AN1066

Microchip MiWi™ Wireless Networking Protocol Stack

Author: *David Flowers and Yifeng Yang*
Microchip Technology Inc.

INTRODUCTION

Implementing applications with wireless networking is now common. From consumer devices to industrial applications, there is a growing expectation that devices will have built-in the ability to communicate with each other without a hard-wired connection. The challenge is to select the right wireless networking protocol and implement it in a cost-effective manner.

The Microchip MiWi™ Wireless Networking Protocol Stack is a simple protocol designed for low data rate, short distance, low-cost networks. Fundamentally based on IEEE 802.15.4™ for Wireless Personal Area Networks (WPANs) later expanded to support Microchip proprietary RF transceivers, the MiWi protocol provides an easy-to-use alternative for wireless communication. In particular, it targets smaller applications that have relatively small network sizes, with few hops between. MiWi protocol now is one of the wireless communication protocols that are supported in MiWi™ Development Environment (DE). It uses MiMAC interface to communicate with Microchip RF transceivers and uses MiApp interface to interact with application layer.

For more information, please refer to the Microchip application note AN1283 "*Microchip Wireless Media Access Controller - MiMAC*" (DS01283) and AN1284 "*Microchip Wireless Application Programming Interface - MiApp"* (DS01284).

This application note covers the definition of the MiWi Wireless Networking Protocol Stack and how it works. For completeness, this document also introduces several aspects of wireless networking, as well as key features of IEEE 802.15.4. However, it is assumed that the user is already familiar with the C programming language and IEEE 802.15.4. You are strongly advised to read the IEEE 802.15.4 specification and MiMAC/ MiApp application notes in detail prior to using the Microchip MiWi Wireless Networking Protocol Stack.

FEATURES

The current implementation of the MiWi protocol has these features:

- Support all Microchip RF transceivers on different frequency bands.
- Portable between various Microchip MCU families.
- RTOS and application independent
- Out-of-box support for the MPLAB® C18, C30 and C32 compilers
- Easy-to-use API

CONSIDERATIONS

A network using the MiWi protocol is capable of having a maximum of 1024 nodes on a network. Each coordinator is only capable of having 127 children, with a maximum of 8 coordinators in a network. Packets can travel a maximum of 4 hops in the network and 2 hops maximum from the PAN coordinator.

If, after reading this application note, you determine that you require a standardized wireless platform, larger network sizes or common marketing logos, please refer to the application notes AN1232 "*Microchip ZigBee-2006 Residential Stack Protocol*" (DS01232) and AN1255 "*Microchip ZigBee PRO Feature Set Protocol Stack*" (DS01255). Alternatively, users may consider using the basic MiWi protocol and modifying it to suit their own applications.

For more information on the most up-to-date list of limitations of the Stack, refer to the Readme file located with the Stack download at http://www.microchip.com/miwi.

TERMINOLOGY

In describing the MiWi protocol, two specific terms are used throughout that are borrowed from the IEEE standard.

The first term is **cluster**, which refers to a grouping of nodes that form a network. A MiWi protocol cluster can be up to 3 nodes deep and is controlled by a cluster-head. In the current implementation of the MiWi protocol, the cluster-head is always the PAN coordinator. (For more information, see Table 2.)

The second term is **socket**, also referred as "Indirect Message" in MiApp interface. It refers to a virtual connection between two devices. Rather than have an exclusive hard-wired connection between devices, many devices with many types of sockets share a common communications medium and use some common method to associate applications and devices. When a new device or application is added to the network, it requires configuration to communicate

to other devices or applications. By using sockets, nodes in the network can find communication partners dynamically without having to know any information about them.

MiWi PROTOCOL OVERVIEW

The MiWi protocol is based on the MAC and PHY layers of the IEEE 802.15.4 specification, and is tailored for simple network development in the 2.4 GHz and subGHz ISM frequency bands. The protocol provides the features to find, form and join a network, as well as discovering nodes on the network and route to them. It does not cover any application-specific issues, such as how to select which network to join to, how to decided when a link is broken or how often devices should communicate.

IEEE 802.15.4 MAC

The MiWi protocol uses IEEE Standard 802.15.4 as reference to develop its MAC layer.

Similar to IEEE 802.15.4, MiWi protocol uses an Acknowledged data transfer mechanism in the MAC. This method uses a special ACK flag in the packet header. When this flag is set, Acknowledgement to the transmitter by its receiver is required; this ensures that a frame is, in fact, delivered. If the frame is transmitted with an ACK flag set and the Acknowledgement is not received within a certain time-out period, the transmitter will retry the transmission for a fixed number of times before declaring an error.

It is important to note that the reception of an Acknowledgement simply indicates that a frame was properly received by the MAC layer. However, it does not indicate that the frame was processed correctly. It is possible that the MAC layer of the receiving node received and Acknowledged a frame correctly, but due to the lack of processing resources, a frame might be discarded by upper layers. As a result, the upper layers of the application may require additional Acknowledgement response.

Device Types

IEEE 802.15.4 defines devices based on their overall functionality. There are basically two device types as shown in Table 1.

The MiWi protocol defines three types of MiWi protocol devices, based on their functions in the network: PAN Coordinator, Coordinator and End Device. The MiWi Wireless Networking Protocol Stack functionality helps to determine the type of IEEE functionality that the device requires. The MiWi protocol device types and their relationship to IEEE device types are shown in Table 2.

TABLE 1: IEEE 802.15.4™ FUNCTIONAL DEVICE TYPES

Device Type	Services Offered	Typical Power Source	Typical Receiver Idle Configuration
Full Function Device (FFD)	All or Most	Mains	On
Reduced Function Device (RFD)	Limited	Battery	Off

TABLE 2: MiWi™ PROTOCOL DEVICE TYPES

Device Type	IEEE Device Type	Typical Function
PAN Coordinator	FFD	One per network. Forms the network, allocates network addresses, holds binding table.
Coordinator	FFD	Optional. Extends the physical range of the network. Allows more nodes to join the network. May also perform monitoring and/or control functions.
End Device	FFD or RFD	Performs monitoring and/or control functions.

MiWi PROTOCOL NETWORK CONFIGURATIONS

Of the three device types defined in the MiWi protocol, the most essential type to networking is the PAN coordinator. The PAN coordinator is the device that starts the network, and selects the channel and the PAN ID of the network. All other devices joining onto the PAN have to obey the instructions of the PAN coordinator.

Star Network Configuration

A star network configuration (Figure 1) consists of one PAN coordinator node and one or more end devices. In a star network, all end devices communicate only with the PAN coordinator. If an end device needs to transfer data to another end device, it sends its data to the PAN coordinator, which in turn, forwards the data to the intended recipient.

Cluster Tree Network Configuration

In a cluster tree network (Figure 2) there is still only one PAN coordinator; however, other coordinators are allowed to join on to the network. This forms a tree-like structure, where the PAN coordinator is the root of the tree, the coordinators are the branches of the tree and the end devices are the leaves of the tree. In a cluster tree network, all of the messages sent through the network follow the path of the tree structure. Since messages may be routed through more than one node to reach their eventual destination, cluster tree networks are sometimes also referred to as multi-hop networks.

