MiWi™

MiWi™ Software Design Guide

Introduction

The MiWi is Microchip's proprietary wireless networking stack designed to support Low Rate Personal Area Networks (LRPANs). This guide describes the MiWi applications implemented on the MiWi protocol available in the SAM platforms (SAMR21 and SAMR30).

The MiWi supports the following three network topologies:

- Peer-to-Peer (P2P)
- Star
- Mesh

Features

Earlier versions of the MiWi Mesh networking stack (until version 2.10), released in the MiWi protocol v5.30 of Microchip Libraries for Applications (MLA) v2017-03-06, supports a library-based Mesh networking stack. However, this is redesigned with the following changes:

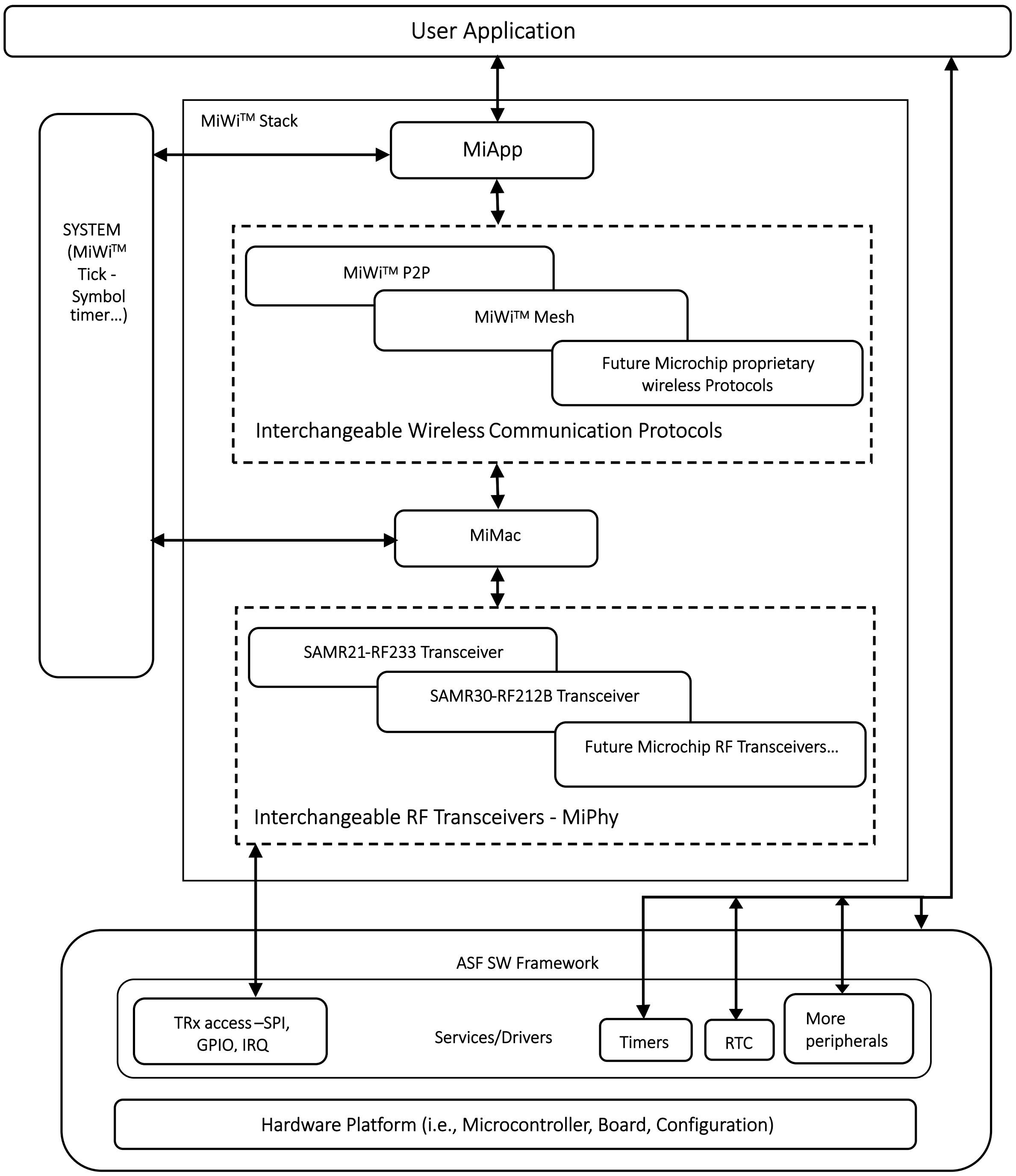
1. Optimization of current APIs to improve simplicity.
2. Redesign of the MiWi Mesh with additional features for next generation platforms.
3. A new commissioning procedure to improve the secured inclusion of devices to the network.
4. Dynamic switching between device types in the MiWi Mesh.
5. Network secure feature for all network messages.
6. Over-The-Air Upgrade to upgrade all the nodes in the network.

Table of Contents

1. MiWi Architecture

The following is the MiWi protocol architecture on Advanced Software Framework (ASF) which allows the user to obtain required components, services and drivers from ASF Wizard. For more details, refer to the ASF Wizard section at the Atmel Software Framework web page.

Figure 1-1. MiWi™ Architecture


User Application
MiWi™ Stack
MiApp
SYSTEM (MiWi™ Tick - Symbol timer...)
MiWi™ P2P
MiWi™ Mesh
Future Microchip proprietary wireless Protocols
Interchangeable Wireless Communication Protocols
MiMac
SAMR21-RF233 Transceiver
SAMR30-RF212B Transceiver
Future Microchip RF Transceivers...
Interchangeable RF Transceivers - MiPhy
ASF SW Framework
TRx access –SPI, GPIO, IRQ
Services/Drivers
Timers
RTC
More peripherals
Hardware Platform (i.e., Microcontroller, Board, Configuration)

2. Supported Topologies

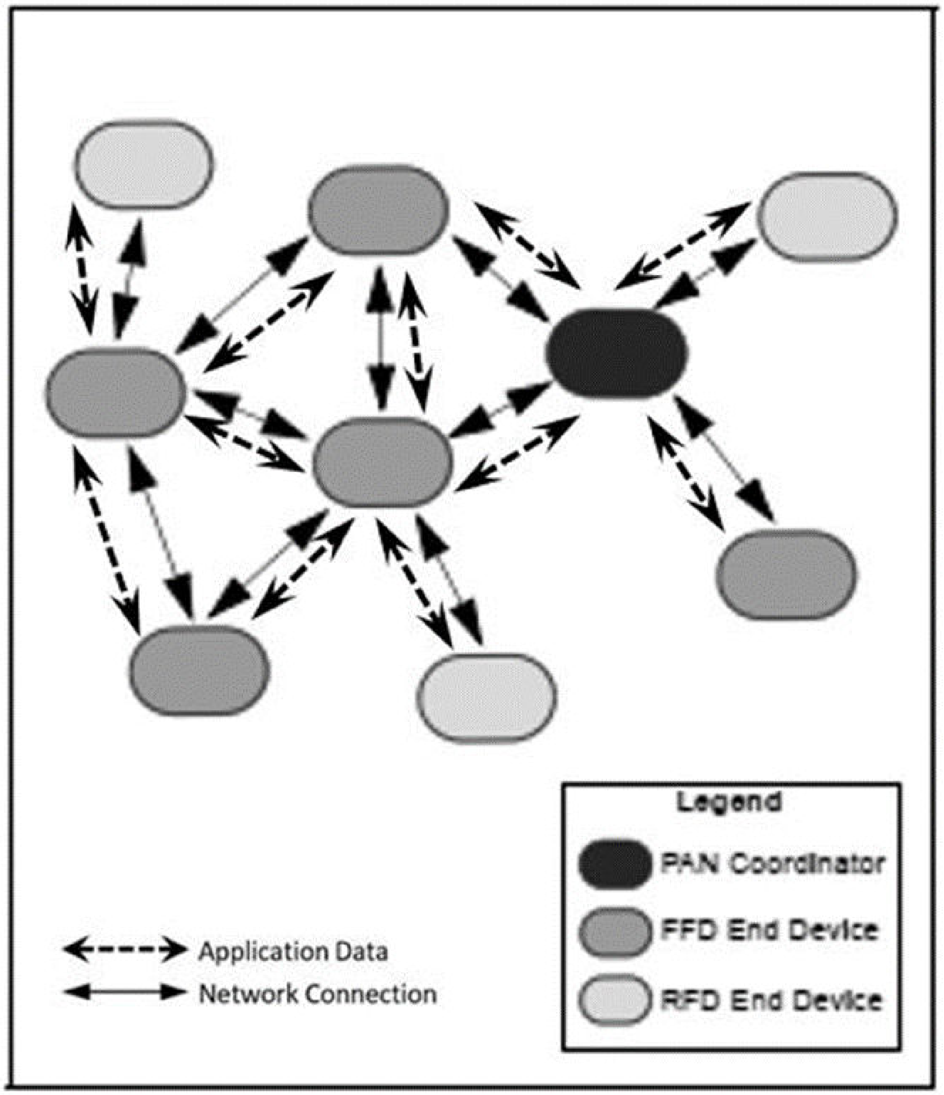
2.1 Peer-To-Peer (P2P) Topology

A typical P2P topology is shown in the following figure. From a device role perspective, this topology has one PAN Coordinator that starts communication from the end devices. When joining the network, however, end devices do not have to establish their connection with the PAN Coordinator.

As to functional types, the PAN Coordinator is an FFD (Full Function Device) and the end devices can be FFDs or RFDs (Reduced Function Device). In this topology, however, end devices that are FFDs can have multiple connections. Each of the end device RFDs, however, can connect to only one FFD and cannot connect to another RFD.

As shown in the following figure, the application data can be sent to all the devices in the radio range only.

Figure 2-1. Peer-To-Peer Topology



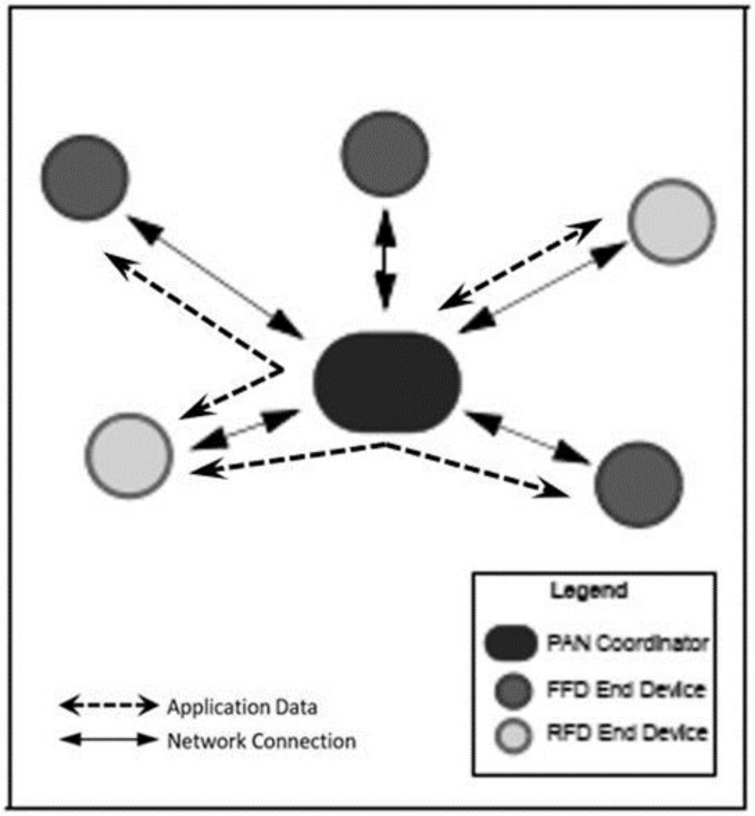
2.2 Star Topology

A typical Star topology is shown in the following figure. From a device role perspective, the topology has one Personal Area Network (PAN) Coordinator that initiates communications and accepts connections from other devices. It has several end devices that join the communication. The end devices can establish connections only with the PAN Coordinator.

As to functionality type, the PAN Coordinator of the Star topology is an FFD. The end device can be an FFD with its radios ON all the time, or an RFD with its radio OFF when it is idle. Regardless of its functional type, the end devices can only communicate to the PAN Coordinator.

As shown in the below figure, the application data can be sent to any device in the network, as the PAN Coordinator forwards data from one end device to another.

