm-wifi-pdn-enter -h for details)
pm-wifi-pdn-exit

#
```

### 5.1.2 system-conf

The system-conf command displays the system configuration information. It includes SDK version number, Compiler version, CPU Clock frequency of the SDK.

#### Syntax:

```
# system-conf
```

### Sample Output:

```
# system-conf

SDK version: 3.0
Compiler version: 4.8.3 20131129 (release) [ARM/embedded-4_8-branch
revision 205641]
CPU clock frequency: 199600000Hz
Code execution mode: Non-XIP
```

## 5.2 WLAN Commands

### 5.2.1 wlan-scan

This command scans and displays available wireless networks with their BSSID, SSID (network name), channel, WMM information and security information.

| | |
|---|---|
| **Note** | The wlan-scan command executes asynchronously. That is, the command returns immediately, while the scan operation is being carried out by the connection manager. As a result, the output from the command is displayed when the scan operation is complete. The user can continue interacting with the shell while the scan operation is executing. |

### Syntax:

```
# wlan-scan
```

### Sample Output:

```
# wlan-scan
Scan scheduled…

# 4 networks found:
00:0A:B8:7E:D7:C0  “DemoNet” infra
channel: 6
        rssi: -48 dBm
        security: WPA
        WMM: NO
        WPS: NO
00:50:43:20:F1:1A  “Home-Network” infra
        channel: 6
        rssi: -43 dBm
        security: WPA2
        WMM: NO
        WPS: YES, Session: Not active
00:19:E8:AF:4E:90  ”Mrvl” infra
        channel: 6
rssi: -48 dBm
security: WPA
```

```
        WMM: NO
        WPS: NO
02:2E:DE:AF:80:1E  ”upstairs” adhoc
channel: 6
        rssi: -52 dBm
        security: OPEN
        WMM: NO
        WPS: YES, Session: Not active
```

## 5.2.2 wlan-add

The wlan-add command is used to create a new network profile and add it to the list of networks known to the connection manager. The WLAN connection manager expects a unique name for each network profile, which is used to refer to the network in other commands.

The wlan-add command expects several additional arguments in the form of `<key><value>` to input relevant characteristics of the network profile. Some examples of how to use wlan-add are given below:

### Syntax:

```
For static IP address assignment:
    wlan-add <profile_name> ssid <ssid> bssid <bssid> channel
<channel> ip:<ip_addr>,<gateway_ip>,<netmask>
For DHCP IP Address assignment:
    wlan-add <profile_name> ssid <ssid> [bssid <bssid>]
For WEP security mode:
    wlan-add <profile_name> ssid <ssid> wep open <WEP_key>
For WPA/WPA2 security mode:
    wlan-add <profile_name> ssid <ssid> wpa2 <secret>
```

### Sample Output:

```
# wlan-add home ssid Home-network wpa2 mypasswd
Added “home”
```

## 5.2.3 wlan-remove

This command deletes an existing wireless connection profile from the WLAN Connection Manager’s list of known profiles.

### Syntax:

```
# wlan-remove <profile name>
```

### Sample Output:

```
# wlan-remove home
Removed “home”
```

## 5.2.4 wlan-list

This command lists all existing wireless connection profiles.

### Syntax:

```
# wlan-list
```

### Sample Output:

```
# wlan-list
1 network:
"default"
        SSID: DemoNet
        BSSID: 00:21:29:76:78:8B
        channel: 1
        mode: infra
        security: none
        address: static
                IP: 192.168.1.99
                gateway: 192.168.1.1
                netmask: 255.255.255.0
                dns1: 192.168.1.1
                dns2: 192.168.1.1
```

## 5.2.5 wlan-connect

This command can be used to connect to wireless network using a specified profile. If no network is specified it connects to the first available network.

| | |
|---|---|
| Note | The wlan-connect command executes asynchronously. . That is, the command returns immediately, while the connect operation is being carried out by the connection manager. As a result, the output from the command is displayed when the connect operation is complete. The user can continue interacting with the shell while the connect operation is executing. |

### Syntax:

```
# wlan-connect <profile name>
```

### Sample Output:

```
# wlan-connect test
Connecting to network…
Use 'wlan-stat' for current connection status.

#
```

## 5.2.6 wlan-disconnect

This command can be used to disconnect from a wireless network.

### Syntax:

```
# wlan-disconnect
```

### Sample Output:

```
# wlan-disconnect
#
```

## 5.2.7 wlan-stat

This command displays the state of the wireless connection.

### Syntax:

```
# wlan-stat
```

### Sample Output:

```
# wlan-stat
connected
```

## 5.2.8 wlan-info

This command displays detailed information about the connected network.

### Syntax:

```
# wlan-info
```

### Sample Output:

```
# wlan-info
Connected to:
“default”
        SSID: DemoNet
        BSSID: 00:21:29:76:78:8B
        channel: 1
mode: infra
        security: none
        address: static
                IP: 192.168.1.99
                gateway: 192.168.1.1
                netmask: 255.255.255.0
                dns1: 192.168.1.1
                dns2: 192.168.1.1
```

## 5.2.9 wlan-address

This command displays the IP address assignment method, device IP address, gateway IP address and netmask.

### Syntax:

```
# wlan-address
```

### Sample Output:

```
# wlan-address
        address: static
                IP: 192.168.1.99
                gateway: 192.168.1.1
                netmask: 255.255.255.0
                dns1: 192.168.1.1
                dns2: 192.168.1.1
```

## 5.2.10 wlan-mac

This command displays the MAC address of the device.

### Syntax:

```
# wlan-mac
```

### Sample Output:

```
# wlan-mac
00:21:E8:2D:99:83
```

## 5.2.11 wlan-gethostbyname

This command performs a DNS resolution and returns the IP Address associated with a hostname.

### Syntax:

```
# wlan-gethostbyname <hostname>
```

### Sample Output:

```
# wlan-gethostbyname marvell.com
10.4.5.110
```

## 5.2.12 uaputl

This command is used to configure the microAP interface

### Syntax:

```
# Usage: uaputl <parameter> [<value>]
```

```
To get/set the ssid:
uaputl sys_cfg_ssid [<ssid_name>]

To set wpa2 passphrase:
uaputl sys_cfg_wpa_passphrase <passphrase>

To start in uap mode:
uaputl bss_start

To stop the uap mode:
uaputl bss_stop
```

### Sample Output:

```
# uaputl sys_cfg_ssid test_uap
SSID setting successful

# uaputl sys_cfg_wpa_passphrase marvell88
WPA passphrase setting successful

# uaputl bss_start
Added "uap"
```

## 5.2.13 iwpriv

This command allows you to configure Wi-Fi device-specific private configurations for a wireless device. Note that some of these commands are available only under the *Enhanced WLAN CLI commands* option. The currently supported options are:

- version: Retrieve the WLAN firmware version
- verext: Retrieve the extended WLAN firmware version
- sta_list: If in AP mode retrieve the list of associated clients
- pscfg: Get/set the power save configuration parameters
- deauth: Disconnect from a network
- passphrase: Connect to the given network
- deepsleep: Get/set deep sleep mode
- getdatarate: Retrieve the data date
- getsignal: Retrieve RSSI, SNR and RF
- antcfg: Get/set the mode of Tx/Rx antenna

### Syntax:

```
# iwpriv
Usage: iwpriv <interface><command>

For getting the Firmware version:
iwpriv <interface> version
```

For getting the **Firmware version ext**:
iwpriv <interface> verext

For getting the **stations list**:
iwpriv <interface> sta_list

For **disconnecting** from the network:
iwpriv <interface> deauth

For **connecting** to a network with wpa2 security:
iwpriv <interface> passphrase
"1;ssid=<ssid>;passphrase=<passphrase>"

To set/get the **mode of Tx/Rx antenna**:
iwpriv <interface> antcfg [m]
where value of m is:
Bit 0 -- Tx/Rx antenna 1.
Bit 1 -- Tx/Rx antenna 2.
0xFFFF (65535) -- Tx/Rx antenna diversity.
Examples:
iwpriv <interface> antcfg : Get Tx/Rx antenna mode
iwpriv <interface> antcfg 1 : Set Tx/Rx antenna 1
iwpriv <interface> antcfg 65535 : Set Tx/Rx antenna diversity

To get the **data rate** (index) being used in last Tx packet and last Rx packet:
iwpriv <interface> getdatarate

To get the last and average value of **RSSI, SNR and NF** of Beacon and Data:
iwpriv <interface> getsignal
<interface> getsignal:-32 -33 -35 -36 67 59 63 56 -99 -92 -98 -92
RSSI info: beacon last -32, beacon average -33, data last -35, data average -36
SNR info: beacon last 67, beacon average 59, data last 63, data average 56
NF info: beacon last -99,beacon average -92, data last -98, data average -92
Note: This command should be used only when STA is connected to some access point.

For set/get auto **deep sleep** mode:
iwpriv <interface> deepsleep [n] [m]
where the parameters are:
[n]: Enable/disable auto deep sleep mode (1/0)
[m]: Idle time in milliseconds after which firmware will put the device
in deep sleep mode. Default value is 100 ms.

For set/get **STA PS configuration** parameters:
iwpriv <interface> pscfg [k] [d] [l] …
Where the parameters:
[k]: Keep alive null packet interval (0: Unchanged, -1: Disable, n: Interval in seconds)

```
[d]: DTIM interval (    0: Unchanged,
                      1-5: Value,
                    65534: DTIM will be ignored, listen interval
will be used,
                    65533: Closest DTIM to the listen interval
period will be used )
[l]: Local listen interval (     0: Unchanged,
                                -1: Disable,
                             1-49: Value in beacon intervals,
>= 50: Value in Tus )
[b]: Beacon miss timeout (0: Unchanged, 1-50: Value in
milliseconds, 65535: Disable)
[p]: Delay to PS (0-65535: Value in milliseconds, default 1000ms)
[m]: PS mode (0: Unchanged, 1: Auto mode, 2:  PS-Poll mode, 3: PS
Null mode)
No change if parameters are not provided.
```

### Sample Output:

```
# iwpriv mlan0 version
Version: SD878x-14.61.6.p36-702.1.0-WM

# iwpriv mlan0 verext
Version ext: w8787-Ax, RF878X, FP61, 14.61.6.p36, BT_SDIO

# iwpriv mlan0 getdatarate
mlan0     getdatarate:11  0


# iwpriv mlan0 getsignal
mlan0     getsignal:-55  -55  -58  -98  51  43  47  0  -106  -98
-105  -98

# iwpriv mlan0 antcfg
mlan0     antcfg:65535
```

## 5.2.14 iwlist, iwconfig, ifconfig

These commands work as their Linux counterparts. Please type the command name, followed by enter key to get the list of supported options.

# 5.3 File System Commands

ftfs is a tiny flash file system layer. This section describes basic commands that are implemented on this ftfs layer.

## 5.3.1 ftfs-ls

This command lists all the files available on the ftfs flash partition.

### Syntax:

```
# ftfs-ls
```

### Sample Output:

```
# ftfs-ls
"404.html" (146 bytes)
"index.html" (482 bytes)
"network.conf" (102 bytes)
```

## 5.3.2 ftfs-cat

This command displays the contents of a specified file.

### Syntax:

```
# ftfs-cat <filename>
```

### Sample Output:

```
# ftfs-cat network.conf
opened "network.conf", length: 102 bytes
"ssid=home-network
ip=dhcp
addr=192.168.1.100"
```

## 5.3.3 ftfs-hexdump

This command displays the contents of a specified file in the hex format.

### Syntax:

```
# ftfs-hexdump <filename>
```

### Sample Output:

```
# ftfs-hexdump network.conf
opened "network.conf", length: 102 bytes
0000 :: 3C 21 2D 2D 23 69 6E 63 6C 75 64 65 20 76 69 72
0010 :: 74 75 61 6C 3D 22 61 6° 61 6D 77 63 2E 70 68 70
```

# 5.4 Power Manager Commands

## 5.4.1 pm-reboot

This command reboots the Wireless Microcontroller device.

### Syntax:

```
# pm-reboot
```

### Sample Output:

```
# pm-reboot
```

## 5.4.2 pm-mcu-state

This command switches the CM3 core into the various power save modes.

### 5.4.2.1 PM2

This command switches the Cortex-M3/M4 core, most of the MCU peripherals and SRAM arrays in a low power-mode. The PMU and RTC are operational. The system can get out of standby mode by a pre-defined timeout. If no timeout is specified, the *SW3* key on the evaluation kit can be used to get the system out of standby.

### Syntax:

```
# pm-mcu-state2 [duration in milli-seconds]
```

### Sample Output:

```
# pm-mcu-state 2 10000

Timeout specified :10000 millisec
The system can be brought out of standby by an RTC timeout or a
wakeup key press
 PM0->PM2        For 10000 millisec

 PM2->PM0        (Wakeup by Timeout 10000 millisec)
```

### 5.4.2.2 PM3

This command switches the CM3/CM4 core, most of the MCU peripherals and SRAM arrays in a low power-mode. PM3does not retain registers and retains smaller amount of RAM than PM2. The PMU and RTC are operational. The system can get out of standby mode by a pre-defined timeout. If no timeout is specified, the *SW3* key on the evaluation kit can be used to get the system out of standby.

### Syntax:

```
# pm-mcu-state 3 [duration in milli-seconds]
```

### Sample Output:

```
# pm-mcu-state 3 10000

Timeout specified :10000 millisec
```

```
The system can be brought out of standby by an RTC timeout or a
wakeup key press
 PM0->PM3        For 10000 millisec

 PM3->PM0        (Wakeup by Timeout 10000 millisec)
```

#### 5.4.2.3 PM4

This command switches most of the components off. RTC is kept active.On wakeup from this state execution begins at the bootrom.

### Syntax:

```
# pm-mcu-state 4 [duration in milli-seconds]
```

### Sample Output:

```
# pm-mcu-state 4
No Timeout specified
Wakeup will occur by Wakeup key press
 PM0->PM4
[boot] SDK-2.9.23 gcc-4.7.2 (Feb  8 2013 04:52:30)
```

## 5.4.3 pm-ieeeps-hs-cfg

This command configures the power manager framework to opportunistically take the Wi-Fi chip into IEEEPS power save modes. This command should be called only in the station connected mode. The first parameter indicates whether the ieeeps power save should be enabled or disabled, while the second parameter indicates on what network conditions should the MC200/MW300 be woken up

### Syntax:

```
# pm-ieeeps-hs-cfg <enabled><wakeup_condition>

# pm-ieeeps-hs-cfg -h
Usage:
pm-ieeeps-hs-cfg <enabled><wakeup condition>
enabled: 1 to enable
         0 to disable
wakeup conditions : host wakeup conditions
bit0=1: broadcast data
bit1=1: unicast data
bit2=1: mac events
bit3=1: multicast data
bit4=1: arp broadcast data
```

### Sample Output:

```
# pm-ieeeps-hs-cfg 1 15
```

```
# [app-wm-demo] wm_demo app: Received event 29
[app-wm-demo] Power save enter :1
```

### 5.4.4 pm-mcu-cfg

This command configures the power manager framework to opportunistically take MCU into power save modes whenever possible. The power manager framework internally calls a function that is equivalent of pm-mcu-state at opportune moments to take MCU into power save. The first parameter indicates whether to enable (1) or disable (0) this power save mode, the second parameter indicates whether the mcu should be taken into the PM2 (2) or PM3 (3) mode, while the third parameter indicates *threshold-time*. The power manager will put the MCU into PS mode if there is no scheduled activity for atleast *threshold-time* in the future.

#### Syntax:

```
# pm-mcu-cfg [enable] [mode] [threshold]
```

#### Sample Output:

```
# pm-mcu-cfg 1 2 10
```

### 5.4.5 pm-wifi-pdn-enter

This will put the Wi-Fi card into the power down mode. The mode gets enabled until a corresponding *pm-wifi-pdn-exit* command gets called. On wakeup from PDN, the wireless firmware is downloaded to the Wi-Fi card/section again.

#### Syntax:

```
# pm-wifi-pdn-enter
```

#### Sample Output:

```
# pm-wifi-pdn-enter
Enabled Power Down mode

# pm-wifi-pdn-exit

# Disabled Power Down mode
[wlan] WLAN SDIO card detected
[wlan] SDIO : Set card to 50 MHz 4-bit mode
[wlan] SDIO : Change Clock Frequency
[wlan] SDIO : Set to 4-bit mode
[wlan] SDIO : Card Version - (0x32)

