

Introduction

The purpose of this module is to introduce you to the Medium Access Controller (MAC). Some communications needs can be met with the MAC capabilities only, while others can only be met with the ZigBee stack. Either way the MAC forms the basis for both.

Objectives

- Look at the MAC architecture
- See the different network topologies
- Look at the controlling APIs
- Do a lab using beacon and non-beacon networks

*** where is my flying car? ***

Module Topics

TI-MAC.....	7-1
<i>Module Topics.....</i>	7-3
<i>Why Use the MAC?.....</i>	7-5
<i>Topology.....</i>	7-6
<i>Beacon vs. Non-Beacon.....</i>	7-7
<i>Beacon Configuration.....</i>	7-8
<i>Non-Beacon Configuration.....</i>	7-9
<i>Polling and Header Structures.....</i>	7-10
<i>PIB and Additional TI-MAC Features.....</i>	7-11
<i>Obtaining the TI-MAC and Lab Hardware.....</i>	7-12
<i>Lab7 – Using the MAC.....</i>	7-13
Hardware list:.....	7-14
Software list:.....	7-14
<i>Procedure.....</i>	7-15
Initial setup.....	7-15
Non-beacon network.....	7-17
Beacon-enabled network.....	7-18
<i>Further Information.....</i>	7-20

*** if you could do it all over again, would you? ***

Why Use the MAC?

Why Use the TI-MAC?

- ◆ The Medium Access Control stack offers full 802.15.4 compatibility
- ◆ Easy, low overhead peer to peer and star networking
- ◆ Beacon and non-beacon support
- ◆ Easy migration to ZigBee

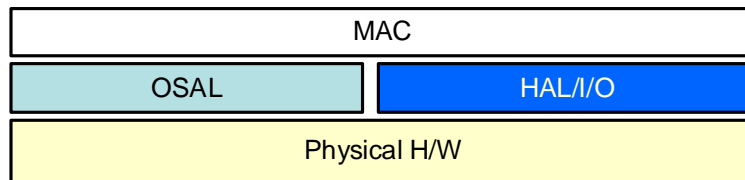


Highlights ...

2

802.15.4 MAC Highlights

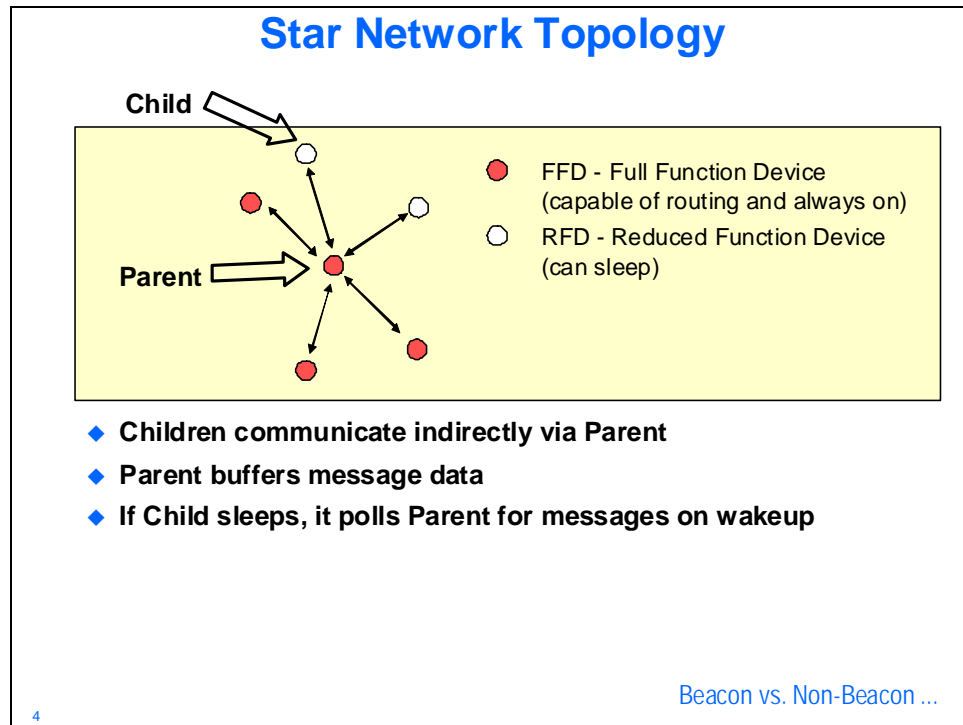
- ◆ Physical layer data rate of 250kbps
- ◆ Application data rate of approximately 120kbps
- ◆ 64 bit and 16 bit dynamic device addressing
- ◆ CSMA-CA channel access (“cocktail party” rule)
- ◆ Full hand-shake protocol for transfer reliability
- ◆ Link Quality Index (LQI) / Received Signal Strength Indicator (RSSI) support
- ◆ Energy Detection support



