Morse Micro



Offload Features
APPNOTE-21

Copyright © Morse Micro

Table of Contents

1 Overview 4
1.1 Definitions 4
1.2 Restrictions 4
2 DHCP Offload 5
2.1 Scope 5
2.1.1 Capabilities 5
2.1.1.1 DHCP Discovery 5
2.1.1.2 Lease Renewal 5
2.1.1.3 Lease Application 5
2.1.2 Restrictions 5
2.2 Theory of Operation 6
2.2.1 Firmware Actions 6
2.2.2 Driver Actions 7
2.2.3 DHCP Lease Update Script Actions 7
2.3 How to Configure 8
2.4 How to Verify 8
3 ARP Response Offload 9
3.1 Scope 9
3.2 Theory of Operation 9
3.2.1 Firmware Actions 9
3.2.2 Driver Actions 10
3.3 How to Configure 10
3.4 How to Verify 10
4 ARP Refresh Offload 12
4.1 Scope 12
4.2 Theory of Operation 12
4.3 How to Configure 12
4.4 How to Verify 13
5 BSS Keepalive Offload 14
5.1 Scope 14
5.2 Theory of Operation 14
5.3 How to Configure 14
5.4 How to Verify 15
6 TCP Keepalive Offload 16
6.1 Scope 16
6.2 Theory of Operation 16
6.2.1 Other Considerations 16

6.3 How to Configure 17
6.4 How to Verify 18
7 ICMP Response Offload 19
7.1 Scope 19
7.2 Theory of Operation 19
7.2.1 Firmware Actions 19
7.2.2 Restrictions 19
7.3 How to Configure 19
7.4 How to Verify 20
8 Packet Whitelist Filtering 21
8.1 Scope 21
8.2 Theory of Operation 21
8.3 How to Configure 22
8.4 How to Verify 22
9 Revision History 24

1 Overview

This document describes Morse Micro chip features that can significantly reduce power consumption by offloading some network traffic processing from the host processor to the Morse Micro chip. Some features operate only while the host processor is hibernating or powered off (also known as **standby** or **sleep mode**), while others operate continuously.

These features are available in MM6108 and MM8108 from Software Release 1.17.0 onward.

1.1 Definitions

DUT (Device Under Test): the Linux-based host device with an MM6108 or MM8108 chip that is being tested.

FNO (Full Network Offload): a mode in which the Morse Micro chip runs the Wi-Fi stack and the IP network stack, and the host communicates with it using the MMAGIC interface.

FullMAC mode: A mode in which the Wi-Fi MAC layer is implemented within the Morse Micro chip itself. The chip handles association, authentication, and connection management autonomously, without relying on the host processor. FullMAC mode is not supported on MM6108 or in software release packages prior to 2.0.0 (OpenWrt release 3.1.1).

modparam: a parameter provided to a Linux kernel module during initialization to define specific operational settings.

Standby mode: Host sleep mode, where the host processor is in hibernation or powered off, but the firmware is still operational.

1.2 Restrictions

Offload features are not supported for stations using 4-address mode.

These features apply to Linux hosts using the morse driver. They are not available in FNO firmware, where DHCP, ARP, ICMP, TCP, and BSS keepalives are handled by the network stack on the Morse Micro chip.

2 DHCP Offload

2.1 Scope

The DHCP Offload feature offloads some DHCP client processing from the host processor to the Morse Micro chip, primarily so leases can be renewed without waking the host processor, reducing the device's overall power consumption.

The Dynamic Host Configuration Protocol is described in RFC 2131. The Morse Micro chip supports only selected parts.

2.1.1 Capabilities

2.1.1.1 DHCP Discovery

Automatically obtain a lease (IP address, subnet mask, router address, and DNS server).

2.1.1.2 Lease Renewal

Automatically renew the lease.

2.1.1.3 Lease Application

Send a message to the host processor on lease update (acquisition, renewal, and expiry) indicating the current lease parameters and addresses.

2.1.2 Restrictions

- The DHCP server must be on the same subnet (no relay agents).
- A DHCP client must not be running on the station's host device.
- The host processor must apply the obtained IP address to the Wi-Fi HaLow interface.
- DHCPv6 is not supported (IPv4 only).
- The DHCP client does not send an ARP request to ensure the IP address is not taken prior to accepting a lease. This is the server's responsibility.
- The lease is not released if cleared by a user command.
- An OFFER or ACK without a subnet mask, server ID, or lease time is rejected.
- Only the first router and the first DNS server are kept.

