TI-15.4-STACK Co-Processor

Accelerate your RF Network Development

Applications

- Point to Point Networks
- Point to Multipoint Networks
- Electronic Shelf Labels

Description

The TI-15.4-STACK Co-Processor on CC1310 is a cost effective, low power, TI-15.4-STACK Co-Processor that provides IEEE 802.15.4 implementation via minimal development effort.

The CC1310 TI-15.4-STACK Co-Processor is an entity which implements the MAC IEEE 802.15.4-2006 standard in a dedicated system on a chip (SoC), providing a simple serial interface to an external host processor for control and processing of the Co-Processor operations.

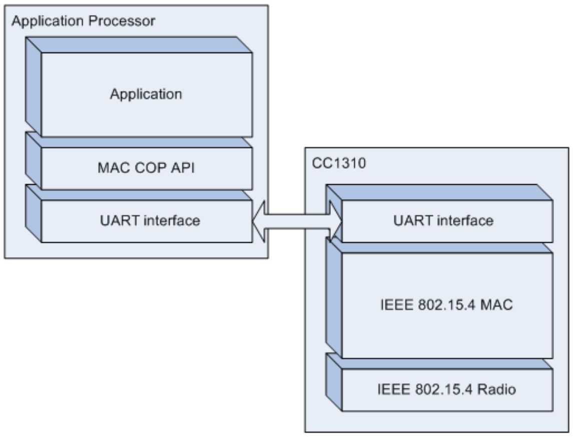
The TI-15.4-STACK Co-Processor approach is a scalable architecture that fits perfectly for configurations where the host co-processor runs protocol stack layers over IEEE 802.15.4(g/e) MAC/PHY (generic IP over 6LoWPAN, ZigBee IP, or ZigBee Pro) or a proprietary application that simply uses the MAC/PHY for the data link.

The TI-15.4-STACK Co-Processor will connect to any microcontroller through UART interface. For example, a Co-Processor can be combined with a Windows or Linux host processor, or be part of an embedded system using MSP430 or other microcontroller.

The TI-15.4-STACK Co-Processor architecture makes it easy for the users to add IEEE 802.15.4 functionality to an existing product and also provides great flexibility in choice of microcontrollers.

The TI-15.4-STACK Co-Processor provides for configuration of network operation in Beacon or non-Beacon mode, with or without Security, and Frequency Hopping. Refer to the *TI-15.4 Stack Developers Guide* for details on setting up and running the various network configurations.

Message frames transported over the serial link follow the formats specified in this document.



Key Features

- *UART interface to application processor*
- *Developer extendable interface API*

Contents

Figures

Tables

1. Physical Interface

The SimpleLink^{TM} CC1310 wireless MCU is the newest member of the family of TI MAC CoP platforms. The CC1310-based CoP includes significant differences compared with existing CC253x platforms:

- TI-RTOS: As with all MAC software products on CC1310, TI-15.4-STACK-CoP is built over TI-RTOS, a Real-Time Operating System developed by Texas Instruments.
- NPI: On CC1310, the TI-15.4-STACK-CoP architecture incorporates a new NPI (Network Processor Interface) subsystem. The NPI subsystem represents a convergence of Texas Instruments Network Processor-based software products (e.g. MAC, BLE, ZigBee) onto a single common architecture. In the Network Processor approach, the core stack operations run on the embedded device, while applications run on the external host.
- ROM Bootloader: The CC1310 provides a ROM bootloader which can be used to program the flash memory. The out-of-box CoP is configured to enable the bootloader if the “backdoor” DIO pin is active low when the device is reset. See Table 1 for bootloader pin configurations on the CC1310.

1.1 *Network Processor Signals*

The figure below shows how an application processor interfaces with the CC1310 TI-15.4-STACK-CoP.

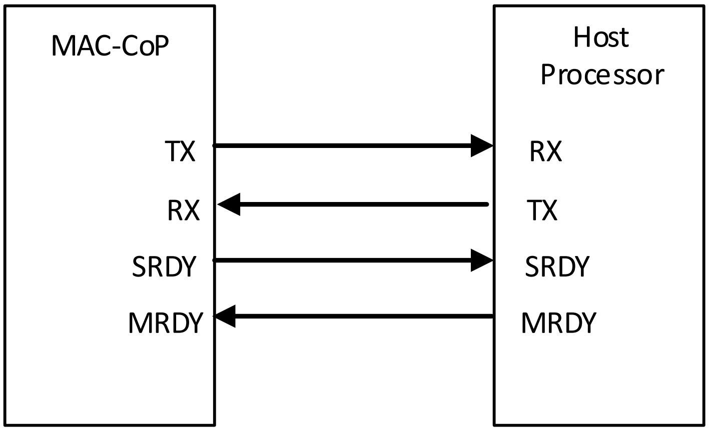


Figure 1: CC1310 Interface

The CC1310-TI-15.4-STACK-CoP uses the following signals for the hardware interface.

- RX/TX for UART: These are the standard signals used for UART communication.
  Please refer to [R1] for details.
  NOTE: Hardware-based UART flow-control is currently not supported on the CC1310.
- SRDY: This active low signal is asserted by the CC1310 for power management and transaction control. The application processor can use a regular GPIO pin to poll the status of this signal, or connect it to a GPIO with edge configurable interrupt capability. Please refer to [R3] for details.
- MRDY: This active low signal is asserted by the application processor for power management and transaction control. Please refer to [R3] for details.

1.2 *Pin Configuration*

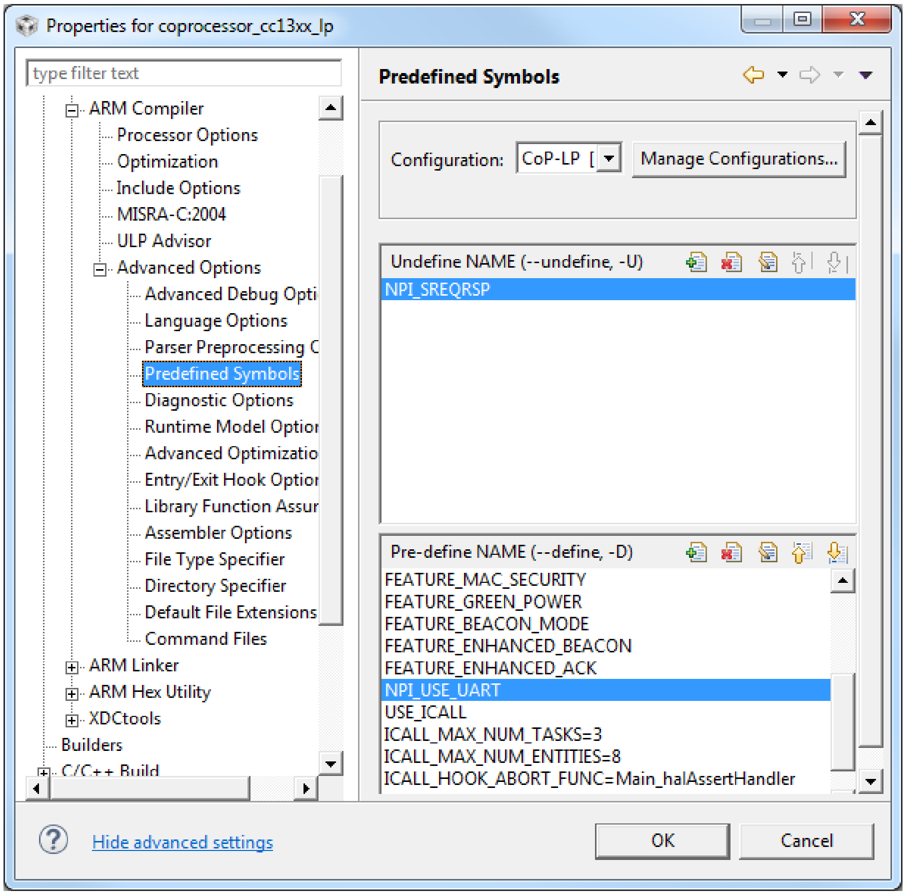
The Pin Configuration for TI-15.4-STACK-CoP on the CC1310 is defined in the following table. Note that TI-15.4-STACK-CoP supports three different package sizes for the CC1310:

Type	TI-15.4-STACK-CoP signal	*Direction (on CC1310)*	CC1310 7x7 PIN	CC1310 5x5 PIN	CC1310 4x4 PIN
POWER_SAVING	SRDY	*Out*	DIO_12	DIO_4	DIO_3
POWER_SAVING	MRDY	*In*	DIO_19	DIO_6	DIO_4
UART	TX	*Out*	DIO_3	DIO_0	DIO_2
UART	RX	*In*	DIO_2	DIO_1	DIO_1
ROM BOOTLOADER	BACKDOOR	*In (low)*	DIO_13	-	-

Table 1: CC1310 Pin Configurations

1.3 *Interface Configuration via CCS Project*

The CCS project in the CC1310 TI-15.4-STACK-CoP SDK supports UART for network processor to host connectivity. Navigate to: ***Project->Properties->ARM Compiler->Advanced Options->Predefined Symbols***



Configuration: CoP-LP [

Undefine NAME (--undefine, -U)

NPI_SREQRSP

Pre-define NAME (--define, -D)

FEATURE_MAC_SECURITY
FEATURE_GREEN_POWER
FEATURE_BEACON_MODE
FEATURE_ENHANCED_BEACON
FEATURE_ENHANCED_ACK
NPI_USE_UART
USE_ICALL
ICALL_MAX_NUM_TASKS=3
ICALL_MAX_NUM_ENTITIES=8
ICALL_HOOK_ABORT_FUNC=Main_halAssertHandler

Figure 2: TI-15.4-STACK-CoP UART Configuration

2. Serial Communication Interface

2.1 *UART Transport*

2.1.1 Configuration

The following default UART configuration is used:

- Baud rate: 115200
- Hardware (RTS/CTS) flow control.
- 8-N-1 byte format.

2.1.2 Signal Description

The following standard UART signals are used:

- TX: Transmit data.
- RX: Receive data.
- CT: Clear to send.
- RT: Ready to send.

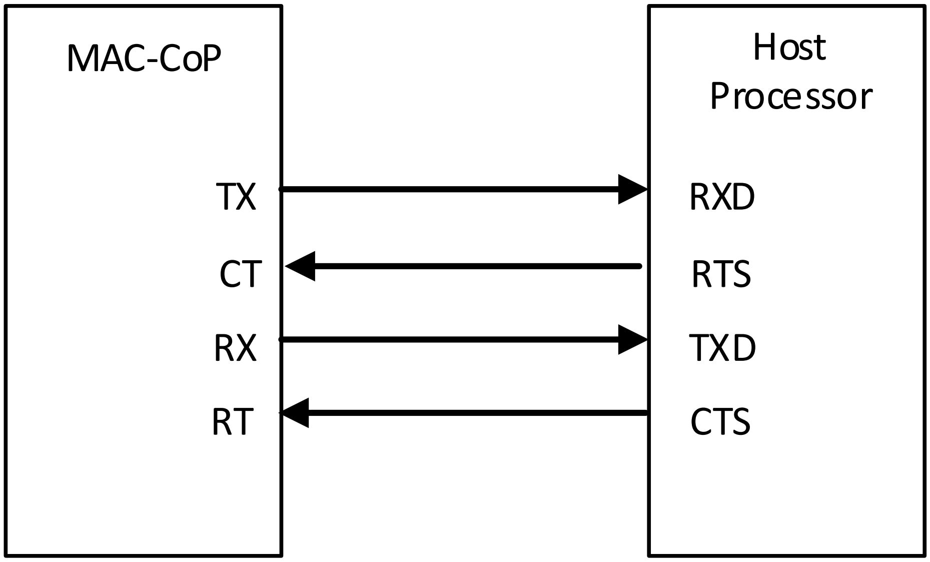


Figure 3: RTS/CTS Flow Control Connections

2.1.3 Signal Operation

UART transport sends and receives data asynchronously. Data can be sent and received simultaneously and transfer of frames can be initiated at any time by either the host processor or the Co-Processor.

2.1.4 Transport Frame Format

The UART transport frame format is shown in the following figure. The left-most field is transmitted first over the wire. As shown, valid frames can range from 5 to 255 bytes in length, depending on the length of the general frame format, which is detailed later in this document.

Bytes: **1**	**3-253**	**1**
SOF	Monitor/Test Frame Format	FCS

Figure 4: UART Transport Frame Format

SOF: Start of frame indicator, which is always set to 0xFE.

Monitor/Test frame format: This is the MT frame format as described in section 2.2.

FCS: Frame check sequence, computed as an XOR of all the bytes in the general format frame field.

Here is example “C” code for the FCS calculation:

```
unsigned char calcFCS(unsigned char *pMsg, unsigned char len)
{
  unsigned char result = 0;
  while(len--)
  {
    result ^= *pMsg++;
  }
  return(result);
}
```

2.2 *Monitor and Test Frame Formats*

The TI-15.4-STACK-CoP interface defines two different types of Monitor and Test (MT) frames used to transfer commands and data between Host and CoP devices. MT frames, designated as Standard or Extended, occupy the General Format Frame portion of a UART Transport Frame, described above. Standard MT frames typically are used when the command and data block can be sent in one serial transaction. Extended MT frames are used when fragmentation is required to transfer larger data blocks.

Both of these frame formats start with a 3-byte *Header* field, consisting of an 8-bit length byte, followed by 8-bit *CMD0* and *CMD1* command bytes. *CMD0* contains the command type and the MT sub-system, and *CMD1* provides an 8-bit command ID for that specific sub-system. Extended MT frames follow the 3-byte *Header* with a variable length *Extended Header* field, from 1 to 4 bytes in length. After the *Header* bytes, a variable length *Data* field may be appended to form a complete MT frame of up to 250 bytes.

Header and Data elements are packed on consecutive one-byte boundaries – there is no padding between elements of different sizes and data types. For multi-byte elements, the lowest order byte is buffered first. For example, a 16-bit value will have its least significant byte (LSB) sent first, followed by its most significant byte (MSB). As shown in the following sections, a valid *Data* block can range from 0 to 250 bytes in length, depending on the specific command and the type of MT frame in use.

2.2.1 Standard MT Frame Format

The standard MT frame format consists of the 3-byte MT header and an optional data field of up to 250 bytes. Note that the upper bit (bit 7) of *CMD0* is set to zero in this format. The *Len* element of the MT header indicates the number of bytes in the *DATA* part of the frame.

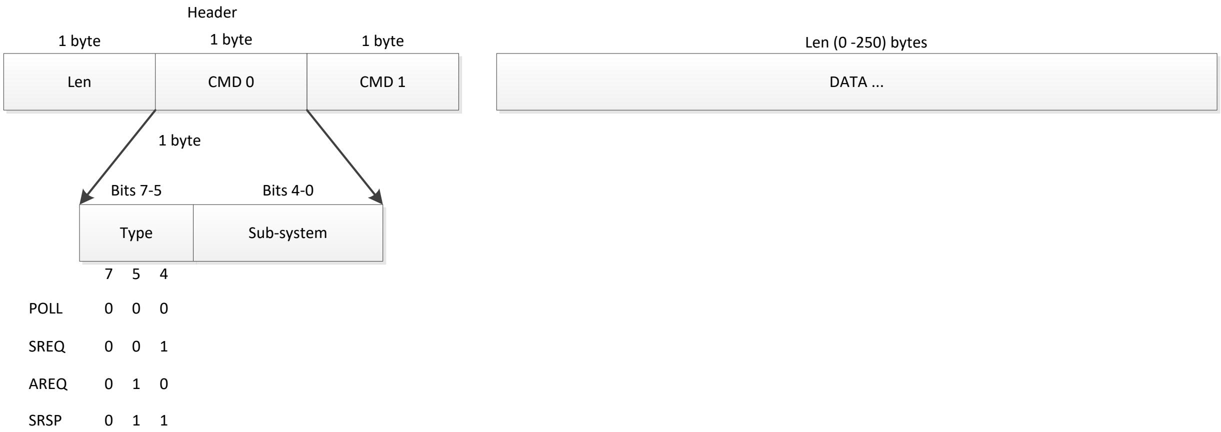


Figure 5: Standard MT Frame Format

2.2.2 Extended MT Frame Format

The extended MT frame format consists of the 3-byte MT header, a variable length "Extended Header", and an optional data field of up to 246 bytes. Note that the upper bit (bit 7) of *CMD0* is set to one in this format, designated as EXTN below. The *Len* element of the MT header indicates the number of bytes in the "Extended Header" and the *DATA* part of the frame.

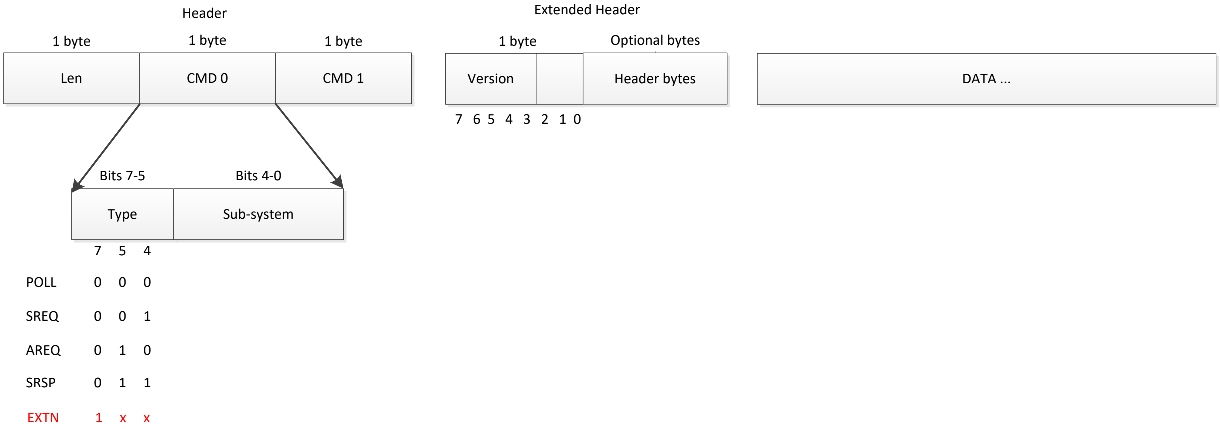


Figure 6: Extended MT Frame Format

2.2.3 MT Command Codes

The command codes consist of two bytes, “Cmd0” and “Cmd1”, as illustrated in the following figure. “Cmd0” encodes the command *Type* in bits[7:5] and the MT *Subsystem* in bits[4:0]. “Cmd1” provides the 8-bit command ID code for the specified *Subsystem*. The “Cmd0” byte is transmitted first.

Cmd0	
Bits: 7-5	**4-0**
Type	**Subsystem**

Cmd1
7-0
ID

Figure 7: MT Command Codes

The following table lists the 3-bit Cmd0 *Type* codes for Standard and Extended MT frames:

Standard	Extended	Description
0	4	POLL: Not used in the TI-15.4-STACK-CoP
1	5	SREQ: A synchronous request that requires an immediate response. For example, a function call with a return value would use an SREQ command.
2	6	AREQ: An asynchronous request that does not require an immediate response. For example, a function call with no return value or a callback event would use an AREQ command.
3	7	SRSP: A synchronous response. This type of command is only sent in response to an SREQ command. For an SRSP command, the subsystem and ID codes are set to the same values as the corresponding SREQ. The length of an SRSP is generally non-zero, so an SRSP with length=0 can be used to indicate an error.

Table 2: Cmd0 *Type* Codes

The following table lists the available and reserved 5-bit Cmd0 *Subsystem* codes for all MT frames:

Subsystem Code	Subsystem Name
0	RPC Error
1	SYS interface
2	MAC Interface
3-6	Reserved
7	UTIL interface
8-31	Reserved

Table 3: Cmd0 *Subsystem* Codes

Cmd1 provides an 8-bit command *ID* code which maps to a specific interface message for the *Subsystem* specified in Cmd0. Therefore, each MT subsystem can provide up 256 message handling functions.

2.2.3.1 MT Command Error

When an SREQ command from the Host is not recognized by the TI-15.4-STACK Co-Processor, an 'error' SRSP is returned, detailed in the two tables below. The formats of the tables are representative of all other MT commands and responses that are presented in this document. The shaded upper row of the SRSP byte stream indicates the size (bytes) of each element. The lower row provides the title of each element, always starting with the 3-byte MT header at the left, followed by any *Data* elements (in this case *ErrorCode*, *ReqCmd0,* and *ReqCmd1*). The table of *Attributes* shows information for each element in the *Data* part of the byte stream.