Topology ...



3

Topology





Beacon vs. Non-Beacon

Beacon vs. Non-Beacon Configurations

<p style="text-align: center;">Beacon</p>  <ul style="list-style-type: none"> • Parent provides a periodic beacon • Networked devices synchronize with beacon and only communicate during the active communication period 	<p style="text-align: center;">Non-Beacon</p>  <ul style="list-style-type: none"> • Networked device requests Parent beacon • Networked devices are free to communicate (using CSMA/CA rules) at any time
---	--

5

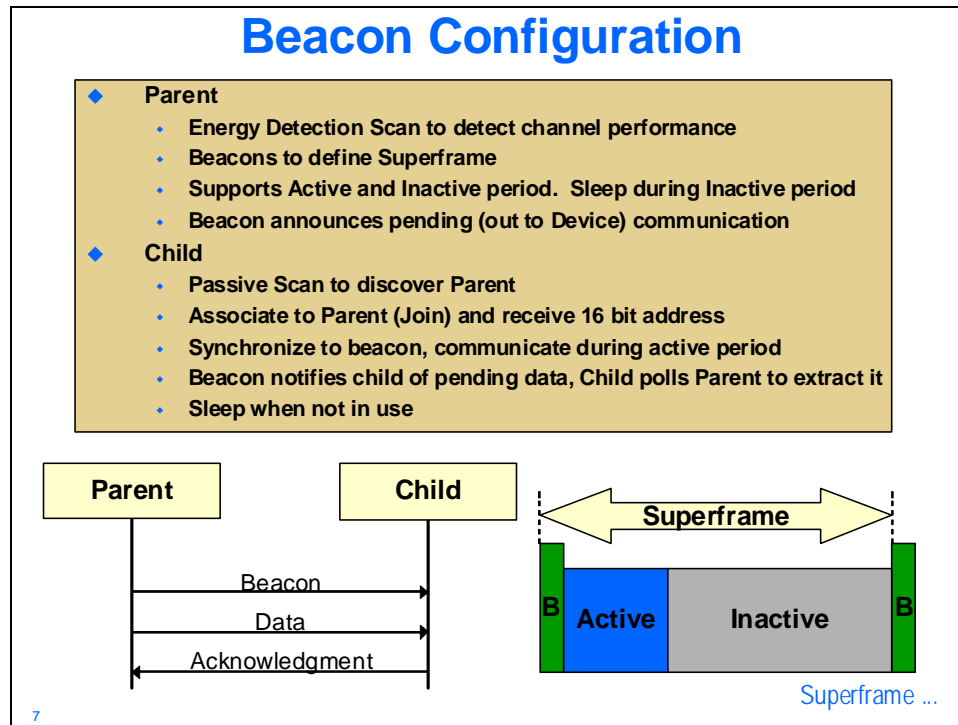
Beacon vs. Non-Beacon Configurations

<p style="text-align: center;">◆ Beacon Configuration</p>  <ul style="list-style-type: none"> • Parent may sleep during the inactive portion of the frame • Children don't have to poll for data, beacon notifies them • Reduced polling optimizes channel bandwidth • Children get a quick response time when data is available 	<p style="text-align: center;">◆ Non-Beacon Configuration</p>  <ul style="list-style-type: none"> • no overhead of periodic beacons • parent on at all times to receive communication • data can be polled at any time • allows for Peer to Peer communication to eventually build full "mesh"
--	--

Beacon config ...