2.2 Theory of Operation

This feature is triggered when the following conditions are met:

- The device is in STA mode.
- DHCP offload is enabled in the driver.

2.2.1 Firmware Actions

Upon receiving a DHCP packet, the Morse Micro chip performs the following actions and consumes the packet. The chip will not forward DHCP packets to the host processor.

DHCP_OFFER (2)

1. Extract and store the following DHCP option field values:
    - Server ID (54)
    - IP lease time (51)
    - Renewal time (58)
    - Rebinding time (59)
    - Subnet mask (1)
    - Router (3)
    - DNS (6)
2. Send a DHCP request packet to the DHCP server.

DHCP_ACK (5)

1. Extract and store the DHCP option field values, as in DHCP_OFFER.
2. If Renewal and/or Rebinding time are missing, they are derived from the lease time plus a random fuzz value.
3. Send a MORSE_CMD_ID_EVT_DHCP_LEASE_UPDATE event to the host processor.

DHCP_NAK (6)

The server has rejected the request, so it clears any current lease, then restarts discovery.

Other

Ignore the packet.

Lease renewal

At the renewal time (T1), the firmware sends a DHCP request directly to the server that granted the lease. If no ACK is received by the rebinding time (T2), it broadcasts a DHCP request to any server. A lease with an infinite lease time is never renewed.

If the lease expires without being renewed, the firmware will:

1. Send a MORSE_CMD_ID_EVT_DHCP_LEASE_UPDATE event to the host processor to indicate that the lease is no longer valid.
2. Send a DHCP discovery packet to re-acquire a lease.

Rapid Commit (option 80) is also supported in the firmware. An ACK in reply to the DISCOVER will bind without sending a REQUEST.

2.2.2 Driver Actions

Upon receipt of a `MORSE_CMD_ID_EVT_DHCP_LEASE_UPDATE` event from the Morse Micro chip, the morse driver in the host processor will invoke a script from either the default path (`/morse/scripts/dhcpc_update.sh`) or the path specified in the `dhcpc_lease_update_script` modparam. The release package includes a default script that can be modified as needed. It is called with the following parameters:

```
STA1# dhcpc_update.sh <interface> <IP addr> <subnet mask> <router addr> <DNS addr>
```

Example

```
STA1# dhcpc_update.sh wlan0 192.168.1.2 255.255.255.0 192.168.1.1 192.168.1.1
```

2.2.3 DHCP Lease Update Script Actions

The default script will execute the following Linux commands when provided with valid DHCP lease parameters:

```
ip addr add $IP_ADDR/$CIDR dev $IFNAME broadcast $(broadcast $IP_ADDR $CIDR)
ip route add default via $ROUTER_ADDR dev $IFNAME metric 200
rm -f /tmp/resolv.conf
echo "nameserver $DNS_ADDR" > /tmp/resolv.conf
echo "$(basename $0): nameserver $DNS_ADDR added to /tmp/resolv.conf" > /dev/kmsg
if [ ! -f "/etc/resolv.conf" ]; then
      ln -s /tmp/resolv.conf /etc/resolv.conf
fi
```

If the lease has expired, all addresses will be set to 0.0.0.0, and the script will execute the following command:

```
ip -4 addr flush dev $IFNAME
```

The script exits early if the address already exists, so renewals don't re-apply the route or DNS. It only adds the route and DNS if they're non-zero.

2.3 How to Configure

The DHCP Offload feature is enabled by setting the `enable_dhcpc_offload` modparam when loading the morse driver in the host processor:

```
STA1# modprobe morse enable_dhcpc_offload=1 ...
```

Alternatively, the `morse` driver can be compiled with the `MORSE_DHCP_OFFLOAD` Linux kernel configuration parameter set to **y**. Setting enable_dhcpc_offload to 0 can then be used to disable DHCP Offload.

2.4 How to Verify

On the DUT, verify that the device obtains an IP address from the DHCP server within 10 seconds after association, when configured for DHCP and no other DHCP client is running.