FIGURE 1: STAR NETWORK CONFIGURATION

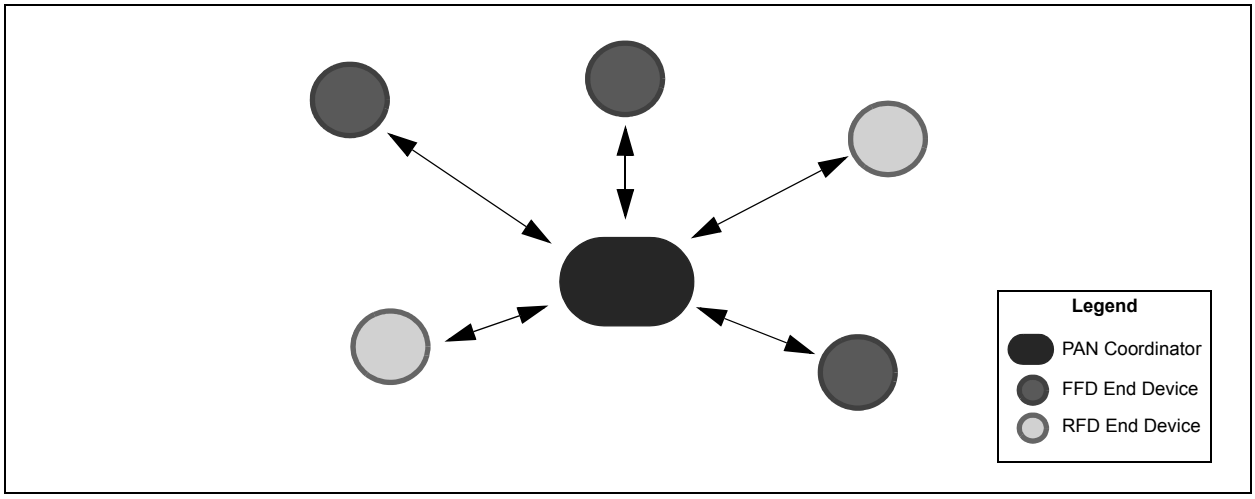
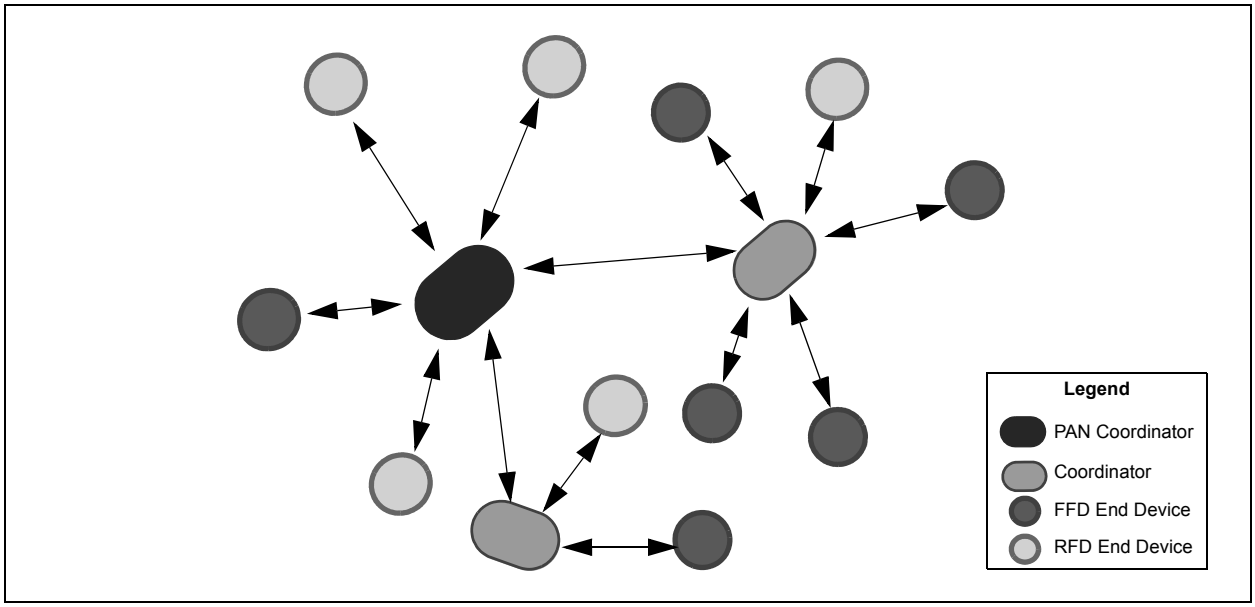


FIGURE 2: CLUSTER TREE TOPOLOGY



Mesh Network Configuration

A mesh network (Figure 3) is similar to a cluster tree configuration, except that Full Function Device (FFDs) can route messages directly to other FFDs instead of following the tree structure. Messages to Reduced Function Device (RFDs) must still go through the RFD's parent node. The advantages of this topology are that message latency can be reduced and reliability is increased. Like cluster tree networks, mesh networks are multi-hop.

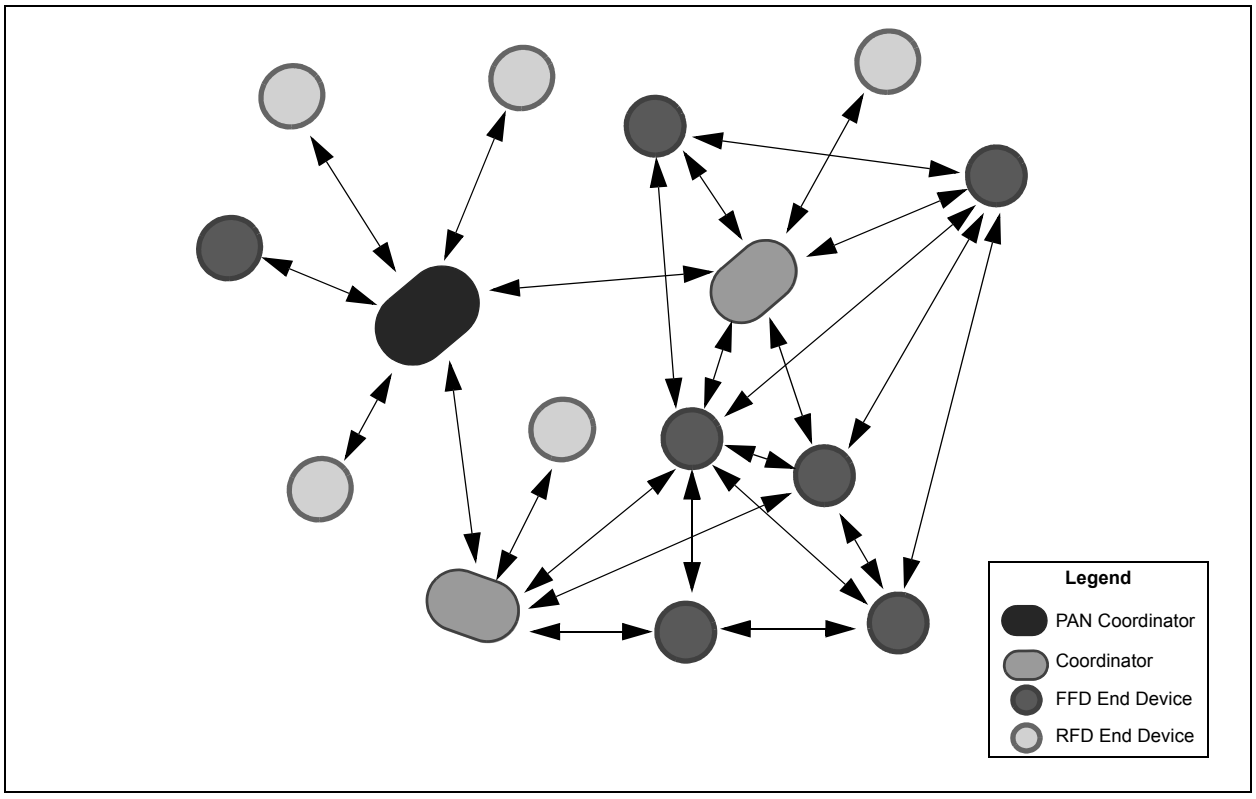
Multi-Access Networks

An IEEE 802.15.4 network is a multi-access network, meaning that all nodes in a network have equal access to the medium of communication. There are two types of multi-access mechanisms: beacon and non-beacon.

