MAC co - Processor

Accelerate your Network Development

Applications

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

Description

The MAC co-Processor on CC253x and CC26xx is a cost effective, low power, MAC co-Processor that provides IEEE 802.15.4 Implementation via a minimal development effort.

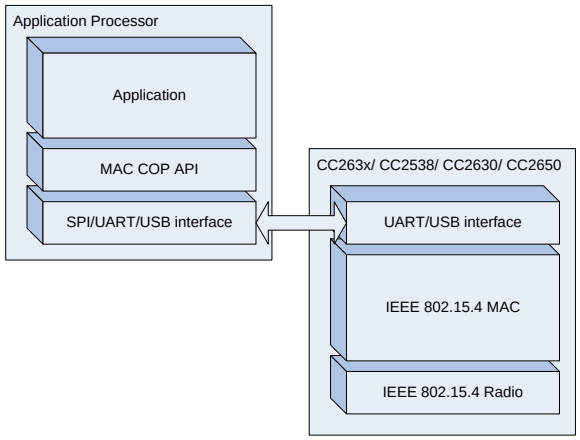
MAC co-Processor is an entity which implements the MAC – 802.15.4-2006 IEEE standard in a dedicated system on a chip and provides a serial interface to an external processor for control and processing of the co-processor operations.

MAC co-Processor approach is a scalable architecture split that fits perfectly for configurations where the host co-processor runs protocol stack layers over IEEE 802.15.4(g/e) MAC/PHY (ex - generic IP over 6LowPAN, ZigBee IP or ZigBee Pro) or even an application that wants to simply use the MAC/PHY as data link.

The CC253x MAC co-Processor interfaces to any microcontroller through UART, or USB interface. For ex, it can be combined with an MSP430, or Stellaris ARM Cortex-M3 microcontroller. The CC26xx MAC co-Processor currently supports UART only.

The frames transported over the physical serial link follow the format specified in this document.

MAC co-Processor Architecture makes it easy for the users to add IEEE 802.15.4 functionality to a existing product and also provides great flexibility in choice of microcontrollers.



Key Features

- *UART or USB interface to application processor.*

Index

Table of Figures

1. Pin configuration

1.1 *CC2630/CC2650*

The SimpleLink™ CC2630 MCU (and equivalently the SimpleLink™ CC2650) is the newest member of the family MAC cOP platforms. On CC2630, MAC cOP includes some significant differences when compared with the other established CC253x MAC cOP platforms:

- TI-RTOS: As with all MAC software products on CC2630, it is built on top of TI-RTOS, a Real-Time Operating System developed by Texas Instruments.
- NPI: On CC2630, the MAC 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. ZigBee, BLE, MAC) 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.

  The following sections for CC2630 are provided as a simple summary. For more information on the NPI-based MAC coP for CC2630, please refer to [5].

  NOTE: At this time, CC2630 MAC coP only supports the UART NPI transport.

1.1.1 Network Processor Signals

The figure below shows how an application processor interfaces with the CC2630 MAC coP.

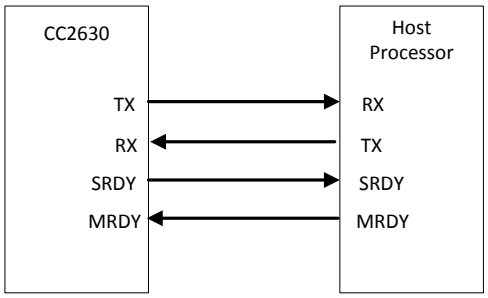


Figure 1 CC2630 Interface

The CC2630-MAC 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 CC2630.

- SRDY: This active low signal is asserted by the CC2630 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 [5] for details
- MRDY: This active low signal is asserted by the application processor for power management and transaction control. Please refer to [5] for details.

Pin Configuration

The Pin Configuration for MAC coP on CC22630 is defined in the following table. Note that MAC coP supports three different package sizes for the CC2630:

Type	CC2630-MAC coP signal	CC2630 7x7 PIN	CC2630 5x5 PIN	CC2630 4x4 PIN	*Direction (on C2630)*
POWER_SAVING	SRDY	DIO_12	DIO_4	DIO_3	*Out*
POWER_SAVING	MRDY	DIO_19	DIO_6	DIO_4	*In*
UART	TX	DIO_3	DIO_0	DIO_2	*Out*
UART	RX	DIO_2	DIO_1	DIO_1	*In*

1.1.2 Interface Configuration

IAR Project Configuration

The CC2630-MAC coP IAR project that is included in the MAC software package currently only supports UART for network processor host connectivity. Go to *Project->Options->C/C++ Compiler->Preprocessor->Defined Symbols* and ensure the following are defined:

NPI - enables the new NPI subsystem

NPI_USE_UART - enables the UART transport layer of the NPI subsystem

NPI_UART_BR=11520 - configures the baud rate of the UART interface. This can be modified to suit your project's requirements.

NPI_SREQRSP - enables support for Synchronous REQ/RSP messaging.

POWER_SAVING - configures support for power savings operation which requires the connection and use of the MRDY and SRDY signals.

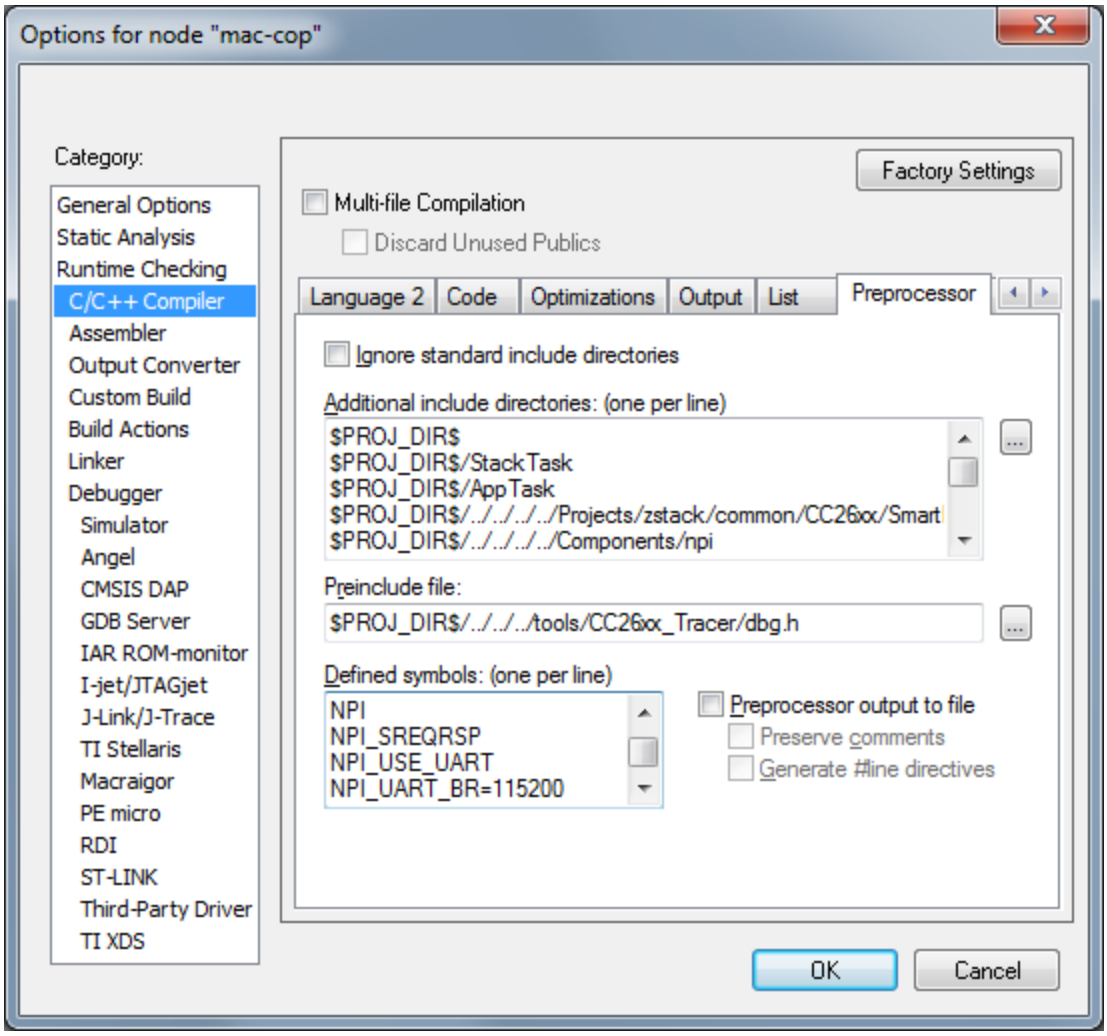


Figure 2 IAR Setup for CC2630 MAC coP

1.2 *CC253X*

The figure below shows how an application processor interfaces with the CC2530.

