Network Objects

Instructor: Rob Faludi

Plan for Today

- XBee API Review & I/O API
- XBee Sleep
- Final Project Presentation Info
- Final Project Progress (2 minute reports)
- ZigBee
- Wireless Sound Objects
- Readings & Assignments

XBee API Mode

API Mode

- Application Programming Interface
  - “An application programming interface (API) is a source code interface that an operating system or library provides to support requests for services to be made of it by computer programs.”

http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=43487

- XBees in API mode are ready to talk to computers and microcontrollers
  - structured
  - predictable
  - reliable

API Structure

- Used in serial communications with the XBee radio
- Frames of data
    - envelope structure contains data with metadata inside a constrained format
- Radio must be in API Mode
    - AT command ATAP 1 on Series 1 radios
    - API firmware on Series 2 radios

Why API

- Rather than:

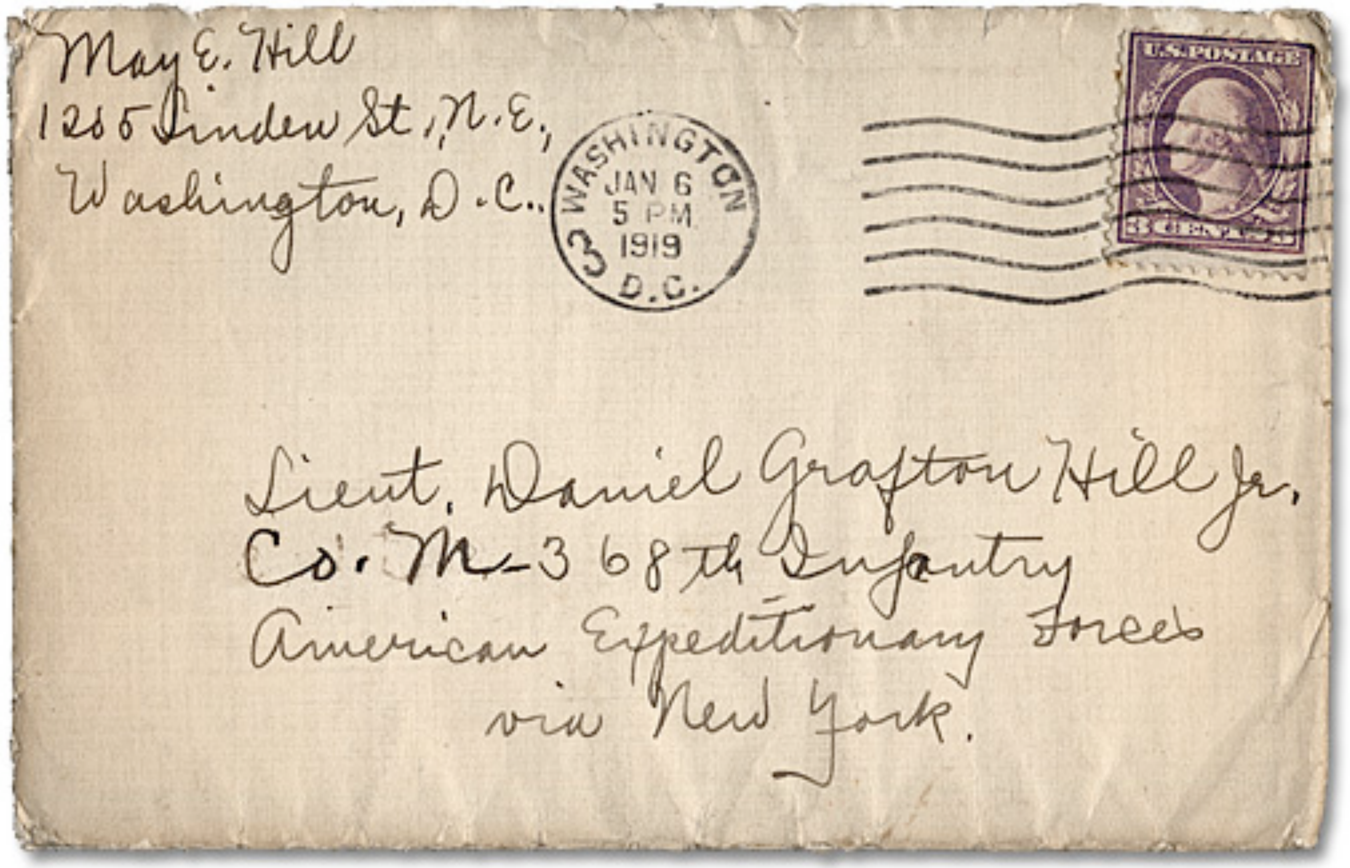
```
 delay(1100);
// put the XBee in command mode
 Serial.print("+++");
 delay(1100);
 if (checkFor("OK", 1000)) {
    Serial.println("ATID7777,CN");
     if (checkFor("OK", 1000)) {
         // if an OK was received then continue
         debugPrintln("SetupOK");
         success = true;
    }
 }
```

- With a library you just write:

```
sendCommand(ID,0x7777);
```

Envelope Has:

- From address, to address, outside, inside, size, contents, error check

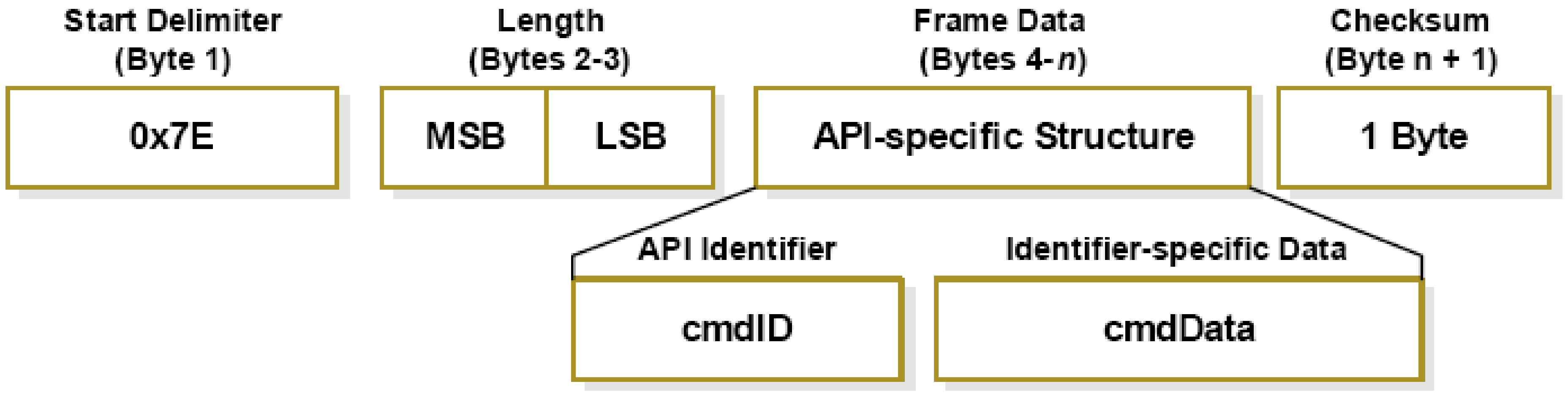

May E. Hill
1265 Linden St, N.E.,
Washington, D.C.