In a beacon enabled network, nodes are allowed to transmit in predefined time slots only. The PAN coordinator periodically begins with a superframe, identified as a beacon frame, and all nodes in the network are expected to synchronize to this frame. Each node is assigned a specific slot in the superframe, during which, it is allowed to transmit and receive its data. A superframe may also contain a common slot during which all nodes compete to access the channel.

In a non-beacon enabled network, all nodes in a network are allowed to transmit at any time as long as the channel is Idle. The current version of the Microchip MiWi Wireless Networking Protocol Stack supports only non-beacon networks.

FIGURE 3: MESH NETWORK



ADDRESS ASSIGNMENT

The MiWi protocol uses the addresses provided by IEEE 802.15.4. There are three different addresses defined by the specification:

1. **Extended Organizationally Unique Identifier (EUI)**: This is an 8-byte number that is globally unique. Every device shipped using the IEEE 802.15.4 specification, should have a unique EUI address. The upper 3 bytes of the EUI are purchased from IEEE (see link for the site in the **Section "References"** to buy them). The lower 5 bytes of the EUI are available for the user, as they see fit, as long as they are globally unique. For SubGHz proprietary RF transceivers, the EUI address length is in the range of two to eight bytes, defined by the application.
2. **PAN Identifier (PANID)**: The PANID is a 16-bit address that defines a group of nodes. All nodes in the PAN share a common PANID. A device assumes the PANID for a network when it selects to join that PAN.
3. **Short Address**: Also known as the device address, this is a 16-bit (2-byte) address that is assigned to a device by its parent. This short address is unique within a PAN and is used for addressing and messaging within the network. IEEE specifies that the PAN coordinator always has an address of 0000h. The address allocation is up to the PAN coordinator from that point forward.

The MiWi protocol uses the 16 available bits in the short address to help with routing and exchanging node information. The bit fields within the address are shown in Figure 4.

The Parent's Number field (bits 10-8) is unique for each coordinator on the network, including the PAN coordinator. As the Parent's Number field is only 3-bits long, this limits the number of coordinators in a network to 8.

The Child's Number field (bits 6-0) of any coordinator on the network will be 00h. This indicates that they are operating as a coordinator. Other values for this field are determined by the type of device (FFD or RFD), as well its function within the PAN. Figure 5 gives a general idea of how short addresses are determined.

The `RxOffWhenIdle` field (bit 7) is the inverse of the IEEE 802.15.4 defined property of `RxOnWhenIdle`. When this bit is set, it indicates that this device will turn off its transceiver when it is Idle and will be unable to receive packets. Any device, other than this device's parent, should route any packets that have this bit set to the device's parent. The target device's parent will buffer the message for the child until it wakes up and requests the data. If this bit is not set in the device's address, then this device is always capable of receiving packets.

Bits 15 through 11 are always '`0`' in this implementation.

FIGURE 4: BIT FIELD ARRANGEMENT FOR THE MiWi™ PROTOCOL SHORT ADDRESS

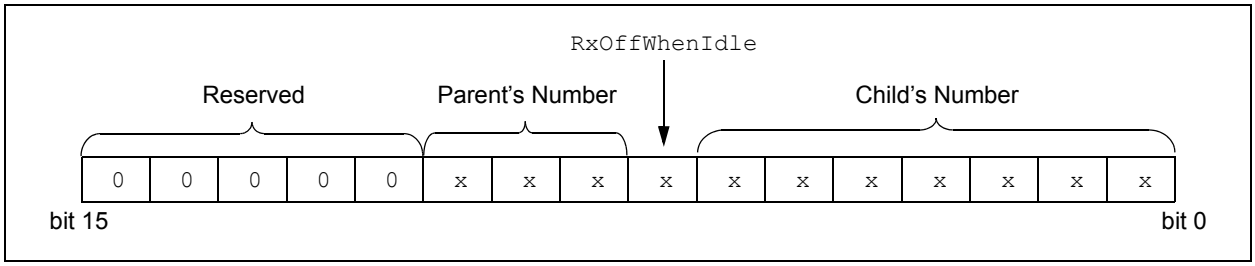
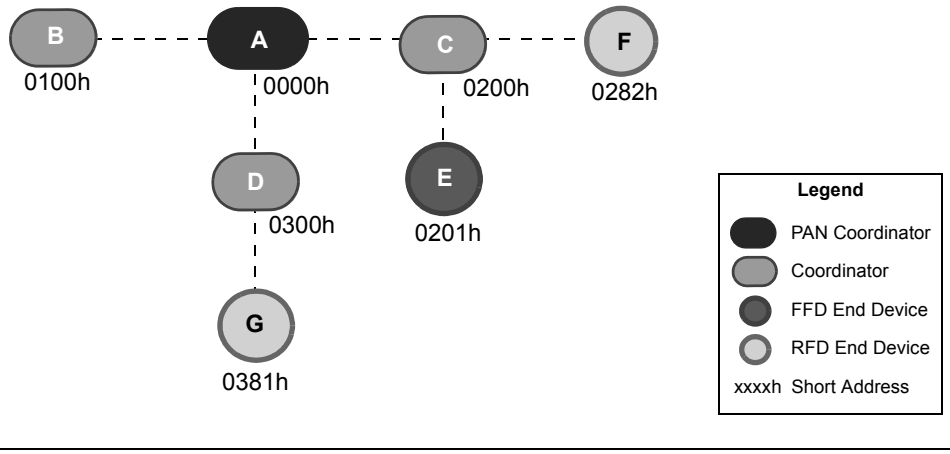


FIGURE 5: ASSIGNING SHORT ADDRESSES WITHIN A TYPICAL MiWi™ PROTOCOL NETWORK



MiWi PROTOCOL MESSAGING

Once a network has been formed, the next major concern is how to send messages through the network. Any device that is a member of a MiWi protocol network will use its short address to communicate through the network. This short address helps other devices in the network to determine the location of the node and how to route to that device.

Packet Format

For a IEEE 802.15.4 compliant RF transceiver, MiWi protocol uses 16-bit short address to transmit and receive messages whenever possible. The packets should be constructed according to Section 7.2 of the IEEE 802.15.4 specification. Proprietary SubGHz RF transceiver, however, use EUI address in MAC layer. For more information on the packet format in MAC layer for Microchip proprietary RF transceiver, refer to the Microchip application note AN1283 "*Microchip Wireless Media Access Controller - MiMAC*" (DS01283).