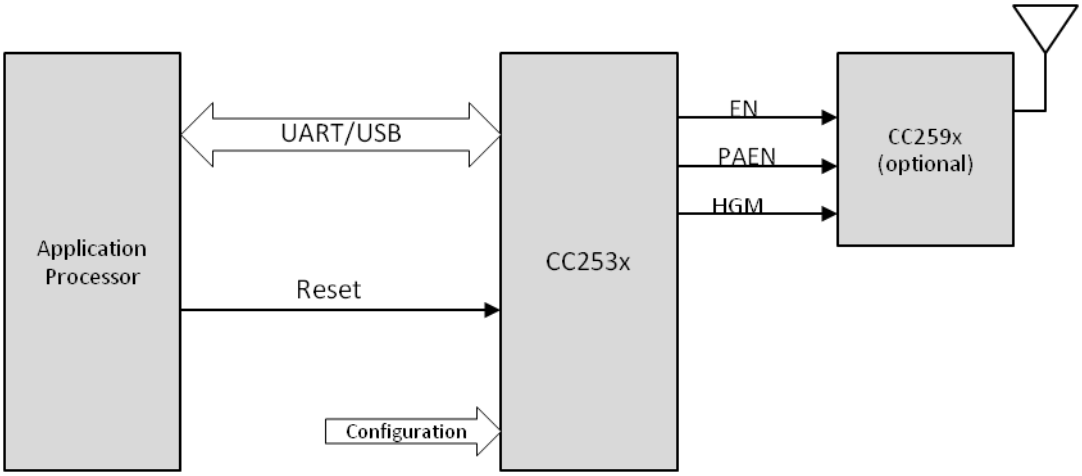


Figure 3 CC2530 Interface

1.2.1 Network processor signals

The MAC coP uses the following signals for the hardware interface

- **RX/TX/RT/CT for UART:** These are the standard signals used for UART communication. See 3.1.3 (for UART) for details. This naming convention is consistent with [R1].
- **RESET:** This signal is used by the application processor to reset the CC2530.
- **PAEN, EN, HGM:** These signals are used to control the CC259x PA/LNA and should be connected to the appropriate pins on the CC259x. See [R4] for details on the CC259x.
- **CFG0:** The signal on this pin is used to configure the CC2530-MAC coP. The CC2530-MAC coP reads these signals at power up and configures its operation accordingly. See section 2.1.2 for details.

2. CC253x-MAC co-Processor physical Interface

The CC2530-MAC coP supports UART, or USB interface to the application processor.

2.1 *CC253x- MAC co-Processor default configuration*

2.1.1 IAR project configuration

The CC2530-MAC coP IAR project that is included in the TIMAC software package has two project configurations – CC2530-MAC coP and CC2531-MACcoP. As the name indicates, the configurations are intended for use with the CC2530 and CC2531 (USB) chips.

2.1.2 Configuration pins

The CC253x-MAC coP project reads the two hardware configuration pins at power-up and configures itself accordingly.

The CFG0 pin is used to indicate the presence (if pin is high) or absence of the 32kHz crystal connected to the CC253x-MAC coP. This is the sleep crystal that is used to maintain accurate timing when the device is in sleep mode. The advantage of using this instead of the internal 32kHz oscillator is that it typically provides faster wakeup time for sleep and a lower power consumption during this time. If this crystal is not populated, then the CC2530 can use the internal RC oscillator.

2.1.2.1 UART pin configuration

CC2530-MAC coP signal	CC2530 PIN	CC2530 NAME	*Direction (on C2530)*
SS / CT	15	P0_4	*In*
C / RT	14	P0_5	*In / Out*
MO / TX	16	P0_3	*In / Out*
MI / RX	17	P0_2	*Out / In*
RESET	20	RESET_N	*In*
PAEN	9	P1_1	*Out*
EN	6	P1_4	*Out*
HGM	12	P0_7	*Out*
CFG0	8	P1_2	*In*

2.1.3 USB pin configuration

This is only available when used with the CC2531 chip. In this configuration, the CC2530-MAC coP will use the USB transport with the UART pin configuration. The pin-out of the CC2531 can be found in the datasheet [R3]. The USB transport exposes the CDC (communication device class) class USB interface and exposes a virtual COM port to the host. The host processor would then access this device as a regular COM port device and communicate with the MAC coP using the UART Transport.

3. UART Transport

3.1.1 Configuration

The following UART configuration is supported:

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

3.1.2 Frame Format

UART transport frame format is shown in the following figure. The left-most field is transmitted first over the wire.

Bytes: 1	3-253	1
SOF	General format frame	FCS

Figure 4 UART Transport Frame Format

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

General frame format: This is the general frame format as described in 3.2.

FCS: Frame-check sequence. This field is computed as an XOR of all the bytes in the general format frame fields.

Shown below is a C example for the FCS calculation:

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

3.1.3 Signal Description

The following standard UART signals are used:

- TX: Transmit data.

- RX: Receive data.
- CT: Clear to send.
- RT: Ready to send.
- The MRDY and SRDY signals are not used with UART transport.

Figure 6 shows the RTS/CTS flow control connections to the host processor. On the CC2530, RT and CT are active-low signals. The RT output is driven low when the receive register is empty and reception is enabled. Transmission of a byte does not occur before the CT input goes low.

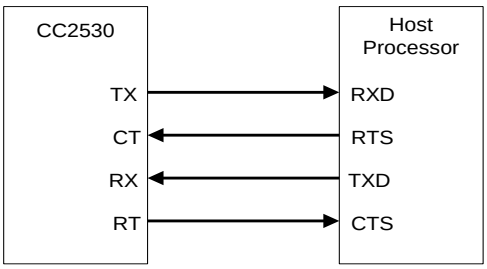


Figure 5 RTS/CTS Flow Control Connections

3.1.4 Signal Operation

UART transport sends and receives data asynchronously. Data can be sent and received simultaneously and the transfer of a frame can be initiated at any time by either the application processor or the CC2530.

3.2 *Monitor and Test Frame Format*

The general frame format for a frame between the host the MAC co-Processor is shown in the following figure. The left-most field is transmitted first over the wire. For multi-byte fields, the lowest order byte is transmitted first.

Bytes: 1	2	0-250
Length	Command	Data

Figure 6 General Frame Format

Length: The length of the data field of the frame. The length can range from 0-250.

Command: The command of the frame.

Data: The frame data. This depends on the command field and is described for each command in Section -4-.

3.2.1 Command Field

The command field is constructed of two bytes. The bytes are formatted as shown in the following figure. The Cmd0 byte is transmitted first.

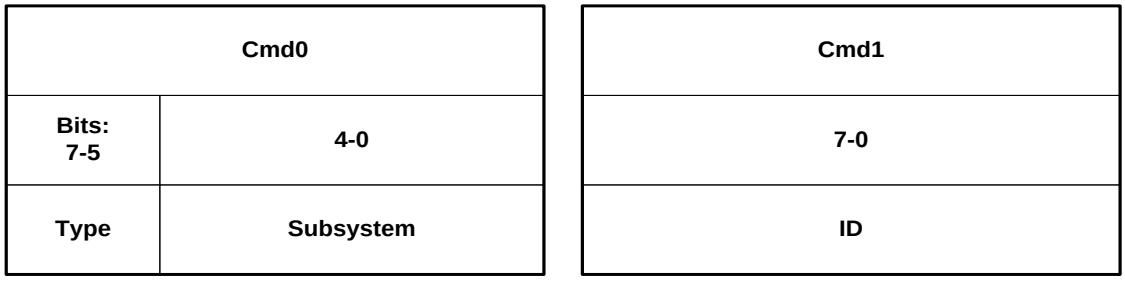


Figure 7 Command Field

Type: The command type has one of the following values:

- 1: SREQ: A synchronous request that requires an immediate response. For example, a function call with a return value would use an SREQ command.
- 2: AREQ: An asynchronous request. For example, a callback event or a function call with no return value would use an AREQ command.
- 3: SRSP: A synchronous response. This type of command is only sent in response to a SREQ command. For an SRSP command the subsystem and ID are set to the same values as the corresponding SREQ. The length of an SRSP is generally nonzero, so an SRSP with length=0 can be used to indicate an error.
- 4-7: Reserved.

Subsystem: The subsystem of the command. Values are shown below:

Subsystem Value	Subsystem Name
0	RPC Error interface
1	SYS interface
2	MAC Interface
3	Reserved
4	Reserved

5	Reserved
6	Reserved
7	UTIL interface
8-32	Reserved

ID: The command ID. The ID maps to a particular interface message. Value range: 0-255.

When the MAC co-Processor cannot recognize an SREQ command from the host processor, the following SRSP is returned:

SRSP:

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

Attributes:

<table>
<tr><th>Attribute</th><th>Length (byte)</th><th>Description</th></tr>
<tr><td>ErrorCode</td><td>1</td><td>The error code maps to one of the following enumerated values.

<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></table></td></tr>
<tr><td>ReqCmd0</td><td>1</td><td>The Cmd0 value of the processed SREQ</td></tr>
<tr><td>ReqCmd1</td><td>1</td><td>The Cmd1 value of the processed SREQ</td></tr>
</table>

4. Setting up a TIMAC Network

4.1 Starting a PAN Coordinator

To Start a PAN coordinator following sequence of commands can be used to the MAC-coP from the Host via the MT interface

1. Reset the MAC using the API : MAC_RESET_REQ
2. Set the parameters using the API: MAC_SET_REQ
    a. Set the Short Address attributeID: ZMAC_SHORT_ADDRESS
    b. Set the Extended Address attributeID :ZMAC_EXTENDED_ADDRESS
3. Scan the area for pre-existing PAN ID's. send a MAC_SCAN_REQ
4. After receiving MAC_SCAN_CNF (AREQ) determine the existing PANID's and select a unique PAN ID to start the network.
5. Start the MAC network using MAC_START_REQ
6. Set the parameters: MAC_SET_REQ
    a. Allow Association: ZMAC_ASSOCIATION_PERMIT
    b. Device On All the time : ZMAC_RX_ON_IDLE, set true for coordinator