Figure 2-2. Star Topology



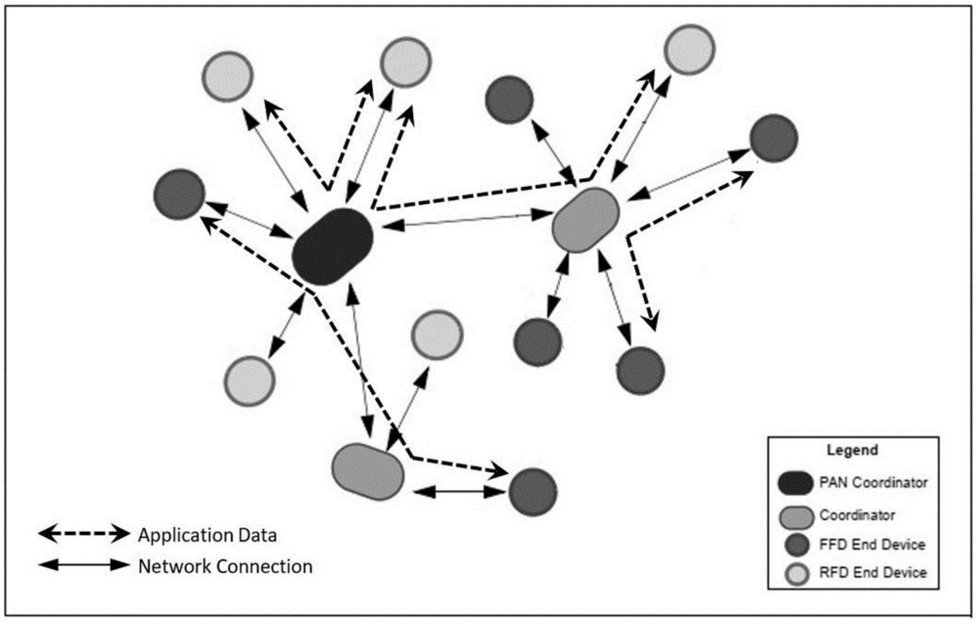
2.3 Mesh Topology

A typical Mesh topology is shown in the following figure. From a device role perspective, this topology has one PAN Coordinator that starts network, all the coordinators will request a short address from the PAN Coordinator. End devices joining the network will join to the nearby coordinator.

As to functional types, the PAN Coordinator and Coordinator are FFD and the end devices can be FFDs or RFDs. Each of the end device FFDs/RFDs, however, can connect to only one Coordinator and cannot connect to another end device.

As shown in the below figure, the application data can be sent from any device to any device though Mesh routing.

Figure 2-3. Mesh Topology



3. MiWi P2P

3.1 Network Addressing

The IEEE® 802.15.4 specification defines two kinds of addressing mechanisms:

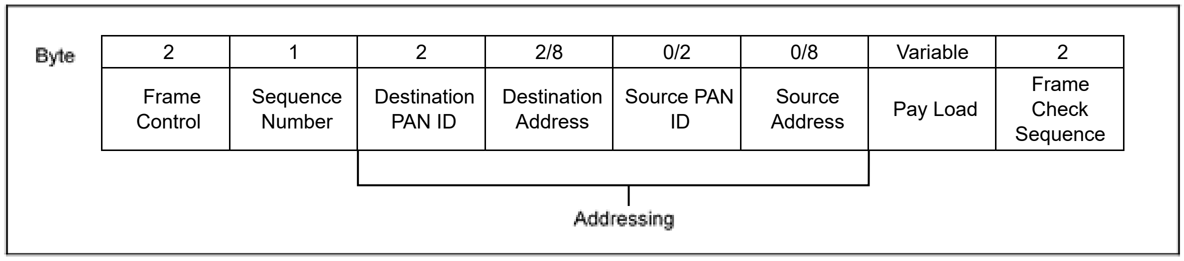
- Long Address or Extended Organizationally Unique Identifier (EUI) – an 8-byte address that is unique for each device, worldwide. The upper three bytes are purchased from IEEE by the company that releases the product. The lower five bytes are assigned by the device manufacturer as long as the EUI of each device is unique. The 8-byte unique address is usually called the MAC address of the wireless device/node and is predominantly associated with the node hardware.
- Short Address – a 2-byte address that is assigned to the device by its parent when it joins the network. The short address must be unique within the network.

The MiWi P2P protocol supports only one-hop communication; hence it transmits messages through EUI or long address. Short addressing is used only when the stack transmits a broadcast message.

3.2 Message Format

The message format of the MiWi P2P protocol is a subset of the message format of the IEEE 802.15.4 specification. The following figure illustrates the packet format of the stack and its fields.

Figure 3-1. MiWi P2P Wireless Protocol Packet Format



Byte	2	1	2	2/8	0/2	0/8	Variable	2
	Frame Control	Sequence Number	Destination PAN ID	Destination Address	Source PAN ID	Source Address	Pay Load	Frame Check Sequence

3.2.1 Frame Control

The following figure illustrates the format of the 2-byte Frame Control field.

Figure 3-2. Frame Control

Bit	3	1	1	1	1	3	2	2	2
	Frame Type	Security Enabled	Frame Pending	Acknowledgement Request	Intra PAN	(Reserved)	Destination Address Mode	(Reserved)	Source Address Mode

The 3-bit Frame Type field defines the type of packet with the following values:

- Data frame =`001`
- Acknowledgement =`010`
- Command frame =`011`

The Security Enabled bit indicates if the current packet is encrypted. There is an additional security header if encryption is used. For more information, see 2.3 Mesh Topology.

The Frame Pending bit is used only in the Acknowledgement packet handled by the MRF24J40 radio hardware. This bit indicates if an additional packet follows the Acknowledgement after a data request packet is received from a RFD end device.

The Intra-PAN bit indicates if the message is within the current PAN. If this bit is set to ‘1’, the Source PAN ID field in the addressing fields is omitted. In the stack, this bit is always set to ‘1’, but it can be set to ‘0’ to enable inter-PAN communication. Resetting the bit to ‘0’ can be done in the application layer, if it is necessary.

The Destination Address mode can be either 16-bit Short Address mode = 10 or 64-bit Long Address mode

= 11

In the MiWi P2P protocol, the Destination Address mode is usually set to the Long Address mode. The Short Address mode is used only for a broadcast message. For broadcast messages, the Destination Address field in the addressing fields is fixed to 0xFFFF.

The Source Address mode for the MiWi P2P protocol can only be the 64-bit Long Address mode.

3.2.2 Sequence Number

The sequence number is 8 bits long. It starts with a random number and increases by one each time a data or command packet is sent. The number is used in the Acknowledgement packet to identify the original packet. The sequence number of the original packet and the Acknowledgement packet must be same.

3.2.3 Destination Pan ID

This is the PAN identifier for the destination device. If the PAN identifier is not known, or not required, the broadcast PAN identifier (0xFFFF) is used.

3.2.4 Destination Address

The destination address can either be a 64-bit long address or a 16-bit short address. The destination address must be consistent with the Destination Address mode defined in the Frame Control field. If the 16-bit short address is used, it must be the broadcast address of 0xFFFF.

3.2.5 Source Pan ID

The source PAN identifier is the PAN identifier for the source device and must match the intra-PAN definition in the Frame Control field. The source PAN ID exists in the packet only if the intra-PAN value is ‘0’.

In the current MiWi P2P protocol implementation, all communication is intra-PAN. As a result, all packets do not have a Source PAN ID field.

However, the stack reserves the capability for the application layer to transmit the message inter-PAN. If a message needs to transmit inter-PAN, the source PAN ID is used.

3.2.6 Source Address

The Source Address field is fixed to use the 64-bit extended address of the source device.

3.3 Transmitting and Receiving

3.3.1 Transmitting Messages

There are two ways to transmit a message: Broadcast and Unicast.

Broadcast packets have all devices in the radio range as their destination. The IEEE 802.15.4 defines a specific short address as the broadcast address but has no definition for the long address. As a result, for a IEEE 802.15.4 compliant transceiver, broadcasting is the only situation when the MiWi P2P stack uses a short address.

There is no Acknowledgement for broadcasting messages. Unicast transmissions have only one destination and use the long address as the destination address. The MiWi P2P protocol requires Acknowledgement for all unicast messages.

If the transmitting device has at least one device that turns off its radio when idle, the transmitting device saves the message in RAM and waits for the sleeping device to wake up and request the message. This kind of data transmitting is called *Indirect Messaging*.

If the sleeping device fails to acquire the indirect message, it expires and becomes discarded. Usually, the indirect message timeout needs to be longer than the pulling interval for the sleeping device.

3.3.2 Receiving Messages

In the MiWi P2P protocol, only the messaged device is notified by the radio. If the messaged device turns off its radio when idle, it can only receive a message from the device to which it is connected.

For the idling device with the turned off radio to receive the message, the device must send a data request command to its connection peer. Then, it acquires the indirect message if there is one.

In Star topology, only the PAN Coordinator is enabled for connections and end devices (FFD/RFD) are all connected to the PAN Coordinator. Hence, the end devices in Star topology have single connections.

3.4 Handshaking

Handshaking is the elaborate process of joining a network. A device can join only a single device as its parent and hence, the initial handshaking is the actual process of choosing a parent.

Choosing the parent requires the following steps:

1. Listing all the possible parents
2. Choosing the right one as its parent

The MiWi P2P protocol is designed for simplicity and direct connections in Star and P2P communication topologies. Some IEEE 802.15.4 requirements obstruct that design:

- The five-step handshaking process, plus two time-outs, requires a more complex stack.
- The association process uses one-connection communication rather than the multi-connection concept of peer-to-peer topology.

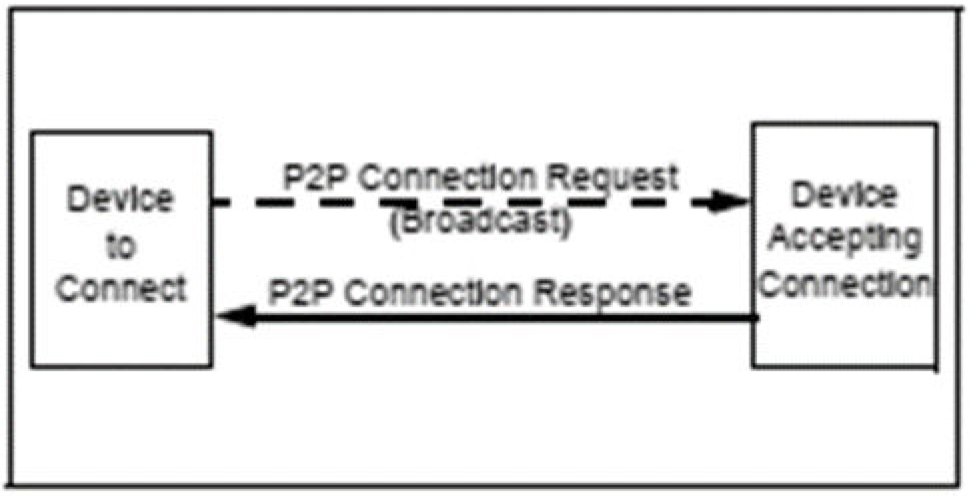
For the preceding reasons, the MiWi P2P protocol uses its own two-step handshaking process as shown in the following figure:

1. The initiating device sends out a P2P connection request command.
2. Any device within radio range responds with a P2P connection response command that finalizes the connection.

This is a one-to-many process that may establish multiple connections, where possible, to establish a peer-to-peer topology. Since this handshaking process uses a MAC layer command, CSMA-CA is applied for each transmission. This reduces the likelihood of packet collision.

RFDs may receive the Connection Request command from several FFDs, but can connect to only one FFD. An RFD chooses the FFD, from which it receives the first P2P connection response as its peer.

Figure 3-3. Handshaking Process For MiWi P2P Wireless Protocol


Device to Connect
P2P Connection Request (Broadcast)
P2P Connection Response
Device Accepting Connection

3.5 Custom MAC Commands for MiWi P2P Wireless Protocol

The MiWi P2P protocol extends the functionality of the IEEE 802.15.4 specification by using custom MAC commands for removing the connection between two devices. The following table lists all of the custom MAC commands of the protocol.

Table 3-1. Custom MAC Commands for MiWi P2P Wireless Protocol

Command Identifier	Command Name	Description
0x81	P2P Connection Request	Request to establish a P2P connection. Usually broadcast to seek P2P connection after powering up. Alternately, unicast to seek an individual connection.
0x82	P2P Connection Removal Request	Removes the P2P connection with the other end device.
0x83	P2P Data Request	Similar to the IEEE 802.15.4 specification Data Request command (0x04), a request for data from the other end of a P2P connection if the local node had its radio turned off. Reserved for the previously sleeping device to request the other node to send the missed message (indirect messaging).
0x84	Channel Hopping	Request to change operating channel to a different channel. Usually used in the feature of frequency agility.
0x87	Active Scan Request	Checks available nodes in the current and accessible channels.
0x91	P2P Connection Response	Response to the P2P connection request. Also can be used in active scan process.
0x92	P2P Connection Removal Response	Response to the P2P connection removal request.
0x97	Active Scan Response	Response returns the node information including the Channel, PAN ID and Node ID.

Note:
See 3.7 Active Scan for details on Active Scan Request and Active Scan Response.