SRSP:

1	1	1	1	1	1
Length = 0x03	Cmd0 = 0x60	Cmd1 = 0x00	ErrorCode	ReqCmd0	ReqCmd1

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td>ErrorCode</td><td>1</td><td>Error code to indicate reason for command failure:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x01</td><td>Invalid subsystem</td></tr>
<tr><td>0x02</td><td>Invalid command ID</td></tr>
<tr><td>0x03</td><td>Invalid parameter</td></tr>
<tr><td>0x04</td><td>Invalid length</td></tr>
<tr><td>0x05</td><td>Unsupported extended header type</td></tr>
<tr><td>0x06</td><td>Memory allocation failure</td></tr>
</table></td></tr>
<tr><td>ReqCmd0</td><td>1</td><td>Cmd0 value of the processed SREQ</td></tr>
<tr><td>ReqCmd1</td><td>1</td><td>Cmd1 value of the processed SREQ</td></tr>
</table>

2.2.4 MT Extended Frames

This section details the MT Extended frames that are provided by the TI-15.4-STACK-CoP. Each of these frames is identified by the unique 5-bit *Version* field in the first byte of its "Extended Header". This means that parsing of an extended frame must start with analysis of the 4th byte in the MT frame, since the *Version* field of that byte indicates the structure of the "Extended Header" and any following *Data*.

Version Description	Value
Not Used	0
Stack ID	1
Fragmentation Data Packet	2
Fragmentation Acknowledgment	3
Extended Frame Status	4
Available - new version formats	5-30
Reserved – version field extension	31

Table 4: Extended Frame Versions

2.2.4.1 Stack ID Frame (Version = 1)

The Stack ID frame is an MT extension to permit support of multiple 802.15.4-based protocol stacks by a single TI-15.4-STACK-CoP. The figure below shows the 1-byte Extended Header field for Stack ID frames.

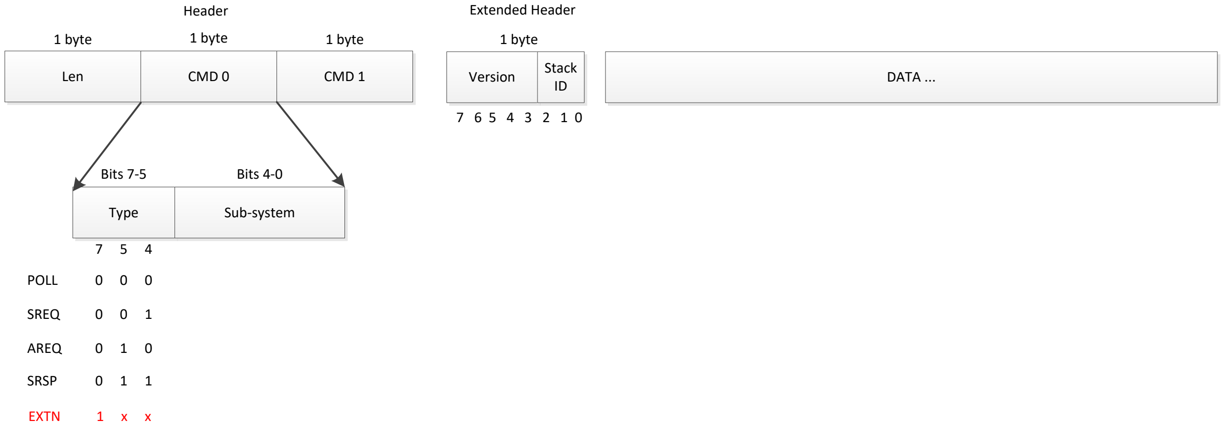


Figure 8: Stack ID Frame Format

2.2.4.1.1 Stack ID field

The stack ID field is the lower 3 bits of the Extended Header. The stack ID indicates which host stack process issued the MT message or for which stack process the incoming MT response message is sent to. The stack ID field values can range from 0 to 7.

2.2.4.2 Fragmentation Frame (Version = 2)

The Fragmentation frame is an MT extension to support transfer of message packets that exceed the length allowed for a single MT frame. The figure below shows the 4-byte Extended Header field for Fragmentation frames. Transfer of fragmentation frames involves a handshake sequence where each transmitted fragment packet must be acknowledged (Ack Frame) by the receiving device. Therefore, only one fragmentation process can be active, in each direction, at any given time.

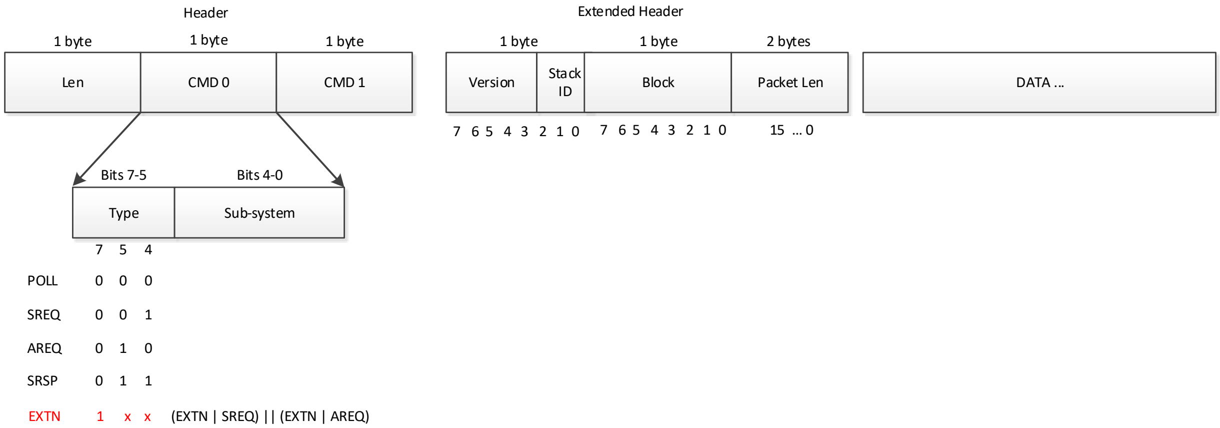


Figure 9: Fragmentation Frame Format

2.2.4.2.1 Stack ID field

The stack ID field is the lower 3 bits of the Extended Header. The stack ID indicates which host stack process issued the MT message or for which stack process the incoming MT response message is sent to. The stack ID field values can range from 0 to 7. Not used by the CoP – set to a value of zero.

2.2.4.2.2 Block field

Large packets are divided into equal length blocks (except the last block), then each block is sent, in a fragmented packet. This field is the corresponding block number. The first fragment (block 0) sets the block length, which must be maintained until the last block. The block length is arbitrary - defined by the application programmer, but the maximum block length is 246 bytes (max MT frame *Len* is 250, minus the extended fragmentation header of 4 bytes).

Example:

- Suppose: a long data packet has 1100 bytes (*Packet Len* field = 1100),
- Programmer choice: transfer the data packet in 128-byte fragments,
- Send 9 fragments (128, 128, 128, 128, 128, 128, 128, 128, and 76 bytes),
- The *Block* field in these transfers starts with a value of 0 and ends with 8

2.2.4.2.3 Packet Len field

The *Packet Len* is a 16-bit field and represents the length of the entire *Data* field when the fragmented packets are reassembled by the receiver.

2.2.4.3 Fragmentation Ack Frame (Version = 3)

Each received fragmentation packet must be acknowledged by an Ack Frame to start the transfer of the next packet. The figure below shows the 3-byte Extended Header field for Fragmentation Ack frames. The CMD0 *Type* field will be either (EXTN | SRSP) for an Ack sent in response to an SREQ message or (EXTN | AREQ) for response to AREQ message (there isn't an ARSP type).

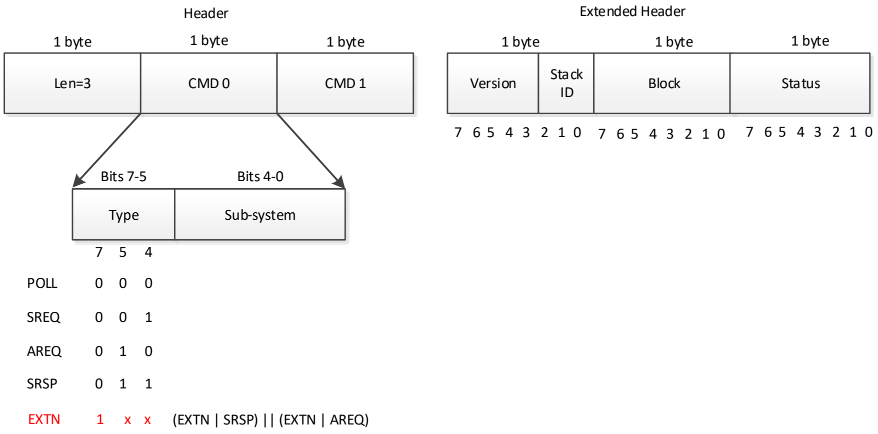


Figure 10: Fragmentation Ack Frame Format

2.2.4.3.1 Stack ID field

This field is a copy of the *Stack ID* field from the received fragmented packet.

2.2.4.3.2 Block field

This field is a copy of the *Block* field from the received fragmented packet.

2.2.4.3.3 Status field

This field returns the status of the fragmented packet reception, with one of the following values:

Status Description	Value
Success	0
Request - resend last frame	1
Unsupported Stack ID	2
Block out of order – fragmentation aborted	3
Block length changed – fragmentation aborted	4
Memory allocation error – fragmentation aborted	5
Fragmentation sequence completed	6

Table 5: Fragmentation Ack Status Values

2.2.4.4 Extended Status Frame (Version = 4)

Extended frame handling may result in a situation where status should be provided to indicate what happened. For example, a Host processor could be informed by the TI-15.4-STACK-CoP of dropped incoming message (possibly due to a memory allocation failure). The figure below shows the 3-byte Extended Header field for Extended Status frames.

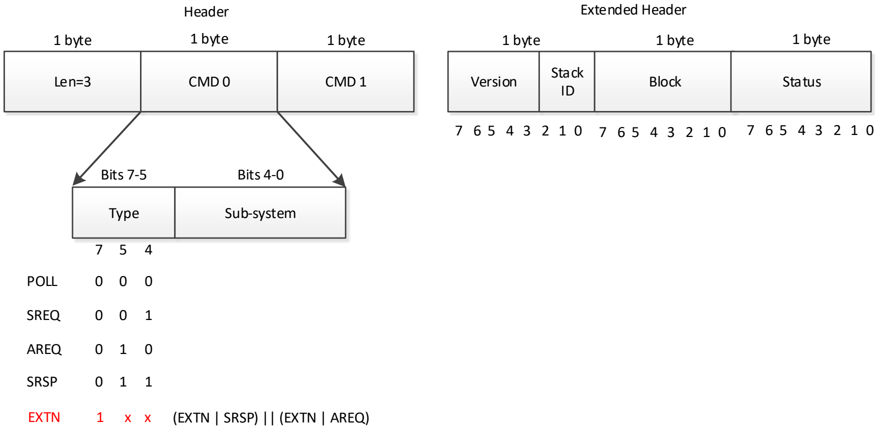


Figure 11: Extended Status Frame Format

2.2.4.4.1 Stack ID field

The *Stack ID* field is the lower 3 bits of the Extended Header. The *Stack ID* indicates which host stack process issued the MT message or for which stack process the incoming MT response message is sent to. The *Stack ID* field values can range from 0 to 7. Not used by the CoP – set to a value of zero.

2.2.4.4.2 Block field

This field is a copy of the *Block* field from the fragmented packet that 'caused' the error condition. For incoming messages that could not initiate a transfer to the Host, the *Block* will be set to zero.

2.2.4.4.3 Status field

This field returns the status of MT command/data transfer operation, with one of the following values:

Status Description	Value
Memory allocation error	5
Fragmentation sequence completed	6
Fragmentation sequence aborted	7
Unsupported Fragmentation Ack Status	8

Table 6: Extended Status Values

3. TI-15.4-STACK-CoP Software Command Interface

The TI-15.4-STACK Co-Processor software command interface consists of APIs from three MT sub-systems. The MT MAC sub-system provides commands and callbacks for RF network communication. The MT SYS and MT UTIL sub-systems provide support functionality for robust Co-Processor operation. The APIs allow developers to implement various functionalities for deploying an IEEE 802.15.4 based network using a host controlling the TI-15.4-STACK-CoP. The sections below list the API calls for each MT sub-system. Note that usage diagrams in this section depict Standard MT frames (section 2.2.1) but all of the messages can be used with Extended MT frames (section 2.2.2) as well. Normally, Extended MT frames are only used when parameters and data for a command/response message exceeds 250 bytes.

3.1 MT MAC Initialization Interface

Initialization Interface is used to configure the MAC with default MAC PIB values. Additional features are enabled by using the APIs in data or management interface.

3.1.1 MAC_INIT

Description:

This command initialized the MAC subsystem in legacy MAC-CoP implementations. It was called once when the software system was started and before any other MAC API is called. NOTE: In current CoP implementations, this command is executed automatically on startup, so the Host application is not required to use it.

Usage:

SREQ:

1	1	1
Length = 0x00	Cmd0 = 0x22	Cmd1 = 0x02

Attributes: None

SRSP:

1	1	1	1
Length = 0x01	Cmd0 = 0x62	Cmd1 = 0x02	Status

Attributes:

Attribute	Length	Description
Status	1	Status of MAC_INIT message delivery. Refer to Section 6.1 for enumerated list of status values.

3.2 MT MAC Data Interface

This interface provides APIs to send and receive data between Application and the TI-15.4-STACK-CoP.

3.2.1 MAC_DATA_REQ

Description:

This API is used to send application data to the TI-15.4-STACK-CoP for transmission.

The TI-15.4-STACK-CoP can only buffer a certain number of data request frames. When the MAC is congested and cannot accept the data request it sends a MAC_DATA_CNF with status MAC_TRANSACTION_OVERFLOW. Eventually the MAC will become uncongested and send a MAC_DATA_CNF for a buffered request. At this point the application can attempt another data request. Using this scheme, the application can send data whenever it wants but it must queue data to be resent if it receives an overflow status.

The MAC_DATA_REQ allocates transmit data from the heap memory. When the transmit data length is greater than 446 bytes, it may become difficult to allocated memory due to heap memory fragmentation. Therefore, *DataPayload* greater than 446 should be avoided.

Usage:

SREQ:

Byte: 1	1	1	1	8	2	1
Length = 0x23-0xFF	Cmd0 = 0x22	Cmd1 = 0x05	DestAddressMode	DestAddress	DestPanId	SrcAddressMode

1	1	1	1	8	1	1	1
Handle	TxOption	Channel	Power	KeySource	SecurityLevel	KeyIdMode	KeyIndex

4	2	2	DataLength	IELength
IncludeFhIEs	DataLength	IELength	DataPayload	IEPayload

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td>DestAddressMode</td><td>1</td><td>Specifies the format of the destination address.
<table><tr><th>Mode</th><th>Value</th><th>Description</th></tr><tr><td>ADDRESS_16_BIT</td><td>0x02</td><td>Address 16 bit</td></tr><tr><td>ADDRESS_64_BIT</td><td>0x03</td><td>Address 64 bit</td></tr></table></td></tr>
<tr><td>DestAddress</td><td>8</td><td>Address of the destination.</td></tr>
<tr><td>DestPanId</td><td>2</td><td>PAN Id of the destination.</td></tr>
<tr><td>SrcAddressMode</td><td>1</td><td>Specifies the format of the source address.
<table><tr><th>Mode</th><th>Value</th><th>Description</th></tr><tr><td>ADDRESS_16_BIT</td><td>0x02</td><td>Address 16 bit</td></tr><tr><td>ADDRESS_64_BIT</td><td>0x03</td><td>Address 64 bit</td></tr></table></td></tr>
<tr><td>Handle</td><td>1</td><td>Application-defined handle value associated with this data request.</td></tr>
</table>

<table>
<tr><td><b>TxOption</b></td><td>1</td><td><b>Transmitting options:</b>
<table>
<tr><th>Option</th><th>Value</th><th>Description</th></tr>
<tr><td>MAC_TXOPTION_NOACK</td><td>0x00</td><td>Non -acknowledged transmission.</td></tr>
<tr><td>MAC_TXOPTION_ACK</td><td>0x01</td><td>Acknowledged transmission. The MAC will attempt to retransmit the frame until it is acknowledged</td></tr>
<tr><td>MAC_TXOPTION_GTS</td><td>0x02</td><td>GTS transmission (unused)</td></tr>
<tr><td>MAC_TXOPTION_INDIRECT</td><td>0x04</td><td>Indirect transmission. The MAC will queue the data and wait for the destination device to poll for it. This can only be used by a coordinator device</td></tr>
<tr><td>MAC_TXOPTION_PEND_BIT</td><td>0x08</td><td>Force setting of pending bit for direct transmission</td></tr>
<tr><td>MAC_TXOPTION_NO_RETRANS</td><td>0x10</td><td>This proprietary option prevents the frame from being retransmitted</td></tr>
<tr><td>MAC_TXOPTION_NO_CNF</td><td>0x20</td><td>This proprietary option prevents a MAC_DATA_CNF
event from being sent for this frame</td></tr>
<tr><td>MAC_TXOPTION_ALT_BE</td><td>0x40</td><td>Use PIB value MAC_ALT_BE for the minimum backoff exponent</td></tr>
<tr><td>MAC_TXOPTION_PWR_CHAN</td><td>0x80</td><td>Use the power and channel values in macDataReq_t instead of the PIB values</td></tr>
</table></td></tr>
<tr><td><b>Channel</b></td><td>1</td><td><b>Transmit the data frame on this channel. This value is ignored if TxOption MAC_TXOPTION_PWR_CHAN is not used.</b></td></tr>
<tr><td><b>Power</b></td><td>1</td><td><b>Transmit the data frame at this power level. This value is ignored if TxOption MAC_TXOPTION_PWR_CHAN is not used.</b></td></tr>
<tr><td><b>KeySource</b></td><td>8</td><td><b>Key Source of this data frame.</b></td></tr>
<tr><td><b>SecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b>
<table>
<tr><th>Security Level</th><th>Value</th></tr>
<tr><td>NO_SECURITY</td><td>0x00</td></tr>
<tr><td>MIC_32_AUTH</td><td>0x01</td></tr>
<tr><td>MIC_64_AUTH</td><td>0x02</td></tr>
<tr><td>MIC_128_AUTH</td><td>0x03</td></tr>
<tr><td>AES_ENCRYPTION</td><td>0x04</td></tr>
<tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr>
<tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr>
<tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr>
</table></td></tr>
<tr><td><b>KeyIdMode</b></td><td>1</td><td><b>Key Id Mode of this data frame:</b>
<table>
<tr><th>Key Id Mode</th><th>Value</th></tr>
<tr><td>NOT_USED</td><td>0x00</td></tr>
<tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr>
<tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr>
<tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr>
</table></td></tr>
<tr><td><b>KeyIndex</b></td><td>1</td><td><b>Key Index of this data frame.</b></td></tr>
<tr><td><b>IncludeFhIEs</b></td><td>4</td><td><b>Bitmap to indicate which frequency hopping IEs to include:</b>
<table>
<tr><th>Frequency hopping IE bits</th><th>Value</th></tr>
<tr><td>MAC_FH_UTT_IE</td><td>0x00000002</td></tr>
<tr><td>MAC_FH_BT_IE</td><td>0x00000008</td></tr>
<tr><td>MAC_FH_US_IE</td><td>0x00010000</td></tr>
<tr><td>MAC_FH_BS_IE</td><td>0x00020000</td></tr>
</table></td></tr>
<tr><td><b>DataLength</b></td><td>2</td><td><b>Length of the data payload (DL)</b></td></tr>
<tr><td><b>IELength</b></td><td>2</td><td><b>Length of IE payload (PL)</b></td></tr>
<tr><td><b>DataPayload</b></td><td>DL</td><td><b>Actual data payload that will be sent</b></td></tr>
<tr><td><b>IEPayload</b></td><td>PL</td><td><b>Actual IE payload list that will be sent</b></td></tr>
</table>

SRSP:

Byte: 1	1	1	1

Length = 0x01	**Cmd0 = 0x62**	**Cmd1 = 0x05**	**Status**

Attributes:

Attribute	Length	Description
Status	**1**	**Status of DATA_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values.**

3.2.2 MAC_PURGE_REQ

Description:

This API is used to send a request the purge of a data frame from the TI-15.4-STACK-CoP data Queue.

Usage:

SREQ:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x22**	**Cmd1 = 0x0E**	**Handle**

Attributes:

Attribute	Length	Description
Handle	**1**	**The application-defined handle value associated with the data request**

SRSP:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x62**	**Cmd1 = 0x0E**	**Status**

Attributes:

Attribute	Length	Description
Status	**1**	**Status of PURGE_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values.**

3.2.3 MAC_DATA_CNF

Description:

This command is sent by the TI-15.4-STACK-CoP to the host application after it receives MAC_DATA_REQ. For each MAC_DATA_REQ a MAC_DATA_CNF is always returned. If the MAC is congested and cannot buffer any more frames, then it will return with status of MAC_TRANSACTION_OVERFLOW. Else it will return with success if the MAC data transmission was successful or an error status value will indicate the reason for failure.

