WINC1500 Software

Release Notes

VERSION : **19.7.3**

DATE : **NOV, 2020**

Abstract

This document presents an overview of the WINC15x0 firmware release version 19.7.3, and corresponding driver.

1 Introduction 3
1.1 Firmware readiness 3
2 Release summary 4
2.1 Auditing information 4
2.2 Version information 4
2.3 Released components 5
2.4 Release Comparison 6
3 Test Information 9
4 Known issues 10
5 New Features 12
5.1 TLS Application Layer Protocol Negotiation 12
5.2 Read and apply I/Q calibration values from efuse 13
6 Fixes and enhancements 14
6.1 Issues fixed 14
6.2 Enhancements 17
7 Terms and Definitions 19

1 Introduction

This document describes the WINC15x0 version 19.7.3 release package.

The release package contains all the necessary components (binaries and tools) required to make use of the latest features including tools, and firmware binaries.

1.1 Firmware readiness

Microchip Technology Inc. considers version 19.7.3 firmware to be suitable for production release

2 Release summary

2.1 Auditing information

Master Development Ticket	:	https://jira.microchip.com/projects/W1500/versions/65180
Release Repository	:	Wifi_M2M
Source Branch	:	/branches/rel_1500_19.7.3
Subversion Revision	:	r19062

2.2 Version information

WINC Firmware version	:	19.7.3
Host Driver version	:	19.7.3
Minimum driver version	:	19.3.0

Please note that the SVN revision advertised in the firmware serial trace will be **19057**.

```
(10)NMI M2M SW  VER 19.7.3 REV 19057
(10)NMI MIN DRV VER 19.3.0
(10)FW URL branches/rel_1500_19.7.3
(10)Built Oct 30 2020   03:59:06
```

2.3 Released components

The release contains documentation, sources and binaries.

2.3.1 Documentation overview

The Application manuals, Release notes and Software API guides can be found in the `doc/` folder of the release package.

Release Notes:

This document

Software APIs:

WINC1500_IoT_SW_APIs.chm

2.3.2 Binaries and programming scripts

The main WINC15x0 firmware binary is located in the `firmware` directory and named `m2m_aio_3a0.bin`. This can be flashed to a WINC device using, for example, a serial bridge application available from ASF.

An OTA image is provided in the `ota_firmware` directory named `m2m_ota_3a0.bin`.

2.3.3 Sources

Source code for the host driver can be found under the `src/host_drv` directory.

Source code for the tools, including crypto_lib, can be found under the `src/Tools` directory.

2.4 Release Comparison

Features in 19.6.5	Changes in 19.7.3
Wi-Fi STA	
• IEEE 802.11 b/g/n. • OPEN, WEP security. • WPA Personal Security (WPA1/WPA2). • WPA Enterprise Security (WPA1/WPA2) supporting: EAP-TTLSv0/MS-Chapv2.0 EAP-PEAPv0/MS-Chapv2.0 EAP-PEAPv1/MS-Chapv2.0 EAP-TLS EAP-PEAPv0/TLS EAP-PEAPv1/TLS	• Add WPA/WPA2 Enterprise option for TLS handshake certificate expiry checking mode
Wi-Fi Hotspot	
• Only ONE associated station is supported. After a connection is established with a station, further connections are rejected. • OPEN, WEP and WPA/WPA2 security modes. • The device cannot work as a station in this mode (STA/AP Concurrency is not supported).	• Fix to ensure DHCP offered address is consistent when STA disconnects/reconnects • Fix to close race condition when a STA disconnects and reconnects that could cause the WINC to disallow all further connection attempts.
• Wi-Fi direct client is not supported.	No change
WPS	
The WINC1500 supports the WPS protocol v2.0 for PBC (Push button configuration) and PIN methods.	No change
TCP/IP Stack	
The WINC1500 has a TCP/IP Stack running in firmware side. It supports TCP and UDP full socket operations (client/server). The maximum number of supported sockets is currently configured to 11 divided as: • 7 TCP sockets (client or server). • 4 UDP sockets (client or server).	• Improvements to socket closing code • Improvements to TCP Rx windowing • Address “Amnesia” vulnerabilities
TLS	
• Support TLS v1.2. • Client and server modes.	• Added TLS ALPN support • Fix verification of certificate chains which include ECDSA