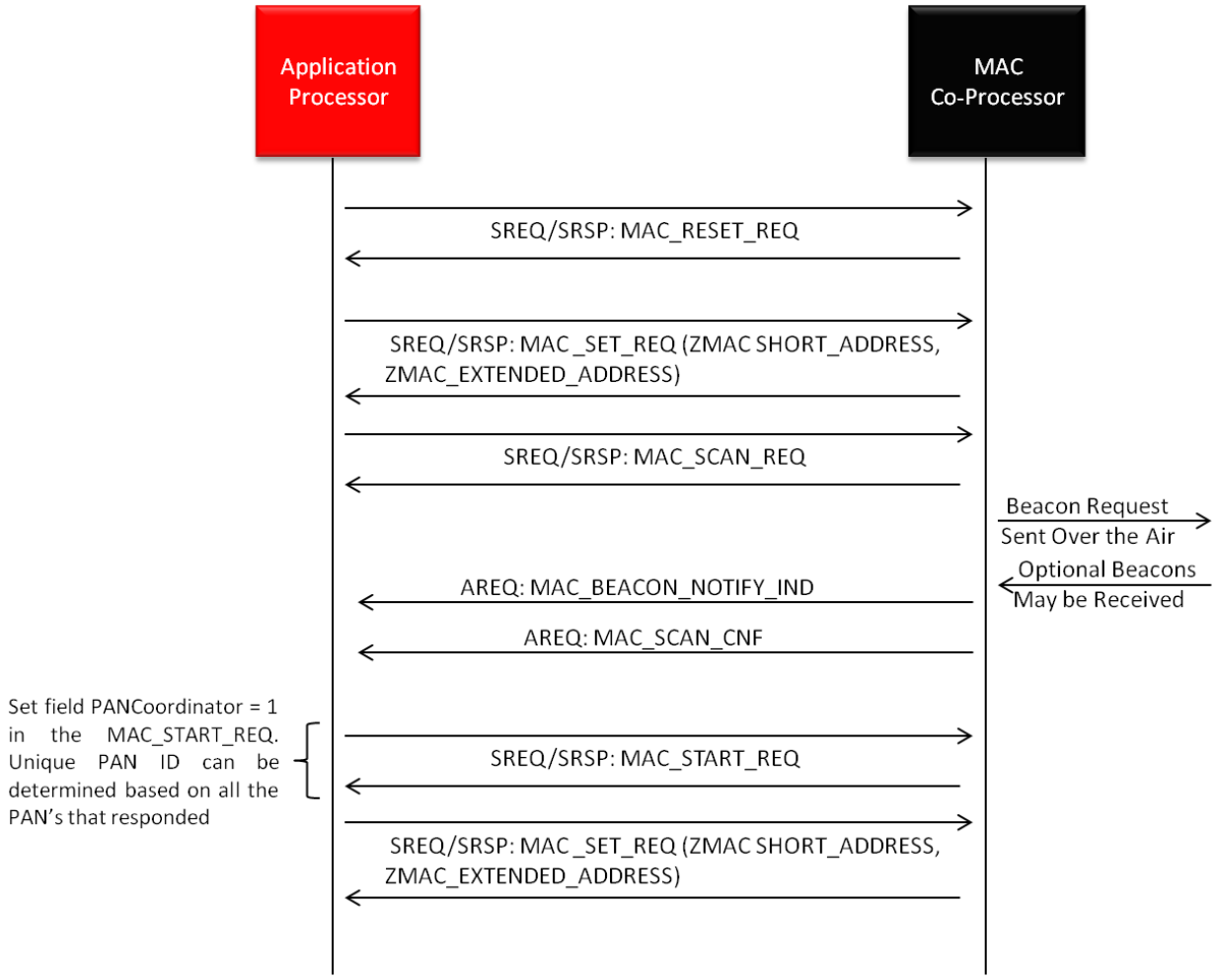


Figure 8: Flow Diagram showing the commands to send to the host and receive from the MAC co-Processor to start the Network

4.2 Starting Network Devices

To introduce a device(s) in the network, following sequence of commands can be used to the MAC-coP from the Host via the MT interface

1. Reset the MAC State Machine using the API: MAC_RESET_REQ
2. After the reset. Set the following parameters using the API : MAC_SET_REQ
    a. Set the Extended Address : ZMAC_EXTENDED_ADDRESS
    b. Set RX On when idle = true
3. Scan the area in the radio range for existing PAN's using the API MAC_SCAN_REQ. Various beacons will be received via the indication from the MAC – MAC_BEACON_NOTIFY_IND. Select an appropriate PAN to join after MAC_SCAN_CNF is received from the MAC which indicates that the network discovery has completed.

4. Then send the association request to the selected PAN Coordinator using the API: ZMAC_ASSOCIATE_REQ

On the PAN coordinator when MAC Association request is received send the association response

1. ZMAC_ASSOCIATE_IND is received when a device requests permission to connect.
    a. Send a association response message using the API: MAC_ASSOICATE_RSP with format as suggested in section 4.

You can then use MAC_DATA_REQ API to send data. On the receiving data from a network device a MAC_DATA_IND will be received to the application processor.

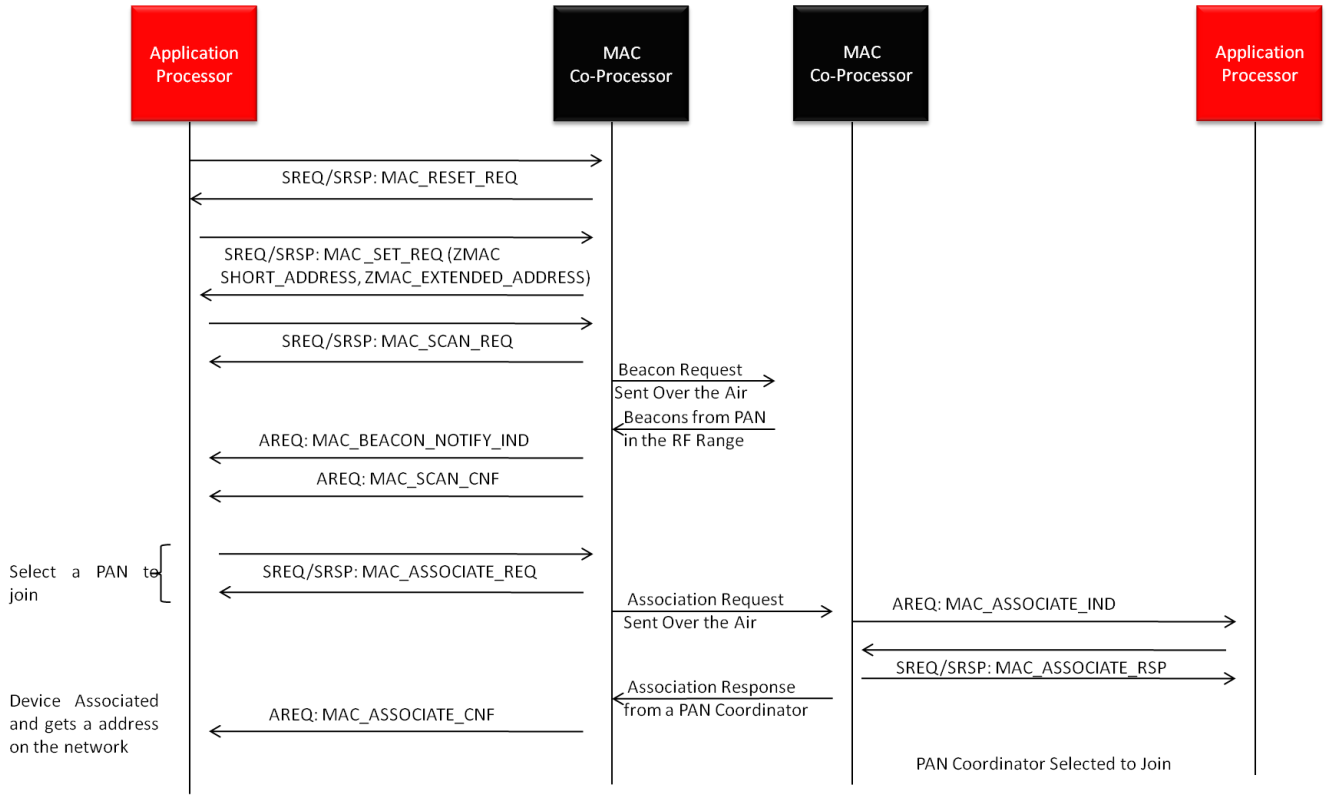


Figure 9: Flow Diagram showing the commands join a network using a MAC co-Processor Configuration. Also shows the commands exchange at the PAN coordinator.

5. MAC coP Software Command Interface

The CC253x MAC co-Processor Software command interface uses the MT MAC interface and the MT UTIL Interface. The API's allow the developers to implement various functionalities for deploying an IEEE 802.15.4 based network using a host controlling the MAC coP. This section below lists the API calls.

5.1 MT MAC Initialization Interface

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

5.1.1 MAC_INIT

Description:

This API initializes the MAC subsystem. It must be called once when the software system is started and before any other MAC API is called. This is called up within MAC coP automatically on the MAC coP statup in the main function. So this is not required at first start up from the Application.

Usage:

SREQ:

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

Attributes: None

SRSP:

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

Attributes:

Attribute	Length (byte)	Description
Status	1	Status is either Success (0) or Failure (1).

5.2 MT MAC Data Interface

This interface provides API's to send and receive the data between Application and the MAC coP.

5.2.1 MAC_DATA_REQ

Description:

This API is used to send application data to the MAC coP for transmission.

The MAC 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.

All MAC Security related API parameters can be currently set to zero. They are all don't care in the currently MAC coP implementation.

Maximum application data length without security is 116 Bytes.

