Sociable Objects Workshop

Instructor: Rob Faludi

Plan for Today

- Doorbell Projects: review
- I/O Mode
- I/O Demo
- Voltage Dividers
- API Mode Overview
- API Mode Details
- Readings & Assignments

Doorbell Projects Review

I/O Mode

Direct, Indirect, Subtext

- What data can we sense directly?
- How about inferences that we can make from the data?
- What's the subtext of the data? What can we infer from the inference?

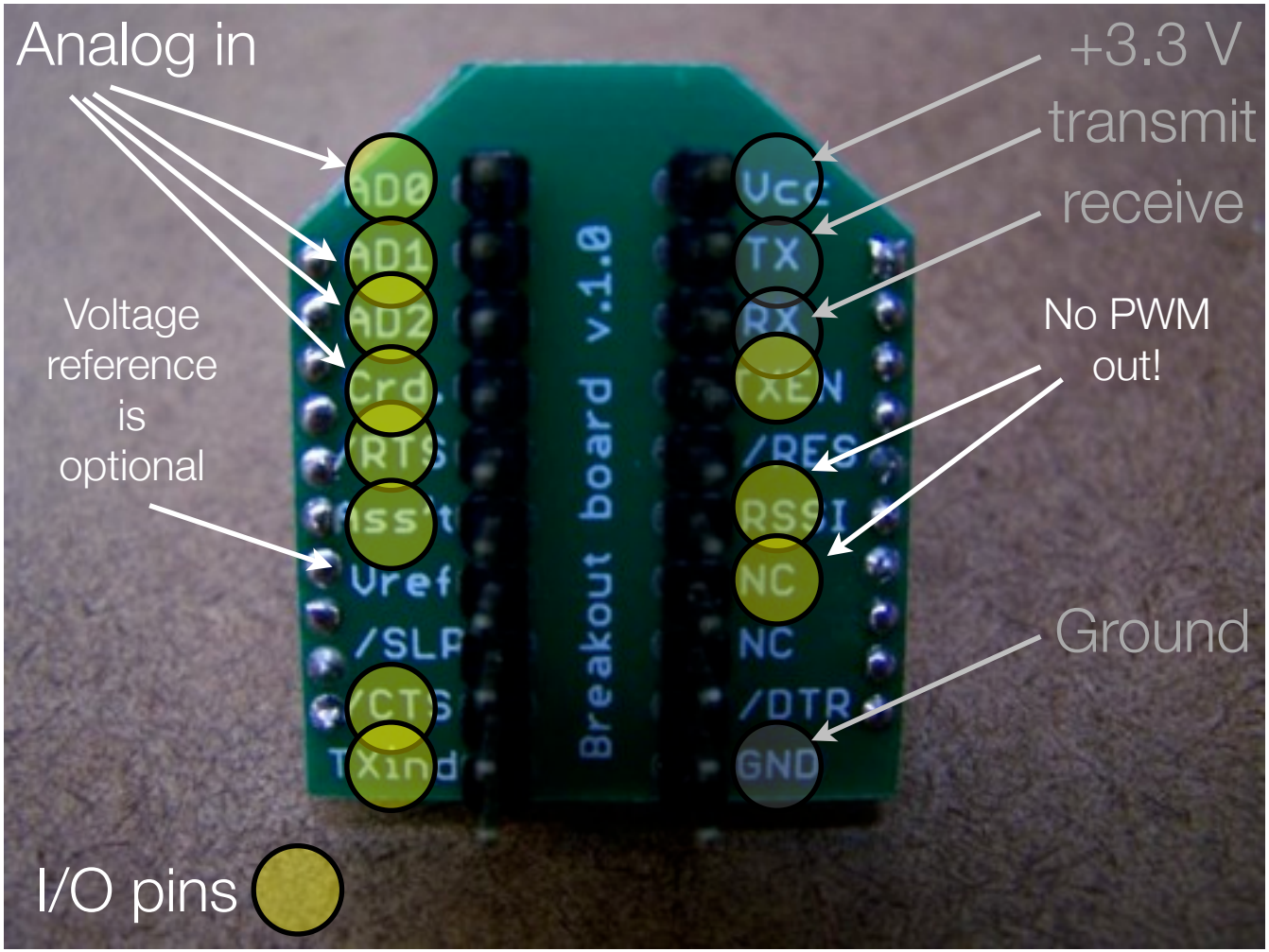
I/O Intro: ZigBee

- For simple input and/or output
- Ten digital input/outputs
- Four analog inputs
- No analog outputs on ZigBee
- But not all at once! Pins are shared.

I/O Why

- Why:
  - Save space, save power, save weight and save money
  - Reduce complications
- Why not:
  - Limited inputs/outputs
  - No access to logic
  - No analog output on ZigBee radios

Input/Output Wiring: ZigBee



I/O AT Commands: ZigBee

- ATD0...D7 -> configure pins for I/O (D8 and D9 not supported yet)
- ATP0...P1 -> configure pins 10 - 11 for I/O (P3 not supported yet)
- ATIR -> sample rate
- samples before transmit is always 1
- destination address receives sample info
- ALL PINS READ BETWEEN 0 AND 1.2 VOLTS ONLY

Example Configuration

- SENDER:
  ATID3456 (PAN ID)
  ATDH -> set to SH of partner radio
  ATDL -> set to SL of partner radio
  ATJV1 -> rejoin with coordinator on startup
  ATD02 pin 0 in analog in mode
  ATD13 pin 1 in digital in mode
  ATIR64 sample rate 100 millisecs (hex 64)

- RECEIVER
  ATID3456 (PAN ID)
  ATDH -> set to SH of partner radio
  ATDL -> set to SL of partner radio