WASHINGTON JAN 6 5 PM 1919 D.C. 3

U.S. POSTAGE 3 CENTS 3

Lieut. Daniel Grafton Hill Jr.
Co. M - 368th Infantry
American Expeditionary Forces
via New York.

API Basic Frame Envelope



Start Byte

- 0x7E --> also known as the tilde in ASCII: ~
- First thing to do is look for it:

```
  // ARDUINO VERSION:
if (Serial.available() > 0) { // if a byte is waiting in the buffer
    inByte = Serial.read(); // read a byte from the buffer
    if (inByte == 0x7E) {
       // we're at the start of an API frame!
       // add more code here
     }
  }


  // PROCESSING VERSION:
if (port.available() > 0 {
  int inByte = port.read();
    if (inByte == 0x7E) {
       // we're at the start of an API frame!
       // add more code here
}
```

Length Bytes

- MSB: the Most Significant Byte
    - the big part of the number
- LSB: the Least Significant Byte
    - the small part of the number
- bit shift MSB to the right and add it to LSB

```
 // PROCESSING VERSION:
int lengthMSB = port.read(); // high byte for length of packet
int lengthLSB = port.read(); // low byte for length of packet

int lengthTotal = (lengthMSB << 8) + lengthLSB; // bit shift and add for total
```

API Identifier

- Specifies the remaining structure of the frame
  - modem status: 0x8A
  - AT command (immediate): 0x08
  - AT command (queued): 0x09
  - AT command response: 0x88
  - TX request (64 bit): 0x00
  - TX request (16 bit): 0x01
  - TX status response: 0x89
  - RX packet (64 bit): 0x80
  - RX packet (16 bit): 0x81
  - RX packet I/O data (64 bit): 0x82
  - RX packet I/O data (16 bit): 0x83

```
 // PROCESSING VERSION:
int API_ID = port.read(); // API Identifier indicates type of packet received
```

Identifier-specific Data

- Structures are different for each API identifier and might include:
  - addressing information (333B)
  - status information (received OK)
  - source information (broadcast packet)
  - unstructured data (“Hello World, this is Rob!”)
  - structured data (typically for I/O packets)

Checksum

- Simple check to detect errors
- To calculate: Not including frame delimiters and length, add all bytes keeping only the lowest 8 bits of the result and subtract from 0xFF.
- To verify: Add all bytes (include checksum, but not the delimiter and length). If the checksum is correct, the sum will equal 0xFF.

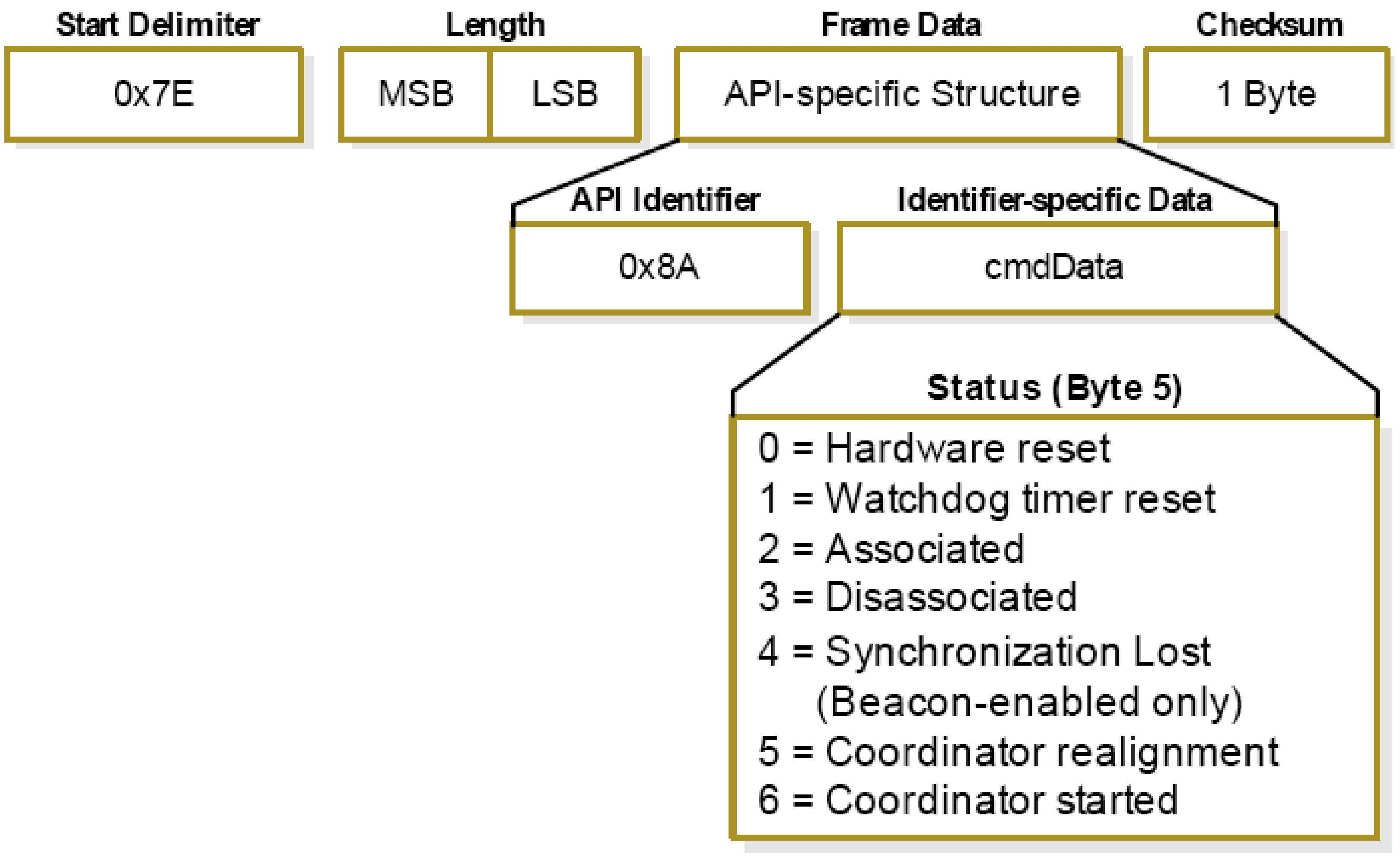
```
 // PROCESSING VERSION:
int localChecksum = (API_ID + addrMSB + addrLSB + RSSI + options + dataSum);

int checksum = port.read();
localChecksum = byte(0xFF -localChecksum);

if ( (byte) checksum - localChecksum == 0) {
  returnVal = dataADC[0];
}
else {
  print("\n\nchecksum error!  " + "\n\n");
}
```