Usage:

AREQ:

1	1	1	1	1	4	2
Length = 0x10	**Cmd0 = 0x42**	**Cmd1 = 0x84**	**Status**	**Handle**	**Timestamp**	**Timestamp2**

1	1	1	1	4
Retries	**LinkQuality**	**Correlation**	**RSSI**	**FrameCounter**

Attributes:

Attribute	Length (byte)	Description
Status	1	**Status of the MAC_DATA_REQ operation.** **Refer to Section 6.1 for enumerated list of status values.**
Handle	1	**Application-defined handle value associated with the data request.**
Timestamp	4	**The time, in *aUnitBackoffPeriod* units, at which the frame was transmitted.**
Timestamp2	2	**The time, in internal MAC timer units, at which the frame was transmitted.**

Retries	1	**Number of retries to send a data frame**
LinkQuality	1	**The link quality of the received data frame. The value is based on the energy detect calculation, with values ranging linearly from 0x00 to 0xFF with the higher value indicating higher link quality.**
Correlation	1	**The raw correlation value of the received data frame. This value depends on the radio. See the chip data sheet for details**
RSSI	1	**The received RF power in units of dBm.**
FrameCounter	4	**Frame counter (if any) for the transmitted frame**

3.2.4 MAC_DATA_IND

Description:

This callback message transfers the incoming data from the TI-15.4-STACK-CoP to the application.

Usage:

AREQ:

1	1	1	1	8	1	8	4
Length = 0x33-0xFF	Cmd0 = 0x42	Cmd1 = 0x85	SrcAddrMode	SrcAddr	DstAddrMode	DstAddr	Timestamp

2	2	2	1	1	1	1	8	1
Timestamp2	SrcPanId	DstPanId	LinkQuality	Correlation	RSSI	DSN	KeySource	SecurityLevel

1	1	4	2	2	Datalength	IELength
KeyIdMode	KeyIndex	FrameCounter	DataLength	IELength	DataPayload	IEPayload

Attributes:

<table>
<tr><th>Attribute</th><th>Length (byte)</th><th>Description</th></tr>
<tr><td><b>SrcAddrMode</b></td><td>1</td><td><b>Source address mode</b>
<table>
<tr><th>Mode</th><th>Value</th><th>Description</th></tr>
<tr><td>ADDRESS_16_BIT</td><td>0x02</td><td>Address 16 bit</td></tr>
<tr><td>ADDRESS_64_BIT</td><td>0x03</td><td>Address 64 bit</td></tr>
</table></td></tr>
<tr><td><b>SrcAddr</b></td><td>8</td><td><b>Source address</b></td></tr>
<tr><td><b>DstAddrMode</b></td><td>1</td><td><b>Destination address mode</b></td></tr>
<tr><td><b>DstAddr</b></td><td>8</td><td><b>Destination address</b></td></tr>
<tr><td><b>Timestamp</b></td><td>4</td><td><b>The time, in <i>aUnitBackoffPeriod</i> units, at which the frame was received.</b></td></tr>
<tr><td><b>Timestamp2</b></td><td>2</td><td><b>The time, in internal MAC timer units, at which the frame was received.</b></td></tr>
<tr><td><b>SrcPanId</b></td><td>2</td><td><b>Pan Id of the source address</b></td></tr>
<tr><td><b>DstPanId</b></td><td>2</td><td><b>Pan Id of the destination address</b></td></tr>
<tr><td><b>LinkQuality</b></td><td>1</td><td><b>The link quality of the received data frame. The value is based on the energy detect calculation, with values ranging linearly from 0x00 to 0xFF with the higher value indicating higher link quality.</b></td></tr>
<tr><td><b>Correlation</b></td><td>1</td><td><b>The raw correlation value of the received data frame. This value depends on the radio. See the chip data sheet for details</b></td></tr>
<tr><td><b>RSSI</b></td><td>1</td><td><b>The received RF power in units of dBm.</b></td></tr>
<tr><td><b>DSN</b></td><td>1</td><td><b>Data sequence number of received frame</b></td></tr>
<tr><td><b>KeySource</b></td><td>8</td><td><b>Key Source of this data frame.</b></td></tr>
<tr><td><b>SecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b>
<table>
<tr><th>Security Level</th><th>Value</th></tr>
<tr><td>NO_SECURITY</td><td>0x00</td></tr>
<tr><td>MIC_32_AUTH</td><td>0x01</td></tr>
<tr><td>MIC_64_AUTH</td><td>0x02</td></tr>
<tr><td>MIC_128_AUTH</td><td>0x03</td></tr>
<tr><td>AES_ENCRYPTION</td><td>0x04</td></tr>
<tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr>
<tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr>
<tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr>
</table></td></tr>
</table>

KeyIdMode	1	**Key Id Mode of this data frame:** Key Id Mode | Value NOT_USED | 0x00 KEY_1BYTE_INDEX | 0x01 KEY_4BYTE_INDEX | 0x02 KEY_8BYTE_INDEX | 0x03
KeyIndex	1	**Key Index of this data frame**
FrameCounter	4	**Frame counter (if any) for the received data frame**
DataLength	2	**Length of received data payload (DL)**
IELength	2	**Length of received IE payload (PL)**
DataPayload	DL	**Actual received data payload**
IEPayload	PL	**Actual received IE payload**

3.2.5 MAC_PURGE_CNF

Description:

This callback message sends the status of the MAC_PURGE_REQ to the application.

Usage:

AREQ:

1	1	1	1	1
Length = 0x02	**Cmd0 = 0x42**	**Cmd1 = 0x90**	**Status**	**Handle**

Attributes:

Attribute	Length	Description
Status	1	**Status of PURGE_CNF message delivery.** **Refer to Section 6.1 for enumerated list of status values.**
Handle	1	**Application defined handle of the message**

3.2.6 MAC_WS_ASYNC_IND

Description:

This event is sent to the application when the TI-15.4-STACK-CoP receives a WiSUN async frame indication.

Usage:

AREQ:

1	1	1	1	8	1	8	4
Length = 0x34-0xFF	**Cmd0 = 0x42**	**Cmd1 = 0x93**	**SrcAddrMode**	**SrcAddr**	**DstAddrMode**	**DstAddr**	**Timestamp**

2	2	2	1	1	1	1	8	1
Timestamp2	**SrcPanId**	**DstPanId**	**LinkQuality**	**Correlation**	**RSSI**	**DSN**	**KeySource**	**SecurityLevel**

1	1	4	1	2	2	Datalength	IELength
KeyIdMode	**KeyIndex**	**FrameCounter**	**FrameType**	**DataLength**	**IELength**	**DataPayload**	**IEPayload**

Attributes:

Attribute	Length (byte)	Description

<table>
<tr><td><b>SrcAddrMode</b></td><td>1</td><td><b>Source address mode</b>
<table>
<tr><th>Mode</th><th>Value</th><th>Description</th></tr>
<tr><td>ADDRESS_16_BIT</td><td>0x02</td><td>Address 16 bit</td></tr>
<tr><td>ADDRESS_64_BIT</td><td>0x03</td><td>Address 64 bit</td></tr>
</table></td></tr>
<tr><td><b>SrcAddr</b></td><td>8</td><td><b>Source address</b></td></tr>
<tr><td><b>DstAddrMode</b></td><td>1</td><td><b>Destination address mode</b></td></tr>
<tr><td><b>DstAddr</b></td><td>8</td><td><b>Destination address</b></td></tr>
<tr><td><b>Timestamp</b></td><td>4</td><td><b>The time, in <i>aUnitBackoffPeriod</i> units, at which the frame was received.</b></td></tr>
<tr><td><b>Timestamp2</b></td><td>2</td><td><b>The time, in internal MAC timer units, at which the frame was received.</b></td></tr>
<tr><td><b>SrcPanId</b></td><td>2</td><td><b>Pan Id of the source address</b></td></tr>
<tr><td><b>DstPanId</b></td><td>2</td><td><b>Pan Id of the destination address</b></td></tr>
<tr><td><b>LinkQuality</b></td><td>1</td><td><b>The link quality of the received data frame. The value is based on the energy detect calculation, with values ranging linearly from 0x00 to 0xFF with the higher value indicating higher link quality.</b></td></tr>
<tr><td><b>Correlation</b></td><td>1</td><td><b>The raw correlation value of the received data frame. This value depends on the radio. See the chip data sheet for details</b></td></tr>
<tr><td><b>RSSI</b></td><td>1</td><td><b>The received RF power in units of dBm.</b></td></tr>
<tr><td><b>DSN</b></td><td>1</td><td><b>Data sequence number of received frame</b></td></tr>
<tr><td><b>KeySource</b></td><td>8</td><td><b>Key Source of this data frame.</b></td></tr>
<tr><td><b>SecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b>
<table>
<tr><th>Security Level</th><th>Value</th></tr>
<tr><td>NO_SECURITY</td><td>0x00</td></tr>
<tr><td>MIC_32_AUTH</td><td>0x01</td></tr>
<tr><td>MIC_64_AUTH</td><td>0x02</td></tr>
<tr><td>MIC_128_AUTH</td><td>0x03</td></tr>
<tr><td>AES_ENCRYPTION</td><td>0x04</td></tr>
<tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr>
<tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr>
<tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr>
</table></td></tr>
<tr><td><b>KeyIdMode</b></td><td>1</td><td><b>Key Id Mode of this data frame:</b>
<table>
<tr><th>Key Id Mode</th><th>Value</th></tr>
<tr><td>NOT_USED</td><td>0x00</td></tr>
<tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr>
<tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr>
<tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr>
</table></td></tr>
<tr><td><b>KeyIndex</b></td><td>1</td><td><b>Key Index of this data frame</b></td></tr>
<tr><td><b>FrameCounter</b></td><td>4</td><td><b>Frame counter (if any) for the received data frame</b></td></tr>
<tr><td><b>FrameType</b></td><td>1</td><td><b>WiSUN Async frame type:</b>
<table>
<tr><th>Async Frame Type</th><th>Value</th></tr>
<tr><td>MAC_WS_ASYNC_PAN_ADVERT</td><td>0x00</td></tr>
<tr><td>MAC_WS_ASYNC_PAN_ADVERT_SOL</td><td>0x01</td></tr>
<tr><td>MAC_WS_ASYNC_PAN_CONFIG</td><td>0x02</td></tr>
<tr><td>MAC_WS_ASYNC_PAN_CONFIG_SOL</td><td>0x03</td></tr>
<tr><td>MAC_WS_ASYNC_DATA</td><td>0x04</td></tr>
<tr><td>MAC_WS_ASYNC_ACK</td><td>0x05</td></tr>
<tr><td>MAC_WS_ASYNC_EAPOL</td><td>0x06</td></tr>
<tr><td>MAC_WS_ASYNC_INVALID</td><td>0xFF</td></tr>
</table></td></tr>
<tr><td><b>DataLength</b></td><td>2</td><td><b>Length of received data payload (DL)</b></td></tr>
<tr><td><b>IELength</b></td><td>2</td><td><b>Length of received IE payload (PL)</b></td></tr>
<tr><td><b>DataPayload</b></td><td>DL</td><td><b>Actual received data payload</b></td></tr>
<tr><td><b>IEPayload</b></td><td>PL</td><td><b>Actual received IE payload</b></td></tr>
</table>

3.3 MT MAC Management Interface

The following APIs are used for 802.15.4 network management.

3.3.1 MAC_ASSOCIATE_REQ

Description:

This API is used to send an associate request to a coordinator device. The application shall attempt to associate only with a PAN that is currently allowing association, as indicated in the results of the scanning procedure. In a beacon-enabled PAN the beacon order must be set by using the API MAC_SET_REQ before making the call to MAC_ASSOCIATE_REQ.

When the associate request is complete the TI-15.4-STACK-CoP sends a MAC_ASSOCIATE_CNF to the application.

Usage:
SREQ:

Byte: 1	1	1	1	1	1	1
Length = 0x1A	Cmd0 = 0x22	Cmd1 = 0x06	LogicalChannel	ChannelPage	PhyId	CoordAddressMode

Byte: 8	2	1	8	1	1	1
CoordAddress	CoordPanId	CapabilityInformation	KeySource	SecurityLevel	KeyIdMode	KeyIndex

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td>LogicalChannel</td><td>1</td><td>Channel on which to attempt association</td></tr>
<tr><td>ChannelPage</td><td>1</td><td>The channel page to be used.</td></tr>
<tr><td>PhyId</td><td>1</td><td>PHY ID for the PHY descriptor to use</td></tr>
<tr><td>CoordAddressMode</td><td>1</td><td>Specifies the format of the coordinator address.
<table>
<tr><th>Mode</th><th>Value</th><th>Description</th></tr>
<tr><td>ADDRESS_16_BIT</td><td>0x02</td><td>Address 16 bit</td></tr>
<tr><td>ADDRESS_64_BIT</td><td>0x03</td><td>Address 64 bit</td></tr>
</table></td></tr>
<tr><td>CoordAddress</td><td>8</td><td>Address of the Coordinator.</td></tr>
<tr><td>CoordPanId</td><td>2</td><td>PAN Id of the Coordinator.</td></tr>
<tr><td>CapabilityInformation</td><td>1</td><td>Bit map which specifies the operational capabilities of the device.
<table>
<tr><th>Capability</th><th>Value</th></tr>
<tr><td>MAC_CAPABLE_PAN_COORD</td><td>0x01</td></tr>
<tr><td>MAC_CAPABLE_FFD</td><td>0x02</td></tr>
<tr><td>MAC_CAPABLE_MAINS_POWER</td><td>0x04</td></tr>
<tr><td>MAC_CAPABLE_RX_ON_IDLE</td><td>0x08</td></tr>
<tr><td>MAC_CAPABLE_SECURITY</td><td>0x40</td></tr>
<tr><td>MAC_CAPABLE_ALLOC_ADDR</td><td>0x80</td></tr>
</table></td></tr>
<tr><td>KeySource</td><td>8</td><td>Key Source of this data frame</td></tr>
</table>

<table>
<tr><td><b>SecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b>
<table>
<tr><th>Security Level</th><th>Value</th></tr>
<tr><td>NO_SECURITY</td><td>0x00</td></tr>
<tr><td>MIC_32_AUTH</td><td>0x01</td></tr>
<tr><td>MIC_64_AUTH</td><td>0x02</td></tr>
<tr><td>MIC_128_AUTH</td><td>0x03</td></tr>
<tr><td>AES_ENCRYPTION</td><td>0x04</td></tr>
<tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr>
<tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr>
<tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr>
</table></td></tr>
<tr><td><b>KeyIdMode</b></td><td>1</td><td><b>Key Id Mode of this data frame:</b>
<table>
<tr><th>Key Id Mode</th><th>Value</th></tr>
<tr><td>NOT_USED</td><td>0x00</td></tr>
<tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr>
<tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr>
<tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr>
</table></td></tr>
<tr><td><b>KeyIndex</b></td><td>1</td><td><b>Key Index of this data frame.</b></td></tr>
</table>

SRSP:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x62**	**Cmd1 = 0x06**	**Status**

Attributes:

Attribute	Length	Description
Status	**1**	**Status of ASSOCIATE_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values.**

3.3.2 MAC_ASSOCIATE_RSP

Description:

This API is used to send an associate response to a device requesting to associate. This API must be used after receiving a MAC_ASSOCIATE_IND. When the associate response is complete the TI-15.4-STACK-CoP sends a MAC_COMM_STATUS_IND to the application to indicate the success or failure of the operation.

Usage:
SREQ:

Byte: 1	1	1	8	2	1
Length = 0x16	**Cmd0 = 0x22**	**Cmd1 = 0x50**	**ExtendedAddress**	**AssocShortAddress**	**AssocStatus**

8	1	1	1
KeySource	**SecurityLevel**	**KeyIdMode**	**KeyIndex**

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>ExtendedAddress</b></td><td>8</td><td><b>Extended Address of the device requesting association</b></td></tr>
<tr><td><b>AssocShortAddress</b></td><td>2</td><td><b>Short address for the associated device. Allocated by the coordinator.</b></td></tr>
<tr><td><b>AssocStatus</b></td><td><b>1</b></td><td><b>Status of the association:</b>
<table>
<tr><th>Status</th><th>Value</th></tr>
<tr><td>SUCCESSFUL_ASSOCIATION</td><td>0x00</td></tr>
<tr><td>PAN_AT_CAPACITY</td><td>0x01</td></tr>
<tr><td>PAN_ACCESS_DENIED</td><td>0x02</td></tr>
</table></td></tr>
</table>

<table>
<tr><td><b>KeySource</b></td><td>8</td><td><b>Key Source of this data frame</b></td></tr>
<tr><td><b>SecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b>
<table>
<tr><th>Security Level</th><th>Value</th></tr>
<tr><td>NO_SECURITY</td><td>0x00</td></tr>
<tr><td>MIC_32_AUTH</td><td>0x01</td></tr>
<tr><td>MIC_64_AUTH</td><td>0x02</td></tr>
<tr><td>MIC_128_AUTH</td><td>0x03</td></tr>
<tr><td>AES_ENCRYPTION</td><td>0x04</td></tr>
<tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr>
<tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr>
<tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr>
</table></td></tr>
<tr><td><b>KeyIdMode</b></td><td>1</td><td><b>Key Id Mode of this data frame:</b>
<table>
<tr><th>Key Id Mode</th><th>Value</th></tr>
<tr><td>NOT_USED</td><td>0x00</td></tr>
<tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr>
<tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr>
<tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr>
</table></td></tr>
<tr><td><b>KeyIndex</b></td><td>1</td><td><b>Key Index of this data frame.</b></td></tr>
</table>

SRSP:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x62**	**Cmd1 = 0x50**	**Status**

Attributes:

Attribute	Length	Description
Status	**1**	**Status of ASSOCIATE_RSP message delivery.** **Refer to Section 6.1 for enumerated list of status values.**

3.3.3 MAC_DISASSOCIATE_REQ

Description:

This API is used by an associated device to notify the coordinator of its intent to leave the PAN. It is also used by the coordinator to instruct an associated device to leave the PAN. When the disassociate procedure is complete the TI-15.4-STACK-CoP sends a MAC_DISASSOCIATE_CNF to the application.

Usage:

SREQ:

Byte: 1	1	1	1	8	2
Length = 0x18	**Cmd0 = 0x22**	**Cmd1 = 0x07**	**DeviceAddressMode**	**DeviceAddress**	**DevicePanId**

1	1	8	1	1	1
DisassociateReason	**TxIndirect**	**KeySource**	**SecurityLevel**	**KeyIdMode**	**KeyIndex**

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>DeviceAddressMode</b></td><td>1</td><td><b>Specifies the format of the device address.</b>
<table>
<tr><th>Mode</th><th>Value</th><th>Description</th></tr>
<tr><td>ADDRESS_16_BIT</td><td>0x02</td><td>Address 16 bit</td></tr>
<tr><td>ADDRESS_64_BIT</td><td>0x03</td><td>Address 64 bit</td></tr>
</table></td></tr>
<tr><td><b>DeviceAddress</b></td><td>8</td><td><b>Device Address.</b></td></tr>
</table>

<table>
<tr><td><b>DevicePanId</b></td><td>2</td><td><b>Network PAN Id of device.</b></td></tr>
<tr><td><b>DisassociateReason</b></td><td>1</td><td><b>Reason of disassociation:</b>
<table>
<tr><th>Reason</th><th>Value</th></tr>
<tr><td>RESERVED</td><td>0x00</td></tr>
<tr><td>COOR_WISHES_DEV_LEAVE</td><td>0x01</td></tr>
<tr><td>DEV_WISHES_LEAVE</td><td>0x02</td></tr>
</table></td></tr>
<tr><td><b>TxIndirect</b></td><td>1</td><td><b>Set to true if the disassociate notification is to be sent indirectly</b></td></tr>
<tr><td><b>KeySource</b></td><td>8</td><td><b>Key Source of this data frame.</b></td></tr>
<tr><td><b>SecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b>
<table>
<tr><th>Security Level</th><th>Value</th></tr>
<tr><td>NO_SECURITY</td><td>0x00</td></tr>
<tr><td>MIC_32_AUTH</td><td>0x01</td></tr>
<tr><td>MIC_64_AUTH</td><td>0x02</td></tr>
<tr><td>MIC_128_AUTH</td><td>0x03</td></tr>
<tr><td>AES_ENCRYPTION</td><td>0x04</td></tr>
<tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr>
<tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr>
<tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr>
</table></td></tr>
<tr><td><b>KeyIdMode</b></td><td>1</td><td><b>Key Id Mode of this data frame:</b>
<table>
<tr><th>Key Id Mode</th><th>Value</th></tr>
<tr><td>NOT_USED</td><td>0x00</td></tr>
<tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr>
<tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr>
<tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr>
</table></td></tr>
<tr><td><b>KeyIndex</b></td><td>1</td><td><b>Key Index of this data frame.</b></td></tr>
</table>

SRSP:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x62**	**Cmd1 = 0x07**	**Status**

Attributes:

Attribute	Length	Description
Status	**1**	**Status of DISASSOCIATE_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values.**

3.3.4 MAC_GET_REQ

Description:

This command is used to read the value of an attribute from the MAC PIB.