Above this layer resides the MiWi protocol header that contains information needed for routing and packet processing. This header format is shown in Figure 6. It is comprised of the following components:

- **Hops**: The number of hops that the packet is allowed to be retransmitted (00h means don't retransmit this packet – 1 byte).
- **Frame Control**: The Frame Control field is a bitmap that defines the behavior of this packet. The individual bits are defined in Table 3 (1 byte).
- **Dest PANID**: The PANID of the final destination node (2 bytes in the MiWi protocol).
- **Dest Short Address**: The final destination's short address (2 bytes).
- **Source PANID**: The PANID of the node that originally sent the packet (2 bytes).
- **Source Short Address**: The short address of the node that originally sent the packet (2 bytes).
- **Sequence Number**: A sequence number that can be used to track the status of packets as they travel through the network (1 byte).

FIGURE 6: MiWi™ PROTOCOL PACKET HEADER FORMAT

Hops	Frame Control	Dest PANID	Dest Short Address	Source PANID	Source Short Address	Sequence Number	Report Type	Report ID
1	1	2	2	2	2	1	1	1

Legend: Numbers indicate packet component size in bytes.

TABLE 3: FRAME CONTROL BIT FIELD

0	0	0	0	0	x	1	0
r	r	r	r	r	ACKREQ	INTRCLST	ENCRYPT
bit 7							bit 0

bit 7-3 **Reserved:** Maintain as '0' in this implementation

bit 2 **ACKREQ:** Acknowledge Request bit
When set, the source device requests an upper layer Acknowledgement of receipt from the destination device.

bit 1 **INTRCLST:** Intra Cluster bit
Reserved in this implementation, maintain as '1'.

bit 0 **ENCRYPT:** Encrypt bit
When set, data packet is encrypted at the application level.

Note: Abbreviated bit names are for convenience of display only; they are not an official part of IEEE 802.15.4™.

Routing

Routing in wireless networks can be a very difficult and resource intensive task. The MiWi protocol solves this problem by using the address allocation to indicate the parent of the device you want to send the packet to, and by using the already provided IEEE services to help exchange and relay routing information in the network.

LEARNING ABOUT NEIGHBORING COORDINATORS

One of the tasks of a routing algorithm is determining the next hop for any outgoing packet. The MiWi protocol uses the IEEE network join mechanism, in addition to regular network traffic, to discover these paths. When any device is joining onto the network, it first sends out a beacon request packet. All of the coordinators that hear the beacon request packet sends out a beacon packet informing neighboring devices of their network information.

In the MiWi protocol, three bytes of additional information are attached to the beacon payload to assist with routing:

- **Protocol ID (1 byte)**: This helps distinguish MiWi protocol networks from other IEEE 802.15.4 networks that may be operating in the same radio range. Protocol ID should always be 4Dh.
- **Version Number (1 byte)**: The version number of the specification.
- **Local Coordinators (1 byte):** This field is a bitmap that indicates which coordinators are currently visible by the coordinator that is sending the beacon. Each bit position directly represents one of 8 possible coordinators. Bit 0 is 0000h (the PAN coordinator). Bit 1 indicates that this coordinator can talk directly to 0100h, and so on.

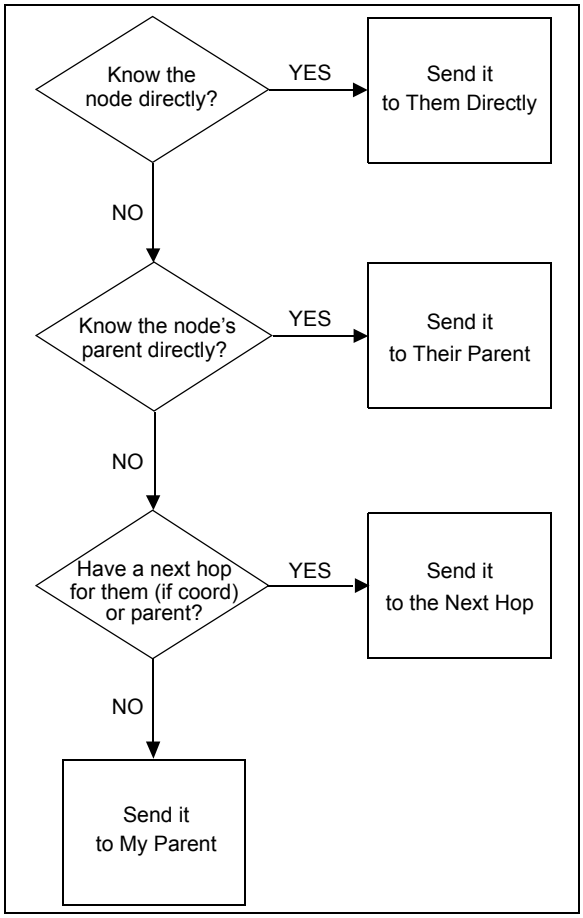
For example: Coordinator 0x200 is capable of talking to 0x500 and the PAN coordinator. The Local Coordinators field would be 'b00100101.

Through the Local Coordinators field of the beacon payload, all of the coordinators on the network will learn about various possible routes to all of those nodes without having to send out unique requests.

ROUTING TO OTHER DEVICES

Routing in MiWi protocol networks becomes easy once we have knowledge of the neighboring coordinators, as well as what those coordinators can see. Sending a packet to another node follows the logic, as shown in Figure 7.

FIGURE 7: DECISION TREE FOR PACKET FORWARDING



Broadcast Messages

When a MiWi protocol network coordinator receives a broadcast packet, it will rebroadcast the packet as long as the Hops counter (the first byte of the header) is not equal to zero.

Broadcast packets should always have their ACK request bits (in both the MiWi protocol header and the MAC header) set to '0'. Coordinators that receive broadcast packets should then process the packet after retransmitting it.

MiWi Protocol Reports

The MiWi protocol transports packets between devices by using special packets called reports. The protocol allows for the implementation of up to 256 Report Types, and up to 256 separate Report IDs for each Report Type. The Report ID is the specification function of the packet.

Report Type 00h is reserved for MiWi Wireless Networking Protocol Stack packets, which have packet payload that is directed to the Stack. For example, a MiWi protocol ACK has a Report Type of 00h (because it is a Stack packet) and a Report ID of 30h. All other Report Types are available for the user.

The Report Type and Report ID are defined in the packet header, as previously described in the **Section "Packet Format"**. The size and contents of the payload of a report depends on the particular Report ID. In this implementation of the MiWi protocol, the payload size varies from 0 bytes (i.e., sending a packet with a specific Report Type/ID is essentially the entire message) to 10 bytes, with multi-byte payloads being delivered Least Significant Byte (LSB) first. A list of the implemented reports in the current version of the protocol is provided in Table 4, with detailed descriptions immediately following.

TABLE 4: REPORTS IMPLEMENTED IN THE MiWi™ PROTOCOL

Report Type	Report ID	Name
00h	10h	OPEN_CLUSTER_SOCKET_REQUEST
	11h	OPEN_CLUSTER_SOCKET_RESPONSE
	12h	OPEN_P2P_SOCKET_REQUEST
	13h	OPEN_P2P_SOCKET_RESPONSE
	20h	EUI_ADDRESS_SEARCH_REQUEST
	21h	EUI_ADDRESS_SEARCH_RESPONSE
	30h	ACK_REPORT_TYPE
	40h	CHANNEL_HOPPING_REQUEST
	41h	RESYNCHRONIZATION_REQUEST
	42h	RESYNCHRONIZATION_RESPONSE
01h-FFh	00h-FFh	Available for use

OPEN_CLUSTER_SOCKET_REQUEST

Report Type (1 byte)	Report ID (1 byte)	Requesting EUI Address (8 bytes)
00h	10h	The EUI of the initiating device.