6

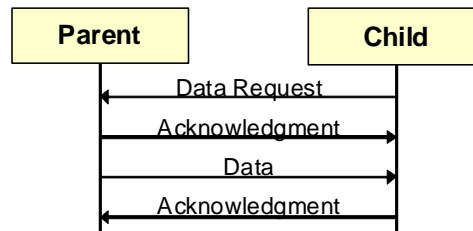
Beacon Configuration



Non-Beacon Configuration

Non-Beacon Configuration

- ◆ **Parent**
 - always on
 - Energy Detection Scan to detect channel performance
 - Respond to Beacon Request for network discovery
 - Buffer outgoing data for sleeping Child
- ◆ **Child**
 - Active Scan to discover available Parent
 - Associate to Parent (Join) and receive 16 bit Address
 - Optional sleep when not communicating
 - Wake up and send to Parent using CSMA/CA
 - Poll Parent to extract incoming data on demand or periodic



Polling ...

9

Polling and Header Structures

Polling

Non-Beacon mode


- Child polls its Parent anytime to obtain pending data

Beacon Mode

- Child synchs to Parent's beacon
- Beacons announces pending data
- Child polls Parent to obtain data
- Parent can notify Children to stay awake so they can obtain additional data

Both Modes

- A bit field in the acknowledgement message indicates additional data is present and will require polling to retrieve it



Header Structures ...

10

Header Structures

PHY Header

Preamble 32-bits	Start of Packet Delimiter 8-bits	Header 8-bits	PHY Payload 0-1016 bits
---------------------	--	------------------	----------------------------

Payload length ↑

- ◆ 802.15.4 PHY supports 128 byte packets preceded by a preamble and SFD for synchronizing on the packet
- ◆ Chipcon parts provide automatic filtering and buffering

MAC Header

Frame Control	Sequence Number	Addressing fields	MAC Payload < type dependent >
2 bytes	1 byte	4 to 10 bytes	variable

↑ Available frame types:

- ◆ Beacon Frame
- ◆ Command Frame
- ◆ Data Frame
- ◆ Ack Frame

PIB ...

11

PIB and Additional TI-MAC Features

PAN Information dataBase (PIB)

```

/* MAC PIB type */
typedef struct
{
    uint8    ackWaitDuration;
    bool     associationPermit;
    bool     autoRequest;
    bool     battLifeExt;
    uint8    battLifeExtPeriods;

    uint8    *pBeaconPayload;
    uint8    beaconPayloadLength;
    uint8    beaconOrder;
    uint32   beaconTxTime;
    uint8    bsn;

    sAddr_t  coordExtendedAddress;
    uint16   coordShortAddress;
    uint8    dsn;
    bool     gtsPermit;
    uint8    maxCsmBackoffs;

    uint8    minBe;
    uint16   panId;
    bool     promiscuousMode;
    bool     rxOnWhenIdle;
    uint16   shortAddress;

    uint8    superframeOrder;
    uint16   transactionPersistenceTime;
    bool     associatedPanCoord;
    uint8    maxBe;
    uint16   maxFrameTotalWaitTime;

    uint8    maxFrameRetries;
    uint8    responseWaitTime;
    uint8    syncSymbolOffset;
    bool     timeStampSupported;
    bool     securityEnabled;

    /* Proprietary */
    uint8    phyTransmitPower;
    uint8    logicalChannel;
    sAddr_t  extendedAddress;
    uint8    altBe;
}
    
```


- ◆ **mac_pib.h** contains many configuration parameters for the MAC

Additional MAC Features ...

