

Microchip Wireless (MiWi™) Media Access Controller – MiMAC

Author: Yifeng Yang
Microchip Technology Inc.

INTRODUCTION

The primary function of wireless communication protocol is to transmit and/or receive information between two nodes. The Media Access Controller (MAC) layer provides the basic channel access, addressing and data transmission/receiving functionalities, on top of the Physical (PHY) layer that handles raw data. In the standard Open Systems Interconnection (OSI) model, it serves as the Data Link Layer (DLL). Because of the wide variety of possible implementations in the PHY layer, the MAC is the lowest possible layer to standardize in the software for communication protocols.

This application note defines the Microchip MAC layer, MiMAC, for communication protocols and transceivers supported by Microchip for short-range, low data rate and low-power wireless applications.

Implementing MiMAC benefits wireless application developers in multiple ways:

- Traditionally, wireless communication protocol stacks are complicated to implement and difficult to use. With the new definition of MiMAC, it is possible to make the protocol stack available for widely different RF transceivers.

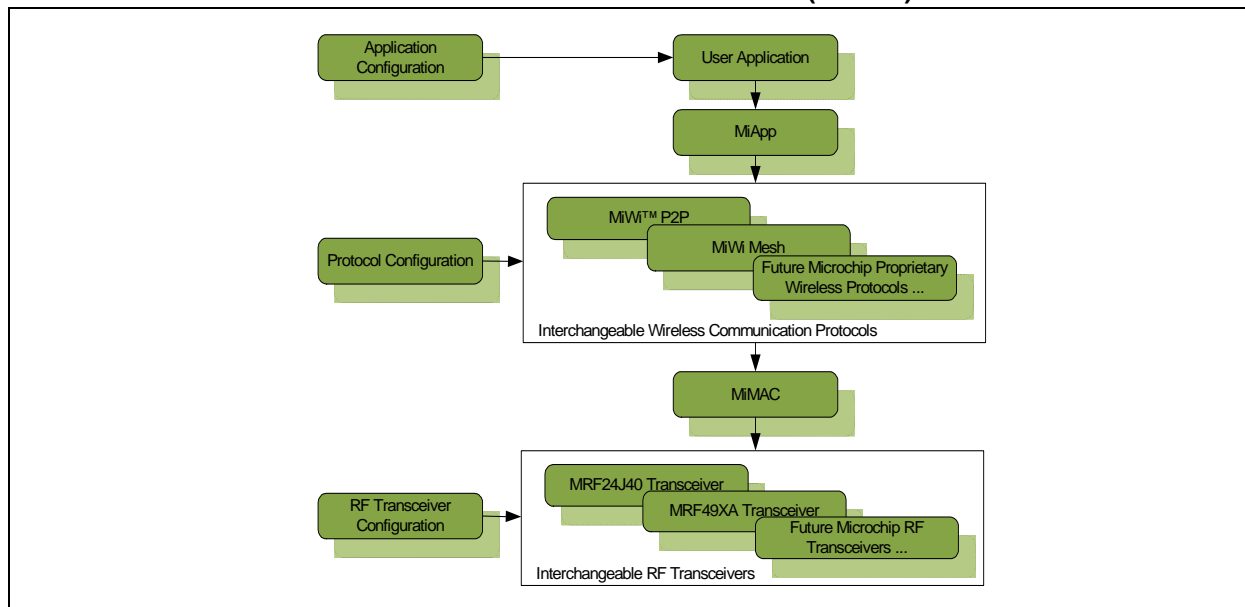
- The learning curve for MiMAC can be flattened and applied to all Microchip transceivers across different frequency bands and modulations. It significantly reduces the development risk for wireless application developers by providing end users the capability of changing different transceivers at any stage of software development. Choosing a transceiver in the firmware is a process that is transparent to the customer by modifying a configuration parameter.

MiMAC FEATURES

The MiMAC implements the following features:

- Easy to learn, implement and support.
- Flexible enough to be implemented on microcontrollers (MCUs) and RF transceivers from Microchip.
- Powerful enough to address most short-range, low data rate applications.
- Simple, but strong, security module with its Security modes for transceivers that do not have a hardware security engine.
- Concise, but powerful, programming interface between MiMAC and all Microchip proprietary wireless communication protocols.
- Minimum impact to the firmware footprint.

FIGURE 1: BLOCK DIAGRAM OF MICROCHIP WIRELESS (MiWi™) SOLUTIONS



Microchip Application Programming Interface (MiApp)

In addition to standardizing in the MiMAC layer, Microchip also aims to standardize the interfaces in the application layer. The standard interface in the application layer is called, Microchip Wireless Application Programming Interface (API) or MiApp. The definition of MiApp enables all Microchip proprietary wireless protocols to be interchangeable, with little or no change in the software application code. For details on MiApp, refer to AN1284, “Microchip Wireless (MiWi™) Application Programming Interface (MiApp)”.

MiMAC standardizes the interfaces between Microchip wireless protocols and Microchip RF transceivers. MiMAC makes all Microchip RF transceivers interchangeable with little or no change in the software application code.

Both MiMAC and MiApp enable wireless application developers the maximum flexibility to choose the RF transceivers and wireless communication protocols at any stage of software development, thus reducing development risk to the minimum.

Microchip Wireless Configurations

There are three layers of configurations for application protocol stacks and RF transceivers:

- “Application Configurations” might change between devices in the same application according to their hardware design, role in the application and/or network. Wireless application developers tend to do the majority of the configurations in the application layer.
- “Protocol Stack Configurations” fine tune the behavior of the protocol stack. The majority of the configurations in the stack level is to set the timing of the stack, specify the routing mechanism, etc.
- “Transceiver Configurations” define the frequency band, data rate and other RF related features of the RF transceiver.

The default settings for both protocol stack and transceiver configurations might work fine with the application without any modification. The application configurations, however, tend to be changed to fit the needs of different wireless applications.

Figure 1 demonstrates the Microchip Wireless (MiWi™) solutions.

MiMAC OVERVIEW

The MiMAC layer consists of three separate, but closely related, major parts. Among the three major parts, the first and second are defined for Microchip proprietary RF transceivers that have limited hardware support in the MAC layer. The third is defined for all Microchip RF transceivers. The three parts are:

1. MiMAC Frame Format

The frame format defines how the packet appears over the air. Basically, the MiMAC frame format decides the capability and efficiency of the MiMAC specification. It serves as the foundation for the other two parts in MiMAC architecture.

2. MiMAC Security Module

For all wireless communication, the message is transmitted through the open air. It is relatively easier to intercept information from wireless communication than from wired communication. Therefore, security may be a serious consideration for many applications. The MiMAC security module defines a low-cost block cipher with strong security strength. The MiMAC security module also defines multiple Security modes to work with the block cipher for different requirements from the applications.

3. MiMAC Universal Programming Interface

The MiMAC universal programming interface serves as a driver between all Microchip RF transceivers and Microchip proprietary wireless communication protocols. The programming interface enables the Microchip RF transceivers to work under any Microchip proprietary wireless protocol; they also enable all Microchip proprietary wireless communication protocols to use Microchip RF transceivers.

The transceivers supported by Microchip differ widely in features. Some transceivers have a well-defined hardware MAC layer, including frame format and/or security engine. There may be hardware features that are built into the transceivers to comply with the specification. Microchip MRF24J40 is a good example of such transceivers; it complies with the IEEE 802.15.4™ specification. MiMAC does not intend to regulate the frame format and/or security engine if they are already implemented in the transceiver hardware, as prior experiences demonstrate that the hardware feature is often faster and consumes less system resources.

For those transceivers that have built-in hardware support in the frame format and/or security engine, it is recommended to use the hardware implementation on the transceiver and the MiMAC programming interface.

For other proprietary RF transceivers, there is very limited, or virtually no, MAC layer defined in the hardware. For these types of transceivers, all three major parts of the MiMAC specification are recommended. With a powerful MiMAC definition in the software, Microchip enables those simple RF transceivers virtually the same communication or networking capability in the software as their siblings, with much more complexity in silicon.

Subsequent sections describe each of the three major parts in the MiMAC specification.