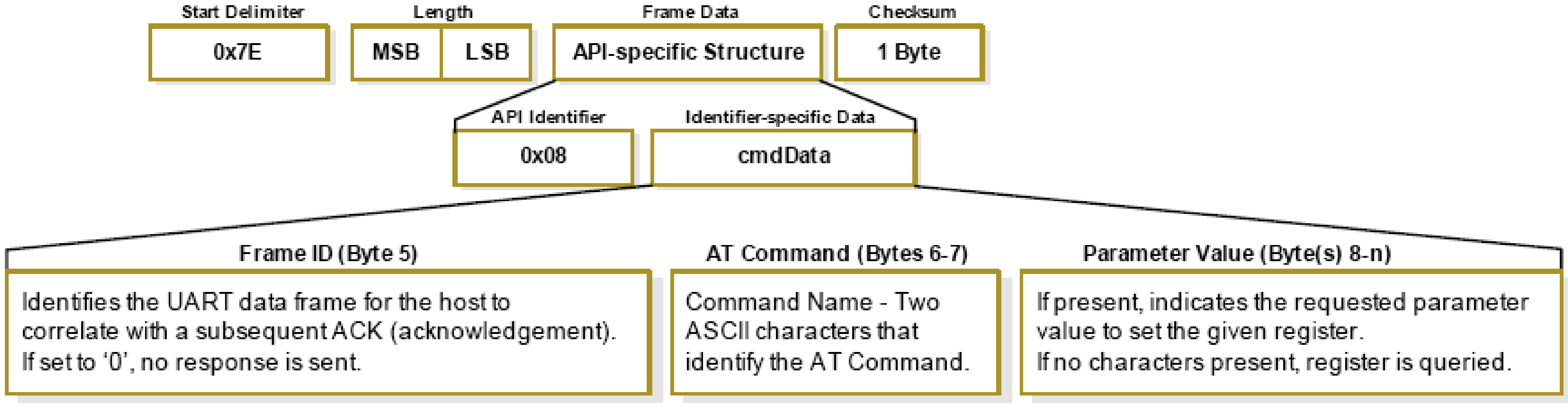
Many Kinds of Envelopes

Modem Status



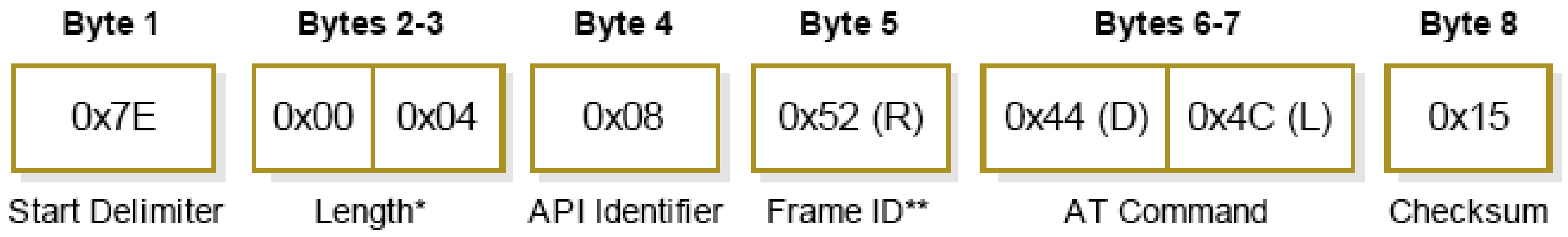
0 = Hardware reset
1 = Watchdog timer reset
2 = Associated
3 = Disassociated
4 = Synchronization Lost (Beacon-enabled only)
5 = Coordinator realignment
6 = Coordinator started

AT Command



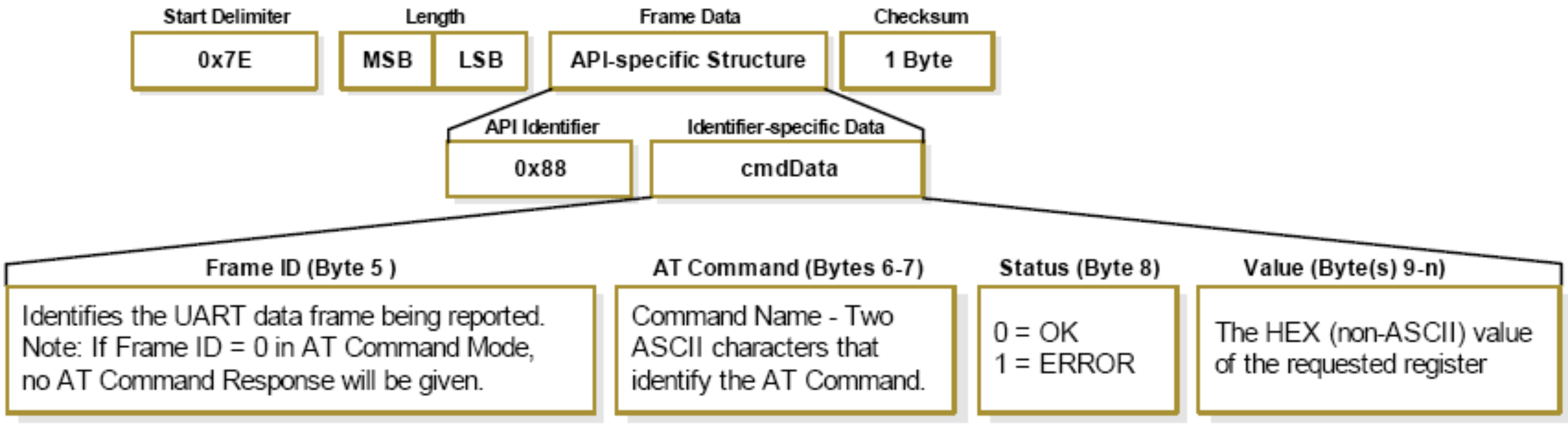
Start Delimiter
0x7E
Length
MSB
LSB
Frame Data
API-specific Structure
Checksum
1 Byte
API Identifier
0x08
Identifier-specific Data
cmdData
Frame ID (Byte 5)
Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to '0', no response is sent.
AT Command (Bytes 6-7)
Command Name - Two ASCII characters that identify the AT Command.
Parameter Value (Byte(s) 8-n)
If present, indicates the requested parameter value to set the given register. If no characters present, register is queried.

Figure 3-06. Example: API frames when reading the DL parameter value of the module.