3.5.1 P2P Connection Request

The P2P connection request (0x81) is broadcasted to establish a P2P connection with other devices after powering-up. The request can also be unicast to a specific device to establish a single connection.

When the transmitting device receives a P2P connection response (0x91) from the other end, a P2P connection is established.

The P2P connection request custom command can also start an active scan to determine what devices are available in the neighborhood.

When a P2P connection request command is sent for active scan purposes, the capability information and optional payload is not attached. The receiving device uses the attachment, or absence of capability information, and an optional payload to determine if the command is a request to establish a connection or just an active scan.

The MiWi P2P protocol can enable or disable a device to allow other devices to establish connections. After a device is disabled from making connections, any new P2P connection request is discarded, except under the following conditions:

- The P2P connection request is coming from a device in which the receiving end established a connection.
- The P2P connection request is an active scan.

The following figure shows the format of the P2P connection request command frame.

Figure 3-4. P2P Connection Request Command Format

Octet	15/21	1	1	1 (Optional)	Various (Optional)
	MAC Header	Command Identifier (0x81)	Operating Channel	Capability Information	Optional payload to identify the node. It is not required for the stack, but may be useful for applications.

Figure 3-5. Capability Information Format

Bit	0	1	2	3	4-7
	Receiver ON when Idle	Request Data on Wake-up	Need Time Synchronization (Reserved)	Security Capable	(Reserved)

The operating channel is used to bypass the effect of subharmonics that may come from another channel. It avoids the false connections with devices that operate on different channels. The capability information byte, as shown in Figure 3-4, uses a format as illustrated in the preceding figure.

The optional payload of the P2P connection request is provided for specific applications. A device may need additional information to identify itself, either its unique identifier or information about its capabilities in the application. With the optional payload, no additional packets are required to introduce or identify the device after the connection is established. The optional payload is not used in the stack.

3.5.2 P2P Connection Removal Request

The P2P connection removal request (0x82) is sent to the other end of the connection to remove the P2P connection. The following figure shows the format of the request.

Figure 3-6. P2P Connection Removal Request Format

Octet	15/21	1
	MAC Header: Send to the other end of the P2P connection to cut the communication	Command Identifier (0x82)

3.5.3 Data Request

The data request (0x83) command is the same as the data request (0x04) command of the IEEE 802.15.4 specification. The following figure shows the format of the request.

If one side of a P2P connection node is able to sleep when idle, and that node can receive a message while in sleep, the active side of the connection must store the message in its RAM. The active side delivers the message when the sleeping device wakes up and requests the message.

If an application involves such conditions, the `ENABLE_INDIRECT_MESSAGE` feature needs to be activated. The sleeping node must send the data request command after it wakes up.

Figure 3-7. Data Request Format

Octet	21	1
	MAC Header: Unicast from extended source address to extended destination address	Command Identifier (0x83 or 0x04)

3.5.4 Channel Hopping

The channel hopping command (0x84) requests for the destination device to change the operating channel to another channel. The following figure shows the format of the command.

This command is usually sent by the frequency agility initiator which determines when to change channels and what channel to select.

This command is usually broadcasted to notify all devices, with their radios ON when idle, to switch channels. To ensure that every device receives this message, the frequency agility initiator performs the broadcast three times and all the FFD devices perform the rebroadcast.

When the channel hopping sequence is carried out and all FFDs hop to a new channel, RFDs have to perform resynchronization to restore connection to their respective FFD peers.

Figure 3-8. Channel Hopping Format

Octet	15/21	1	1	1
	MAC Header: Broadcast or unicast from the Frequency Agility Starter	Command Identifier (0x84)	Current Operating Channel	Destination Channel to Jump to

3.5.5 P2P Connection Response

The P2P connection response (0x91) command is used to respond to the P2P connection request. The following figure shows the format of the command.

The P2P connection response command can be used to establish a connection. Alternately, the command can be used by a device responding to an active scan, identifying itself as active in the neighborhood.

Figure 3-9. P2P Connection Response Format

Octet	21	1	1	1 (Optional)	Various (Optional)
	MAC Header: Unicast from extended source address to extended destination address.	Command Identifier (0x91)	Status. 0x00 means successful. All other values are error codes.	Capability Information	Optional payload to identify the node. Not required for the stack, but possibly useful for applications.

3.5.6 P2P Connection Removal Response

The P2P connection removal response command (0x92) is used to respond to the P2P connection removal request. It notifies the other end of the P2P connection that a P2P connection request is received early and the resulting connection is removed. The following figure shows the format of the command.

Figure 3-10. P2P Connection Removal Response Format

Octet	21	1	1
	MAC Header: Unicast from extended source address to extended destination address	Command Identifier (0x92)	Status. • 0x00 means successful. • All other values are error codes

3.6 Idle Devices Turning Off Radios

For devices operating on batteries, reducing power consumption is essential. This can be done by having the devices turn off their radios when not transmitting data. The MiWi P2P protocol includes features for putting radios into Sleep mode and then waking up the device.

To activate this feature, "`ENABLE_SLEEP_FEATURE`" must be defined in the file, `miwi_config.h`.

To determine as to when a device is put into Sleep mode is identified by the specific application. The possible triggers include:

- Length of radio idle time
- Receipt of a packet from a connected FFD, requesting the device to go to Sleep mode

The conditions for awakening a device can be determined by the specific application. Possible triggers include:

- An external event like a button is pressed
- Expiration of a predefined timer

While a device is sleeping, its peer device may need to send a message. If no message is sent, no additional feature must be enabled by the peer device.

If the peer device sends a message to the sleeping device, the peer device must store the message in its volatile memory until the sleeping device wakes up and acquires the message. Since the message is not directly delivered to the sleeping device, this process is called an *Indirect Message*.

If an indirect message is delivered, the peer device of the sleeping node must define "`ENABLE_INDIRECT_MESSAGE`" in the `miwi_config.h` file.

If indirect messaging is enabled, there must be a specified maximum number of indirect messages that can be stored in the volatile memory. The maximum size of the message depends on the free RAM memory available in the peer device and from the number of RFDs connected to the same parent FFD.

The maximum number of indirect messages is defined by the "`INDIRECT_MESSAGE_SIZE`" in the `miwi_config_p2p.h`file. For indirect messaging, the timeout period for the indirect messages also needs to be defined. If a timeout period is not defined and an RFD device is inactive or not visible, the indirect message remains forever in the volatile memory.

The indirect message time-out period is defined by the "`INDIRECT_MESSAGE_TIMEOUT`" in the `miwi_config_p2p.h`file, with seconds as the unit of measurement.

3.7 Active Scan

Active scan is the process of acquiring information about the local PAN. The active scan determines:

- The device's operating channel
- The device's signal strength in the PAN
- The PAN's identifier code for IEEE 802.15.4 compliant transceiver

Active scan is particularly useful if there is no predefined channel or PAN ID for the local devices.

The maximum number of PANs that an active scan can acquire is defined in the stack as `ACTIVE_SCAN_RESULT_SIZE`.

The scan duration and channels to be scanned are determined before the active scan begins.

The scan duration is defined by the IEEE 802.15.4 specification and its length of time, measured in symbols, is calculated with the formula shown in the following equation (One second equals 62,500 symbols.).

Equation 3-1. Scan Duration

$$Scan\ Time\ Period \equiv 960 * \left(2^{ScanDuration} + 1\right)$$

Note:
Scan duration = The user-designated input parameter for the scan. An integer is from 1 to 14.

A scan duration of 10 results in a scan time period of 61,500 symbols or about 1 second. A scan duration of 9 is about half second.

The scan channels are defined by a bitmap with each channel number represented by its comparable bit number in the double word. Channel 11 is `b'0000 0000 0000 0000 0000 1000 00000000`.

Channels 11 to 26, supported in the 2.4 GHz spectrum, is `b'0000 0111 1111 1111 11111000`

`0000 000` or 0x07FFF800.

When an active scan broadcasts a P2P connection request command, it expects any device in radio range to answer with a P2P connection response command. The active scan determines only what PANs are available in the neighborhood, not how many individual devices are available for new connections. Every device responds to the scan, including those devices that do not allow new connections.

To invoke the active scan feature, "`ENABLE_ACTIVE_SCAN`" must be defined in the `miwi_config.h` file.

3.8 Energy Scan

On each frequency band, there may have multiple channels, but a PAN must operate on one. The best channel to use is the one with the least amount of energy or noise.

Energy scan is used to scan all available channels and determine the channel with the least noise.

The scan duration and channels to be scanned are determined before the energy scan is performed.

The scan duration is defined by the IEEE 802.15.4 specification and its length of time, measured in symbols, is calculated with the formula as shown in Equation 3-1. For more information on measurement, see 3.7 Active Scan.

After the scan is complete, the channel identifier with the least noise is returned. To activate the energy scan feature, "`ENABLE_ED_SCAN`" must be defined in the `miwi_config.h` file.

3.9 Frequency Agility

Frequency agility enables the MiWi P2P Protocol PAN to move to a different channel if required by the operating conditions.

In implementing this feature, the affected devices fall into one of these two roles:

- Frequency agility initiators – these are devices that determine whether channel hopping is necessary and which new channel is applicable to use.
- Frequency agility followers – these are devices that change to another channel when directed.

3.9.1 Frequency Agility Initiators

Each PAN can have one or more devices as a frequency agility initiator. An initiator must be an FFD. Each initiator must have the energy scanning feature enabled to determine the optimal channel for the hop. The initiator broadcasts a channel hopping command to the other devices on the PAN.

3.9.2 Frequency Agility Followers

A frequency agility follower can be an FFD or an RFD device.

The FFD makes the channel hop by performing one of the following steps:

- Receiving the channel hopping command from the initiator.
- Resynchronizing the connection, if data transmissions continuously fail.

An RFD device makes the message hop using the resynchronization method, which reconnects to the PAN when communication fails.

3.9.3 Enable Frequency Agility Feature

The application determines when to perform a frequency agility operation. Frequency agility is usually triggered by continuous transmission failure, either by CCA failure or no Acknowledgement received.

To activate the frequency agility feature, the "`ENABLE_FREQUENCY_AGILITY`" must be defined in the `miwi_config.h` file.

4. MiWi Star

MiWi Star protocol is an extension of the MiWi P2P protocol which is defined by Microchip. From a device role perspective, the topology has one PAN Coordinator that initiates communications and accepts connections from other devices. It can have several end devices that join the communication. End devices can establish connections only with the PAN Coordinator. As to functionality type, the Star topology's PAN Coordinator is an FFD. An end device can be an FFD with its radios ON all the time, or an RFD with its radio OFF when idle. Regardless of its functional type, the end devices can only communicate to the PAN Coordinator.

4.1 Unique Features of the MiWi Star Wireless Protocol

The Star topology supported by the MiWi P2PProtocol stack provides all the features supported by the peer-to-peer topology, however, Star topology supports several more features based on the device roles.

PAN Coordinator supports the following features:

- Shares peer device connection (FFDs and RFDs) information to all the peer devices
- Forwards data packet from one end device to another end device
- Checks network health periodically(optional)
- Transmits data packet to end devices
- Handles Indirect Messages for sleeping end devices (RFDs)
- Supports software ACK to indicate successful data transmission

The FFDs (end devices) or RFDs (sleeping end devices) support the following features:

- Link status
- Leave Network command

4.2 Custom MAC Commands for MiWi Star Wireless Protocol

The MiWi Star protocol extends the functionality of the IEEE 802.15.4 specification by using custom MAC commands for removing the connection between two devices. The following table lists all the custom MAC commands of the protocol.

Table 4-1. Custom Mac Commands For MiWi Star Wireless Protocol

Command Identifier	Command Name	Description
0xCC	Forward Packet CMD with Payload	0XCC (1 byte) Command. Destination end device address (3 bytes). Data payload.
0xDA	Software ACK to END Device	N/A
0x7A	LINK STATUS	N/A
0x77	Connection Table Broadcast Command	0x77 (1 byte) Command. Total number of end devices in the network.

The following figure shows the modified connection mode details in the star protocol.