12

Additional TI-MAC Features

- ◆ **Acknowledgement**
 - ◆ **Setting the Ack Request bit requests acknowledgement from the receiver**
- ◆ **Retries**
 - ◆ **Messages with the Ack Request bit set will be retried if an acknowledgement is not received within a period defined by the MACAckWaitDuration field.**



Obtaining the TI-MAC ...

13

Obtaining the TI-MAC and Lab Hardware

Obtaining the TI-MAC

- ◆ TI-MAC object code for the following platforms is available free on the TI web site for the following platforms (as of 9/2008):
 - ◆ CC2430
 - ◆ CC2420 + MSP430F461x (Experimenter's Board platform)
 - ◆ CC2520 + MSP430F2618 (CCMSP-EM430F2618 platform)
- ◆ Supports IEEE 802.15.4 – 2003
- ◆ Certified Compliant
- ◆ Free of charge. No royalties.
- ◆ From www.ti.com, use the keyword TIMAC



Target Hardware ...

14

Target Hardware



CC2520EM

IEEE 802.15.4 2.4GHz transceiver
AES-128 security module
49dB adjacent channel rejection
103dB link budget
RX 18.5mA, TX(0dBm) 25.8mA



CCMSP-EM430F2618

16-bit, 16MHz RISC microcontroller
116KB Flash, 8KB RAM
3 channel internal DMA
12-bit ADC
Two 12-bit DACs
Two 16-bit timers
USCI (UART/IrDA/SPI/I2C)

Emulator Interface



SmartRF05EB

3x16 character LCD
UART
LEDs
Serial Flash
Potentiometer
Joystick
Buttons



Lab time ...

15


Lab7 – Using the MAC

Lab 7 – Using the TI-MAC


- ◆ Run a non-beacon and a beacon enabled network
- ◆ Use Packet Sniffer

Packet Sniffer
(SmartRF04 + CC2430EM)



MSP430FET Emulator



Networked Boards
(SmartRF05 + CC2520EM)

[Additional Info ...](#)

Group	Channel
1	MAC_CHAN_12 (0x0C)
2	MAC_CHAN_13 (0x0D)
3	MAC_CHAN_14 (0x0E)
4	MAC_CHAN_15 (0x0F)
5	MAC_CHAN_16 (0x10)
6	MAC_CHAN_17 (0x11)
7	MAC_CHAN_18 (0x12)

16

Hardware list:

- ✓ 3 SmartRF05EB boards
- ✓ 3 CC2520EM boards
- ✓ 3 CCMSP-EM430F2618 boards
- ✓ MSP-430 FET + ribbon cable
- ✓ 5 USB A/B cables
- ✓ 1 SmartRF04EB board
- ✓ 1 CC2430EM board
- ✓ 4 antennas
- ✓ MSP-FET430UIF Emulator and ribbon cable

Software list:

- ✓ IAR Embedded Workbench for MSP430 version 4.11D
- ✓ TI Packet Sniffer version 2.10.1

(You will find shortcuts for the above applications on the desktop)