On the DUT, check the following statistics to verify that the Morse Micro chip handled DHCP exchanges as expected:

```
STA1# morse_cli stats -f DHCP
DHCP packets received                          : 4
DHCP discoveries sent                          : 1
DHCP requests sent                             : 1
DHCP lease updates sent                        : 1
DHCP leases fully expired                      : 0
DHCP leases obtained                           : 1
DHCP leases renewed/rebound                    : 0
```

These counters will not increment if the feature is not enabled.

DHCP packets received: The total number of DHCP packets received (and not forwarded to the host processor).

DHCP discoveries sent: The number of DHCP discovery packets sent.

DHCP requests sent: The number of DHCP requests sent.

DHCP lease updates sent: The number of lease update events sent to the host processor.

DHCP leases fully expired: The number of DHCP leases that have expired.

DHCP leases obtained: The number of DHCP leases obtained.

DHCP leases renewed/rebound: The number of DHCP leases renewed or rebound.

3 ARP Response Offload

3.1 Scope

The ARP Response Offload feature allows the Morse Micro chip to respond directly to ARP request packets without involving the host processor.

3.2 Theory of Operation

This feature is triggered when all of the following conditions are met:

- The device is in STA mode.
- The morse driver `enable_arp_offload` modparam is set.
- The Wi-Fi HaLow interface has one or more IPv4 addresses configured.
- The device receives an IPv4 ARP request on the Wi-Fi HaLow interface.

3.2.1 Firmware Actions

The Morse Micro chip takes the following actions.

When a `MORSE_CMD_ID_ARP_OFFLOAD` command is received:

- If no IPv4 addresses are configured:
  - ARP offload and ICMP Response Offload functionality are disabled (ARP and ICMP packets will be forwarded to the host processor for processing).
- If one or more IPv4 addresses are configured:
  - ARP offload and ICMP Response Offload functionality are enabled.
  - If associated, a gratuitous ARP broadcast packet is transmitted to announce the IP address to the network.

When an ARP request packet (the LLC header contains `LLC_SNAP_PROTOCOL_ARP` (0x0806)) is received:

- If the ARP request is for the primary (first) configured IPv4 address, the Morse Micro chip sends an ARP response to the requestor instead of forwarding the ARP request to the host processor.
- If the ARP request packet is for any of the other (max three) configured IPv4 addresses, the chip will forward the ARP request packet to the host for processing.
- If the chip receives an ARP request packet for any other IPv4 address, it will drop the packet.

3.2.2 Driver Actions

Whenever the station associates or an IPv4 address on the Wi-Fi HaLow interface is added or removed, the morse driver sends the interface's IPv4 addresses (up to four) to the Morse Micro chip.

3.3 How to Configure

The ARP Response Offload feature is enabled by setting the `enable_arp_offload` modparam when loading the morse driver in the host processor:

```
STA1# modprobe morse enable_arp_offload=1 ...
```

Alternatively, the `morse` driver can be compiled with the `MORSE_ARP_OFFLOAD` Linux kernel configuration parameter set to **y**. Setting enable_arp_offload to 0 can then be used to disable ARP Offload.

The ICMP Response Offload feature is automatically enabled when this feature is enabled.

3.4 How to Verify

On the AP, delete the ARP table entry for the DUT:

```
AP# arp -d 192.168.1.2
```

From the AP, ping the DUT and verify that it is responsive:

```
AP# ping -c3 192.168.1.2
PING 192.168.1.2 (192.168.1.2): 56 data bytes
64 bytes from 192.168.1.2: seq=0 ttl=64 time=112.309 ms
64 bytes from 192.168.1.2: seq=1 ttl=64 time=33.814 ms
64 bytes from 192.168.1.2: seq=2 ttl=64 time=57.527 ms


--- 192.168.1.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 33.814/67.883/112.309 ms
```

Use a sniffer tool such as tshark on the DUT to verify that ARP packets are not being received on the host processor:

```
STA1# tshark -i wlan0
```

On the DUT, check the following statistics to verify that packets were processed as expected:

```
STA1# morse_cli stats -f ARP
ARP requests detected                          : 123
ARP requests answered                          : 16
ARP requests dropped                           : 107
ARP requests let through                       : 0
ARP table updated                              : 1
ARP gratuitous generated                       : 1
Periodic ARP frames sent                       : 0
```

ARP requests detected: The number of ARP request packets received by the Morse Micro chip. This counter will not increment if the feature is not enabled.