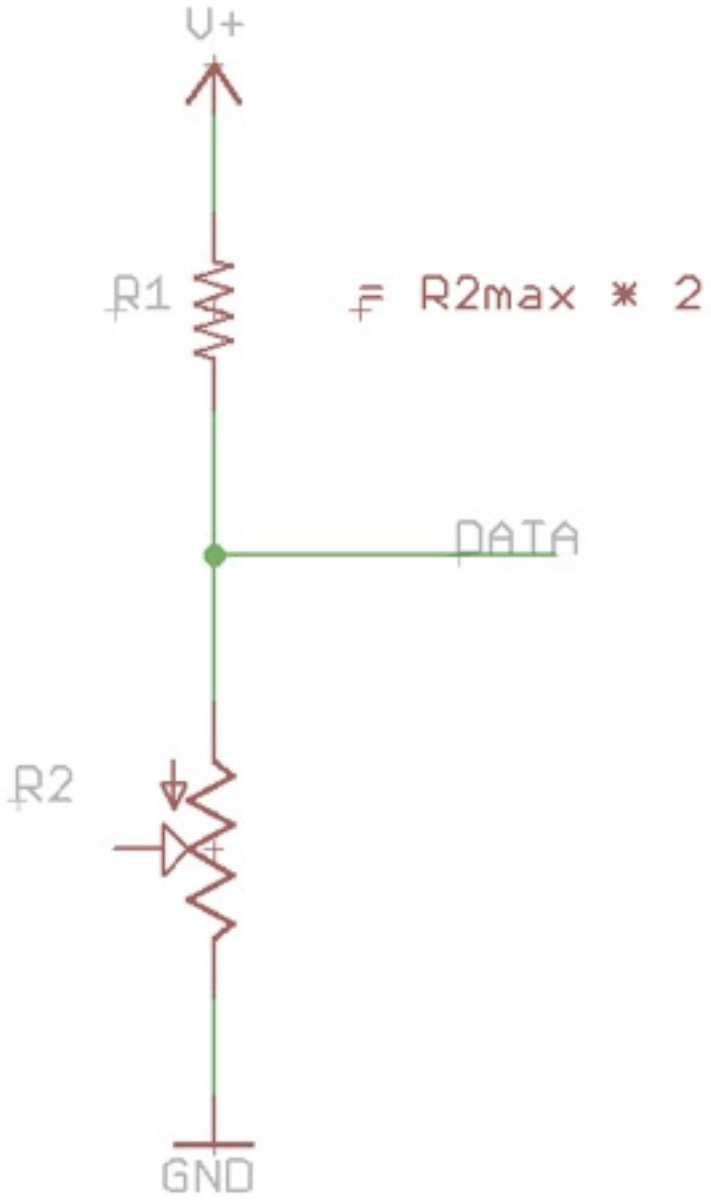
Settting I/O Pins

- ATDx 0 Disabled
- ATDx 1 Built-in Function (sometimes)
- ATDx 2 Analog Input (sometimes)
- ATDx 3 Digital Input
- ATDx 4 Digital Output, low to start with
- ATDx5 Digital Output, high to start with
    - ...so ATD32 would set digital pin 3 to analog input mode

