WINC1500 Software

Release Notes

VERSION : **19.6.5**

DATE : **NOV, 2019**

Abstract

This document presents an overview of the WINC15x0 firmware release version 19.6.5, 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 Upgrade a TCP socket to TLS after connection as a client 12
5.2 WPA/WPA2 Enterprise TLS options 13
6 Fixes and enhancements 14
6.1 Issues fixed 14
6.2 Enhancements 15
7 Terms and Definitions 16

1 Introduction

This document describes the WINC15x0 version 19.6.5 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 19.6.5 firmware to be suitable for production release

2 Release summary

2.1 Auditing information

Master Development Ticket	:	`https://jira.microchip.com/projects/W1500/versions/59110`
Release Repository	:	`Wifi_M2M`
Source Branch	:	`/branches/rel_1500_19.6.5`
Subversion Revision	:	**`r18277`**

2.2 Version information

WINC Firmware version	:	19.6.5
Host Driver version	:	19.6.4
Minimum driver version	:	19.3.0

Please note that the SVN revision advertised in the firmware serial trace will be **18275**.

```
(10)NMI M2M SW  VER 19.6.5 REV 18275
(10)NMI MIN DRV VER 19.3.0
(10)FW URL branches/rel_1500_19.6.5
(10)Built Nov 28 2019   16:24:09
```

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 19.6.3	Changes in 19.6.5
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 options for phase 1 TLS handshake: ○ Bypass server authentication ○ Specify root certificate for server authentication ○ Session caching
Wi-Fi Hotspot	
• Only ONE associated station is supported. After a connection is established with a station, further connections are rejected. • OPEN and WEP, 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
• 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
TLS	
• Support TLS v1.2. • Client and server modes.	• Add feature to allow a TCP socket to connect (as client) without TLS, then upgrade to TLS later • Fix to make client certificate type independent of cipher

Features in 19.6.3	Changes in 19.6.5
• 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)	suite type
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	• None
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 WPA2 secured).	No change

Features in 19.6.3	Changes in 19.6.5
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
ATE Test Mode	
Embedded ATE test mode for production line testing driven from the host MCU.	No change.
Miscellaneous features	
	• Fix TX Power mode API

3 Test Information

Please refer to ticket W1500-585 for full details.

Testing was performed against the release candidate 19.6.5 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.**
W1500-177	Under high interference and high data throughput (TCP/UDP), the WINC15x0 occasionally runs out of memory for receiving data and does not recover. This occurred 4 times during 9 hours of high interference high throughput rx/bidirectional testing. Recommended workaround: **Close all sockets then retry the data transfer.**
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.**
W1500-370	When provisioning the WINC15x0 using a mobile phone, 5% of provisioning attempts cause an error message “Request Failed” to pop up on the phone, even though the provisioning has succeeded. Recommended workaround: **Ignore the “Request Failed” message.**

W1500-381	When connecting to a TL-WR841N router, data transfer is sometimes unavailable until several seconds after DHCP. Occasionally the data-plane is never established. Recommended workaround: **If DHCP completes but data transfer fails, disconnect and reconnect to the router.**
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.**
W1500-510	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.**

5 New Features

5.1 Upgrade a TCP socket to TLS after connection as a client

5.1.1 Overview

This feature allows a TCP socket to be created in one of the following configurations:

i) Not secure

ii) Secure

iii) Potential to be secure

Configuration (iii) “potential to be secure” is new in this release.

A TCP socket created with the “potential to be secure” configuration can connect (as a client), but will remain insecure until the application calls the secure() API. In the meantime, the socket is a plain TCP socket and the application may send and receive data unencrypted.

This configuration allows the application to implement HTTPS via a proxy tunnel, as described in https://tools.ietf.org/html/rfc7230.

Please note, a 19.6.4 driver or above is required to access the new functionality in this feature.

5.1.2 Usage

These steps describe how to make use of the feature:

1. Create a socket using the socket() API, with the u8Type parameter set to SOCK_STREAM and the u8Config parameter set to SOCKET_CONFIG_SSL_DELAY.
2. Open a TCP connection on the socket using the connect() API and await notification of successful connection via the SOCKET_MSG_CONNECT message.
3. Send and receive data unencrypted over the TCP connection, using the send() and recv() APIs. As an example, this data may comprise an HTTP CONNECT message, in order to set up a proxy tunnel to a secure server.
4. Make the socket secure by calling the secure() API. This initiates a TLS handshake. Await notification of successful securing via the SOCKET_MSG_SECURE message.