The destination address of the MiWi protocol header should be the PAN coordinator (0000h). The source address of the MiWi protocol header should be the address of the device that is initiating the request. The Requesting EUI Address field specifies the EUI of the device initiating the request, LSB first.

OPEN_CLUSTER_SOCKET_RESPONSE

Report Type (1 byte)	Report ID (1 byte)	Resulting EUI Address (8 bytes)	Resulting Short Address (2 bytes)
00h	11h	The EUI of the resulting device.	The short address of the resulting device.

The destination address of the MiWi protocol header should be the original requesting device. The source address should be the PAN coordinator (as this packet will only originate from the PAN coordinator). The Resulting EUI Address field specifies the EUI of the device that responded to the request (LSB first). Note that this is a different address than the MiWi protocol destination address.

The Resulting Short Address field is the short address of the device that responded to the request. With the combination of EUI and short address sent to both requesting nodes, they should be able to communicate on the network and find each other if either of them happens to move in the network. Once the OPEN_CLUSTER_SOCKET_RESPONSE is sent out, the PAN coordinator will no longer maintain any of the socket information.

OPEN_P2P_SOCKET_REQUEST

Report Type (1 byte)	Report ID (1 byte)
00h	12h

Both the source and destination information in the MiWi protocol header should be FFFFh. The Hops field of the MiWi protocol header should be 00h in order to prevent rebroadcast of the packet. The MAC level source and destination PANID should be FFFFh. The MAC destination short address should be FFFFh. The MAC level source address should be Long Address mode.

OPEN_P2P_SOCKET_RESPONSE

Report Type (1 byte)	Report ID (1 byte)
00h	13h

Both the source and destination information in the MiWi protocol header should be FFFFh. The Hops field of the MiWi protocol header should be 00h. The MAC level source and destination PANID should be FFFFh. The MAC destination and source addresses should be both long addresses (EUIs).

EUI_ADDRESS_SEARCH_REQUEST

Report Type (1 byte)	Report ID (1 byte)	Search EUI Address (8 bytes)
00h	20h	The EUI of the device that is being searched for.

The destination short address and PANID of the MiWi protocol header should be the broadcast address, FFFFh. The source address and PANID of the MiWi protocol header should be the address information that is requesting the search. On reception of this packet, a coordinator in the network will rebroadcast this packet if the number of hops is more than 00h. The coordinator will decrement the Hops counter before rebroadcasting the packet. The coordinator will not change the value of the MiWi protocol sequence number when broadcasting the packet.

EUI_ADDRESS_SEARCH_RESPONSE

Report Type (1 byte)	Report ID (1 byte)	Search EUI Address (8 bytes)	Search Results PANID (2 bytes)	Search Results Short Address (2 bytes)
00h	21h	The EUI of the device that is being searched for.	The resulting device's PANID.	The resulting device's short address.

The EUI_ADDRESS_SEARCH_RESPONSE should be unicast back to the address of the device that originally sent the request (the device mentioned in the MiWi protocol source fields of the request packet).

ACK_REPORT_TYPE

Report Type (1 byte)	Report ID (1 byte)
00h	30h

The MiWi protocol source address of the MiWi protocol ACK packet should equal the MiWi protocol destination address of the packet that requires Acknowledgement. The MiWi protocol destination address of the MiWi protocol ACK packet should equal the MiWi protocol source address of the packet that requires Acknowledgement.

CHANNEL_HOPPING_REQUEST

Report Type (1 byte)	Report ID (1byte)	Current Channel (1 byte)	Channel to hop (1 byte)
00h	40h	Current operating channel	Channel to hop to

CHANNEL_HOPPING_REQUEST is the command that PAN Coordinator broadcast to every node on the network to move to a different channel. This is the start of frequency agility process.

RESYNCHRONIZATION_REQUEST

Report Type (1 byte)	Report ID (1 byte)	Current Channel (1 byte)
00h	41h	Current operating channel

When a sleeping device wakes up but cannot communicate with its parent continuously, it may be due to the fact that its parent has hopped to a different channel due to frequency agility capability. RESYNCHRONIZATION_REQUEST is the request sent from the sleeping device and request recommunicate with its parent within all possible channels.

RESYNCHRONIZATION_RESPONSE

Report Type (1 byte)	Report ID (1 byte)
00h	42h

When a sleeping device wakes up but cannot communicate with its parent continuously, it may be due to the fact that its parent has hopped to a different channel due to frequency agility capability. RESYNCHRONIZATION_RESPONSE is the response to the RESYNCHRONIZATION_REQUEST command from the parent to its child.

STACK MESSAGES AND SERVICES

Providing address allocation and routing services are just the beginning of a wireless network solution. In addition to these basic features, the MiWi protocol provides other optional services that assist developers to more rapidly reach a solution. These services include dynamically creating connections between two devices without having to know any information about the device ahead of time (i.e., sockets), and the ability to search the network for a specific device's long address.

Discovering Nodes in the Network by EUI

When two nodes communicate over a MiWi protocol network, they use their short addresses. If the network topology ever changes, it is useful to be able to find that device again. Because the short address of a device is assigned to it by its parent, the short address of the destination device may have changed. If this is the case, then the new short address must be discovered before communication can be re-established. Unlike the short address, the EUI of a device never changes and is globally unique. If one device knows another device's EUI, it will be able to distinguish that device from any other device. Searching the network for a specific EUI then becomes important in reestablishing communication with nodes that have moved. The MiWi protocol provides such a feature to search the network for a specific EUI.

There are two Stack packets that are defined in order to assist with searching for a specific EUI on the network: EUI_ADDRESS_SEARCH_REQUEST and EUI_ADDRESS_SEARCH_RESPONSE. The Search Request is unicast to the first coordinator (Figure 8, sequence 1) and then broadcast among the coordinators into the network with the EUI of the device that needs to be located (sequence 2). It is repeated by all of the coordinators until the Hops counter dies (sequence 3).

If one of the coordinators currently has this device as a child, it returns a EUI_ADDRESS_SEARCH_RESPONSE packet with the device's EUI and its short address. The EUI_ADDRESS_SEARCH_RESPONSE is sent unicast, node by node, back to the MiWi protocol source address of the packet that originally sent the packet (sequence 4).

Opening a Socket to a Device

Another feature that may be important to some networks is the ability to dynamically form communication links on the network. This is useful in networks where preconfiguration of nodes needs to be minimal.

As an example, a user may wish to add a new light switch to a lighting control system. How does that light switch know which light to send its packets to? One way would be to have some type of interface where the device installer manually programs controller and target addresses into the devices.

CLUSTER SOCKET

A more dynamic method would be to have a push button on both the light and the light switch. You first press the button on the light, and then the light switch, within a specified time interval. This lets these two devices know that they need to communicate with each other. This allows for dynamic run-time changes in the behavior of the network in a simple, user-friendly method.