I/O Demo

XBee ZigBees inputs are 1.2V range

Voltage Divider to map 3.3V range to 1.2V range



Romantic Lighting Sensor

API Mode Overview

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;
   }
}
```

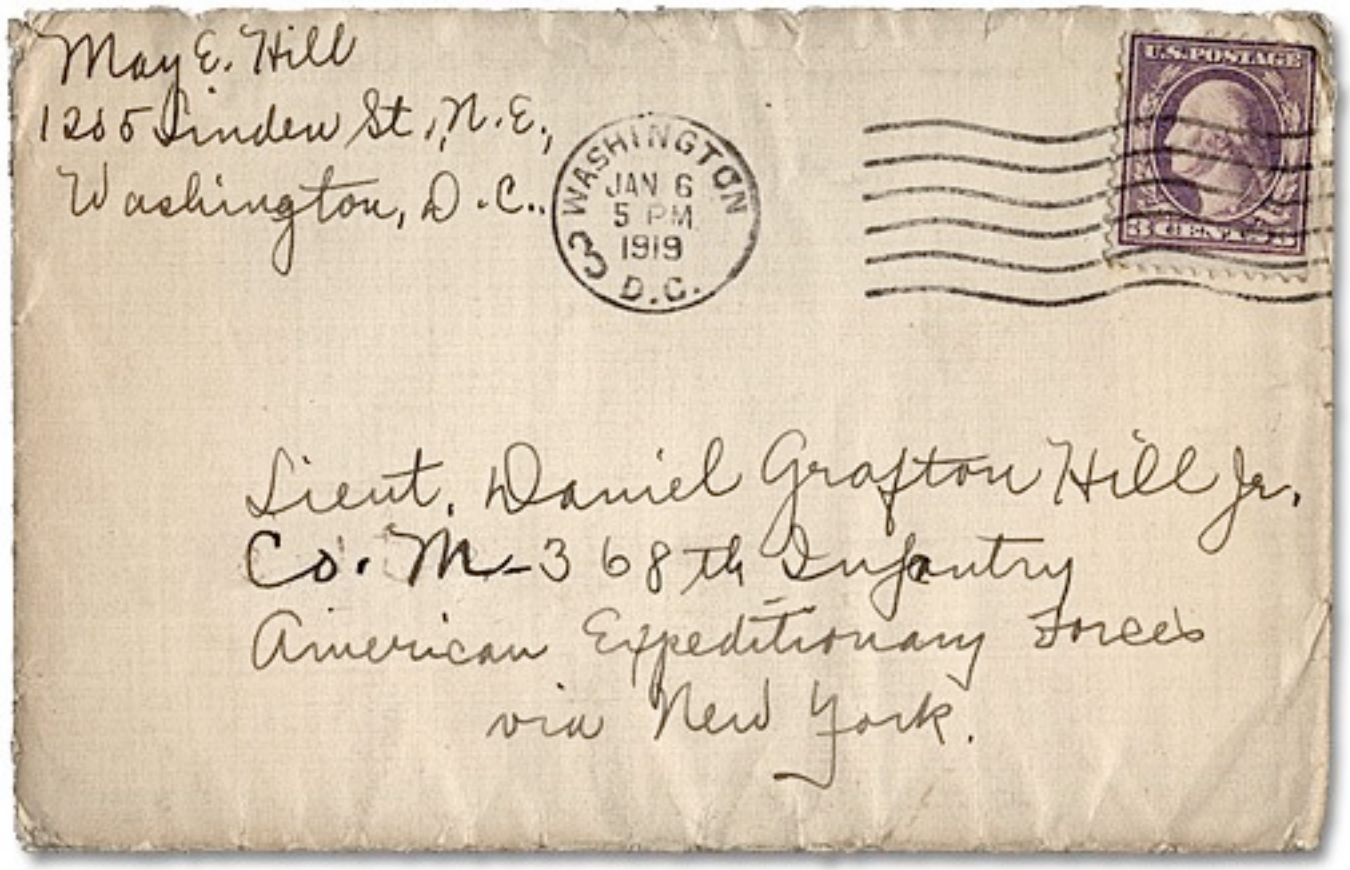
- With a library you just write:

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