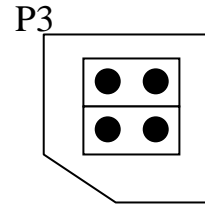
- ✓ TI-MAC version 1.2.1

Procedure

Initial setup

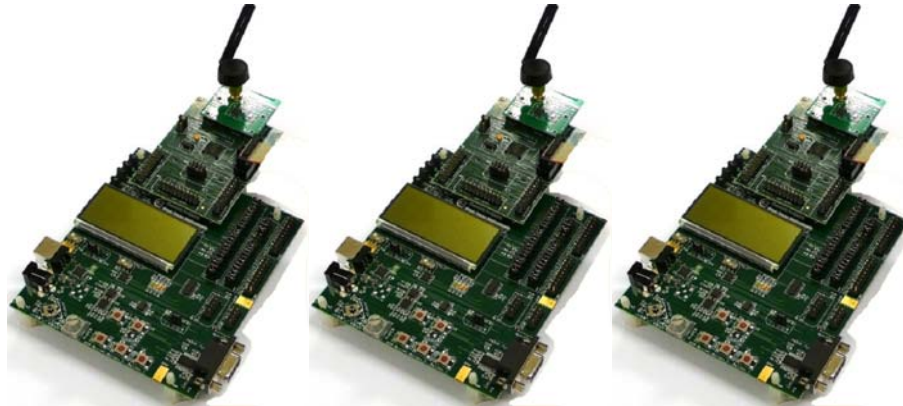
1. Sniffer

If you haven't done so already, remove the **CC2510EM** from the **SmartRF04EB** board. Replace it with the **CC2430EM** board. The **CC2430** will provide the IEEE 802.15.4 sniffing we'll need for the TI-MAC lab to follow. Change the **P3** jumpers on the **SmartRF04EB** board so that they are **parallel** to the LCD display.



2. Network Nodes

Connect a **CCMSP-EM430F2618** board to each of your three **SmartRF05EB** boards. Connect a **CC2520EM** to the top of each **CCMSP** board. Make sure each **CC2520EM** board has an antenna.



3. Check SmartRF05EB Drivers

Make sure that the **Power source** jumper on all three **SmartRF05EB** boards is connecting the **rightmost** two pins (**USB power**). Then, one at the time, **plug** all three boards into an open USB port using a USB A/B cable. Switch the **power switch on**, and following the procedure in step 3, make sure the drivers are properly installed. Do this for all three boards.

4. Connect the Emulator

Connect the ribbon cable from the **MSP-FET430UIF** to the emulator port of one of the **CCMSP** boards as shown below. Make sure that the **SmartRF05EB** and **Emulator** are connected to USB ports and that the **SmartRF05EB** power switch is **on**.



5. Open the MAC Sample Application

Start IAR Embedded Workbench, click **Open existing workspace** and open the MAC Sample Application project located at:

C:\Texas Instruments\TI-MAC-1.2.1\Projects\mac\Sample\MSP2618CC2520\IAR Project\msa_msp430.eww

6. Open msa.h


In the **Application** workspace group, open **msa.h** and change the following line to the **assigned channel** for your workgroup was assigned earlier. Note that channel numbers are in decimal. Everyone will have the same PAN ID, but since we're all on different channels, that's not a problem.

```
#define MSA_MAC_CHANNEL  MAC_CHAN_11
```



Several of default settings in this file enable a non-beaconed network. We'll be changing those in the next part of the lab.

7. Build and Download Code

Build and download the project to the target board. If you are prompted to update the FET firmware, click **Yes**.

When the download completes, click the **Go** button . The **D4** LED (red) on the **SmartRF05EB** board should be flashing rapidly. Click the **Stop Debugging** button.

8. Second Board

Disconnect the **MSP-FET430UIB** emulator from the first target and connect it to another. Make sure this board is connected to a USB port and powered. Click the **Debug** button  and download the project to the target. Click the **Go** button . The D4 LED should be blinking on this board too. Click the **Stop Debugging** button.

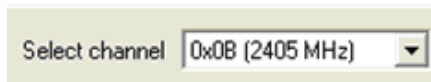
9. Third Board


Follow the same procedure with board number three. **Build/Load/Run** and make sure D4 is blinking. Click the **Stop Debugging** button. **Minimize** the IAR Embedded Workbench and disconnect the emulator ribbon cable from the last target.

10. Run the Packet Sniffer

Run the **Packet Sniffer** by double-clicking its' shortcut on the desktop. Select **IEEE 802.15.4/ZigBee (CC2430)** as your Protocol/Chip and click **Start..**

Set your assigned **channel**, in hex, as shown below.