Usage:

SREQ:

Byte: 1	1	1	1	8	2
Length = 0x1-0x74	**Cmd0 = 0x22**	**Cmd1 = 0x05**	**DestAddressMode**	**DestAddress**	**DestPanId**

Byte: 1	1	1	1	1	8	1	1
SrcAddressMode	**Handle**	**TxOption**	**LogicalChannel**	**Power**	**KeySource**	**SecurityLevel**	**KeyIdMode**

Byte: 1	1	0-250
KeyIndex	**MSDULength**	**MSDU**

Attributes:

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

<table>
<tr><td>TxOption</td><td>1</td><td>Transmitting options:

<table>
<tr><th>Option</th><th>Value</th><th>Description</th></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_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>LogicalChannel</td><td>1</td><td>Transmit the data frame on this channel. This value is ignored if TxOption MAC_TXOPTION_PWR_CHAN is not used.</td></tr>
<tr><td>Power</td><td>1</td><td>Transmit the data frame at this power level. This value is ignored if TxOption MAC_TXOPTION_PWR_CHAN is not used.</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>
<tr><td>MSDULength</td><td>1</td><td>Length of the data.</td></tr>
<tr><td>MSDU</td><td>0-250</td><td>Actual data that will be sent.</td></tr>
</table>

SRSP:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x62**	**Cmd1 = 0x05**	**Status**

Attributes:

Attribute	Length (byte)	Description
Status	**1**	**Status is either Success (0) or Failure (1).**

5.2.1 MAC_PURGE_REQ

Description:
This API is used to send a request the purge of a data frame from the MAC coP data Queue.

Usage:
SREQ:

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

Attributes:

Attribute	Length (byte)	Description
MsduHandle	**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 (byte)	Description
Status	**1**	**Status is either Success (0) or Failure (1).**

5.2.2 MAC_DATA_CNF

Description:

This command is sent by the MAC 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 a error status value will indicate the reason for failure.

Usage:

AREQ:

1	1	1	1	1	4	2
Length = 0x08	**Cmd0 = 0x42**	**Cmd1 = 0x84**	**Status**	**Handle**	**Timestamp**	**Timestamp2**

Attributes:

Attribute	Length (byte)	Description
Status	1	**This field indicates the status of the MAC_DATA_REQ operation.** **NAME** / **DESCRIPTION** **MAC_SUCCESS** — **Operation successful.** **MAC_CHANNEL_ACCESS_FAILURE** — **Data transmission failed because of congestion on the channel.** **MAC_FRAME_TOO_LONG** — **Data is too long to be processed by the MAC.**

		MAC_INVALID_PARAMETER	**The API parameter is out of range.**
		MAC_NO_ACK	**No acknowledgement was received from the peer device.**
		MAC_TRANSACTION_EXPIRED	**Indirect data transmission failed because the peer device did not respond before the transaction expired or was purged.**
		MAC_TRANSACTION_OVERFLOW	**MAC data buffers are full.**
		MAC_COUNTER_ERROR	**The outgoing secured frame has a frame counter value 0xffffffff.**
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.**	

5.2.3 **MAC_DATA_IND**

Description:

This callback message transfers the incoming data from the MAC coP to the application.

Usage:

AREQ:

1	1	1	1	8	1	8
Length = 0x01-0x74	**Cmd0 = 0x42**	**Cmd1 = 0x85**	**SrcAddrMode**	**SrcAddr**	**DstAddrMode**	**DstAddr**

4	2	2	2	1	1	1
Timestamp	**Timestamp2**	**SrcPanId**	**DstPanId**	**LinkQuality**	**Correlation**	**RSSI**

1	8	1	1	1	1	0-128
DSN	**KeySource**	**SecurityLevel**	**KeyIdMode**	**KeyIndex**	**Length**	**Data**

Attributes:

Attribute	Length (byte)	Description
SrcAddrMode	1	**Source address mode** Mode / Value / Description ADDRESS_16_BIT / 0x02 / Address 16 bit ADDRESS_64_BIT / 0x03 / Address 64 bit
SrcAddr	8	**Source address**
DstAddrMode	1	**Destination address mode**
DstAddr	8	**Destination address**
Timestamp	4	**The time, in *aUnitBackoffPeriod* units, at which the frame was received.**
Timestamp2	2	**The time, in internal MAC timer units, at which the frame was received.**
SrcPanId	2	**Pan Id of the source address**
DstPanId	2	**Pan Id of the destination address**
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.**
DSN	1	**Data sequence number of received frame**
KeySource	8	**Key Source of this data frame.**
SecurityLevel	1	**Security Level of this data frame:** Security Level: Value NO_SECURITY: 0x00 MIC_32_AUTH: 0x01 MIC_64_AUTH: 0x02 MIC_128_AUTH: 0x03 AES_ENCRYPTION: 0x04 AES_ENCRYPTION_MIC_32: 0x05 AES_ENCRYPTION_MIC_64: 0x06 AES_ENCRYPTION_MIC_128: 0x07
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.**
Length	1	**Data length**
Data	**0-128**	**Data**

5.2.4 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 = 0x01	**Cmd0 = 0x42**	**Cmd1 = 0x9A**	**Status**	**Handle**

Attributes:

Attribute	Length (byte)	Description
Status	1	**This field indicates status of the MAC_PURGE_REQ** NAME: DESCRIPTION MAC_SUCCESS: Operation successful. MAC_INVALID_HANDLE: The purge request contained an invalid handle.
Handle	**1**	**Application defined handle of the message**

5.3 MT MAC Management Interface

Following API's are used for 802.15.4 network management.

5.3.1 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 MAC coP sends a MAC_ASSOCIATE_CNF to the application.

Usage:

SREQ:

Byte: 1	1	1	1	1	1
Length = 0x12	**Cmd0 = 0x22**	**Cmd1 = 0x06**	**LogicalChannel**	**ChannelPage**	**CoordAddressMode**

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

Attributes:

Attribute	Length (byte)	Description
LogicalChannel	1	**Channel on which to attempt association**
ChannelPage	1	**The channel page to be used.**
CoordAddressMode	1	**Specifies the format of the coordinator address.** Mode / Value / Description ADDRESS_16_BIT / 0x02 / Address 16 bit ADDRESS_64_BIT / 0x03 / Address 64 bit
CoordAddress	8	**Address of the Coordinator.**
CoordPanId	2	**PAN Id of the Coordinator.**

<table>
<tr><td>CapabilityInformation</td><td>1</td><td>Bit map which specifies the operational capabilities of the device.

<table>
<tr><th>NAME</th><th>DESCRIPTION</th></tr>
<tr><td>MAC_CAPABLE_PAN_COORD</td><td>Device is capable of becoming a PAN coordinator.</td></tr>
<tr><td>MAC_CAPABLE_FFD</td><td>Device is an FFD.</td></tr>
<tr><td>MAC_CAPABLE_MAINS_POWER</td><td>Device is mains powered rather than battery powered.</td></tr>
<tr><td>MAC_CAPABLE_RX_ON_IDLE</td><td>Device has its receiver on when idle.</td></tr>
<tr><td>MAC_CAPABLE_SECURITY</td><td>Device is capable of sending and receiving secured frames.</td></tr>
<tr><td>MAC_CAPABLE_ALLOC_ADDR</td><td>Request allocation of a short address in the associate procedure.</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>

SRSP:

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

Attributes:

Attribute	Length (byte)	Description
Status	**1**	**Status is either Success (0) or Failure (1).**

5.3.2 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 MAC 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 = 0x0B	**Cmd0 = 0x42**	**Cmd1 = 0x50**	**ExtAddr**	**AssocShortAddress**	**AssocStatus**

Attributes:

Attribute	Length (byte)	Description
ExtAddr	8	**Extended Address of the device requesting association**
AssocShortAddress	2	**Short address for the associated device. Allocated by the coordinator.**
AssocStatus	**1**	**Status of the association:** Status: SUCCESSFUL_ASSOCIATION — Value: 0x00 PAN_AT_CAPACITY — 0x01 PAN_ACCESS_DENIED — 0x02

SRSP:

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

Attributes:

Attribute	Length (byte)	Description
Status	**1**	**Status is either Success (0) or Failure (1).**

5.3.3 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 MAC 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**

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

Attributes:

Attribute	Length (byte)	Description

DeviceAddressMode	1	**Specifies the format of the device address.** Mode / Value / Description ADDRESS_16_BIT / 0x02 / Address 16 bit ADDRESS_64_BIT / 0x03 / Address 64 bit
DeviceAddress	8	**Device Address.**
DevicePanId	2	**Network PAN Id of device.**
DisassociateReason	1	**Reason of disassociation:** Reason / Value RESERVED / 0x00 COOR_WISHES_DEV_LEAVE / 0x01 DEV_WISHES_LEAVE / 0x02
TxIndirect	1	**Set to true if the disassociate notification is to be sent indirectly**
KeySource	8	**Key Source of this data frame.**
SecurityLevel	1	**Security Level of this data frame:** Security Level / Value NO_SECURITY / 0x00 MIC_32_AUTH / 0x01 MIC_64_AUTH / 0x02 MIC_128_AUTH / 0x03 AES_ENCRYPTION / 0x04 AES_ENCRYPTION_MIC_32 / 0x05 AES_ENCRYPTION_MIC_64 / 0x06 AES_ENCRYPTION_MIC_128 / 0x07
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.**

SRSP:

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

Attributes:

Attribute	Length (byte)	Description
Status	**1**	**Status is either Success (0) or Failure (1).**

5.3.4 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**	**Attribute**

Attributes:

Attribute	Length (byte)	Description

Attribute	1	Specifies the MAC PIB Attributes:

MAC PIB Attribute	Value
ZMAC_ACK_WAIT_DURATION	0x40
ZMAC_ASSOCIATION_PERMIT	0x41
ZMAC_AUTO_REQUEST	0x42
ZMAC_BATT_LIFE_EXT	0x43
ZMAC_BATT_LEFT_EXT_PERIODS	0x44
ZMAC_BEACON_MSDU	0x45
ZMAC_BEACON_MSDU_LENGTH	0x46
ZMAC_BEACON_ORDER	0x47
ZMAC_BEACON_TX_TIME	0x48
ZMAC_BSN	0x49
ZMAC_COORD_EXTENDED_ADDRESS	0x4A
ZMAC_COORD_SHORT_ADDRESS	0x4B
ZMAC_DSN	0x4C
ZMAC_GTS_PERMIT	0x4D
ZMAC_MAX_CSMA_BACKOFFS	0x4E
ZMAC_MIN_BE	0x4F
ZMAC_PANID	0x50
ZMAC_PROMISCUOUS_MODE	0x51
ZMAC_RX_ON_IDLE	0x52
ZMAC_SHORT_ADDRESS	0x53
ZMAC_SUPERFRAME_ORDER	0x54
ZMAC_TRANSACTION_PERSISTENCE_TIME	0x55
ZMAC_ASSOCIATED_PAN_COORD	0x56
ZMAC_MAX_BE	0x57
ZMAC_FRAME_TOTAL_WAIT_TIME	0x58
ZMAC__MAC_FRAME_RETRIES	0x59
ZMAC_RESPONSE_WAIT_TIME	0x5A
ZMAC_SYNC_SYMBOL_OFFSET	0x5B
ZMAC_TIMESTAMP_SUPPORTED	0x5C
ZMAC_SECURITY_ENABLED	0x5D
ZMAC_PHY_TRANSMIT_POWER	0xE0
ZMAC_LOGICAL_CHANNEL	0xE1
ZMAC_EXTENDED_ADDRESS	0xE2
ZMAC_ALT_BE	0xE3

SRSP:

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

Attributes:

Attribute	Length (byte)	Description
Status	1	**Status is either Success (0) or Failure (1).**
Data	**16**	**1-16 bytes value of the PIB attribute.**

5.3.5 MAC_SET_REQ

Description:
This command is used to request the MAC coP to write a MAC PIB value.

Usage:

SREQ:

Byte: 1	1	1	1	16

Length = 0x11	Cmd0 = 0x22	Cmd1 = 0x09	Attribute	AttributeValue

Attributes:

Attribute	Length (byte)	Description
Attribute	1	**Specified the MAC PIB Attribute:**
AttributeValue	**16**	**1-16 bytes of the PIB attribute value.**

MAC PIB Attribute	Value
ZMAC_ACK_WAIT_DURATION	0x40
ZMAC_ASSOCIATION_PERMIT	0x41
ZMAC_AUTO_REQUEST	0x42
ZMAC_BATT_LIFE_EXT	0x43
ZMAC_BATT_LEFT_EXT_PERIODS	0x44
ZMAC_BEACON_MSDU	0x45
ZMAC_BEACON_MSDU_LENGTH	0x46
ZMAC_BEACON_ORDER	0x47
ZMAC_BEACON_TX_TIME	0x48
ZMAC_BSN	0x49
ZMAC_COORD_EXTENDED_ADDRESS	0x4A
ZMAC_COORD_SHORT_ADDRESS	0x4B
ZMAC_DSN	0x4C
ZMAC_GTS_PERMIT	0x4D
ZMAC_MAX_CSMA_BACKOFFS	0x4E
ZMAC_MIN_BE	0x4F
ZMAC_PANID	0x50
ZMAC_PROMISCUOUS_MODE	0x51
ZMAC_RX_ON_IDLE	0x52
ZMAC_SHORT_ADDRESS	0x53
ZMAC_SUPERFRAME_ORDER	0x54
ZMAC_TRANSACTION_PERSISTENCE_TIME	0x55
ZMAC_ASSOCIATED_PAN_COORD	0x56
ZMAC_MAX_BE	0x57
ZMAC_FRAME_TOTAL_WAIT_TIME	0x58
ZMAC__MAC_FRAME_RETRIES	0x59
ZMAC_RESPONSE_WAIT_TIME	0x5A
ZMAC_SYNC_SYMBOL_OFFSET	0x5B
ZMAC_TIMESTAMP_SUPPORTED	0x5C
ZMAC_SECURITY_ENABLED	0x5D
ZMAC_PHY_TRANSMIT_POWER	0xE0
ZMAC_LOGICAL_CHANNEL	0xE1
ZMAC_EXTENDED_ADDRESS	0xE2
ZMAC_ALT_BE	0xE3

SRSP:

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

Attributes:

Attribute	Length (byte)	Description
Status	**1**	**Status is either Success (0) or Failure (1).**

5.3.6 MAC_SECURITY_GET_REQ

Description:

This API is used to retrieve a MAC SECURITY PIB value. The attributes listed below have been tested and an example is provided in the Linux MAC Sample Application. Other attributes may be queried as listed in mac_api.h but are not currently supported.

Usage:

SREQ:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x22**	**Cmd1 = 0x30**	**Attribute**

Attributes:

Attribute	Length (byte)	Description
Attribute	1	Specified the MAC SECURITY PIB Attribute: MAC PIB Attribute: 0x7A — MAC_PAN_COORD_EXTENDED_ADDRESS MAC PIB Attribute: 0x7B — MAC_PAN_COORD_SHORT_ADDRESS MAC PIB Attribute: 0xD5 — MAC_SECURITY_LEVEL_ENTRY

SRSP:

Byte: 1	1	1	1	27
Length = 28	Cmd0 = 0x62	Cmd1 = 0x30	Status	Data

Attributes:

Attribute	Length (byte)	Description
Status	1	Status is either Success (0) or Failure (1).

<table>
<tr>
<td>Data</td>
<td>AL</td>
<td>Bytes providing value of the SECURITY PIB attribute requested. Table below describes the data returned for the queried MAC Security PIB Attribute. The data length is fixed at 27 but for each attribute only first few data bytes contain relevant data. Table below lists the attributes, number of bytes (length) of useful data for that attribute and a brief description
<table>
<tr><th>MAC PIB Attribute</th><th>Length</th><th>Description</th></tr>
<tr><td>MAC_PAN_COORD_EXTENDED_ADDRESS</td><td>8</td><td>Eight bytes that specify the coordinator’s long address in little-endian order</td></tr>
<tr><td>MAC_PAN_COORD_SHORT_ADDRESS</td><td>2</td><td>Two bytes that specify the coordinator’s short address in little-endian order</td></tr>
<tr><td>MAC_SECURITY_LEVEL_ENTRY</td><td>1</td><td>One byte that specifies the security level being used, can be used to determine if device supports security</td></tr>
</table>
</td>
</tr>
</table>

5.3.7 MAC_SECURITY_SET_REQ

Description:
This command is used to request the MAC-coP to write a MAC SECURITY PIB value.

Usage:
SREQ:

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

AL = Attribute Length

Attributes:

<table>
<tr><th>Attribute</th><th>Length (byte)</th><th>Description</th></tr>
<tr>
<td>Attribute</td>
<td>1</td>
<td>Specified the MAC SECURITY PIB Attribute:
<table>
<tr><th>MAC PIB Attribute</th><th>Value</th><th>AL</th><th>Description</th></tr>
<tr><td>MAC_KEY_TABLE</td><td>0x61</td><td>0</td><td>Any bytes after the attribute identifier are ignored. This API triggers initialization of the key table. This must be done before adding any keys</td></tr>
<tr><td>MAC_DEVICE_TABLE</td><td>0x62</td><td>0</td><td>Any bytes after the attribute identifier are ignored. This API triggers initialization of the device table</td></tr>
<tr><td>MAC_SECURITY_LEVEL_TABLE</td><td>0x63</td><td>0</td><td>Any bytes after the attribute identifier are ignored. This API triggers initialization of the security level table</td></tr>
<tr><td>MAC_DEFAULT_KEY_SOURCE</td><td>0x79</td><td>8</td><td>Eight bytes specifying default key source, e.g.,</td></tr>
</table>
</td>
</tr>
</table>

					{0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33}
		MAC_PAN_COORD_EXTENDED_ADDRESS	0x7A	8	Two bytes that specify the coordinator's short address in little-endian order
		MAC_PAN_COORD_SHORT_ADDRESS	0x7B	2	Two bytes that specify the coordinator's short address in little-endian order
		MAC_SECURITY_LEVEL_ENTRY	0xD5	0	The MAC security level entry, an entry of the security level table
AttributeValue	AL	1-16 bytes of the SECURITY PIB attribute value.			

SRSP:

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

Attributes:

Attribute	Length (byte)	Description
Status	1	Status is either Success (0) or Failure (1).

5.3.8 MAC_WRITE_KEY_WITH_ID_REQ

Description:

This command is used to add a key to the key table. MAC_SECURITY_SET_REQ should be invoked with parameter MAC_KEY_TABLE in order to initialize the key table prior to making this call

Usage:

SREQ:

Byte: 1	1	1	16	4	1
Length = 33+	Cmd0 = 0x22	Cmd1 = 0x34	Key	Frame Counter	Key Table Index

Byte: 1	11+
New Key Flag	Lookup List

Attributes:

Attribute	Length (byte)	Description
Key	16	The 16-byte key to use
Frame Counter	4	current frame counter for this key (typically 0)
Key Table Index	1	typically 0
New Key Flag	1	typically 1

<table>
<tr><td>Lookup List</td><td>11+</td><td>
<table>
<tr><th>Component</th><th>Bytes</th><th>Description</th></tr>
<tr><td>Blob Count</td><td>1</td><td>number of blobs in list</td></tr>
<tr><td>Bytes in Blob</td><td>1</td><td>Bytes in a blob, usually 9</td></tr>
<tr><td>key source</td><td>8</td><td>eg, 0x33, 0x33,…</td></tr>
<tr><td>Key ID</td><td>1</td><td>Used to lookup key, eg, 3</td></tr>
<tr><td>next blob…</td><td>10</td><td>ignored by MAC</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.