Notes:

- If either connecting or securing fails, the application must call the close() API to close the socket. Even if the failure is at the securing stage, the socket must be closed in this way.
- If a TCP connection is established via listening and accepting a connection, the connection cannot be made secure via the secure() API.

Please refer to *WINC1500_IoT_SW_APIs.chm* for full details of these APIs and their parameters.

5.2 WPA/WPA2 Enterprise TLS options

5.2.1 Overview

All Enterprise authentication methods use a TLS handshake in phase 1 (or in the single phase in the case of EAP-TLS).

New set/get option APIs allow the application to configure some details of this TLS handshake.

Please note, a 19.6.4 driver or above is required to access the new functionality in this feature.

5.2.2 Available options

Please refer to *WINC1500_IoT_SW_APIs.chm* for full details of these options, including default values.

5.2.2.1 Bypass Server Authentication

This option allows server authentication to be bypassed during Enterprise connection. Bypassing server authentication is not recommended as it allows the phase 2 data to be sent to a rogue server.

5.2.2.2 Specify Root Certificate for Server Authentication

This option allows the server to be verified against a particular root certificate out of the many which may be present in the WINC15x0 flash store. This is recommended in order to truly verify the server.

5.2.2.3 TLS Session Caching

This option allows TLS session resume capability to be enabled or disabled.

TLS session resume allows a much shorter Enterprise conversation when connecting to access points that share the same authentication server. Up to 4 entries can be stored in the WINC1500 TLS session cache.

TLS session resume requires the authentication server to be configured for TLS session caching. For a hostapd server, this is done by adding “tls_session_lifetime=3600” to the hostapd.conf file.

Note that this option is ignored for EAP-PEAPv0 authentication methods, as the WINC implementation of EAP-PEAPv0 does not support TLS session resume.

5.2.3 Usage

Each option may be set via the m2m_wifi_1x_set_option() API and retrieved via the m2m_wifi_1x_get_option() API.

The option settings apply to all subsequent connection attempts via the WPA/WPA2 Enterprise connect APIs (i.e. m2m_wifi_connect_1x_mschap2() and m2m_wifi_connect_1x_tls()).

Connection attempts via the m2m_wifi_default_connect() API use the option settings which were in place at the time of the original connection.

Please refer to *WINC1500_IoT_SW_APIs.chm* for full details of these APIs.

6 Fixes and enhancements

6.1 Issues fixed

These are the fixes and enhancements since the previous released version (19.6.3)

ID	Description
W1500-452	**SNTP server obtained via DHCP is not cleared when switching between networks** Fixed so the DHCP subnet is now cleared for the new connection.
W1500-344	**Unable to connect to AP when setting power mode via TX power API `m2m_wifi_set_tx_power()`** A firmware problem with this API was found and fixed.
W1500-551	**In AP mode, WINC starts offering 0.0.0.0 as subnet when the STA is power-cycled and reconnects** Firmware bug found and fixed
W1500-558	**In bypass (eth) mode, the destination MAC address of broadcast frames is incorrectly replaced with the WINC MAC address** Firmware bug found and fixed.
W1500-555	**WINC expects TLS client certificate to use the same type of key as the server certificate** The WINC expected the client and server certificates to both be RSA or both be ECDSA. This is now fixed so they can be different.
W1500-592	**Application loses TCP Rx data if it does not read entire firmware buffer** This is now fixed, so partially-read buffers remain in the firmware until the application issues read requests via recv().

6.2 Enhancements

ID	Description
W1500-43	**Improve the TCP windowing algorithm** Allow the TCP window to shrink and grow more effectively with the amount of resource currently available to the stack.

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
ESS	Extended Service Set (infrastructure network)
HIF	Host Interface
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electronic and Electrical Engineers
JSON	JavaScript Object Notation
MIB	Management Information Base
OTA	Over The Air update
PEAP	Protected Extensible Authentication Protocol
PLL	Phase Locked Loop
PMK	Pair-wise Master Key
PSK	Pre-shared Key
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
TCP	Transmission Control Protocol
TIM	Traffic Indication Map
TLS	Transport Layer Security
TTLS	Tunneled TLS

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)