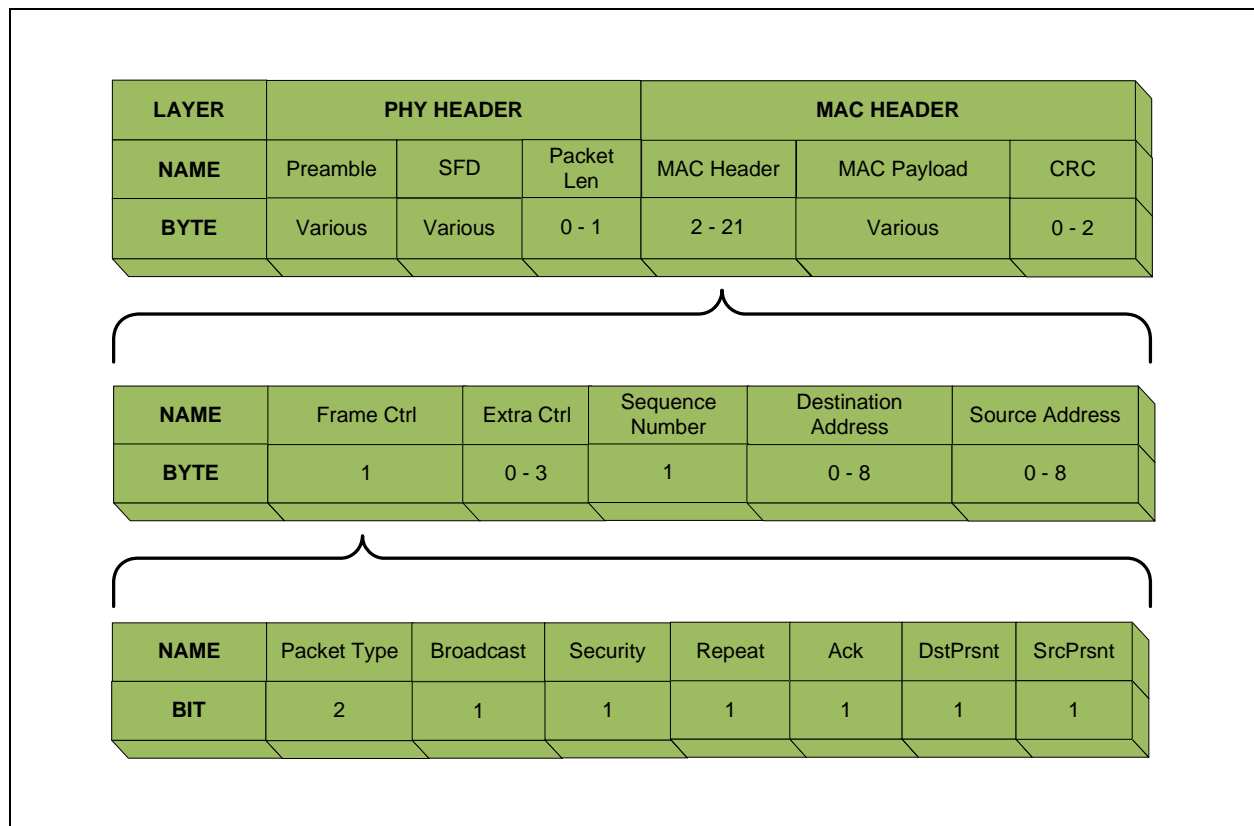
MiMAC FRAME FORMAT

The MiMAC frame format definition ensures that it is easy to learn and easy to support for wireless application developers. As a byproduct, the universal packet format simplifies the sniffer implementation – it is possible to implement only one sniffer software running on the PC, while using different hardware transceivers to sniff the air and send packets to the PC to interpret them. Since all packets have the same format in the MiMAC frame format definition, the interpolation in the MiMAC layer is the same across all RF transceivers from Microchip.

The criteria to evaluate the frame format are its capability and its efficiency. Compared to IEEE 802.15.4, the virtual industrial standard for short-range, low data rate and low-power wireless PAN, the MiMAC frame format provides essentially the same capability with more efficiency. As a comparison, a typical minimum IEEE 802.15.4 frame is 9 bytes in the MAC header, while MiMAC unicast can be as short as 2 bytes.

Figure 2 shows the details of the MiMAC frame format.

FIGURE 2: MiMAC FRAME FORMAT



The packet format of the RF transceivers consists of at least two parts at the top layer:

1. PHY Layer
2. MAC Layer

PHY Layer

The PHY layer is used by the transceiver to synchronize communication and ensure reliability of the communication. The functionalities of the individual field in the PHY layer are:

- a) Preamble is used to synchronize the communication. For different transceivers, the preamble may be of different lengths and the contents may be different. Some transceivers may be able to configure the length and content of the preamble. If the preamble is configurable, simply try to configure the preamble according to the recommendation mentioned in the RF transceiver data sheet. The MiMAC frame format does not regulate the Preamble field.
- b) Start-of-Frame Delimiter (SFD) is usually used with preamble to ensure synchronization of the communication. Some transceivers may be able to enable/disable the SFD or configure the contents of the SFD. If the SFD is configurable, it is strongly recommended to enable the SFD and set the content according to the recommendations of the transceiver data sheet. The MiMAC frame format does not regulate the SFD field.
- c) The Packet Length field is to specify the length of the MAC frame. Some transceivers have this mode to only transmit packets with a fixed length. In this case, the Packet Length field in the PHY header can be omitted. The Packet Length field is not regulated by the MiMAC frame format.

MAC Layer

The MAC layer of the MiMAC frame format consists of three sublayers; the MiMAC frame format regulates all three parts:

- MAC Header
- MAC Payload
- Cyclic Redundancy Check (CRC)

MAC HEADER

The MAC Header field provides crucial information to the receiver of the packet on how to interpret the packet. It consists of five subfields:

- Frame Control
- Extra Control
- Sequence Number
- Destination Address
- Source Address

Frame Control

The Frame Control field is used to interpret the MAC header. It has seven separate subfields to control different aspects of the MAC layer. The detailed descriptions of each subfield in Frame Control are:

- The 2-bit Packet Type field specifies how to interpret the packet, including its payload. For different packet types, the MiMAC layer should handle the packet differently.
 - For a data packet, the packet type is `0b00`. When receiving a data packet, MiMAC will usually pass the MAC payload directly to the upper protocol layer. A data packet may be handled in the upper protocol layer, or directly in the application.
 - For a command packet, the packet type is `0b01`. In this case, the first byte of the effective MAC payload is the MAC command, followed by optional command parameters. When receiving a command packet, MiMAC will usually pass the MAC payload to the upper protocol layer, with a flag to indicate that it is a command frame. It is for the upper protocol layer to interpret the command. A command packet is usually handled in the upper protocol layer.
 - For an Acknowledgement packet, the packet type is `0b10`. An Acknowledgement packet has neither a source address nor a destination address. It depends on the sequence number to identify the packet which is to be Acknowledged. The Acknowledgement packet will be handled by MiMAC; sometimes only by the transceiver hardware. The advanced features in the MiMAC layer, such as automatic Acknowledgement and retransmission, all depend on the Acknowledgement packet. The Acknowledgement frame is not passed to the upper protocol layer.
 - The packet type, `0b11`, is reserved for advanced features for some transceivers and Microchip proprietary protocols. The MiMAC layer will directly pass the received packet with this packet type to the upper protocol layer. When the MiMAC layer receives a request to send such a packet, it will send out the packet without any modification.
- The 1-bit Broadcast field specifies if the packet is a broadcast or unicast. When this bit is set to '1', this packet is a broadcast without the destination address; otherwise, clearing this bit means a unicast message with a destination that is either present or inferred. By using 1 bit in the frame control to specify the broadcast, the MiMAC frame format specification essentially avoids transmitting a special broadcast address in the Destination Address field.

- The 1-bit Security field specifies if the MAC payload has been encrypted during transmitting. Setting this bit indicates that the MAC payload requires a decryption process to get the raw data. When security is enabled, an additional auxiliary security header will be present after the MAC header. Refer to the “**MiMAC Security Module**” section to interpret the auxiliary security header.
- The 1-bit Repeat field specifies if the packet needs a repeater to forward this packet. This bit is useful only for the device with repeating capability. When this bit is set, the repeater that receives this packet will forward this packet to extend the range of communication coverage, on the condition that the destination address is not the repeater's address.
- The 1-bit Acknowledgement field specifies if an Acknowledgement packet is expected from the receiver. When this bit is set to '1', an Acknowledgement packet with the same sequence number needs to be received by the sender in a predefined period. The time-out period for the Acknowledgment depends on the transceiver design. This bit is different from the packet type Acknowledgement. The Acknowledgement bit indicates that a packet of packet type Acknowledgement is expected to confirm the delivery of the current packet. While the packet of packet type Acknowledgement is the response to the packet with the Acknowledgement bit set.
- The 1-bit Destination Present field determines if the destination address exists in the MAC header. When this bit is set, the destination address, with the length defined by the transceiver or the upper communication protocol, is present in the MAC header. When this bit is cleared, the destination address does not show up in the MAC header. The absence of the destination address can happen in the following conditions:
 - In the Acknowledgment packet, there is no destination address. When the packet type is 0b10, the Destination Present bit must be cleared.
 - In a broadcast packet, there is no destination address. When the Broadcast bit is set, the Destination Present bit must be cleared.
- The destination address can be omitted if an inferred destination is used. In such case, the Destination Present bit must be cleared. When the Inferred Destination mode is used, the destination address is still used when calculating CRC, but not transmitted. When other transceivers receive the packet, they will check the CRC with their own address added into the packet at the position of the destination address. A CRC error, in this case, is either because of transmission error or the message is not for this receiving node. In any of the above conditions, the packet will be discarded by the receiving node. Only the intended target transceiver does not generate a CRC error when its own address is used to calculate the CRC as the destination address, thus, the packet is accepted and handled accordingly in the upper protocol layer only by the intended target device. Hiding the destination address not only saves time and energy to transmit those addresses, but also provides minimal protection to avoid complete exposure of the network activities.

There is a very slight chance (about 0.0015% for 2-byte CRC) that two transceivers with different addresses might generate the same CRC code in the transmission range. The Inferred Destination mode is suitable for the majority of applications. For applications which require absolute certainty of the destination, it is recommended to set the Destination Present bit.

<p>Note: The inferred destination address method is Microchip's Intellectual Property (IP). Patent application for this method is now pending for approval.</p>
--

- The 1-bit Source Present field determines if the source address exists in the MAC header. When this bit is set, the source address, with the length defined by the transceiver or the networking protocol, is present in the MAC header. When this bit is cleared, the source address does not show up in the MAC header. The existence of the source address during normal data transmission depends on the application needs.