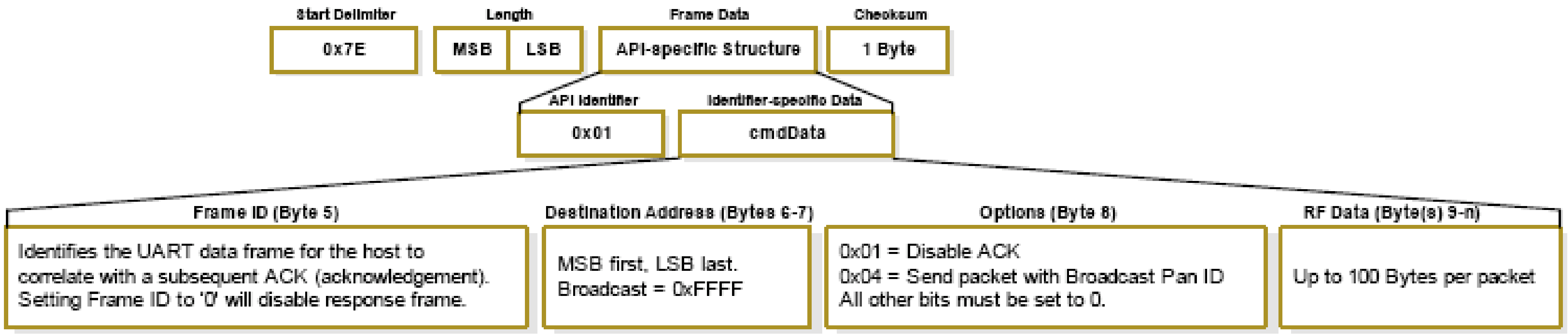
AT Response

- Frame ID for the response is the same as the matching AT Command request


Start Delimiter
0x7E
Length
MSB
LSB
Frame Data
API-specific Structure
Checksum
1 Byte
API Identifier
0x88
Identifier-specific Data
cmdData
Frame ID (Byte 5)
Identifies the UART data frame being reported. Note: If Frame ID = 0 in AT Command Mode, no AT Command Response will be given.
AT Command (Bytes 6-7)
Command Name - Two ASCII characters that identify the AT Command.
Status (Byte 8)
0 = OK
1 = ERROR
Value (Byte(s) 9-n)
The HEX (non-ASCII) value of the requested register

TX (Transmit) Request

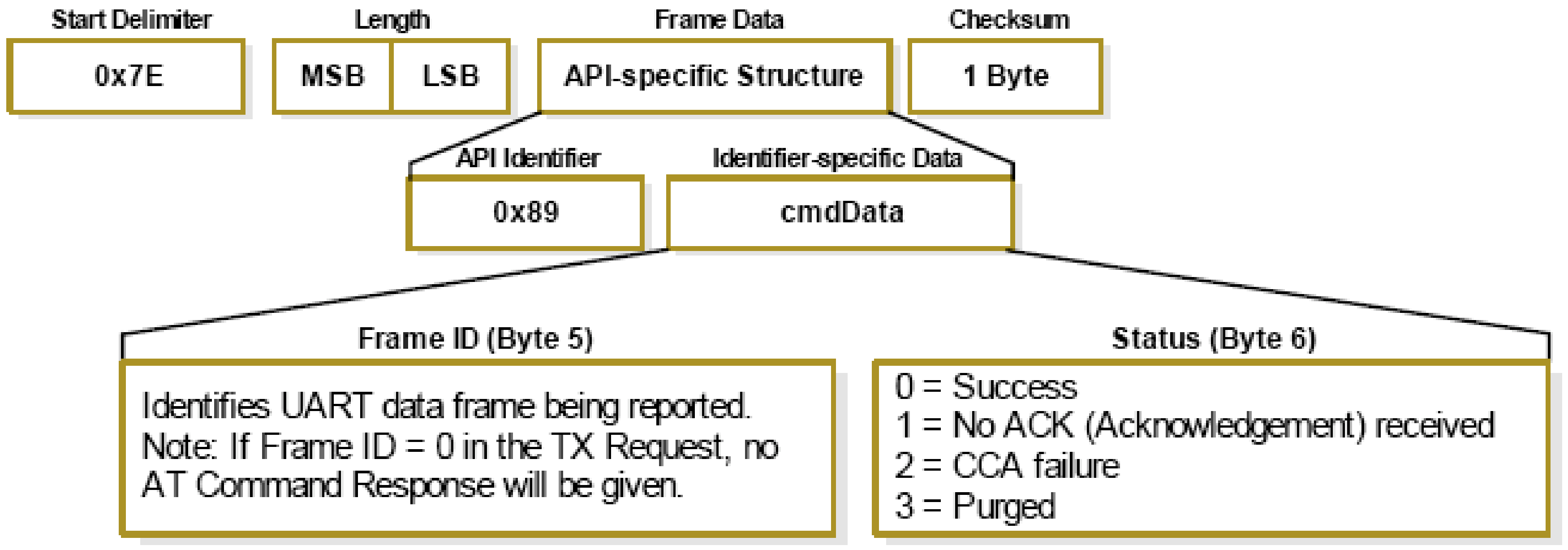
- Remember that this is a request
- Also need to check for results by Frame ID


Start Delimiter
0x7E
Length
MSB
LSB
Frame Data
API-specific Structure
Checksum
1 Byte
API Identifier
0x01
Identifier-specific Data
cmdData
Frame ID (Byte 5)
Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). Setting Frame ID to '0' will disable response frame.
Destination Address (Bytes 6-7)
MSB first, LSB last. Broadcast = 0xFFFF
Options (Byte 8)
0x01 = Disable ACK
0x04 = Send packet with Broadcast Pan ID
All other bits must be set to 0.
RF Data (Byte(s) 9-n)
Up to 100 Bytes per packet

TX Status (Results)

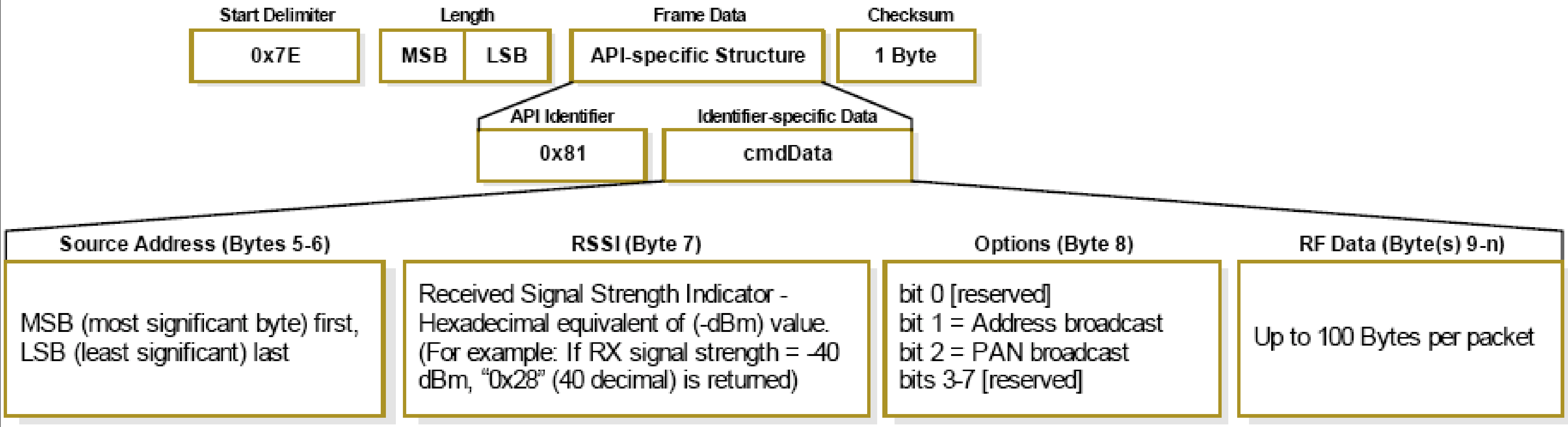
- See if your message was transmitted or not