[wlan] WLAN FW download Successful
```

### 5.4.6 pm-wifi-configure-listen-interval

This command configures the listen interval for the Wi-Fi card/section when it is in IEEE PS mode. Longer listen intervals enable the Wi-Fi card/section to conserve more power by going to sleep for longer durations. Running the *pm-wifi-ieeeps-enter* command after configuring the listen interval will put the Wi-Fi card/section into IEEE PS as configured. The maximum supported value of the listen interval is 65 seconds.

#### Syntax:

```
# pm-wifi-configure-listen-interval<time-in-msecs>
```

#### Sample Output:

```
# pm-wifi-configure-listen-interval 30
#
```

### 5.4.7 pm-wifi-deepsleep-enter

This will put the Wi-Fi card/section into deep sleep mode. The mode gets enabled until a corresponding *pm-wifi-deepsleep-exit* command gets called. In this mode, connection with the Access Point is lost.

#### Syntax:

```
# pm-wifi-deepsleep-enter
```

#### Sample Output:

```
# pm-wifi-deepsleep-enter
Enabled DeepSleep mode

# pm-wifi-deepsleep-exit
Disabled DeepSleep mode
```

## 5.5 Persistent Storage Manager Commands

The persistent storage manager (psm) provides a mechanism to store and retrieve key-value pairs (variables) in Flash. This is typically used for persistent configuration data.

The space is divided into partitions of 4KB each. This allows each partition to be stored on a single sector on the Flash. Due to the characteristics of the serial flash, any updates to a variable must read the entire sector corresponding to the partition, and then update it as a whole.

Each variable stored on the flash is associated with a module, which enables partitioning of the name-space to avoid variable-name conflicts. The modules may share the same partition as long as the data for both these modules can be accommodated within the partition.

### 5.5.1 psm-register

This command is used to register a module name with the persistent storage manager, and associate a partition with the module. The partition key is an arbitrary string that is used only as an identifier. If two modules are registered with the same partition key, then they share the same physical psm partition. If they have different partition keys, then they will be on distinct psm partitions.

#### Syntax:

```
# psm-register <module><partition-key>
```

#### Sample Output:

```
# pm-register network key1
```

### 5.5.2 psm-set

This command sets the value of a variable and stores it in flash.

#### Syntax:

```
# psm-set <module><variable-name><value>
```

#### Sample Output:

```
# psm-set network ssid myhome-ap
```

### 5.5.3 psm-get

This command retrieves the value of a variable from flash.

#### Syntax:

```
# psm-get <module><variable-name>
```

#### Sample Output:

```
# pm-get network ssid
[psm] Value: myhome-ap
```

### 5.5.4 psm-dump

This command will dump all of the key-value pairs stored in a given partition on the flash.

#### Syntax:

```
# psm-dump <partition>
```

### Sample Output:

```
# psm-dump 1
WMC.network.ssid=myhome-ap
WMC.network.bssid=01:21:a3:5f:00:33
```

## 5.5.5 psm-erase

This command will erase and re-format the entire block used by the persistent storage manager. This will also erase any modules that were registered with the psm. The modules should be re-registered with the psm. Note that if you are using the application framework it will automatically register the modules that it uses on every boot-up.

### Syntax:

```
# psm-erase
```

### Sample Output:

```
# psm-erase
```

## 5.5.6 psm-get-free-space

This command will notify the amount of free space left on a psm partition.

### Syntax:

```
# psm-get-free-space <module>
```

### Sample Output:

```
# psm-get-free-space network
[psm] Free space available: 4090 bytes
```

# 5.6 Remote Firmware Upgrade Commands

## 5.6.1 updatefw

This command will download a new firmware image from a given host and burn it in the flash. On next bootup the newly burnt firmware image is loaded.

### Syntax:

```
# updatefw <http_url>
```

### Sample Output:

```
# updatefw http://192.168.2.102/updates/io_demo.bin
```

```
[rfget] Updating firmware at address 0x68000
[rfget] Firmware update succeeded
```

**Note**

You'll have to setup an HTTP server with the upgrade image at the right location for this to work.

### 5.6.2 updatefs

This command will download a new FTFS filesystem image from a given host and burn it in the flash.

#### Syntax:

```
# updatefs<fs-name><http_url>
```

#### Sample Output:

```
# updatefs ftfs http://192.168.2.102/updates/ftfs_v2.bin
[rfget] FTFS update succeeded
```

**Note**

You'll have to setup an HTTP server with the upgrade image at the right location for this to work.

## 5.7 RTC Commands

### 5.7.1 time-get

This command displays the current time. If no time has been set, time starts with 1 Jan 1970.

#### Syntax:

```
# time-get
```

#### Sample Output:

```
# time-get
Thu 2010 Apr 08 16:44:22
```

### 5.7.2 time-set

This command sets the current time.

#### Syntax:

```
# time-set <year><month><day><hour><min><sec> (all numeric)
```

### Sample Output:

```
# time-set 2010 4 8 16 44 20
New time set:
Thu 2010 Apr 08 16:44:20
```

## 5.7.3 time-get-posix

This command displays the current time in terms of epoch (number of seconds since 1 Jan 1970) . If no time has been set, time starts with 1 Jan 1970.

### Syntax:

```
# time-get-posix
```

### Sample Output:

```
# time-get-posix
912
```

## 5.7.4 time-set-posix

This command sets the current time in terms of epoch.

### Syntax:

```
# time-set-posix<epoch> (all numeric)
```

### Sample Output:

```
# time-set-posix 120000
```

# 6 Appendix: Older Changelog

## 6.1 WMSDK 3.1

### 6.1.1 Feature Enhancements

2. **Secure Boot**: This release includes support for Secure Boot. The 88MW300's BootROM can load images that are encrypted (AES-CCM) and/or signed (RSA). This release includes support for a trusted chain of booting such that:
    a. The BootROM can decrypt and/or verify the boot2 boot-loader
    b. The boot-loader can decrypt and/or verify the microcontroller firmware
    c. The Microcontroller firmware can store encrypted values into the PSM
    d. Additionally, support is also available for locking out JTAG and disabling other boot options
3. **Secure Firmware Upgrade**: The secure firmware upgrades now:
    a. Use version 2 of the upgrade file format. This makes it easier to identify the header and the other payload
    b. Support upgrading from server content where clients push the firmware upgrade. Earlier only support for client, or device pulling the firmware upgrade was available
    c. Support for upgrading Wi-Fi firmware
    **d.** Provide a convenience function for handling ChaCha20-ED25519 based secure firmware upgrade
4. **JSON**: The JSON parser has been updated to
    a. parse strings that are a JSON array at the top-level instead of JSON objects
    b. parse 64-bit integers
5. **EZConnect v2**: This release includes support for EZConnect provisioning version 2. This modified the EZConnect provisioning on-air protocol such that the provisioning experience is more reliable and robust than version 1. There are still known interoperability issues with APs that buffer outgoing multicast traffic
6. **Work Queues**: Work queues have been introduced to offload work to be done in a worker thread's context
7. **mDNS/DNS-SD**: The Bonjour module has been refactored and cleaned up. The latest changes make Bonjour service announcements more robust, and fix the issue of the service disappearing after certain time
8. **HTTP Server**:
    a. Consume all data of a request on the socket in case of an error
    b. Conserve footprint by allowing callers of httpd_use_tls_certificates() to free up the buffers holding the certificate
    c. Fix an issue of the HTTP server hanging up when simultaneous IPv4/v6 requests are made
9. **HTTP Client**: The TLS context creation is now separated from the session creation. With this in place multiple sessions can be created using the same context

10. **Embedded Supplicant**: The 88W8801 firmware now uses an embedded-supplicant. This saves valuable footprint on the 88MC200 host, by offloading the supplicant into the firmware
    a. A configuration option has also been provided to perform the PMK calculation on the host. If the association time using the embedded supplicant is not applicable to you, you can turn this configuration option. While this will add to some footprint on the 88MC200, it won't be as large as the entire supplicant, while at the same time, speeding up the association time.
11. **WebSockets**: This release includes support for secure masking in the WebSockets client.

## 6.1.2 Release Notes

This release includes bug-fixes, cleanups and enhancements through the various modules of the SDK.

A brief list of new additions is as follows.

### 6.1.2.1 Newly Supported Platforms

1. Support for AW-CU277 module based on 88MC200 SoC

### 6.1.2.2 New Features

1. Secure boot support for 88MW30x, ability to boot trusted application firmware along with secure storage (PSM)

### 6.1.2.3 Enhancements

1. JSON: The JSON parse has been updated to
    a. Allow parsing of JSON strings that are Arrays and not Objects
    b. Parse 64-bit integers
2. Secure Firmware Upgrade:
    a. Specification updated to version 2. Please refer secure_upgrade.h for details
    b. Allow server mode upgrades, where clients push the firmware upgrade image to the device
    c. Secure Wi-Fi firmware upgrade support added
3. EZConnect Provisioning v2:
    a. Newer on-air protocol for EZConnect provisioning for making provisioning more robust
    b. There are known issues with APs that buffer outgoing multicast traffic
4. Work queues have been introduced. These can be used for offloading work that can be done in a worker thread
5. Software CRC tables are pre-generated to save on retention RAM in PM3 mode
6. mDNS: Overall cleanup and robustness fixes to ensure the service doesn't disappear after a certain time.
7. HTTPD:
    a. Consume all the data of a request on a socket in case of an error
    b. Allow callers to free certificate buffers after call to httpd_use_tls_certificates()
    c. Fix issue of HTTP server hang-up in networks with both IPv4/v6 traffic

8. HTTPC: separate out tls context creation from session creation
9. Updated 8801 defconfigs to use embedded supplicant within the firmware
    a. Also provided an option for performing the PMK calculations on 88MC200
10. Enable support of masking in websockets
11. Added wlan_trpc_demo and wlan_frame_inject_demo in sample apps

#### 6.1.2.4 API Changes

1. API user_app_hardfault_handler() has been removed. Instead applications need to implement critical_error() function which aggregates different kinds of critical errors including the hard fault
2. New APIs
    - json_set_val_bool() and json_set_array_bool() to create a key with boolean value and create an boolean array element respectively
    - json_is_object() and json_is_array() to detect if the anchor is in an array or object
    - json_array_get_num_elements() to find the number of elements in the array
3. added key-value pair separator as an additional parameter in the mdns_set_txt_rec() API
4. sys_work_queue_get_handle() does not take work queue handle as an argument instead it now returns work queue handle
5. http_open_session() now accepts an additional TLS context parameter. Other parameters are re-organized inside a configuration structure
    - New API http_get_tls_context_from_handle() to get TLS context for the given session
6. New Wi-Fi APIs:
    - wlan_set_low_pwr_mode() API for low power (and low range) mode operation
    - wlan_inject_frame() to support frame injection from 88MC200
    - wlan_remain_on_channel() to stay on a specified channel for specified duration
    - wifi_mem_access() to read/write WLAN adapter memory
    - wlan_enable_ed_mac_mode() API to enable ED MAC mode in Wi-Fi firmware
    - wlan_set_trpc() API to set configuration data for Tx power limitation
7. secure_upgrade() API renamed to client_mode_secure_upgrade(). It also requires a flash component as an additional argument
    - New API server_mode_secure_upgrade() introduced for server mode upgrades
8. Secure firmware upgrade specification updated to version 2. Backward compatibility has been maintained only in fw_generator utility. If an old fw_generator config file is passed while generating the upgrade image, it will generate a version 1 image. Newer configs have an additional fwupg_version field. Once upgraded to newer firmware, only images with version 2 or later will work.
9. New APIs app_provisioning_only_start() and app_provisioning_only_stop() to start and stop provisioning module without registering the provisioning web-handlers
10. New API app_crtl_getmode() to get the current status of provisioning
11. os_recursive_mutex_create(), os_recursive_mutex_get() and os_recursive_mutex_put() APIs added to support recursive mutex

12. New API asctime() to convert the broken-down time value tm into a null-terminated string
13. New API os_get_current_task_handle() to get the current task handle
14. New API adc_drv_deinit() to De-initialize ADC Driver
15. i2c_drv_set_callback() API has been modified to take in user's private pointer as a parameter

#### 6.1.2.5 Bugfixes

1. When TCP/IP send() returns '0' due to peer reset httpd_send() hanged indefinitely
2. dhcp-server: fix a race condition with socket from multiple threads
3. Hardfault in wlan_disconnect if no interface is up
4. Provisioning/wlcmgr: a race condition causes a hard-fault when provisioning is stopped
5. wlcmgr: failed to reconnect to AP with WEP encryption
6. Build error when ROM and PM3 both are enabled in configuration
7. Wlan association failure when CyaSSL FP3 is used
8. A potential deadlock exists if os_rw_lock APIs are used

#### 6.1.2.6 Wi-Fi Firmware Changelog

88W8801:

1. Embedded supplicant support
2. Extended scan support

## 6.2 WMSDK 3.0

### 6.2.1 SDK Features and Enhancements

This release includes the following new features.

1. **ROM Support:** This release includes support for the ROM that is available with the MW300 B0 chipset**.** The use of ROM provides more usable RAM for application firmware's use. The ROM support can be enabled by setting the configuration option *ENABLE_ROM_LIBS* in the build configuration (*System Type → Enable ROM libraries in 88MW300*) The ROM on MW300 contains the text and rodata for
    a. Cryptographic libraries like AES/DES/SHA among others
    b. Compression library xz
    c. Standard C library
2. **CyaSSL:**
    a. The WMSDK now supports multiple CyaSSL feature packs. Pre-built images with various feature-footprint tradeoffs have been included with the SDK. Please refer to Section 1.3.2.12.1 for more details.
    b. The CyaSSL library has now been upgraded to 3.3.0-commercial release.
3. **PM3 in XIP:** The PM3 power-save mode is an aggressive power save mode. In this mode, only a small amount of RAM (192KB) out of 512KB is retained. 160KB in SRAM0

and 32KB in SRAM1. Further all peripherals are switched off in PM3, and their configuration is lost. This helps realize usecases with much lower power requirements. This release includes support for PM3 mode along with XIP. Since in the XIP mode, most of the text/rodata is in flash, this leaves applications with more room to use RAM to realize their usecases.

4. **DMA Driver:** This release includes a generic DMA driver support. This release also includes support for I2C Master DMA read and DMA write.
5. **Extended mDNS (xmDNS):** Extended mDNS extends the specification of mDNS to site-local scope in order to support multi-hop LANs that forward multicast packets.
6. **Other Enhancements:**
    a. Support for Wi-Fi Frame Injection is included. This allows an application to instruct the firmware to transmit full-crafted frames directly from the application.
    b. Wi-Fi Driver is restructured such that it is easier to use independently instead of through the WLAN Connection Manager.

#### 6.2.1.1 Unsupported/Untested Features

1. The Link Time Optimization feature is not functional when XIP is enabled.
2. CyaSSL/TLS cannot be used when ROM support is enabled.
3. While the release builds for the MC200 platform, extensive testing has not been performed for MC200 platforms on this release.

### 6.2.2 Release Notes

**API Changes:**

1. os_timer_is_active changed to os_timer_is_running. Earlier the return values were WM_SUCCESS/-WM_FAIL, now it returns true/false.
2. The CyaSSL CFLAGS were earlier embedded as part of the build system. With this release, all the CyaSSL specific CFLAGS are consolidated in the respective CyaSSL feature pack's settings.h file.
3. The following configuration flags are renamed:

    CONFIG_ENABLE_HTTP ==> CONFIG_ENABLE_HTTP_SERVER

    CONFIG_ENABLE_HTTPS ==> CONFIG_ENABLE_HTTPS_SERVER
4. New API dhcp_get_ip_from_mac() is introduced to retrieve the IP Address of a client connected to the micro-AP interfacel
5. mDNS/DNS-SD refactoring has been performed to optimize the service announcement and deannouncement requests. As a consequence the APIs exposed to the application have changed as follows:
    a. app_mdns_add_service_iface() ==> app_mdns_add_service_arr()
        i. mdns_service_config is no longer used as a parameter
    b. app_mdns_remove_service_iface() ==> app_mdns_remove_service_all(), app_mdns_remove_service_arr()
6. Similarly, the corresponding mdns/DNS-SD APIs have been updated as follows:
    a. mdns_add_service_iface() ==> mdns_announce_service_arr(), mdns_announce_service()
        i. mdns_service_config is no longer used as a parameter
        ii. a call to mdns_iface_state_change() is no longer required after these calls

   b. mdns_remove_service_iface() ==> mdns_deannounce_service_all(), mdns_deannounce_service_arr(), mdns_deannounce_service()

7. mdns_query_monitor() and dnssd_query_monitor() now take an interface handle as an argument, instead of registering interface via mdns_add_service_iface()
8. API user_app_hardfault_handler() has been removed. Instead applications need to implement critical_error() function which aggregates different kinds of critical errors including Hard fault
9. API json_str_init is changed. The last argument regarding text formatting is not required now.
10. New API app_get_connection_status() has been introduced to give out connection status, failure reason and number of failed attempts of connection to the application.
11. New APIs app_provisioning_only_start()/app_provisioning_only_stop() have been added to start/stop the provisioning module without registering/deregistering the provisioning web handlers.
12. Return value of API httpd_get_data() has been changed. Earlier it returned number of bytes remaining to be read, now it returns number of bytes read from the connection.
13. New APIs have been introduced to support an application to get the scan results and provision a network ==> prov_get_lock_scan_results(), prov_put_lock_scan_results(), prov_get_scan_count(), prov_get_scan_result(), prov_set_network()
14. The API lwip_perf_on() has been renamed to wifi_set_packet_retry_count()

**Bugfixes:**

1. Fix a crash in TLS code, because of AES context structure's pointer mismatch
2. psm-create tool could not be built on MinGW
3. CyaSSL multi-threaded configuration is not enabled
4. Cannot control LED property from Arrayent cloud
5. HTTP Client doesn't work when neither Content-Length nor Chunked encoding is received in the HTTP headers
6. HTTP Client: Fix URL parsing. A ':' in the query section was mistaken to be a delimiter for port number
7. HTTP Client: Allow absence of content length in non-chunked data
8. HTTP Server: Handle '0' return value of send() appropriately
9. Fix a bug in JSON v2 which was causing errors if number of parsed tokens is greater than 127
10. Fix data corruption in SSP full duplex communication

# 6.3 WMSDK 3.0 Preview R1

## 6.3.1 SDK Features and Enhancements

1. **88MW300 B0 Support:** This release includes support for the 88MW300 B0 silicon. The release supports the module 88MW300 RD Board v2.0 and the development board AB-88MW30x v2.0.

2. **802.11d and region configurations**: After Jan 2015, FCC will not allow devices running 802.11d. Explicit country configurations should be performed on the Wi-Fi station to configure it to the US region. In light of the above, the following changes have been made
    a. The previous build configuration option of 11d has now been removed. APIs have instead been introduced for configuring the region settings
    b. The API wlan_set_country() has been introduced. This can be used to explicitly configure the country where the device should operate. These settings are applied to the uAP and station interface
    c. The API wlan_enable_11d() has been introduced. This can be used to enable the 802.11d behavior for the station interface, in countries where 802.11d is allowed. This configures the uAP interface with the most conservative settings
3. **Bonjour Conformance Updates:** This release updates the Bonjour (mDNS/SD) responder to be completely conformant with the Bonjour Conformance Testsuite.
4. **UDP Broadcast Packet Filtering**: This release includes support for UDP broadcast packet filtering within the Wi-Fi driver. On networks with lot of UDP broadcast traffic, this minimizes the pressure on the network pools by freeing buffer at the lower most layer in the stack.
5. **JSON parser v2**: A newer JSON parser implementation based on the JSMN open source project is now available. This version of JSON parser provides better support for the parsing of multi-level nested JSON objects and the parsing of JSON arrays.
6. **IPv6:** This release includes support for IPv6. A build configuration is available to enable this functionality. This does result in a larger firmware size.
    **a.** Support in the HTTP server to respond over IPv6
    **b.** Support in the Bonjour (MDNS/DNS-SD) module to parse IPv6 AAAA records
7. **Link Local Address:** This release includes support for assigning Link-Local (Auto IP) based IP Addresses, if a DHCP server is not present in the configure network. A build configuration option is available to enable this functionality.
8. **CyaSSL:** The following changes have been done to the CyaSSL module:
    **a.** CyaSSL version is upgraded to 3.1.0-commercial
    **b.** The CyaSSL license is now updated. Product manufacturers need not work on a separate license with wolfSSL if they wish to work with CyaSSL, provided they use the binary image of the CyaSSL library that is available with the WMSDK. Please work with your Marvell Sales representative for more details about the same.
9. **HTTP Persistent Connections & Pipelining**: The HTTP Web server now supports HTTP persistent connections. This allows multiple simultaneous open connections with HTTP pipelining.
10. **Tools and Utilities:**
    **a. ping:** A ping utility is now available that can be run from the device. The call nw_utils_cli_init() can be used by applications to register the ping command line.
    **b. uarboot:** This host utility allows you to perform a UART boot of the MC200 with a firmware image that you have.
    **c. footprint.pl:** The footprint.pl utility has been significantly updated to analyse and understand the footprint of the generated firmware image much more effectively.
    **d. flashprog:** The flashprog utility has been updated to specifically address requirements on the manufacturing line.
11. **Firmware Upgrades:** This SDK release includes the following security enhancements to the firmware upgrades module:

**a.** **Upgrades over TLS:** The rfget module has been upgraded to the allow downloading firmware images over HTTPS (TLS). The API rfget_client_mode_update() has been updated to accept the TLS configuration.

b. **Secure Firmware Upgrades:** For customers who didn't wish to pull in the entire overhead of TLS, a secure firmware upgrades feature has also been included. This uses an AES/RSA mechanism to encrypt and sign a firmware image that can be used for performing the firmware upgrade. The secure_fw_upg_rsa2048_aes128_sha512() API can be used for this.

## 6.3.2 Release Notes

**Newly Supported Platforms:**

1) Marvell's 88MW300 RD Board V2.0 + AB-88MW30X V2.0 with MW300 B0 silicon

**Enhancements:**

1) 801.11d and Sub-band Changes: Please refer to API changes for more details
2) Include a sample application, cert_demo, that can be used for Wi-Fi certification
3) Include UDP broadcast packet filtering support in Wi-Fi driver
4) PSMv2: A new PSM module that provides the following new features:
    i) support for binary values
    ii) optimizes flash erase and write operations
    iii) preserves data across power failures during write
    iv) allows multiple simultaneous psm instances
    v) backward compatible API

    NOTE: This includes a change of the on-flash data format.
5) JSONv2: A newer JSON parser implementation based on the JSMN open source project is now available
6) Support is included to build using MinGW on Windows
7) Wi-Fi Certification: Fixes for complete Wi-Fi certification using
8) the approved ASD 1029 on 88W8801 chipsets
9) Network Enhancements:
    i) IPv6 is now supported. Support is available in HTTP Server as well as Bonjour
    ii) Link Local Address allocation is now supported
10) CyaSSL:
    i) CyaSSL licensing terms have changed. Anyone can now use the CyaSSL implementation. A consequence of this is that only binary-only release of CyaSSL is available.
11) HTTP Persistent Connections and pipelining support (multiple simultaneous open connections)
12) Tools/Utilities:
    i) footprint.pl utility has major updates for better analysis
    ii) flashprog utility is updated to handle manufacturing line usage better
13) Firmware Upgrades:

i) rfget module updated to also download upgrades over HTTPS

ii) A simpler secure upgrade mechanism without involving the TLS footprint overhead

14) Others:

i) Initial support for BT in the SDIO driver (88W8777 chipset)

ii) A new application hci_bridge is added for BT HCI commands

iii) Added basic support for critical error handling

**Bugfixes:**

1) Fix race condition in mdev aes driver which could result in silent failures in encrypt/decrypt operations when AES CTR is used in parallel with other AES modes
2) Multiple bug-fixes in mDNS module - both querier and responder
3) Fix for association failure with WEP OPEN mode AP in station mode
4) Fix for various issues in I2C driver for both master and slave mode of operation
5) Fix to calculate divider values based on XTAL value in USB driver
6) Fix higher packet loss issue on 8801 in PM2+IEEEPS mode
7) Fix for time not getting updated in RTC driver in PM4 mode
8) Incorrect JTAG frequency specified in the OpenOCD's configuration
9) TLS session initialization doesn't cleanup completely in case of a failure

**API Changes:**

1) New API mdns_set_txt_rec() for setting TXT record field of a particular mDNS service. SET_TXT_REC() macro has been removed
2) New API psm_delete() to delete a psm variable from psm
3) New API strdup() to duplicate the input string
4) API gpio_drv_set_cb() takes additional data pointer along with callback
5) New API gpio_drv_get_io_domain() to query voltage IO domain for given GPIO pin
6) New API pinmux_drv_get_gpio_func() to query GPIO pinmux function for given GPIO pin
7) New API dac_modify_default_config() to modify default DAC configuration parameters
8) New API wlan_get_current_signal_strength() to query current WLAN signal strength
9) API ba_pool_create() changed to not accept mutex_flag option, mutex is made mandatory
10) New APIs secure_upgrade(), secure_fw_upg_rsa2048_aes128_sha512() for secure firmware upgrades
11) New APIs wlan_get_sta_tx_rate_index(), wlan_set_sta_tx_rate_index() to configure station side TX rate
12) New API ssp_drv_deinit() to deinit SSP driver
13) New APIs net_get_if_ipv6_addr(), net_get_if_ipv6_pref_addr, ipv6_addr_state_to_desc(), ipv6_addr_type_to_desc() and net_ipv6stack_init() for IPv6 use cases
14) New API app_httpd_only_start() to start web server without associated web services
15) New APIs i2c_drv_set_callback() to set callback for notifications, i2c_drv_wait_till_inactivity() to check and wait for I2C module inactivity, i2c_drv_deinit() to deinit I2C driver
16) New API wlan_pm_cli_init() to register WLAN Power Mgmt CLI commands

17) Usage of the security type WLAN_SECURITY_WPA is no longer valid. Please use WLAN_SECURITY_WPA_WPA2_MIXED instead. Wi-Fi certification mandates this requirement
18) New API app_sta_save_network() has been added which saves the network configuration in PSM
19) Earlier some build options were supported from the wmsdk-x.y.z/ directory within the bundle. These are no longer supported. All 'make' commands should be run from the top of the bundle
20) A new configuration option (ENABLE_HTTPC_SECURE) has been included to enable TLS support in the HTTP Client module
21) New API httpd_parse_hdr_tags_with_cb() which executes a callback for each HTTP header line in the request
22) UDP broadcast packet filtering is included in the Wi-Fi driver. By default, only traffic on ports for DNS and DHCP is allowed. All other broadcast traffic is dropped at the lower most layer. The API netif_add_udp_broadcast_filter() can be used to add UDP port numbers that should NOT be blocked by the Wi-Fi driver
23) 802.11D is no longer a WMSDK build configuration option. Two new APIs introduced to replace this:
    a) wlan_set_country() to set country specific sub-band information instead of using 802.11D
    b) wlan_enable_11d() to enable 802.11D behavior as was available prior to this release

**Unsupported/Untested Features:**

1) USB Boot is not tested with this release.
2) Analog units (viz. ACOMP, ADC, DAC) and DMA are not tested with this release
3) Support for IAR toolchain/IDE is removed
4) Support for Eclipse IDE is not tested
5) WPA2 Enterprise is not tested with this release
6) This release is not tested with MC200 + 8801 and Lark development boards

# 6.4 WMSDK 3.0 Alpha Releases

## 6.4.1 SDK Features and Enhancements

### 6.4.1.1 88MW300 Support

WMSDK 3.0-Alpha 2 brings in support for the Marvell's 88MW300 Wi-Fi Microcontroller SoC. Support is provided for RD-88MW300-QFN-88 development board. Please refer to Section 1.1 for more details about the microcontroller.

### 6.4.1.2 PSM Version 2

A newly designed PSM that is resilient to intermittent power failures. Following are the features:

- One-level deep namespace for variable names.
- Binary values are supported.
- Specially designed to be run over flashes. It uses a special algorithm to avoid excessive erasing of flash sectors thus significantly prolonging the usable life of the flash device.

- Power resiliency. It uses reserved sectors to ensure that ever if power fails during write operation, existing variables are safe from corruption. This handling is transparent to user of PSM.
- Read only mode of operation. This allows user to store key value pairs permanently. This is useful to store factory data that is preserved across reset-to-factory scenarios.
- Multiple instances of PSM module can be run simultaneously to allow multiple separate PSM storage areas on the same device.

### 6.4.2 Release Notes

This release includes bug fixes, cleanups and enhancements through the various modules of the SDK. A brief list of new additions is as follows.

**Newly Supported Platforms:**

1. Marvell's 88MW300 Reference Design V1.0
2. Marvell's 88MW300 Development Board V1.0

**Enhancements:**

1. Support for XIP (Execute in place) on 88MW300
2. CyaSSL version upgraded to 2.9.4
3. Support for ping networking utility
4. Enhance uaputl command to support more options
5. Experimental support to build WMSDK on Windows using MinGW development environment.
6. Enhance flash driver for efficient erase operation
7. Support for Boot over UART using uartboot utility
8. Redesigned persistent storage manager with power resiliency.

**Bugfixes:**

1. Fix memory leak in EZConnect provisioning
2. Fix memory leak in DHCP server in case of failure

**API Changes:**

1. The AF_EVT_NORMAL_ADVANCED_SETTINGS and AF_EVT_INTERNAL_PROV_NW_SET have been combined into a single event AF_EVT_NW_SET
2. The API app_configure_network() and app_sta_start_by_network() have been modified so that the network configuration that is passed through the application need not be global
3. The implementation of app_p2p_stop() and app_configure_network() APIs have been changed. Now it requires additional 208 bytes of memory. So, the application should ensure that the caller thread's stack has sufficient memory space
4. The API wlan_remove_network() have been modified so that station network will not be removed if it is in connected state and uAP network will not be removed if it is in started state.
5. The API wlan_start_network() have been modified so that new uAP network will not be started if uAP is already in “started” state.
6. Block allocator API's are no longer supported from wm_os.h. API's from the header file block_alloc.h should be used directly.

**Known Issues:**

1. UART is not functional when CPU is using RC32M internal clock
2. Wi-Fi firmware and FTFS upgrades are not supported on 88MW300 Dev board as no space is available for redundant partitions
3. Slow Ping response in PM2 + IEEEPS mode
4. Code execution after wakeup from PM3 in XIP mode is not functional

# 6.5 WMSDK 2.13

## 6.5.1 SDK Features and Enhancements

## 6.5.2 88W8801 Support

WMSDK 2.13 brings in support for the Marvell's 88W8801 Wi-Fi SoC. Support is provided for RD-88MC200-W8801-V2 development board which includes a Marvell reference module with 88MC200 and 88W8801 SoC.

## 6.5.3 Third-Party Cloud

Two reference NodeJS-based cloud implementations were shipped up until WMSDK2.12:

- the long-polling cloud
- the Websockets cloud

Marvell has also partnered with 2 cloud-services companies – Arrayent and Xively. This release includes support for developing devices that will interact with their cloud service. Demo applications are provided that are available for a quick evaluation. For more information on them, please visit their respective web-sites – http://www.arrayent.com and http://www.xively.com. Please contact your Marvell sales representative for additional information.

## 6.5.4 New Provisioning Mechanisms

Up until WMSDK2.12 the following network provisioning mechanisms were supported:

- Marvell Provisioning (micro-AP based provisioning)
- WPS Provisioning (Wi-Fi Protected Setup)

This release includes the support for the following new provisioning method:

### 6.5.4.1 EZConnect Provisioning

While it is required to connect to the device's micro-AP network to start network provisioning in case of Marvell Provisioning, this is not required for EZConnect provisioning. In EZConnect provisioning, a native smartphone application securely broadcasts the network credentials over the currently connected network. The WMSDK uses a patent-pending technology to intelligently read such broadcast messages and retrieve the network credentials.

This simplifies the provisioning mechanism and makes it truly a one-touch provisioning experience. This provisioning method is supported for the following Wi-Fi chipsets:

- 88W8801: supported in the default Wi-Fi firmware
- SD8782: supported with ezconnect version of the Wi-Fi firmware

- SD8787: not supported

### 6.5.5 Smart Phone Apps for Provisioning

WMSDK2.13 includes fully-functional apps for the following provisioning methods:

- Marvell Provisioning (Android and iOS apps)
- EZConnect Provisioning (Android and iOS apps)

These apps can be used to experience the workflow for both the above provisioning mechanisms. Feel free to easily customize these apps for your brand and color theme. Or use this as a reference code to start building your own apps.

### 6.5.6 Mac OS-X as Development Host

Mac OS-X is now natively supported as a development host.

### 6.5.7 Sample Applications

1. Demo applications *arrayent_demo* and *xively_demo* are included to demonstrate the usage of cloud communication APIs for communicating with third-party cloud services.

### 6.5.8 Release Notes

**Enhancements:**

1. Support for RD-88MC200-W8801-V2 boards
2. Third-Party Cloud Support:
    a. Support for Xively cloud
    b. Support for Arrayent cloud
3. EZConnect provisioning: significantly reduces the provisioning UI complexity
4. Include Phone (Android/iOS) apps and source code for:
    c. Marvell Provisioning
    d. EZConnect Provisioning
5. Support for the GNU ARM Toolchain
6. Support for Mac OS-X as Development Host
7. Support for Mixed WPA/WPA2 mode security in Marvell Provisioning
8. Wi-Fi:
    a. Support to limit the number of clients that can connect to themicro-AP interface
    b. Support for external authenticator/supplicant while working with 8801
    e. Include support for including vendor-specific custom IEs inmicro-AP beacons
    f. Connect to the network with best signal strength, whenmultiple networks with the same name are detected
    g. Sniffer-mode Support
9. footprint.pl:
    a. Sort symbols in descending order of size
    b. Provide option to shows symbols based on section instead of library
    c. Include heap size in the output

10. Marvell Provisioning modified to also support on-demand scan(previously only periodic scan was supported)
11. FreeRTOS: Include support for displaying runtime CPU statistics
12. Provisioning: Include support for 'wildcard' security type in the /sys/network API, for use when the client isn't aware of thesecurity of the target Access Point

**Bugfixes:**

1. Fix erroneous buffer write after timeout in I2C Slave mode
2. In PM2, current consumption changes significantly with temperature (Vfl is now in deep off)
3. Wi-Fi scan fails after exit from Wi-Fi PDN
4. Fix memory leak in websockets/long-polling cloud code in case of error conditions
5. Cannot connect to hidden SSID with 802.11d is enabled

**API Changes:**

1. New API wlan_sniffer_start/stop for starting and stopping the sniffer mode
2. os_init() is now a weak function. This can be overridden byapplications' that wish to use no OS
3. New API os_get_runtime_stats() can be used for getting runtimestatistics of various threads
4. New API in the application framework:app_ezconnect_provisioning_start/stop for starting and stopping theEZConnect provisioning. Lower level APIs prov_ezconnect_start/stopare also exposed by the provisioning module for EZConnectprovisioning. APIs prov_ezconn_set_device_key /prov_ezconn_unset_device_key for setting and unsetting encryptionkey during EZConnect provisioning
5. New APIs app_mdns_add_service/app_mdns_remove_service to add orremove mDNS/SD services. While the earlier APIapp_mdns_add_service_iface provides the entire set of services to be advertised in a single go, the new APIs allow for incrementally adding to or removing from the existing set of mDNS/SD services to be announced
6. New APIs i2c_drv_set_clkcnt for I2C clock configuration
7. New API httpd_send_response_301() for sending an HTTP 301 responseto the client
8. New API prov_set_scan_policy() that can switch the scanning policyin Marvell provisioning between periodic and on-demand
9. New APIs wlan_set_11d(), wlan_set_region_code() andwlan_set_domain_params() for 802.11d support in the station mode
10. New security type WLAN_SECURITY_WILDCARD for cases where the useris not aware of the security type of the target network
11. New APIs wlan_{set|get|clear}_mgmt_ie() for managing the IEs(Information Elements) in the micro-AP's beacons
12. New APIs wlan_get_sta_tx_power() and wlan_set_sta_tx_power() tocontrol the transmit power of the station interface

# 6.6 WMSDK 2.12

## 6.6.1 SDK Features and Enhancements

### 6.6.1.1 WebSocket

The WebSocket protocol provides a full-duplex communication over single TCP communication. The WebSocket standard is obeyed by most Web Servers. Thus an existing HTTP connection can be upgraded to a WebSocket connection facilitating full-duplex communication.

In HTTP-based cloud communication scenarios the device usually posts data to the cloud and looks for any commands from the cloud. HTTP being a command-response protocol, the device has to periodically poll the WebServer for any commands.

The WebSocket protocol makes this transaction more efficient. Once the HTTP cloud communication is upgraded to use WebSocket, the connection can stay intact at both the ends. Either end can send data on the connection (the device when some event occurs on the device, or the web server if there are some commands to be sent to the device), whenever any event occurs. The resultant transaction is not only efficient, but information is updated at the other end almost instantaneously.

### 6.6.1.2 USB Host CDC Driver

WMSDK 2.12 implements USB-CDC support while in the USB-Host mode. The USB *Communication Device Class* (CDC) enables the 88MC200 to communicate with asynchronous usb-serial devices.

A sample application usb_host_cdc is now part of the WMSDK bundle. This application implements the USB-CDC driver. When the application is loaded, if a usb-serial device is interfaced with the USB OTG port of 88MC200, this will be detected as a virtual COM port. The application transmits a pre-determined string over this interface. Any data received over this interface is printed to the serial console.

### 6.6.1.3 P2P in Application Framework

The P2P module has undergone significant restructuring in WMSDK2.12. The events that are generated by the P2P module have been better aligned with the events for the station and micro-AP interface.

The WMSDK 2.12 includes support for making P2P calls through the application framework. The calls app_p2p_start() and app_p2p_stop() are used to enable / disable the P2P module. The call app_p2p_scan() is introduced to scan for available P2P devices in the vicinity. A P2P session can be established using app_p2p_session_start(), while an existing connection can be torn down with app_p2p_disconnect().

The sample application p2p_demo is updated to implement the same functionality using the application framework's P2P APIs. The earlier p2p_demo application has now been moved to raw_p2p_demo.

### 6.6.1.4 Other Enhancements

The following are some of the other enhancements in WMSDK2.12:

- Support for the external flash on the High-Flying module HF LPB-200. The flash id 2 is used in the layout file as an identifier for this flash.
- **UART Driver:**
    - Support for configuring partiy and stop-bits
    - Support for configuring DMA read-block size
    - Software Flow Control support
- **ADC Driver:** DMA support for the ADC driver

- **WPA2 Enterprise + TLS:** Enable normal TLS functionality when WPA2-Enterprise is used for Wi-Fi association.
- Support for easily starting open-mode micro-AP network using app_uap_start().
- Support for limiting the number of clients connected to the micro-AP network.
- The code for ctaocrypt is no longer a part of the libcyassl.a. A separate library libctaocrypt.a is created for this code This change is useful for users of the Overlays feature. Prior to WMSDK 2.12, CyaSSL and WPS couldn't belong to two mutually exclusive parts of the overlays because WPS used some of the crypto code. This splitting of libraries enables WPS and CyaSSL to reside in two mutually exclusive overlays, thus providing more RAM savings.
- The wm_demo application's cloud interface is modified to now support two types of cloud communication strategies:
    - Long-polling
    - WebSocket

## 6.6.2 Sample Applications

1. The application that is used for FCC certification is now available within the wmsdk bundle itself.
2. The p2p_demo application now uses the P2P APIs exposed by the application framework. The earlier p2p_demo application is now renamed to raw_p2p_demo. The purpose of the raw_p2p_demo is to ensure that existing users of P2P understand the restructuring of the P2P module.
3. An application usb_host_cdc is now included to demonstrate the USB Host CDC feature.

## 6.6.3 Release Notes

**Enhancements:**

1. USB Host CDC Support
2. WebSockets Support
3. Support for P2P in application framework
4. Support for the external flash of the High-Flying module HF LPB-200. The flash id 2 should be used in the layout files for referring to this flash
5. Driver Enhancements:
    - UART: Support for configuring Parity, Stop-bits and DMA read block size. Support for Software based flow-control
    - ADC: DMA support for ADC driver

**Bugfixes:**

1. When WPA2 Enterprise is enabled, the normal TLS functionality was not available.
2. HTTP Client can now differentiate between socket time out, socket errors and peer disconnected scenarios.
3. Fix an issue with pvPortRealloc() / os_mem_realloc(), where realloc tries to read extra bytes at the end of the pool which may be in a non-existent address space.

**API Changes:**

1. New API os_total_ticks_get() is added which returns a 64 bit OS tick counter value. The current applications which use os_ticks_get() for delay measurement, are vulnerable to wrap around problems. The internal tick counter value returned by os_ticks_get() wraps around after approximately 49 days. This new API is safer to use as it is 64 bit.
2. New member HDR_ADD_CONTENT_TYPE_JSON is now included in http_hdr_field_sel_t.
3. If a NULL passphrase is passed to app_uap_start* functions the micro-AP network comes up in Open mode.
4. Earlier, two variants of the data type tls_init_config_t were used, one when WPA2-Enterprise was enabled, and the other otherwise. Now the same structure is used in both the cases.
5. The file wmutils.h is now renamed to wm_utils.h.
6. New APIs adc_drv_calib(), adc_modify_default_config() and adc_get_config() are included for calibrating the ADC, modifying the ADC configuration and retrieving the ADC configuration respectively.
7. The values of the UAP Power Save enum are updated to be exclusive from those of the Station Power Save enum.
8. New API http_getsockopt() to get socket options related to the HTTP Client's socket.
9. HTTP Client APIs now return two additional error values:
    a. WM_E_HTTPC_SOCKET_SHUTDOWN: if the peer end has closed the connection
    b. WM_E_HTTPC_SOCKET_ERROR: for any socket error. The corresponding error should be looked up in 'errno'

    This now solves the problem of not being able to distinguish between a peer-closed and timed-out for HTTPC Client connections.
10. New API uart_drv_dma_rd_blk_size() is included to configure the UART DMA's read block length.
11. New API uart_drv_set_opts() to control the parity and stop-bits option of the UART driver.
12. New APIs for websockets are introduced:
    a. ws_upgrade_socket() to upgrade the HTTP request to websocket
    b. ws_frame_send() to send a frame over WebSocket
    c. ws_frame_recv() to received a frame over WebSocketn
    d. ws_close_ctx() to close the Websocket context
13. Members server_cert and server_cert_size are now removed from the tls_init_config_t data type. Server certificate validation can be performed using the ca_cert and ca_cert_size members of the same data type.
14. New API app_ctrl_event_name() that converts an application framework event id to a string representation.
15. New API adc_drv_get_samples() introduced to read n samples of analog input over DMA.
16. usb_host_drv_*() APIs now enable USB Host CDC read/writes
17. enum wlan_mode is no longer used. enum wlan_bss_role is used to indicate the role of the interface (station and micro-AP). And enum wlan_bss_type is used to indicate whether this is WiFi-Direct interface or the standard interface. WLAN_MODE_INFRASTRUCTURE is changed to WLAN_BSS_ROLE_STA and WLAN_MODE_UAP is changed to WLAN_BSS_ROLE_UAP. WLAN_MODE_P2P* are

no longer used. This information is provided using the enum wlan_bss_type as mentioned above.

18. WLAN_REASON_P2P_* events are now deprecated. The events for the station and micro-AP themselves are used to indicate success or failure of the P2P interface.
19. The libctaocrypt.a is now a separate library instead of being a part of the libcyassl.a. The rest of the licensing requirements stay the same.
20. New APIs app_p2p_start(), app_p2p_stop(), app_p2p_disconnect(), app_p2p_session_start() and app_p2p_scan() are introduced to use P2P from the application framework.
21. The wlan_connect API now cannot be used with a NULL profile name.
22. New API added to control the number of uAP interface connections wlan_get_uap_supported_max_clients() and wlan_get_uap_max_clients().
23. uart_drv_set_opts() API is modified to include another parameter to control software based flow control.

# 6.7 WMSDK 2.11

This section documents the changes in WMSDK 2.11 over 2.10

## 6.7.1 SDK Features and Enhancements

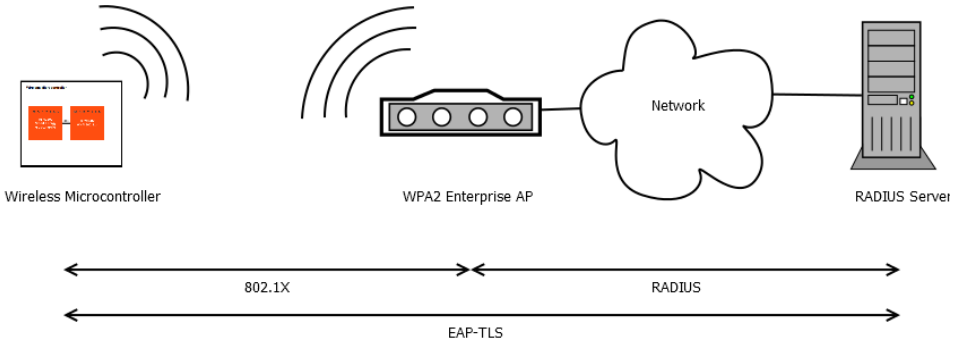
### 6.7.1.1 WPA2 Enterprise/802.1X Station

WMSDK 2.11 includes support for WPA2-Enterprise/802.1X authentication with Access Points using EAP-TLS mechanism.

The WPA2 Enterprise authentication uses 802.1X. The standard allows for encapsulation of EAP (Extensible Authentication Protocol) over LAN (EAPOL). This type of authentication is typically used in commercial and enterprise deployments. This provides stronger control over the network at the cost of additional setup complexity. An 802.1X RADIUS server, is required to be setup to manage the authentication.

The 802.1X standard can be used to encapsulate a variety of authentication protocols. WMSDK 2.11 supports authentication using EAP-TLS (Transport Layer Security). With EAP-TLS the PKI (Public Key Infrastructure) is used to authenticate clients on the network on the basis of client certificates.

A typical deployment of WPA2 Enterprise with EAP-TLS is as shown below:




A RADIUS Server (Remote Access Dial In User Service) is used to provide centralized authentication. An AP that supports 802.1X is configured to use this server. The Wireless

Microcontroller device is pre-programmed with unique certificates. The RADIUS server has an ability to validate genuine certificates. It can also be configured to allow/deny access on a per-certificate basis.

Although the WPA2 Enterprise setup is complex, it provides the following advantages:

- No pre-shared secret and thus, no threat of leaking the secret
- Centralized control over who joins the network, thus better security. Any certificate can be denied or allowed access
- Simplifies network provisioning when a large number of Wireless Microcontroller devices should be connected to the network

Please refer to the Application Note for WPA2 Enterprise for build and setup instructions.

#### 6.7.1.2 USB Host Support

WMSDK 2.11 supports USB Host functionality. With this, a USB endpoint (like a USB camera) can be directly connected to the Wireless Microcontroller and can be operated using an appropriate driver.

WMSDK 2.11 also includes support for the USB Video Class (UVC) 1.1. This driver enables connectivity with devices capable of streaming video, like webcams, digital camcorders etc. The UVC driver has been tested with the following hardware:

- Microsoft LifeCam Cinema
- Microsoft LifeCam VX-2000

Please note that the USB Host driver and the USB Video Class driver are available as a binary-only image. Please talk to a Marvell sales representative to get access to the USB Host driver.

#### 6.7.1.3 HTTPD WSGI Restructuring

The HTTPD WSGI Interface has been restructured, to make it simpler to implement and easier to understand. Care has been taken to make sure that the simplicity offered does not inhibit the advanced users from implementing advanced WSGI handlers.

A new API *httpd_register_wsgi_handler()* is introduced that registers the WSGI handlers. A simple HTTP GET handler can call *httpd_send_response()* to respond to the HTTP GET request. A simple HTTP POST handler will read data using *httpd_get_data()/http_get_data_json()* and send the response with *httpd_send_response()*. Please refer to the WMSDK Reference Manual for more information. The reference manual also highlights the callgraph of the send and receive APIs. This enables advanced users to choose the complexity level at which they wish to interact with the HTTPD WSGI functions, such that it offers them the most flexibility.

#### 6.7.1.4 Wi-Fi Driver Enhancements

The Wi-Fi driver has been updated as follows:

- Overall cleanups and refactoring.
- Support for association with stealth-mode (hidden SSID) APs.
- Support for non-HT (non High Throughput) association with 802.11n APs. When 802.11n APs with a security such as WEP, WPA-TKIP or WPA-AES is specified, earlier the association request was dropped. Now, the association goes through, but the happens in a non-HT mode. This facilitates interoperability with larger AP configurations without violating the 802.11n specifications.
- Until WMSDK 2.10, the micro-AP interface could be configured only with the WPA2 security type. Now the micro-AP interface can also be configured in Open (no-security) mode.

- If an AP lists multiple credentials in response to a successful WPS session, then this situation is now handled better in the WPS module. Preference is given to the network with the highest security. Preference is also given to a network in the 2.4GHz band.
- The Wi-Fi Connection Manager was not handling disconnect events properly in the *connecting* state. This behavior has been fixed.
- Support is included to configure NULL packet interval in IEEEPS. Some Access Points drop a Wi-Fi station if they do not see NULL packets at regular intervals from these stations. Until WMSDK2.10, by default the NULL packet interval was 30 seconds. The NULL packet was disabled if long listen interval was set. With WMSDK2.11, the API *wlan_configure_null_pkt_interval()* can be used to enable, disable or configure the NULL packet interval. With WMSDK2.11 the default interval during IEEEPS is 30 seconds.
- Support for configuration of region/domain codes, required for 802.11d. This ensures that the device adheres to country-specific wireless communication regulations.

#### 6.7.1.5 Other Enhancements

WMSDK 2.11 includes enhancements throughout the modules, some of these include:

- Support for revision A1 of the 88MC200 chip. Support for Lark Mini Development Board v3, that hosts the A1 revision of 88MC200, is also included.
- **DHCP Server:** Support for MAC-IP address affinity. With this feature, the DHCP server remembers the IP Address that is assigned to a given device, and reassigns the same IP address to that device on subsequent renewal requests. The MAC-IP address association is maintained for the first 4 devices only.
- **TLS:**
  - The CyaSSL library has been upgraded to version 2.7.0 from version 2.5.0.
  - TLS servers can now optionally validate the client certificates of the Wireless Microcontroller.
- **mDNS/DNS-SD:**
  - The mDNS/DNS-SD querier has a cache size of 2. This implies that it can monitor 2 services actively. Since the cache size is quite small, this can be restrictive when the network contains many devices that have a matching mDNS/DNS-SD service. With WMSDK 2.11, applications have a finer-grained control on the services that are cached. The mDNS/DNS-SD querier will cache entries only if the application's query callback returns *WM_SUCCESS* in response to an *MDNS_DISCOVERED* and *MDNS_UPDATED* events. If the query callback returns anything else, that service is not cached.
  - WMSDK 2.11 includes support for sub-type based service discovery in mDNS/DNS-SD querier. With this support, the mDNS/DNS-SD querier can discover services based on subtype (like *_marvell._sub._http._tcp*). Earlier, the querier could only discover services based on the type (like *_http._tcp*). This allows for fine-tuned monitoring of services of interest.
- **FTFS (Filesystem):** FTFS filesthat are readover HTTP are now read in larger chunks thus improving file serving performance.
- **CLI (Command Line Interface):** The CLIs of all the modules is no longer a compilation option. Any CLI can now be used by calling the corresponding *module_cli_init()* function from the application.
- TCP transmit performance improvements especially when Wi-Fi Power Save mode is enabled.

- Semaphore debugging support is included. This provides a CLI that can be used to query the values of all the semaphores of the system. Information is also provided about any threads that maybe waiting on any of these semaphores.

## 6.7.2 Sample Applications

The sample applications that are distributed with the WMSDK have been restructured for better understanding of the workflow. Please review the sample apps tarball.

The *boards/* directory is no longer part of the applications tarball. For ease of access, this has been maintained within the WMSDK itself.

## 6.7.3 Release Notes

**Enhancements:**

1. Support for association with WPA2-Enterprise AP using EAP-TLS authentication
2. USB Driver Support:
    - USB Host driver
    - USB Video class driver
3. Wi-Fi Driver
    - Cleanups and refactoring
    - Support for association with stealth mode AP
    - Support for non-HT association with 11n AP when the security specified is WEP, WPA-TKIP or WPA-AES
    - Support for Open security in micro-AP mode
    - Support for handling multiple credentials returned after a successful WPS session
    - Handle disconnect events in 'connecting' state
    - Support for configuration of null packet interval in IEEEPS mode
    - Support for configuration of region/domain codes required for 802.11d
4. mDNS/DNS-SD
    - Finer grained application control on the services to monitor using mDNS querier
    - Subtype based discovery support
5. uaputl CLI: get/set channel, get/set tx power, get/set data rates
6. Support for revision A1 of the 88MC200 chip
7. TLS
    - Support for server-side verification of client certificates
    - CyaSSL library upgraded to version 2.7.0 from 2.5.0
8. A New Provisioning Protocol is now implemented that is more robust and handles most common scenarios
9. DHCP Server: Support for MAC-IP Address affinity. For the first few devices, the IP Addresses assigned to them will be remembered and the same address will be assigned for lease renewal
10. HTTPD WSGI Refactoring
    - Updated to make it much easier for applications to write WSGI handlers
    - Offer greater flexibility to advanced users

11. CLI: Removed WMSDK configuration options for all CLIs, provide the capability to the application to decide what should be initialized by directly calling the corresponding module_cli_init() function
12. FTFS: More efficient file serving
13. Debugging: Semaphore debugging support

**Bugfixes:**

1. The httpc client lowlevel read/write would not work when 0 or more http transactions are complete. After this fix, low level read/write can be done immediately after http_open_session()
2. PM4 + DS wakeup is not reliable
3. In case of WPS Provisioning -> Reset to Provisioning -> WPS Provisioning, the second WPS provisioning does not return the correct credentials
4. Race condition in UART driver leads to data lock ups after large data transfer
5. Dynamic Frequency Selection (DFS) doesn't work correctly in 5GHz mode. Probe requests were being sent on channels where radar interference was detected
6. Incorrect security type reported in output of wlan-scan
7. Power Manager framework with PM3 hangs after a certain data transfer is performed
8. Get/Set Antenna Diversity does not work correctly
9. With some APs, entering incorrect passphrase causes (i) infinite looping between connecting and link-loss events or (ii) the wlcmgr state machine to stall
10. Sometimes DHCP client stalls indefinitely while trying to acquire the DHCP address
11. BSS section not initialized correctly in overlays
12. Memory leak after i2c_close
13. UART communication with baud rates greater than 115200 does not work when 88MC200 is running at 32MHz frequency
14. WPS Interoperability issue with some APs

**API Changes:**

1. The HTTPD WSGI handler interface has undergone a major change. The older APIs HTTPD_WSGI_CALL(), pattern_http_common_handler() are removed. The following new APIs are introduced:
    a. httpd_register_wsgi_handler()/unregister_wsgi_handler() - for registering WSGI handlers
    b. httpd_send_response() to send the complete response of a WSGI handler
    c. httpd_get_data()/httpd_get_data_json() to read data received in a POST request
2. Support for the EEPROM driver, I2C-based GPIO expander and the I2C-based UART is removed
3. The application framework event AF_EVT_NORMAL_AUTH_FAILED is no longer available in the application framework. Instead the single event AF_EVT_NORMAL_CONNECT_FAILED is used to report all the connection failures types. A new data type app_conn_failure_reason_t is passed as the 'data' for this event that describes the exact reason for failure (like authentication failed, network not found etc.)

4. In case of AF_EVT_NORMAL_CONNECT_FAILED event from the application framework, returning non-zero value from event handler causes auto reconnection to stop
5. The command line interface of various modules is no longer a build-time configuration option. Applications can determine which CLIs to include by calling the corresponding module_cli_init() function. New APIs wlan_cli_init(), wlan_enhanced_cli_init(), wlan_iw_cli_init(), wlan_uaputl_cli_init(), pm_cli_init(), wmtime_cli_init() and ftfs_cli_init() are introduced for this purpose
6. In mDNS/DNS-SD, the DEFINE_TXT_REC API is removed. Users should directly pass the string buffer to SET_TXT_REC
7. AES CCM is disabled by default in CyaSSL configuration
8. app_httpd_with_fs_start() is modified to include an additional parameter. This is an out parameter that returns the filesystem that is initialized by the HTTPD server
9. The first parameter of http_lowlevel_write() is changed from 'http_session_t *' to 'http_session_t'
10. The behaviour of mdns_query_cb() is modified to provide applications with a finer control of what entries the mDNS should cache. If the query callback returns WM_SUCCESS in response to a MDNS_DISCOVERED or MDNS_UPDATED events, then the service is deemed to be of interest to the query callback and hence is cached. If any value other than WM_SUCCESS is returned, then the service is not cached/monitored
11. TLS_USE_CLIENT_CERT is introduced in tls_session_flags. When enabled, the client's certificate is passed to the server. This enables server to perform client certificate validation. Members are also introduced in tls_init_config_t to make a note of the client certificate and key that can then be passed to the server
12. mdns_query_monitor() now includes support for sub-type based service discovery as well. The fqst parameter can now include a sub-type based service as well
13. New API app_sta_stop() is introduced to stop the station interface. This disconnects the station and makes it idle. The station can be started again using app_sta_start()
14. New API app_sta_start_by_network() is introduced to start a station on the network that is provided as a parameter instead of the one in the PSM
15. New API app_ctrl_notify_application_event() is introduced. This allows the application to send events to itself. This is useful when the application wishes to perform some action in the context of the app_framework's thread itself
16. New API app_uap_start_with_dhcp() is introduced to start the micro-AP interface, and starting the DHCP Server on this interface
17. New APIs are introduced for the USB Host driver. The APIs usb_host_drv_init(), usb_host_drv_open(), usb_host_drv_start(), usb_host_drv_stop(), usb_host_drv_read(), usb_host_drv_write(), usb_host_drv_close() are introduced to work with the USB Host driver
18. New APIs for the I2C driver are introduced:
    a. i2c_drv_get_status_bitmap() - Get status of the I2C Controller
    b. i2c_drv_enable() - Enable the I2C port
    c. i2c_drv_disable() - Disable the I2C port
    d. i2c_drv_timeout() - Set timeout for I2C read/write for I2C master
19. New API os_mem_calloc() introduced to allocate and clear the allocated heap memory in a single call
20. New API wlan_configure_null_pkt_interval() is introduced to configure the NULL packet interval when the Wi-Fi station interface is in the IEEEPS mode

21. New APIs wlan_get_region_code(), wlan_set_region_code() and wlan_uap_set_domain_params() are introduced for control over the region and country code settings as per 802.11d
22. New API wlan_set_mac_addr() is introduced to optionally set the MAC Address of the Wi-Fi interface at runtime
23. New API wlan_set_cal_data() is introduced to optionally set the calibration data of the Wi-Fi interface at runtime
24. New API dhcp_server_lease_timeout() is introduced to configure the lease duration that is offered by the DHCP server
25. New API http_setsockopt() is introduced to set socket options for the socket associated with the HTTP client
26. New API cli_stop() is introduced to stop the CLI module

# 6.8 WMSDK 2.10

This section describes the changes in WMSDK 2.10 over WMSDK 2.9.

## 6.8.1 SDK Features and Enhancements

### 6.8.1.1 Power Management Framework Robustness

Changes for the robustness of the power manager framework are a major part of the 2.10 release. This is no longer a compilation option. Support for power manager framework is available by default in the SDK. Applications can call *pm_mcu_cfg* and *pm_ieeeps_hs_cfg* for using the power manager framework. Some of the major changes related to this include:

- **Optimized thread/timer wakeups:** The power manager framework ensures that MC200 comes out of the power save mode to serve any scheduled threads and timers in the system. This release includes optimizations to all the network timers and WMSDK threads such that they wakeup only when they have to. This ensures that the MC200 continues to stay in PS state for the longest possible time.
- **IEEEPS/Deep Sleep State Machines:** This release includes refactoring of the Wi-Fi power-save state machines for the IEEEPS and Deep Sleep mode. The refactoring makes it easier to handle all the corner cases with these power-save modes. This also provides robust handling of the IEEEPS state after the Wi-Fi card has woken up from the Deep Sleep mode.
- **Provisioning changes:** The application framework now provides the application with an additional event from the web application. This event indicates to the application that the web application has reliably determined that the provisioning of the device was successful. With the introduction of this event, the application can now keep the micro-AP interface up for a time duration long enough for the web application to determine if provisioning is successful. Once this is done, the application can enter the PM2+IEEEPS power-save modes using the power manager framework APIs.

### 6.8.1.2 WMSDK Robustness

Some of the other robustness fixes that have gone in the release include:

- **Brown-out detection and reset:** The default WMSDK configuration now includes brown-out detection support. Upon detection of such a condition a reset is generated. This ensures that the system does not enter an indeterminate state. Brown-out detection is enabled on the input voltage to the module (VBAT) with a threshold of 1.85V. A reset is generated if the voltage drops below 1.85V.

- **Redundant Partition Tables:** All components of the flash layout that can be upgraded have active-passive redundancy apart from the partition layout. This release includes redundancy for the partition table as well. Any writes to the partition table write to the passive partition table and mark it as active for the next bootup.
- **PSM enhancements:** The PSM module is enhanced to differentiate between the corruption of a metadata partition or a data partition. The PSM also includes support to erase a single data partition. With these two mechanisms, the users of the PSM can detect and erase any corrupted PSM partitions. The application framework is modified to ensure that it cleanly handles any state where the psm data partitions could be corrupted.

#### 6.8.1.3 Other Enhancements

- Up until this release the WMSDK supported over-the-air upgrades for the 88MC200firmware and the filesystem images. This release introduces the support for upgrading the Wi-Fi Firmware over the air
- By default the 32KHz external crystal is now used as the input to the RTC, instead of the RC32K
- CyaSSL library is upgraded to version 2.5.0
- AES CCM and CTR mode support
- Enhancements to the device driver include:
    - Blocking read support from UART
    - DMA support for transmit/receive of data over UART
    - 10-bit address mode support for I2C
    - User configurable receive buffers for I2C and UART
- Faster association transactions with WPS and Wi-Fi Direct
- /sys/perf HTTP API exposed by the application framework to quickly get a sense of network performance

### 6.8.2 Release Notes

**Enhancements**

1. CyaSSL library upgraded to version 2.5.0
2. AES CCM and CTR mode support
3. Power Manager Framework (PMF) robustness. This is no longer a compilation option. Applications call pm_mcu_cfg/pm_ieeeps_hs_cfg to use PMF
    - re-factoring of the IEEEPS and deepsleep state machines
    - timer and thread wakeup optimizations to ensure that the processor stays in PS modes for longer durations
    - provisioning changes to facilitate connection status indicator when PMF is used
    - PM4+deepsleep and PM2+IEEEPS power states are reliably supported
4. PSM: Support for erasing single partition
5. Blocking UART read support
6. Faster ram loading support via OpenOCD
7. Brown-out detection and reset support

8. Support for Wi-Fi Firmware upgrades
9. Support for redundant partition tables
10. Optimize association time with WPS and Wi-Fi-Direct
11. Driver Changes:
    - Receive/Transmit DMA Support for UART
    - I2C 10-bit address mode support
    - User configurable receive buffer for I2C
12. /sys/perf HTTP API for quickly getting a sense of network performance
13. Seperate linker section for peripheral DMA I/O buffers

**API Changes**

1. New board API board_32k_osc that informs whether a 32KHz oscillator is available and functioning on the board
2. http_add_header() takes additional parameter http_req_t
3. The wmtime module now uses standard "struct tm". All the members are of the type int
4. New Application framework event: AF_EVT_PROV_CLIENT_DONE: an indication that the http provisioningclient has acknowledged successful provisioning
5. New Application framework API app_sys_register_perf_handler for applications that wish to register performance measurement HTTP handlers
6. pm_mcu_state_hs, wlan_ieeeps_on, wlan_ieeeps_off are deprecated since they are potentially unsafe to use. Prefer to use pm_ieeeps_hs_cfg for configuration of IEEEPS and hostsleep.
7. WAKE_ON_BROADCAST wakeup condition is modified. It is now split into two separate wakeup conditions:
    a. WAKE_ON_ALL_BROADCAST: Wake the host processor on any broadcast request
    b. WAKE_ON_ARP_BROADCAST: Wake the host processor only when an ARP request for the device's IP Address is received
8. wakelock_get/put calls now take an additional string parameter. This string is used to match corresponding get/put requests for debugging purposes
9. New APIs for AES aes_drv_ccm_setkey, aes_drv_ccm_encrypt, aes_drv_ccm_decrypt for AES CCM based operations
10. New API uart_drv_blocking_read() that configures the UART such that reads will be blocking
11. New API uart_drv_xfer_mode to enable DMA for UART data transfers
12. New APIs uart_drv_rxbuf_size, i2c_drv_rxbuf_size to change the rx ring buffer size of UART/I2C
13. New PSM API psm_get_partition_id and psm_erase_partition for selectively erasing partitions

**Bugfixes**

Following bugs were fixed:

1. After scanning, the first association attempt may fail

2. Device doesn't get DHCP address after link-loss (Power Manager Framework)
3. Sometimes MC200 does not wake from PM2/4 even after wakeup is received
4. Device doesn't associate with WEP mode AP, if previously associated with WPA2 mode AP
5. Association doesn't work with Open mode AP
6. System stops entering PM1 opportunistically when pm_mcu_cfg is called with PM2/3
7. WPS is not robust when link is lost with the AP during initial handshake
8. ssp_drv_pio_write sometimes corrupts SPI read data in full-duplex mode
9. Hardfault during association when 11n is disabled

### 6.8.3 Upgrading from older releases

Please take note of the following changes:

The support for redundant partition tables introduced in the 2.10 release warrants a change in the boot2 and the partition layout.

Please update the flash layout on your boards as follows:

a. Goto *wmsdk/tools/OpenOCD*
b. Execute