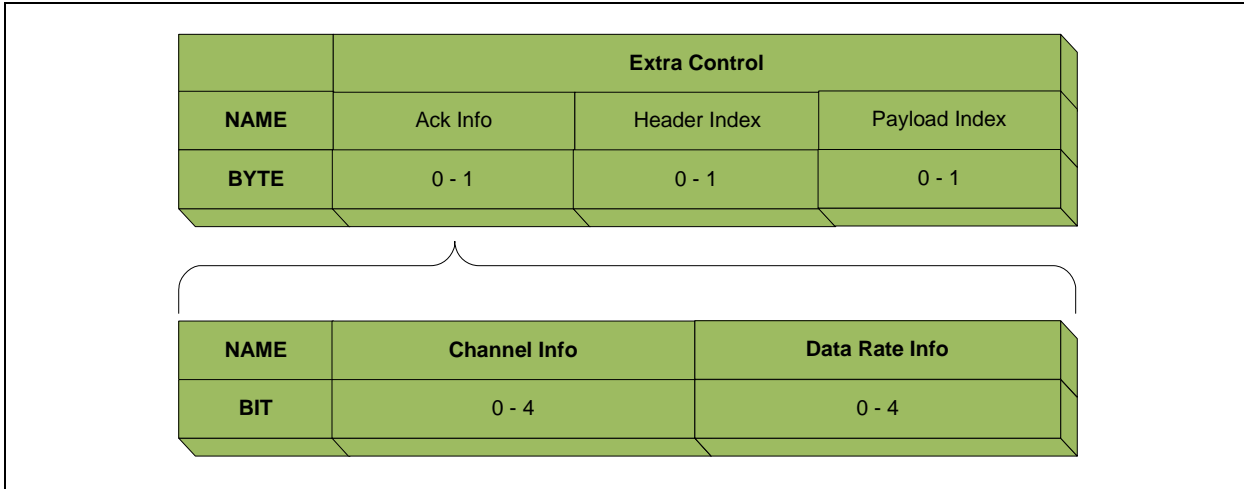
Extra Control

For some transceivers with advanced features, such as an upper layer security module, adaptive data rate and channel, more information is required to interpret the

MAC information. Usually, these fields are only reserved for high-end transceivers that will be used by the transceiver hardware, instead of software.

Figure 3 provides the definition for the Extra Control field.

FIGURE 3: EXTRA CONTROL FIELD FOR ADVANCED FEATURES



The Extra Control field consists of three parts:

- Acknowledgement Information
- Header Index
- Payload Index

The Acknowledgement Information is present only if an Acknowledgement is required and adaptive channel feature or data rate feature is turned on. The Acknowledgement Information is mainly used by the hardware to decide the data rate or channel to be used to send back an Acknowledgement. The Channel Info field is used for the adaptive channel feature and the Data Rate Info field is used for the adaptive data rate feature. The adaptive channel feature enables the transceiver to transmit and receive data at different frequencies. This feature is very useful for big networks working in crowded, unlicensed frequency band. For large networks, this feature enables each and every individual wireless node to receive at the frequency (channel) with the lowest noise and transmit at the receiving frequency (channel) according to the destination device. The adaptive data rate feature enables the transceivers to transmit and receive packets at different data rates. It is similar to the adaptive channel feature and enables more efficient data transfers in the network.

Note: The adaptive channel feature is Microchip's Intellectual Property (IP). Patent application for this method is pending for approval.

The Header Index and Payload Index are specifically used for the hardware security engine, especially for encryption and authentication procedures. It is used to identify the authentication materials and secured materials if the security is not performed in the MAC layer, but at the higher protocol layers. The Header Index and Payload Index are present only if security is enabled, but not performed in the MiMAC layer. The MiMAC specification does not define how to handle security that is not performed in the MiMAC layer. It is up to the upper protocol layer to use these extra control fields to perform a security operation in the corresponding security layer.

Sequence Number

The sequence number is used to identify individual transmitting packets. The sequence number for any transceiver must start from a random number and then increase with every packet transferred. The sequence number is usually used in the Acknowledgement packet to identify the packet that is Acknowledged. As a rule, the sequence number for the Acknowledgement packet must be the same as the packet that is to be Acknowledged.

When there is no network layer provided by the upper protocol layer, the sequence number is used to identify the broadcast message; thus, no rebroadcast is necessary if such a rebroadcast has been performed before.

Destination Address

The destination address defines the target address of the unicast packet. This length of the field is 0 to 8 bytes. The destination address in the MAC header is decided by the Destination Present flag in the Frame Control field.

If the length of the Destination Address field is not zero, the length of the destination address is decided by the transceiver addressing mechanism and the application. The application layer can select the address length from 2 to 8 bytes, depending on the network size and the specific application.

If the length of the Destination Address field is zero, the possible scenarios are:

- The packet is an Acknowledgement.
- The packet is a broadcast message, indicated by the Broadcast bit, which is set in the Frame Control field.
- The destination address is inferred by using CRC.

Source Address

The source address defines the address of the transmitting device. The length of this field is 0 to 8 bytes. The source address is decided by the Source Present flag in the Frame Control field.

If the length of the Source Address field is not zero, the length of the source address is decided by the transceiver addressing mechanism and the application. The application layer can choose the address length from 2 to 8 bytes, depending on the network size and the specific application.

The address length for the destination and source address must be identical for the same network. If the length of the Source Address field is zero, the source address of the unicast message is not essential for this particular application. Whether to include the Source Address field in the MAC header, during normal packet unicast, can be configured in the MiMAC layer.

MAC PAYLOAD

The MAC payload is the information transmitted by Microchip proprietary wireless protocols or by the application layer. It is up to the Microchip proprietary wireless protocol layers or customer application to interpret the information. The MAC payload will be directly passed to the Microchip proprietary wireless protocol layers by the MiMAC programming interface without any modification. If the MAC payload has been secured, it will be unsecured by the MiMAC security module first. Only the decrypted plain text of the MAC payload will be passed to the upper layer by the MiMAC programming interface. If the security checking failed due to any reason, the whole packet will be discarded in the MiMAC layer. With the MAC payload, its length will also be passed to the upper protocol layer. The MAC payload length is calculated from the packet length from the PHY layer, minus the MAC header length, and the possible adjustment for the security module.

MAC CRC

The CRC field in the MAC layer is used to ensure the integrity of the packet during transmission. Hardware CRC generating/checking are provided in some RF transceivers. For transceivers, which do not have the hardware CRC generating/checking capability, the CRC software is used.

When CRC software is used, both loop and look-up table CRC generation methods can be used. Generally, the loop CRC generation method uses about 600 bytes less programming space, but runs 3-4 times slower than the look-up table method. Both methods generate an identical CRC value, thus they are interchangeable. The choice of either method depends on the individual application requirements.

In normal conditions, 2-byte CRC is preferred, balanced by its reliability and simplicity. CRC is highly recommended for all data transmissions. CRC is mandatory when the destination address is omitted during unicast. The “**Destination Address**” section describes how to omit the destination address.

MiMAC SECURITY MODULE

Due to the physical aspect of wireless communication, the content of the information exchange over the air is equally easy to access for all parties, either intended or unintended listeners. Therefore, securing the packets is essential to some applications. The MiMAC security module helps to address the security needs of the applications by the following ways:

- If the transceiver hardware supports a security module, including cipher and different Security modes, it is recommended to use the hardware security engine directly. To encrypt and decrypt a packet in firmware consumes a relatively large amount of MCU system resources, thus it lowers the throughput, and raises the speed and power consumption requirement for the transceiver host MCU. In this case, the MiMAC security specification does not apply.
- If the hardware security engine provides only the block cipher, but not Security modes, it is recommended to use the hardware security cipher but apply the software Security modes on top of the hardware cipher. In this case, the MiMAC security cipher does not apply but the MiMAC Security modes specification applies.
- If the transceiver hardware does not provide any security support, both Cipher and Security modes in the MiMAC security specification apply. If users prefer block cipher for the one chosen by MiMAC, an alternative MiMAC security module provides a predefined interface to invoke any block cipher.

Selecting Default MiMAC Security Engine

Selecting the default MiMAC security engine depends on three criteria:

- Security Engine IP Issues
- Low-Cost Security
- Enhanced Security Strength

SECURITY ENGINE IP ISSUES

Among all the popular security engines that are in the public domain, the good candidates which have no IP issues are:

- Data Encryption Standard (DES/TDES)
- Blowfish/Twofish
- Serpent
- Advanced Encryption Standard (AES)
- Tiny Encryption Algorithm (TEA/XTEA/XXTEA) Family

All these security engines are freely available, have reference designs and are implemented in real products in large volume.

LOW-COST SECURITY

Low-cost implementation ensures that the security module can be implemented on a low-cost MCU with limited system resources and computation speed.

DES/TDES – Previous generation of crypto standards; known to be complex and require relatively more system resources relative to their security strength.

Blowfish/Twofish, Serpent and AES – Provide more secured algorithm while the implementation is simpler than DES families. However, the system resources required for these ciphers are still higher than expected for an embedded system.

Note that, typical implementation of these encryption engines requires at least a 4-Kbyte programming space. On the contrary, the typical implementation of a TEA family requires a couple hundred bytes of programming space and the speed of execution is faster.

Considering the system resources for an embedded system, the security engines of a TEA family meet the requirement for this criterion.

ENHANCED SECURITY STRENGTH

A security engine with a known weakness is not preferred for the MiMAC security specification.

Within the security engines of the TEA family, there are 3 variants: TEA, XTEA and XXTEA. TEA is the original implementation, first published in 1994. It has a known weakness of the equivalent keys. The best related key attack on the TEA security engine requires 2^{32} chosen plain texts under a related key pair, with 2^{32} time complexity. Like XTEA, XXTEA was developed to enhance the security strength beyond TEA. It is a heterogeneous, unbalanced Feistel network block cipher that does not restrict the block size. As a result, XXTEA is likely to be more efficient to handle longer messages, since XXTEA can be applied to an entire message instead of encrypting block by block. However, XXTEA has the limitation of requiring at least 8 bytes of encryption data. XXTEA cannot become a hands-on choice without modification to the security engine itself.