Bit: 0 – Alternate PAN Coordinator
1 – Device type: 1- ZigBee Router; 0 – End Device
2 – Power Source: 1 Main powered
3 – Receiver on when Idle
4 – Reserved
5 – Reserved
6 – Security capability
7 – Reserved</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 = 0x06	Status

Attributes:

Attribute	Length (byte)	Description
Status	1	Status is either Success (0) or Failure (1).

5.3.9 MAC_ADD_DEVICE_REQ

Description:

This command is used to add a device to the device table and associate it with a key. MAC_SECURITY_SET_REQ should be invoked with parameter MAC_KEY_TABLE in order to initialize the key table and again with MAC_DEVICE_TABLE to initialize the device table prior to making this call

Usage:

SREQ:

Byte: 1	1	1	2	2	8
Length = 30	Cmd0 = 0x22	Cmd1 = 0x35	Pan ID	short device address	long device address

1	1	Bytse: 10	4	1	1
exempt flag	key lookup list size (9)	key lookup list data	frame counter	unique device flag	duplicate device flag

Attributes:

Attribute	Length (byte)	Description
Pan ID	2	PAN ID of network in little endian order
Short device address	2	short device address (little endian)
Long device address	8	long device address (little endian)
exempt flag	1	exempt flag, TRUE or FALSE
key lookup list size	1	9
Lookup List	9	Component / Bytes / Description key source / 8 / eg, 0x33, 0x33,… Key ID / 1 / Used to lookup key, eg, 3
Frame Counter	4	current frame counter for the key for this device
unique device flag	1	TRUE if unique key, FALSE if group key
duplicate device Flag	1	controls whether we create a new entry in table for this device

5.3.10 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 = 0x0B	**Cmd0 = 0x42**	**Cmd1 = 0x51**	**ExtAddr**	**AssocShortAddress**	**AssociatedMember**

Attributes:

Attribute	Length (byte)	Description
ExtAddr	**8**	**Extended Address of the device sending the orphan notification**

AssocShortAddress	**2**	**Short address of the orphan device**
AssociatedMember	**1**	**TRUE if the orphaning device is an associated member. FALSE otherwise.**

SRSP:

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

Attributes:

Attribute	Length (byte)	Description
Status	**1**	**Status is either Success (0) or Failure (1).**

5.3.11 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:

Attribute	Length (byte)	Description
CoordAddressMode	1	Mode / Value / Description ADDRESS_16_BIT / 0x02 / Address 16 bit ADDRESS_64_BIT / 0x03 / Address 64 bit
CoordAddress	8	**64-bit Coordinator Address**
CoordPanId	2	**Coordinator PanId**
KeySource	8	**Key Source of this data frame.**
SecurityLevel	1	**Security Level of this data frame:** Security Level / Value NO_SECURITY / 0x00 MIC_32_AUTH / 0x01 MIC_64_AUTH / 0x02 MIC_128_AUTH / 0x03 AES_ENCRYPTION / 0x04 AES_ENCRYPTION_MIC_32 / 0x05 AES_ENCRYPTION_MIC_64 / 0x06 AES_ENCRYPTION_MIC_128 / 0x07

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.**

SRSP:

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

Attributes:

Attribute	Length (byte)	Description
Status	**1**	**Status is either Success (0) or Failure (1).**

5.3.12 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 will set the MAC PIB to default values. (file mac_pib.c: see structure macPibDefaults for default MAC PIBvalues in the MAC coP project)

Usage:

SREQ:

Byte: 1	1	1	1
Length = 0x02	**Cmd0 = 0x22**	**Cmd1 = 0x01**	**SetDefault**

Attributes:

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

SRSP:

Byte: 1	1	1	1
Length = 0x01	**Cmd0 = 0x62**	**Cmd1 = 0x00**	**Status**

Attributes:

Attribute	Length (byte)	Description
Status	**1**	**Status is either Success (0) or Failure (1).**

5.3.13 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 ignore duplicate beacons.

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. But 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	4	1	1
Length = 0x13	**Cmd0 = 0x22**	**Cmd1 = 0x0C**	**ScanChannels**	**ScanType**	**ScanDuration**

Byte: 1	1	8	1	1	1
ChannelPage	**MaxResults**	**KeySource**	**SecurityLevel**	**KeyIdMode**	**KeyIndex**

Attributes:

Attribute	Length (byte)	Description
ScanChannels	4	**This represents a bit-mask of channels to be scanned when starting the device:**
ScanType	1	**Specifies the scan type:**
ScanDuration	1	**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.**

Channel	Value
NONE	0x00000000
ALL_CHANNELS	0x07FFF800
CHANNEL 11	0x00000800
CHANNEL 12	0x00001000
CHANNEL 13	0x00002000
CHANNEL 14	0x00004000
CHANNEL 15	0x00008000
CHANNEL 16	0x00010000
CHANNEL 17	0x00020000
CHANNEL 18	0x00040000
CHANNEL 19	0x00080000
CHANNEL 20	0x00100000
CHANNEL 21	0x00200000
CHANNEL 22	0x00400000
CHANNEL 23	0x00800000
CHANNEL 24	0x01000000
CHANNEL 25	0x02000000
CHANNEL 26	0x04000000

Scan Type	Value
ENERGY_DETECT	0x00
ACTIVE	0x01
PASSIVE	0x02
ORPHAN	0x03

ChannelPage	1	**The channel page on which to perform the scan.**
MaxResults	1	**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.**
KeySource	8	**Key Source of this data frame.**
SecurityLevel	1	**Security Level of this data frame:** Security Level: NO_SECURITY = 0x00; MIC_32_AUTH = 0x01; MIC_64_AUTH = 0x02; MIC_128_AUTH = 0x03; AES_ENCRYPTION = 0x04; AES_ENCRYPTION_MIC_32 = 0x05; AES_ENCRYPTION_MIC_64 = 0x06; AES_ENCRYPTION_MIC_128 = 0x07
KeyIdMode	1	**Key Id Mode of this data frame:** Key Id Mode: NOT_USED = 0x00; KEY_1BYTE_INDEX = 0x01; KEY_4BYTE_INDEX = 0x02; KEY_8BYTE_INDEX = 0x03
KeyIndex	**1**	**Key Index of this data frame.**

SRSP:

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

Attributes:

Attribute	Length (byte)	Description
Status	**1**	**Status is either Success (0) or Failure (1).**

5.3.14 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 parameter panCoordinator 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 parameter beaconOrder 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
Length = 0x23	Cmd0 = 0x22	Cmd1 = 0x03	StartTime	PanId	LogicalChannel	ChannelPage

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

Byte: 1	1	1	8	1
RealignSecurityLevel	RealignKeyIdMode	RealignKeyIndex	BeaconKeySource	BeaconSecurityLevel

Byte: 1	1
BeaconKeyIdMode	BeaconKeyIndex

Attributes:

Attribute	Length (byte)	Description
StartTime	4	**The 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 *aUnitBackoffPeriod* symbol periods.**
PanId	2	**The PAN Id to use. This parameter is ignored if Pan Coordinator is FALSE**
LogicalChannel	1	**The logical channel to use. This parameter is ignored if Pan Coordinator is FALSE**
ChannelPage	1	**The channel page to use. This parameter is ignored if Pan Coordinator is FALSE**
BeaconOrder	1	**The exponent used to calculate the beacon interval. The beacon interval is 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	**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.**
PanCoordinator	1	**Set to TRUE to start a network as PAN coordinator**
BatteryLifeExt	1	**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.**
CoordRealignment	1	**Set to TRUE to transmit a coordinator realignment prior to changing the superframe configuration.**
RealignKeySource	8	**Key Source of this data frame**
RealignSecurityLevel	1	**Security Level of this data frame:** Security Level / Value: NO_SECURITY 0x00 MIC_32_AUTH 0x01 MIC_64_AUTH 0x02 MIC_128_AUTH 0x03 AES_ENCRYPTION 0x04 AES_ENCRYPTION_MIC_32 0x05 AES_ENCRYPTION_MIC_64 0x06 AES_ENCRYPTION_MIC_128 0x07
RealignKeyIdMode	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

RealignKeyIndex	1	**Key Index of this data frame**
BeaconKeySource	8	**Key Source of this data frame**
BeaconSecurityLevel	1	**Security Level of this data frame:** Security Level / Value NO_SECURITY 0x00 MIC_32_AUTH 0x01 MIC_64_AUTH 0x02 MIC_128_AUTH 0x03 AES_ENCRYPTION 0x04 AES_ENCRYPTION_MIC_32 0x05 AES_ENCRYPTION_MIC_64 0x06 AES_ENCRYPTION_MIC_128 0x07
BeaconKeyIdMode	1	**Key Id Mode of this data frame**
BeaconKeyIndex	**1**	**Key Index of this data frame**

SRSP:

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

Attributes:

Attribute	Length (byte)	Description
Status	**1**	**Status is either Success (0) or Failure (1).**

5.3.15 MAC_SYNC_REQ

Description:

This API requests the MAC 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
Length = 0x03	**Cmd0 = 0x22**	**Cmd1 = 0x04**	**LogicalChannel**	**ChannelPage**	**TrackBeacon**

Attributes:

Attribute	Length (byte)	Description
LogicalChannel	1	**The logical channel to use.**

ChannelPage	1	**The channel page to use.**
TrackBeacon	**1**	**Set to TRUE to continue tracking beacons after synchronizing with the first beacon. Set to FALSE to only synchronize with the first beacon**

SRSP:

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

Attributes:

Attribute	Length (byte)	Description
Status	**1**	**Status is either Success (0) or Failure (1).**

5.3.16 MAC_SET_RX_GAIN_REQ

Description:
This command is used to send a request to the device to set Rx gain when a PA/LNA (CC2590/1) is used along with MAC coP. Also, for CC2591 – compile option HAL_PA_LNA must be used to build the MAC coP image. For CC2590 conpile option HAL_PA_LNA_CC2590 must be used to build MAC coP image.