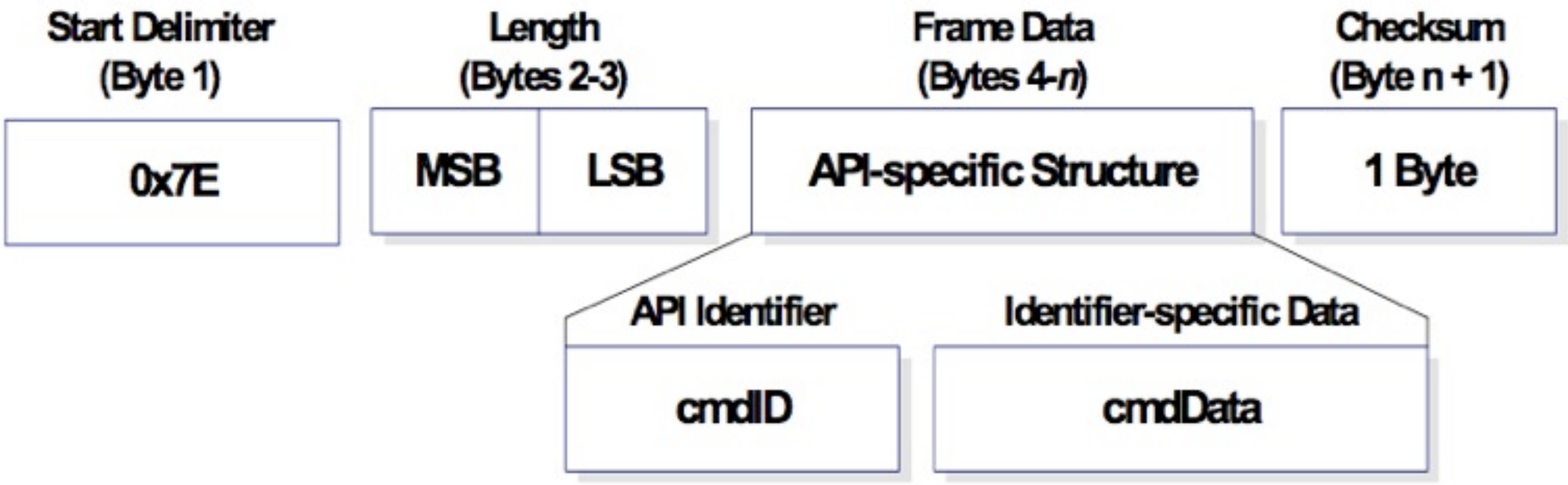
API Mode Details

Envelope Has:

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



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: 0x10
  - TX status response: 0x8B
  - RX packet: 0x90
  - RX packet I/O data: 0x92

```
 // 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)