```
./flashprog.sh --new-layout layout.txt
```

c. Flash boot2, firmware and filesystem and new wireless firmware images generated with the WMSDK 2.10 release

Note: All settings stored in psm will be lost..

## 6.9 WMSDK 2.9

This section describes the changes in WMSDK 2.9 over WMSDK 2.8

### 6.9.1 SDK Features and Enhancements

#### 6.9.1.1 Power Conservation: Power Management Framework

This release introduces the Power Manager Framework as an experimental feature. The power manager framework provides an effective way for power management for multi-threaded systems. It provides an ability to effectively enter/exit the 88MC200and Wi-Fi power save states.

**88MC200 Power Save**: Applications can use the API *pm_mc200_cfg()* to configure the power manager framework for the 88MC200. Once enabled, the power manager framework will intelligently detect processor idle times and switch to the PM2/PM3 modes. The framework will time a wakeup from this power save mode such that the processor is woken up when the next thread is due for execution. This relieves the applications from the burden of power management and focus on the application logic.

A wakelock mechanism is provided with which applications can keep the processor awake for short durations of time. Applications may use this mechanism while they are waiting for inputs from external hardware.

The power manager framework provides an API to ensure that the processor woken up on certain Wi-Fi events. Please read on for more information.

**Wi-FiPower Save**: Applications can use the API *pm_ieeeps_cfg()* to configure the Wi-Fi card's IEEEPS power save mechanisms. Once enabled the power manager framework will put the card into IEEEPS power save.

The Wi-Fi card is also configured to wake the processor up on certain Wi-Fi events. This provides a coupling between the 88MC200 and Wi-Fi power save mechanisms. This provides a powerful yet easy-to-use API for application developers to attain aggressive power conservation.

Please refer to the Section 6.9.2 for details about the enhancements made to *wm_demo*.

#### 6.9.1.2 Footprint: Overlays

Overlays provide a mechanism for mapping multiple mutually exclusive modules of code within the same region of RAM. This provides better RAM utilization since the code is loaded into RAM only when it is required. The overlay manager provides an API to load the appropriate region of code into the RAM.

Application developers can create the overlays by modifying the linker script. A utility *ov_utl* is provided that can be used to generate code stubs that can be used in the linker script and a corresponding C file with the appropriate data structures. Applications can then pass this data structure to overlays_load and indicate the region that should be loaded in the RAM. The APIs of the overlay manager are available in the *overlays.h* header file.

Please note that applications that use the overlay cannot use the *ramload* utility to load images in RAM. This is because there has to be a corresponding image in flash for the overlay to load. Hence once an application uses overlays, the application can be written and executed from flash using the *flashprog* utility.

#### 6.9.1.3 Footprint: Production Configuration

This release includes a *mc200_prod_defconfig* configuration that disables all the features typically used during development and debugging. This includes the CLI, the debug and the logging messages. Applications built with this WMSDK configuration are significantly smaller in size. For example, this configuration cuts down the size of the *wm_demo* flash binary image by about 50KB (out of the current 276KB).

#### 6.9.1.4 IAR Support

This release includes support for building the application images using the IAR development environment. This will enable customers with IAR experience to easily work with WMSDK. As of this release, the libraries of the WMSDK are built using the CodeSourcery toolchain; while the applications (sample_apps) are built against these libraries using IAR.

#### 6.9.1.5 Other Enhancements

- The WLAN driver for the station interface has been refactored to better work with the latest Wi-Fi firmware specification. This enables better maintainability of the WLAN driver.
- The hardware RTC is now used for keeping time in the WMSDK. The RTC driver is so used that it maintains the time properly across transitions to the power management modes PM2, PM3 and PM4. This support is available for MC200 chips with A0 version only. WMSDK automatically switches to soft-RTC for chipsets with previous versions.

- The MC200 firmware image structure has been modified. The new structure assists better in the use of RAM footprint conservation techniques like overlays and XIP.
- Modular web applications for wm_demo. Modularizing the web applications ensures that developers can easily pick-and-choose any components of the web application.
- Note that the 2.9 release disables TLS in the default configuration. This is to better ensure that developers using TLS are aware of the licensing implications.
- Leakage current in low power state further reduced by putting AON in deep power save.
- Support for chunked transfer encoding of the HTTP client.
- Device driver enhancements:
    - New APIs to modify cloud frequery/buffer size for SSP driver
    - DMA support for SPI slave read

### 6.9.2 New Applications

- **wm_demo with power manager framework**: When the Power Manager Framework option is enabled in the WMSDK build, the wm_demo is compiled to use the power manager framework. This makes the normal mode in wm_demo power conscious. The system will aggressively enter low power modes as and when possible. Some comments:
    - No change has been made to the provisioning mode. This is because the power manager framework supports aggressive power management with the station mode.
    - The power manager framework is enabled in the normal mode.
    - The micro-AP interface is not available in the normal mode. This ensures that the system enters the lowest power mode.
    - Normal network operations like httpd, mdns wakeup the mc200processor if it is in low power mode.
    - The wm_demo sets the behavior of the wakeup-key such that once pressed it disables the entire aggressive low power mode feature. This can be enabled again using the web interface.
- **wm_demo with overlays:** The wm_demo is enhanced such that it can be compiled to use overlays. This can be done by passing the following flag to the sample_apps build:

  ```
  $ make SDK_PATH=/path/to/wmsdk WMDEMO_ENABLE_OVERLAYS=y
  ```

  The wm_demo application uses WPS and the cloud communication module as the overlays. The WPS module is loaded into RAM when in provisioning mode, while the cloud communication module is loaded into RAM when in normal mode.
- **pm_mc200_demo:** A demo that puts the standalone MC200 in various power mode combinations. This allows the developer to measure power consumption in these various modes.
- **pm2_ieeeps_hs_demo/pm3_ieeeps_hs_demo:** Thesedemos are modified to use the power manager framework APIs when the option is enabled in the build configuration.

### 6.9.3 Release Notes

**Enhancements:**

1. Power Manager Framework support (Experimental)
2. Support for Overlays (overlays.h)
3. Overhauling of the WLAN driver

4. Support for Hardware RTC (A0 chips only)
5. Limited XIP Support
6. Support for IAR toolchain/IDE
7. Modular web applications for wm_demo
8. OpenOCD updated to latest version 0.6.1
9. Device driver enhancements:
    a. new APIs to modify clock frequency/buffer size for SSP
    b. DMA support for SPI slave read
10. Support for chunked transfer encoding in HTTP Client

**Bugfixes:**

Fixed the following:

1. httpd url parsing doesn't stop at '?'
2. multi-byte read issue in mdev I2C driver
3. Webserver hangs sometimes when the client closes connectionabruptly
4. Ring-buffer full condition incorrect in SSP, UART and I2Cdrivers
5. ssp_drv_read() needs to disable interrupts
6. CRC calculation APIs (block/stream mode) truncate data length to256KB

**API Changes:**

1. i2c_drv_{read/write} APIs no longer contain the flags parameter
2. New API for overlay management (overlays.h)
3. The APIs  httpd_parse_post_form()/httpd_lookup_tag() are now deprecated
4. httpd_get_tag_from_post_data()/httpd_get_tag_from_url() areavailable for querying tag/value pairs in GET/POST requests
5. New APIs for power manager framework introduced: pm_mcu_cfg() andpm_ieeeps_cfg()
6. The security type WLAN_SECURITY_WPA_WPA2_MIXED is available fornetwork profiles
7. Additional events AF_EVT_PROV_WPS_SUCCESSFUL andAF_EVT_PROV_WPS_UNSUCCESSFUL are added in the application framework toindicate intermediate steps during provisioning
8. New APIs ssp_drv_set_clk() and ssp_drv_rxbuf_size() for modifyingSSP clock frequency and receive buffer size
9. New APIs httpc_write_chunked(), for chunked transfer encoding andhttp_get_response_hdr_all() for reading HTTP response headername-value pairs
10. rtc_drv_config() is replaced by rtc_drv_set() that correctlydescribes the API. New API rtc_drv_get() for querying the uppval
11. Power Manager APIs pm_mc200_{standby|sleep|shutdown|idle} replacedby a single API pm_mcu_state().APIs pm_mc200_{sleep|standby}_hsreplaced by single API pm_mc200_state_hs().

### 6.9.4 Upgrading from older releases

Please take note of the following changes:

The 2.9 release changes the format of the firmware binary. The new format makes it easier to implement features like overlays and XIP that assist in better RAM utilization.

The entities that should be aware of the image format are boot2, flashprog and the firmware upgrades code. Thus,

- the boot2 code from WMSDK 2.8 or older will not be able to boot images generated from 2.9.
- the firmware upgrade code from WMSDK 2.8 or older will not be able to upgrade to images from 2.9.

Please reflash the boot2 generated from WMSDK-2.9 for booting application images.

## 6.10 WMSDK 2.8

### 6.10.1 SDK Features and Enhancements

#### 6.10.1.1 Power Management Enhancements

This release includes the following enhancements to the power management:

- **PM3 + ieeeps + Hostsleep:** Support for PM3+IEEEPS with hostsleep is now available. A new demo *pm3_ieeeps_hs_demo* is included that demonstrates this feature. This demo works with the A0 revision of the 88MC200 chip which has 192KB of retention ram in the PM3 mode. The *README* file within the *pm3_ieeeps_hs_demo* documents the build procedure that should be used for building this demo.
- **PM4 + Deepsleep:** Support for PM4 + deepsleep is available with this release. This can be exercised in any of the available demos. For example, in order to try this feature within wm_demo (assuming the station is connected to a valid AP), please perform the following steps as shown in the logs below. User actions are indicated in **bold,** while comments are written in italics.