Usage:

SREQ:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x22**	**Cmd1 = 0x08**	**AttributeID**

Attributes:

Attribute	Length	Description
AttributeID	**1**	**Specifies the MAC PIB attribute ID** **Refer to Section 6.2 for enumerated list of attribue ID values.**

SRSP:

Byte: 1	1	1	1	16
Length = 0x11	**Cmd0 = 0x62**	**Cmd1 = 0x08**	**Status**	**Data**

Attributes:

Attribute	Length	Description
Status	1	**Status of GET_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values.**
Data	16	**1-16 bytes value of the PIB attribute.**

3.3.5 MAC_SET_REQ

Description:

This command is used to request the TI-15.4-STACK-CoP to write a MAC PIB value.

Usage:

SREQ:

Byte: 1	1	1	1	16
Length = 0x11	Cmd0 = 0x22	Cmd1 = 0x09	AttributeID	AttributeValue

Attributes:

Attribute	Length	Description
AttributeID	1	**Specifies the MAC PIB attribute ID** **Refer to Section 6.2 for enumerated list of attribute ID values.**
AttributeValue	16	**1-16 bytes of the PIB attribute value.**

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x62	Cmd1 = 0x09	Status

Attributes:

Attribute	Length	Description
Status	**1**	**Status of SET_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values.**

3.3.6 MAC_SECURITY_GET_REQ

Description:

This API is used to retrieve a MAC SECURITY PIB value. This command supports 3 types of PIB parameters – single data values, one-dimensional arrays of values, and two-dimensional arrays of values. The Index1 and/or Index2 parameters are ignored when used with PIB attributes that do not use them.

Usage:

SREQ:

Byte: 1	1	1	1	1	1
Length = 0x03	Cmd0 = 0x22	Cmd1 = 0x30	AttributeID	Index1	Index2

Attributes:

Attribute	Length	Description
AttributeID	**1**	**Specifies the Security PIB attribute ID** **Refer to Section 6.3 for enumerated list of attribute ID values.**

<table>
<tr><td><b>Index1</b></td><td><b>2</b></td><td><b>First array index for only the following attributes, ignored otherwise:</b>
<table>
<tr><th>Security PIB Attribute</th><th>Value</th></tr>
<tr><td>MAC_KEY_ID_LOOKUP_ENTRY</td><td>0xD0</td></tr>
<tr><td>MAC_KEY_DEVICE_ENTRY</td><td>0xD1</td></tr>
<tr><td>MAC_KEY_USAGE_ENTRY</td><td>0xD2</td></tr>
<tr><td>MAC_KEY_ENTRY</td><td>0xD3</td></tr>
<tr><td>MAC_DEVICE_ENTRY</td><td>0xD4</td></tr>
<tr><td>MAC_SECURITY_LEVEL_ENTRY</td><td>0xD5</td></tr>
</table></td></tr>
<tr><td><b>Index2</b></td><td><b>2</b></td><td><b>Second array index for only the following attributes, ignored otherwise:</b>
<table>
<tr><th>Security PIB Attribute</th><th>Value</th></tr>
<tr><td>MAC_KEY_ID_LOOKUP_ENTRY</td><td>0xD0</td></tr>
<tr><td>MAC_KEY_DEVICE_ENTRY</td><td>0xD1</td></tr>
<tr><td>MAC_KEY_USAGE_ENTRY</td><td>0xD2</td></tr>
</table></td></tr>
</table>

SRSP:

Byte: 1	1	1	1	1	1	AL
Length = 3+AL	Cmd0 = 0x62	Cmd1 = 0x30	Status	Index1	Index2	Data

AL = Attribute Length

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>Status</b></td><td><b>1</b></td><td><b>Status of SECURITY_GET_REQ message delivery.</b>
<b>Refer to Section 6.1 for enumerated list of status values.</b></td></tr>
<tr><td><b>Index1</b></td><td><b>2*</b></td><td><b>First array index for the following attributes.</b>
<table>
<tr><th>Security PIB Attribute</th><th>Value</th></tr>
<tr><td>MAC_KEY_ID_LOOKUP_ENTRY</td><td>0xD0</td></tr>
<tr><td>MAC_KEY_DEVICE_ENTRY</td><td>0xD1</td></tr>
<tr><td>MAC_KEY_USAGE_ENTRY</td><td>0xD2</td></tr>
<tr><td>MAC_KEY_ENTRY</td><td>0xD3</td></tr>
<tr><td>MAC_DEVICE_ENTRY</td><td>0xD4</td></tr>
<tr><td>MAC_SECURITY_LEVEL_ENTRY</td><td>0xD5</td></tr>
</table>
<b><i>*NOTE: this item should be zero for all other PIB attributes</i></b></td></tr>
<tr><td><b>Index2</b></td><td><b>2*</b></td><td><b>Second array index for the following attributes.</b>
<table>
<tr><th>Security PIB Attribute</th><th>Value</th></tr>
<tr><td>MAC_KEY_ID_LOOKUP_ENTRY</td><td>0xD0</td></tr>
<tr><td>MAC_KEY_DEVICE_ENTRY</td><td>0xD1</td></tr>
<tr><td>MAC_KEY_USAGE_ENTRY</td><td>0xD2</td></tr>
</table>
<b><i>*NOTE: this item should be zero for all other PIB attributes</i></b></td></tr>
<tr><td><b>Data</b></td><td><b>AL</b></td><td><b>1-38 bytes value of the PIB attribute.</b></td></tr>
</table>

3.3.7 MAC_SECURITY_SET_REQ

Description:
This command is used to request the TI-15.4-STACK-CoP to write a MAC SECURITY PIB value.

Usage:

SREQ:

Byte: 1	1	1	1	1	1	AL
Length = 1+AL	Cmd0 = 0x22	Cmd1 = 0x31	AttributeID	Index1	Index2	Attribute Value

AL = Attribute Length

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td>AttributeID</td><td>1</td><td>Specifies the Security PIB attribute ID
Refer to Section 6.3 for enumerated list of attribute ID values.</td></tr>
<tr><td>Index1</td><td>2</td><td>First array index for only the following attributes, ignored otherwise:
<table>
<tr><th>Security PIB Attribute</th><th>Value</th></tr>
<tr><td>MAC_KEY_ID_LOOKUP_ENTRY</td><td>0xD0</td></tr>
<tr><td>MAC_KEY_DEVICE_ENTRY</td><td>0xD1</td></tr>
<tr><td>MAC_KEY_USAGE_ENTRY</td><td>0xD2</td></tr>
<tr><td>MAC_KEY_ENTRY</td><td>0xD3</td></tr>
<tr><td>MAC_DEVICE_ENTRY</td><td>0xD4</td></tr>
<tr><td>MAC_SECURITY_LEVEL_ENTRY</td><td>0xD5</td></tr>
</table></td></tr>
<tr><td>Index2</td><td>2</td><td>Second array index for only the following attributes, ignored otherwise:
<table>
<tr><th>Security PIB Attribute</th><th>Value</th></tr>
<tr><td>MAC_KEY_ID_LOOKUP_ENTRY</td><td>0xD0</td></tr>
<tr><td>MAC_KEY_DEVICE_ENTRY</td><td>0xD1</td></tr>
<tr><td>MAC_KEY_USAGE_ENTRY</td><td>0xD2</td></tr>
</table></td></tr>
<tr><td>AttributeValue</td><td>AL</td><td>1-38 bytes of the SECURITY PIB attribute value.</td></tr>
</table>

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x62	Cmd1 = 0x31	Status

Attributes:

Attribute	Length	Description
Status	1	Status of SECURITY_SET_REQ message delivery. Refer to Section 6.1 for enumerated list of status values.

3.3.8 MAC_UPDATE_PANID_REQ

Description:

This command is used to request the TI-15.4-STACK-CoP to write a new PAN ID to the PIB and device table.

Usage:

SREQ:

Byte: 1	1	1	2
Length = 0x02	Cmd0 = 0x22	Cmd1 = 0x32	PanID

Attributes:

Attribute	Length	Description
PanID	2	New PAN ID for the device

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x62	Cmd1 = 0x32	Status

Attributes:

Attribute	Length	Description
Status	1	Status of UPDATE_PANID_REQ message delivery. Refer to Section 6.1 for enumerated list of status values.

3.3.9 MAC_ADD_DEVICE_REQ

Description:
This command is used to request the TI-15.4-STACK-CoP to add an entry to the MAC device table.

Usage:
SREQ:

Byte: 1	1	1	2	2	8	4
Length = 0x1D	Cmd0 = 0x22	Cmd1 = 0x33	PanID	ShortAddr	ExtAddr	FrameCounter

1	1	1	1	9
Exempt	Unique	Duplicate	DataSize	LookupData

Attributes:

Attribute	Length	Description
PanID	2	PAN ID of the new device
ShortAddr	2	16-bit address of the new device
ExtAddr	8	64-bit address of the new device
FrameCounter	4	Initial frame counter for the new device
Exempt	1	Boolean indicator of whether this device can override the minimum security level setting
Unique	1	Boolean indicator of whether the key is a unique link key
Duplicate	1	Boolean indicator of whether the device entry should be duplicated even for the keys that do not match the key ID lookup data
DataSize	1	Key ID lookup data size as it is stored in PIB: 0=5 bytes, 1=9 bytes
LookupData	9	Key ID lookup data, used to look for the key table entry and create proper key device descriptor for this device

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x62	Cmd1 = 0x33	Status

Attributes:

Attribute	Length	Description
Status	1	Status of ADD_DEVICE_REQ message delivery. Refer to Section 6.1 for enumerated list of status values.

3.3.10 MAC_DELETE_DEVICE_REQ

Description:
This command is used to request the TI-15.4-STACK-CoP to remove an entry from the MAC device table.

Usage:
SREQ:

Byte: 1	1	1	8
Length = 0x08	Cmd0 = 0x22	Cmd1 = 0x34	ExtAddr

Attributes:

Attribute	Length	Description
ExtAddr	8	64-bit address of the device

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x62	Cmd1 = 0x34	Status

Attributes:

Attribute	Length	Description
Status	1	Status of DELETE_DEVICE_REQ message delivery. Refer to Section 6.1 for enumerated list of status values.

3.3.11 MAC_DELETE_ALL_DEVICES_REQ

Description:
This command is used to request the TI-15.4-STACK-CoP to remove all entries from the MAC device table.

Usage:
SREQ:

Byte: 1	1	1
Length = 0x00	Cmd0 = 0x22	Cmd1 = 0x35

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x62	Cmd1 = 0x35	Status

Attributes:

Attribute	Length	Description
Status	1	Status of DELETE_ALL_DEVICES_REQ message delivery. Refer to Section 6.1 for enumerated list of status values.

3.3.12 MAC_DELETE_KEY REQ

Description:
This command is used to request the TI-15.4-STACK-CoP to remove a security key at the specified key index and remove all MAC device table entries associated with that key.

Usage:
SREQ:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x22	Cmd1 = 0x36	Index

Attributes:

Attribute	Length	Description
Index	1	Index of security key to be removed

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x62	Cmd1 = 0x36	Status

Attributes:

Attribute	Length	Description
Status	**1**	**Status of DELETE_KEY_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values.**

3.3.13 MAC_READ_KEY_REQ

Description:
This command is used to request the TI-15.4-STACK-CoP to read the frame counter value associated with a MAC security key specified by the designated key index and the default key source.

Usage:
SREQ:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x22**	**Cmd1 = 0x37**	**Index**

Attributes:

Attribute	Length	Description
Index	**1**	**Index of security key to have its frame counter value returned**

SRSP:

Byte: 1	1	1	1	4
Length = 0x05	**Cmd0 = 0x62**	**Cmd1 = 0x37**	**Status**	**FrameCounter**

Attributes:

Attribute	Length	Description
Status	**1**	**Status of READ_KEY_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values.**
FrameCounter	**4**	**Frame counter value for specified security key**

3.3.14 MAC_WRITE_KEY_REQ

Description:
This command is used to request the TI-15.4-STACK-CoP to add a MAC security key, add the associated lookup data for the key, and initialize the frame counter for the key.

Usage:
SREQ:

Byte: 1	1	1	1	1	16	4	1	9
Length = 0x20	**Cmd0 = 0x22**	**Cmd1 = 0x38**	**New**	**Index**	**Key**	**FrameCounter**	**DataSize**	**LookupData**

Attributes:

Attribute	Length	Description
New	**1**	**Boolean indicator of whether to duplicate the device table entries associated with the previous key and associate them with this new key**
Index	**2**	**Index of the MAC security key table where the key will be written**
Key	**16**	**MAC security key**
FrameCounter	**4**	**Initial frame counter value for new security key**
DataSize	**1**	**Key ID lookup data size as it is stored in PIB: 0=5 bytes, 1=9 bytes**
LookupData	**9**	**Key ID lookup data, used to look for the key table entry and create proper key device descriptor for this device**

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x62	Cmd1 = 0x38	Status

Attributes:

Attribute	Length	Description
Status	1	Status of WRITE_KEY_REQ message delivery. Refer to Section 6.1 for enumerated list of status values.

3.3.15 MAC_ORPHAN_RSP

Description:

This API is called in response to an orphan notification from a peer device. This API must be called after receiving a MAC_ORPHAN_IND. When the orphan response is complete the MAC sends a MAC_COMM_STATUS_IND to the application to indicate the success or failure of the operation.

Usage:

SREQ:

Byte: 1	1	1	8	2	1
Length = 0x016	Cmd0 = 0x22	Cmd1 = 0x51	ExtendedAddress	AssocShortAddress	AssociatedMember

8	1	1	1
KeySource	SecurityLevel	KeyIdMode	KeyIndex

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td>ExtendedAddress</td><td>8</td><td>Extended Address of the device sending the orphan notification</td></tr>
<tr><td>AssocShortAddress</td><td>2</td><td>Short address of the orphan device</td></tr>
<tr><td>AssociatedMember</td><td>1</td><td>TRUE if the orphaning device is an associated member. FALSE otherwise.</td></tr>
<tr><td>KeySource</td><td>8</td><td>Key Source of this data frame</td></tr>
<tr><td>SecurityLevel</td><td>1</td><td>Security Level of this data frame:
<table>
<tr><th>Security Level</th><th>Value</th></tr>
<tr><td>NO_SECURITY</td><td>0x00</td></tr>
<tr><td>MIC_32_AUTH</td><td>0x01</td></tr>
<tr><td>MIC_64_AUTH</td><td>0x02</td></tr>
<tr><td>MIC_128_AUTH</td><td>0x03</td></tr>
<tr><td>AES_ENCRYPTION</td><td>0x04</td></tr>
<tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr>
<tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr>
<tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr>
</table></td></tr>
<tr><td>KeyIdMode</td><td>1</td><td>Key Id Mode of this data frame:
<table>
<tr><th>Key Id Mode</th><th>Value</th></tr>
<tr><td>NOT_USED</td><td>0x00</td></tr>
<tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr>
<tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr>
<tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr>
</table></td></tr>
<tr><td>KeyIndex</td><td>1</td><td>Key Index of this data frame.</td></tr>
</table>

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x62	Cmd1 = 0x51	Status

Attributes:

Attribute	Length	Description
Status	**1**	**Status of ORPHAN_RSP message delivery.** **Refer to Section 6.1 for enumerated list of status values**.

3.3.16 MAC_POLL_REQ

Description:

This API is used to request pending data from the coordinator. When the poll request is complete the MAC sends a MAC_POLL_CNF to the application. If a data frame of nonzero length is received from the coordinator the MAC sends a MAC_POLL_CNF with status MAC_SUCCESS and then sends a MAC_DATA_IND with the data.

Usage:

SREQ:

Byte: 1	1	1	1	8	2
Length = 0x16	**Cmd0 = 0x22**	**Cmd1 = 0x0D**	**CoordAddressMode**	**CoordAddress**	**CoordPanId**

8	1	1	1
KeySource	**SecurityLevel**	**KeyIdMode**	**KeyIndex**

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>CoordAddressMode</b></td><td>1</td><td><table><tr><th>Address Mode</th><th>Value</th><th>Description</th></tr><tr><td>ADDRESS_16_BIT</td><td>0x02</td><td>Address 16 bit</td></tr><tr><td>ADDRESS_64_BIT</td><td>0x03</td><td>Address 64 bit</td></tr></table></td></tr>
<tr><td><b>CoordAddress</b></td><td>8</td><td><b>64-bit Coordinator Address</b></td></tr>
<tr><td><b>CoordPanId</b></td><td>2</td><td><b>Coordinator PanId</b></td></tr>
<tr><td><b>KeySource</b></td><td>8</td><td><b>Key Source of this data frame.</b></td></tr>
<tr><td><b>SecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b><table><tr><th>Security Level</th><th>Value</th></tr><tr><td>NO_SECURITY</td><td>0x00</td></tr><tr><td>MIC_32_AUTH</td><td>0x01</td></tr><tr><td>MIC_64_AUTH</td><td>0x02</td></tr><tr><td>MIC_128_AUTH</td><td>0x03</td></tr><tr><td>AES_ENCRYPTION</td><td>0x04</td></tr><tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr><tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr><tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr></table></td></tr>
<tr><td><b>KeyIdMode</b></td><td>1</td><td><b>Key Id Mode of this data frame:</b><table><tr><th>Key Id Mode</th><th>Value</th></tr><tr><td>NOT_USED</td><td>0x00</td></tr><tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr><tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr><tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr></table></td></tr>
<tr><td><b>KeyIndex</b></td><td><b>1</b></td><td><b>Key Index of this data frame.</b></td></tr>
</table>

SRSP:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x62**	**Cmd1 = 0x0D**	**Status**

Attributes:

Attribute	Length	Description
Status	**1**	**Status of POLL_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values**.

3.3.17 MAC_RESET_REQ

Description:

This command is used to send a MAC Reset command to reset the MAC. This API should be called once at system startup with SetDefault set to TRUE before any other function in the MAC API is called. This sets the MAC PIB to default values (mac_pib.c: see structure macPibDefaults for default PIB values in the TI-15.4-STACK-CoP project).

Usage:

SREQ:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x22	Cmd1 = 0x01	SetDefault

Attributes:

Attribute	Length	Description
SetDefault	**1**	**TRUE – Set the MAC pib values to default values.**

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x62	Cmd1 = 0x01	Status

Attributes:

Attribute	Length	Description
Status	**1**	**Status of RESET_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values**.

3.3.18 MAC_SCAN_REQ

Description:

This API initiate's energy detect, active, passive, or orphan scan on one or more channels. Energy detect scan measures the peak energy on each requested channel. An active scan sends a beacon request on each channel and then listen's for beacons. A passive scan is a receive-only operation that listens for beacons on each channel. An orphan scan is used to locate the coordinator with which the scanning device had previously associated. When a scan operation is complete the MAC sends a MAC_SCAN_CNF to the application.

For active or passive scans, the application sets the maxResults parameter the maximum number of PAN descriptors to return. The MAC will store up to MaxResults PAN descriptors and filter out duplicate beacons. Due to the large number of possible scan channels, the co-processor may limit the actual number of MaxResults to reduce the size of allocated memory. In this case, the host can repeat the request with the returned UnscannedChannels.

An alternative way to get results for an active or passive scan is to set maxResults to zero or set PIB attribute MAC_AUTO_REQUEST to FALSE. Then the MAC will not store results but rather send a MAC_BEACON_NOTIFY_IND for each beacon received. In this scenario, the MAC will not filter out duplicate beacons.

An energy detect, active or passive scan may be performed at any time, if a scan is not already in progress. However a device cannot perform any other MAC management operation or send or receive MAC data until the scan is complete.