- Use your Frame ID to see which message is being described



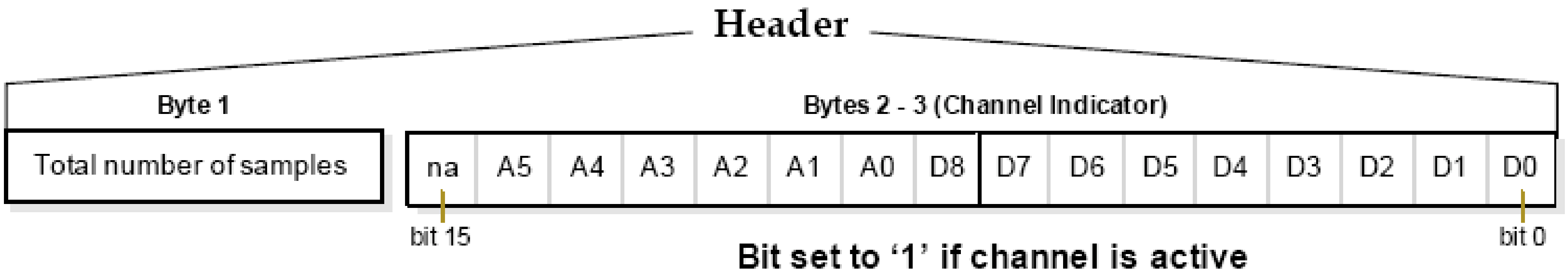
RX Packet (16 bit addressing)

- Maximum of 100 bytes of data per packet
- RF Data section is basis for I/O packets


Start Delimiter
0x7E
Length
MSB
LSB
Frame Data
API-specific Structure
Checksum
1 Byte
API Identifier
0x81
Identifier-specific Data
cmdData
Source Address (Bytes 5-6)
MSB (most significant byte) first, LSB (least significant) last
RSSI (Byte 7)
Received Signal Strength Indicator - Hexadecimal equivalent of (-dBm) value. (For example: If RX signal strength = -40 dBm, "0x28" (40 decimal) is returned)
Options (Byte 8)
bit 0 [reserved]
bit 1 = Address broadcast
bit 2 = PAN broadcast
bits 3-7 [reserved]
RF Data (Byte(s) 9-n)
Up to 100 Bytes per packet

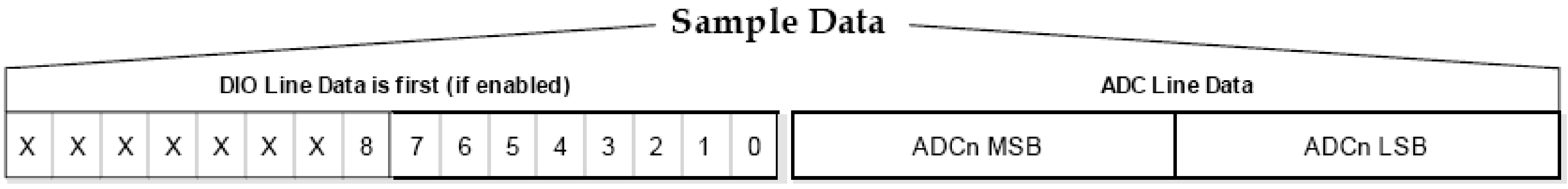
I/O Data Header

- Inside the RF Data section of the RX Packet
  - Total number of samples set with ATIS
  - Channels set with ATD0...9



I/O Data Sample

- Follows the header
- Two bytes of digital data IF ANY DIGITAL CHANNELS ENABLED followed by...
- ...two bytes for EACH analog channel enabled...
- ...then repeats for each sample


Sample Data
DIO Line Data is first (if enabled)
ADC Line Data
X X X X X X X 8 7 6 5 4 3 2 1 0
ADCn MSB
ADCn LSB

- How many bytes if ATIS5 ATD02 ATD12 ATD23?

I/O Code: Basic

- Fixed parameters make for easier programming
- Assume we are just reading a single sample of one ADC channel at a time:

```
 // PROCESSING VERSION:
int totalSamples = port.read(); // this is the number of samples we're receiving
int channelIndicatorHigh = port.read(); // this tells us which analog channels
                   // are in use (and one digital channel)
int channelIndicatorLow = port.read();  // this tells us which digital channels
                   // are in use.

int  dataADCMSB = port.read(); // read in the most significant ADC byte
int dataADCLSB = port.read(); // read in the least significant ADC byte
int dataADC = (dataADCMSB << 8) + dataADCLSB;  // bit shift the MSB into
                   // position and add it to the LSB

      print(dataADC);  // print the information
}
```

XBee Sleep Mode

Sleeping the XBee: Basics

- Why Sleep?
- ATSM
  - 1: pin hibernate, <10 μA, 13.2 ms wakeup, uses pin 9
  - 2: pin doze, <50 μA, 2 ms wakeup
  - 3: <nothing>
  - 4: cyclic sleep, also <50 μA, 2 ms wakeup, module must be idle
  - 5: cyclic sleep with pin wakeup
- ATSP: Sleep Period (* 10 ms)
- ATST: Time before Sleep (* 1 ms)

Sleeping the XBee: Example

- ATSM5,SP64,ST14
  - Will wake up on pin 9 high, and also every 1000 ms for 20 ms
- Use in conjunction with I/O readings
  - Wakeup will always trigger an I/O sample
  - More samples if ATIR allows it during the awake period
  - More samples if ATIT (Samples before TX) is set
- ATIC (Pin Change Detect) will not affect wakeup