```
 [af] app_ctrl [sta]: State Change: NORMAL_CONNECTING =>
NORMAL_CONNECTED
[app-wm-demo] wm_demo app: Received event 12
#
# wlan-disconnect
# [af] Warn: network_mgr: disconnected
[app-wm-demo] wm_demo app: Received event 17
[af] app_ctrl [sta]: State Change: NORMAL_CONNECTED =>
NORMAL_DISCONNECTED

# wlan-stop-network
```

```
# [af] app_ctrl [uap]: State Change: UAP_UP => UAP_INIT

#
# pm-wifi-deepsleep-enter

# Enabled Deep Sleep mode
[app-wm-demo] wm_demo app: Received event 27
 Power save enter


# pm-mc200-shutdown 3000
Timeout specified :3000 millisec
The system can be brought out of shutdown by an RTC timeout or a
wakeup key press
PM0 (Active)->PM4 (Shutdown)      For 3000 millisec
 [boot] SDK-2.8.13 gcc-4.6.3 (Nov 21 2012 04:03:40)
[boot] Boot Flags: 0x1
[boot]  - Firmware Partition: 1
[boot]  - Main firmware
[boot]  - Boot Override Time: 0.000 sec
[boot] Boot Info:
[boot]  - Chip revision id: 1
[boot]  - Bootrom version: 3
[boot]  - Boot option: flash
[boot] Reset Cause Register: 0x1059c1
[wlan] Waking from PM4, calling sdio_drv_reinit()
[wlan] txportno 2
```

As can be seen on exit from PM4 (i.e. the following bootup), the wlcmgr detects that the processor has exited from PM4. Since the Wi-Fi card is only in deepsleep mode, it already retains the WLAN firmware that was downloaded to it. Hence this step is skipped during bootup.

- The PMU clock is dropped to 1MHz in low-power modes which results in better current consumption.

### 6.10.1.2 WLAN Firmware Compression

This release includes support for WLAN firmware compression. The compressed WLAN firmware image occupies lesser space in flash. The *xz* algorithm is used for compressing the firmware image. This feature is enabled by-default, but can be disabled from the build configuration. xz-compressed firmware binaries are shipped with WMSDK2.8.

#### 6.10.1.3 Provisioning Enhancements

The provisioning module has been significantly modified to provide a better user-experience while provisioning. The wlan micro-AP state management that was earlier a part of the provisioning module is now removed. Provisioning is modified to be a pure service that can be started by the application. The application framework is so modified that the micro-AP interface will never have to be stopped while transitioning from provisioning to normal mode. This facilitates the querying of provisioning status from the UI once provisioning is performed. In case of unsuccessful provisioning the user can switch back to the provisioning mode and reprovision the device with the correct settings.

#### 6.10.1.4 Other Enhancements

- Support for new boards:
  - Lark mini Development Board v2
  - Azurewave AW-CU2XX
- WMSDK internally uses a CMSISv3.0-compliant 88MC200 driver API
- The UUID used for cloud communication now also considers the unique flash id of the internal flash of each chip, thus making it wider in size

### 6.10.2 API Enhancements

1. Provisioning changes:
   a. prov_start: The member 'struct wlan_network' has been removed from the struct prov_config. Connection mgmt is now left upto the application
   b. PROV_NETWORK_START/PROV_NETWORK_STOP/PROV_SUCCESSFUL events are now removed
   c. app_provisioning_start/app_provisioning_with_fs_start are replaced by a single API app_provisioning_start. The only parameter that is accepted is the provisioning mode
   d. AF_EVT_PROV_NW_START/AF_EVT_PROV_NW_STOP events are removed. AF_EVT_PROV_WPS_START event is introduced to indicate the start of WPS session
2. The application framework doesn't start the station interface by default. A new API app_sta_start() is introduced for allowing the applications to control when to start the station interface
3. New application framework event AF_EVT_NORMAL_AUTH_FAILED to indicate and authentication failure with the AP
4. board_gpio_power_on is now used such that boards can perform any boot-time GPIO settings in this function
5. New APIs json_set_val_uint/json_set_val_uint_64 are available to set unsigned 32/64-bit integer values
6. The board_led and board_button API names are modified as follows:

| Old Name | New Name |
|---|---|
| board_led_green | board_led_1 |
| board_led_yellow | board_led_2 |
| board_led_white | board_led_3 |

| | |
|---|---|
| board_led_blue | board_led_4 |
| board_button_wps | board_button_1 |
| boad_button_reset_prov | board_button_2 |
| board_button_boot_override | board_button_3 |

## 6.10.3 Release Notes

**Enhancements:**

1. Drop PMU clock to 1MHz in low-power modes. This results in better current consumption
2. Support for PM3 + IEEEPS with hostsleep
3. Support for PM4 + Deepsleep power save
4. Provisioning changes:
    a. Provisioning is now a pure service. WLAN state management is managed by the application
    b. Provisioning success/failure result available in the UI pages
    c. Avoid reboot for reset-to-provisioning
5. WMSDK internally uses a CMSISv3.0-compliant 88mc200 driver API
6. WLAN firmware compression support. Compressed WLAN firmware images consume lesser flash space
7. Support for new boards:
    a. Lark mini Development Board v2
    b. Azurewave AW-CU2XX
8. USB Mass Storage Client Demo

**Bugfixes:**

1. Change network in normal does not always work

**API Changes:**

1. Provisioning changes:
    a. prov_start: The member 'struct wlan_network' has been removed from the struct prov_config. Connection mgmt is now left upto theapplication
    b. PROV_NETWORK_START/PROV_NETWORK_STOP/PROV_SUCCESSFUL events are now removed
    c. app_provisioning_start/app_provisioning_with_fs_start are replaced by a single API app_provisioning_start. The only parameter that is accepted is the provisioning mode
2. AF_EVT_PROV_NW_START/AF_EVT_PROV_NW_STOP events are removed. AF_EVT_PROV_WPS_START event is introduced to indicate the start of WPS session
3. The application framework doesn't start the station interface by default. A new API app_sta_start() is introduced for allowing the applications to control when to start the station interface

4. New application framework event AF_EVT_NORMAL_AUTH_FAILED to indicate and authentication failure with the AP
5. board_gpio_power_on is now used such that boards can perform any boot-time GPIO settings in this function
6. New APIs json_set_val_uint/json_set_val_uint_64 are available to set unsigned 32/64-bit integer values
7. The board_led and board_button API names are modified as follows:

Old name - New name

board_led_green - board_led_1

board_led_yellow - board_led_2

board_led_white - board_led_3

board_led_blue - board_led_4

board_button_wps - board_button_1

board_button_reset_prov - board_button_2

board_button_boot_override - board_button_3

### 6.10.4 Upgrading from older releases

Please take note of the following changes:

1. The 2.8 release uses a compressed WLAN firmware image in order to use lesser flash space.
2. The 2.8 release modifies the flash layout such that the ftfs partitions are moved to the external flash, and the wireless firmware image is moved into the internal flash. Such a layout allows for maximum flexibility for sizes of various components in the system.

Please update the flash layout on your boards as follows:

1. Goto *wmsdk/tools/OpenOCD*
2. Execute