Features in 19.6.5	Changes in 19.7.3
• Mutual authentication in client mode. • X509 certificate revocation scheme. • SHA384 and SHA512 support in X509 certificates processing. • Integration with ATECC508 (ECDSA and ECDHE support). • Supported cipher suites are: TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_256_CBC_SHA256 TLS_DHE_RSA_WITH_AES_128_CBC_SHA TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 TLS_DHE_RSA_WITH_AES_256_CBC_SHA TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (requires ECC508) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (requires ATECC508)	signatures
Networking Protocols	
DHCPv4 (client/server) DNS Resolver IGMPv1, v2. SNTP	• SNTP server allocated from DHCP is now cleared when switching between networks
Power saving Modes	
• M2M_PS_MANUAL • M2M_PS_DEEP_AUTOMATIC	No change
Device Over-The-Air (OTA) upgrade	
• Built-in OTA upgrade available. • Backwards compatible as far as 19.4.4, with the exception of: - Wi-Fi Direct (removed in 19.5.3) - Monitor mode (removed in 19.5.2)	No change
Wi-Fi credentials provisioning via built-in HTTP server	
Built-in HTTP/HTTPS (TLS server mode) provisioning using AP mode (Open, WEP or WPA/WPA2 secured).	No change

Features in 19.6.5	Changes in 19.7.3
Ethernet Mode (TCP/IP Bypass)	
Allow WINC1500 to operate in WLAN MAC only mode and let the host send/receive Ethernet frames.	• Ensure broadcast frames contain correct destination MAC address • Ensure NULL frames are sent to keep the AP connection alive during periods of low activity.
ATE Test Mode	
Embedded ATE test mode for production line testing driven from the host MCU.	• I/Q calibration values read and applied from efuse (in ATE firmware)
Miscellaneous features	
	• I/Q calibration values read and applied from efuse (in production firmware)

3 Test Information

Please refer to ticket W1500-735 for full details.

Testing was performed against the release candidate 19.7.3 against the following configuration(s):

H/W Version : WINC1510 Xplained module

Host MCU : ATSAMD21-Xplained

The following testing was performed in both open air and shielded environments;

1. General functionality including:
    1. HTTP Provisioning
    2. Station Mode
    3. AP Mode
    4. IP (TCP and UDP client and server)
    5. HTTP POST/GET
    6. WPS (PIN and PushButton methods)
    7. Over-The-Air (OTA) update functionality and robustness (with and without TLS)
2. TLS functionality including:
    1. RSA cipher-suites:
        i. TLS_RSA_WITH_AES_128_CBC_SHA
        ii. TLS_RSA_WITH_AES_128_CBC_SHA256
        iii. TLS_RSA_WITH_AES_128_GCM_SHA256
        iv. TLS_DHE_RSA_WITH_AES_128_CBC_SHA
        v. TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
        vi. TLS_DHE_RSA_WITH_AES_128_GCM_SHA256

        Testing uses 1024-bit, 2048-bit and 4096-bit server certificates, with a chain of 7 certificates of varying key lengths (1024,2048 and 4096 bit) leading to a 2048-bit root certificate.
    2. ECDSA ciphersuites:
        i. TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
        ii. TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

        Testing uses a NIST standard ECC P256 prime curve server certificate with two chains, one leading back to an ECDSA root certificate and the other leading to an RSA root certificate.
    3. Client authentication
3. Performance under interference
4. TCP/IP stack robustness testing
    1. Using an internal implementation of IPerf.
    2. Verification of multi socket functionality

4 Known issues

ID	Description
W1500-63	Occasionally WINC15x0 fails to receive an individual UDP broadcast frame when in M2M_PS_DEEP_AUTOMATIC powersave mode. Recommended workaround: **Use M2M_NO_PS powersave mode if reliability is preferred for UDP broadcast frames. Otherwise ensure the overlying protocol can handle the odd missing frame.**
W1500-108	The WINC15x0 cannot handle two simultaneous TLS handshakes, due to memory constraints. Recommended workaround: **When attempting to open two secure sockets in STA mode, the application should wait to be notified of the first one completing (succeeding or failing) before attempting the second one.**
W1500-325	1% of Enterprise conversations fail due to the WINC15x0 not sending an EAP response. The response is prepared and ready to send but does not appear on the air. After 10 seconds the firmware times-out the connection attempt and the application is notified of the failure to connect. Recommended workaround: Configure the authentication server to retry EAP requests (with interval < 10 seconds). **The application should retry the connection request when it is notified of the failure.**
W1500-369	When connected to certain access points, the WINC15x0 sometimes fails to roam when the access point changes channel. The issue is seen with these access points: Linksys E2500, Linksys E4200, Linksys 6500. The failures to roam are due to two issues: 1. Sometimes the access point takes a long time to start sending beacons or probe responses on the new channel, so it is not discoverable. 2. Sometimes the access point does not initiate the 4-way handshake (for WPA/WPA2 PSK reconnections). Recommended workaround: **On reception of M2M_WIFI_DISCONNECTED event, the application should attempt to discover the access point using `m2m_wifi_request_scan()` API.**