Usage:

SREQ:

Byte: 1	1	1	1	1	1	1	1
Length = 0x17-0x1B	Cmd0 = 0x22	Cmd1 = 0x0C	ScanType	ScanDuration	ChannelPage	PhyId	MaxResults

1	1	1	1	1	2	8	1	1
PermitJoin	LinkQuality	RspFilter	MpmScan	MpmType	MpmDuration	KeySource	SecLevel	KeyIdMode

1	17
KeyIndex	Channels

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td>ScanType</td><td>1</td><td>Specifies the scan type:
<table>
<tr><th>Scan Type</th><th>Value</th></tr>
<tr><td>ENERGY_DETECT</td><td>0x00</td></tr>
<tr><td>ACTIVE</td><td>0x01</td></tr>
<tr><td>PASSIVE</td><td>0x02</td></tr>
<tr><td>ORPHAN</td><td>0x03</td></tr>
<tr><td>ACTIVE</td><td>0x05</td></tr>
</table></td></tr>
<tr><td>ScanDuration</td><td>1</td><td>The exponent used in the scan duration calculation. The scan duration is calculated as follows:
scan duration (ms) = (*aBaseSuperframeDuration* ms) * (2^ScanDuration + 1)
Valid range is 0-14.</td></tr>
<tr><td>ChannelPage</td><td>1</td><td>The channel page on which to perform the scan.</td></tr>
<tr><td>PhyId</td><td>1</td><td>PHY identifier indicates which PHY descriptor to use:
<table>
<tr><th>MAC PHY ID</th><th>Value</th></tr>
<tr><td>MAC_STD_US_915_PHY_1</td><td>0x01</td></tr>
<tr><td>MAC_STD_ETSI_863_PHY_3</td><td>0x03</td></tr>
<tr><td>MAC_MRFSK_GENERIC_PHY_ID_BEGIN</td><td>0x04</td></tr>
<tr><td>MAC_MRFSK_GENERIC_PHY_ID_END</td><td>0x06</td></tr>
</table></td></tr>
<tr><td>MaxResults</td><td>1</td><td>The maximum number of PAN descriptor results to return for an active or passive scan. This parameter is ignored for energy detect and orphan scans.</td></tr>
<tr><td>PermitJoin</td><td>1</td><td>Specifies when enhanced beacon response is allowed:
<table>
<tr><th>Beacon Response</th><th>Value</th></tr>
<tr><td>All Beacon Requests</td><td>0x00</td></tr>
<tr><td>Only If Permit Join Is Enabled</td><td>0x01</td></tr>
</table></td></tr>
<tr><td>LinkQuality</td><td>1</td><td>Device will respond to the enhanced beacon request if LinkQuality is equal or higher than this value</td></tr>
<tr><td>RspFilter</td><td>1</td><td>Device will randomly determine whether to respond to the enhanced beacon request based on meeting this probability (0 to 100%)</td></tr>
<tr><td>MpmScan</td><td>1</td><td>Specifies whether MPM scan mode is enabled:
<table>
<tr><th>MPM Scan Mode</th><th>Value</th></tr>
<tr><td>Disabled – use **ScanDuration**</td><td>0x00</td></tr>
<tr><td>Enabled – use **MpmDuration**</td><td>0x01</td></tr>
</table></td></tr>
<tr><td>MpmType</td><td>1</td><td>Specifies the MPM scan type:
<table>
<tr><th>MPM Scan Type</th><th>Value</th></tr>
<tr><td>BPAN (beacon enabled)</td><td>0x01</td></tr>
<tr><td>NBPAN (non-beacon enabled)</td><td>0x02</td></tr>
</table></td></tr>
</table>

<table>
<tr><td><b>MpmDuration</b></td><td>2</td><td><b>Parameter (D) used to compute scan duration:</b><table><tr><th>MPM Scan Type</th><th>Value</th></tr><tr><td>BPAN</td><td>D=1..14: duration = aBaseSuperframeDuration * 2^D</td></tr><tr><td>NBPAN</td><td>D=1..16383: duration = aBaseSlotDuration * D</td></tr></table></td></tr>
<tr><td><b>KeySource</b></td><td>8</td><td><b>Key Source of this data frame.</b></td></tr>
<tr><td><b>SecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b><table><tr><th>Security Level</th><th>Value</th></tr><tr><td>NO_SECURITY</td><td>0x00</td></tr><tr><td>MIC_32_AUTH</td><td>0x01</td></tr><tr><td>MIC_64_AUTH</td><td>0x02</td></tr><tr><td>MIC_128_AUTH</td><td>0x03</td></tr><tr><td>AES_ENCRYPTION</td><td>0x04</td></tr><tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr><tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr><tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr></table></td></tr>
<tr><td><b>KeyIdMode</b></td><td>1</td><td><b>Key Id Mode of this data frame:</b><table><tr><th>Key Id Mode</th><th>Value</th></tr><tr><td>NOT_USED</td><td>0x00</td></tr><tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr><tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr><tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr></table></td></tr>
<tr><td><b>KeyIndex</b></td><td>1</td><td><b>Key Index of this data frame.</b></td></tr>
<tr><td><b>Channels</b></td><td>17</td><td><b>Bit mask of channels to be scanned when starting the device
Trailing 0x00 bytes don’t need to be sent</b></td></tr>
</table>

SRSP:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x62**	**Cmd1 = 0x0C**	**Status**

Attributes:

Attribute	Length	Description
Status	**1**	**Status of SCAN_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values**.

3.3.19 MAC_START_REQ

Description:

This API is called by a coordinator or PAN coordinator to start or reconfigure a network. Before starting a network the device must have set its short address. A PAN coordinator sets the short address by setting the attribute MAC_SHORT_ADDRESS using the API MAC_SET_REQ. A coordinator sets the short address through association.

When the PanCoordinator parameter is TRUE, the MAC automatically sets attributes MAC_PAN_ID and MAC_LOGICAL_CHANNEL to the PanId and LogicalChannel parameters. If PanCoordinator is FALSE, these parameters are ignored (they would be set through association).

The BeaconOrder parameter controls whether the network is beacon-enabled or non beacon-enabled. For a beacon-enabled network, this parameter also controls the beacon transmission interval.

When the operation is complete the MAC sends a MAC_START_CNF to the application.

Usage:

SREQ:

Byte: 1	1	1	4	2	1	1	1
Length = 0x2A+NumIEs	Cmd0 = 0x22	Cmd1 = 0x03	StartTime	PanId	LogicalChannel	ChannelPage	PhyId

1	1	1	1	1	8
BeaconOrder	SuperFrameOrder	PanCoordinator	BatteryLifeExt	CoordRealignment	RealignKeySource

1	1	1	8	1	1
RealignSecurityLevel	RealignKeyIdMode	RealignKeyIndex	BeaconKeySource	BeaconSecurityLevel	BeaconKeyIdMode

1	1	1	1	2	1	NumIEs
BeaconKeyIndex	StartFH	EnhBeaconOrder	OfsTimeSlot	NonBeaconOrder	NumIEs	IEIDList

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td>StartTime</td><td>4</td><td>Time to begin transmitting beacons relative to the received beacon. This parameter is ignored if the device is a PAN coordinator or when starting a non beacon-enabled network. The time is specified in symbol periods and is rounded to the nearest <i>aUnitBackoffPeriod</i> symbol periods.</td></tr>
<tr><td>PanId</td><td>2</td><td>The PAN Id to use. This parameter is ignored if Pan Coordinator is FALSE</td></tr>
<tr><td>LogicalChannel</td><td>1</td><td>The logical channel to use. This parameter is ignored if Pan Coordinator is FALSE</td></tr>
<tr><td>ChannelPage</td><td>1</td><td>The channel page to use. This parameter is ignored if Pan Coordinator is FALSE</td></tr>
<tr><td>PhyId</td><td>1</td><td>PHY identifier indicates which PHY descriptor to use:
<table>
<tr><th>MAC PHY ID</th><th>Value</th></tr>
<tr><td>MAC_STD_US_915_PHY_1</td><td>0x01</td></tr>
<tr><td>MAC_STD_ETSI_863_PHY_3</td><td>0x03</td></tr>
<tr><td>MAC_MRFSK_GENERIC_PHY_ID_BEGIN</td><td>0x04</td></tr>
<tr><td>MAC_MRFSK_GENERIC_PHY_ID_END</td><td>0x06</td></tr>
</table></td></tr>
<tr><td>BeaconOrder</td><td>1</td><td>The exponent used to calculate the beacon interval. The beacon interval is calculated as follows:
interval (ms) = (<i>aBaseSuperframeDuration</i> ms) * 2^BeaconOrder
Valid range is 0-14. For a non beacon-enabled network set to 15.</td></tr>
<tr><td>SuperFrameOrder</td><td>1</td><td>It can also be set to 15 to configure a network that sends a beacon but has no CAP. For a non beacon-enabled network this value is ignored.</td></tr>
<tr><td>PanCoordinator</td><td>1</td><td>Set to TRUE to start a network as PAN coordinator</td></tr>
<tr><td>BatteryLifeExt</td><td>1</td><td>If this value is TRUE, the receiver is disabled after MAC_BATT_LIFE_EXT_PERIODS full backoff periods following the interframe spacing period of the beacon frame. This parameter is ignored for non beacon-enabled networks.</td></tr>
<tr><td>CoordRealignment</td><td>1</td><td>Set to TRUE to transmit a coordinator realignment prior to changing the superframe configuration.</td></tr>
<tr><td>RealignKeySource</td><td>8</td><td>Key Source of this data frame</td></tr>
<tr><td>RealignSecurityLevel</td><td>1</td><td>Security Level of this data frame:
<table>
<tr><th>Security Level</th><th>Value</th></tr>
<tr><td>NO_SECURITY</td><td>0x00</td></tr>
<tr><td>MIC_32_AUTH</td><td>0x01</td></tr>
<tr><td>MIC_64_AUTH</td><td>0x02</td></tr>
<tr><td>MIC_128_AUTH</td><td>0x03</td></tr>
<tr><td>AES_ENCRYPTION</td><td>0x04</td></tr>
<tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr>
<tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr>
<tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr>
</table></td></tr>
<tr><td>RealignKeyIdMode</td><td>1</td><td>Key Id Mode of this data frame:
<table>
<tr><th>Key Id Mode</th><th>Value</th></tr>
<tr><td>NOT_USED</td><td>0x00</td></tr>
<tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr>
<tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr>
<tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr>
</table></td></tr>
<tr><td>RealignKeyIndex</td><td>1</td><td>Key Index of this data frame</td></tr>
<tr><td>BeaconKeySource</td><td>8</td><td>Key Source of this data frame</td></tr>
</table>

<table>
<tr><td><b>BeaconSecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b><table><tr><th>Security Level</th><th>Value</th></tr><tr><td>NO_SECURITY</td><td>0x00</td></tr><tr><td>MIC_32_AUTH</td><td>0x01</td></tr><tr><td>MIC_64_AUTH</td><td>0x02</td></tr><tr><td>MIC_128_AUTH</td><td>0x03</td></tr><tr><td>AES_ENCRYPTION</td><td>0x04</td></tr><tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr><tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr><tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr></table></td></tr>
<tr><td><b>BeaconKeyIdMode</b></td><td>1</td><td><b>Key Id Mode of this data frame</b></td></tr>
<tr><td><b>BeaconKeyIndex</b></td><td>1</td><td><b>Key Index of this data frame</b></td></tr>
<tr><td><b>StartFH</b></td><td>1</td><td><b>Frequency hopping control:</b><table><tr><th>Frequency Hopping</th><th>Value</th></tr><tr><td>DISABLE</td><td>0x00</td></tr><tr><td>ENABLE</td><td>0x01</td></tr></table></td></tr>
<tr><td><b>EnhBeaconOrder</b></td><td>1</td><td><b>Exponent used to calculate the enhanced beacon interval
A value of 15 indicates no enhanced beacon in a beacon enabled PAN</b></td></tr>
<tr><td><b>OfsTimeSlot</b></td><td>1</td><td><b>Time between the enhanced beacon and preceding periodic beacon (supported values: 10-15)</b></td></tr>
<tr><td><b>NonBeaconOrder</b></td><td>2</td><td><b>How often to TX the enhanced beacon in a non-beacon enabled PAN
A value of 16383 indicates no enhanced beacon in a non-beacon enabled PAN</b></td></tr>
<tr><td><b>NumIEs</b></td><td>1</td><td><b>Number of Information Elements in the enhanced beacon (reserved for future use – set to 0 now)</b></td></tr>
<tr><td><b>IEIDList</b></td><td><b>NumIEs</b></td><td><b>List of 8-bit Information Elements in the enhanced beacon (reserved for future use)</b></td></tr>
</table>

SRSP:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x62**	**Cmd1 = 0x03**	**Status**

Attributes:

Attribute	Length	Description
Status	**1**	**Status of START_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values**.

3.3.20 MAC_SYNC_REQ

Description:

This API requests the TI-15.4-STACK-CoP to synchronize with the coordinator by acquiring and optionally tracking its beacons. Synchronizing with the coordinator is recommended before associating in a beacon-enabled network. If the beacon could not be located on its initial search or during tracking, the MAC sends a MAC_SYNC_LOSS_IND to the application with status MAC_BEACON_LOSS.

Before calling this API the application must set PIB attributes MAC_BEACON_ORDER, MAC_PAN_ID and either MAC_COORD_SHORT_ADDRESS or MAC_COORD_EXTENDED_ADDRESS to the address of the coordinator with which to synchronize.

The application may wish to set PIB attribute MAC_AUTO_REQUEST to FALSE before calling this API. Then when the MAC successfully synchronizes with the coordinator it will send the application a MAC_BEACON_NOTIFY_IND. After receiving the event the application may set MAC_AUTO_REQUEST to TRUE to stop receiving beacon notifications.

This API is only applicable to beacon-enabled networks.

Usage:

SREQ:

Byte: 1	1	1	1	1	1	1
Length = 0x04	Cmd0 = 0x22	Cmd1 = 0x04	LogicalChannel	ChannelPage	TrackBeacon	PhyId

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td>LogicalChannel</td><td>1</td><td>The logical channel to use.</td></tr>
<tr><td>ChannelPage</td><td>1</td><td>The channel page to use.</td></tr>
<tr><td>TrackBeacon</td><td>1</td><td>Set to TRUE to continue tracking beacons after synchronizing with the first beacon.
Set to FALSE to only synchronize with the first beacon</td></tr>
<tr><td>PhyId</td><td>1</td><td>PHY identifier to indicate which PHY descriptor to use:
<table>
<tr><th>MAC PHY ID</th><th>Value</th></tr>
<tr><td>MAC_STD_US_915_PHY_1</td><td>0x01</td></tr>
<tr><td>MAC_STD_ETSI_863_PHY_3</td><td>0x03</td></tr>
<tr><td>MAC_MRFSK_GENERIC_PHY_ID_BEGIN</td><td>0x04</td></tr>
<tr><td>MAC_MRFSK_GENERIC_PHY_ID_END</td><td>0x06</td></tr>
</table>
</td></tr>
</table>

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x62	Cmd1 = 0x04	Status

Attributes:

Attribute	Length	Description
Status	1	Status of SYNC_REQ message delivery. Refer to Section 6.1 for enumerated list of status values.

3.3.21 MAC_SET_RX_GAIN_REQ

Description:

This command sends a request to the device to set RX gain when a PA/LNA is used along with TI-15.4-STACK-CoP.

Usage:

SREQ:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x22	Cmd1 = 0x0F	Mode

Attributes:

Attribute	Length	Description
Mode	1	True – Enables high gain mode of the LNA. False – Disables the high gain mode of the LNA.

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x62	Cmd1 = 0x0F	Status

Attributes:

Attribute	Length	Description
Status	1	Status of SET_RX_GAIN_REQ message delivery. Refer to Section 6.1 for enumerated list of status values.

3.3.22 MAC_WS_ASYNC_REQ

Description:

This command is used to send a WiSUN async operation command to the TI-15.4-STACK-CoP.

Usage:

SREQ:

Byte: 1	1	1	1	1
Length = 0x26	Cmd0 = 0x22	Cmd1 = 0x44	Operation	FrameType

8	1	1	1	25
KeySource	SecurityLevel	KeyIdMode	KeyIndex	Channels

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>Operation</b></td><td>1</td><td><b>WiSUN Async operation to perform:</b>
<table><tr><th>Async Operation</th><th>Value</th></tr><tr><td>MAC_WS_OPER_ASYNC_START</td><td>0x00</td></tr><tr><td>MAC_WS_OPER_ASYNC_STOP</td><td>0x01</td></tr></table></td></tr>
<tr><td><b>FrameType</b></td><td>1</td><td><b>WiSUN Async frame type:</b>
<table><tr><th>Async Frame Type</th><th>Value</th></tr><tr><td>MAC_WS_ASYNC_PAN_ADVERT</td><td>0x00</td></tr><tr><td>MAC_WS_ASYNC_PAN_ADVERT_SOL</td><td>0x01</td></tr><tr><td>MAC_WS_ASYNC_PAN_CONFIG</td><td>0x02</td></tr><tr><td>MAC_WS_ASYNC_PAN_CONFIG_SOL</td><td>0x03</td></tr><tr><td>MAC_WS_ASYNC_DATA</td><td>0x04</td></tr><tr><td>MAC_WS_ASYNC_ACK</td><td>0x05</td></tr><tr><td>MAC_WS_ASYNC_EAPOL</td><td>0x06</td></tr><tr><td>MAC_WS_ASYNC_INVALID</td><td>0xFF</td></tr></table></td></tr>
<tr><td><b>KeySource</b></td><td>8</td><td><b>Key Source of this data frame.</b></td></tr>
<tr><td><b>SecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b>
<table><tr><th>Security Level</th><th>Value</th></tr><tr><td>NO_SECURITY</td><td>0x00</td></tr><tr><td>MIC_32_AUTH</td><td>0x01</td></tr><tr><td>MIC_64_AUTH</td><td>0x02</td></tr><tr><td>MIC_128_AUTH</td><td>0x03</td></tr><tr><td>AES_ENCRYPTION</td><td>0x04</td></tr><tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr><tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr><tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr></table></td></tr>
<tr><td><b>KeyIdMode</b></td><td>1</td><td><b>Key Id Mode of this data frame:</b>
<table><tr><th>Key Id Mode</th><th>Value</th></tr><tr><td>NOT_USED</td><td>0x00</td></tr><tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr><tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr><tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr></table></td></tr>
<tr><td><b>KeyIndex</b></td><td>1</td><td><b>Key Index of this data frame</b></td></tr>
<tr><td><b>Channels</b></td><td>17</td><td><b>Bit mask for channels to send Async frames for a start operation</b></td></tr>
</table>

SRSP:

Byte: 1	1	1	1
Length = 1	Cmd0 = 0x62	Cmd1 = 0x44	Status

Attributes:

Attribute	Length	Description

Status	**1**	**Status of WS_ASYNC_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values**.

3.3.23 MAC_FH_ENABLE_REQ

Description:
This command is used to send a frequency hopping enable command to the TI-15.4-STACK-CoP.

Usage:
SREQ:

Byte: 1	1	1
Length = 0x00	Cmd0 = 0x22	Cmd1 = 0x40

Attributes: None

SRSP:

Byte: 1	1	1	1
Length = 1	Cmd0 = 0x62	Cmd1 = 0x40	Status

Attributes:

Attribute	Length	Description
Status	**1**	**Status of FH_ENABLE_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values**.

3.3.24 MAC_FH_START_REQ

Description:
This command is used to send a frequency hopping start command to the TI-15.4-STACK-CoP.

Usage:
SREQ:

Byte: 1	1	1
Length = 0x00	Cmd0 = 0x22	Cmd1 = 0x41

Attributes: None

SRSP:

Byte: 1	1	1	1
Length = 1	Cmd0 = 0x62	Cmd1 = 0x41	Status

Attributes:

Attribute	Length	Description
Status	**1**	**Status of FH_START_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values**.

3.3.25 MAC_FH_GET_REQ

Description:
This command is used to read the value of an attribute from the MAC frequency hopping PIB.

Usage:
SREQ:

Byte: 1	1	1	2
Length = 0x02	Cmd0 = 0x22	Cmd1 = 0x42	AttributeID

Attributes:

Attribute	Length	Description
AttributeID	2	Specifies the Frequency Hopping PIB attribute ID Refer to Section 6.4 for enumerated list of attribute ID values.

SRSP:

Byte: 1	1	1	1	AL
Length = 1+AL	Cmd0 = 0x62	Cmd1 = 0x42	Status	Data

AL = Attribute Length

Attributes:

Attribute	Length	Description
Status	1	Status of FH_GET_REQ message delivery. Refer to Section 6.1 for enumerated list of status values.
Data	AL	FH PIB attribute data

3.3.26 MAC_FH_SET_REQ

Description:
This command is used to write an attribute to the MAC frequency hopping PIB.

Usage:
SREQ:

Byte: 1	1	1	2	AL
Length = 2+AL	Cmd0 = 0x22	Cmd1 = 0x43	AttributeID	Data

AL = Attribute Length

Attributes:

Attribute	Length	Description
AttributeID	2	Specifies the Frequency Hopping PIB attribute ID Refer to Section 6.4 for enumerated list of attribute ID values.
Data	AL	FH PIB attribute data

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x62	Cmd1 = 0x43	Status

Attributes:

Attribute	Length	Description
Status	1	Status of FH_SET_REQ message delivery. Refer to Section 6.1 for enumerated list of status values.

3.4 MT MAC Callback Interface

Following APIs provide callbacks for 802.15.4 network indications and confirms.

3.4.1 MAC_SYNC_LOSS_IND

Description:

This event is sent to the application when the TI-15.4-STACK-CoP loses synchronization with the coordinator or has a PAN ID conflict. The status indicates the reason for the event.