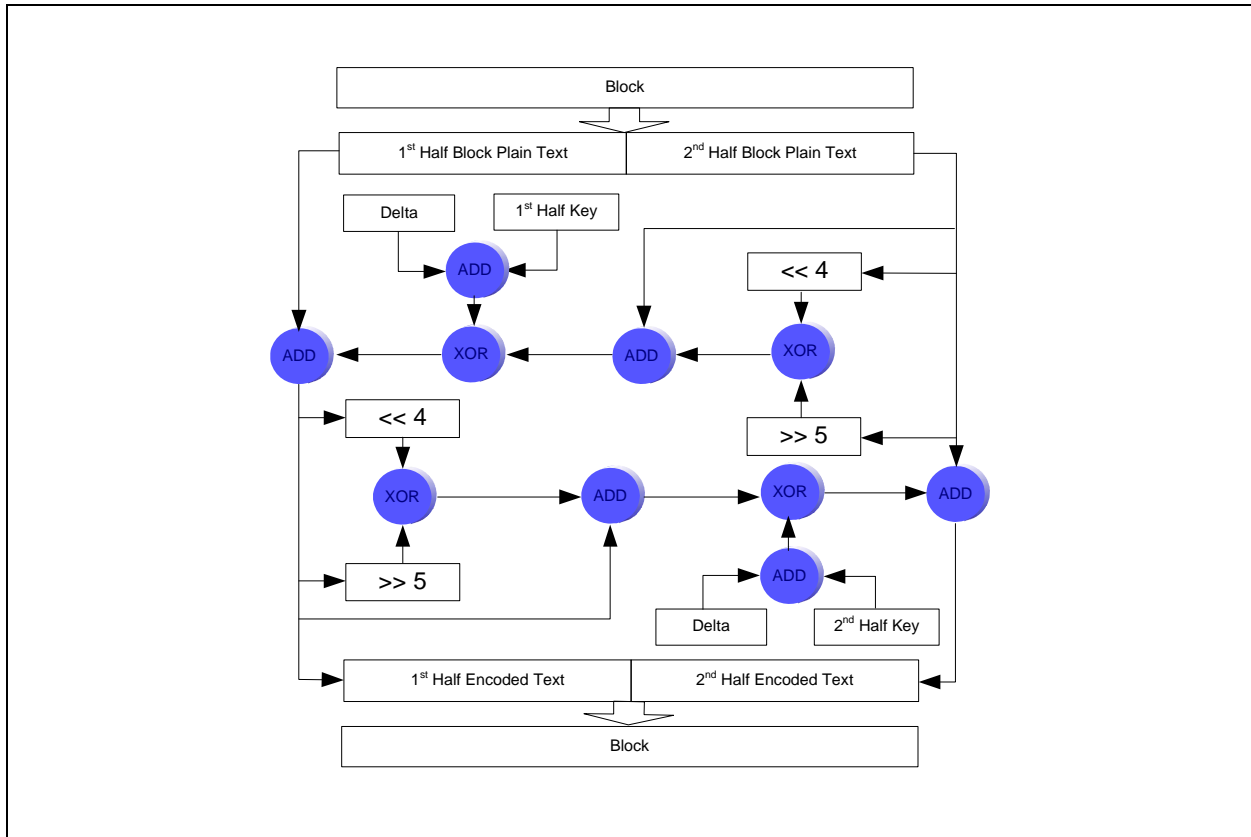
After analyzing all criteria for choosing a security engine, we are left with the XTEA in the TEA family as our choice of a default security engine in the MiMAC security specification.

XTEA Block Cipher

XTEA is a 64-bit block cipher with 128-bit keys. It is designed to bypass the weakness found in the TEA cipher. It was first published in 1997 by David Wheeler and Roger Needham of the Cambridge Computer Laboratory in Cambridge University, UK, and now is in the public domain. It is not subjected to any patent.

Figure 4 shows how an XTEA cipher works.

FIGURE 4: BLOCK DIAGRAM OF XTEA CIPHER



The latest crypto analysis shows that XTEA can only be broken with a related key differential attack under extreme conditions. To perform the related key differential attack, the attacker needs to observe the cipher operation under several different keys and obtain encrypted contents for a set of known plain texts. The best known attack result is 26, out of 64 rounds of XTEA, requiring $2^{20.5}$ chosen plain texts and a time complexity of $2^{115.15}$. (Youngdai Ko, Seokhie Hong, Wonil Lee, Sangjin Lee and Jongin Lim. "Related Key Differential Attacks on 26 Rounds of XTEA and Full Rounds of GOST". In proceedings of FSE '04, lecture notes in Computer Science, 2004 Springer-Verlag.) This means that the conditions and complexity for breaking the XTEA are extremely difficult. Even if every condition has been met, the time to break XTEA on a 1000 MIPS computer will be 1.46×10^{18} years! On the contrary, the latest estimate on the age of the universe is only about 1.4×10^{10} years.

ADVANTAGES OF XTEA

One of the greatest advantages of XTEA is that the system resources required to encrypt or decrypt the information are very limited. A closer look at the XTEA algorithm reveals that the volatile memory requirement for XTEA is extremely low compared to other security engines with similar strength. Therefore, the XTEA is well known to be used in embedded systems with few resources.

Another advantage of XTEA is that the required resources and complexity of the algorithm can be fine tuned by applying different round times to the algorithm. Fewer rounds will perform the algorithm faster and the complexity decreased linearly with the rounds. However, it is easier to break the algorithm with fewer rounds. For wireless applications that MiMAC serves, the required security level and response time varies significantly. The capability of easily adjusting the security level and system resource requirement in XTEA is very valuable for working with a wide range of applications.

Modifying XTEA Block Cipher

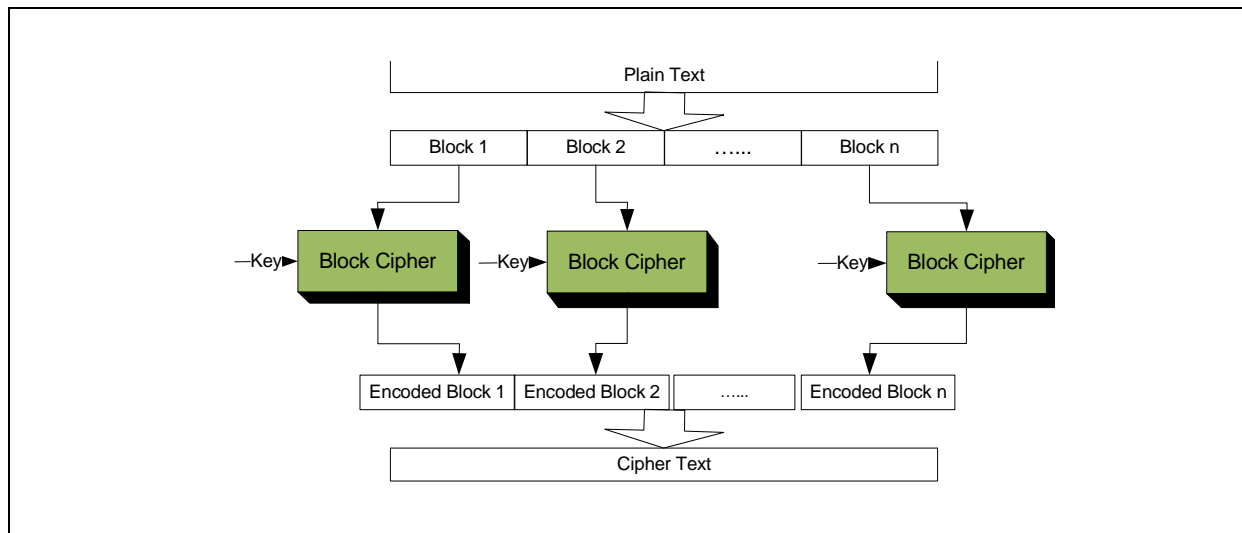
The XTEA cipher engine suits the security needs of an embedded system. However, XTEA needs further modification to best fit into the MiMAC security strategy.

SECURITY MODES

Usually, a security engine applies different Security modes to secure the data. The simplest implementation of a Security mode for block cipher is Electronic CodeBook (ECB) mode. In simple words, the message is divided into multiple blocks with the same block size defined by the cipher, and then the cipher is applied to each individual block to encrypt the input data. Similarly, when a block cipher decoder is used, the process is reversed and the data is decrypted.

Figure 5 shows how the block cipher works to encode in ECB mode.

FIGURE 5: FLOW DIAGRAM FOR BLOCK CIPHER IN ECB MODE



However, the ECB mode has a disadvantage – it does not hide the data pattern. For instance, if all the blocks of plain text are the same, the output encrypted data will also be the same, thus giving a significant hint to the hackers on how to break the security engine.

To overcome the disadvantage of ECB mode, Counter (CTR) mode uses a non-repeated nonce to hide the pattern in the plain text. This requires additional resources, but it significantly improves the security on the output message.