Make sure your **SmartRF04EB/CC2430EM** sniffer hardware is connected and powered. Click the  button to start capturing packets. Of course, the Packet Sniffer display will remain blank since we haven't started sending packets yet

Non-beacon network


11. Beacon request

Choose one of the boards and press the joystick **up** (towards the LCD). Label this board **COORD**. Notice that the LED stops blinking and observe a **beacon request** command frame on the Packet Sniffer. This board has performed an active scan and started a network, becoming the coordinator.

TIP: If you screw up somewhere during this process and want to start back at the beginning, cycle power on all boards with the slide switches. You can clear the Packet Sniffer display by clicking the **Pause** button , then the delete button  and restarting the capture .

12. Association

Select a second board and press the joystick **up** (towards the LCD). Label this board **ED**. Notice that the LED now blinks slowly. On the Packet Sniffer observe a **beacon request** followed by a beacon frame, followed by an **association request**. The second board is now associated to the coordinator as an End Device.

TIP: Click the Enable/Disable Scrolling button  to easily see the last sent frame.

13. Data request

On the third board, press the joystick **up** (towards the LCD). Notice the LED now blinks slowly. This board has also associated to the coordinator as an End Device. Label it ED.

14. Data transmission

On the board labeled **COORD** press the joystick **right** (towards the RS-232 serial connector). This initiates data transmission from the coordinator to both End Devices. Observe the data frames sent to each device on the Packet Sniffer. By design, the MAC sample application on the device sends a response data frame for each data frame received by the coordinator. You can increase the traffic and initiate traffic on both End Device boards by pressing their joysticks **right**.

Observe the source and destination addresses. The Coordinator address is preprogrammed in **msa.h** as **0xAABB**. End devices that join the network should join at addresses **1,2** and so on.

Beacon-enabled network

15. Change hardware setup

Stop, clear and minimize the Packet Sniffer. **Maximize** the IAR Embedded Workbench.

16. Modify msa.h header file

Modify **msa.h** as follows to configure the beacon-enabled network:

```
#define MSA_WAIT_PERIOD          500           // was 200
#define MSA_MAC_BEACON_ORDER     6            // was 15 (disabled)
#define MSA_MAC_SUPERFRAME_ORDER 1          // was 15 (disabled)
```

17. Build/Load/Run

Build and **download** the project to each board. Click the **Go** button after each download, verify that the D4 LED is flashing, and then click **Stop Debugging**. Program and run all three boards.

18. Start the Sniffer

Minimize IAR Embedded Workbench, **maximize** the Packet Sniffer and **start** capturing frames.

19. Network start

Make sure that all three boards are powered and that the D4 LED is flashing on all of them. On the **COORD** board, press the joystick **up** (towards the LCD). Notice that the LED stops blinking. Observe **beacon frames** on the Packet Sniffer. The board has started a network, becoming the coordinator.

20. Association

On both of the **End Device** boards, press the joystick **up**. Notice the LED now blinks slowly. On the Packet Sniffer observe the association sequence as before.

21. Data transmission

One at the time, on all **three** boards, press the joystick **right**. This initiates data transmission from both the coordinator and the associated devices. Notice that the D4 LED flashing is synchronized on all the boards. Observe the data frames sent on the Packet Sniffer are synchronized with the beacon transmission.