The MiWi protocol defines this dynamically formed communication link as a cluster socket. A cluster socket exists between two nodes that are members of the network and is formed based on the short address. Cluster Socket is also known as "Indirect Connection" in MiApp interface. For more information, refer to the Microchip application Note AN1284 "*Microchip Wireless Application Programming Interface – MiApp*" (DS01284).

In the previous example, pushing the button on the light switch sends an OPEN_CLUSTER_SOCKET_REQUEST to the PAN coordinator with that device's information (Figure 9, sequence 1). This notifies the PAN coordinator that the device is looking for someone to communicate to. The PAN coordinator keeps that request open for an amount of time that is specified by the application. If a similar request comes from the light (sequence 2), the PAN coordinator combines the short address information from both devices into one OPEN_CLUSTER_SOCKET_RESPONSE and sends it back to the switch and the light (sequence 3). Finally, the PAN coordinator removes the open socket request. If the PAN coordinator doesn't hear a second open socket request within the specified amount of time, then it will terminate the open socket request without sending a response to the requesting node.

When the devices at F and G receive the OPEN_CLUSTER_SOCKET_RESPONSE report, they can examine the payload of that packet and determine if that device is, in fact, the device that they wish to talk to. This is an application layer decision; it is not provided by the Stack.

FIGURE 8: SEQUENCE FOR EUI ADDRESS SEARCH REQUEST AND RESPONSE

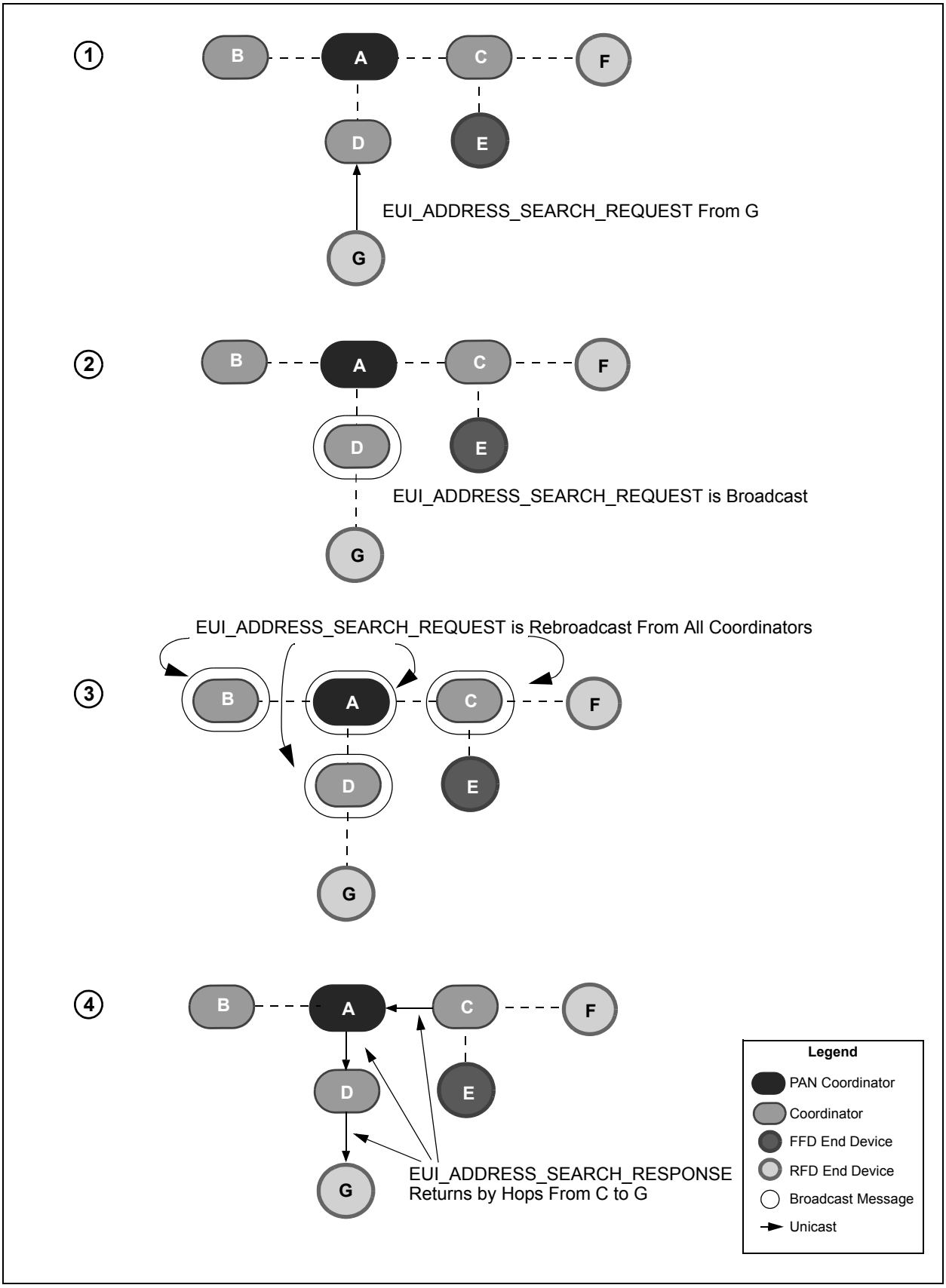
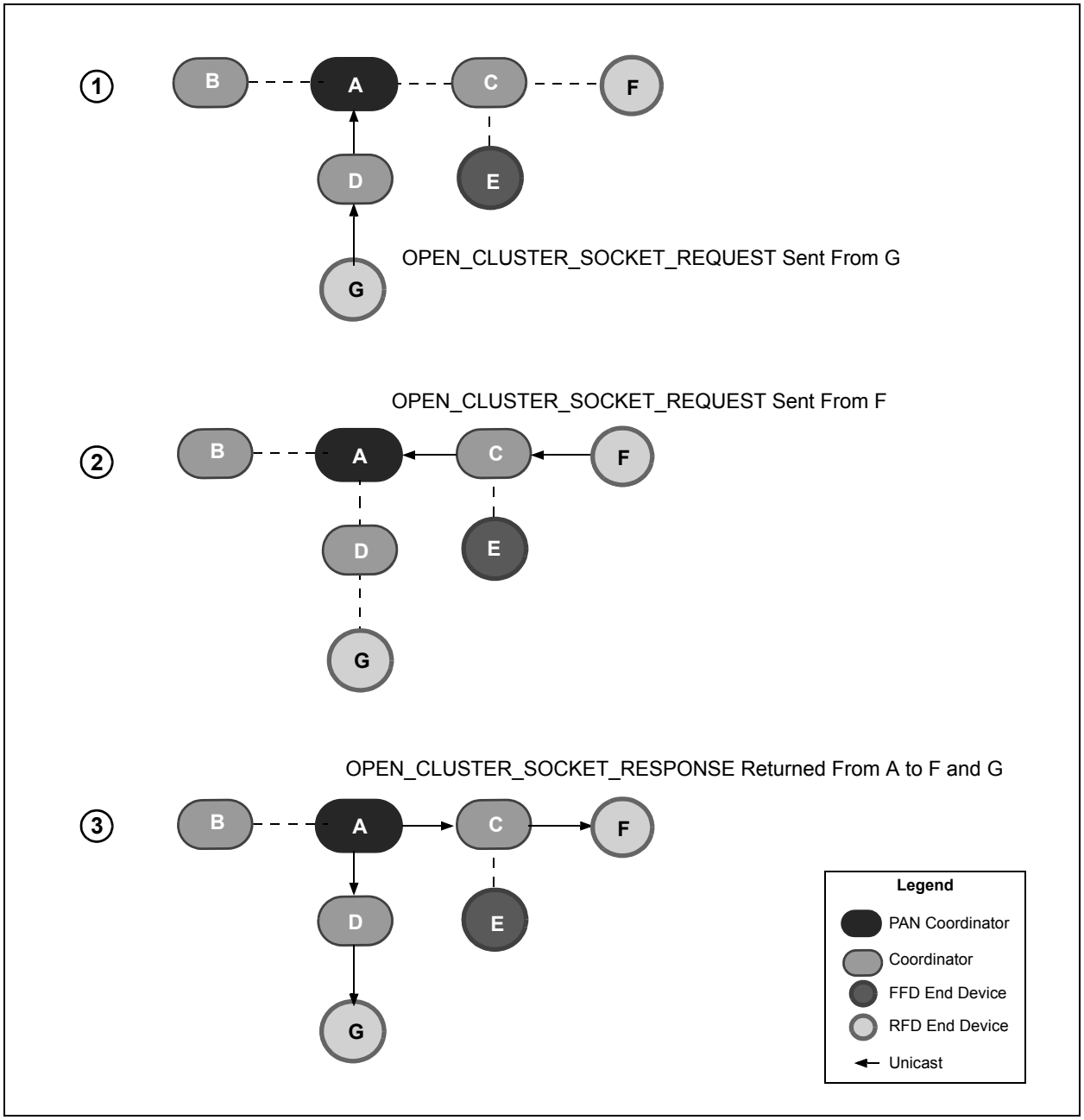