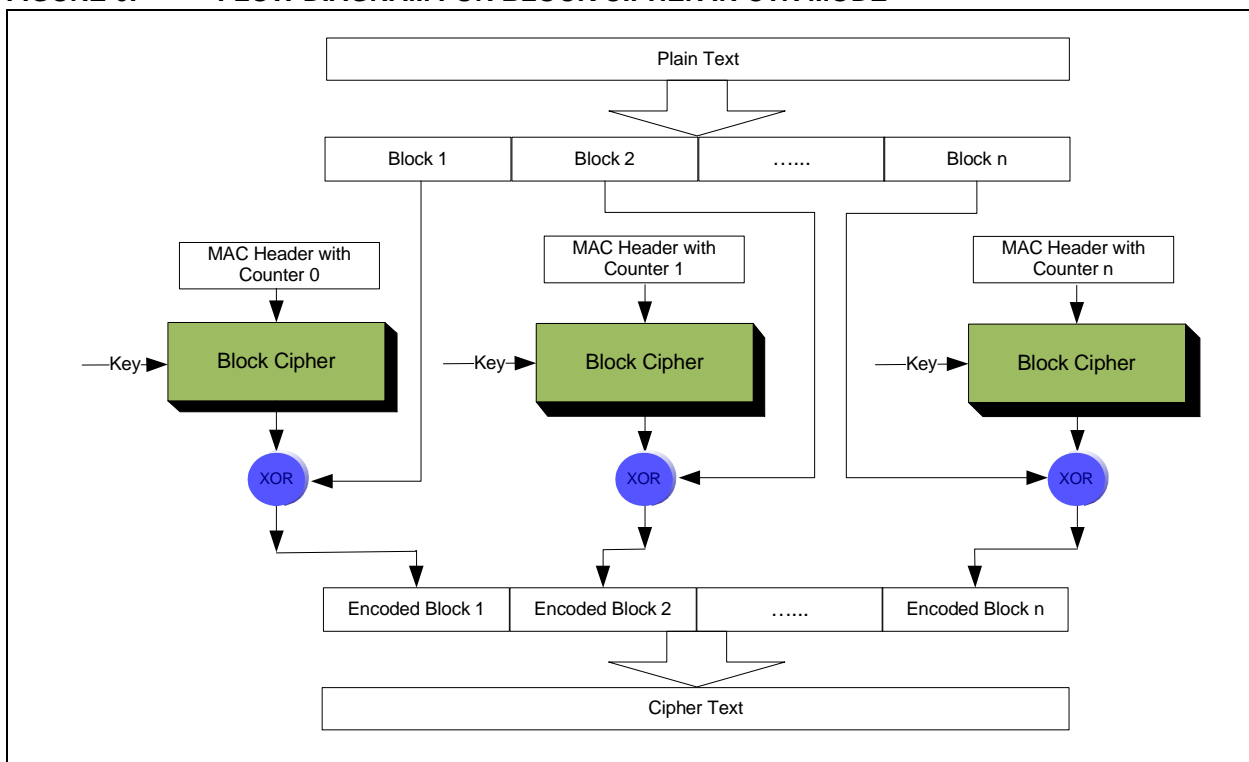
The MiMAC security module specifies the nonce as the MAC header.

- If the MAC header is longer than the block size, fill the nonce with the MAC header up to the size limit, starting from the frame control byte as the lowest byte in the nonce.
- If the MAC header is shorter than the block size, fill the nonce with the MAC header, starting with the frame control as the lowest byte in the nonce and fill the rest of the nonce as zero.

Finally, the highest byte of the nonce will be the counter, starting at zero, and automatically increased for the subsequent blocks.

Figure 6 shows how the block cipher works in CTR mode.

FIGURE 6: FLOW DIAGRAM FOR BLOCK CIPHER IN CTR MODE



AN1283

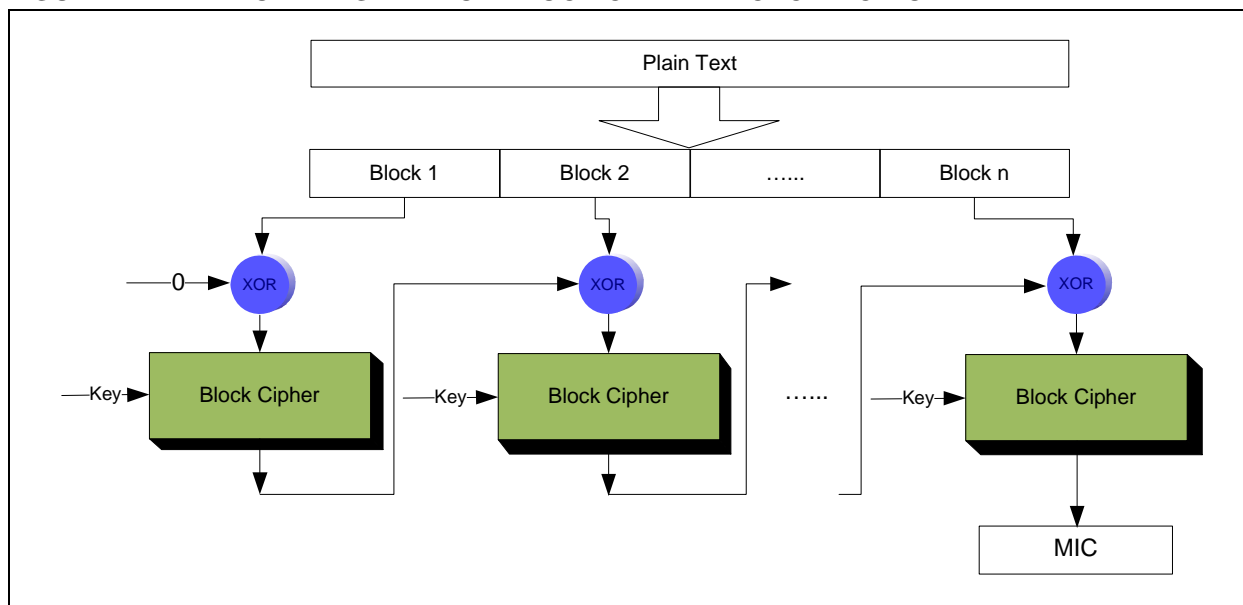
The wireless communication should not only prevent exposing the information, it should also ensure that the information does not have interference over the normal operation of the network, either intentionally or unintentionally. The encryption of the information in CTR mode may prove to be not enough for the following reasons:

- The replay attack can be performed easily with a simple sniffer. It will seriously affect the network operation in some applications. Replay attack is performed by transmitting the identical packet received. In some applications, a receive identical message may be undesirable. A good example is a message to toggle a light.
- The decryption process cannot detect any failure, thus any random data transferred may be potentially operable on the network after the decryption process.

Apart from the CTR Encryption mode, MiMAC needs to define operation modes to authenticate the message. Authentication ensures that the transferred message has not been altered in any way by checking the attached Message Integrity Code (MIC). For the block cipher, the Standard Authentication mode is the Cipher Block Chaining Message Authentication Code (CBC-MAC). CBC-MAC is an operation mode not associated with any particular security engine. In the IEEE 802.15.4 specification, CBC-MAC mode is applied with the AES-128 engine. In the MiMAC security specification, CBC-MAC can be applied to the XTEA block cipher. In the MiMAC security specification, Authentication modes, XTEA-CBC-MAC-32 and XTEA-CBC-MAC-64, are defined to generate a 32-bit or 64-bit MIC.

Figure 7 shows the CBC-MAC mode procedure.

FIGURE 7: FLOW DIAGRAM FOR BLOCK CIPHER IN CBC-MAC MODE



As shown in Figure 7, the XTEA block cipher acts as a Hash function. To invoke CBC-MAC mode in XTEA, the message is broken into small blocks with the block size defined by the block cipher. By default, XTEA defines a 64-bit block size. If the final block is only partially full, fill the rest of the block with zero. The first block is used as the input to the XTEA engine with a predefined key. After the crypto process, the output from the XTEA engine will be XORed with the next block as the input to the XTEA block cipher. After the final block has been processed, the final output from the XTEA engine is the MIC. For XTEA-CBC-MAC-64 mode, the full final block will serve as the MIC; for XTEA-CBC-MAC-32, only the lower 32 bits of the final result will serve as the MIC. MiMAC will transmit the MIC attached to the end of the packet. At the receiving side, the node will calculate the MIC in exactly the same process. Then, the receive node will compare the calculated MIC with the MIC that

is received as an attachment to the original message. If the two MICs are identical, the entire received packet will be accepted; otherwise, the packet will be discarded.

CBC-MAC can be used to prevent a replay attack. Usually, the packet that has been sent with CBC-MAC authentication will include a Frame Counter field with a predefined length (typically 4 bytes) after the MAC header. For every packet that is transmitted, the frame counter will be increased by one. At the receiving side, only the packet with a frame counter value higher than the recorded value will be accepted. As a result, sending a repeated packet as a replay attack is performed and will be discarded. If the sender intentionally modifies the frame counter to be a higher value, the packet cannot pass the authentication check, since the frame counter value is used to calculate the MIC that is attached at the end of the packet.

CBC-MAC mode is used to authenticate the message, but the mode itself does not encrypt the message. IEEE 802.11i uses Offset CodeBook (OCB) mode to authenticate and encrypt the data at the same time, thus saving requirements for computing power. However, OCB mode has appeared in a patent application. Even though there is a special exemption to use OCB mode under the GNU General Public License (GPL), it is wise for the MiMAC security specification not to depend on OCB mode with the potential IP issues.