ARP requests answered: The number of ARP request packets received by the chip and responded to with an ARP response packet.

ARP requests dropped: The number of ARP request packets received by the chip that did not match any of the configured IP addresses, and therefore were dropped.

ARP requests let through: The number of ARP request packets received by the chip that matched one of the secondary IP addresses and were passed on to the host processor.

ARP table updated: The number of changes to the Wi-Fi HaLow interface's IP address configuration that caused updates to the chip's list of managed addresses.

ARP gratuitous generated: The number of gratuitous ARP packets sent, both on IP address changes and by ARP Refresh Offload.

Periodic ARP frames sent: The number of ARP request frames and gratuitous ARP frames sent. This counter is for the ARP Refresh Offload feature, described below.

4 ARP Refresh Offload

4.1 Scope

The ARP Refresh Offload feature periodically sends an ARP packet to the AP so that the AP's ARP table entry for the station does not time out.

4.2 Theory of Operation

This feature is triggered when the following conditions are met:

- The device is in STA mode.
- The morse driver `enable_arp_offload` modparam is set.
- The Wi-Fi HaLow interface has one or more IPv4 addresses configured.
- The `morse_cli arp_periodic_refresh` command has been used to enable the feature.

While the station is associated, the Morse Micro chip sends an ARP request for the configured IP address to the AP at the configured interval, or a gratuitous ARP for the station's primary IP address if -g is used. ARP replies are not handled by the chip and are passed to the host processor.

4.3 How to Configure

ARP Response Offload must be enabled (see section 3.3 How to Configure).

Enable periodic ARP refresh by using the following command:

```
STA1# morse_cli arp_periodic_refresh --help
arp_periodic_refresh [-g] -t <period> [-d <dest IP>]
    Configure the firmware to send a periodic ARP packet
    -t <period>      Period in seconds between ARP transmissions (0 to disable)
    -d <dest IP>     IP in dotted decimal notation - target protocol address
                     field of the ARP request
    -g               Send as a gratuitous ARP (GARP) instead of an ARP request
```

-d <dest IP> is required when enabling.

Gratuitous ARP packets are more efficient than ARP request packets because they do not cause any peer device to send an ARP response packet. However, many devices will ignore gratuitous ARP packets unless specifically configured to accept them, for example, using the following command on a Linux device.

```
AP# sysctl net.ipv4.conf.all.arp_accept=1
```

The following example configures the Morse Micro chip to send an ARP request to 192.168.1.1 every 10 seconds.

```
STA1# morse_cli arp_periodic_refresh -d 192.168.1.1 -t 10
```

4.4 How to Verify

After enabling periodic ARP refresh on the DUT, delete the ARP table entry for its IP address on the AP. Wait for the refresh period, then verify that the ARP table entry has been re-added.

On the DUT, check the following statistics to verify that packets were processed as expected:

```
STA1# morse_cli stats -f ARP
ARP requests detected                          : 24
ARP requests answered                          : 13
ARP requests dropped                           : 11
ARP requests let through                       : 0
ARP table updated                              : 1
ARP gratuitous generated                       : 1
Periodic ARP frames sent                       : 13
```

Section 3.4 How to Verify describes these counters.

5 BSS Keepalive Offload

5.1 Scope

The BSS Keepalive Offload feature lets the Morse Micro chip maintain 802.11 association by handling BSS keepalive functionality in the chip, which wpa_supplicant on the host processor usually handles. The aim is to keep the station associated without requiring action from the local host processor.

If the inactivity timer is set in the AP's hostapd configuration file (`ap_max_inactivity`) and the AP has received no packets from a station in that many seconds, it sends one or more empty data frames to the station. The AP disassociates the station if it does not acknowledge any frames. This feature makes the Morse Micro chip send a frame to the AP within each BSS max idle period, so the AP does not consider the station inactive.

5.2 Theory of Operation

When configured, the BSS Keepalive Offload feature transmits keepalive packets directly from the Morse Micro chip without the host processor being involved, which is useful in the following cases:

- When the host processor is suspended or idle, so it does not need to wake up to send keepalives.
- When in Standby mode (wpa_supplicant is not running on the host processor).
- To keep the AP and station in sync if either fails to receive a disassociation or deauthentication frame from the peer, because data frames from an unknown source will trigger clean-up and reassociation.
- When TWT is operational, keepalive packets from wpa_supplicant could be delayed. When offloaded to the chip, the chip will ensure that empty data frames are sent before the max BSS idle time, even if that means waking up between TWT SPs.