FIGURE 9: SEQUENCE FOR OPEN SOCKET REQUEST AND RESPONSE



PROGRAMMING INTERFACES

The programming interfaces from MiWi protocol to RF transceivers have been documented in detail in Microchip application note AN1283 “Microchip Wireless Media Access Controller - MiMAC”. The application programming interface from application layer to MiWi protocol has been documented in detail in Microchip application note AN1284 *“Microchip Wireless Application Programming Interface - MiApp”* (DS01284). Please refer to above two application notes for configurations as well as prototype of function calls.

USER CONSIDERATIONS

There are several different network situations and circumstances that are not inherently covered by the Stack. Each of these situations should be considered. Some of the situations must be implemented. Others can be implemented if the system requires it.

Which Network to Join?

The network discovery feature built into the MiWi protocol searches the available channels for networks. It does not, however, choose which network to join. This is left as an application decision. Some applications will want to pick one network over another, or a certain coordinator over another coordinator, within the same network. After the network discovery is complete, each device will then need to search through the list and determine to which coordinator it will join.

Failure Recovery

Failure recovery is an interesting issue in wireless communications. Some developers require their networks to dynamically heal when parts of the network fail. Other developers need the network to stay as unchanged as possible, even when failures do occur. Because of this variation in requirements, the MiWi protocol Stack doesn't default to either implementation.

The MiWi protocol provides ACKs on both the MAC and MiWi protocol layers. This allows users to know that their packet reached the destination correctly or to determine that the packet did not make it to the destination successfully. These ACKs can be used to determine when there is a network failure.

Determining when to communicate to a node is also not a feature of this implementation. Determining a node is no longer available could be a failed packet ratio, a number of consecutive packets failed, a low RSSI, a low data throughput, and so on. The current Stack does not implement any of these requirements. If an application has such requirements, then it should be added in at the application level (or change the Stack functionality, if required).

One possible mode of failure is if the parent of a device fails. In some networks, it is preferred that the orphaned device find a new parent, while in other networks, the device is left off of the network until the parent returns online. Some implementations may prefer to rejoin the same network in a different location, while others may prefer to search for a new network.

Another problem that is not covered in the Stack is how to promote a node from a coordinator (or end device) to a PAN coordinator if the PAN coordinator fails. Because the Stack does not determine when a device is offline, it also cannot initiate this process. Promoting a coordinator to a PAN coordinator can create issues in that the coordinator's old children must all be dropped off of the network. Remember that the PAN coordinator always has the address, 0000h. This means that the coordinator that is taking over the role must change its address, and thus, all of it previous children's addresses much change as well.

Which coordinator should take control of the network is also a decision that is better suited for the application to decide. Many networks can exist just fine without the PAN coordinator present. Both routing and end device joining works without the PAN coordinator. However, no new coordinators will be allowed to join the network while the PAN coordinator is offline.

Exchanging EUIs to Protect Against Node Migration

Nodes in a wireless network may change their parent for various reasons, including failure and mobility. In this situation, the device's short address will change. Nodes that were communicating with the node that moved will no longer be able to communicate with the node. It is because of this reason that devices that care about maintaining connectivity, despite node mobility, may find it useful to request a node's EUI after establishing communications with that device.

Accepting an Open Socket Response

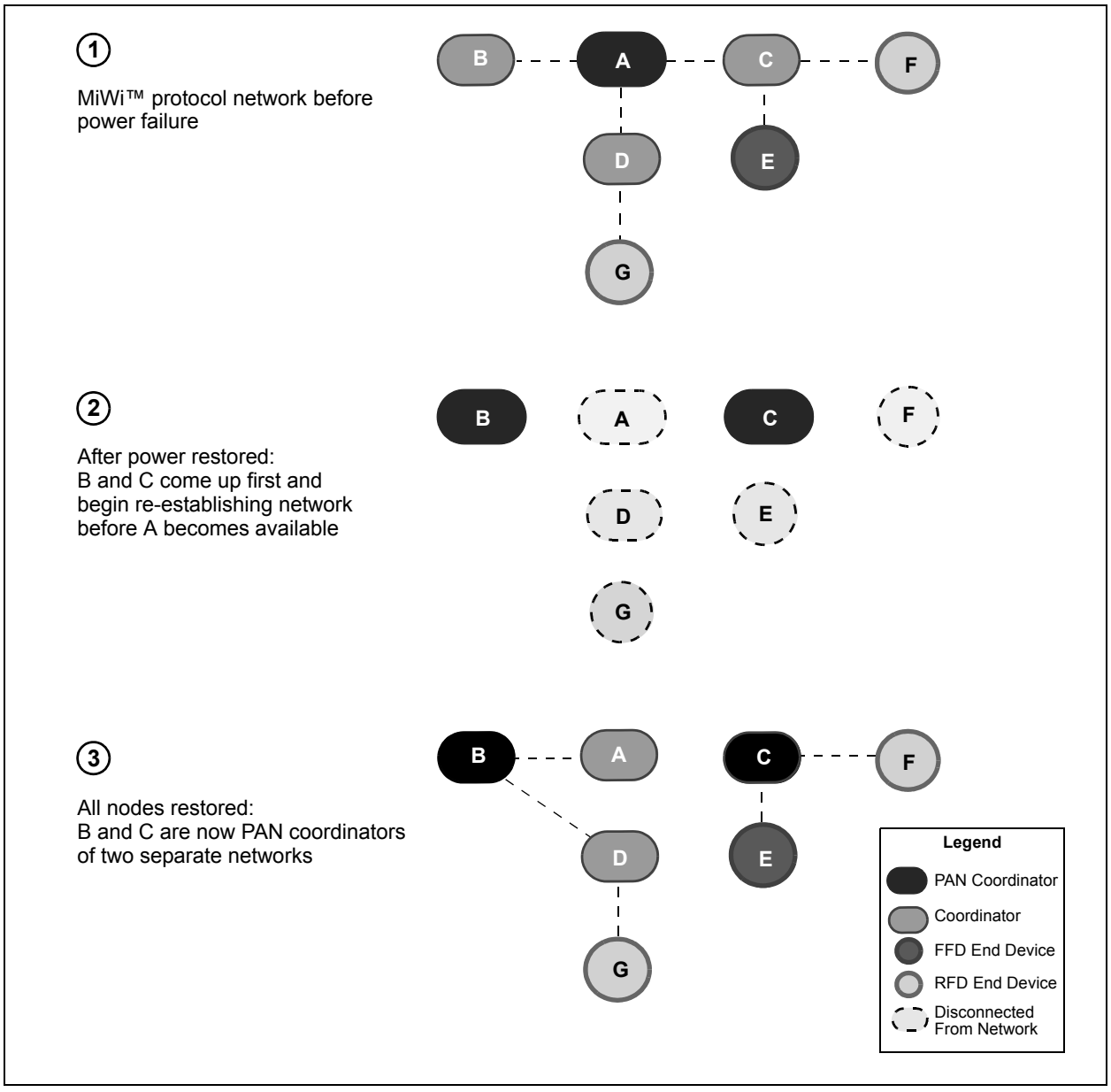
The Stack provides a means of forming dynamic bonds between two devices through the request to establish indirect connection (also called socket in MiWi protocol term). When the request returns a result, this merely indicates that another device was looking for a communication partner in the same time frame. This does not imply that the devices are meant to communicate to one another. Deciding if the returned node is acceptable or not is an application level decision and may require implementation if required.