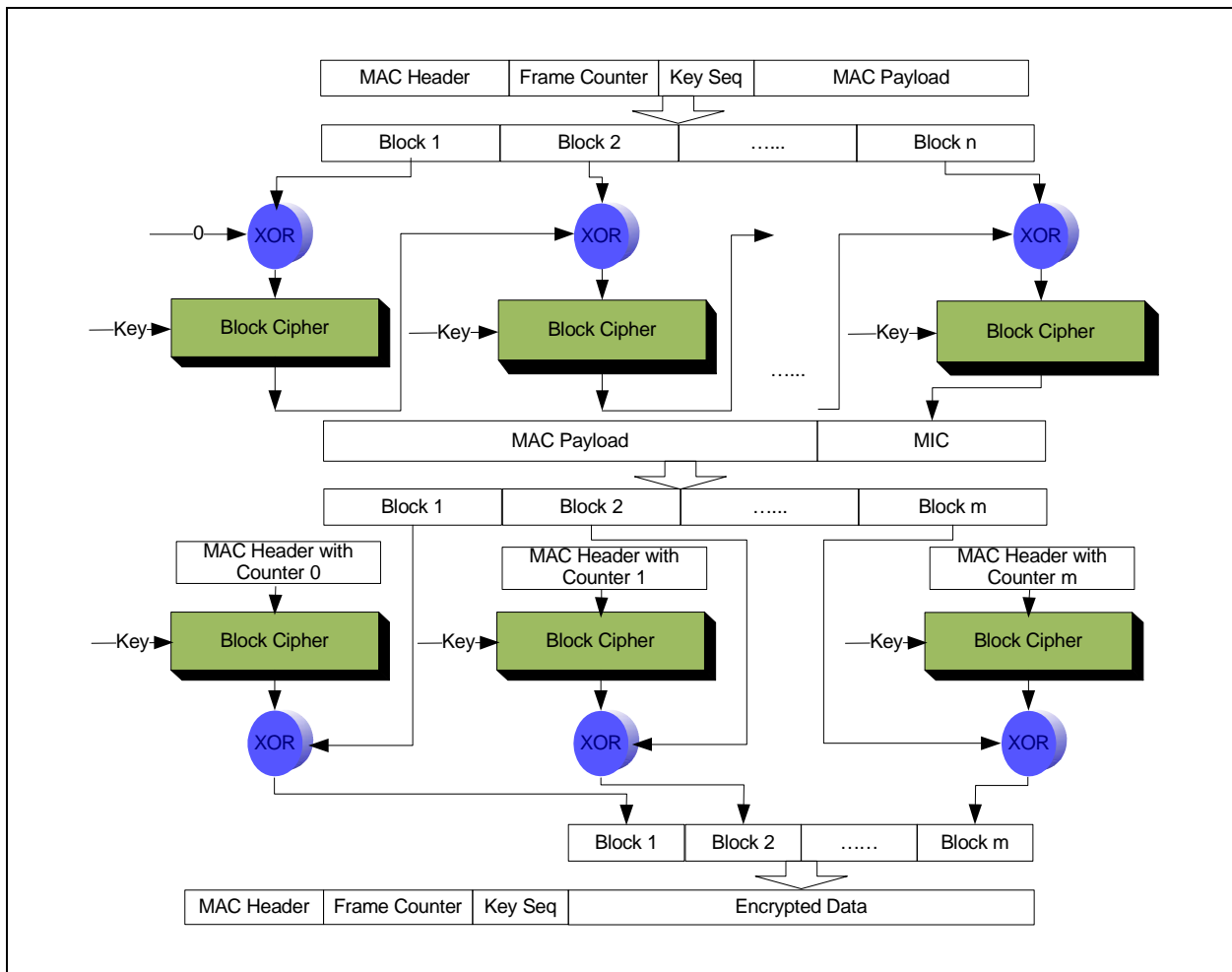
As an alternative to OCB mode, Counter with CBC-MAC (CCM) mode performs 2 passes to the message instead of 1 pass in OCB mode, thus doubling the computing resources requirement. The CCM mode is basically applied to the CBC-MAC mode first to generate the 32-bit or 64-bit MIC, followed by applying CTR

mode to the message and the MIC to encrypt the message. CCM mode requires doubling the computing resources compared to CTR mode or CBC-MAC mode, but provides the most complete protection over the data transferred over the air. Depending on the length of the MIC, there are two CCM modes available: CCM-64 and CCM-32.

When MiMAC applies CCM mode, the complete packet, including the MAC header, security auxiliary header and MAC payload, are used to authenticate the message, generating the MIC to protect the whole packet. Only the MAC payload and the MIC will be encrypted.

Figure 8 shows the entire CCM mode process.

FIGURE 8: FLOW DIAGRAM FOR BLOCK CIPHER IN CCM MODE



When a wireless node receives a packet that is secured by CCM:

- It first applies CTR mode to decrypt the MAC payload.

The decrypted plain text consists of raw data and the MIC.

- Then CBC-MAC mode is applied to the raw data to calculate the MIC.
- Finally, the calculated MIC is compared with the MIC decrypted from original data.

If the two MICs do not match, the whole packet is discarded. Similar to CBC-MAC mode, a frame counter can be included in the packet when CCM mode is used. The additional Frame Counter field in the packet can effectively prevent the replay attack for the same reason that is described above for CBC-MAC mode.

As a result of using Security modes on top of the XTEA block cipher, one minor benefit is that only the encoding function for XTEA is required to be implemented.

KEY STRENGTH

Generally, the longer the key, the more security strength for the crypto engine. XTEA was first developed and published as a 64-bit block cipher with a 128-bit symmetric key.

However, export regulation from the U.S. government requires particular steps to export an encryption engine with a symmetric key of more than 64 bits.

Therefore, the XTEA security engine needs to be demoted to support a 64-bit symmetric key as one of the operating modes.

Even though XTEA is developed as a 64-bit block cipher with 128-bit symmetric key, it can be modified to support a 32-bit block cipher with a 64-bit symmetric key following the same concept. The differences are the block size and the definition of DELTA, the constant magic number coming from the golden ratio. With the introduction of a 32-bit block size and 64-bit key, there are two modes for the XTEA block cipher: XTEA-128 and XTEA-64. For the XTEA-64, CBC-MAC-64 and CCM-64 Security modes are no longer available due to the reduced block size.

Because of demoting XTEA, it is expected that the speed of the security process increases, while the strength of the security module decreases. As the XTEA-64 cipher is not a standard security engine, no crypto analysis has been performed on this algorithm, thus the complexity of breaking the algorithm is unknown so far. It is believed that the XTEA-64 still provides enough security for casual users of a crypto engine. Users are encouraged to increase the round to make the engine more secure to some extent. For customers requiring more confidence about security, the standard XTEA, or XTEA-128, is always ready to deliver the stronger confidence upon authorization to meet U.S. export regulation regarding a security engine.

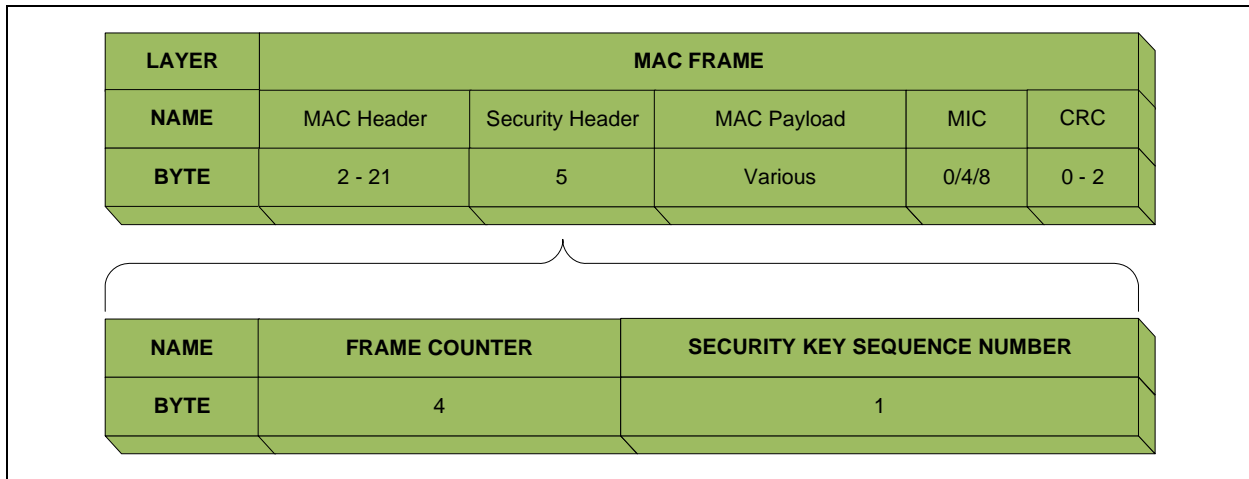
SECURED PACKET FORMAT

When the MiMAC packet is secured, additional information is needed to ensure the successful process of the decryption process. Basically, the transmitting side needs to send all necessary materials that may be useful to decrypt the packet, except the security key itself.

Two parts are considered to be security materials that are essential to the security process: frame counter and security key sequence number. They are part of the auxiliary security headers that immediately follow the MiMAC header.

Figure 9 provides the frame format for a secured packet.

FIGURE 9: FRAME FORMAT FOR SECURED PACKET



The frame format for a secured packet differs from an unsecured packet in the following ways:

- The secEn flag in the frame control byte is set.
- Includes an additional auxiliary field, called Security Header; it contains two parts:
 - Frame counter of 4 bytes is used to avoid replay attack. The details of how to avoid a replay attack can be found in the “**Security Modes**” section.
 - Security Key Sequence Number is used to identify which security key is to be used.
- The MIC that is attached to the end of the MAC payload. The length of the MIC depends on the Security modes to be used.

Under different Security modes, the secured portion in the packet varies:

- For CTR mode, only the MAC payload is encrypted; there is no MIC attached.
- For MAC-CBC mode, the authenticated information starts from the MAC header until the end of the MAC payload. The MIC is either 4 bytes or 8 bytes in length for the MAC-CBC-32 and MAC-CBC-64 modes, respectively.
- For CCM mode, the authenticated information starts from the MAC header until the end of the MAC payload. The encrypted information starts from the start of the MAC payload until the end of the MIC. The MIC length is either 4 bytes or 8 bytes for CCM-32 and CCM-64 modes, respectively.

ALTERNATIVE CIPHER ENGINE SUPPORT

XTEA-128 is a strong security engine, which is suitable for a majority of the applications. However, for some wireless application developers who require a particular security engine, the MiMAC security module suggests an alternative. A security engine interface has been defined for block ciphers. Once this security interface has been implemented for the alternative block cipher, the new security engine can be used in MiMAC without any further modification. Be aware that the three Security modes, CTR, CBC-MAC and CCM, still apply to the alternative security engine to ensure proper protection of the data. Because of these Security modes, only the encoding process for the alternative block cipher needs to be implemented.

The interface to the alternative security engine is shown below:

```
BOOL encode(BYTE *buffer, BYTE *key);
```

In the security interface function call encode, there are two parameters: buffer and key.

- **Buffer**
The buffer parameter is the pointer to the plain text when this function is called. When the function is returned, this buffer will be replaced by encoded data.
- **Key**
The key parameter is the pointer to the security key for the block cipher.