5.3 How to Configure

`wpa_supplicant_s1g` must be built with `CONFIG_MORSE_KEEP_ALIVE_OFFLOAD` (enabled by default).

BSS Keepalive Offload can be enabled with the following configuration option in `wpa_supplicant_s1g.conf`:

```
vendor_keep_alive_offload=1
```

Alternatively, it can be enabled using the following command:

```
STA1# morse_cli keepalive --help
keepalive [-a] <period>
    Set the BSS max idle period
    <period>      BSS idle period (1000 TUs) after which a keepalive will be sent
    -a            Interpret idle period as per IEEE802.11ah spec
```

The parameters are defined as follows.

- **<period>:** The BSS max idle period, in units of 1000 TUs (1.024 s). 0 disables the feature. Set to the AP's BSS max idle period (ap_max_inactivity on a hostapd AP).
- **-a:** Interpret <period> according to 802.11ah rules for idle period values. This is a placeholder for future support and is not currently supported.

When vendor_keep_alive_offload is set, wpa_supplicant configures the chip on every association, using the BSS max idle period advertised by the AP. This replaces any value previously set with morse_cli keepalive. The morse_cli command is intended for testing, or for when wpa_supplicant is not configured to offload keepalives.

In FullMAC firmware, BSS keepalives are always handled on the Morse Micro chip and no configuration is required.

5.4 How to Verify

1. Check the following statistics to identify if the feature is working:

```
STA1# morse_cli stats -f "BSS keepalive"
BSS keepalive TX                                : 13
BSS keepalive too many TX failures              : 0
```

BSS keepalive TX: The number of keepalive packets sent.

BSS keepalive too many TX failures: The number of times the chip disconnected from the AP in Standby mode because no transmission had been acknowledged for four BSS max idle periods.

2. Use a sniffer tool such as tshark on the AP to verify that a keepalive packet is received from the DUT once per BSS max idle period.

Note: Wireshark reports keepalive packets as **malformed IPv4 packets** because the LLC header has its type field set to IPv4, but there is no IPv4 header or payload. This is harmless and can safely be ignored.

6 TCP Keepalive Offload

6.1 Scope

The TCP Keepalive Offload feature lets the Morse Micro chip support TCP keepalive functionality for an active TCP session. This includes both responding to keepalives from the remote TCP endpoint and generating TCP keepalives as required. The main aim is to keep the TCP session active without requiring action from the local host processor's TCP stack.

The feature is not specific to Standby mode, but is only available in STA mode.

6.2 Theory of Operation

The feature assumes control of all TCP keepalive operations on behalf of the TCP stack to ensure the remote endpoint does not drop the connection, particularly when the host processor is suspended. It does not process other TCP packets, except to snoop them and gather the latest values for things like sequence numbers so it can process and generate keepalive frames correctly. The chip must receive at least one packet on the TCP session after the feature is enabled before it can start to operate. This could be a keepalive or data exchange.

The feature responds to TCP keepalives from the remote endpoint. It generates a TCP keepalive only if it sees nothing from the remote endpoint during the configured period.

The host processor must enable and disable the feature when the TCP session is established (or as needed) and when the session completes (or the feature is no longer needed), via a `morse_cli` command. When enabling the feature, the addresses for the TCP endpoints (both local and remote) should be provided.

If all retries fail, the chip stops handling the session. If the device is in Standby mode, the chip wakes the host processor, indicating that the TCP connection has been lost.

This feature supports only one TCP session at a time.

6.2.1 Other Considerations

Proper operation requires the local host processor's TCP stack to be configured with a long keepalive timeout, so the host processor does not start generating its own keepalive frames.

Related offload features that ensure the association and relevant ARP entries are maintained need to be used when using this feature with Standby mode. Depending on other configuration (such as whether DHCP or static IP configuration is in use), that might include any combination of the other features mentioned in this document.