Figure 4-1. Modified Connection Mode Details in Star Protocol

Node Capacity	Sleep	Reserved	Security Enabled	Connection Mode	Reserved
Bit	0	1-2	3	4-5	6-7

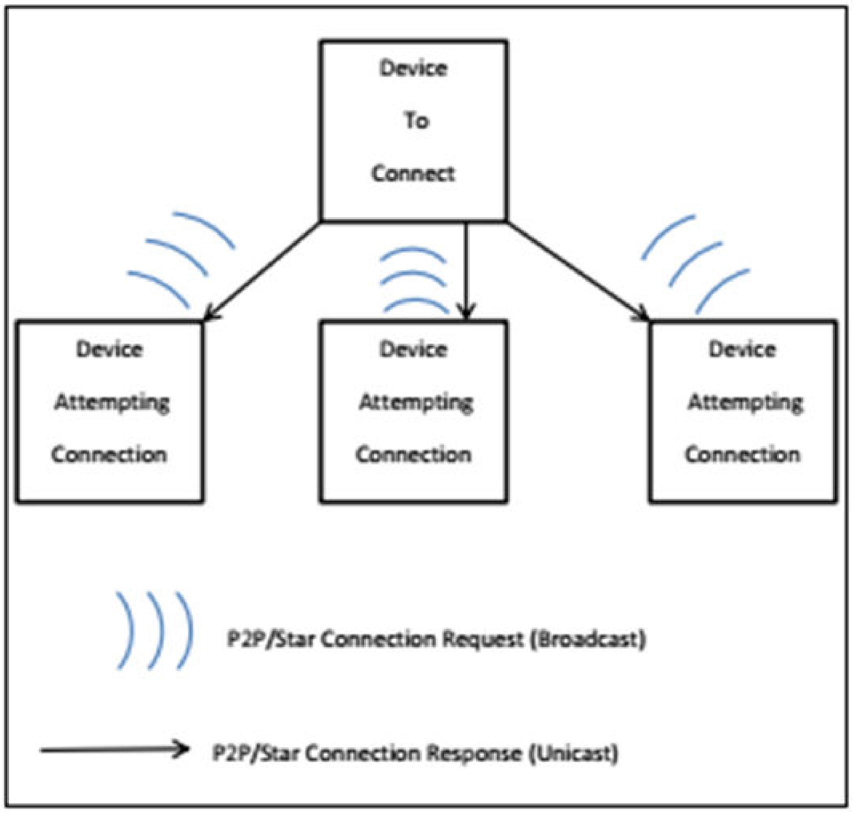
Value	Connection Mode
00	Enable All Connections
01	Enable Previous Connections
10	Enable Active Scan Response
11	Disable All Connections

4.3 Handshaking In MiWi Star Wireless Protocol

4.3.1 MiWi Star Routing

The following figure shows that a MiWi Star network consists of two types of devices (PAN Coordinator and end devices - FFDs or RFDs). The PAN Coordinator creates the network while the end devices join the PAN Coordinator. The PAN Coordinator can send messages to all the end devices in the network in a single hop. If an end device wants to communicate to another end device which may or may not be in the vicinity, the source end device must first send the packet to the PAN Coordinator and then the PAN Coordinator forwards that packet to the destination end device (2 hops).

Figure 4-2. MiWi Star Routing

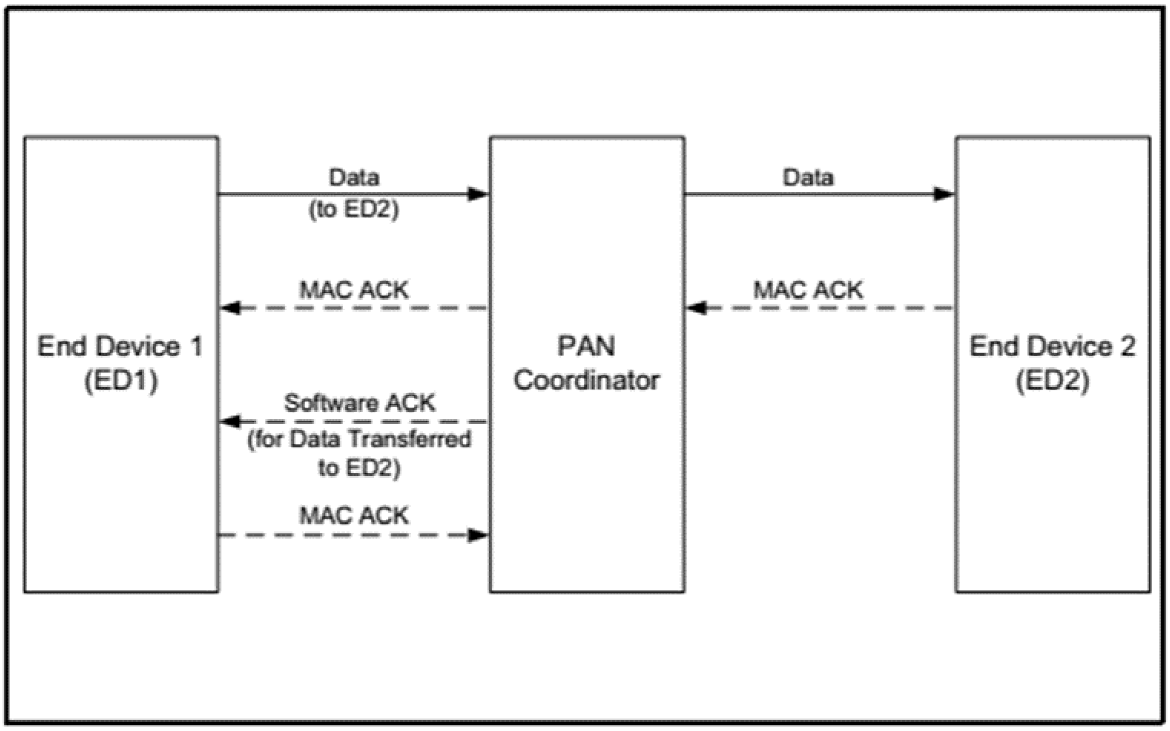

Device To Connect
Device Attempting Connection
Device Attempting Connection
Device Attempting Connection
P2P/Star Connection Request (Broadcast)
P2P/Star Connection Response (Unicast)

In a MiWi Star network, it is the responsibility of the PAN Coordinator to share the peer connections (end device addresses). In this way, all the end devices in the network know about the existence of every other device in network. When an end device wants to send a message to another end device, the source end device includes the address of the destination end device in the data payload. The source end device payload comprises of the type of packet (0xCC), destination end device address (only first 3 bytes) and the data payload. When this packet is received by the PAN Coordinator, it indicates that this packet is intended for another end device, hence, it forwards the packet to the destination end device.

4.3.2 MiWi Star Data Transfer

The connection requests and responses are similar to that of P2P between the nodes. However, in MiWi Star, the PAN Coordinator forms the network, connects the end devices and also supports the end devices to communicate between each devices (via PAN Coordinator). The following figure shows a simple data transfer between the end devices in the MiWi Star network.

Figure 4-3. Data Transfer Between End Devices in the MiWi Star Network



5. MiWi Mesh

5.1 MiWi Mesh Device Types

The MiWi Mesh protocol supports the following device types:

1. PAN Coordinator
   1.1. Starts the network
   1.2. Assigns and maintains the coordinators and its end-devices addresses
   1.3. Behaves as coordinator for routing frames
   1.4. Controls the devices which can be included into the network through commissioning
2. Coordinator
   2.1. Joins a network as an end-device
   2.2. Requests PAN coordinator for role upgrade to become a coordinator
   2.3. Supports routing of frames within the network
   2.4. Stores the commissioning information from PAN coordinator and allows only the commissioned devices to participate in the network
   2.5. Maintains its end-devices and their addresses
   2.6. Maintains data for sleeping end-devices
3. End-Device
   3.1. Joins to network though available coordinators
   3.2. Supports Rx-On end-device and Sleeping end-device for battery operated devices
   3.3. Supports dynamic switching between Rx-On to Sleeping end-device and vice versa

5.2 MiWi Mesh Frame Format

The network header and application payload of the MiWi Mesh are encapsulated inside the standard IEEE 802.15.4 data frame payload, but the stack does not adhere to the standard. Therefore, the MiWi Mesh does not receive and process IEEE 802.15.4 command frames. The following figure illustrates a general frame format composed of an IEEE 802.15.4 MAC header, network header, application payload, optional message integrity code (MIC), and a check sum (CRC).

Figure 5-1. General MiWi Frame Format

2	1	2	2/8	2	0/2/8	1	1	1	0/2	0/2	0/2/8	0/5	Variable	0/4	2
Frame Control	Sequence number	Dest. PANID	Dest. Address	Source PANID	Source Address	Hops	Frame Control	Sequence number	Dest. PANID	Dest. Address	Source Address	Auxiliary Security Header	Payload	MIC	CRC
MAC Header						Network Header							Payload	Network Footer	

5.3 MAC Header – Frame Control Field

The following figure illustrates the Frame Control field of the MAC header.

Figure 5-2. MAC Header – Frame Control Field

Bits:0-2	3	4	5	6	7:9	10:11	12:13	14:15
Frame Type	Security Enabled	Frame Pending	Ack. Request	PAN ID Compression	Reserved	Dest. Address Mode	Frame Version	Source Address Mode

These are the fixed MAC Frame control field settings used in MiWi Mesh. The following table lists the settings used for a Frame Control field of the MAC header.

Table 5-1. MAC Frame Control Field Settings

Field Name	Settings
Frame Type	Data
Security Enabled	False
Frame Pending	True if pending data available for sleeping end device, otherwise false
Acknowledgment Request	True for unicast frames and false for broadcast frames
PAN ID Compression	True
Destination Addressing Mode	0 for no address fields, 2 for 16-bit short address and 3 for 64-bit extended address
Frame Version	0
Source Addressing Mode	0 for no address fields, 2 for 16-bit short address and 3 for 64-bit extended address

5.3.1 Frame Type

The Frame Type subfield is 3 bits in length and shall be set to 001 - Data.

5.3.2 Security Enabled

The Security Enabled subfield is 1 bit in length, and it shall be set to one if the frame is protected by the MAC sublayer and set to zero otherwise. The Auxiliary Security Header field of the MHR shall be present only if the Security Enabled subfield is set to one.

5.3.3 Frame Pending

The Frame Pending subfield is 1 bit in length and shall be set to one if the device sending the frame has more data for the recipient. This subfield shall be set to zero otherwise.

The Frame Pending subfield shall be used only in beacon frames or frames transmitted either during the CAP by devices operating on a beacon-enabled PAN or at any time by devices operating on a non-beacon enabled PAN. At all other times, it shall be set to zero on transmission and ignored on reception.

5.3.4 Acknowledgment Request

The Acknowledgment Request subfield is 1 bit in length and specifies whether an acknowledgment is required from the recipient device on receipt of a data or MAC command frame. If this subfield is set to one, the recipient device shall send an acknowledgment frame only if, upon reception, the frame passes the third level of filtering. If this subfield is set to zero, the recipient device shall not send an acknowledgment frame.

5.3.5 PAN ID Compression

The PAN ID Compression subfield is 1 bit in length and specifies whether the MAC frame is to be sent containing only one of the PAN identifier fields when both source and destination addresses are present. If this subfield is set to one and both the source and destination addresses are present, the frame shall contain only the Destination PAN Identifier field and the Source PAN Identifier field shall be assumed equal to that of the destination. If this subfield is set to zero and both the source and destination addresses are present, the frame shall contain both the Source PAN Identifier and Destination PAN Identifier fields. If only one of the addresses is present, this subfield shall be set to zero, and the frame shall contain the PAN Identifier field corresponding to the address. If neither address is present, this subfield shall be set to zero, and the frame shall not contain either PAN Identifier field.

5.3.6 Destination Addressing Mode

The Destination Addressing Mode subfield is 2 bits in length and shall be set to one of the nonreserved values as listed in the following table. If this subfield is equal to zero and the Frame Type subfield does not specify that this frame is an acknowledgment or beacon frame, the Source Addressing Mode subfield shall be nonzero, implying that the frame is directed to the PAN coordinator with the PAN identifier as specified in the Source PAN Identifier field.

Table 5-2. Possible values of the Destination Addressing Mode and Source Addressing Mode subfields

Addressing Mode Value b_1b_0	Description
00	PAN Identifier and address fields are not present
01	Reserved
10	Address field contains a 16-bit short address
11	Address field contains a 64-bit extended address

5.3.7 Frame Version

The Frame Version subfield is 2 bits in length and specifies the version number corresponding to the frame. This subfield shall be set to 0x00 to indicate a frame compatible with IEEE Std 802.15.4-2003 and 0x01 to indicate an IEEE 802.15.4 frame. All other subfield values shall be reserved for future use.

5.3.8 Source Addressing Mode

The Source Addressing Mode subfield is 2 bits in length and shall be set to one of the nonreserved values listed in Table 5-2. If this subfield is equal to zero and the Frame Type subfield does not specify that this frame is an acknowledgment frame, the Destination Addressing Mode subfield shall be nonzero, implying that the frame has originated from the PAN coordinator with the PAN identifier as specified in the Destination PAN Identifier field.

5.4 Network Header

5.4.1 Hops Field

The Hops field provides the number of hops the packet is allowed to be retransmitted. For example, 00h indicates that the packet is not retransmitted. Maximum possible hop is 0xFF.

5.4.2 Frame Control Field

The Frame Control field is a bitmap which defines the behavior of a packet as shown in the following figure.

Figure 5-3. Network Header – Frame Control Field

Bits:0-1	2	3	4	5	6-7
Frame Type	Security Enabled	Infra Cluster	Ack. Request	Address same as MAC	Reserved

The following table details the Frame Control field of the Network Header.