Usage:

AREQ:

1	1	1	1	2	1	1	1
Length = 0x11	Cmd0 = 0x42	Cmd1 = 0x80	Status	PanId	LogicalChannel	ChannelPage	PhyId

8	1	1	1
KeySource	SecurityLevel	KeyIdMode	KeyIndex

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td>Status</td><td>1</td><td><table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>MAC_BEACON_LOSS</td><td>0xE0</td><td>Beacon was lost following a synchronization request.</td></tr><tr><td>MAC_PAN_ID_CONFLICT</td><td>0xEE</td><td>A PAN identifier conflict has been detected.</td></tr><tr><td>MAC_REALIGNMENT</td><td>0xEF</td><td>Coordinator realignment command has been received.</td></tr></table></td></tr>
<tr><td>PanId</td><td>2</td><td>PAN Id of the device</td></tr>
<tr><td>LogicalChannel</td><td>1</td><td>Logical Channel of the device where the synchronization is lost</td></tr>
<tr><td>ChannelPage</td><td>1</td><td>Channel Page of the device where the synchronization is lost</td></tr>
<tr><td>PhyId</td><td>1</td><td>PHY identifier indicates which PHY descriptor to use:<table><tr><th>MAC PHY ID</th><th>Value</th></tr><tr><td>MAC_STD_US_915_PHY_1</td><td>0x01</td></tr><tr><td>MAC_STD_ETSI_863_PHY_3</td><td>0x03</td></tr><tr><td>MAC_MRFSK_GENERIC_PHY_ID_BEGIN</td><td>0x04</td></tr><tr><td>MAC_MRFSK_GENERIC_PHY_ID_END</td><td>0x06</td></tr></table></td></tr>
<tr><td>KeySource</td><td>8</td><td>Key Source of this data frame.</td></tr>
<tr><td>SecurityLevel</td><td>1</td><td>Security Level of this data frame:<table><tr><th>Security Level</th><th>Value</th></tr><tr><td>NO_SECURITY</td><td>0x00</td></tr><tr><td>MIC_32_AUTH</td><td>0x01</td></tr><tr><td>MIC_64_AUTH</td><td>0x02</td></tr><tr><td>MIC_128_AUTH</td><td>0x03</td></tr><tr><td>AES_ENCRYPTION</td><td>0x04</td></tr><tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr><tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr><tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr></table></td></tr>
<tr><td>KeyIdMode</td><td>1</td><td>Key Id Mode of this data frame:<table><tr><th>Key Id Mode</th><th>Value</th></tr><tr><td>NOT_USED</td><td>0x00</td></tr><tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr><tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr><tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr></table></td></tr>
<tr><td>KeyIndex</td><td>1</td><td>Key Index of this data frame.</td></tr>
</table>

3.4.2 MAC_ASSOCIATE_IND

Description:

This event is sent to the application when the MAC receives an associate request from another device. The application must call MAC_ASSOCIATE_RSP after receiving this event. This event will only be sent to FFD applications which set PIB attribute MAC_ASSOCIATION_PERMIT to TRUE.

Usage:
AREQ:

1	1	1	8	1
Length = 0x14	Cmd0 = 0x42	Cmd1 = 0x81	ExtendedAddress	Capabilities

8	1	1	1
KeySource	SecurityLevel	KeyIdMode	KeyIndex

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>ExtendedAddress</b></td><td>8</td><td><b>Extended address of the device</b></td></tr>
<tr><td><b>Capabilities</b></td><td>1</td><td><b>Operating capabilities of the device being directly joined:</b>
<table>
<tr><th>Capability</th><th>Value</th></tr>
<tr><td>MAC_CAPABLE_PAN_COORD</td><td>0x01</td></tr>
<tr><td>MAC_CAPABLE_FFD</td><td>0x02</td></tr>
<tr><td>MAC_CAPABLE_MAINS_POWER</td><td>0x04</td></tr>
<tr><td>MAC_CAPABLE_RX_ON_IDLE</td><td>0x08</td></tr>
<tr><td>MAC_CAPABLE_SECURITY</td><td>0x40</td></tr>
<tr><td>MAC_CAPABLE_ALLOC_ADDR</td><td>0x80</td></tr>
</table></td></tr>
<tr><td><b>KeySource</b></td><td>8</td><td><b>Key Source of this data frame.</b></td></tr>
<tr><td><b>SecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b>
<table>
<tr><th>Security Level</th><th>Value</th></tr>
<tr><td>NO_SECURITY</td><td>0x00</td></tr>
<tr><td>MIC_32_AUTH</td><td>0x01</td></tr>
<tr><td>MIC_64_AUTH</td><td>0x02</td></tr>
<tr><td>MIC_128_AUTH</td><td>0x03</td></tr>
<tr><td>AES_ENCRYPTION</td><td>0x04</td></tr>
<tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr>
<tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr>
<tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr>
</table></td></tr>
<tr><td><b>KeyIdMode</b></td><td>1</td><td><b>Key Id Mode of this data frame:</b>
<table>
<tr><th>Key Id Mode</th><th>Value</th></tr>
<tr><td>NOT_USED</td><td>0x00</td></tr>
<tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr>
<tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr>
<tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr>
</table></td></tr>
<tr><td><b>KeyIndex</b></td><td><b>1</b></td><td><b>Key Index of this data frame.</b></td></tr>
</table>

3.4.3 MAC_ASSOCIATE_CNF

Description:

This event is sent to the application in response to a MAC_ASSOCIATE_REQ. The event indicates the status of the associate attempt. If the associate was successful and a short address was requested, then the short address is included in the event. Otherwise the short address parameter is not valid.

Usage:

AREQ:

1	1	1	1	2	8	1	1	1
Length = 0x0E	**Cmd0 = 0x42**	**Cmd1 = 0x82**	**Status**	**ShortAddress**	**KeySource**	**SecurityLevel**	**KeyIdMode**	**KeyIndex**

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>Status</b></td><td>1</td><td><b>Status of preceding ASSOCIATE_REQ operation.
Refer to Section 6.1 for enumerated list of status values.</b></td></tr>
<tr><td><b>ShortAddress</b></td><td>2</td><td><b>Short address of the device</b></td></tr>
<tr><td><b>KeySource</b></td><td>8</td><td><b>Key Source of this data frame.</b></td></tr>
<tr><td><b>SecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b>
<table>
<tr><th>Security Level</th><th>Value</th></tr>
<tr><td>NO_SECURITY</td><td>0x00</td></tr>
<tr><td>MIC_32_AUTH</td><td>0x01</td></tr>
<tr><td>MIC_64_AUTH</td><td>0x02</td></tr>
<tr><td>MIC_128_AUTH</td><td>0x03</td></tr>
<tr><td>AES_ENCRYPTION</td><td>0x04</td></tr>
<tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr>
<tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr>
<tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr>
</table></td></tr>
<tr><td><b>KeyIdMode</b></td><td>1</td><td><b>Key Id Mode of this data frame:</b>
<table>
<tr><th>Key Id Mode</th><th>Value</th></tr>
<tr><td>NOT_USED</td><td>0x00</td></tr>
<tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr>
<tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr>
<tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr>
</table></td></tr>
<tr><td><b>KeyIndex</b></td><td><b>1</b></td><td><b>Key Index of this data frame.</b></td></tr>
</table>

3.4.4 MAC_BEACON_NOTIFY_IND

Description:

This indication is sent to the application when the TI-15.4-STACK-CoP receives beacon frame(s) for an active or passive scan with “maxResults” set to zero or with PIB attribute MAC_AUTO_REQUEST set to FALSE. One MAC_BEACON_NOTIFY_IND is sent for each beacon received, with no filtering of duplicate beacons. The frame format is different for Standard (type = 0x00) and Enhanced (type = 0x01) beacons, as specified below:

3.4.4.1 Standard Beacon

Usage:

AREQ:

1	1	1	1	1	4	1
Length = 0x26-+DL	**Cmd0 = 0x42**	**Cmd1 = 0x83**	**BeaconType = 0x00**	**BSN**	**Timestamp**	**CoordAddressMode**

8	2	2	1	1	1	1
CoordExtendedAddress	**PanId**	**SuperframeSpec**	**LogicalChannel**	**ChannelPage**	**GTSPermit**	**LinkQuality**

1	8	1	1	1	1	1	1
SecurityFailure	**KeySource**	**SecurityLevel**	**KeyIdMode**	**KeyIndex**	**ShortAddrs**	**ExtAddrs**	**SDULength**

2 * ShortAddrs	8 * ExtAddrs	SDULength
ShortAddrList	**ExtAddrList**	**NSDU**

DL = (2 * ShortAddrs) + (8 * ExtAddrs) + SDULength

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>BeaconType</b></td><td>1</td><td><b>0x00 = Standard Beacon frame</b></td></tr>
<tr><td><b>BSN</b></td><td>1</td><td><b>Beacon sequence number</b></td></tr>
<tr><td><b>Timestamp</b></td><td>4</td><td><b>The time at which the beacon was received, in <i>aUnitBackoffPeriod</i> units</b></td></tr>
<tr><td><b>CoordAddressMode</b></td><td>1</td><td><b>Address mode of the coordinator:</b>
<table>
<tr><th>Mode</th><th>Value</th><th>Description</th></tr>
<tr><td>ADDRESS_16_BIT</td><td>0x02</td><td>Address 16 bit</td></tr>
<tr><td>ADDRESS_64_BIT</td><td>0x03</td><td>Address 64 bit</td></tr>
</table></td></tr>
<tr><td><b>CoordExtendedAddress</b></td><td>8</td><td><b>Extended address of the coordinator</b></td></tr>
<tr><td><b>PanId</b></td><td>2</td><td><b>Pan Id of the device</b></td></tr>
<tr><td><b>SuperframeSpec</b></td><td>2</td><td><b>Superframe specification of the network</b></td></tr>
<tr><td><b>LogicalChannel</b></td><td>1</td><td><b>Logical channel of the network</b></td></tr>
<tr><td><b>ChannelPage</b></td><td>1</td><td></td></tr>
<tr><td><b>GTSPermit</b></td><td>1</td><td><b>TRUE/FALSE - Permit/ does Not permit GTS</b></td></tr>
<tr><td><b>LinkQuality</b></td><td>1</td><td><b>Link quality of the message</b></td></tr>
<tr><td><b>SecurityFailure</b></td><td>1</td><td><b>Set to true if there was an error in security processing</b></td></tr>
<tr><td><b>KeySource</b></td><td>8</td><td><b>Key Source of this data frame.</b></td></tr>
<tr><td><b>SecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b>
<table>
<tr><th>Security Level</th><th>Value</th></tr>
<tr><td>NO_SECURITY</td><td>0x00</td></tr>
<tr><td>MIC_32_AUTH</td><td>0x01</td></tr>
<tr><td>MIC_64_AUTH</td><td>0x02</td></tr>
<tr><td>MIC_128_AUTH</td><td>0x03</td></tr>
<tr><td>AES_ENCRYPTION</td><td>0x04</td></tr>
<tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr>
<tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr>
<tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr>
</table></td></tr>
<tr><td><b>KeyIdMode</b></td><td>1</td><td><b>Key Id Mode of this data frame:</b>
<table>
<tr><th>Key Id Mode</th><th>Value</th></tr>
<tr><td>NOT_USED</td><td>0x00</td></tr>
<tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr>
<tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr>
<tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr>
</table></td></tr>
<tr><td><b>KeyIndex</b></td><td>1</td><td><b>Key Index of this data frame.</b></td></tr>
<tr><td><b>ShortAddrs</b></td><td>1</td><td><b>Number of 16-bit short addresses</b></td></tr>
<tr><td><b>ExtAddrs</b></td><td>1</td><td><b>Number of 64-bit short addresses</b></td></tr>
<tr><td><b>SDULength</b></td><td>1</td><td><b>Length of beacon payload</b></td></tr>
<tr><td><b>ShortAddrList</b></td><td>2*ShortAddrs</td><td><b>List of short addresses for which beacon sender has data</b></td></tr>
<tr><td><b>ExtAddrList</b></td><td>8*ExtAddrs</td><td><b>List of extended addresses for which beacon sender has data</b></td></tr>
<tr><td><b>NSDU</b></td><td>SDULength</td><td><b>Beacon payload</b></td></tr>
</table>

3.4.4.2 Enhanced Beacon

Usage:

AREQ:

1	1	1	1	1	1	1
Length = 0x0A	Cmd0 = 0x42	Cmd1 = 0x83	BeaconType = 0x01	BSN	BeaconOrder	SuperFrameOrder

1	1	1	1	2
FinalCapSlot	EnhBeaconOrder	OfsTimeSlot	CapBackOff	NonBeaconOrder

Attributes:

Attribute	Length	Description
BeaconType	1	0x01 = Enhanced Beacon frame
BSN	1	Beacon sequence number
BeaconOrder	1	Beacon interval, calculated as follows: interval (ms) = (*aBaseSuperframeDuration* ms) * 2^BeaconOrder Valid range is 0-14. For a non beacon-enabled network set to 15.
SuperFrameOrder	1	Length of time during which the superframe is active.
FinalCapSlot	1	Final CAP slot extracted from the SuperFrameSpec
EnhBeaconOrder	1	Exponent used to calculate the enhanced beacon interval
OfsTimeSlot	1	Time difference between the enhanced beacon and preceding periodic beacon
CapBackOff	1	Actual slot position for transmission of the enhanced beacon
NonBeaconOrder	2	How often to TX the enhanced beacon in a non-beacon enabled PAN A value of 16383 indicates no enhanced beacon in a non-beacon enabled PAN

3.4.5 MAC_DISASSOCIATE_IND

Description:

This event is sent to the application to indicate that the device has been disassociated from the network.

Usage:

AREQ:

1	1	1	8	1
Length = 0x14	Cmd0 = 0x42	Cmd1 = 0x86	ExtendedAddress	DisassociateReason

8	1	1	1
KeySource	SecurityLevel	KeyIdMode	KeyIndex

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td>ExtendedAddress</td><td>8</td><td>Extended address of the device leaving the network</td></tr>
<tr><td>DisassociateReason</td><td>1</td><td>Reason of the disassociation:
<table>
<tr><th>Reason</th><th>Value</th></tr>
<tr><td>Coordinator wishes the device to disassociate</td><td>0x01</td></tr>
<tr><td>Device itself wishes to disassociate</td><td>0x02</td></tr>
</table></td></tr>
<tr><td>KeySource</td><td>8</td><td>Key Source of this data frame.</td></tr>
</table>

<table>
<tr><td><b>SecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b>
<table>
<tr><th>Security Level</th><th>Value</th></tr>
<tr><td>NO_SECURITY</td><td>0x00</td></tr>
<tr><td>MIC_32_AUTH</td><td>0x01</td></tr>
<tr><td>MIC_64_AUTH</td><td>0x02</td></tr>
<tr><td>MIC_128_AUTH</td><td>0x03</td></tr>
<tr><td>AES_ENCRYPTION</td><td>0x04</td></tr>
<tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr>
<tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr>
<tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr>
</table></td></tr>
<tr><td><b>KeyIdMode</b></td><td>1</td><td><b>Key Id Mode of this data frame:</b>
<table>
<tr><th>Key Id Mode</th><th>Value</th></tr>
<tr><td>NOT_USED</td><td>0x00</td></tr>
<tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr>
<tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr>
<tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr>
</table></td></tr>
<tr><td><b>KeyIndex</b></td><td><b>1</b></td><td><b>Key Index of this data frame.</b></td></tr>
</table>

3.4.6 MAC_DISASSOCIATE_CNF

Description:

This event is sent to the application in response to a MAC_DISASSOCIATE_REQ. The event indicates the status of the disassociate attempt.

Usage:

AREQ:

1	1	1	1	1	8	2
Length = 0x0C	**Cmd0 = 0x42**	**Cmd1 = 0x87**	**Status**	**DeviceAddrMode**	**DeviceAddr**	**DevicePanId**

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>Status</b></td><td>1</td><td><b>Status of preceding DISASSOCIATE_REQ operation.
Refer to Section 6.1 for enumerated list of status values.</b></td></tr>
<tr><td><b>DeviceAddrMode</b></td><td>1</td><td><b>Address mode of the device</b>
<table>
<tr><th>Mode</th><th>Value</th><th>Description</th></tr>
<tr><td><b>ADDRESS_16_BIT</b></td><td>0x02</td><td><b>Address 16 bit</b></td></tr>
<tr><td><b>ADDRESS_64_BIT</b></td><td><b>0x03</b></td><td><b>Address 64 bit</b></td></tr>
</table></td></tr>
<tr><td><b>DeviceAddr</b></td><td>8</td><td><b>Address of the device</b></td></tr>
<tr><td><b>DevicePanId</b></td><td>2</td><td><b>Pan Id of the device</b></td></tr>
</table>

3.4.7 MAC_ORPHAN_IND

Description:

This event is sent to the application when the MAC receives an orphan notification from another device. The application must call MAC_ORPHAN_RSP after receiving this event. This event will only be sent to FFD applications.

Usage:

AREQ:

1	1	1	8	8	1	1	1
Length = 0x13	**Cmd0 = 0x42**	**Cmd1 = 0x8A**	**ExtendedAddress**	**KeySource**	**SecurityLevel**	**KeyIdMode**	**KeyIndex**

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>ExtendedAddress</b></td><td>8</td><td><b>Extended address of the orphan device</b></td></tr>
<tr><td><b>KeySource</b></td><td>8</td><td><b>Key Source of this data frame.</b></td></tr>
<tr><td><b>SecurityLevel</b></td><td>1</td><td><b>Security Level of this data frame:</b>
<table>
<tr><th>Security Level</th><th>Value</th></tr>
<tr><td>NO_SECURITY</td><td>0x00</td></tr>
<tr><td>MIC_32_AUTH</td><td>0x01</td></tr>
<tr><td>MIC_64_AUTH</td><td>0x02</td></tr>
<tr><td>MIC_128_AUTH</td><td>0x03</td></tr>
<tr><td>AES_ENCRYPTION</td><td>0x04</td></tr>
<tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr>
<tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr>
<tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr>
</table></td></tr>
<tr><td><b>KeyIdMode</b></td><td>1</td><td><b>Key Id Mode of this data frame:</b>
<table>
<tr><th>Key Id Mode</th><th>Value</th></tr>
<tr><td>NOT_USED</td><td>0x00</td></tr>
<tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr>
<tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr>
<tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr>
</table></td></tr>
<tr><td><b>KeyIndex</b></td><td>1</td><td><b>Key Index of this data frame.</b></td></tr>
</table>

3.4.8 MAC_POLL_CNF

Description:

This event is sent to the application in response to a MAC_POLL_REQ. If the poll request was successful and data was received the status is set to MAC_SUCCESS. If the poll request was successful and no data was received the status is set to MAC_NO_DATA. Other status values indicate failure as described below.

Usage:

AREQ:

1	1	1	1	1
Length = 0x02	**Cmd0 = 0x42**	**Cmd1 = 0x8B**	**Status**	**FramePending**

Attributes:

Attribute	Length	Description
Status	1	**Status of preceding POLL_REQ operation.** **Refer to Section 6.1 for enumerated list of status values.**
FramePending	1	**TRUE indicates that framePending bit in the data packet is set**

3.4.9 MAC_POLL_IND

Description:

This event is sent to the application in response to a MAC_POLL_REQ.

Usage:

AREQ:

1	1	1	1	8	2	1
Length = 0x0C	**Cmd0 = 0x42**	**Cmd1 = 0x91**	**AddrMode**	**DevAddr**	**PanID**	**NoResponse**

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>AddrMode</b></td><td>1</td><td><b>Address mode of the device:</b>
<table>
<tr><th>Mode</th><th>Value</th><th>Description</th></tr>
<tr><td><b>ADDRESS_16_BIT</b></td><td>0x02</td><td><b>Address 16 bit</b></td></tr>
<tr><td><b>ADDRESS_64_BIT</b></td><td><b>0x03</b></td><td><b>Address 64 bit</b></td></tr>
</table></td></tr>
<tr><td><b>DevAddr</b></td><td>8</td><td><b>Address of the device</b></td></tr>
<tr><td><b>PanID</b></td><td>2</td><td><b>PAN ID of the device</b></td></tr>
<tr><td><b>NoResponse</b></td><td>1</td><td><b>TRUE if no response is needed</b></td></tr>
</table>

3.4.10 MAC_SCAN_CNF

Description:

This event is sent to the application in response to a MAC_SCAN_REQ when the scan operation is complete. The event indicates the status of the scan. For an energy detect scan a list of energy measurements is returned. For an active or passive scan a list of PAN descriptors is returned.