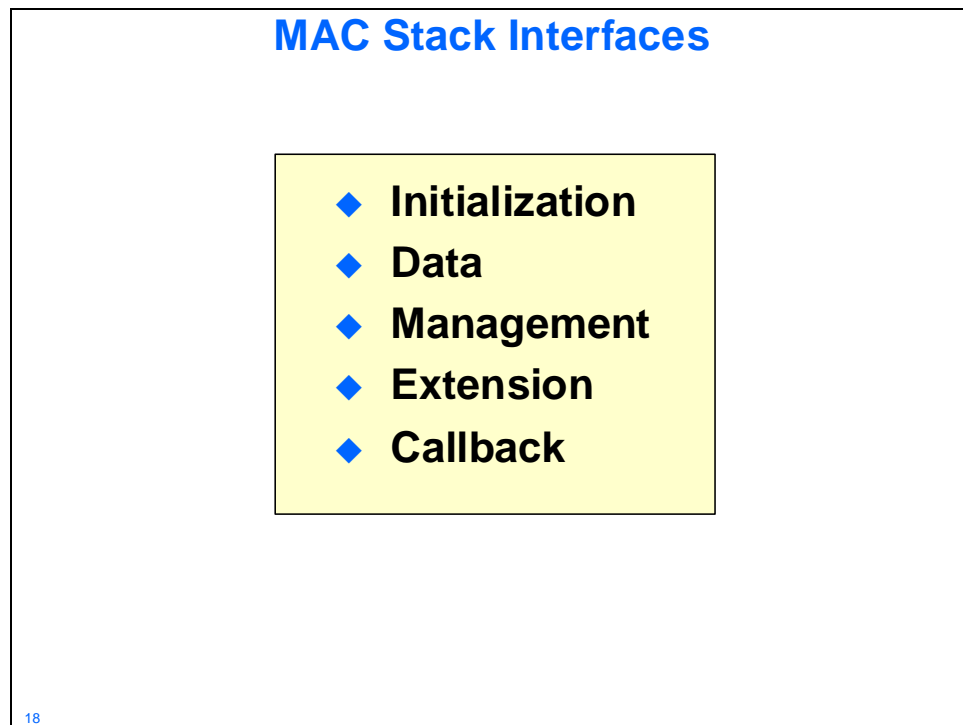
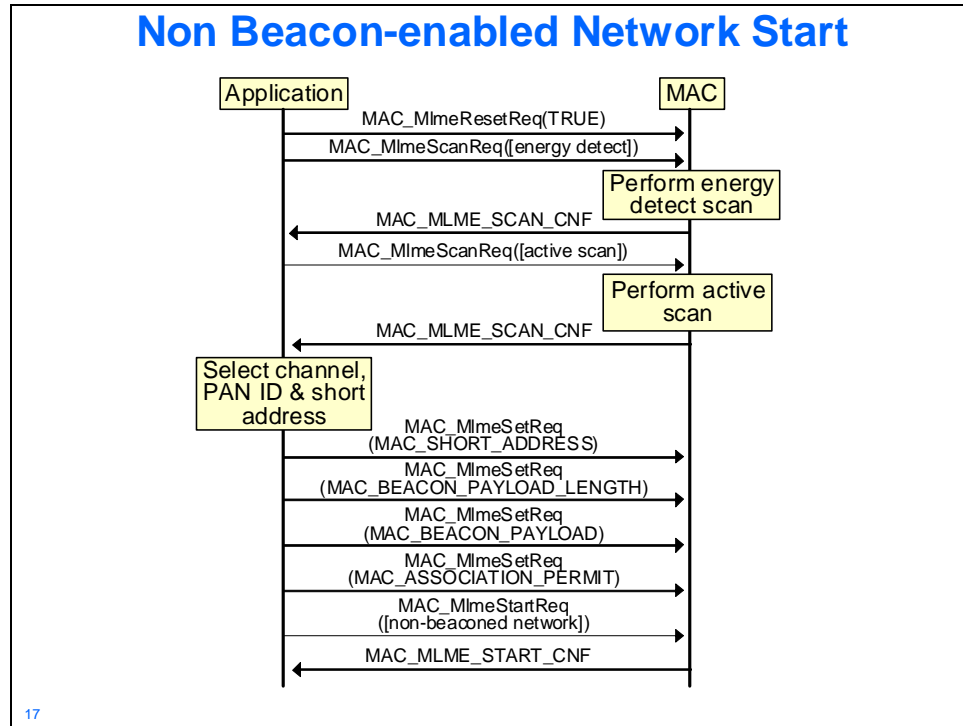
22. Power down

Close Packet Sniffer and IAR Embedded Workbench. Power all the boards down.