6.3 How to Configure

The command to configure TCP Keepalive Offload is:

```
STA1# morse_cli tcp_keepalive --help
tcp_keepalive {enable|disable} [-p <period>] [-c <retry count>]
        [-i <retry interval>] [-s <src IP>] [-d <dest IP>]
        [-S <src port>] [-D <dest port>]
    Configure TCP keepalive offload parameters
    {enable|disable}              Enable/disable TCP keepalive offload
    -p <period>                   Period in seconds (1-65535)
    -c <retry count>              Number of retries (0-255)
    -i <retry interval>           Seconds between retries (1-255)
    -s <src IP>                   Source IP address in dotted decimal notation
    -d <dest IP>                  Destination IP address in dotted decimal notation
    -S <src port>                 TCP source port (1-65535)
    -D <dest port>                TCP destination port (1-65535)
```

The parameters are defined as follows.

- **period:** The time in seconds between locally generated TCP keepalive packets.
- **retry count:** The number of keepalives to send without a response before deciding the connection is lost.
- **retry interval:** The number of seconds between retries.
- **src IP:** *The IP address of the remote TCP endpoint (the source IP address for received TCP packets).
- **src port:** *The port number of the remote TCP endpoint.
- **dest IP:** *The IP address for the local TCP endpoint (the destination IP address for received TCP packets).
- **dest port:** *The port number for the local TCP endpoint.
- **enable:** Activate the feature.
- **disable:** Deactivate the feature and clear all TCP session information.

***Note:** Source and destination are specified from the perspective of packets received from the remote endpoint, not packets sent by the local host.

To enable the feature, supply the source and destination IP addresses and ports. Only the TCP stream matching those values will be processed for TCP keepalives. If source/destination IP addresses and ports are not supplied, the first TCP stream received will be used. This also applies after a connection is lost.

The following examples show how to set up TCP Keepalive Offload.

```
STA1# morse_cli tcp_keepalive -p 5 -c 4 -i 2 -s 192.168.1.1 -S 5001
    -d 192.168.1.2 -D 45003 enable
```

Enables the feature with a 5-second keepalive period, retries every 2 seconds, and gives up after 4 keepalives without a response. The TCP stream is for local IP address 192.168.1.2, local port 45003, remote IP address 192.168.1.1, and remote port 5001.

```
STA1# morse_cli tcp_keepalive -s 10.57.23.7 -S 5050 -d 10.57.23.45 -D 37003 enable
```

Enables the feature. Default period settings (period 5 s, retry count 5, retry interval 3 s) are applied if not previously set. The TCP stream is for local IP address 10.57.23.45, local port 37003, remote IP address 10.57.23.7, and remote port 5050.

To disable the feature:

```
STA1# morse_cli tcp_keepalive disable
```

6.4 How to Verify

Check the following statistics to identify if the feature is working:

```
STA1# morse_cli stats -f "TCP keepalive"
TCP keepalives answered                         : 6147
TCP keepalives to host                          : 33
TCP keepalives generated                        : 468
TCP keepalive connection lost                   : 1
```

TCP keepalives answered: The number of TCP keepalive ACK frames sent from the feature. This shows the feature is receiving TCP keepalives from the remote TCP endpoint and is responding.

TCP keepalives to host: This counter increments when keepalives are forwarded to the host processor during synchronization with the TCP session, but may increment faster than the number of keepalives actually forwarded. Once keepalive offload is operational, this value should not increase.

TCP keepalives generated: The number of TCP keepalives sent by the chip.

TCP keepalive connection lost: The number of times the feature woke the host processor while in Standby mode because all keepalive retries failed.

7 ICMP Response Offload

7.1 Scope

The ICMP Response Offload feature allows the Morse Micro chip to respond directly to ICMP echo (ping) requests without involving the host processor.

7.2 Theory of Operation

This feature is triggered when the following conditions are met:

- The device is in STA mode.
- The morse driver `enable_arp_offload` modparam is set.
- The Wi-Fi HaLow interface has one or more IPv4 addresses configured.
- The device receives an ICMP echo (ping) request on the Wi-Fi HaLow interface.

7.2.1 Firmware Actions

When ICMP offload is enabled (via the `enable_arp_offload` modparam), the Morse Micro chip takes the following actions:

- If the chip receives an ICMP echo packet for any of the (max 4) configured IPv4 addresses, it sends an ICMP echo response packet to the requestor instead of forwarding the ICMP echo request packet to the host processor. If the packet is using QoS, the response is sent using the same TID.
- If the chip receives an ICMP echo request packet for any other IPv4 address, it drops the packet.
- All ICMP packets other than echo request packets will be forwarded to the host processor.