Usage:

AREQ:

1	1	1	1	1	1	1
Length = 0x0C+RL	**Cmd0 = 0x42**	**Cmd1 = 0x8C**	**Status**	**ScanType**	**ChannelPage**	**PhyId**

17	1	RL
UnscannedChannels	**ResultListCount**	**ResultList**

RL = Result List Length

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>Status</b></td><td>1</td><td><b>Status of preceding SCAN_REQ operation.</b>
<b>Refer to Section 6.1 for enumerated list of status values.</b></td></tr>
<tr><td><b>ScanType</b></td><td>1</td><td><b>Specifies the scan type:</b>
<table>
<tr><th>Scan Type</th><th>Value</th></tr>
<tr><td>ENERGY_DETECT</td><td>0x00</td></tr>
<tr><td>ACTIVE</td><td>0x01</td></tr>
<tr><td>PASSIVE</td><td>0x02</td></tr>
<tr><td>ORPHAN</td><td>0x03</td></tr>
<tr><td>ACTIVE_ENHANCED</td><td>0x05</td></tr>
</table></td></tr>
<tr><td><b>ChannelPage</b></td><td>1</td><td><b>Channel Page of scan</b></td></tr>
</table>

<table>
<tr><td><b>PhyId</b></td><td>1</td><td><b>PHY identifier to indicate which PHY descriptor to use:</b>
<table>
<tr><th>MAC PHY ID</th><th>Value</th></tr>
<tr><td>MAC_STD_US_915_PHY_1</td><td>0x01</td></tr>
<tr><td>MAC_STD_ETSI_863_PHY_3</td><td>0x03</td></tr>
<tr><td>MAC_MRFSK_GENERIC_PHY_ID_BEGIN</td><td>0x04</td></tr>
<tr><td>MAC_MRFSK_GENERIC_PHY_ID_END</td><td>0x06</td></tr>
</table></td></tr>
<tr><td><b>UnscannedChannels</b></td><td>17</td><td><b>Bit mask of un-scanned channels</b></td></tr>
<tr><td><b>ResultListCount</b></td><td>1</td><td><b>Number of items in the result list. Zero if scanType is MAC_SCAN_ORPHAN.</b></td></tr>
<tr><td><b>ResultList</b></td><td><b>RL</b></td><td><b>Result list, depending on ScanType:</b>

<b>ORPHAN: none, (RL = ResultListCount = 0)</b>

<b>ENERGY: array of 8-bit energy values, one for each channel scanned (RL = ResultListCount)</b>

<b>ACTIVE:</b>
<b>PASSIVE: array of PAN Descriptors, one for each network found (RL = ResultListCount * 33)</b>
<table>
<tr><th>PAN Descriptor Element</th><th>Length (bytes)</th><th>Data Type</th></tr>
<tr><td>coordAddrMode</td><td>1</td><td>uint8</td></tr>
<tr><td>coordAddress</td><td>8</td><td>uint8</td></tr>
<tr><td>coordPanId</td><td>2</td><td>uint16</td></tr>
<tr><td>superframeSpec</td><td>2</td><td>uint16</td></tr>
<tr><td>logicalChannel</td><td>1</td><td>uint8</td></tr>
<tr><td>channelPage</td><td>1</td><td>uint8</td></tr>
<tr><td>gtsPermit</td><td>1</td><td>bool</td></tr>
<tr><td>linkQuality</td><td>1</td><td>uint8</td></tr>
<tr><td>timestamp</td><td>4</td><td>uint32</td></tr>
<tr><td>securityFailure</td><td>1</td><td>bool</td></tr>
<tr><td>keySource[ ]</td><td>8</td><td>uint8</td></tr>
<tr><td>securityLevel</td><td>1</td><td>uint8</td></tr>
<tr><td>keyIdMode</td><td>1</td><td>uint8</td></tr>
<tr><td>keyIndex</td><td>1</td><td>uint8</td></tr>
</table></td></tr>
</table>

3.4.11 MAC_COMM_STATUS_IND

Description:

This event is sent to the application for various reasons. It indicates the status of a MAC_ASSOCIATE_RSP or MAC_ORPHAN_RSP. It also indicates the TI-15.4-STACK-CoP has received a secure frame that generated an error during security processing.

Usage:

AREQ:

1	1	1	1	1	8	1	8
Length = 0x21	**Cmd0 = 0x42**	**Cmd1 = 0x8D**	**Status**	**SrcAddrMode**	**SrcAddr**	**DstAddrMode**	**DstAddr**

2	1	8	1	1	1
DevicePanId	**Reason**	**KeySource**	**SecurityLevel**	**KeyIdMode**	**KeyIndex**

Attributes:

Attribute	Length	Description
Status	1	**Status of preceding ASSOCIATE_RSP operation.** **Refer to Section 6.1 for enumerated list of status values.**

<table>
<tr><td>SrcAddrMode</td><td>1</td><td>Source address mode<table><tr><th>Mode</th><th>Value</th><th>Description</th></tr><tr><td>ADDRESS_16_BIT</td><td>0x02</td><td>Address 16 bit</td></tr><tr><td>ADDRESS_64_BIT</td><td>0x03</td><td>Address 64 bit</td></tr></table></td></tr>
<tr><td>SrcAddr</td><td>8</td><td>Source address</td></tr>
<tr><td>DstAddrMode</td><td>1</td><td>Destination address mode<table><tr><th>Mode</th><th>Value</th><th>Description</th></tr><tr><td>ADDRESS_16_BIT</td><td>0x02</td><td>Address 16 bit</td></tr><tr><td>ADDRESS_64_BIT</td><td>0x03</td><td>Address 64 bit</td></tr></table></td></tr>
<tr><td>DstAddr</td><td>8</td><td>Destination address</td></tr>
<tr><td>DevicePanId</td><td>2</td><td>Pan Id of the device that generate the indication</td></tr>
<tr><td>Reason</td><td>1</td><td>The reason the event was generated:<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>MAC_COMM_ASSOCIATE_RSP</td><td>0x00</td><td>Event sent in response to MAC_AssociateRsp().</td></tr><tr><td>MAC_COMM_ORPHAN_RSP</td><td>0x01</td><td>Event sent in response to MAC_OrphanRsp().</td></tr><tr><td>MAC_COMM_RX_SECURE</td><td>0x02</td><td>Event sent as a result of receiving a secure frame.</td></tr></table></td></tr>
<tr><td>KeySource</td><td>8</td><td>Key Source of this data frame.</td></tr>
<tr><td>SecurityLevel</td><td>1</td><td>Security Level of this data frame:<table><tr><th>Security Level</th><th>Value</th></tr><tr><td>NO_SECURITY</td><td>0x00</td></tr><tr><td>MIC_32_AUTH</td><td>0x01</td></tr><tr><td>MIC_64_AUTH</td><td>0x02</td></tr><tr><td>MIC_128_AUTH</td><td>0x03</td></tr><tr><td>AES_ENCRYPTION</td><td>0x04</td></tr><tr><td>AES_ENCRYPTION_MIC_32</td><td>0x05</td></tr><tr><td>AES_ENCRYPTION_MIC_64</td><td>0x06</td></tr><tr><td>AES_ENCRYPTION_MIC_128</td><td>0x07</td></tr></table></td></tr>
<tr><td>KeyIdMode</td><td>1</td><td>Key Id Mode of this data frame:<table><tr><th>Key Id Mode</th><th>Value</th></tr><tr><td>NOT_USED</td><td>0x00</td></tr><tr><td>KEY_1BYTE_INDEX</td><td>0x01</td></tr><tr><td>KEY_4BYTE_INDEX</td><td>0x02</td></tr><tr><td>KEY_8BYTE_INDEX</td><td>0x03</td></tr></table></td></tr>
<tr><td>KeyIndex</td><td>1</td><td>Key Index of this data frame.</td></tr>
</table>

3.4.12 MAC_START_CNF

Description:

This event is sent to the application in response to a MAC_START_REQ. The event indicates the status of the start request.

Usage:

AREQ:

1	1	1	1
Length = 0x01	Cmd0 = 0x42	Cmd1 = 0x8E	Status

Attributes:

Attribute	Length	Description
Status	1	Status of preceding START_REQ operation. Refer to Section 6.1 for enumerated list of status values.

3.4.13 MAC_WS_ASYNC_CNF

Description:

This callback is called by the MAC to send a MAC WiSUN async frame confirmation.

Usage:

AREQ:

1	1	1	1
Length = 0x01	**Cmd0 = 0x42**	**Cmd1 = 0x92**	**Status**

Attributes:

Attribute	Length	Description
Status	1	**Status of preceding ASYNC_REQ operation.** **Refer to Section 6.1 for enumerated list of status values.**

3.5 *MT SYS Interface*

3.5.1 SYS_RESET_REQ

Description:

This command is used to reset the target device.

Usage:

AREQ:

1	1	1	1
Length = 0x01	**Cmd0 = 0x41**	**Cmd1 = 0x00**	**Type**

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>Type</b></td><td>1</td><td><b>Type of reset requested:</b>
<table>
<tr><th>Reset Type</th><th>Value</th></tr>
<tr><td><b>Hard</b></td><td><b>0</b></td></tr>
<tr><td><b>Soft</b></td><td><b>1</b></td></tr>
</table>
</td></tr>
</table>

3.5.2 SYS_PING_REQ

Description:

This command is used to confirm serial communication with the device and get the device’s MT capabilities.

Usage:

SREQ:

1	1	1
Length = 0x00	**Cmd0 = 0x21**	**Cmd1 = 0x01**

SRSP:

Byte: 1	1	1	2
Length = 0x01	**Cmd0 = 0x61**	**Cmd1 = 0x01**	**Capabilites**

Attributes:

Attribute	Length	Description

<table>
<tr><td>Capabilites</td><td>2</td><td>Bit-mask indicates available MT sub-systems:
<table>
<tr><th>Subsystem</th><th>Enable Bit</th></tr>
<tr><td>MT_SYS</td><td>0x0001</td></tr>
<tr><td>MT_MAC</td><td>0x0002</td></tr>
<tr><td>MT_UTIL</td><td>0x0040</td></tr>
<tr><td>MT_APP</td><td>0x0100</td></tr>
</table></td></tr>
</table>

3.5.3 SYS_VERSION_REQ

Description:

This command is used to obtain the device’s version information.

Usage:

SREQ:

1	1	1
Length = 0x00	Cmd0 = 0x21	Cmd1 = 0x02

SRSP:

Byte: 1	1	1	1	1	1	1	1
Length = 0x05	Cmd0 = 0x61	Cmd1 = 0x02	Transport	Product	Major	Minor	Maint

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td>Transport</td><td>1</td><td>Transport protocol revision:
<table>
<tr><th>Revision</th><th>Description</th></tr>
<tr><td>2</td><td>Standard RPC frame, no fragmentation</td></tr>
<tr><td>3</td><td>Extended RPC frame, fragmentation</td></tr>
</table></td></tr>
<tr><td>Product</td><td>1</td><td>Product ID code:
<table>
<tr><th>ID</th><th>Product</th></tr>
<tr><td>0</td><td>Z-Stack</td></tr>
<tr><td>1</td><td>TI-15.4-Stack</td></tr>
</table></td></tr>
<tr><td>Major</td><td>1</td><td>Software major release version number</td></tr>
<tr><td>Minor</td><td>1</td><td>Software minor release version number</td></tr>
<tr><td>Maint</td><td>1</td><td>Software maintenance release version number</td></tr>
</table>

3.5.4 SYS_NV_CREATE_REQ

Description:

This command is used to create an item in the TI-15.4-STACK-CoP non-volatile memory.

Usage:

SREQ:

Byte: 1	1	1	1	2	2	4
Length = 0x09	Cmd0 = 0x21	Cmd1 = 0x30	SysID	ItemID	SubId	Length

Attributes:

Attribute	Length	Description
SysID	1	System ID of the NV item
ItemID	2	Item ID of the NV item
SubID	2	Sub ID of the NV item

Length	4	**Length of the NV item**

SRSP:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x61**	**Cmd1 = 0x30**	**Status**

Attributes:

Attribute	Length	Description
Status	1	**Status of NV_CREATE_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values.**

3.5.5 SYS_NV_DELETE_REQ

Description:

This command is used to delete an item from the TI-15.4-STACK-CoP non-volatile memory.

Usage:

SREQ:

Byte: 1	1	1	1	2	2
Length = 0x05	**Cmd0 = 0x21**	**Cmd1 = 0x31**	**SysID**	**ItemID**	**SubId**

Attributes:

Attribute	Length	Description
SysID	1	**System ID of the NV item**
ItemID	2	**Item ID of the NV item**
SubID	2	**Sub ID of the NV item**

SRSP:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x61**	**Cmd1 = 0x31**	**Status**

Attributes:

Attribute	Length	Description
Status	1	**Status of NV_DELETE_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values.**

3.5.6 SYS_NV_LENGTH_REQ

Description:

This command is used to determine the length of an item in the TI-15.4-STACK-CoP non-volatile memory.

Usage:

SREQ:

Byte: 1	1	1	1	2	2
Length = 0x05	**Cmd0 = 0x21**	**Cmd1 = 0x32**	**SysID**	**ItemID**	**SubId**

Attributes:

Attribute	Length	Description
SysID	1	**System ID of the NV item**
ItemID	2	**Item ID of the NV item**
SubID	2	**Sub ID of the NV item**

SRSP:

Byte: 1	1	1	4
Length = 0x04	Cmd0 = 0x61	Cmd1 = 0x32	Length

Attributes:

Attribute	Length	Description
Length	4	Length of data for specified NV item, 0=item does not exist

3.5.7 SYS_NV_READ_REQ

Description:

This command is used to read an item from the TI-15.4-STACK-CoP non-volatile memory.

Usage:

SREQ:

Byte: 1	1	1	1	2	2	2	1
Length = 0x08	Cmd0 = 0x21	Cmd1 = 0x33	SysID	ItemID	SubId	Offset	Length

Attributes:

Attribute	Length	Description
SysID	1	System ID of the NV item
ItemID	2	Item ID of the NV item
SubID	2	Sub ID of the NV item
Offset	2	Offset into NV data item
Length	1	Length of the NV item

SRSP:

Byte: 1	1	1	1	1	DL
Length = 2+DL	Cmd0 = 0x61	Cmd1 = 0x33	Status	Length	Data

DL = Returned NV Data Length

Attributes:

Attribute	Length	Description
Status	1	Status of NV_READ_REQ message delivery. Refer to Section 6.1 for enumerated list of status values.
Length	1	Length of NV data returned from CoP
Data	DL	NV data returned from CoP

3.5.8 SYS_NV_WRITE_REQ

Description:

This command is used to write an item to the TI-15.4-STACK-CoP non-volatile memory.

Usage:

SREQ:

Byte: 1	1	1	1	2	2	2	1	DL
Length = 8+DL	Cmd0 = 0x21	Cmd1 = 0x34	SysID	ItemID	SubId	Offset	Length	Data

DL = NV Data Length

Attributes:

Attribute	Length	Description
SysID	1	**System ID of the NV item**
ItemID	2	**Item ID of the NV item**
SubID	2	**Sub ID of the NV item**
Offset	2	**Offset into NV data item**
Length	1	**Length of the NV item**
Data	DL	**NV data to be written to CoP**

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x61	Cmd1 = 0x34	Status

Attributes:

Attribute	Length	Description
Status	1	**Status of NV_WRITE_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values.**

3.5.9 SYS_NV_UPDATE_REQ

Description:

This command is used to create (if needed) and write an item to the TI-15.4-STACK-CoP non-volatile memory.

Usage:

SREQ:

Byte: 1	1	1	1	2	2	1	DL
Length = 6+DL	Cmd0 = 0x21	Cmd1 = 0x35	SysID	ItemID	SubId	Length	Data

DL = NV Data Length

Attributes:

Attribute	Length	Description
SysID	1	**System ID of the NV item**
ItemID	2	**Item ID of the NV item**
SubID	2	**Sub ID of the NV item**
Length	1	**Length of the NV item**
Data	DL	**NV data to be written to CoP**

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x61	Cmd1 = 0x35	Status

Attributes:

Attribute	Length	Description
Status	**1**	**Status of NV_UPDATE_REQ message delivery.** **Refer to Section 6.1 for enumerated list of status values.**

3.5.10 SYS_NV_COMPACT_REQ

Description:

This command is used to compact the active page in the TI-15.4-STACK-CoP non-volatile memory.

Usage:

SREQ:

Byte: 1	1	1	2
Length = 0x02	Cmd0 = 0x21	Cmd1 = 0x36	Threshold

Attributes:

Attribute	Length	Description
Threshold	2	Perform compaction if number of available bytes in NV is less than this value. Setting this value to zero forces compaction and active page change.

SRSP:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x61	Cmd1 = 0x36	Status

Attributes:

Attribute	Length	Description
Status	1	Status of NV_COMPACT_REQ message delivery. Refer to Section 6.1 for enumerated list of status values.

3.5.11 SYS_RESET_IND

Description:

This indication is received after the target device resets.

Usage:

AREQ:

Byte: 1	1	1	1	1	1	1	1	1
Length = 0x06	Cmd0 = 0x41	Cmd1 = 0x80	Reason	Transport	Product	Major	Minor	Maint

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td>Reason</td><td>1</td><td>Reason for the reset:
<table>
<tr><th>Reset Type</th><th>Value</th></tr>
<tr><td>Hardware</td><td>0</td></tr>
<tr><td>Host request</td><td>1</td></tr>
<tr><td>HAL assert</td><td>2</td></tr>
<tr><td>MAC assert</td><td>3</td></tr>
<tr><td>RTOS assert</td><td>4</td></tr>
</table></td></tr>
<tr><td>Transport</td><td>1</td><td>Transport protocol revision:
<table>
<tr><th>Revision</th><th>Description</th></tr>
<tr><td>2</td><td>Standard RPC frame, no fragmentation</td></tr>
<tr><td>3</td><td>Extended RPC frame, fragmentation</td></tr>
</table></td></tr>
<tr><td>Product</td><td>1</td><td>Product ID code:
<table>
<tr><th>ID</th><th>Product</th></tr>
<tr><td>0</td><td>Z-Stack</td></tr>
<tr><td>1</td><td>TI-15.4-Stack</td></tr>
</table></td></tr>
<tr><td>Major</td><td>1</td><td>Software major release version number</td></tr>
<tr><td>Minor</td><td>1</td><td>Software minor release version number</td></tr>
<tr><td>Maint</td><td>1</td><td>Software maintenance release version number</td></tr>
</table>

3.6 *MT UTIL Interface*

3.6.1 UTIL_CALLBACK_SUB_CMD

Description:

This command subscribes/unsubscribes to software layer callbacks. For particular subsystem callbacks to work, the software must be compiled with a special flag that is unique to that subsystem to enable the callback mechanism. For example to enable MAC callbacks, the MT_MAC_CB_FUNC flag must be compiled when the software is built.

Usage:

SREQ:

Byte: 1	1	1	1	4
Length = 0x05	Cmd0 = 0x27	Cmd1 = 0x06	SubsystemId	Enables

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td>SubsystemId</td><td>1</td><td>ID to select sub-systems to alter callbacks:
<table>
<tr><th>Subsystem</th><th>ID</th></tr>
<tr><td>MT_SYS</td><td>0x01</td></tr>
<tr><td>MT_MAC</td><td>0x02</td></tr>
<tr><td>MT_UTIL</td><td>0x07</td></tr>
<tr><td>ALL_SUBSYSTEMS</td><td>0xFF</td></tr>
</table></td></tr>
<tr><td>Enables</td><td>4</td><td>Bit-mask to enable individual callbacks
<table>
<tr><th>MT_MAC</th><th>Enable Bit</th></tr>
<tr><td>MAC_ASSOCIATE_CNF</td><td>0x00000001</td></tr>
<tr><td>MAC_ASSOCIATE_IND</td><td>0x00000002</td></tr>
<tr><td>MAC_BEACON_NOTIFY_IND</td><td>0x00000004</td></tr>
<tr><td>MAC_COMM_STATUS_IND</td><td>0x00000008</td></tr>
<tr><td>MAC_DATA_CNF</td><td>0x00000010</td></tr>
<tr><td>MAC_DATA_IND</td><td>0x00000020</td></tr>
<tr><td>MAC_DISASSOCIATE_CNF</td><td>0x00000040</td></tr>
<tr><td>MAC_DISASSOCIATE_IND</td><td>0x00000080</td></tr>
<tr><td>MAC_ORPHAN_IND</td><td>0x00000100</td></tr>
<tr><td>MAC_POLL_CNF</td><td>0x00000200</td></tr>
<tr><td>MAC_POLL_IND</td><td>0x00000400</td></tr>
<tr><td>MAC_PURGE_CNF</td><td>0x00000800</td></tr>
<tr><td>MAC_SCAN_CNF</td><td>0x00001000</td></tr>
<tr><td>MAC_START_CNF</td><td>0x00002000</td></tr>
<tr><td>MAC_SYNC_LOSS_IND</td><td>0x00004000</td></tr>
<tr><td>MAC_WS_ASYNC_CNF</td><td>0x00008000</td></tr>
<tr><td>MAC_WS_ASYNC_IND</td><td>0x00010000</td></tr>
<tr><td>DISABLE_SELECTED_CALLBACKS</td><td>0x80000000</td></tr>
</table>
<table>
<tr><th>MT_SYS</th><th>Enable Bit</th></tr>
<tr><td>SYS_RESET_IND</td><td>0x00000001</td></tr>
<tr><td>DISABLE_SELECTED_CALLBACKS</td><td>0x80000000</td></tr>
</table></td></tr>
</table>

SRSP:

Byte: 1	1	1	1	4
Length = 0x05	Cmd0 = 0x67	Cmd1 = 0x06	Status	Enables

Attributes:

Attribute	Length	Description

Status	1	**Status of CALLBACK_SUB_CMD message delivery.** **Refer to Section 6.1 for enumerated list of status values.**
Enables	**4**	**Bit-mask of enabled callbacks for selected sub-system**

3.6.2 MT_UTIL_GET_EXT_ADDR

Description:

This API is used to get one of several 64-bit extended addresses from the device, including the “active” device address from the MAC PIB, the unique “factory-programmed” address from the chip’s INFO memory, and the “user-programmable” address stored in the configuration page.