```
./flashprog.sh --new-layout layout.txt
```

3. Flash boot2, firmware and filesystem and new wireless firmware images.

**Note**: All settings stored in psm will be lost.

## 6.11 WMSDK 2.7

### 6.11.1 SDK Features and Enhancements

#### 6.11.1.1 Power Management

This release includes enhancements and fixes in the power management code to ensure optimal power utilization in the various host and Wi-Fi power save modes. These include

- Switch off all I/O domains on entry into the 88MC200 power modes
- Reduce the Wi-Fi wakeup pulse duration. This reduces the time taken by the Wi-Fi card to send data to the host after wakeup. Thus the overall wake-time is reduced.

This release also modifies the IEEEPS sleep-wakeup mechanism to ensure that it works better with the 88MC200 hostsleep configuration.

#### 6.11.1.2 Quick Network Reconnect

The Wi-Fi association code path has been updated to ensure quick network reconnect. The associated configuration parameters are cached during the first connection attempt. These parameters are used for subsequent connection attempts thus eliminating valuable steps and time for reconnection attempts.

#### 6.11.1.3 Other Enhancements

- Support for P2P disconnect
- Allow apps to initialize the network stack earlier even before the Wi-Fi is initialized. This ensures that application threads can create and communicate using local sockets earlier in the boot cycle

### 6.11.2 API Enhancements

- The *p2p_disconnect* API is now available to disconnect from an existing P2P session.
- The WPS APIs are updated to be consistent with the P2P APIs. Thus, the *wps_send_command* API has been replaced with the *wps_connect* API.
- The *sdio_drv_deinit* function is available to de-initialize the SDIO driver.
- http_get_response_hdr_value API is now available that allows apps to read various fields in the HTTP response.

### 6.11.3 Release Notes

**Enhancements:**

1. Power management changes:
    a. switch off all I/O domains in low power mode
    b. wifi host-wakeup pulse duration is reduced
    c. optimized pm2+ieeeps power consumption in low_power_demo
2. Enable fast wlan connection on link-loss
3. Allow apps to create local sockets even for wifi is initialized
4. p2p_disconnect() support
5. httpc: function to read individual http headers

**Bugfixes:**

1. Fix a race condition in wlan_stop
2. Interoperability fixes for P2P
3. Fix i2c driver for multibyte reads
4. Fix ieeeps wakeup-sleep logic in conjunction with hostsleep

**API Changes:**

1. The p2p_disconnect() API is now available to disconnect from an existing P2P session

2. The WPS APIs are updated to be consistent with the P2P APIs. Thus, the wps_send_command API has been replaced with the wps_connect API
3. The sdio_drv_deinit function is available to de-initialize the SDIO driver
4. http_get_response_hdr_value API is now available that allows apps to read various fields in the HTTP response

### 6.11.4 Upgrading from older releases

No configuration changes are required for upgrading from the 2.6 release to the 2.7 release.

## 6.12 WMSDK 2.6

### 6.12.1 Features and Enhancements

#### 6.12.1.1 802.11n Features

Apart from bugfixes throughout the datapath and refactoring, this release additionally supports the following features of 802.11n:

- A-MPDU / A-MSDU: The WMSDK supports reception of Aggregation of MAC Service Data Units (MSDU) and MAC Protocol Data Units (MPDU). The aggregation process combines multiple MPDUs and MSDUs together in order to minimize per-frame transmission overhead and amortize them over the individual frames.
- Support for the 5GHz band: Along with the 2.4GHz spectrum, the device also supports connection and data transfer with Access Points in the 5GHz band. The 5GHz band allows for lesser wireless interference and greater throughput by means of channel bonding.

#### 6.12.1.2 Wi-Fi-Direct/P2P

This release includes the support for Wi-Fi-Direct/P2P. P2P enabled devices can connect to each other in a peer-to-peer fashion without the need of an Access Point. The P2P APIs are available in the WMSDK Reference Manual and available via the *p2p.h* header file. A new sample application *p2p_demo* has been included to demonstrate the use of P2P. You may connect your P2P-enabled smart phone to this application.

#### 6.12.1.3 USB CDC Driver

A driver for the CDC-ACM USB class is included in the WMSDK. This class provides serial emulation for communication with a host. When connected to a host with appropriate drivers, the device is detected as a virtual serial port (Virtual COM Port on Windows) and can be communicated to using appropriate applications (HyperTerminal on Windows). The APIs of the USB driver are available from the header file *mdev_usb.h.* A new sample application *usb_demo* is provided that uses the USB CDC class.

#### 6.12.1.4 Interface-aware HTTPd WSGI Handlers

This enables applications to implement different functionalities on the different interfaces of the device. For example, administration functionality can be made available only on the micro-AP interface while such requests will not be available on the station interface. Network API *net_sock_to_interface* is included in the WMSDK to assist this. WSGI handlers can use this API to determine whether to wish to service the requests from a given interface.

#### 6.12.1.5 Other Enhancements

- DTP Driver changes: The dtp driver protocol has been updated to be more robust.
- SDIO Driver Refactoring: The SDIO driver has been refactored to use the mdev infrastructure. The mdev APIs for SDIO are mentioned in the mdev_sdio.h header file.
- SSP Driver: Bugfixes in the SSP driver and enhancements to use DMA mode.
- 802.11d: Support for region codes and split scan using these.
- The following re-organizations have been performed for a cleaner SDK usage:
    - wlcmgr and wifidriver are now available in the src/wlan/ directory.
    - mc2k references have been changed to mc200 throughout the SDK.
    - *main()* available as the application entry point.

#### 6.12.1.6 API Enhancements

- The new mdev SDIO driver exposes the APIs *sdio_drv_init, sdio_drv_open, sdio_drv_creg_read, sdio_drv_creg_write, sdio_drv_read, sdio_drv_write* are available to applications. Thus far the SDIO driver APIs were not available for applications to use.
- The new USB CDC driver exposes the APIs *usb_drv_init, usb_drv_open, usb_drv_read, usb_drv_write, usb_drv_close*.
- The P2P features introduces the APIs *p2p_start, p2p_stop* for starting and stopping the P2P service. And APIs *p2p_scan, p2p_connect and p2p_disconnect* for scanning and connecting or disconnecting to other P2P devices or groups.
- The word *mc2k,* if found in an API, is replaced by *mc200.*
- All of the APIs of the dtp driver have changed.

### 6.12.2 Sample Applications

The following new demos have been included:

- usb_demo: This demonstrates the USB CDC class functionality. When connected to a host with appropriate drivers, the device is detected as a virtual serial port (Virtual COM Port on Windows) and can be communicated to using appropriate applications (HyperTerminal on Windows). The USB OTG port (if available on your board) should be used for this communication.
- p2p_demo: This application demonstrates the P2P functionality. When the application comes up, a P2P device named 'WMdroid-XXYY' (where XXYY are the last two components of your Wi-Fi card's MAC Address), is visible. P2P-enabled smart phones or devices are able to scan and connect to this P2P enabled device. The IP Address of the device is visible on the LCD. You may open this IP Address in your smartphone's browser to communicate with the device over P2P.

### 6.12.3 Release Notes

**Enhancements:**

1. Support for receiving A-MPDU and A-MSDU frames
2. Support for 802.11d region codes
3. Support for 5GHz band in 802.11n mode
4. Support for Wi-Fi-Direct (P2P)
5. USB CDC Driver support

6. DMA support in SSP mdev driver
7. LWIP sources updated to 1.4.1-rc1
8. Provide APIs such that HTTPD WSGI handlers can be interface-aware
9. Refactoring of the SDIO driver. This is now mdev-based

**Bugfixes:**

1. Multiple failures are observed before wlan_connect() can connect to a configured network
2. WPS reliability fixes
3. Race-condition in the heap allocator observed under heavy load
4. Data communication when IEEPS mode on causes firmware hang

**API Changes:**

1. SSP mdev driver init API accepts an additional parameter for dma
2. New APIs introduced for SDIO driver, USB CDC driver and Wi-Fi P2P
3. main() is also now available as an application entry point
4. pm_mc2k_sleep, pm_mc2k_shutdown, pm_mc2k_standy, pm_mc2k_idle, pm_mc2k_standby_hs are changed to pm_mc200_sleep, pm_mc200_shutdown, pm_mc200_standy, pm_mc200_idle, pm_mc200_standby_hs
5. All the APIs for the DTP driver

### 6.12.4 Upgrading from older releases

This Wireless Firmware available with this release includes the support for P2P. This increases the wireless firmware size. The same has been updated in the default flash layout. Please update the flash layout on your boards as follows:

4. Goto *wmsdk/tools/OpenOCD*
5. Execute