Table 5-3. Network Header Frame Control Field Description

Bit Number	Field Name	Description
6-7	Reserved	Set the bit as '0' for this implementation.
5	Address same as MAC	This bit is set when the MAC Address fields and Network Address fields are same. This is useful when the sleeping end-device polls the parent for data, with relatively less bytes over-the-air for single hop from the network layer.

...........continued

Bit Number	Field Name	Description
4	Acknowledgment Request	This bit is set when the source device requests an Network layer acknowledgment of receipt from the destination device.
3	Intra Cluster	Reserved in this implementation. Set the bit as ‘1’.
2	Security Enabled	This bit is set when data packet is encrypted at the application level.
0-1	Frame Type	These bits indicate as following: • 00 – Data • 01 – Command • 10 – Manufacturer specific • 11 – Reserved

5.4.3 Sequence Number Field

The Sequence Number field is 1 byte in length and specifies the sequence identifier for the frame. The Sequence Number field shall be increased by 1 for every outgoing frame, originating on the node and it must not be changed for routed frames.

5.4.4 Destination PANID Field

The Destination PANID field is 2 bytes in length, specifies the PAN identifier of the intended recipient of the frame. This field will be present only if Address is same as MAC bit which is set to 0.

5.4.5 Source Address Field

The Source Address field is 2 bytes in length and specifies the network address of the node originating the frame.

5.4.6 Destination Address Field

The Destination Address field is 2 bytes in length and specifies the network address of the destination node. The Destination Address field can be set as per the following table for other frames except unicast to a node. Data transmission using long address is not supported.

Table 5-4. Network Header Destination Address Field Description

Destination Address Value	Description
0xFFFF	Broadcast to every device
0xFFFE	Multicast to all FFD’s
0xFFFD	Multicast to all Coordinators

5.4.7 Auxiliary Security Header Field

The Auxiliary Security Header field specifies information required for security processing, including how the frame is protected (security level) and frame counter. This field shall be present only if the Security Enabled sub-field in Frame control field is set to one.

Table 5-5. Auxiliary Security Header Field

Bytes: 1	4	8
Security Level	Frame Counter	Source long address

5.4.7.1 Security Level

The supported security level are:

- 0 (No Security)

- 1 (Authentication - 4 bytes MIC)
- 4 (Encryption only)
- 5 (Encryption with Authentication - 4 bytes MIC)

5.5 MiWi Mesh – Device Addressing Mechanism

The MiWi Mesh uses a 2 bytes short address to specify nodes in the network when performing routing across the network. The address is allocated during the joining process. The lower byte is used to identify the end-devices. The higher byte is used to identify the coordinators.

Bit 15:8	Bit 7	Bit 6:0
Coordinator identifier	RxOnWhenIdle	End-device identifier

5.6 MiWi Mesh – Networking

The MiWi Mesh network features are categorized as follows:

1. Network commissioning
2. Start and join network
3. Routing in network

5.6.1 Network Commissioning

The network commissioning controls the devices which can participate in the network.

1. Application on the PAN Coordinator reads the IEEE address (for example, it can be improved to read from bar code) from one or more devices.
2. PAN coordinator calculates the 64 byte bloom filter value with the read information.
3. Calculated bloom filter value is sent to all the coordinators in the network.
4. Coordinators provide beacon to only the devices which have its IEEE address in the bloom filter.

By default, the PAN Coordinator allows any device to join since BLOOM_AUTO_JOIN is enabled which means no filtering of IEEE addresses.

If the user wants to filter devices based on the IEEE addresses set through MiApp_Commissioning_AddNewDevice, then the user needs to set BLOOM_AUTO_JOIN to 0 using MiApp_Set API.

The user can add new devices using MiApp_Commissioning_AddNewDevice during run time as well. The network will allow those newly added devices.

5.6.2 Start and Join Network

1. Only the PAN coordinator can start a network.
2. Joining device sends a beacon request to obtain information about the available networks in its personal operating space.
3. The PAN coordinator or coordinator evaluates the beacon request by parsing the given IEEE address with the bloom filter value. If found, it sends a beacon frame with a beacon payload which includes PAN coordinator hop count and bloom filter value (64 bytes). If not found, it discards the packet.
4. Upon receiving the beacon frames, the joining device parses it and checks its own address in the bloom filter value and then decides its parent based on associate permit, children capacity and Link Quality Indicator (LQI) of the received beacons. After choosing the parent, it unicasts Mesh Connection Request packet (includes its capability and JoinWish field) to the selected parent.
   The JoinWish field has 2-bits C and ED, and remaining bits are reserved.
   - If both bits are set in JoinWish, then the particular device joins as an end-device if the coordinator capacity is currently unavailable in the network.
   - If only the C bit is set, then the device joins as a Coordinator only.
   - If only the ED bit is set, then the device joins as an end-device only.

5. If the parent is the PAN coordinator and the JoinWish field is set with C and ED or C only, then the PAN coordinator checks whether it has a new coordinator address. If available, it sends the Mesh Connection Response with device address as allocated new coordinator address. If address is unavailable or JoinWish field has only ED set, then it allocates the end-device address and sends the Mesh connection Response.
6. If the parent is the coordinator, then it allocates end-device address and sends Mesh Connection Response with device address as allocated end-device address.
7. The joining device parses the Mesh Connection Response and uses the received network address along with the received network key for further communications in the network.
8. The joining device which is coordinator capable, receives an end-device address, and based on Role Upgrade Timeout (configurable), the device sends a role upgrade request packet to the PAN coordinator in order to upgrade its role from an end-device to the coordinator.
9. When the PAN coordinator receives a role upgrade request, it checks whether coordinator address is available. If the address is available, it allocates a new coordinator address and sends the role upgrade response with the allocated address and status as success. If an address is unavailable, then it sends a role upgrade response with failure status.

5.6.3 Routing in Network

1. During the joining procedure and role upgrade, the route table is updated in all the coordinators.
2. The route table in coordinators is used to route the packet to the destination device.
3. When the device does not have the next hop address for the destination, it will trigger a broadcast for a route request to the destination.
4. Unlike the legacy route request in AODV routing protocols, the reply is generated from any node which has the next hop information in its routing table.
5. The source device (which initiated the route request) selects the route reply for the destination based on the fewer hops and best LQI.
6. However, to establish and synchronize the network periodically, the route table update is broadcasted to a single hop based on pre-configured intervals.
7. This ensures that the coordinators in the network share the neighbor's information with its neighbors.

5.7 Macros for MiWi Mesh

This section describes the macros for the MiWi Mesh.

5.7.1 CHANNEL_MAP

Description	Channel map is a bit map used to select appropriate channels for starting or establishing connection in the network.
Default Value	• For SAMR21 - (1<<25) • For SAMR30 - (1<<2)
Range	Bit map based on the physical layer. Set or clear of any bits in the below range is valid. • For 2.4GHz (SAMR21) – 0x07FFF800 • For SubGHz (SAMR30) – 0x000007FF
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	For SAMR30, only channels 1-10 can be changed using this Macro. To use Channel 0, `PHY_Init()` must be modified to include TX Power and PHY Mode setting as per the recommendation from the data sheet for European band.

5.7.2 KEEP_ALIVE_COORDINATOR_SEND_INTERVAL

Description	Time interval in seconds on which a coordinator capable device sends keep alive frame to PAN coordinator. Upon reception of this frame, PAN coordinator refreshes the timeout for that particular coordinator.
Default Value	120
Range	1 – 65535
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	`KEEP_ALIVE_COORDINATOR_TIMEOUT_IN_SEC` is based on this value.

5.7.3 KEEP_ALIVE_COORDINATOR_TIMEOUT_IN_SEC

Description	Timeout in seconds for which the PAN coordinator maintains the entry of coordinator, for holding its address. Each coordinator is expected to send at least one keep alive frame to PANC within this timeout.
Default Value	KEEP_ALIVE_COORDINATOR_SEND_INTERVAL *10 The default value is 1200 when KEEP_ALIVE_COORDINATOR_SEND_INTERVAL is set as 120.
Range	1 – 65535
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.4 KEEP_ALIVE_RXONENDDEVICE_SEND_INTERVAL

Description	Time interval in seconds on which an end-device sends keep alive frame to its coordinator. Upon reception of this frame, coordinator refreshes the timeout for that particular end-device.
Default Value	120
Range	1 – 65535
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	`KEEP_ALIVE_RXONENDDEVICE_TIMEOUT_IN_SEC` is based on this value.

5.7.5 KEEP_ALIVE_RXONENDDEVICE_TIMEOUT_IN_SEC

Description	Timeout in seconds for which the coordinator maintains the entry of end-device, for holding its address. Each end-device is expected to send at least one keep alive frame to coordinator within this timeout.
Default Value	KEEP_ALIVE_RXONENDDEVICE_SEND_INTERVAL *10 (that is, 1200) The default value is 1200 when KEEP_ALIVE_COORDINATOR_SEND_INTERVAL is set as 120.
Range	1 – 65535
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.6 DATA_REQUEST_SEND_INTERVAL

Description	Time interval in seconds on which a sleeping end-device sends Data Request frame to its coordinator. Upon reception of this frame, coordinator refreshes the timeout for that particular sleeping end-device and sends any data cached in indirect queue.
Default Value	3
Range	1 – 254
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	`RXOFF_DEVICE_TIMEOUT_IN_SEC` and `MAXIMUM_DATA_REQUEST_SEND_INTERVAL` is based on this value

5.7.7 RXOFF_DEVICE_TIMEOUT_IN_SEC

Description	Timeout in seconds for which the coordinator maintains the entry of sleeping end-device, to hold its address. Each sleeping end-device is expected to send at least one Data Request to coordinator within this timeout.
Default Value	DATA_REQUEST_SEND_INTERVAL * 20 (that is, 60) The default value is 60 when DATA_REQUEST_SEND_INTERVAL is set as 3.
Range	1 – 65535
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.8 MAXIMUM_DATA_REQUEST_SEND_INTERVAL

Description	Maximum time interval in seconds for Data Request of end-device in the network.
Default Value	DATA_REQUEST_SEND_INTERVAL * 2 (that is, 6)
Range	1 – 254
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.9 MAX_NUMBER_OF_DEVICES_IN_NETWORK

Description	This macro is used to configure the number of device's IEEE addresses to be stored for commissioning.
Default Value	32
Range	1 – 255
Memory Usage	256 bytes of RAM for 32 entries, that is, 8 bytes per entry
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.10 JOIN_WISH

Description	Configuration to join the network based on defined roles. For more information, see 5.6.2 Start and Join Network.
Default Value	• For Coordinator – JOINWISH_ANY • For End-device – JOINWISH_ENDEVICE
Range	• JOINWISH_ENDEVICE - 0x01 • JOINWISH_COORD_ALONE - 0x02 • JOINWISH_ANY - 0x03
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.11 ROLE_UPGRADE_INTERVAL_IN_SEC

Description	Time interval in seconds on which a coordinator capable end-device requests the PANC to upgrade its role to coordinator. For more information on Role Upgrade, see 5.6.2 Start and Join Network.
Default Value	25
Range	1 – 254
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.12 CONNECTION_RESPONSE_WAIT_IN_SEC

Description	Time interval in seconds to wait for Connection Response after sending Connection Request to any coordinator in the network.
Default Value	5
Range	1 – 254
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.13 NUM_OF_COORDINATORS

Description	This macro is used to configure the number of coordinators in the network. Also used to allocate the coordinator table on PANC to maintain the IEEE addresses and timeout for each coordinator.
Default Value	64
Range	1 – 200
Memory Usage	768 bytes of RAM for 64 entries, that is, 12 bytes per entry
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.14 NUM_OF_NONSLEEPING_ENDDEVICES

Description	This macro is used to configure the number of non-sleeping end-devices in the network. Also used to allocate the device table on each coordinator to maintain the IEEE addresses and timeout for each non-sleeping end-device.
Default Value	5
Range	1 – 127
Memory Usage	80 bytes of RAM for 5 entries, that is, 16 bytes per entry
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.15 NUM_OF_SLEEPING_ENDDEVICES

Description	This macro is used to configure the number of sleeping end-devices in the network. Also used to allocate the sleeping device table on each coordinator to maintain the IEEE addresses and timeout for each sleeping end-device.
Default Value	5
Range	1 – 128
Memory Usage	100 bytes of RAM for 5 entries, that is, 20 bytes per entry
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.16 ROUTE_UPDATE_INTERVAL

Description	Periodic time interval in seconds to send route update for neighboring devices after joining the network.
Default Value	60
Range	1 – 254
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.17 ROUTE_REQ_WAIT_INTERVAL

Description	Timeout in seconds to wait for route replies after sending route request to discover route for a specific coordinator.
Default Value	5
Range	1 – 254
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.18 INDIRECT_DATA_WAIT_INTERVAL

Description	Timeout in seconds to hold indirect data to its sleeping end-devices. This must be maintained at least more than twice the Data Request interval to ensure reliable data transfer.
Default Value	25
Range	1 – 254
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.19 ED_LINK_FAILURE_ATTEMPTS

Description	Number of consecutive attempts on end-device made with the parent before confirming link failure.
Default Value	15
Range	1 – 254
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.20 FRAME_RETRY

Description	Defines the number of retries to be performed during failure to reach the destination.
Default Value	3
Range	0 – 254
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	This configuration to retry is apart from the basic MAC level retries. **Note:** Any frame going out of the device is retried three times at MAC layer.