You're done

Further Information



Initialization Interface

- ◆ Used to configure optional features
- ◆ All functions are direct execute

MAC_Init	Main initialization function
MAC_InitDevice	assoc w/ non-beaconed network
MAC_InitCoord	init for operation as a coordinator
MAC_InitBeaconCoord	init coord function in beaconed network
MAC_InitBeaconDevice	assoc and track beaconed network

19

Data Interface

- ◆ MAC data request frame queue is configurable (1-255) Default is 2
- ◆ MAC will report overflow and ready states
- ◆ Application allocates data buffer(s)

MAC_McpsDataReq	Send data to MAC for transmission
MAC_McpsPurgeReq	Discards data request from queue
MAC_McpsDataAlloc	Allocates data buffer and preps pointer
MAC_MCPS_DATA_IND*	Sends data from MAC to application*
MAC_MCPS_DATA_CNF*	Event sent to app w/ status of data request
MAC_MCPS_PURGE_CNF*	Event sent to app w/ status of purge request

* **Data Callback**

** **After reading data, application should deallocate buffer**

20

Management Interface

- ◆ **Select channel**
- ◆ **Get/Set attributes**
- ◆ **Scan type**
- ◆ **Sent from MAC to Application**

<i>MAC_MlmeAssociateReq</i>	Sends an associate request to a coord
<i>MAC_MlmeAssociateRsp</i>	Response to requesting device
<i>MAC_MlmeDisassociateReq</i>	Request to coord to leave network
<i>MAC_MlmeGetReq</i>	Retrieves an attribute from MAC PIB
<i>MAC_MlmeOrphanRsp</i>	Response to orphan notification from peer
<i>MAC_MlmePollReq</i>	Requests pending data from coordinator
<i>MAC_MlmeResetReq</i>	Resets MAC
<i>MAC_MlmeScanReq</i>	Initiates a scan on one or more channels
<i>MAC_MlmeSetReq</i>	Sets an attribute in the MAC PIB
<i>MAC_MlmeStartReq</i>	Called by a coordinator to start a network
<i>MAC_MlmeSyncReq</i>	Sync with coordinator by acquiring beacons

PIB – PAN Information Base

21

Management Callbacks

- ◆ **Sent to application from MAC**

<i>MAC_MLME_ASSOCIATE_IND</i>	Associate request received
<i>MAC_MLME_ASSOCIATE_CNF</i>	Status of associate request
<i>MAC_MLME_DISASSOCIATE_IND</i>	Device has been disassociated
<i>MAC_MLME_DISASSOCIATE_CNF</i>	Status of disassociate request
<i>MAC_MLME_BEACON_NOTIFY_IND</i>	Beacon frame received
<i>MAC_MLME_ORPHAN_IND</i>	Orphan notification received
<i>MAC_MLME_SCAN_CNF</i>	Status of scan request
<i>MAC_MLME_START_CNF</i>	Status of start request
<i>MAC_MLME_SYNC_LOSS_IND</i>	MAC has lost sync with coord
<i>MAC_MLME_POLL_CNF</i>	Status of poll request
<i>MAC_MLME_COMM_STATUS_IND</i>	Various comm status indications
<i>MAC_MLME_POLL_IND</i>	Data request received

22

Extension Interface

- ◆ **Additional features beyond 802.15.4 spec**
 - ◆ **Power**
 - ◆ **Random number generation**

<i>MAC_PwrOffReq</i>	Power off radio and sleep
<i>MAC_PwrOnReq</i>	Power on radio and wake
<i>MAC_PwrMode</i>	Returns current MAC power mode
<i>MAC_PwrNextTimeout</i>	Returns next MAC timer expiration
<i>MAC_RandomByte</i>	Returns a random byte

23

Callback Interface

- ◆ **Applications must implement these functions**

<i>MAC_CbackEvent</i>	Sends events to app via OSAL messaging
<i>MAC_CbackCheckPending</i>	Returns number of queued messages

24