XBee ZigBee

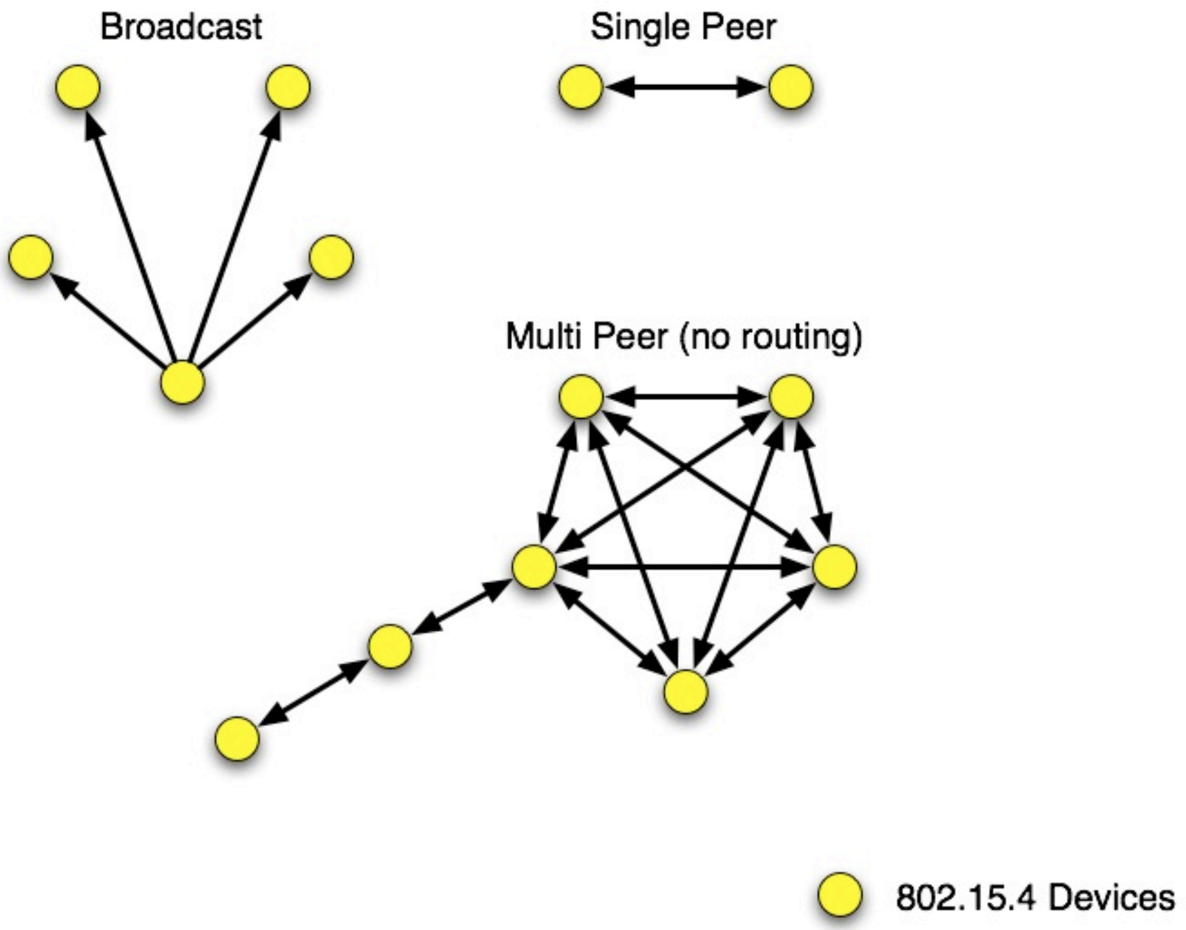
XBee Series 1 vs. Series 2

- **802.15.4 (SERIES 1)**
    - 802.15.4 only
    - ADC & Digital I/O
    - point-to-point networking
    - unicast or broadcast
    - low power with good range
    - mature and simpler

- **ZNet 2.5 (SERIES 2) and ZB**
    - ZigBee only
    - I/O with less Analog and 1.2V
    - Full ZigBee mesh networking
    - unicast, broadcast or multicast
    - slightly better range & power
    - newer more complexity

802.15.4 Topologies

- single peer
- multi-peer
- broadcast



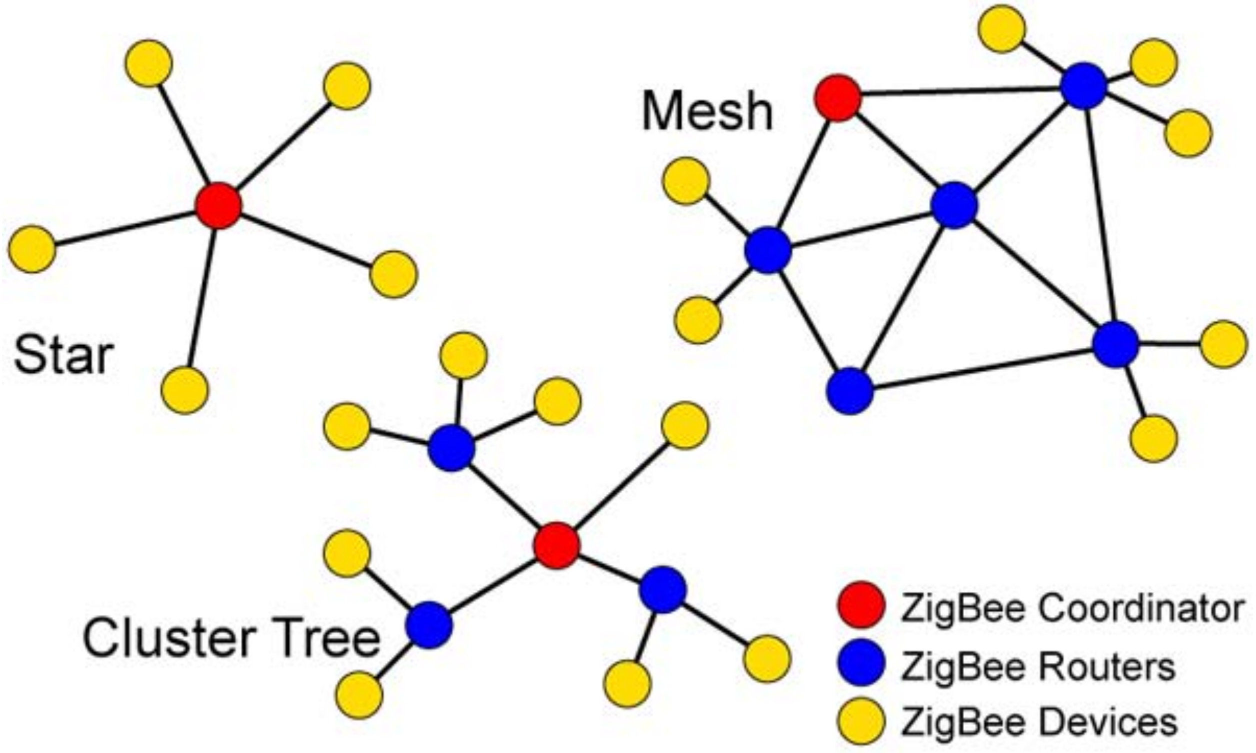
ZigBee Basics

- Coordinator
- Routers
- End devices

- A ZigBee network is minimally: 1 coordinator and 1 router (or end device)