5.7.21 REBROADCAST_TABLE_SIZE

Description	This macro is used to configure the number of entries to be stored to avoid duplicate rebroadcast for every broadcast in the network.
Default Value	10
Range	1 – 255
Memory Usage	40 bytes of RAM for 10 entries, that is, 4 bytes per entry.
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.22 REBROADCAST_TIMEOUT

Description	Timeout in seconds to hold the broadcasted data in rebroadcast table to avoid rebroadcasting again.

Default Value	5
Range	1 – 254
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.23 DUPLICATE_REJECTION_TABLE_SIZE

Description	Size of duplicate rejection table used to avoid multiple data indication to the application.
Default Value	10
Range	1 – 255
Memory Usage	40 bytes of RAM for 10 entries, that is, 4 bytes per entry.
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.24 MAX_BEACON_RESULTS

Description	Number of entries allocated to receive beacon response during active scan.
Default Value	5
Range	1 – 255
Memory Usage	90 bytes of RAM for 5 entries, that is, 18 bytes per entry.
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.25 MESH_SECURITY_LEVEL

Description	Security levels for CCM* as defined in IEEE 802.15.4.
Default Value	5
Range	0 – 7
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.7.26 PUBLIC_KEY_DEFAULT

Description	Public key is the initial key stored in all devices, the initial communications use this key until it gets the network key.
Default Value	{0x00,0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F}
Range	Any 16 bytes value
Memory Usage	None
Configurable in	`miwi_config_mesh.h`

Remarks	None

5.7.27 NETWORK_KEY_DEFAULT

Description	Network key used to transact after successful join to the network.
Default Value	{0x00,0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF}
Range	Any 16 bytes value
Memory Usage	None
Configurable in	`miwi_config_mesh.h`
Remarks	None

5.8 Recommendation for Macros

For example, consider the default tested network (with WSN Demo application) which has following considerations.

1. Network size as 1 PAN coordinator, 50 coordinators, and 30 end-devices.
2. All the devices in network must report to PAN Coordinator at a periodic interval.
3. Data flow is mostly unidirectional (that is, uplink requires more bandwidth).
4. Report from all the devices are monitored using WSNMonitor.

This network is programmed with default values (see 5.7 Macros for MiWi Mesh), and tested to be working for more than 48 hours.

By default, in the WSN Demo application, each end-device joins to a coordinator based on its available end-device capacity. Since the number of coordinators are greater than end-devices, most end-devices join to coordinators with 100% end-device capacity.

The following macros must be proportionally modified (that is, when the number of coordinators increases, configuration values must be increased and vice versa) based on the change in number of coordinators in the network.

1. NUM_OF_COORDINATORS
2. KEEP_ALIVE_COORDINATOR_SEND_INTERVAL
3. KEEP_ALIVE_COORDINATOR_TIMEOUT_IN_SEC
4. ROUTE_UPDATE_INTERVAL
5. ROLE_UPGRADE_INTERVAL_IN_SEC

The following macros must be proportionally modified (that is, when the number of coordinators increases, configuration values must be increased and vice versa) based on the change in number of end-devices in the network.

- For sleeping end-device:
  1. NUM_OF_SLEEPING_ENDDEVICES
  2. DATA_REQUEST_SEND_INTERVAL
  3. RXOFF_DEVICE_TIMEOUT_IN_SEC
  4. MAXIMUM_DATA_REQUEST_SEND_INTERVAL
  5. INDIRECT_DATA_WAIT_INTERVAL
- For non-sleeping (that is, RXON end-device) end-device:
  1. NUM_OF_NONSLEEPING_ENDDEVICES
  2. KEEP_ALIVE_RXONENDDEVICE_SEND_INTERVAL
  3. KEEP_ALIVE_RXONENDDEVICE_TIMEOUT_IN_SEC

Note: This recommendation of macros is tested on SAMR21 (that is, 2.4 GHz); therefore, for the same network size in SAMR30 (that is, Sub-GHz) the values must be increased to compensate on the reduction in data rate.

5.9 Extending Battery Life for Sleeping End-device

Apart from the Sleep mode supported by the controller, DATA_REQUEST_SEND_INTERVAL configuration directly impacts the frequency of wake up from sleep and the consumption of battery.

6. Network Freezer

The Network Freezer feature saves critical network information into the Nonvolatile Memory (NVM) and restores them after power cycle. In this way, the application supports the power cycle scenario and the network can be restored to the previous state of the power cycle without many message exchanges after the power cycle.

Additionally, wear-leveling implementation reduces the number of “backup-erase-re-write” cycles and thereby improves the Flash lifetime. Refer to the `miwi_mesh_pds.c` file which specifies information about the parameters stored in the NVM.

6.1 Interface

The Network Freezer feature is enabled by defining `ENABLE_NETWORK_FREEZER` API in the configuration file of the application project. This feature is invoked by calling the MiApp function `MiApp_ProtocolInit`. When Network Freezer is enabled in the application, the network information is restored from NVM; otherwise, the wireless node starts from initial stage. If Network Freezer is disabled, the node always starts as a factory new device.

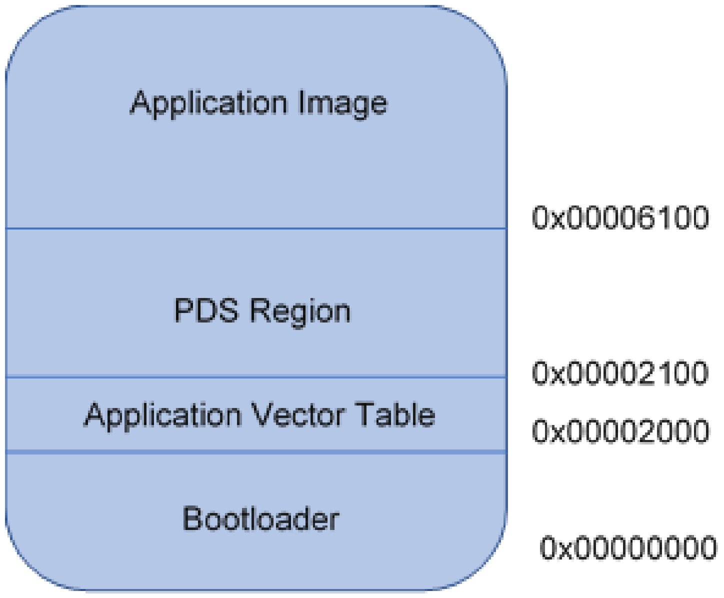
6.2 Additional Notes

The Network Freezer feature requires NVM to store the critical network information. The NVM used for this implementation is the internal Flash.

6.3 Default Memory Layout

The details of memory layout with Network Freezer (PDS – Persistent Data Storage) enabled is shown below. The user can change the size of the sector by changing D_NV_SECTOR_SIZE in code and linker file correspondingly.

Figure 6-1. Default Memory Layout



7. Sleep Mode

For most of the applications, it is critical to provide long battery life for the sleeping devices. A device can be in either the Active mode or Sleep mode. After being powered-up, a node always starts in the Active mode, with its MCU fully turned on. An application can check whether the stack is allowing it to sleep or not using the `ENABLE_SLEEP_FEATURE` API. If it allows, the application can go to sleep at a maximum of allowable time by stack for proper operation.

In the Sleep mode, the RF chip and the MCU are in Low-Power state and only the functionality required for MCU wake up remains active. Thus, in Sleep mode, the application cannot perform any radio Tx/Rx operations or communicate with the external periphery.

Major power is consumed during the Active mode, requesting for and sending data in the duty cycle. Therefore, for a device to be active is based on its polling period. This can be controlled using a configuration option. Among all nodes, only end-devices can sleep.

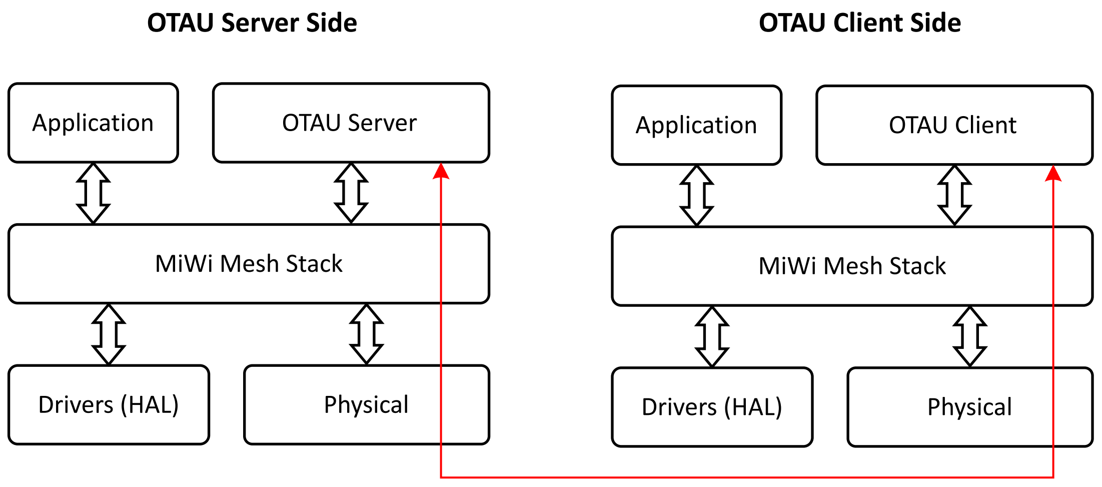
7.1 Interface

The Sleep mode can be enabled by defining `ENABLE_SLEEP_FEATURE` API in the configuration file of the application project. For more details, see MiApp API Description.

8. Over-The-Air Upgrade

The following figure shows the firmware architecture of the Over-The-Air Upgrade (OTAU).

Figure 8-1. OTAU Firmware Architecture



8.1 OTAU Server

The OTAU server receives or transmits the command from or to the PC through UART or USB. To upgrade, transmit required frames through MiWi Mesh stack layer to reach the clients. The server acts as a bridge between the clients and an OTAU tool running in the PC; that is, there is no additional intelligence in OTAU module on the server end.

8.2 OTAU Client

The OTAU client receives or transmits the proprietary commands over-the-air to communicate with the OTAU server.

8.3 Domains of OTAU

The following sections describe the Notify and Upgrade domains of the OTAU server and client.

8.3.1 Notify

1. Provides basic information about the client such as, IEEE address, short address, and next hop address to reach the OTAU server for plotting network topology.
2. Commands to power LED on clients to identify visually on large network.
3. Provision for user to fetch additional information related to the application such as, firmware name, firmware version, board name, and board version.

8.3.2 Upgrade

- Supports OTAU of each client through proprietary protocol exchange.
- Provides support to switch to an new image individually when all the nodes are upgraded.

8.4 Compiler Switches for OTAU

OTAU_ENABLED
OTAU_ENABLED switch must be included in project symbols to enable the upgrade support.

OTAU_SERVER
When OTAU_SERVER switch is enabled on the project symbols, the node acts as the OTAU server. If this symbol is not enabled, then the node acts as a Client for OTAU.

9. MiApp APIs

The following table lists the supported APIs.