```
./flashprog.sh --new-layout layout.txt
```

6. Flash boot2, firmware and filesystem and new wireless firmware images.

**Note**: All settings stored in psm will be lost.

## 6.13 WMSDK 2.5

This section describes the changes in WMSDK v2.5 over the WMSDK v2.4 release.

### 6.13.1 SDK Features and Enhancements

The following new features are available as part of this release.

### 6.13.1.1 Wake-on-WLAN Support

This release includes the support for Wake-on-WLAN. With this feature, the host MC200 processor can configure the Wi-Fi card to wake the host processor on the occurance of certain events (unicast data, multicast data etc.) and then enter the PM2 power save mode. The Wi-Fi card can then wake the host processor up when the configured event occurs.

The Wi-Fi card can itself be in the IEEE power save mode, thus ensuring that system is in a very low power mode. This support is only available with the Lark-mini Development Boards. Please consult your Marvell Sales representative for support and workarounds for other boards.

### 6.13.1.2 802.11n Support

This release includes 802.11n support for the station mode. This enables association and data transfer with pure 11n Access Points apart from mixed b/g/n Access Points. 802.11n support brings enables a more efficient modulation that brings the maximum data rate of a single 802.11n OFDM channel to 65Mbps as against 54Mbps for 802.11a/g.

For APs that support Wi-Fi MultiMedia (WMM), support for transmitting data using AC_BE (Best Effort) is also included.

802.11n support is enabled in WMSDK by default. All the functionality benefits of 802.11n will thus be available to all applications without any change in the existing APIs. The Wi-Fi driver will automatically detect a pure-11n or b/g/n mixed-mode AP and choose the 802.11n mode for association and data transfer.

The following features of 802.11n are not yet supported:

- Receiver side AMPDU and AMSDU aggregation
- Channels in the 5GHz band

### 6.13.1.3 Multiple Interfaces Support in mDNS/DNS-SD

This release introduces multiple interface awareness in the mDNS/DNS-SD module. This provides the following benefits when multiple interfaces are simultaneously up (for example microAP and Station):

- The mDNS/DNS-SD module responds to its own hostname resolution requests intelligently be detecting the interface on which the request has arrived. For example,
    - Consider that the hostname is wmdemo-aabb.local and the micro-AP interface has IP Address 192.168.10.1 while the station interface has IP Address 192.168.2.24. Now when a host on the 192.168.2.x network wishes to resolve the hostname wmdemo-aabb.local, WMSDK responds with the correct address 192.168.2.24, while if the same request is received on the 192.168.10.x network then 192.168.10.1 is returned.
- Applications can register one set of services on one interface and a completely different set of services on the other interface. mDNS/DNS-SD will appropriately respond to service discovery and announcement requests based on the interface on which the request is received.

For both the above features, we have had to make changes to the API exposed by the mDNS/DNS-SD module. Please refer to the API Enhancements section for more details.

#### 6.13.1.4 Miscellaneous Enhancements

Some of the miscellaneous enhancements include:

- In Provisioning UI, display scan results in decreasing order of signal strength
- Support for the Lark-mini (LK20) Development Board

### 6.13.2 API Enhancements

Here is the list of APIs that have changed in this release. Please refer to the *WMSDK Reference Manual* for more details about the APIs and the corresponding parameters.

- **Finer-grained sleep states for MC200**:

  In order to facilitate a finer-grained control over the power save durations, the input parameter to all these functions has changed. Earlier these functions used to accept the number of *seconds* to enter the power save modes, while in the new API these functions accept the number of *milli-seconds* to enter the power save modes. This affects the APIs *pm_mc2k_shutdown*, pm_*mc2k_sleep*and *pm_mc2k_standby*.

- **SSP driver enhancements**:

  A finer grained control over the *Chip Select* pin used for SSP interfacing is provided by the driver. This affects the API *ssp_drv_open* and *ssp_pin_config*. These APIs now also accept the GPIO number, that will be used for chip-select, as a parameter. New APIs *ssp_drv_cs_activate* and *ssp_drv_cs_deactivate* are also provided to activate or de-activate the chip-select pin.

  The *ssp_drv_write* API has been modified to optionally read out data from the rx fifo.

- **Interface aware mDNS/DNS-SD**:

  This affects the *mdns_start, mdns_set_nw_interface* and *mdns_set_ipaddr* APIs. From this release onwards, mdns_start will start the mDNS service, but it won't be listening on any interface. Applications should register the interfaces that they are interested in using the *mdns_add_service_iface* call. This call also specifies the list of services that application wish to announce on the corresponding interface.

  Applications should notify the change in interface status (up, down etc.) using the *mdns_iface_state_change* call.

- **Wake-on-WLAN**:

  A new API *pm_mc2k_standby_hs* has been introduced to configure wake-on-wlan and enter the PM2 standby mode. One of the parameters to this API is the various wakeup conditions for which the Wi-Fi card should wake up the host processor from the PM2 mode.

### 6.13.3 Sample Applications

#### 6.13.3.1 Additional Applications

An additional application *float_demo* has been introduced to demonstrate some of the constraints while printing float and double data types.

An additional application *low_power_demo* has been introduced that utilizes the host sleep functionality and puts both the host processor and the Wi-Fi card in a low power mode. Wake-on-WLAN is configured such that the host is woken up to serve any network requests.

### 6.13.4 Release Notes

**Enhancements:**

1. Include support for hostsleep and waking up the host using events on the WLAN interface
2. Partial 802.11n support. 802.11n rx-side aggregation (AMPDU/AMSDU) not yet supported
3. Display scan results in decreasing order of strength with better granularity
4. Support for new CodeSourcery toolchain 2012.03-56
5. Support for Lark-mini (LK20) Development Board
6. Added multiple interface support in mDNS
7. WMM: For APs that support WMM, traffic is classified as AC_BE (Best Effort)
8. Enhancement to 'sysinfo thread' output to show thread priorities

**Bugfixes:**

1. hardfault when cloud enable-disable sequence is called frequently
2. pings with size greater than 2048 fail
3. association with an AP with incorrect security settings. If the device is configured to associate with an WPA2 AP, it shouldn't associate with an AP in open mode having the same SSID

**API Changes:**

1. 88MC200 power management APIs which put system in low power mode accepts time duration in milliseconds
2. JSON get_val_str APIs do not truncate the string to the buffer size but return an error instead
3. New API pm_mc2k_standby_hs() for entering standby mode with hostsleep (wake-on-wlan) feature
4. SSP mdev based driver APIs for driver init and and driver write operations

### 6.13.5 Upgrading from older releases

No changes are required for upgrading from WMSDK 2.4 to WMSDK 2.5. If you are using a release older than WMSDK2.4, please refer to the file *wmsdk-2.5.x/README.first* for upgrade instructions.

## 6.14 WMSDK 2.4

### 6.14.1 SDK Features and Enhancements

The following new features are available as part of this release.

#### 6.14.1.1 Improved Flash Layout Configuration

With this release, we have done significant refactoring of the flash layout configuration such that it is easier and less error-prone for applications. As against earlier releases, where the flash layout is specified in the applications, in this release, the flash layout is written to the flash using the flash programming utility. Once written, all entities (application, flashprog) reference this layout to understand where each of the flash components is stored. This makes flash layout management easy-to-use. Also having a single layout available in the flash makes it less error-prone than before.

#### 6.14.1.2 Drivers for 88MC200 Peripherals

We have added device drivers using the Marvell Device Framework (mdev) for the following interfaces:

- PWM support in GPT driver
- I2C slave mode support in I2C driver

#### 6.14.1.3 Miscellaneous Enhancements

- A number of WLAN scan related improvements for a better experience.
    - Preference is given to stations with high signal strength
    - The scan operation is now split into multiple sets in order to ensure that the connected stations are not dropped in the process
- Support for the MC200 Development Board V2
- TCP tx performance improvements because of support for back pressure handling in the network stack
- Auto-channel in micro-AP mode
- Ongoing code refactoring to make the code structure and API's more intuitive, consistent, and easier to use

### 6.14.2 Sample Applications

#### 6.14.2.1 WM Demo

With the availability of TLS over HTTP, writing a cloud communication module is a straight-forward task. Thus the cloud middleware has now been removed. The wm_demo application implements the entire cloud functionality by itself.

#### 6.14.2.2 Additional Applications

An additional application *uap_power_save_demo* has been introduced to demonstrate the power save capabilities in the micro-AP mode. In this demo a micro-AP interface is started and put into the power save mode.

### 6.14.3 Release Notes

**Bugfixes:**

1. Reliable association of Windows-7 devices with the micro-AP mode
2. memory corruption while loading axf
3. 'uaputl bss_stop' doesn't stop micro-AP network
4. Incorrect time representation (and hence incorrect invalidation of TLS certificates)

5. WPS:
    a. Corruption of the scan list
    b. Synchronization errors with some Access Points
6. Alignment fixes while freeing memory in ttcp
7. Fix hang-ups in Wi-Fi Power Down (PDN) mode (a WLAN Firmware command was being sent in PDN mode)

**Enhancements:**

1. micro-AP power save support (along with a sample application)
2. PWM support in GPT driver
3. Interrupt-based SSP slave driver
4. Enhanced script for footprint analysis
5. wlan-scan improvements
    a. preference to networks with better signal strength
    b. separate scanning on channel-sets to ensure connected stations aren't dropped
6. Improved flash layout configuration
7. I2C slave mode support in I2C driver
8. Add back-pressure handling support in LWIP. Improves TCP tx performance
9. I2C support in DTP driver
10. API to modify multicast filters by applications
11. Support for auto-channel in micro-AP
12. Upgrade CyaSSL to version "2.2.0-Commercial"
13. Power Save callback infrastructure for device drivers
14. Support for MC200 Development Board V2

**API Changes:**

1. rfget_server_mode_update/client_mode_update now accept a passive partition entry instead of the type of upgrade
2. ftfs_init/psm_init/wlan_init now accept a flash descriptor that informs their corresponding flash locations
3. New APIs rfget_update_begin/data/complete for implementing upgrades for your own flash component
4. New APIs for maintaing multicast filters for the wireless interface
5. app_httpd_with_fs_start and app_provisioning_with_fs_start now accept the filesystem's friendly name as a parameter

The cloud middleware is no longer available. This functionality is now quite straight forward and hence moved to wm_demo.

# 6.15 WMSDK 2.3

## 6.15.1 SDK Features and Enhancements

The following new features are available as part of this release.

### 6.15.1.1 Application Framework Refactoring

With this release, we have done significant refactoring of the application framework with the aim of providing applications control over what services are used by the application.

The application framework now focuses on managing the Wi-Fi connectivity and transitions between different states, and leaves applications with the control on what services or functionality is used in different states. A set of convenience functions are provided to offer common services such as provisioning, service discovery, Micro-AP networking, etc.

This refactoring provides flexibility to applications and enables a broader class of applications to use the application framework without having to go and modify the application framework itself. It also enables applications to manage their footprint as features that are not needed are not linked into the final application image.

### 6.15.1.2 Drivers for 88MC200 Peripherals

We have added and enhanced device drivers using the Marvell Device Framework (mdev) for the following interfaces:

- SSP (both I2S and SPI)
- ADC and DAC
- GPIO driver has been enhanced to support application callback on interrupts

We have also enhanced other drivers and tried to unify the API's of the different drivers to present a more consistent API.

### 6.15.1.3 Board Support

The SDK now supports

- Lark Development Board v2
- Standalone Lark modules (v2 and v3)

We have also refactored the code such that all board support functions have been moved to the sample applications (under boards sub-directory). Each of the supported board has the support functions in a separate file, and the sample applications must be built to include the correct board support file.

This also facilitates supporting new boards as the changes needed for a new board are all encapsulated in a single place, and can be derived from the board-support functions for other boards as appropriate.

### 6.15.1.4 Miscellaneous Enhancements

- HTTP Server has been enhaced to support
  - Secure connections using TLS library (cyassl)
  - Client-side caching via If-None-Match/Cache-Control support in HTTPD

- 88MC200 PM3 power mode support is now available. An application is available in sample applications to exercise this power mode
- Ongoing code refactoring to make the code structure and API's more intuitive, consistent, and easier to use.

## 6.15.2 Sample Applications

### 6.15.2.1 WM Demo

The WM Demo application has been rewritten to:

- Work with the modified application framework.
- Start a Micro-AP network when it is connected to its "home" network showing simultaneous AP-Client mode
- A new HTML5, mobile friendly web-application to support provisioning and display system information using the JQuery Mobile Framework.

### 6.15.2.2 Additional Applications

A wider range of sample applications, to provide in-depth understanding of the various features are provided with the WMSDK. Apart from the applications hello_world, wm_demo and dtp_demo that were available in the previous WMSDK releases the other applications are categorized as follows:

1. **io_demo**: Demonstrates applications that use the I/O interfaces on the development board. These include gpio_demo, spi_demo and uart_demo.
2. **microap_demo**: Demonstrates applications that start the micro-AP network on the device. A step-by-step evolution of an application up to WSGI handlers is provided by the demos: uap_demo, uap_http_demo and uap_wsgi_demo. Additionally, micro-AP configuration using the application framework is demonstrated in uap_demo_with_af.
3. **module_demo**: Demonstrates the usage of individual middleware modules in the system. These include the CLI, PSM, the TLS HTTP client and demo for the PM3 Power-save mode.
4. **prov_alternatives**: Demonstrates the alternate provisioning methods that can be used in conjunction with the application framework. This includes provisioning without using any network (e.g. using keyboard or touch-screen) and one using a micro-AP network but with custom WSGI network handlers.
5. **rtos_demo:** Demonstrates the simple creation of a thread in the OS.

## 6.15.3 Release Notes

**Bugfixes:**

1. Broken routing when micro-AP-STA are simultaneously activated
2. Micro-AP settings overwrite the default DNS settings
3. Sometimes device hangs in obtaining address state after association with the AP
4. Zero-initialize allocated memory to fix WPS-not working cases on some boards

**Enhancements:**

1. HTTPD TLS support
2. Random Number generator seed derived from system params like signal strength etc.

3. Reorg: Board-bring up files, boot2 are now available in wmapps
4. mdev based drivers for: ADC, SSP (I2S and SPI), DAC
5. Wi-Fi firmware heartbeat handler to take care of situations when the Wi-Fi firmware has crashed
6. If-None-Match/Cache-Control support in HTTPD. Supports client-side caching
7. Support for building images for the standalone Lark module (no development board)
8. PM3 Power Save mode support and demo application
9. Application Framework state machine refactored. More flexibility to applications to decide what should be done on various events

**API Changes:**

1. wmstdio_printf is renamed to wmprintf
2. Changes in mdev APIs: acomp_drv_init, gpt_drv_open, gpt_drv_init,i2c_drv_open, uart_drv_{init|open|read|write|ioctl|close}. Pleaserefer to the reference manual for more information
3. dhcp_server_start() no longer accepts a name_to_spoofparameter. Name resolution is taken care of by mDNS
4. Application Framework only manages the state machine now. More APIsare available for applications to have a greater control on the services available in various modes
5. Modules initialize the drivers that they are dependent on. No needto take care of this in the application
6. Default binary for a project is now project_name.bin (the *_flash*suffix isremoved)

## 6.16 Upgrading from WMSDK 2.2

Please take note of the following changes:

1. This release changes the on-flash signature of some components. It is recommended that you take the following steps:
    a. Using flashprog erase entire internal/external flash (advanced menu option).
    b. Flash boot2, firmware and filesystem and wireless firmware images.

**Note** All settings stored in psm will be lost.

2. The default name of the generated flash binary is now changed to only say project_name.bin as against project_name_flash.bin
3. flash_pack.py now by-default compresses all .html/js/css files in the directory before generating the fttfs image

# 6.17 WMSDK 2.2

## 6.17.1 Features

The following new features are available as part of this release.

1. Transport Layer Security (TLS and SSL) support is added via CyaSSL v2.0.8 libraryintegration in WMSDK
2. HTTP Client Library now supports secure https connections using the TLS library
3. AES hardware acceleration support
4. Application Specific Flash Layout Configuration
5. Antenna Diversity
6. Long Listen Interval
7. Secure Cloud Communication Support
8. mdev driver interface for: General Purpose Timers (GPT), CRC, AES, RTC, Watchdog,Analog Comparator
9. Lark Development Board Support

## 6.17.2 Sample Applications

The sample applications are now available as a separate tarball as against being a part of theWMSDK.
The following additional functionality is now available as part of the wm_demo sample application.

1. **Long Listen Interval**: A long listen interval for the IEEE PS mode can be configured asfollows:

```
# pm-wifi-configure-listen-interval 50
#
```

   This command ensures that the next time you enter IEEE PS mode, the device willadvertise a longer listen interval to the Access Point. Long listen intervals allow the deviceto be in power-save mode for a longer duration of time, thus providing more energyefficiency.

2. **Application specific flash layout**: This enables application the flexibility to configure theflash layout as desired. Every application contains a flash layout file that describes thelocation of these components. Please refer to the Section 7.7 for more details about theflash layout.

## 6.17.3 Release Notes

**Bugfixes:**

1. Fixes in ttcp and network stack to address performance issues
2. Conflict during parallel usage of AES operations
3. mDNS correctly responds to requests without being case-sensitive
4. Provision to update IP Address and hostname in the mDNS service
5.

**Enhancements:**

1. Flashprog

    a. Flashprog utility modified to have consistent entry point address across changes insource code
    b. Batch processing support to flash all binaries (boot2, fw, ftfs, wlanfw) in single step
2. Added support for AES hardware accelerator
3. CyaSSL v2.0.8 (Commercial version) TLS library integrated with WMSDK
4. Mdev abstraction layer for following drivers
    a. General purpose timers (GPT)
    b. CRC
    c. AES
    d. RTC
    e. Watchdog
    f. Analog Comparator
1. TLS (https) support in http client
2. Application Framework: enable mdns service in provisioning mode
3. Flash Layout Selection
    a. Enable per-application flash layout configuration
    b. flashprog refers to the application specific flash layout
6. Support for multiple FTFS partitions
7. Antenna Diversity support: select best possible antenna
8. Demo applications are split into separate wmapps tarball
9. mDNS is enabled in provisioning mode
10. Choice of long-listen interval (upto 65 seconds) for IEEE PS
11. Cloud communication support

**Known Issues:**

1. Hardware RTC is not supported

**API Changes:**

1. *http_open_session()* now accepts TLS configuration parameters for connecting with httpsenabled servers.
2. New APIs *http_prepare_req()* and *httpd_add_header()* are provided to have moreflexibility in deciding the HTTP headers to be sent in an HTTP request.
3. *WMSDK Reference Manual* includes documentation for the newly available APIs forGeneral Purpose Timers, CRC, AES, RTC, Watchdog, Analog Comparator, TLS, Cryptoand Long listen interval.

# 6.18 WMSDK 2.1

## 6.18.1 Features

The following new features are available as part of this release:

1. Power Management support for the micro-controller chip 88MC200. New power statesinclude idle, standby and shutdown.
2. Power Management support for the wireless chipsets 88W8787 and 88W8782. New power save modes include deep sleep, power down and IEEE power save.
3. Support for simultaneously using the station as well as the micro-AP interface.
4. Hardware CRC support.

## 6.18.2 Sample Applications

The following additional functionality is now available as part of the wm_demo sample application.

1. **88MC200 Power Management**: A switch to the various power modes of 88MC200 can be done as shown below:

    a. **Idle**: This command switches the CM3 core into the idle state. When an interrupt to the CM3 core occurs, the core is quickly reactivated at the point where itentered the idle mode.

```
# pm-mc2k-idle
PM0->PM1
Duration in PM1 990 micro secs
PM1->PM0
```

    b. **Standby**: This command switches the CM3 core, most of the MCU peripherals and SRAM arrays in a low power-mode. The PMU and RTC are operational. The system can get out of standby mode by a pre-defined timeout. If no timeout is specified, the *SW3* key on the evaluation kit can be used to get the system out of standby.

```
# pm-mc2k-standby standby 15
PM0->PM2 For 15 sec
PM2->PM0 (Wakeup by Timeout 15 sec)
```

    c. **Shutdown**: This command switches most of the components off. RTC is kept active. On wakeup from this state execution begins at the bootrom.

```
# pm-mc2k-shutdown 15
PM0->PM4 For 15 sec
Boot Flags: 0x0
- Firmware Partition: 0 (main firmware)
- Boot Override Time: 0.000 sec
```

2. **Wi-Fi Power Management**: A switch to the various power modes of the wireless device as shown below. Note that this should be executed only in the normal mode and not in the provisioning mode. This is because Wi-Fi power management for only the stationmode is supported right now.

    a. **Power Down**: This will put the Wi-Fi card into the power down mode. On wakeup from PDN, the wireless firmware is downloaded to the Wi-Fi card again,and reconnection to the Access Point is triggered.

```
# pm-wifi-pdn-enter 15
```

    b. **IEEE PS**: This will put the Wi-Fi card into IEEE power save mode. The modegets enabled for a pre-defined time interval, or until a corresponding *pm-wifiieeeps-exit* command gets called.

```
# pm-wifi-ieeeps-enter 15
Enabled IEEE power save mode
Disabled IEEE power save mode
```

    c. **Deep Sleep**: This will put the Wi-Fi card into deep sleep mode. The mode getsenabled for a pre-defined time interval, or until a corresponding *pm-wifideepsleep-exit* command gets called. In this mode, connection with the AccessPoint is lost. On wakeup, if the WLAN state before deep sleep was connectedthen a reconnection attempt is triggered.

```
# pm-wifi-deepsleep-enter 15
Enabled deep sleep mode
Deep sleep timeout: 0
Disabled deep sleep mode
```

3. **Simultaneous AP-STA**: The wireless interface can now be simultaneously used in thestation as well as the micro-AP mode. In order to do this, execute the followingcommands, once you are in the normal mode.

```
# uaputl sys_cfg_ssid <SSID>
# uaputl sys_cfg_wpa_passphrase <WPA2 passphrase>
# uaputl bss_start
# dhcp-server uap0 mydevice start
```

Once the micro-AP interface is up, you can scan for the micro-AP SSID from your laptopand get connected to it.

### 6.18.3 Release Notes

**Bugfixes:**

1. http client chunked encoding handling issue solved
2. Fixed reset to factory issue with boot2
3. Include GPT and GPIO IRQs in vector table

**Enhancements:**

1. Hardware CRC support
2. 88MC200 Power management support for PM1 (idle),PM2 (standby) and PM4 (shutdown)
3. Simultaneous micro-AP + STA support
4. Support for uaputl and more options in iwpriv (getsignal, getdatarate, antcfg)
5. Easy-to-use API for all device drivers
6. Wi-Fi Power Management support
7. Use optimized versions of memset/memcpy/memmove calls

**Deprecated:**

1. Support for 8688 is removed. freertos and lwip are the only supported OS and networking stack now.
2. Support for AdHoc network is removed

**Known Issues:**

1. Hardware RTC is not supported
2. Cloud communication is not supported

**API Changes:**

1. The API to access all the device drivers has changed. As against the functions prefixedwith *mdev_* for WMSDK v2.0, this version includes an easier to use device driver API.

# 7 Glossary

ACOMP: Analog comparator
ADC: Analog to Digital convertor
AES: Advanced encryption standard
AP: Access Point
AMPDU: Aggregated Mac Protocol Data Unit
AMSDU: Aggregated Mac Service Data Unit
CRC: Cyclic redundancy check
DAC: Digital to Analog convertor
DHCP: Dynamic host configuration protocol
FTFS: File Table file system
GPT: General purpose timer
HTTP: Hypertext transfer protocol
IEEEPS: IEEE power save
JSON: JavaScript Object Notation
MCU: Microcontroller
mDNS: Multicast DNS
uAP: micro access point
P2P: Wi-Fi Peer-to-Peer
PSM: Persistent storage manager
QSPI: Quad serial peripheral interface
REST: Representational state transfer
RTC: Real Time Clock
SDIO: Secure Digital IO
SoC: System on chip
SPI: Serial Peripheral interface
SSL: Secure Sockets Layer
TLS: Transport Layer Security
URL: Uniform resource locator
WMSDK: Wireless microcontroller software development kit.
WPS: Wi-Fi Protected Setup
WSGI: Web Server Gateway Interface
XIP: Execute in place

Marvell Semiconductor, Inc.
5488 Marvell Lane
Santa Clara, CA 95054, USA

Tel: 1.408.222.2500
Fax: 1.408.752.9028

www.marvell.com

**Marvell.Moving Forward Faster**