ZigBee Topologies

- peer
- star
- mesh
- routing

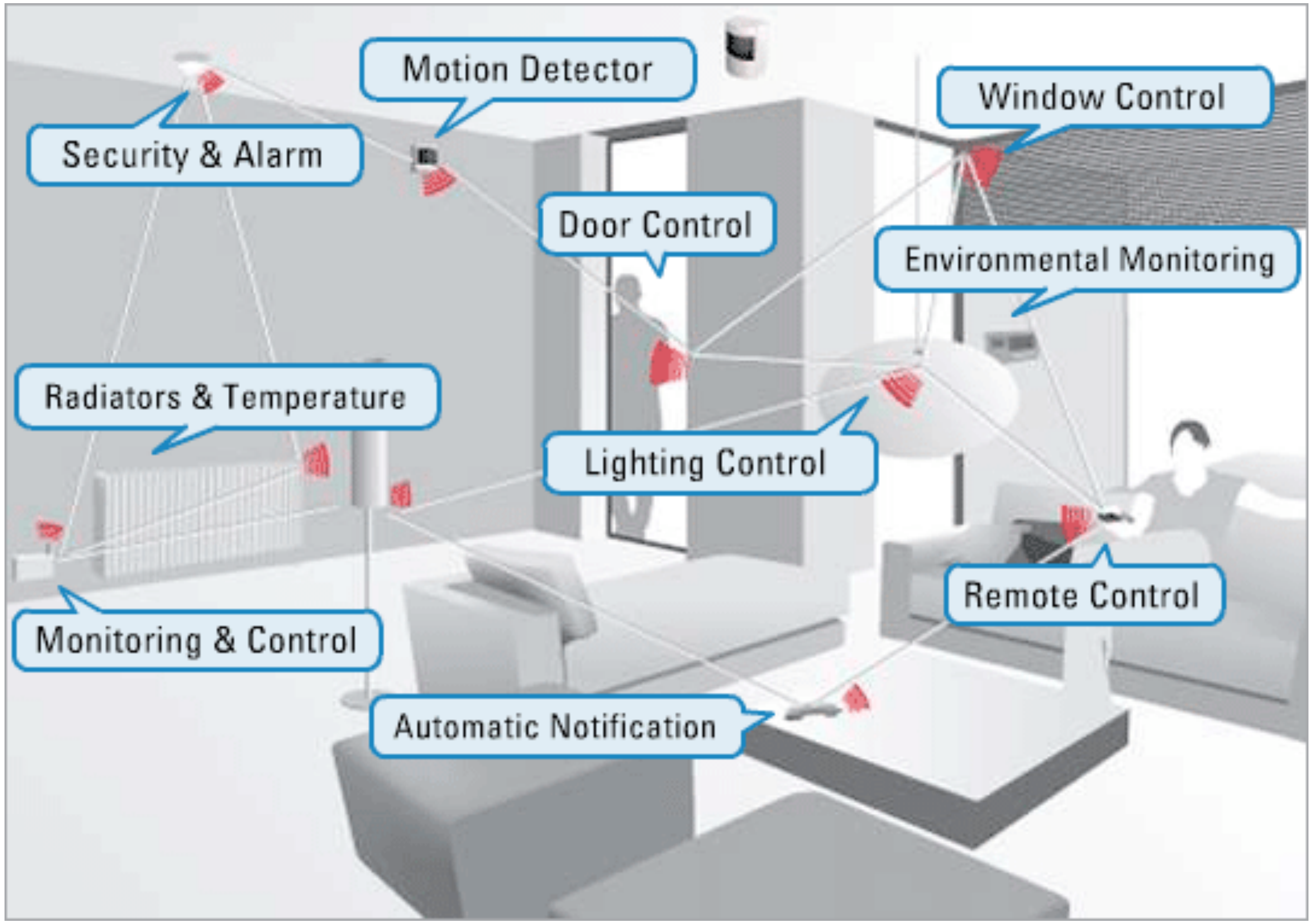


ZigBee Coordinator

- Every ZigBee network <u>must</u> have a coordinator
- There can only be <u>one</u> coordinator
- Coordinator selects channel and PAN ID
- End devices and routers can then join the PAN
- Typically mains-powered
- Coordinator’s 16-bit address is always 0