Table 9-1. MiApp API

S. No.	Supported APIs	Topology Supported
1	`miwi_status_t MiApp_ProtocolInit (defaultParametersRomOrRam_t *defaultRomOrRamParams, defaultParametersRamOnly_t *defaultRamOnlyParams)`	P2P/Star/Mesh
2	`bool MiApp_Set(enum id, uint8_t value )`	P2P/Star/Mesh
3	`bool MiApp_StartNetwork(uint8_t Mode, uint8_t ScanDuration, uint32_t ChannelMap, FUNC ConfCallback)`	P2P/Star/Mesh
4	`uint8_t MiApp_SearchConnection(uint8_t ScanDuration, uint32_t ChannelMap, FUNC ConfCallback)`	P2P/Star/Mesh
5	`uint8_t MiApp_EstablishConnection(uint8_t Channel, uint8_t addr_len, uint8_t addr, uint8_t Capability_info, FUNC ConfCallback)`	P2P/Star/Mesh
6	`void MiApp_RemoveConnection(uint8_t ConnectionIndex)`	P2P/Star/Mesh
7	`void MiApp_ConnectionMode(uint8_t Mode)`	P2P/Star/Mesh
8	`MiApp_SendData(uint8_t addr_len, uint8_t addr, uint8_t len, uint8_t pointer, FUNC ConfCallback)`	P2P/Star/Mesh
9	`MiApp_SubscribeDataIndicationCallback(FUNC callback)`	P2P/Star/Mesh
10	`uint8_t MiApp_NoiseDetection(uint32_t ChannelMap, uint8_t ScanDuration, uint8_t DetectionMode, OUTPUT uint8_t NoiseLevel)`	P2P/Star/Mesh
11	`uint8_t MiApp_TransceiverPowerState(uint8_t Mode)`	P2P/Star/Mesh
12	`bool MiApp_InitChannelHopping(uint32_t ChannelMap)`	P2P/Star/Mesh
13	`bool MiApp_ResyncConnection(uint8_t ConnectionIndex, uint32_t ChannelMap)`	P2P/Star/Mesh
14	`uint8_t Total_Connections(void)`	P2P/Star
15	`void MiApp_BroadcastConnectionTable()`	Star
16	`bool MiApp_Set(enum id, uint8_t value )`	Mesh
17	`bool MiApp_IsMemberOfNetwork(void)`	Mesh
18	`bool MiApp_Get(enum id, uint8_t value )`	Mesh
19	`bool MiApp_Set(enum id, uint8_t value )`	Mesh
20	`bool MiApp_SubscribeReConnectionCallback(ReconnectionCallback_t callback)`	P2P/Star/Mesh
21	`bool MiApp_ResetToFactoryNew(void)`	P2P/Star/Mesh
22	`bool MiApp_ReadyToSleep(uint32_t* sleepTime)`	Mesh

...........continued

S. No.	Supported APIs	Topology Supported
23	`bool MiApp_ManuSpecSendData(uint8_t addr_len, uint8_t *addr, uint8_t msglen, uint8_t *msgpointer, uint8_t msghandle, bool ackReq, DataConf_callback_t ConfCallback)`	Mesh
24	`bool MiApp_SubscribeManuSpecDataIndicationCallback(PacketIndCallback_t callback)`	Mesh
25	`bool MiApp_IsConnected(void)`	Mesh
26	`uint16_t MiApp_MeshGetNextHopAddr(uint16_t destAddress)`	Mesh

10. MiApp API Description

This section describes the MiApp APIs.

10.1 MiApp_ProtocolInit

API	`miwi_status_t MiApp_ProtocolInit(defaultParametersRomOrRam_t *defaultRomOrRamParams, defaultParametersRamOnly_t *defaultRamOnlyParams)`
Description	This is the primary user interface function to initialize the Microchip proprietary wireless protocol, which is chosen by the application layer. Usually, this function must be called after the hardware initialization, before any other MiApp interface can be called.
Pre-Condition	Hardware initialization must be done.
Parameters	• `defaultParametersRomOrRam_t defaultRomOrRamParams` – Default parameters for MiWi™ Mesh. • `defaultParametersRamOnly_t defaultRamOnlyParams` – Default parameters for MiWi™ Mesh. • Ignored in case of P2P / Star
Returns	Status of Initialization

Example	`<code>`
	`HardwareInit();`
	`MiApp_ProtocolInit();`
	`</code>`

Remarks	If RECONNECTION_IN_PROGRESS status is received, then application needs to wait for reconnection callback before proceeding to call further MiApp API's.

10.2 MiApp_Set

API	`bool MiApp_Set(set_params id, uint8_t *value)`
Description	This is the primary user interface function to set the different values in the MiWi™ stack.
Pre-Condition	Protocol initialization must be done.
Parameters	• `set_params id` – The identifier of the value to be set • `value` – The value to be set
Returns	A boolean to indicate if set operation is performed successfully.

Example	`<code>`
	`if( true == MiApp_Set(CHANNEL, 15) )`
	`{`
	`// channel changes successfully`
	`}`
	`</code>`

Remarks	None

10.3 MiApp_StartConnection

API	`bool MiApp_StartConnection(uint8_t Mode, uint8_t ScanDuration, uint32_t ChannelMap,connectionConf_callback_t ConfCallback)`
Description	This is the primary user interface function for the application layer to start PAN. Usually, this function is called by the PAN coordinator which is the first in the PAN. The PAN coordinator may start the PAN after a noise scan if specified in the input mode.
Pre-Condition	Protocol initialization must be done.
Parameters	• `uint8_t Mode` – whether to start a PAN after a noise scan. Possible modes are as follows. – START_CONN_DIRECT – starts PAN directly without noise scan. – START_CONN_ENERGY_SCN – performs an energy scan first, then starts the PAN on the channel with least noise. – START_CONN_CS_SCN – performs a carrier-sense scan first, then starts the PAN on the channel with least noise. • `uint8_t ScanDuration` – maximum time to perform scan on single channel. The value is from 5 to 14. The real time to perform scan can be calculated in following formula from IEEE 802.15.4 specification: 960 x (2^ScanDuration + 1) x 10^(-6) second ScanDuration is discarded if the connection mode is START_CONN_DIRECT. • `uint32_t ChannelMap` – bit map of channels to perform noise scan. The 32-bit double word parameter uses one bit to represent corresponding channels from 0 to 31. For instance, 0x00000003 represent to scan channel 0 and channel 1. `ChannelMap` is discarded if the connection mode is START_CONN_DIRECT. • `connectionConf_callback_t ConfCallback` – callback routine which is called upon the initiated connection procedure is performed.
Returns	A boolean to indicate if PAN is started successfully.

Example	`<code>`
	`// start the PAN on the least noisy channel after scanning all possible channels.`
	`MiApp_StartConnection(START_CONN_ENERGY_SCN, 10, 0x07FFF800, callback);`
	`</code>`

Remarks	None

10.4 MiApp_SearchConnection

API	`uint8_t MiApp_SearchConnection(uint8_t ScanDuartion, uint32_t ChannelMap, SearchConnectionConf_callback_t ConfCallback)`
Description	This is the primary user interface function for the application layer to perform an active scan. After this function call, all active scan response is stored in the global variable `ActiveScanResults` in the format of structure ACTIVE_SCAN_RESULT. The return value indicates the total number of valid active scan response in the active scan result array.
Pre-Condition	Protocol initialization is done.

Parameters	`uint8_t ScanDuration` – maximum time to perform scan on single channel. The value is from 5 to 14. The real time to perform scan can be calculated with the following formula from the IEEE 802.15.4 specification: 960 x (2^ScanDuration + 1) x 10^(-6) second. `uint32_t ChannelMap` – bit map of channels to perform noise scan. The 32-bit double word parameter uses one bit to represent corresponding channels from 0 to 31. For instance, 0x00000003 represents to scan channel 0 and channel 1. `SearchConnectionConf_callback_t ConfCallback` – callback routine which is called when the initiated connection procedure is performed.
Returns	The number of valid active scan response stored in the global variable `ActiveScanResults`.

Example	`<code>`
	`// Perform an active scan on all possible channels`
	`NumOfActiveScanResponse = MiApp_SearchConnection(10, 0xFFFFFFFF, callback);`
	`</code>`

Remarks	None

10.5 MiApp_EstablishConnection

API	`uint8_t MiApp_EstablishConnection(uint8_t Channel, uint8_t addr_len, uint8_t *addr, uint8_t Capability_info, connectionConf_callback_t ConfCallback)`
Description	This is the primary user interface function for the application layer to start communication with an existing PAN. For P2P protocol, this function call can establish one or more connections. For network protocol, this function can be used to join the network, or establish a virtual socket connection with a node out of the radio range.
Pre-Condition	Protocol initialization is done. If only to establish connection with a predefined device, an active scan must be performed before and valid active scan result must be saved.
Parameters	• `uint8_t channel` – selected channel to invoke join procedure. • `uint8_t addr_len` – address length • `uint8_t *addr` – address of the parent • `uint8_t Capability_info` – capability information of the device • `connectionConf_callback_t ConfCallback` – callback routine which will be called upon the initiated connection procedure is performed
Returns	The index of the peer device on the connection table.

Example	`<code>`
	`// Establish one or more connections with any device`
	`PeerIndex = MiApp_EstablishConnection(14, 8, 0x12345678901234567,0x80, callback);`
	`</code>`

Remarks	If more than one connections is established through this function call, the return value points to the index of one of the peer devices.

10.6 MiApp_RemoveConnection

API	`void MiApp_RemoveConnection(uint8_t ConnectionIndex)`

Description	This is the primary user interface function to disconnect connection(s). For a P2P protocol, it removes the connection. For a network protocol, if the device referred by the input parameter is the parent of the device calling this function, the calling device gets out of network along with its children. If the device referred by the input parameter is children of the device calling this function, the target device gets out of network.
Pre-Condition	Transceiver is initialized. Node establishes one or more connections.
Parameters	`uint8_t ConnectionIndex` – index of the connection in the connection table to be removed.
Returns	None

Example	`<code>`
	`MiApp_RemoveConnection(0x00);`
	`</code>`

Remarks	None

10.7 MiApp_ConnectionMode

API	`void MiApp_ConnectionMode(uint8_t Mode)`
Description	This is the primary user interface function for the application layer to configure the way that the host device accepts the connection request.
Pre-Condition	Protocol initialization is done.
Parameters	`uint8_t Mode` - mode to accept the connection request. The privilege for those modes decreases gradually as defined. The higher privilege mode has all the rights of the lower privilege modes. The possible modes are as follows: • ENABLE_ALL_CONN – enables response to all connection request • ENABLE_PREV_CONN – enables response to connection request from device already in the connection table • ENABLE_ACTIVE_SCAN_RSP – enables response to active scan only • DISABLE_ALL_CONN – disables response to the connection request, including an active scan request
Returns	None

Example	`<code>`
	`// Enable all connection request`
	`MiApp_ConnectionMode(ENABLE_ALL_CONN);`
	`</code>`

Remarks	None

10.8 MiApp_SendData

API	`bool MiApp_SendData(uint8_t addr_len, uint8_t *addr,`
	`uint8_t msglen, uint8_t *msgpointer,uint8_t msghandle,`
	`bool ackReq, DataConf_callback_t ConfCallback)`

Description	This is one of the primary user interface functions for the application layer to unicast a message. The destination device is specified by the input parameter `DestinationAddress`. The application payload is filled using `msgpointer`.
Pre-Condition	Protocol initialization is done.
Parameters	• `uint8_t addr_len` – destination address length • `uint8_t *addr` – destination address • `uint8_t msglen` – length of the message • `uint8_t *msgpointer` – message/frame pointer • `uint8_t msghandle` – message handle • `bool ackReq` – set to receive network level acknowledgment **Note:** Discarded for broadcast data. • `DataConf_callback_t ConfCallback` – callback routine which is called when the initiated data procedure is performed.
Returns	A boolean to indicate if the unicast procedure is successful.

Example	`<code>`
	`// Secure and then broadcast the message stored in msgpointer to the permanent address`
	`// specified in the input parameter.`
	`MiApp_SendData(SHORT_ADDR_LEN, 0x0004, 5, “hello”,1, callback);`
	`</code>`

Remarks	None

10.9 MiApp_SubscribeDataIndicationCallback

API	`bool MiApp_SubscribeDataIndicationCallback(PacketIndCallback_t callback)`
Description	This is the primary user interface functions for the application layer to call the Microchip proprietary protocol stack to register the message indication callback to the application. The function calls the protocol stack state machine to keep the stack running.
Pre-Condition	Protocol initialization is done.
Parameters	None
Returns	A boolean to indicate if the subscription operation is successful or not.

Example	`<code>`
	`if( true == MiApp_SubscribeDataIndicationCallback(ind) )`
	`{`
	`}`
	`</code\>`

Remarks	None

10.10 MiApp_NoiseDetection

API	`uint8_t MiApp_NoiseDetection( uint32_t ChannelMap, uint8_t ScanDuration, uint8_t DetectionMode, uint8_t NoiseLevel)`
Description	This is the primary user interface function for the application layer to perform noise detection on multiple channels.

Pre-Condition	Protocol initialization is done.
Parameters	• `uint32_t ChannelMap` – bit map of channels to perform a noise scan. The 32-bit double word parameter uses one bit to represent corresponding channels from 0 to 31. For example, 0x00000003 represents to scan channel 0 and channel 1. • `uint8_t ScanDuration` – maximum time to perform a scan on a single channel. The valid value is from 5 to 14. The real time to perform a scan can be calculated in the following formula from IEEE 802.15.4 specification: 960 x (2^ScanDuration + 1) x 10^(-6) second • `uint8_t DetectionMode` – the noise detection mode to perform the scan. The two possible scan modes are as following. – `NOISE_DETECT_ENERGY` – Energy detection scan mode – `NOISE_DETECT_CS` – Carrier sense detection scan mode • `uint8_t NoiseLevel` - noise level at the channel with least noise level
Returns	The channel that has the lowest noise level.