W1500-387	If an AP uses an 802.11 ACK policy of “No Ack”, then the WINC15x0 sometimes fails to receive 802.11b frames. Recommended workaround: **Avoid using an ACK policy of “No Ack”. If “No Ack” is used, ensure frames are sent at 802.11g or higher rates.**
W1500-397	70% of Enterprise connection requests fail with a TP Link Archer D2 access point (TPLink-AC750-D2). The access point does not forward the initial EAP Identity Response to the authentication server. The issue is bypassed by PMKSA caching (WPA2 only), so reconnection attempts will succeed. Recommended workaround: **The application should retry the connection request when it is notified of the failure.**
W1500-402	Occasionally during AP provisioning, after entering the credentials of the AP to connect to and pressing “connect”, an error will be returned even though provisioning was successful and the connection proceeds. Recommended workaround: **Add a delay in the application between receiving the provisioning info and connecting to the AP. Ignore the “Request Failed” message.**
W1500-510	Using TLS Server mode with a server certificate that is signed with a key size which differs from the key size contained within the certificate can cause the WINC to crash. Recommended workaround: **Only use a TLS Server certificate that is signed using the same key size as the key contained within the certificate.**
W1500-699	When using a driver pre – 19.6.0 with 19.7.2 firmware, upon failure to obtain a DHCP address the WINC will not trigger a WiFi Disconnection and notify the driver of the failure. Recommended workaround: **In this case of an older driver running with later firmware, the application should monitor the time taken to obtain a DHCP address, if it takes too long then it can decide whether to disconnect and try again.**

5 New Features

5.1 TLS Application Layer Protocol Negotiation

Support for Application Layer Protocol Negotiation (ALPN) has been added for SSL sockets. The main utility of this is to allow customer applications to use HTTP/2 over TLS.

The feature is available via new socket APIs `set_alpn_list()` and `get_alpn_index()`.

5.1.1 Using the ALPN feature

To use ALPN with an SSL socket, the customer application must call `set_alpn_list()` before connecting, then `get_alpn_index()` after the socket connection succeeds.

5.1.1.1 set_alpn_list()

The parameters to `set_alpn_list()` are the socket id and a list of one or more application layer protocols, in preference order.

The application layer protocols are represented by IANA strings, defined at https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xhtml#alpn-protocol-ids

The list provided to `set_alpn_list()` must consist of the appropriate IANA strings, separated by spaces.

For example, for HTTP/2 over TLS (1st preference) or HTTP/1.1 (2nd preference), the list provided to `set_alpn_list()` should be: “h2 http/1.1” (including NUL terminator). i.e:

```
0x68 0x32 0x20 0x68 0x74 0x74 0x70 0x2f 0x31 0x2e 0x31 0x00
```

5.1.1.2 get_alpn_index()

The parameter to `get_alpn_index()` is the socket id. The return value indicates which application layer protocol has been negotiated:

- 1: The negotiated protocol is the first one in the list that was provided to `set_alpn_list()`. (In the above example, this would be HTTP/2 over TLS.)
- 2: The negotiated protocol is the second one in the list that was provided to `set_alpn_list()`. (In the above example, this would be HTTP/1.1.)
- etc for return values greater than 0.
- 0: No negotiation occurred, for example because the TLS peer did not support ALPN.

Note that if negotiation occurs unsuccessfully (i.e. the peer does not support any of the protocols listed by the customer application), then the socket connection fails. The customer application can determine the cause of failed socket connections using the socket API `get_error_detail()`.

5.1.2 ALPN APIs

Further details of the APIs can be found at WINC1500_IoT_SW_APIs.chm.