7.2.2 Restrictions

- Fragmented frames are not supported and will be forwarded to the host processor.

7.3 How to Configure

ICMP Response Offload is enabled with ARP Response Offload (see section 3.3 How to Configure). No other configuration is necessary to enable ICMP Response Offload.

7.4 How to Verify

Use a sniffer tool such as tshark on the DUT to verify that no ping request packets reach the host processor. There should be no output from this command while pinging any of the wlan0 interface's IP addresses when this feature is operational:

```
STA1# tshark -i wlan0
```

From the AP, ping the DUT and verify that it is responsive:

```
AP# ping -c3 192.168.1.2
PING 192.168.1.2 (192.168.1.2): 56 data bytes
64 bytes from 192.168.1.2: seq=0 ttl=64 time=112.309 ms
64 bytes from 192.168.1.2: seq=1 ttl=64 time=33.814 ms
64 bytes from 192.168.1.2: seq=2 ttl=64 time=57.527 ms


--- 192.168.1.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 33.814/67.883/112.309 ms
```

Also check the following statistics to verify that packets were processed as expected:

```
STA1# morse_cli stats -f ICMP
ICMP packets received                          : 3
ICMP requests detected                         : 3
ICMP requests answered                         : 3
ICMP requests dropped                          : 0
```

ICMP packets received: The number of ICMP packets (all types) received by the Morse Micro chip.

ICMP requests detected: The number of ICMP echo request packets received by the chip. This counter will not increment if the feature is not enabled.

ICMP requests answered: The number of ICMP echo request packets responded to by the chip. This counter will not increment when requests are forwarded to the host processor for processing, for example when they are fragmented.

ICMP requests dropped: The number of ICMP echo request packets the chip dropped because the IP address did not match any of the (max 4) IP addresses for the Wi-Fi HaLow interface.

8 Packet Whitelist Filtering

8.1 Scope

The Packet Whitelist Filtering feature lets the Morse Micro chip drop uninteresting packets to avoid unnecessarily waking the host processor during Standby mode. If a received packet has not been processed by any offload feature and does not match any whitelist filters, the chip drops it instead of waking the host processor.

8.2 Theory of Operation

Whitelist filters can be configured at any time, but filtering is only performed when the following conditions are met:

- The device is in Standby mode.
- One or more whitelist filters have been configured.

Each whitelist filter may be based on any combination of the following fields:

- Link layer (Ethernet) protocol ID
- IPv4 protocol ID
- IPv4 source address, with an optional netmask
- IPv4 destination address, with an optional netmask
- IPv4 UDP or TCP source port number
- IPv4 UDP or TCP destination port number

When packet whitelisting is enabled, the Morse Micro chip takes the following actions:

- If the chip receives a packet that matches any configured whitelist filter, it wakes the host processor and exits Standby mode.
- All other packets are dropped.

The following packets are not subject to whitelist filtering and are always passed to the host processor, waking it and exiting Standby mode:

- Packets with no LLC (SNAP) header
- EAPOL packets
- ARP packets that are not handled by ARP Response Offload (for example, ARP requests for secondary IP addresses)
- Fragmented 802.11 frames

Packets handled by other offload features (DHCP, ARP, ICMP, TCP keepalive) are consumed by the chip and never reach whitelist filtering. Fragmented IPv4 packets are always dropped.

8.3 How to Configure

Up to five whitelist filters may be configured:

```
STA1# morse_cli whitelist --help
whitelist [-c] [-l <LLC proto>] [-i <IPv4 proto>] [-s <src IP>] [-d <dest IP>]
        [-n <netmask>] [-S <src port>] [-D <dest port>]
    Configure the packet whitelist filter
    -l <LLC proto>                Link layer protocol - e.g. 0x0800 for IPv4
    -i <IPv4 proto>               IPv4 protocol - e.g. 6 for TCP or 17 for UDP
    -s <src IP>                   Source IP address in dotted decimal notation
    -d <dest IP>                  Destination IP address in dotted decimal notation
    -n <netmask>                  Netmask in dotted decimal notation
    -S <src port>                 UDP or TCP source port - range 1-65535
    -D <dest port>                UDP or TCP destination port - range 1-65535
    -c                            Clear all whitelist entries
```

Note: The -n option applies to both -s and -d, and the address must be a network address (for example, 10.57.23.0 with 255.255.255.0).