Usage:

SREQ:

Byte: 1	1	1	1
Length = 0x01	Cmd0 = 0x27	Cmd1 = 0xEE	Type

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>Type</b></td><td><b>1</b></td><td><b>Type of extended address requested:</b>
<table>
<tr><th>Extended Address Type</th><th>Value</th></tr>
<tr><td>DEVICE_MAC_PIB</td><td>0x00</td></tr>
<tr><td>DEVICE_PRIMARY</td><td>0x01</td></tr>
<tr><td>DEVICE_USER_CCFG</td><td>0x02</td></tr>
</table></td></tr>
</table>

SRSP:

Byte: 1	1	1	1	8
Length = 0x09	Cmd0 = 0x67	Cmd1 = 0xEE	Type	ExtAddress

Attributes:

<table>
<tr><th>Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td><b>Type</b></td><td>1</td><td><b>Type of extended address requested:</b>
<table>
<tr><th>Extended Address Type</th><th>Value</th></tr>
<tr><td>DEVICE_MAC_PIB</td><td>0x00</td></tr>
<tr><td>DEVICE_PRIMARY</td><td>0x01</td></tr>
<tr><td>DEVICE_USER_CCFG</td><td>0x02</td></tr>
<tr><td>UNKNOWN</td><td>0xFF</td></tr>
</table></td></tr>
<tr><td><b>ExtAddress</b></td><td>8</td><td><b>The value returned is LSB first.</b></td></tr>
</table>

3.6.3 MT_UTIL_LOOPBACK

Description:

This API is used to test the serial interface between the host and TI-15.4-STACK-CoP. Single (SREQ) and repeated (AREQ) responses from the TI-15.4-STACK-CoP are supported, with variable length data blocks up to 246 bytes.

Usage:

SREQ:

Byte: 1	1	1	1	4	DL
Length = 0x05+DL	Cmd0 = 0x27	Cmd1 = 0x10	Repeats	Interval	Data

DL = Loopback **D**ata **L**ength

Attributes:

Attribute	Length	Description
Repeats	1	**Number of repeats (AREQ) after initial SRSP** **Set to 0xFF for continuous repeats**
Interval	4	**Number of milliseconds between AREQ responses**
Data	DL	**Variable length data block to be echoed back**

SRSP:

Byte: 1	1	1	1	4	DL
Length = 0x05+DL	Cmd0 = 0x67	Cmd1 = 0x10	Repeats	Interval	Data

AREQ:

Byte: 1	1	1	1	4	DL
Length = 0x05+DL	Cmd0 = 0x47	Cmd1 = 0x10	Repeats	Interval	Data

Attributes:

Attribute	Length	Description
Repeats	1	**Number of remaining AREQ responses** **Set to 0xFF for continuous repeats**
Interval	4	**Number of milliseconds until next AREQ response**
Data	DL	**Variable length data block that was echoed back**

3.6.4 MT_UTIL_RANDOM

Description:

This command is used to return a 16-bit random number from the TI-15.4-STACK-CoP.

Usage:

SREQ:

Byte: 1	1	1
Length = 0x00	Cmd0 = 0x27	Cmd1 = 0x12

SRSP:

Byte: 1	1	1	2
Length = 0x02	Cmd0 = 0x67	Cmd1 = 0x12	Number

Attributes:

Attribute	Length	Description
Number	2	**16-bit random number**

4. Status Code and PIB Attributes

4.1 *MAC Status Values*

NAME	DESCRIPTION	VALUE
MAC_SUCCESS	Operation successful.	**0x00**
MAC_UNSUPPORTED	The operation is not supported in the current configuration.	**0x18**
MAC_BAD_STATE	The operation could not be performed in the current state.	**0x19**
MAC_NO_RESOURCES	The operation could not be completed because memory allocation failed.	**0x1A**
RPC_COMMAND_SUBSYSTEM_ERROR	RPC message was received with unknown sub-system id	**0x25**
RPC_COMMAND_ID_ERROR	RPC message was received with unknown command id	**0x26**
RPC_COMMAND_LENGTH_ERROR	RPC message was received with incorrect length	**0x27**
RPC_COMMAND_UNSUPPORTED_TYPE	RPC message was received with unknown operation type	**0x28**
FHAPI_STATUS_ERR	Frequency hopping: general error	**0x61**
FHAPI_STATUS_ERR_NOT_SUPPORTED_IE	IE is not supported in frequency hopping	**0x62**
FHAPI_STATUS_ERR_NOT_IN_ASYNC	There is no ASYNC message in MAC TX queue	**0x63**
FHAPI_STATUS_ERR_NO_ENTRY_IN_THE_NEIGHBOR	Destination address is not in frequency hopping neighbor table	**0x64**
FHAPI_STATUS_ERR_OUT_SLOT	Frequency hopping is not in UC or BC dwell time slot	**0x65**
FHAPI_STATUS_ERR_INVALID_ADDRESS	Frequency hopping: address is invalid	**0x66**
FHAPI_STATUS_ERR_INVALID_FORMAT	IE format is wrong	**0x67**
FHAPI_STATUS_ERR_NOT_SUPPORTED_PIB	PIB is not supported in frequency hopping module	**0x68**
FHAPI_STATUS_ERR_READ_ONLY_PIB	PIB is read only in frequency hopping module	**0x69**
FHAPI_STATUS_ERR_INVALID_PARAM_PIB	Parameter is invalid in frequency hopping PIB API	**0x6A**
FHAPI_STATUS_ERR_INVALID_FRAME_TYPE	Invalid frequency hopping frame type	**0x6B**
FHAPI_STATUS_ERR_EXPIRED_NODE	Expired frequency hopping node	**0x6C**
MAC_COUNTER_ERROR	Frame counter purportedly applied by the originator of the received frame is invalid.	**0xDB**
MAC_IMPROPER_KEY_TYPE	The key purportedly applied by the originator of the received frame is not allowed.	**0xDC**
MAC_IMPROPER_SECURITY_LEVEL	The security level purportedly applied by the originator of the received frame does not meet the minimum security level.	**0xDD**
MAC_UNSUPPORTED_LEGACY	The received frame was secured with legacy security which is not supported.	**0xDE**
MAC_UNSUPPORTED_SECURITY	The security of the received frame is not supported.	**0xDF**
MAC_BEACON_LOSS	The beacon was lost following a synchronization request.	**0xE0**
MAC_CHANNEL_ACCESS_FAILURE	The operation or data request failed because of activity on the channel.	**0xE1**
MAC_DENIED	The MAC was not able to enter low power mode.	**0xE2**
MAC_DISABLE_TRX_FAILURE	Unused.	**0xE3**
MAC_SECURITY_ERROR	Cryptographic processing of the received secure frame failed.	**0xE4**
MAC_FRAME_TOO_LONG	The received frame or frame resulting from an operation or data request is too long to be processed by the MAC.	**0xE5**
MAC_INVALID_GTS	Unused.	**0xE6**
MAC_INVALID_HANDLE	The purge request contained an invalid handle.	**0xE7**
MAC_INVALID_PARAMETER	The API function parameter is out of range.	**0xE8**
MAC_NO_ACK	The operation or data request failed because no acknowledgement was received.	**0xE9**
MAC_NO_BEACON	The scan request failed because no beacons were received or the orphan scan failed because no coordinator realignment was received.	**0xEA**
MAC_NO_DATA	The associate request failed because no associate response was received or the poll request did not return any data.	**0xEB**
MAC_NO_SHORT_ADDRESS	The short address parameter of the start request was invalid.	**0xEC**
MAC_OUT_OF_CAP	Unused.	**0xED**
MAC_PAN_ID_CONFLICT	PAN identifier conflict has been detected and communicated to the PAN coordinator.	**0xEE**
MAC_REALIGNMENT	A coordinator realignment command has been received.	**0xEF**
MAC_TRANSACTION_EXPIRED	The associate response, disassociate request, or indirect data transmission failed because the peer device did not respond before the transaction expired or was purged.	**0xF0**
MAC_TRANSACTION_OVERFLOW	The operation failed because MAC data buffers are full.	**0xF1**

MAC_TX_ACTIVE	Unused.	**0xF2**
MAC_UNAVAILABLE_KEY	The operation or data request failed because the security key is not available.	**0xF3**
MAC_UNSUPPORTED_ATTRIBUTE	The set or get request failed because the attribute is not supported.	**0xF4**
MAC_INVALID_ADDRESS	The data request failed because neither the source address nor destination address parameters were present.	**0xF5**
MAC_ON_TIME_TOO_LONG	Unused.	**0xF6**
MAC_PAST_TIME	Unused.	**0xF7**
MAC_TRACKING_OFF	The start request failed because the device is not tracking the beacon of its coordinator.	**0xF8**
MAC_INVALID_INDEX	Unused.	**0xF9**
MAC_LIMIT_REACHED	The scan terminated because the PAN descriptor storage limit was reached.	**0xFA**
MAC_READ_ONLY	A set request was issued with a read-only identifier.	**0xFB**
MAC_SCAN_IN_PROGRESS	The scan request failed because a scan is already in progress.	**0xFC**
MAC_SUPERFRAME_OVERLAP	The beacon start time overlapped the coordinator transmission time.	**0xFD**
MAC_AUTOACK_PENDING_ALL_ON	The AUTOPEND pending all is turned on.	**0xFE**
MAC_AUTOACK_PENDING_ALL_OFF	The AUTOPEND pending all is turned off.	**0xFF**

Table 7: MAC Status Values

4.2 *MAC PIB Attribute ID Values*

NAME	VALUE	DATA TYPE	ACCESS
MAC_ACK_WAIT_DURATION	**0x40**	uint8	R/W
MAC_ASSOCIATION_PERMIT	**0x41**	bool	R/W
MAC_AUTO_REQUEST	**0x42**	bool	R/W
MAC_BATT_LIFE_EXT	**0x43**	bool	R/W
MAC_BATT_LEFT_EXT_PERIODS	**0x44**	uint8	R/W
MAC_BEACON_PAYLOAD	**0x45**	uint8[16]	R/W
MAC_BEACON_PAYLOAD_LENGTH	**0x46**	uint8	R/W
MAC_BEACON_ORDER	**0x47**	uint8	R/W
MAC_BEACON_TX_TIME	**0x48**	uint32	R/W
MAC_BSN	**0x49**	uint8	R/W
MAC_COORD_EXTENDED_ADDRESS	**0x4A**	uint8[8]	R/W
MAC_COORD_SHORT_ADDRESS	**0x4B**	uint16	R/W
MAC_DSN	**0x4C**	uint8	R/W
MAC_GTS_PERMIT	**0x4D**	bool	R/W
MAC_MAX_CSMA_BACKOFFS	**0x4E**	uint8	R/W
MAC_MIN_BE	**0x4F**	uint8	R/W
MAC_PAN_ID	**0x50**	uint16	R/W
MAC_PROMISCUOUS_MODE	**0x51**	bool	R/W
MAC_RX_ON_WHEN_IDLE	**0x52**	bool	R/W
MAC_SHORT_ADDRESS	**0x53**	uint16	R/W
MAC_SUPERFRAME_ORDER	**0x54**	uint8	R/W
MAC_TRANSACTION_PERSISTENCE_TIME	**0x55**	uint16	R/W
MAC_ASSOCIATED_PAN_COORD	**0x56**	bool	R/W
MAC_MAX_BE	**0x57**	uint8	R/W
MAC_FRAME_TOTAL_WAIT_TIME	**0x58**	uint16	R/W
MAC_MAX_FRAME_RETRIES	**0x59**	uint8	R/W
MAC_RESPONSE_WAIT_TIME	**0x5A**	uint8	R/W
MAC_SYNC_SYMBOL_OFFSET	**0x5B**	uint8	R/W
MAC_TIMESTAMP_SUPPORTED	**0x5C**	bool	R/W
MAC_SECURITY_ENABLED	**0x5D**	bool	R/W
MAC_EBSN	**0x5E**	uint8	R/W
MAC_EBEACON_ORDER	**0x5F**	uint8	R/W
MAC_EBEACON_ORDER_NBPAN	**0x60**	uint16	R/W
MAC_OFFSET_TIMESLOT	**0x61**	uint8	R/W
MAC_INCLUDE_MPMIE	**0x62**	bool	R/W
MAC_PHY_FSK_PREAMBLE_LEN	**0x63**	uint8	R/W
MAC_PHY_MRFSKSFD	**0x64**	uint8	R/W
MAC_PHY_TRANSMIT_POWER_SIGNED	**0xE0**	int8	R/W

MAC_LOGICAL_CHANNEL	**0xE1**	uint8	R/W
MAC_EXTENDED_ADDRESS	**0xE2**	uint8[8]	R/W
MAC_ALT_BE	**0xE3**	uint8	R/W
MAC_DEVICE_BEACON_ORDER	**0xE4**	uint8	R/W
MAC_RF4CE_POWER_SAVINGS	**0xE5**	uint8	R/W
MAC_FRAME_VERSION_SUPPORT	**0xE6**	uint8	R/W
MAC_CHANNEL_PAGE	**0xE7**	uint8	R/W
MAC_PHY_CURRENT_DESCRIPTOR_ID	**0xE8**	uint8	R/W
MAC_FCS_TYPE	**0xE9**	bool	R/W

Table 8: MAC PIB Attribute ID Values

4.3 *Frequency Hopping PIB Attribute ID Values*

NAME	VALUE	DATA TYPE	ACCESS
MAC_FHPIB_TRACK_PARENT_EUI	**0x2000**	uint8[8]	R/W
MAC_FHPIB_BC_INTERVAL	**0x2001**	uint32	R
MAC_FHPIB_UC_EXCLUDED_CHANNELS	**0x2002**	uint8[17]	R/W
MAC_FHPIB_BC_EXCLUDED_CHANNELS	**0x2003**	uint8[17]	R/W
MAC_FHPIB_UC_DWELL_INTERVAL	**0x2004**	uint8	R/W
MAC_FHPIB_BC_DWELL_INTERVAL	**0x2005**	uint8	R
MAC_FHPIB_CLOCK_DRIFT	**0x2006**	uint8	R
MAC_FHPIB_TIMING_ACCURACY	**0x2007**	uint8	R
MAC_FHPIB_UC_CHANNEL_FUNCTION	**0x2008**	uint8	R/W
MAC_FHPIB_BC_CHANNEL_FUNCTION	**0x2009**	uint8	R/W
MAC_FHPIB_USE_PARENT_BS_IE	**0x200A**	uint8	R
MAC_FHPIB_BROCAST_SCHED_ID	**0x200B**	uint16	R
MAC_FHPIB_UC_FIXED_CHANNEL	**0x200C**	uint16	R/W
MAC_FHPIB_BC_FIXED_CHANNEL	**0x200D**	uint16	R/W
MAC_FHPIB_PAN_SIZE	**0x200E**	uint16	R/W
MAC_FHPIB_ROUTING_COST	**0x200F**	uint8	R/W
MAC_FHPIB_ROUTING_METHOD	**0x2010**	uint8	R/W
MAC_FHPIB_EAPOL_READY	**0x2011**	uint8	R/W
MAC_FHPIB_FAN_TPS_VERSION	**0x2012**	uint8	R/W
MAC_FHPIB_NET_NAME	**0x2013**	uint8[32]	R/W
MAC_FHPIB_PAN_VERSION	**0x2014**	uint16	R/W
MAC_FHPIB_GTK_0_HASH	**0x2015**	uint8[8]	R/W
MAC_FHPIB_GTK_1_HASH	**0x2016**	uint8[8]	R/W
MAC_FHPIB_GTK_2_HASH	**0x2017**	uint8[8]	R/W
MAC_FHPIB_GTK_3_HASH	**0x2018**	uint8[8]	R/W
MAC_FHPIB_NEIGHBOR_VALID_TIME	**0x2019**	uint16	R/W

Table 9: Frequency Hopping PIB Attribute ID Values

4.4 *Security PIB Attribute ID Values*

NAME	VALUE	DATA TYPE	ACCESS
MAC_KEY_TABLE	**0x71**	***See 5.3.7***	W
MAC_KEY_TABLE_ENTRIES	**0x81**	uint8	R/W
MAC_DEVICE_TABLE_ENTRIES	**0x82**	uint8	R/W
MAC_SECURITY_LEVEL_TABLE_ENTRIES	**0x83**	uint8	R/W
MAC_FRAME_COUNTER	**0x84**	uint32	none
MAC_AUTO_REQUEST_SECURITY_LEVEL	**0x85**	uint8	R/W
MAC_AUTO_REQUEST_KEY_ID_MODE	**0x86**	uint8	R/W
MAC_AUTO_REQUEST_KEY_SOURCE	**0x87**	uint8[8]	R/W
MAC_AUTO_REQUEST_KEY_INDEX	**0x88**	uint8	R/W
MAC_DEFAULT_KEY_SOURCE	**0x89**	uint8[8]	R/W
MAC_PAN_COORD_EXTENDED_ADDRESS	**0x8A**	uint8[8]	R/W
MAC_PAN_COORD_SHORT_ADDRESS	**0x8B**	uint16	R/W
MAC_KEY_ID_LOOKUP_ENTRY	**0xD0**	***See 5.3.1***	R/W
MAC_KEY_ID_DEVICE_ENTRY	**0xD1**	***See 5.3.2***	R/W
MAC_KEY_ID_USAGE_ENTRY	**0xD2**	***See 5.3.3***	R/W
MAC_KEY_ENTRY	**0xD3**	***See 5.3.4***	R/W
MAC_DEVICE_ENTRY	**0xD4**	***See 5.3.5***	R/W
MAC_SECURITY_LEVEL_ENTRY	**0xD5**	***See 5.3.6***	R/W

Table 10: MAC Security PIB Attribute ID Values

4.4.1 Security PIB Structure: MAC Key ID Lookup Entry

Attribute	Length	Description
Index1	2	Key index
Index2	2	Key ID lookup index
LookupData	9	Data array used to identify the key
LookupSize	1	Data size indicator: 0x00=5 octets, 0x01-9 octets

4.4.2 Security PIB Structure: MAC Key ID Device Entry

Attribute	Length	Description
Index1	2	Key index
Index2	2	Key ID device index
Handle	2	Handle of the device descriptor
Unique	1	TRUE=link key, FALSE=group key
BlackListed	1	TRUE=this key exhausted frame counter

4.4.3 Security PIB Structure: MAC Key ID Usage Entry

Attribute	Length	Description
Index1	2	Key index
Index2	2	Key ID usage index
FrameType	1	Frame type
FrameId	1	Command frame identifier

4.4.4 Security PIB Structure: MAC Key Entry

Attribute	Length	Description
Index1	2	Key index
Index2	2	*Not Used*
KeyEntry	16	Array of bytes for key entry
FrameCounter	4	Frame counter for this key

4.4.5 Security PIB Structure: MAC Device Entry

Attribute	Length	Description
Index1	2	Device index
Index2	2	*Not Used*
PanId	2	Device PAN Id
ShortAddr	2	Device 16-bit address
ExtAddr	8	Device 64-bit address
Exempt	1	TRUE=device can override min security settings
FrameCounter1	4	4 byte frame counter value corresponding to 1st key used by device
Key Idx1	2	2 byte Key Index of the 1st Key for which corresponding frame counter value is to be monitored
FrameCounter2	4	4 byte frame counter value corresponding to 2nd Key used by device
Key Idx2	2	2 byte Key Index of the 2nd Key for which corresponding frame counter value is to be monitored
⋮		
FrameCounterN	4	4 byte frame counter value corresponding to Nth Key used by device
Key IdxN	2	2 byte Key Index of the Nth Key for which corresponding frame counter value is to be monitored

4.4.6 Security PIB Structure: MAC Security Level Entry

Attribute	Length	Description
Index1	2	Security level index
Index2	2	*Not Used*
FrameType	1	Frame type

FrameId	1	**Command frame identifier**
MinSecurity	1	**Minimum expected/required security level for incoming MAC frames**
MinSecurityOverride	1	**TRUE=originating exempt devices can use security level of zero**

4.4.7 Security PIB Structure: MAC Key Table

Attribute	Length	Description
Index1	2	***Not Used***
Index2	2	***Not Used***
Data	0	**Writing to his PIB item initializes the MAC Key Descriptor Table**

5. Document History

Revision	Date	Description/Changes
1.0	2016-06-28	Initial version

6. References

[R1] CC1310 Datasheet: http://www.ti.com/lit/ds/symlink/cc1310.pdf

[R2] CC13xx,CC26xx Technical Reference Manual: http://www.ti.com/lit/ug/swcu117f/swcu117f.pdf

[R3] NPI Users's Guide: <SDK as installed>\docs\NPI User's Guide.pdf

[R4] TI-15.4 Stack Developer's Guide: <SDK as installed>\docs\TI-15.4 Stack Developers Guide.pdf