5.2 Read and apply I/Q calibration values from efuse

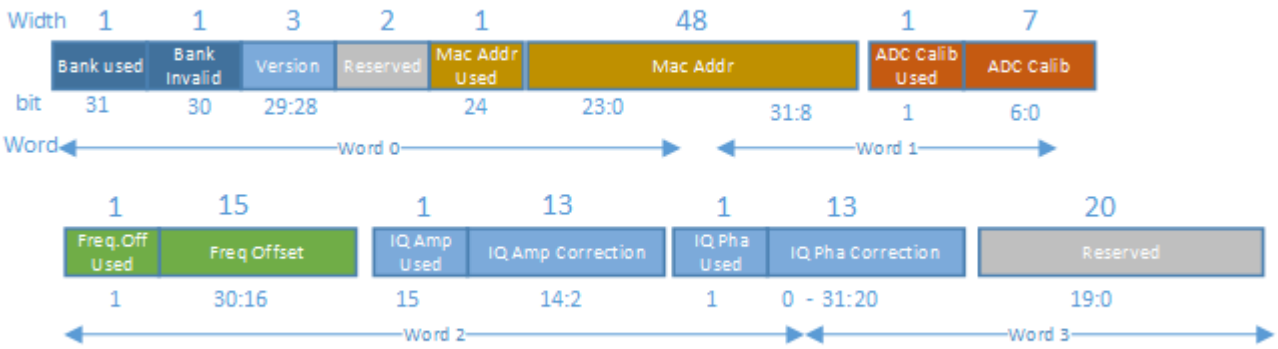
Transmitted RF is composed of a baseband signal modulated with an RF carrier signal.

The QAM modulated signal used in 802.11 Wi-Fi has two carrier components that have the same frequency but are phase shifted by 90 degrees - **I** and **Q** (“In-phase” and “Quadrature”). The transmitted signal is the sum of these two components.

In practice, there may be a slight imbalance between the amplitude and phase of the I and Q components. I/Q calibration compensates for this on a device basis, allowing for the best possible EVM.

5.2.1 Using the I/Q calibration feature

The WINC1500 has two registers, one for phase correction and one for amplitude correction that can be written to by firmware. The actual values programmed into these registers are obtained during production line testing and written to the WINC efuse as follows:



The scheme currently being used to store the WiFi MAC address and frequency offset value has been extended to store the I/Q Amplitude and Phase correction parameters. The existing method of invalidating a bank and writing a new bank to update any of the contents in the efuse can be used if the values need to be modified at any time.

Note – if I/Q calibration correction values are to be written to efuse, it is necessary to only use banks 1-5. Writing these values to bank 0 may have undesired consequences due to legacy bootrom code.

6 Fixes and enhancements

These are the fixes and enhancements since the previous released version (19.6.5).

6.1 Issues fixed

ID	Description
W1500-401	**M2M_WIFI_REQ_DHCP_FAILURE is not handled by drivers older than 19.6.0** If DHCP fails, the firmware sends this HIF message to the driver, but it is only supported by drivers since 19.6.0. Fixed: Firmware no longer sends this HIF message if the driver is older than 19.6.0.
W1500-469	**Roaming reports wrong state after disconnection** When connecting to an AP and disconnecting, after reconnecting, if a roaming event occurs, the WINC would report the wrong state (“connected” instead of “roamed”). Fixed: The roaming state is now reported correctly in all cases.
W1500-474	**Unsafe driver callback to function pointer received over HIF** After sending a ping, the driver receives a response over the HIF and calls a function pointer which is contained in that response. There is no guarantee that the function pointer has not been modified by an attacker. Fixed: The driver calls a function pointer which is stored locally instead.
W1500-491	**Provisioning will return 404 on some error paths if legacy HTTP files are in use** The WINC running in provisioning mode using old HTTP files (I.e. the WINC was originally flashed with a release of 19.6.1 or earlier) will return an HTTP 404 error in some error cases such as submitting the page with a blank SSID. Fixed: The WINC1500 returns a correct error page in these cases.
W1500-498	**spi_flash_enable() disables SPI bus to external flash after every call** The WINC driver disables the SPI bus to external flash after calls to this function. Fixed: The driver now leaves the SPI bus enabled when returning from this function.
W1500-520	**Flush the PMK cache at set time if the time diff is above the PMK cache lifetime** When the application sets the time or time is obtained from the NTP server, the WINC does not perform a check to verify if the contents of the PMK cache should be invalidated or not. Fixed: When setting the time, verify if the PMK cache contents need to be invalidated.
W1500-595	**After upgrading fw from <19.6.4, default Enterprise connect fails** A change in format of Enterprise connection parameters means that parameters stored in flash by fw older than 19.6.4 are not parsed correctly when retrieved from flash by fw 19.6.4 or 19.6.5. Fixed: All formats of Enterprise connection parameters are correctly parsed when retrieved from flash.
W1500-599	**m2m_wifi_request_scan_ssid_list API scans broadcast instead of directed scan** When using the m2m_wifi_request_scan_ssid_list API, the WINC scans for all nearby SSIDs, the intention for this API was to scan specific SSIDs as provided via the API.