8.4 How to Verify

In this test:

- 192.168.1.1 is the AP
- 192.168.1.2 is the DUT (STA1)
- 192.168.1.3 is another station (STA2)

ARP Response Offload (section 3 ARP Response Offload) should be enabled so that ARP requests from other devices do not wake the DUT.

Configure whitelisting on the DUT using the following configuration, which will trigger wake-up from Standby mode if a received IPv4 packet has a source address in the 10.57.23.0/24 subnet (for example, from a server on another network), or is a TCP packet from 192.168.1.3 to port 22 (SSH):

```
STA1# morse_cli whitelist -s 10.57.23.0 -n 255.255.255.0
STA1# morse_cli whitelist -s 192.168.1.3 -i 6 -D 22
```

Put the DUT into Standby mode:

```
STA1# morse_cli standby enter
```

From the AP, attempt to SSH to the DUT. This should fail because the SSH packets do not match either of the configured whitelist filters:

```
AP# ssh 192.168.1.2
```

From STA2, SSH to the DUT. This should work because the first SSH packet will cause the DUT to exit Standby mode:

```
STA2# ssh 192.168.1.2
STA1# hostname
STA1
STA1# exit
Connection to 192.168.1.2 closed.
STA2#
```

Note: This will only work if the DUT has been configured for wake-up from Standby mode, which requires provisioning a wake-up GPIO pin and a mechanism to wake the host processor and then invoking the following command. To simulate that process on a host that does not hibernate or power off when the `morse_cli standby enter` command is invoked, this command can be entered manually from the host processor to restore connectivity with the Morse Micro chip:

```
STA1# morse_cli standby exit
```

Check the following statistics to verify that packets were processed as expected:

```
STA1# morse_cli stats -f Whitelist
Whitelist packets ignored                    : 0
Whitelist packets dropped                    : 20
Whitelist packets allowed                    : 1
```

Whitelist packets ignored: The number of packets received by the Morse Micro chip that were not subject to whitelist filtering (packets with no LLC header, EAPOL packets, and ARP packets not handled by ARP Response Offload) and caused wake-up from Standby mode.

Whitelist packets dropped: The number of packets received by the chip that were dropped because they did not match any of the whitelist filters. This includes all fragmented IPv4 packets. In this example, the chip received 20 SSH packets from the AP and dropped them.

Whitelist packets allowed: The number of packets received by the chip that matched one of the whitelist filters and caused wake-up from Standby. The first matching packet in a stream brings the device out of Standby mode, so subsequent packets do not go through whitelist filtering and do not increment these counters. In this example, the first SSH packet from STA2 brought the device out of Standby mode.

9 Revision History

Version Number	Release Date	Release Notes
1	20 May 2024	Initial release
2	02 Aug 2024	Added DHCP Offload
3	22 Oct 2026	Added BSS Keepalive Offload General improvements

Morse Micro provides this information "as is" without warranties of any kind, express or implied. No guarantee is made as to the accuracy, completeness, or suitability of this information or Morse Micro's products for any specific purpose. Use of this information and products is at the user's sole risk. Morse Micro products are not designed or tested for use in mission-critical systems, and should not be used in such applications. Performance specifications are based on internal testing and are believed to be reliable; however, they are not guaranteed. It is the Buyer's responsibility to test and validate all product performance, compatibility, and compliance, both in isolation and within end applications. Morse Micro assumes no liability for the use or application of any product, circuit, or information described herein. No license or other rights—express or implied—are granted under Morse Micro's intellectual property. This document contains proprietary information of Morse Micro and is subject to change without notice. Wi-Fi®, Wi-Fi HaLow™, and the Wi-Fi logo are trademarks of Wi-Fi Alliance. All other trademarks are the property of their respective owners.



Morse Micro

Morse Micro Pty. Ltd. Corporate Headquarters

Level 8, 10-14 Waterloo Street, Surry Hills, NSW 2010, AUSTRALIA

Email: sales@morsemicro.com

Copyright © Morse Micro Pty. Ltd.