Usage:

SREQ:

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

Attributes:

Attribute	Length (byte)	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 (byte)	Description
Status	**1**	**Status is either Success (0) or Failure (1).**

MT_MAC Callbacks

5.3.17 MAC_SYNC_LOSS_IND

Description:

This event is sent to the application when the MAC 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
Length = 0x10	Cmd0 = 0x42	Cmd1 = 0x80	Status	PanId	LogicalChannel	ChannelPage

8	1	1	1
KeySource	SecurityLevel	KeyIdMode	KeyIndex

Attributes:

Attribute	Length (byte)	Description
Status	1	**NAME** / **DESCRIPTION** **MAC_BEACON_LOSS** — **The beacon was lost following a synchronization request.** **MAC_PAN_ID_CONFLICT** — **A PAN identifier conflict has been detected.** **MAC_REALIGNMENT** — **A coordinator realignment command has been received.**
PanId	2	**PAN Id of the device**
LogicalChannel	1	**Logical Channel of the device where the synchronization is lost**
ChannelPage	1	**Channel Page of the device where the synchronization is lost**
KeySource	8	**Key Source of this data frame.**
SecurityLevel	1	**Security Level of this data frame:** Security Level / Value NO_SECURITY 0x00 MIC_32_AUTH 0x01 MIC_64_AUTH 0x02 MIC_128_AUTH 0x03 AES_ENCRYPTION 0x04 AES_ENCRYPTION_MIC_32 0x05 AES_ENCRYPTION_MIC_64 0x06 AES_ENCRYPTION_MIC_128 0x07
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.

5.3.18 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	DeviceExtendedAddress	Capabilities

8	1	1	1
KeySource	SecurityLevel	KeyIdMode	KeyIndex

Attributes:

Attribute	Length (byte)	Description
DeviceExtendedAddress	8	**Extended address of the device**
Capabilities	1	**Specifies the operating capabilities of the device being directly joined. Bit weighted values follow:** **NAME** / **DESCRIPTION** **MAC_CAPABLE_PAN_COORD** — **Device is capable of becoming a PAN coordinator.** **MAC_CAPABLE_FFD** — **Device is an FFD.** **MAC_CAPABLE_MAINS_POWER** — **Device is mains powered rather than battery powered.** **MAC_CAPABLE_RX_ON_IDLE** — **Device has its receiver on when idle.** **MAC_CAPABLE_SECURITY** — **Device is capable of sending and receiving secured frames.** **MAC_CAPABLE_ALLOC_ADDR** — **Request allocation of a short address in the associate procedure.**
KeySource	8	**Key Source of this data frame.**
SecurityLevel	1	**Security Level of this data frame:** Security Level / Value NO_SECURITY 0x00 MIC_32_AUTH 0x01 MIC_64_AUTH 0x02 MIC_128_AUTH 0x03 AES_ENCRYPTION 0x04 AES_ENCRYPTION_MIC_32 0x05 AES_ENCRYPTION_MIC_64 0x06 AES_ENCRYPTION_MIC_128 0x07

KeyIdMode	1	**Key Id Mode of this data frame:** Key Id Mode: NOT_USED 0x00; KEY_1BYTE_INDEX 0x01; KEY_4BYTE_INDEX 0x02; KEY_8BYTE_INDEX 0x03
KeyIndex	**1**	**Key Index of this data frame.**

5.3.19 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
Length = 0x0E	**Cmd0 = 0x42**	**Cmd1 = 0x82**	**Status**	**DeviceShortAddress**

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

Attributes:

Attribute	Length (byte)	Description
Status	1	NAME / DESCRIPTION: **MAC_CHANNEL_ACCESS_FAILURE** — **Data transmission failed because of congestion on the channel.** **MAC_INVALID_PARAMETER** — **The API API parameter is out of range.** **MAC_NO_ACK** — **No acknowledgement was received from the peer device.** **MAC_NO_DATA** — **No associate response was received from the peer device.**
DeviceShortAddress	2	**Short address of the device**
KeySource	8	**Key Source of this data frame.**
SecurityLevel	1	**Security Level of this data frame:** Security Level: NO_SECURITY 0x00; MIC_32_AUTH 0x01; MIC_64_AUTH 0x02; MIC_128_AUTH 0x03; AES_ENCRYPTION 0x04; AES_ENCRYPTION_MIC_32 0x05; AES_ENCRYPTION_MIC_64 0x06; AES_ENCRYPTION_MIC_128 0x07

KeyIdMode	1	**Key Id Mode of this data frame:** Key Id Mode: NOT_USED = 0x00; KEY_1BYTE_INDEX = 0x01; KEY_4BYTE_INDEX = 0x02; KEY_8BYTE_INDEX = 0x03
KeyIndex	**1**	**Key Index of this data frame.**

5.3.20 MAC_BEACON_NOTIFY_IND

Description:

This event is sent to the application when the MAC coP receives a beacon frame and the beacon contains payload data or attribute MAC_AUTO_REQUEST is set to FALSE. The event also contains an LQI measurement and the time the beacon was received in addition to the beacon information.

Usage:

AREQ:

1	1	1	1	4	1
Length = 0x24-0xBC	**Cmd0 = 0x42**	**Cmd1 = 0x83**	**BSN**	**Timestamp**	**CoordinatorAddressMode**

8	2	2	1	1	1
CoordinatorExtendedAddress	**PanId**	**SuperframeSpec**	**LogicalChannel**	**GTSPermit**	**LinkQuality**

1	8	1	1	1	1	1	1	0-128
SecurityFailure	**KeySource**	**SecurityLevel**	**KeyIdMode**	**KeyIndex**	**PendingAddrSpec**	**AddressList**	**SDULength**	**NSDU**

Attributes:

Attribute	Length (byte)	Description
BSN	1	**Beacon sequence number**
Timestamp	4	**The time at which the beacon was received, in *aUnitBackoffPeriod* units**
CoordinatorAddressMode	1	**Address mode of the coordinator** Mode: ADDRESS_16_BIT = 0x02 (Address 16 bit); ADDRESS_64_BIT = 0x03 (Address 64 bit)
CoordinatorExtendedAddress	8	**Extended address of the coordinator**
PanId	2	**Pan Id of the device**
SuperframeSpec	2	**Superframe specification of the network**
LogicalChannel	1	**logical channel of the network**
GTSPermit	1	**TRUE/FALSE - Permit/ does Not permit GTS**
LinkQuality	1	**Link quality of the message**
SecurityFailure	1	**Set to true if there was an error in security processing**
KeySource	8	**Key Source of this data frame.**

SecurityLevel	1	**Security Level of this data frame:** Security Level: NO_SECURITY = 0x00; MIC_32_AUTH = 0x01; MIC_64_AUTH = 0x02; MIC_128_AUTH = 0x03; AES_ENCRYPTION = 0x04; AES_ENCRYPTION_MIC_32 = 0x05; AES_ENCRYPTION_MIC_64 = 0x06; AES_ENCRYPTION_MIC_128 = 0x07
KeyIdMode	1	**Key Id Mode of this data frame:** Key Id Mode: NOT_USED = 0x00; KEY_1BYTE_INDEX = 0x01; KEY_4BYTE_INDEX = 0x02; KEY_8BYTE_INDEX = 0x03
KeyIndex	1	**Key Index of this data frame.**
PendingAddrSpec	1	
AddressList	1	**List of address associate with the device**
SDULength	1	**Beacon Length**
NSDU	**0-128**	**Beacon payload**

5.3.21 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	8
Length = 0x14	**Cmd0 = 0x42**	**Cmd1 = 0x86**	**ExtendedAddress**	**DisassociateReason**	**KeySource**

1	1	1
SecurityLevel	**KeyIdMode**	**KeyIndex**

Attributes:

Attribute	Length (byte)	Description
ExtendedAddress	8	**Extended address of the device leaving the network**
DisassociateReason	1	**Reason of the disassociation:** Reason: Coordinator wishes the device to disassociate = 0x01; Device itself wishes to disassociate = 0x02
KeySource	8	**Key Source of this data frame.**

<table>
<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>

5.3.22 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 (byte)</th><th>Description</th></tr>
<tr><td>Status</td><td>1</td><td><table><tr><th>NAME</th><th>DESCRIPTION</th></tr><tr><td>MAC_SUCCESS</td><td>Operation successful.</td></tr><tr><td>MAC_CHANNEL_ACCESS_FAILURE</td><td>Data transmission failed because of congestion on the channel.</td></tr><tr><td>MAC_INVALID_PARAMETER</td><td>The API API parameter is out of range.</td></tr><tr><td>MAC_NO_ACK</td><td>No acknowledgement was received from the peer device.</td></tr><tr><td>MAC_TRANSACTION_EXPIRED</td><td>Transmission failed because the peer device did not respond before the transaction expired.</td></tr><tr><td>MAC_TRANSACTION_OVERFLOW</td><td>MAC data buffers are full.</td></tr></table></td></tr>
<tr><td>DeviceAddrMode</td><td>1</td><td>Address mode of the device<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>DeviceAddr</td><td>8</td><td>Address of the device</td></tr>
</table>

DevicePanId	2	Pan Id of the device

5.3.23 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
Length = 0x13	Cmd0 = 0x42	Cmd1 = 0x8A	ExtendedAddr