	Fixed: Correct the SSID included in the probe requests.
W1500-606	**Race condition causes "AP Full" error in AP mode** When running the WINC as an AP, a race condition can occur between one station disconnecting and a second station connecting, which could cause some frames to be discarded and cause the WINC to "lock" by not allowing any more stations to connect. Fixed: Protect the disconnection sequence so that it is not interrupted.
W1500-607	**WINC does not update the remote peer IP address when running TLS server** When running in TLS server mode, the WINC does not populate the IP address of the remote peer device before informing the host. Fixed: WINC now populates the IP address even when running in TLS server mode.
W1500-608	**Bypass/ETH mode fails to send NULL frames** When running in ETH mode (bypassing the WINC onboard network stack), the WINC fails to send NULL frames to keep the AP connection alive during periods of no traffic. Fixed: NULL frames are now transmitted
W1500-609	**Scan does not display different bssid with same ssid in scan results** If two different APs exist configured with the same SSID and within range from the WINC, when instructing the WINC to perform a scan, the scan results only include one of the APs - typically the one with the strongest signal. Fixed: If multiple APs with same SSID are picked up by the scan, display each one as an individual result.
W1500-610	**Ignore beacons from other networks when performing directed scan** When performing a directed scan with the WINC, via m2m_wifi_request_scan_ssid_list, sometimes beacons are picked up in the scan causing a scan entry to be added to the results even if it wasn't included in the list of SSIDs to look for. Fixed: Filter the beacons from different networks when running directed scan.
W1500-659	**WINC attempts to process TLS certificate chains which contravene its signature algorithms extension** If the host processor does not indicate support for ECC processing, the WINC excludes ECDSA from its signature algorithms hello extension. If the server sends an ECDSA signature (in contravention of this), the WINC attempts to process it, passing it up to the host processor. Fixed: WINC rejects certificate chains which contravene its signature algorithms extension.
W1500-671	**Signature verification broken for ECDSA-SHA, ECDSA-SHA224, ECDSA-SHA384 and ECDSA-SHA512** ECDSA signature verification only works if the hash digest is 32 bytes (i.e. SHA256). Fixed: Hashes with shorter/longer digests are handled correctly for ECDSA verification.
W1500-687	**Host File Download fails when file size is not aligned to a 4-byte boundary** Files with a size which is not a multiple of 4 bytes fail to download completely - the last few bytes go missing. Fixed: WINC now handles both files aligned and not aligned to a 4-byte boundary.
W1500-691	**Incorrect parsing of ECDSA signatures**

	If individual coordinates are shorter or longer than the curve's field size they are processed incorrectly. Fixed: Short coordinates are prepended with 0's; long coordinates are rejected.
W1500-713	**Repeated roaming causes memory leak** A memory leak was found in the firmware code path that handles roaming. This could leave the WINC low on resources if it roamed several times. Fixed: The leak is now fixed.
W1500-714	**TLS handshake failure with Truncated 'Issuer' field in device certificate** Certain TLS certificates, including one issued by AWS which has a truncated "issuer" field fail to successfully complete the TLS handshake. Fixed: Sometimes the TLS "finish" message is split between two buffers internally in the WINC, and a problem was found and fixed when this occurred.
W1500-717	**Fix CERT Amnesia vulnerabilities** CERT released a suite of vulnerabilities that affect the uIP stack used internally by the WINC1500. Fixed: Address all relevant vulnerabilities in WINC1500 firmware.
W1500-718	**Transmitted TCP stream from the WINC1500 can suddenly stop** When the WINC1500 sends a TCP packet that hits maximum transmit retries, it tries once more but with an incorrect 802.11 sequence number which means subsequent messages get discarded by the AP. Fixed: Stop the WINC1500 from re-sending the TCP packet with the erroneous 802.11 sequence number