Example	`<code>`
	`uint8_t NoiseLevel;`
	`OptimalChannel = MiApp_NoiseDetection(0xFFFFFFFF, 10, NOISE_DETECT_ENERGY,`
	`&NoiseLevel);`
	`</code>`

Remarks	None

10.11 MiApp_TransceiverPowerState

API	`uint8_t MiApp_TransceiverPowerState(uint8_t Mode)`
Description	This is the primary user interface function for the application layer to set the RF transceiver into sleep or wake up. This function is only available to those wireless nodes that have to disable the transceiver to save battery power.
Pre-Condition	Protocol initialization is done.
Parameters	`uint8_t Mode` – mode of the power state for the RF transceiver to be set. The possible power states are following. • `POWER_STATE_SLEEP` – deep sleep mode for the RF transceiver • `POWER_STATE_WAKEUP` – Wake-Up state, or operating state for the RF transceiver • `POWER_STATE_WAKEUP_DR` – Set the device into the Wake-Up mode and transmit the data request to the device's associated device
Returns	The status of the operation. The following are the possible status. • SUCCESS – operation is successful. • ERR_TRX_FAIL – Transceiver fails to go to the Sleep or Wake-Up mode. • ERR_TX_FAIL – transmission of Data Request command failed. Only available if the input mode is POWER_STATE_WAKEUP_DR. • ERR_RX_FAIL – failed to receive any response to Data Request command. Only available if the input mode is POWER_STATE_WAKEUP_DR. • ERR_INVLAID_INPUT – invalid input mode.

Example	`<code>`
	`// put RF transceiver into sleep`
	`MiApp_TransceiverPowerState(POWER_STATE_SLEEP;`
	`// Put the MCU into sleep`
	`Sleep();`
	`// wakes up the MCU by WDT, external interrupt or any other means`
	`// make sure that RF transceiver to wake up and send out Data Request`
	`MiApp_TransceiverPowerState(POWER_STATE_WAKEUP_DR);`
	`</code>`

Remarks	None

10.12 MiApp_Get

API	`bool MiApp_Get(set_params id, uint8_t *value )`
Description	This is the primary user interface function to get the different values in the MiWi^{TM} stack
Pre-Condition	Protocol initialization is done
Parameters	`get_params id` – identifier of the value to be set
Returns	A boolean to indicate if the get operation is performed successfully

Example	`<code>`
	`value = MiApp_get(CHANNEL)`
	`</code>`

Remarks	None

10.13 MiApp_RoleUpgradeNotification_Subscribe

API	`bool MiApp_RoleUpgradeNotification_Subscribe (roleUpgrade_callback_t callback)` This is applicable only for coordinator.
Description	This API subscribes to notify the role upgrade. Upon successful role upgrade, callback is called with new short address.
Pre-Condition	Protocol initialization is done.
Parameters	`roleUpgrade_callback_t callback` – callback routine which is called upon the role upgrade completion
Returns	A boolean to indicate if the subscription is success or not

10.14 MiApp_Commissioning_AddNewDevice

API	`bool MiApp_Commissioning_AddNewDevice(uint64_t joinerAddress, bool triggerBloomUpdate)`
Description	This is used to add a device to bloom filter on the PAN coordinator. This function is applicable only for the PAN coordinator.
Pre-Condition	Protocol initialization is done.

Parameters	• `uint8_t joinerAddress` – the IEEE address to be added • `bool triggerBloomUpdate` – if set to true then bloom update is sent
Returns	True if successfully added, false otherwise.

10.15 MiApp_SubscribeReConnectionCallback

API	`bool MiApp_SubscribeReConnectionCallback(ReconnectionCallback_t callback)`
Description	This API subscribes to notify the reconnection after power recycle when the device was in network before power recycle. Upon reconnection on a device, this callback is called.
Pre-Condition	Protocol initialization is done.
Parameters	`ReconnectionCallback_t callback`- callback routine which is called upon reconnection
Returns	A boolean to indicate if the subscription is success or not

10.16 MiApp_ResetToFactoryNew

API	`bool MiApp_ResetToFactoryNew(void)`
Description	This API erases all the persistent items in the NVM and resets the system
Pre-Condition	None
Parameters	None
Returns	A boolean to indicate if the operation is success or not

10.17 MiApp_ReadyToSleep

API	`bool MiApp_ReadyToSleep(uint32_t* sleepTime)`
Description	This API helps to know if the stack is ready to sleep and how much time stack allows to sleep if it is ready
Pre-Condition	None
Parameters	`uint32_t* sleep Time` – pointer to sleep time which gets filled with the sleep time if the stack is ready to sleep
Returns	A boolean to indicate if the stack is ready to sleep or not

10.18 MiApp_ManuSpecSendData

API	`bool MiApp_ManuSpecSendData(uint8_t addr_len, uint8_t *addr,`
	`uint8_t msglen, uint8_t *msgpointer,uint8_t msghandle, bool ackReq,`
	`DataConf_callback_t ConfCallback)`

Description	This is an interface function for the manufacturer-specific data. The destination device is specified by the input parameter `DestinationAddress`. The OTAU module uses this API for upgrade support.
Pre-Condition	Protocol initialization is done.
Parameters	• `uint8_t addr_len` – destination address length • `uint8_t *addr` – destination address • `uint8_t msglen` – length of the message • `uint8_t *msgpointer` – message/frame pointer • `uint8_t msghandle` – message handle • `bool ackReq` – set to receive network level acknowledgment **Note:** Discarded for the broadcast data. • `DataConf_callback_t ConfCallback` - callback routine which is called when the initiated data procedure is performed.
Returns	A boolean indicates if the unicast procedure is successful.

Example	`<code>`
	`// Secure and then broadcast the message stored in msgpointer to the permanent address`
	`// specified in the input parameter.`
	`MiApp_ManuSpecSendData(SHORT_ADDR_LEN, 0x0004, 5, “hello”,1, callback);`
	`</code>`

Remarks	None

10.19 MiApp_SubscribeManuSpecDataIndicationCallback

API	`bool MiApp_SubscribeManuSpecDataIndicationCallback (PacketIndCallback_t callback)`
Description	This is the primary user interface functions for the OTAU module to register for manufacturer-specific indication callback.
Pre-Condition	Protocol initialization is done.
Parameters	None
Returns	A boolean indicates if the subscription operation is successful or not.

Example	`<code>`
	`if( true == MiApp_SubscribeManuSpecDataIndicationCallback (ind) )`
	`{`
	`}`
	`</code\>`

Remarks	None

10.20 MiApp_IsConnected

API	`bool MiApp_IsConnected(void)`
Description	This is used to check the connection of MiWi^{TM} Mesh to a network.
Pre-Condition	None

Parameters	None
Returns	A boolean true indicates the node is connected to a network.

10.21 MiApp_MeshGetNextHopAddr

API	`uint16_t MiApp_MeshGetNextHopAddr(uint16_t destAddress)`
Description	This is used to get the address of next hop to reach the `destAddress`.
Pre-Condition	None
Parameters	`uint16_t destAddress` – destination address of the device to which the next hop is required.
Returns	Address of the next hop to reach the `destAddress`.

11. Limitations

The following are the known limitations:

1. Random behavior in some SAMR30 devices that the Back mode sleep fails to wake up when run continuously for 1 or 2 days.
2. It is possible to miss confirmation callback for data in P2P/Star if initiated too fast. It is recommended to have proper debounce in customer applications.
3. OTAU - Device unable (randomly) to wake up from sleep when CPU works at 48 MHz which is derived using DFLL with External 32 KHz crystal/clock source.

12. Document Revision History

Revision	Date	Section	Description
A	02/2019	Document	Initial Revision
B	08/2019	3. MiWi P2P 4. MiWi Star 6.3 Default Memory Layout	New New New

The Microchip Website

Microchip provides online support via our website at http://www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to http://www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Legal Notice

Trademarks

ISBN: 978-1-5224-4932-4

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit http://www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office	**Australia - Sydney**	**India - Bangalore**	**Austria - Wels**
2355 West Chandler Blvd.	Tel: 61-2-9868-6733	Tel: 91-80-3090-4444	Tel: 43-7242-2244-39
Chandler, AZ 85224-6199	**China - Beijing**	**India - New Delhi**	Fax: 43-7242-2244-393
Tel: 480-792-7200	Tel: 86-10-8569-7000	Tel: 91-11-4160-8631	**Denmark - Copenhagen**
Fax: 480-792-7277	**China - Chengdu**	**India - Pune**	Tel: 45-4450-2828
Technical Support:	Tel: 86-28-8665-5511	Tel: 91-20-4121-0141	Fax: 45-4485-2829
http://www.microchip.com/support	**China - Chongqing**	**Japan - Osaka**	**Finland - Espoo**
Web Address:	Tel: 86-23-8980-9588	Tel: 81-6-6152-7160	Tel: 358-9-4520-820
http://www.microchip.com	**China - Dongguan**	**Japan - Tokyo**	**France - Paris**
Atlanta	Tel: 86-769-8702-9880	Tel: 81-3-6880- 3770	Tel: 33-1-69-53-63-20
Duluth, GA	**China - Guangzhou**	**Korea - Daegu**	Fax: 33-1-69-30-90-79
Tel: 678-957-9614	Tel: 86-20-8755-8029	Tel: 82-53-744-4301	**Germany - Garching**
Fax: 678-957-1455	**China - Hangzhou**	**Korea - Seoul**	Tel: 49-8931-9700
Austin, TX	Tel: 86-571-8792-8115	Tel: 82-2-554-7200	**Germany - Haan**
Tel: 512-257-3370	**China - Hong Kong SAR**	**Malaysia - Kuala Lumpur**	Tel: 49-2129-3766400
Boston	Tel: 852-2943-5100	Tel: 60-3-7651-7906	**Germany - Heilbronn**
Westborough, MA	**China - Nanjing**	**Malaysia - Penang**	Tel: 49-7131-72400
Tel: 774-760-0087	Tel: 86-25-8473-2460	Tel: 60-4-227-8870	**Germany - Karlsruhe**
Fax: 774-760-0088	**China - Qingdao**	**Philippines - Manila**	Tel: 49-721-625370
Chicago	Tel: 86-532-8502-7355	Tel: 63-2-634-9065	**Germany - Munich**
Itasca, IL	**China - Shanghai**	**Singapore**	Tel: 49-89-627-144-0
Tel: 630-285-0071	Tel: 86-21-3326-8000	Tel: 65-6334-8870	Fax: 49-89-627-144-44
Fax: 630-285-0075	**China - Shenyang**	**Taiwan - Hsin Chu**	**Germany - Rosenheim**
Dallas	Tel: 86-24-2334-2829	Tel: 886-3-577-8366	Tel: 49-8031-354-560
Addison, TX	**China - Shenzhen**	**Taiwan - Kaohsiung**	**Israel - Ra’anana**
Tel: 972-818-7423	Tel: 86-755-8864-2200	Tel: 886-7-213-7830	Tel: 972-9-744-7705
Fax: 972-818-2924	**China - Suzhou**	**Taiwan - Taipei**	**Italy - Milan**
Detroit	Tel: 86-186-6233-1526	Tel: 886-2-2508-8600	Tel: 39-0331-742611
Novi, MI	**China - Wuhan**	**Thailand - Bangkok**	Fax: 39-0331-466781
Tel: 248-848-4000	Tel: 86-27-5980-5300	Tel: 66-2-694-1351	**Italy - Padova**
Houston, TX	**China - Xian**	**Vietnam - Ho Chi Minh**	Tel: 39-049-7625286
Tel: 281-894-5983	Tel: 86-29-8833-7252	Tel: 84-28-5448-2100	**Netherlands - Drunen**
Indianapolis	**China - Xiamen**		Tel: 31-416-690399
Noblesville, IN	Tel: 86-592-2388138		Fax: 31-416-690340
Tel: 317-773-8323	**China - Zhuhai**		**Norway - Trondheim**
Fax: 317-773-5453	Tel: 86-756-3210040		Tel: 47-72884388
Tel: 317-536-2380			**Poland - Warsaw**
Los Angeles			Tel: 48-22-3325737
Mission Viejo, CA			**Romania - Bucharest**
Tel: 949-462-9523			Tel: 40-21-407-87-50
Fax: 949-462-9608			**Spain - Madrid**
Tel: 951-273-7800			Tel: 34-91-708-08-90
Raleigh, NC			Fax: 34-91-708-08-91
Tel: 919-844-7510			**Sweden - Gothenberg**
New York, NY			Tel: 46-31-704-60-40
Tel: 631-435-6000			**Sweden - Stockholm**
San Jose, CA			Tel: 46-8-5090-4654
Tel: 408-735-9110			**UK - Wokingham**
Tel: 408-436-4270			Tel: 44-118-921-5800
Canada - Toronto			Fax: 44-118-921-5820
Tel: 905-695-1980			
Fax: 905-695-2078			