MiMAC UNIVERSAL PROGRAMMING INTERFACE

As already discussed, standardizing the frame format and security module in the MiMAC layer provides great advantages for wireless application developers. The next step is to standardize the software programming interface between MiMAC and Microchip upper proprietary protocol layers.

Microchip supports multiple short-range, low data rate RF transceivers. As discussed before, the MiMAC universal programming interface:

- Makes all Microchip RF transceivers interchangeable in the software development process.
- Reduces the software development risk for wireless application developers.
- Considers the potential of the different transceivers, while still providing a simple interface to work with the upper protocol layers, without any major footprint increase.

There are two kinds of interfaces defined to work with different Microchip RF transceivers: configuration file and function calls.

For different Microchip RF transceivers, the hardware interface, functionality and register settings vary greatly. For configurations that apply only to individual RF transceivers, a configuration file has been defined. These configurations are unlikely to change when the application runs. Usually, the configurations in the file will be set in the initialization process.

For all wireless protocols, all function calls can be divided into four major categories:

- Configuration
- Transmitting Packets
- Receiving Packets
- Special Functionality

The MiMAC programming interface defines one or more function calls for each function category. Calling those programming interfaces from the upper protocol layers will perform virtually all functionality of the RF transceivers. The succeeding sections define the programming interfaces by function categories.

MiMAC Configuration

The MAC layer configuration interface is called from the upper protocol layer to the MiMAC layer to configure the behavior of the RF transceiver. Unlike the configuration parameters defined in the configuration file, the configuration in these function calls may be changed when the wireless applications run. For those features supported by the transceiver hardware, it is expected that the MiMAC layer should set corresponding register bits in the transceiver. For those features not supported by the transceiver hardware, it is expected that MiMAC layer firmware should handle it before transferring control to the upper layer. The following functions are used to configure the MAC layer:

- `MiMAC_Init`
- `MiMAC_SetPower`
- `MiMAC_SetChannel`
- `MiMAC_SetAltAddress`

MiMAC_Init

`MiMAC_Init` function call initializes the behavior of the MiMAC layer. The following MAC behavior can be configured:

- Permanent Address
- Enable/Disable Repeater Mode

This function is called as one of the first steps in Microchip upper protocol layers. After this function call, the MiMAC layer behavior will be initialized in the RF transceiver.

To represent all the configurable information to the MiMAC layer, the following structure is defined and served as an input parameter to the `MiMAC_Init` function call.

```
typedef struct
{
    union
    {
        BYTE Val;
        struct
        {
            BYTE RepeaterMode:1;
            BYTE PAddrLength:4;
        } bits;
    } actionFlag;

    BYTE *PAddress;
} MACINIT_PARAM;
```


The description of this structure is:

- **RepeaterMode**

Enables the transceiver to forward the packets when the Repeat bit is set in the frame control byte in the MAC header.

- **PAddrLength**

Defines the length of the permanent address for the transceiver. The length can be defined from between 2 to 8 bytes.

- **PAddress**

The pointer that points to the permanent address of the transceiver.

The full signature of the function call is shown below. The return value indicates if the operation was successful.

```
BOOL MiMAC_Init(MACINIT_PARAM initValue);
```

MiMAC_SetPower

MiMAC_SetPower function call sets the output power of the transceiver. It is a PHY function call instead of a MiMAC layer. Here, the MiMAC layer is just an intermediate layer to pass the setting to the PHY layer of the transceiver. Instead of using it as a parameter in the MiMAC_Init function call, this interface is defined separately to give the upper protocol layer, or application layer, flexibility to adjust output power freely during run based on application needs.

The full signature of the function call is shown below. The return value indicates if the operation is successful.

```
BOOL MiMAC_SetPower(BYTE outputPower);
```

The input parameter, `outputPower`, is represented by one byte. The individual transceiver should be responsible to interpret the input value and set the proper output power. It is highly recommended for the firmware in the upper protocol layer, or application layer, to use a predefined value recognized by the transceiver, probably in the definition header file of the target transceiver.

MiMAC_SetChannel

MiMAC_SetChannel function call sets the operating frequency of the RF transceiver. The full signature of the function call is shown below. The return value indicates if the operation is successful.

```
BOOL MiMAC_SetChannel(BYTE channel,
                     BYTE offsetFreq);
```

There are two input parameters to this function call: the channel and the offset frequency.

- **Channel**

Defines the center frequency that the RF transceiver works on. The channel number is defined from 0 to 31.

- **Offset Frequency**

It is used in some of the RF transmitters as an additional configuration to set the center frequency at any frequency not defined by the channel. Usually, the offset frequency cannot be larger than the frequency gap between adjacent channels.

When combining the proper setting of the channel and offset frequency, it is possible to fine tune the center frequency of the RF transceiver to be any possible value. For transceivers that define the fixed channel center frequency, strictly in the specification, the `offsetFreq` parameter can be discarded.

Not all channels are supported for every transceiver and data rate or frequency band. If the input parameter channel is not supported by the RF transceiver at the current condition, the current operating channel will not be changed and the return value will be `FALSE` to indicate the failure of the changing channel. Otherwise, the operating channel will be changed in the transceiver and the return value will be `TRUE`.

MiMAC_SetAltAddress

MiMAC_SetAltAddress function call sets the alternative network address after the wireless node joins the network. This function is used only when the transceiver supports multiple addresses and the concept of the Personal Area Network Identifier (PANID). The PANID and the alternative network address are specified by the IEEE 802.15.4 specification. It is not mandatory for all transceivers to support this function. For the transceivers that do not support multiple addresses or PANID, the input parameters will be discarded and the return value will be `FALSE`.

The full signature of the function call is:

```
BOOL MiMAC_SetAltAddress(BYTE *Address,
                       BYTE *PANID);
```

There are two input parameters to this function call: the pointers that point to the address and the PAN identifier.

- **The Address**

It is required for those transceivers that support network addresses to identify the device in the network.

- **PANID**

It is used to identify the network itself.

MiMAC RF TRANSCEIVER CONFIGURATION FILE

Apart from the function calls to configure the RF transceiver from the upper protocol layer on the parameters that are defined in all RF transceivers, various RF transceivers have their special parameters of configurations, which are hard to unify under the same umbrella. For those configurations, the default values, usually not modified once the RF transceiver is up and running, are configured in a separate configuration file under the RF transceiver's directory. The MiMAC specification does not regulate the individual settings of these control variables. Refer to the RF transceiver data sheet to understand and modify those configuration parameters as required.

Communication is the main functionality of an RF transceiver. There are two steps for the reliable communication between two ends of the packet: transmitting and receiving.

Transmitting Packets

The interfaces to transmit packets are defined in this section.

To configure the transceiver to send the packet in the desired way, the following structure is defined to be used as an input parameter to transmit a packet.

```
typedef struct
{
    union
    {
        BYTE Val;
        struct
        {
            BYTE    packetType: 2;
            BYTE    broadcast: 1;
            BYTE    secEn   : 1;
            BYTE    repeat  : 1;
            BYTE    ackReq: 1;
            BYTE    destPrsnt: 1;
            BYTE    sourcePrsnt: 1;
        } bits;
    } flags;

    #if defined(IEEE_802_15_4)
        BOOL    altDestAddr;
        BOOL    altSrcAddr1;
        BYTE    *DestPANID;
    #endif
    BYTE    *DestAddress;
} MAC_TRANS_PARAM;
```

Each variable is described in the following structure:

- Flags are the collections of configurations used to transmit a packet. The flags parameter contains the following configuration options:
 - `packetType` is used to define the packet type. In the universal MAC strategy, the packet type is defined as the following:

0b01	Data Packet	—
0b10	Acknowledgement Packet	Handled in the MiMAC layer, thus, no need for upper protocol layer to transmit such a packet; it is not used.
0b11	Command Packet	—
0b00	0b00 Reserved	Reserved for special functionality for certain transceiver hardware or protocol layer. It is not supported by all transceivers. MiMAC will transmit this kind of packet directly without any further processing.

- The `broadcast` element defines if the packet is a unicast or broadcast. Setting this bit enables broadcast operation for the current packet. Once this bit is set, the `destPrsnt` bit should be cleared and the Destination Address field in the MAC header should not exist.
- The `secEn` element indicates if the transmitting packet needs to secure the MAC payload. The security level and security key should have been defined before in the application layer. The MAC payload from the Microchip upper protocol layer should be the plain text. The MiMAC layer will add the security auxiliary header automatically, as described in the “**MiMAC Security Module**” section. The MAC payload will be encrypted and/or authenticated before transmitting over the air.
- The `repeat` element indicates that this message needs a repeater between the sender and the receiver to pass the message. The repeater will only forward packets, which are between nodes that are out of radio range of each other. This prevents excessive messages from the repeater, even if the sender and receiver can communicate directly.

- The `ackReq` element is used by the receiving end to send Acknowledgement to ensure reliable delivery of the message. The MiMAC layer, either hardware or software, should be able to handle the Acknowledgement frame sending at the receiving end. The sender MiMAC layer should be able to handle the retransmission if no Acknowledgement is received within the predefined time.
- The `dstPrsnt` element indicates if a destination address is included in the MAC header. The destination address can be absent if the packet is broadcast or the destination address is inferred. The destination address can only be inferred if 2-byte CRC is used. Even if the destination address is inferred, a valid destination address should always be provided to the MiMAC layer for calculating the CRC.
- The `sourcePrsnt` element indicates if a source address is included in the MAC header. When the device first tries to establish the connection(s), the source address is required in the MAC header to make the peer node aware of whom it talks to. For the application layer data, however, the source address is optional, depending on the application needs.
- `altDestAddr` is a boolean to indicate if the destination address is an alternative address or permanent address.

- `altSrcAddr` is a boolean to indicate if the source address is an alternative address or permanent address.