6.2 Enhancements

ID	Description
W1500-456	**Simplify HFD SPI read to remove dependency on driver initialisation** The earlier implementation would require the HFD to retrieve the file handler from the firmware, this means that the driver would need to be loaded in order to set up the correct callbacks to handle receiving the handler ID via the HIF. This was not practical, and this dependency was removed. In this version, the application can store the handler ID in the flash of the host, and issue a HFD SPI read without the need to load the driver.
W1500-561	**Enterprise option for setting certificate expiry checking mode** In previous versions, the certificate expiry checking mode could be set globally for all subsequent connections (both secure socket and Enterprise) using the API sslEnableCertExpirationCheck(). This version allows that setting to be overridden for Enterprise connections using the API m2m_wifi_1x_set_option() with WIFI_1X_TIME_VERIF_MODE.
W1500-616	**Add API to make XO sleep workaround runtime configurable** It has been found that some WINC1500 boards exhibit a problem with the crystal (XO) that causes the WINC to sometimes get stuck when waking from deep sleep. A new API has been provided that controls the XO behaviour during WINC sleep: `m2m_wifi_enable_XO_during_sleep(uint8 bXOSleepEnable)` This API can be called at any time after the WINC has been initialised. A `bXOSleepEnable` value of 1 will ensure the XO is kept on during WINC sleep, a value of 0 will ensure it is switched off. The WINC defaults to switching off the XO during sleep (`bXOSleepEnable` set to 0).
W1500-621	**Add `get_error_detail()` socket API** An application can call this new socket API to obtain more detailed information about an error when notified of a socket failure via the `SOCKET_MSG_CONNECT` or `SOCKET_MSG_RECV.` For further details refer to the Software API Guide.
W1500-656	**Add option to stop scan on first result** Previously, every time a scan was requested, the WINC would scan for a pre-determined amount of time and would never return earlier, even if it had found the SSID it was looking for, causing higher power consumption and longer waiting times. This version introduces a new scan option which lets the application configure the WINC to return the scan results upon finding the first SSID (useful for example for directed scan).
W1500-672	**New m2m_ssl APIs for ECDSA verification** New API m2m_ssl_retrieve_next_for_verifying() is an improvement over m2m_ssl_retrieve_cert(), allowing the application to indicate the lengths of the buffers that it is providing for receiving the verification signature and value. New API m2m_ssl_stop_retrieving() is an improved name for m2m_ssl_stop_processing_certs(). The old APIs are still present for legacy applications.

W1500-708	**Improve API documentation for socket receive behaviour in driver** Improved documentation for structure tstrSocketRecvMsg to make it clear that applications should not attempt to make use of u16RemainingSize for managing data flow.

7 Terms and Definitions

Term	Definition
AES	Advanced Encryption Standard
AJAX	Asynchronous JavaScript and XML
AKM	Authentication and Key Management
ARP	Address Resolution Protocol
ATE	Automated Test Equipment
BSS	Basic Service Set
CBC	Cyclic Block Chaining
DHCP	Dynamic Host Control Protocol
DHE	Diffie-Hellman Ephemeral
DNS	Domain Name Server
DTIM	Directed Traffic Indication Map
EAP	Extensible Authentication Protocol
EAPOL	EAP Over LAN
ECC	Elliptic Curve Cryptography
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
EEPROM	Electrically Erasable Programmable Read Only Memory
ESD	Electrostatic Discharge
EVM	Error Vector Magnitude
HIF	Host Interface
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electronic and Electrical Engineers
MAC	Media Access Control
OTA	Over The Air update
PEAP	Protected Extensible Authentication Protocol
PLL	Phase Locked Loop
PMK	Pair-wise Master Key
PSK	Pre-shared Key
QAM	Quadrature Amplitude Modulation
RSA	Rivest-Shamir-Adleman (public key cryptosystem)
RSN	Robust Security Network
RSSI	Receive Strength Signal Indicator
SHA	Secure Hash Algorithm
SNTP	Simple Network Time Protocol
SPI	Serial Peripheral Interface
SSID	Service Set Identifier
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TIM	Traffic Indication Map
TLS	Transport Layer Security

Term	Definition
WEP	Wired Equivalent Privacy
WINC	Wireless Network Controller
WLAN	Wireless Local Area Network
WMM™	Wi-Fi Multimedia
WMM-PS™	Wi-Fi Multimedia Power Save
WPA™	Wi-Fi Protected Access
WPA2™	Wi-Fi Protected Access 2 (same as IEEE 802.11i)