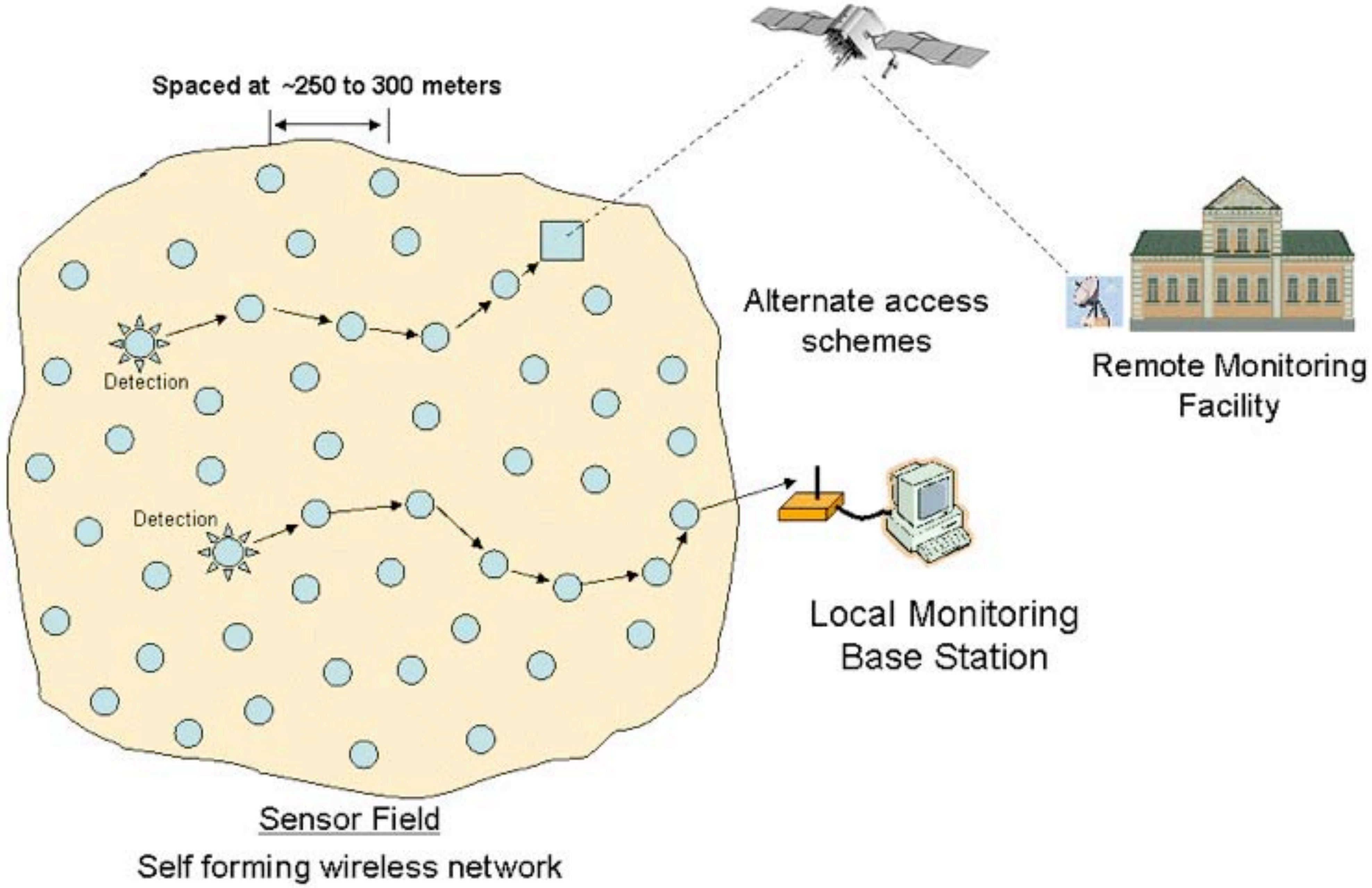
ZigBee Router

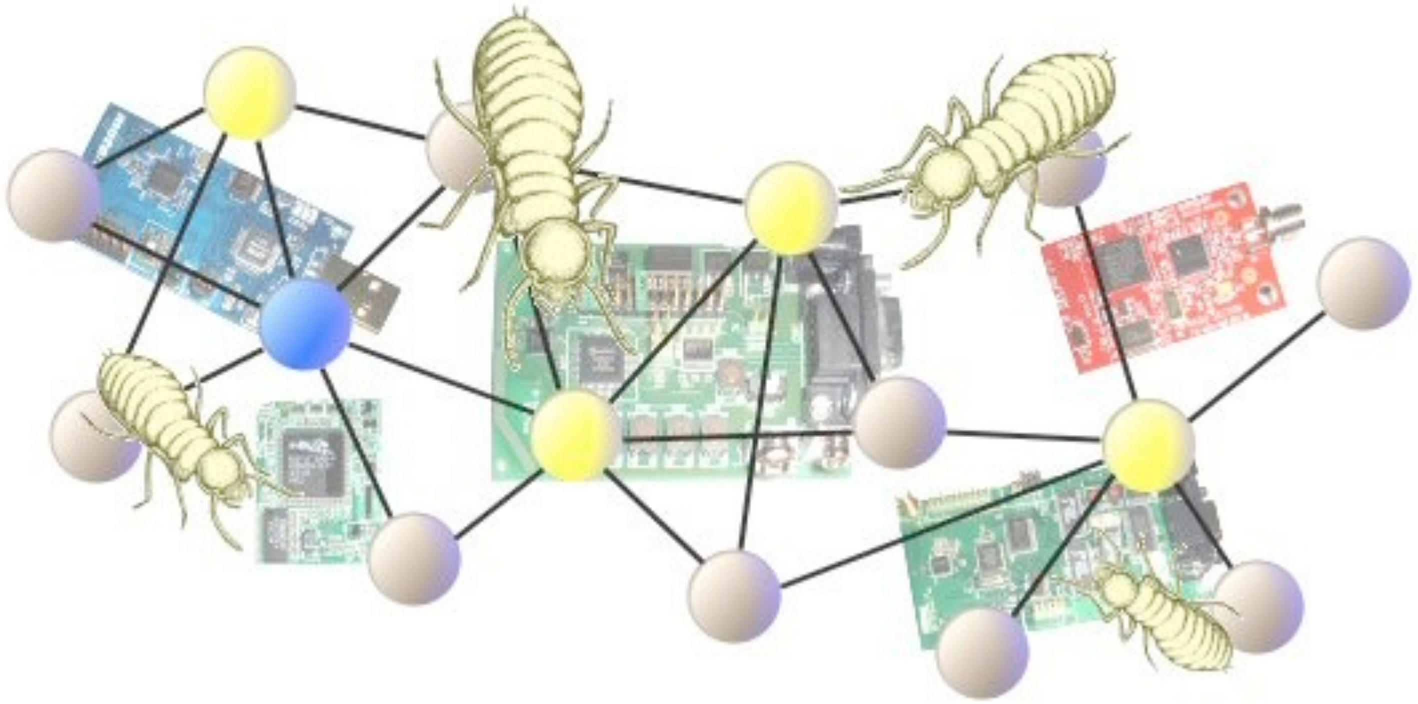
- Non-coordinator routers are optional to ZigBee networks
- Typically mains-powered
- Many can be on each PAN
- Issues a beacon request on startup to locate channel and PAN
- Routers can communicate with any device on the network
- Stores packets for sleeping end devices
- 16-bit address assigned by coordinator


Motion Detector
Security & Alarm
Window Control
Door Control
Environmental Monitoring
Radiators & Temperature
Lighting Control
Remote Control
Monitoring & Control
Automatic Notification

ZigBee End Device

- Optional to ZigBee networks
- Typically battery-powered
- Many can be on each PAN
- Issues a beacon request on startup to locate channel and PAN
- Automatically attempts to join a valid PAN
- End devices can only communicate directly with their parent
- 16-bit address assigned by coordinator


Spaced at ~250 to 300 meters
Detection
Detection
Alternate access schemes
Remote Monitoring Facility
Local Monitoring Base Station
Sensor Field
Self forming wireless network



http://www.stg.com/wireless/ZigBee_Termites.html

XBee Series 2

- Coordinator Firmware
  - for AT commands or API
- Router/End Device Firmware
  - for AT commands or API
- ...so 4 different firmware combinations (you’ll always use 2 at the same time)
- and 4 antennas! whip, chip, U.FL and RPSMA.

Special Features

- Remote AT commands
  - send an AT command request to another node
  - only works in API mode, which means using API version of firmware
- Loopback: ATZA1,CI12,SEE8,DEE8 and then pick a destination node
- Join Indicators: ATJN to send a notification to the coordinator on join
- Battery Monitoring: AT%V for value then (value/1023*1200 = mV)

Starting Up an XBee ZigBee Network

- Coordinator:
    - scans and selects a channel
    - picks a PAN or uses a predetermined one
    - Associate light blinks, ATAI is set to zero (or a value indicating error)
- Router or End Device
    - scans for PANs on each channel
    - selects a PAN to join (often the predetermined one)
    - sends a beacon request to join to a parent router or coordinator
    - Associate light blinks, ATAI is set to zero (or a value indicating error)

Transmitting Data

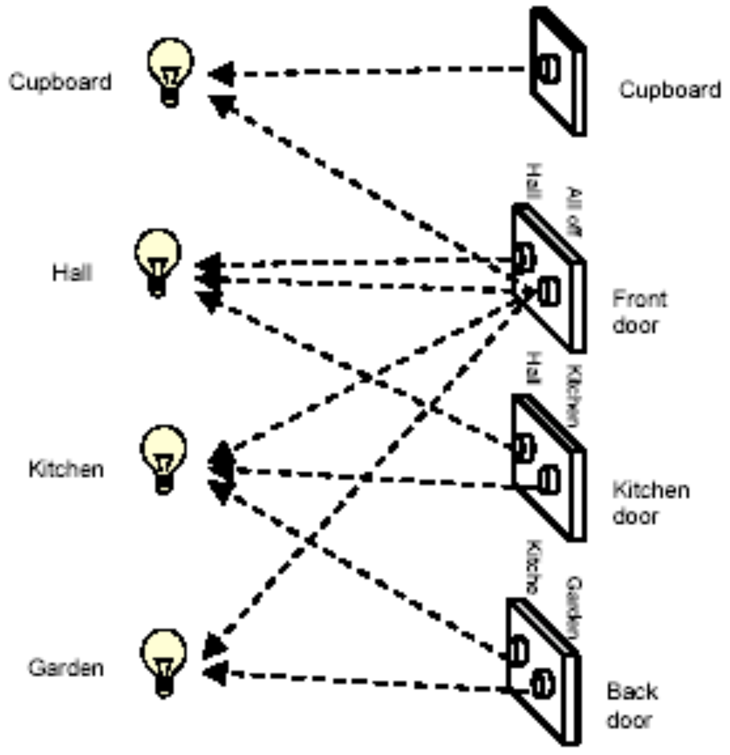
- Read a list of all nodes on the network using ATND

  MY<CR>
  SH<CR>
  SL<CR>
  NI<CR> (Variable length)
  PARENT_NETWORK ADDRESS (2 Bytes)<CR>
  DEVICE_TYPE<CR> (1 Byte: 0=Coord, 1=Router, 2=End Device)
  STATUS<CR> (1 Byte: Reserved)
  PROFILE_ID<CR> (2 Bytes)
  MANUFACTURER_ID<CR> (2 Bytes)
  <CR>

- Set the Destination Node using ATDN

Endpoints, Clusters and Bindings

- Another addressing scheme for defining groups of radios and actions, typically for home networking.
- Beyond the scope of this class and not immediately useful to us.



Readings and Assignments

- Assignment
    - Final Project