Note: For both the `altDestAddr` and `altSrcAddr`, a setting of '0' means the source address is a permanent address, while '1' means an alternative address. This field is only valid in IEEE 802.15.4 mode, when the RF transceiver is capable of sending a packet from either a permanent address or an alternative network address.

- `DestPANID` is the pointer that points to the destination, PANID. This field is only valid in IEEE 802.15.4 mode, where PANID is used as one of the filters to address the destination device, combining with the destination address.
- `DestAddress` is the pointer that points to the destination address. In IEEE 802.15.4 mode, this address can either be a permanent address or an alternative network address, depending on the settings of `altDestAddr`. This field will not be effective if the Broadcast field is set to '1'.

The full function signature to transmit the packet follows. The return value indicates if the operation is successful.

```

BOOL MiMAC_SendPacket (MAC_TRANS_PARAM
transParam, BYTE *MACPayload,
BYTE MACPayloadLen);

```

In the `MiMAC_SendPacket` function call, the `transParam` parameter regulates all aspects of the transmission options. The input parameter, `MACPayload`, points to the buffer to be transferred. The input parameter, `MACPayloadLen`, specifies the length of the MAC payload.

Receiving Packets

Apart from transmitting a packet, the other most important functionality of the transceiver is to receive a packet. Because of the nature that a packet can be received at any time, while the upper protocol firmware is running, there are two ways to handle the message:

- First approach – By invoking a callback function and letting the upper process layer process the packet immediately.

This approach has faster response time, but invokes a set of function calls across layers up to the application, thus filling the stack space quickly.

- Second approach – By interpreting the packet, storing it in a global variable, and labelling an event to let the upper protocol layer be aware that a packet is available.

This option better suits the state machine architecture of common Microchip stacks.

Both approaches are fine.

To implement the second approach, the following type is defined to contain the information from the packet:

```
typedef structure
{
    union
    {
        BYTE Val;
        struct
        {
            BYTE    packetType: 2;
            BYTE    broadcast: 1;
            BYTE    secEn   : 1;
            BYTE    repeat  : 1;
            BYTE    ackReq: 1;
            BYTE    destPrsnt: 1;
            BYTE    sourcePrsnt: 1;
        } bits;
    } flags;

    BYTE * SourceAddress;
    BYTE * Payload;
    BYTE PayloadSize;
    BYTE RSSI;
    BYTE LQI;

    #if defined(IEEE_802_15_4)
        BOOL altSourceAddress;
        BYTE * SourcePANID
    #endif
} MAC_RECEIVED_PACKET;
```

Defined in this `MAC_RECEIVED_PACKET` structure, the element, `flags`, specifies the configuration of the received packet. The definition of the `flags` element is virtually the same as the `flags` defined in the structure, `MAC_TRANSMIT_PARAM`.

Element, `SourceAddress`, is a pointer that points to the source address if the source address is present in the packet. If the RF transceiver supports IEEE 802.15.4, the source address can be either a permanent address or an alternative network address, decided by the settings in the `altSourceAddress` element. Also, only in IEEE 802.15.4 mode, the `SourcePANID` element is available to specify the PAN identifier of the transmitter.

Element, `Payload`, is a pointer that points to the buffer of the MAC payload. The MAC payload size is specified in the `PayloadSize` element. If the MAC payload has been encrypted and/or authenticated, the payload passed to the Microchip upper protocol layer should have been decrypted and/or the authentication checked. The security auxiliary header should also be removed from the payload passed to the upper protocol layer. Only the `secEn` bit in the `flags` element will indicate to the upper protocol layer if security has been applied to the original data.

The `RSSI` and `LQI` elements indicate the physical aspect of the received packet. `RSSI` is the representative of the average signal strength of the received packet, while `LQI` represents the signal quality of the received packet. `RSSI` and `LQI` are not supported by all the RF transceivers.

Once the packet is received, the content of the packet will be placed into this structure and will be available to the upper protocol layer to process.

The upper protocol layer needs to periodically checked if a packet has been received. Once the upper protocol layer is aware that a packet has been received, it will handle this global structure. There are two function call interfaces by the upper protocol layer to handle a received packet: `MiMAC_ReceivedPacket` and `MiMAC_DiscardPacket`.

MiMAC_ReceivedPacket

The `MiMAC_ReceivedPacket` function call is called by the upper protocol layer to periodically check if a packet has been received. It has no input parameters and returns a boolean to indicate if a packet has been received. The full function signature can be found below:

```
BOOL MiMAC_ReceivedPacket(void);
```

The `MiMAC_ReceivedPacket` function call runs the `MiMAC` stack and checks if a packet has been received.

Once the return value of this function call is `TRUE`, the content of the `MAC_RECEIVED_PACKET` structure has been filled and is ready for the upper protocol layer to process.

MiMAC_DiscardPacket

The `MiMAC_DiscardPacket` function call is called by the upper protocol layers to notify the MiMAC layer that the current received packet has been processed and can be discarded. It is important to discard the packet which has already been processed. Otherwise, the host MCU may run out of memory resources quickly. The `MiMAC_DiscardPacket` function call has no input parameter and no return value. The full function signature is:

```
void MiMAC_DiscardPacket(void);
```

Special Functionality

Apart from transmitting or receiving a message, most of the RF transceivers are able to perform certain special functionalities to make sure that the RF transceivers are able to work at optimal conditions. Two common functionalities that may be valuable to all protocol layers are the capability to perform energy scan and save power. The following functions are defined in the MiMAC programming interface:

- `MiMAC_ChannelAssessment`
- `MiMAC_PowerState`

MiMAC_ChannelAssessment

The `MiMAC_ChannelAssessment` function call will perform a channel assessment to determine if a channel or frequency is suitable for reliable communication.

This operation should not be confused with the CSMA-CA clear channel assessment used before transmission.

The CSMA-CA operation would be done in the lower MAC layer in the `MiMAC_SendPacket` function call to avoid transmitting an RF packet at the same time as the neighboring peer nodes. The operation of channel assessment, sometimes called energy detection scan at the protocol layers, is mainly used to check the noise level at different frequencies to decide which frequency to be used in the communication. This operation is usually called by the protocol layers, either before starting a network, or before a frequency agility operation.

The full function signature is:

```
BYTE MiMAC_ChannelAssessment  
(BYTE AssessmentMode);
```

The `MiMAC_ChannelAssessment` function call has one input parameter to indicate the Channel Assessment mode.

There are two possible Channel Assessment modes:

- Energy Detection

Measures total noise level in the operating channel from all possible sources. Usually, energy detection assessment is used to evaluate noise from natural sources, signal from other modulations and signal from neighboring wireless nodes of the same modulation.

- Carrier Sensing

It is used to detect the communication of the same kind of RF transceivers. It only counts the signal strength of communication that this RF transceiver can receive and interpret. It is usually used to avoid operating a network at a frequency with several neighboring wireless nodes of the same kind.

The Input mode parameter specifies which method is used for channel assessment. Not all RF transceivers are able to perform channel assessment by all, or any, of these Assessment modes. For transceivers that support no Assessment mode, this function will not be supported. For transceivers that support only one of the Assessment modes, the other mode will be discarded. The `MiMAC_ChannelAssessment` function call will return the assessment result to the upper layer. A higher return value represents a noisier environment.

MiMAC_PowerState

The `MiMAC_PowerState` function call is used only by the node that may go to Sleep to conserve power when it is Idle. This is not a MAC function; it is a direct PHY function. Here, the MiMAC interface just passes this function directly to the PHY layer. The full function signature is:

```
BOOL MiMAC_PowerState(BYTE PowerState);
```

The `MiMAC_PowerState` function call has one input parameter, `PowerState`, to indicate the desired power state for the transceiver. There are only two generic power states that are defined:

- `DEEP_SLEEP` – The Sleep state for the transceiver with the lowest power consumption.
- `OPERATE` – Full operating state for the transceiver.

All the values, between 0x00 and 0xFF, may represent certain power states. The detailed definitions will depend on the particular transceiver. As a general rule, it is preferred to define the Sleep state with low-power consumption, next to the Deep Sleep state as 0x01, and increase the value as the Sleep mode consumes more current.

CONCLUSIONS

For developers looking for a short range, low data rate, wireless solution, the choices are plenty across multiple frequency bands, at different data rates and other features. The Microchip MAC (MiMAC) specification provides a low-cost and low-complexity design solution for application developers. It enables RF transceivers, supported by Microchip, to be hooked up with any Microchip proprietary wireless protocols. It is highly recommended for the readers of this application note to refer to the companion application note, *AN1284, "Microchip Wireless (MiWi™) Application Programming Interface (MiApp)"*.

- MiApp is designed to allow the flexibility of using any Microchip proprietary wireless protocol with little or no modification in the application layer.
- MiMAC is designed to allow the flexibility of using any Microchip RF transceiver with the same Microchip proprietary protocol layer.

MiMAC and MiApp work together to offer Microchip customers maximum flexibility in the entire software development process.

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, 2003.

Visit the Microchip web site (www.microchip.com) for the following application notes:

- *AN1284, "Microchip Wireless (MiWi™) Application Programming Interface (MiApp)"*.
- *"MRF24J40 Data Sheet"* – IEEE 802.15.4 2.4 GHz RF Transceiver (DS39776)
- *"MRF49XA Data Sheet"* – ISM Band Sub-GHz RF Transceiver (DS70590)
- *AN1066, "MiWi™ Wireless Networking Protocol Stack"* (DS01066)
- *AN1204, "Microchip MiWi™ P2P Wireless Protocol Stack"* (DS01204)

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.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, rPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Omniscent Code Generation, PICC, PICC-18, PICkit, PICDEM, PICDEM.net, PICtail, PIC³² logo, REAL ICE, rLAB, Select Mode, Total Endurance, TSHARC, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2009, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

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

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

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

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4080

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