8	1	1	1
KeySource	SecurityLevel	KeyIdMode	KeyIndex

Attributes:

Attribute	Length (byte)	Description
ExtendedAddr	8	**Extended address of the orphan device**
KeySource	8	**Key Source of this data frame.**
SecurityLevel	1	**Security Level of this data frame:** Security Level: NO_SECURITY = 0x00; MIC_32_AUTH = 0x01; MIC_64_AUTH = 0x02; MIC_128_AUTH = 0x03; AES_ENCRYPTION = 0x04; AES_ENCRYPTION_MIC_32 = 0x05; AES_ENCRYPTION_MIC_64 = 0x06; AES_ENCRYPTION_MIC_128 = 0x07
KeyIdMode	1	**Key Id Mode of this data frame:** Key Id Mode: NOT_USED = 0x00; KEY_1BYTE_INDEX = 0x01; KEY_4BYTE_INDEX = 0x02; KEY_8BYTE_INDEX = 0x03
KeyIndex	**1**	**Key Index of this data frame.**

5.3.24 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_SUCESS. 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
Length = 0x01	Cmd0 = 0x42	Cmd1 = 0x8B	Status

Attributes:

Attribute	Length (byte)	Description

NAME	DESCRIPTION
MAC_SUCCESS	Operation successful.
MAC_CHANNEL_ACCESS_FAILURE	Failed because of congestion on the channel.
MAC_INVALID_PARAMETER	The API parameter is out of range.
MAC_NO_ACK	No acknowledgement was received from the peer device.
MAC_NO_DATA	No data was received from the peer device.

5.3.25 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 = 0x0A-0x8A	Cmd0 = 0x42	Cmd1 = 0x8C	Status	ED	ScanType	ChannelPage

4	1	1	0-128
UnscannedChannelList	ResultListCount	ResultListMaxLength	ResultList

Attributes:

Attribute	Length (byte)	Description
Status	1	NAME: MAC_SUCCESS — DESCRIPTION: Operation successful.

		MAC_INVALID_PARAMETER	**The API parameter is out of range.**
		MAC_NO_BEACON	**The active or passive scan failed because no beacons were received or the orphan scan failed because no coordinator realignment was received.**
ED	1	**ED max energy.**	
ScanType	1	**Specifies the scan type:** Scan Type: ENERGY_DETECT 0x00; ACTIVE 0x01; PASSIVE 0x02; ORPHAN 0x03	
ChannelPage	1	**Channel Page of scan**	
UnscannedChannelList	4	**Bit mask of un-scanned channels**	
ResultListCount	1	**Number of item in the result list. This value is not used if scanType is MAC_SCAN_ORPHAN.**	
ResultListMaxLength	1	**Max length of the result list in bytes**	
ResultList	**0-128**	**Result list**	

5.3.26 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 MAC coP has received a secure frame that generated an error during security processing

Usage:

AREQ:

1	1	1	1	8	1	8
Length = 0x24	**Cmd0 = 0x42**	**Cmd1 = 0x8D**	**Status**	**SrcAddr**	**DstAddrMode**	**DstAddr**

4	2	1	8	1	1	1
Timestamp	**DevicePanId**	**Reason**	**KeySource**	**SecurityLevel**	**KeyIdMode**	**KeyIndex**

Attributes:

Attribute	Length (byte)	Description
Status	1	**NAME** / **DESCRIPTION** **MAC_CHANNEL_ACCESS_FAILURE** — **The response frame failed because of activity on the channel.** **MAC_FRAME_TOO_LONG** — **The response frame or received frame is too long to be processed by the MAC.** **MAC_INVALID_PARAMETER** — **The API parameter is out of range.**

		MAC_NO_ACK: The response frame failed because no acknowledgement was received. MAC_TRANSACTION_EXPIRED: The response frame failed because the peer device did not respond before the transaction expired or was purged. MAC_TRANSACTION_OVERFLOW: The response frame failed because MAC data buffers are full.
DstAddrMode	1	Destination address mode Mode: ADDRESS_16_BIT, Value: 0x02, Description: Address 16 bit Mode: ADDRESS_64_BIT, Value: 0x03, Description: Address 64 bit
SrcAddr	8	Source address
DstAddr	8	Destination address
Timestamp	4	Timestamp of the message
DevicePanId	2	Pan Id of the device that generate the indication
Reason	1	The reason the event was generated. This parameter is not defined in [Error! Reference source not found.] but may be used to distinguish between the different uses of the event. Values are as follows: NAME: MAC_COMM_ASSOCIATE_RSP, DESCRIPTION: Event sent in response to MAC_AssociateRsp(). MAC_COMM_ORPHAN_RSP: Event sent in response to MAC_OrphanRsp(). MAC_COMM_RX_SECURE: Event sent as a result of receiving a secure frame.
dKeySource	8	Key Source of this data frame.
SecurityLevel	1	Security Level of this data frame: Security Level: NO_SECURITY, Value: 0x00 MIC_32_AUTH: 0x01 MIC_64_AUTH: 0x02 MIC_128_AUTH: 0x03 AES_ENCRYPTION: 0x04 AES_ENCRYPTION_MIC_32: 0x05 AES_ENCRYPTION_MIC_64: 0x06 AES_ENCRYPTION_MIC_128: 0x07
KeyIdMode	1	Key Id Mode of this data frame: Key Id Mode: NOT_USED, Value: 0x00 KEY_1BYTE_INDEX: 0x01 KEY_4BYTE_INDEX: 0x02 KEY_8BYTE_INDEX: 0x03
KeyIndex	1	Key Index of this data frame.

5.3.27 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 (byte)	Description
Status	**1**	**This field indicates either SUCCESS (0) or FAILURE (1).**

5.3.28 MAC_RX_ENABLE_CNF

Description:

This callback is called by the MAC to send (on behalf of the next higher layer) a MAC Rx enable confirmation.

Usage:

AREQ:

1	1	1	1
Length = 0x01	**Cmd0 = 0x42**	**Cmd1 = 0x8F**	**Status**

Attributes:

Attribute	Length (byte)	Description
Status	**1**	**This field indicates either SUCCESS (0) or FAILURE (1).**

5.4 *MT UTIL Interface*

5.4.1 MT_UTIL_GET_PRIMARY_IEEE

Description

This API is used to get the factory programmed IEEE address of the CC253x device stored in the information page. Compile option FEATURE_GET_PRIMARY_IEEE needs to be enabled on the MAC coP to enable this API.

Useage

SREQ

1	1	1	1
Length = 0x01	**Cmd0 = 0x27**	**Cmd1 = 0xEF**	**0x00**

SRSP

1	1	1	1	8
Length = 0x09	**Cmd0 = 0x67**	**Cmd1 = 0xEF**	**Status**	**IEEE Address**

Attributes

Attribute	Length (byte)	Description
Status	1	**Success(0) or failure (1)**
IEEE Address	**8**	**The value returned is LSB first.**

6. Status Values

6.1 *Standard Status Values*

NAME	DESCRIPTION	Value
MAC_SUCCESS	Operation successful.	**0x00**
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**
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_COUNTER_ERROR	The frame counter purportedly applied by the originator of the received frame is invalid.	**0xDB**
MAC_DENIED	The MAC was not able to enter low power mode.	**0xE2**
MAC_DISABLE_TRX_FAILURE	Unused.	**0xE3**
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_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_INVALID_ADDRESS	The data request failed because neither the source address nor destination address parameters were present.	**0xF5**
MAC_INVALID_GTS	Unused.	**0xE6**
MAC_INVALID_HANDLE	The purge request contained an invalid handle.	**0xE7**
MAC_INVALID_INDEX	Unused.	**0xF9**
MAC_INVALID_PARAMETER	The API function parameter is out of range.	**0xE8**
MAC_LIMIT_REACHED	The scan terminated because the PAN descriptor storage limit was reached.	**0xFA**
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	**0xEB**

	the poll request did not return any data.	
MAC_NO_SHORT_ADDRESS	The short address parameter of the start request was invalid.	**0xEC**
MAC_ON_TIME_TOO_LONG	Unused.	**0xF6**
MAC_OUT_OF_CAP	Unused.	**0xED**
MAC_PAN_ID_CONFLICT	A PAN identifier conflict has been detected and communicated to the PAN coordinator.	**0xEE**
MAC_PAST_TIME	Unused.	**0xF7**
MAC_READ_ONLY	A set request was issued with a read-only identifier.	**0xFB**
MAC_REALIGNMENT	A coordinator realignment command has been received.	**0xEF**
MAC_SCAN_IN_PROGRESS	The scan request failed because a scan is already in progress.	**0xFC**
MAC_SECURITY_ERROR	Cryptographic processing of the received secure frame failed.	**0xE4**
MAC_SUPERFRAME_OVERLAP	The beacon start time overlapped the coordinator transmission time.	**0xFD**
MAC_TRACKING_OFF	The start request failed because the device is not tracking the beacon of its coordinator.	**0xF8**
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_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**

6.2 *Proprietary Status Values*

NAME	DESCRIPTION	Value
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 no**	**0x1A**

	memory resources were available.	

7. General Information

7.1 *Document History*

Table 1: Document History

Revision	Date	Description/Changes
1.0	2014-24-09	Initial version

8. References

[1]. Z-Stack Monitor and Test API Document

[R1] CC253x User Guide. http://www.ti.com/litv/pdf/swru191

[R2] CC2530 Datasheet. http://www.ti.com/lit/gpn/cc2530

[R3] CC2531 Datasheet. http://www.ti.com/lit/gpn/cc2531

[R4] CC259x Datasheet. http://www.ti.com/lit/gpn/cc2591

[5] NPI Users's Guide