Network Islands

The coordinators and PAN coordinator in the MiWi protocol don't have much difference in terms of functionality. Because of this, it is possible to create a coordinator that will take on the role of a PAN coordinator if it can't find a suitable network. This poses an interesting problem if the entire network is power cycled. For example, a network with a PAN coordinator, two coordinators located outside of each others' radio range and various end devices. If the entire network power cycles, and the two coordinators come online first, they will both search for a PAN coordinator. Neither will see a PAN coordinator and each of them will form a network of their own, possibly on different channels. Some time later, the original PAN coordinator will power on and attempt to find a network. It will find the network formed by each of the PAN coordinators and join one of the networks. This forms two distinct networks that can no longer communicate, instead of one large network.

As an example, consider the network in Figure 10. Sequence 1 illustrates how the network operates when it is functioning correctly. After a power failure, coordinators B and C come back online first (sequence 2). They are unable to see each other and unable to find node A so they both start their own network. After some time, the other nodes in the network join. When node A comes back online, it will see two networks it can join to and pick one. The two networks will remain disjointed unless this is either prevented or corrected.

FIGURE 10: FORMATION OF MiWi™ PROTOCOL NETWORK ISLANDS



To solve this problem, the best solution is to enable the network freeze feature, which is defined in MiApp interface. Network freeze feature stores critical network information into the Non Volatile Memory (NVM). When power cycle happens, it can read the critical network information from the NVM and recovers to the state before the power cycle.

Security

There may be instances where it is necessary to transmit the key. For example: in some applications, devices may join the network without a preconfigured key and get assigned a key once on the network. Another example: some networks update their key periodically.

Users are encouraged to implement some form of key handling procedure in the application layer with a separate Report Type and Report ID. One method of doing it is to transfer the security key and key sequence number every time a node joins the network, and store the keys and key sequence number in RAM. This approach only meets the minimum requirement of a secured network, since the security key is transmitted every time a node joins the network.

An alternative approach is to set a default key for every device and use the default key to secure the transferred key. This approach provides minimum protection for the keys. A third safe approach is to add a new function to store data to the NVM during run time. This way, the security key and key sequence number only need to be transferred once when the device initially joins the network. Once obtained, the security information is retained in program memory, and does not have to be retransmitted if the device leaves and rejoins the network.

These key handling methods are suggestions only. If this functionality is required by an application, it is left to the user to implement a suitable method.

RESOURCE REQUIREMENTS

The size of the final application is determined by its device type, as defined by both IEEE 802.15.4 and the MiWi protocol, its configuration and the use of security features. The footprint of the stack varies from release to release. A Coordinator capable device may be fit into 32KB flash easily, while an end device can be fit into 16KB flash without any problem.

Note:	The memory requirements provided here reflect those of the initial release of the MiWi Wireless Networking Protocol Stack at the time of the initial publication of this document. Subsequent code revisions may increase or decrease these requirements.

CONCLUSION

For developers looking for an entry point into wireless networking, the MiWi protocol provides a low-cost platform to ease the development of short-range, wireless, networked applications. MiWi protocol provides features that help implementing a simple wireless network easier. It is not intended to address all of the scenarios and decisions required in creating a wireless solution.

The MiWi protocol is the entry point for Microchip's wireless solution based on IEEE 802.15.4. Users needing a more complex networking solution may want to consider the Microchip implementation of the ZigBee protocol or expanding the MiWi protocol to suit their needs.

REFERENCES

IEEE Std 802.15.4-2003, *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)*. New York: IEEE, 2006.

"ZENA™ Wireless Network Analyzer User's Guide" (DS51506) http://www.microchip.com

Microchip Application Note AN1283, "*Microchip Wireless Media Access Controller – MiMAC"* http://www.microchip.com/miwi

Microchip Application Note AN1284, "*Microchip Wireless Application Programming Interface – MiApp"* http://www.microchip.com/miwi

APPENDIX A: SOURCE CODE FOR MiWi™ WIRELESS NETWORKING PROTOCOL STACK

Because of its size, a complete source code listing of the MiWi Wireless Networking Protocol Stack is not provided here. The complete source code, including the ZENA analyzer demo software and MiWi protocol demo applications, is available for download from the Microchip corporate web site at: **http://www.microchip.com/miwi**.

REVISION HISTORY

Revision A

This is the initial release of the document.

Revision B (July 2010)

This revision incorporates the following updates:

- Updated the technical changes in the following sections:
  - Introduction
  - Features
  - Terminology
  - User Consideration
  - MiWi protocol overview
  - MiWi Protocol messaging
  - Address assignment protocol
  - Stack messages and services
  - User considerations
  - Resource requirements
  - Conclusion
  - References
- Added the following sections:
  - **“CHANNEL_HOPPING_REQUEST”**
  - **“RESYNCHRONIZATION_REQUEST”**
  - **“RESYNCHRONIZATION_RESPONSE”**
  - **“Programming Interfaces”**
- Tables
  - Updated **TABLE 4: “Reports Implemented in the MiWi™ Protocol”**
- Removed the following sections:
  - MiWi protocol security
  - Setting up a MiWi protocol project
  - Using the Zena™ analyser as a configuration tool
  - Using the stack
  - Data structures
  - MiWi protocol stack operations
  - MiWi Protocol stack functions
- Additional minor corrections such as language and formatting updates are incorporated throughout the document.

NOTES:

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

ISBN: 978-1-60932-346-2

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://support.microchip.com
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-6578-300
Fax: 886-3-6578-370

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/05/10