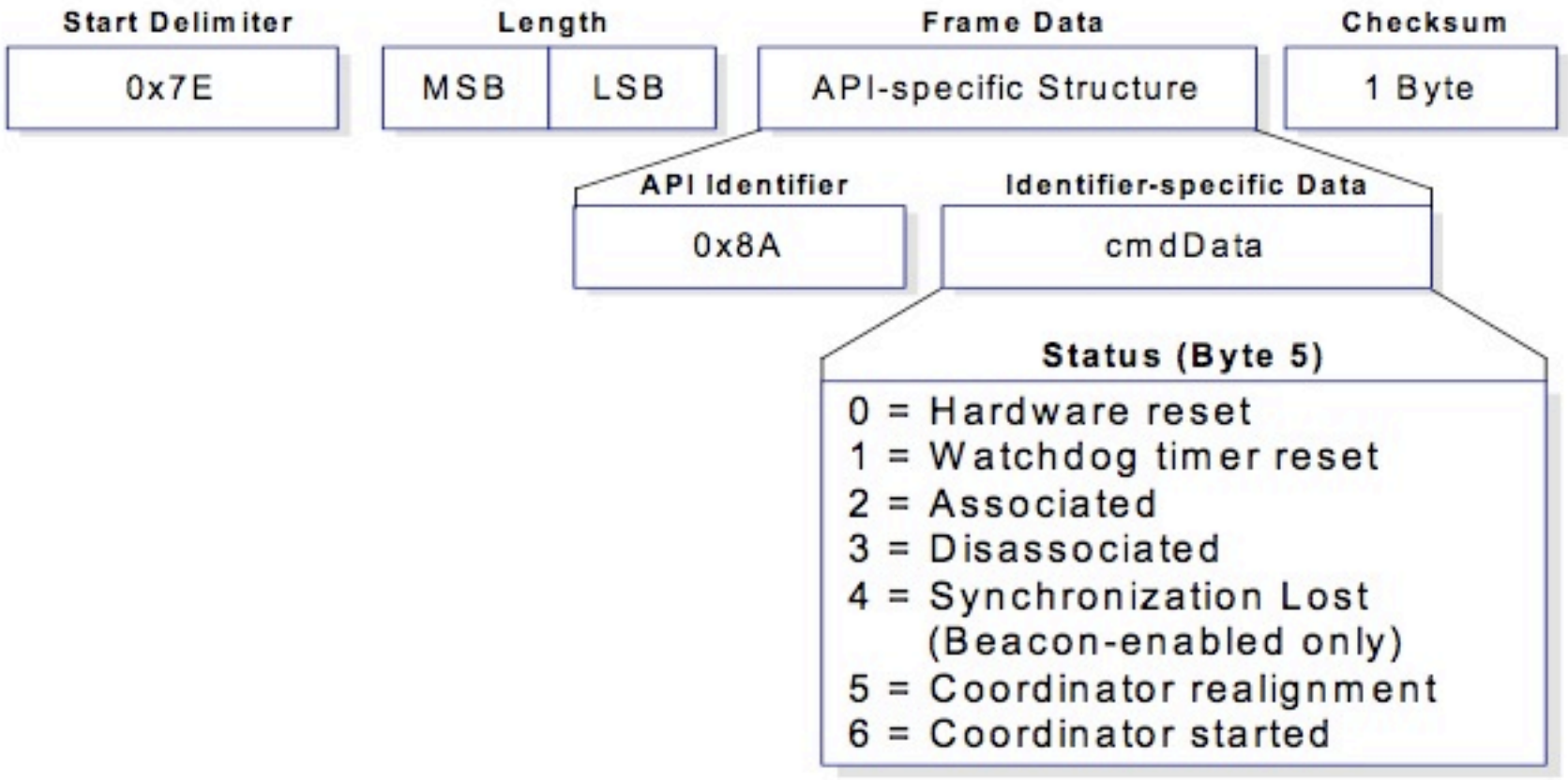
Checksum

- 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.

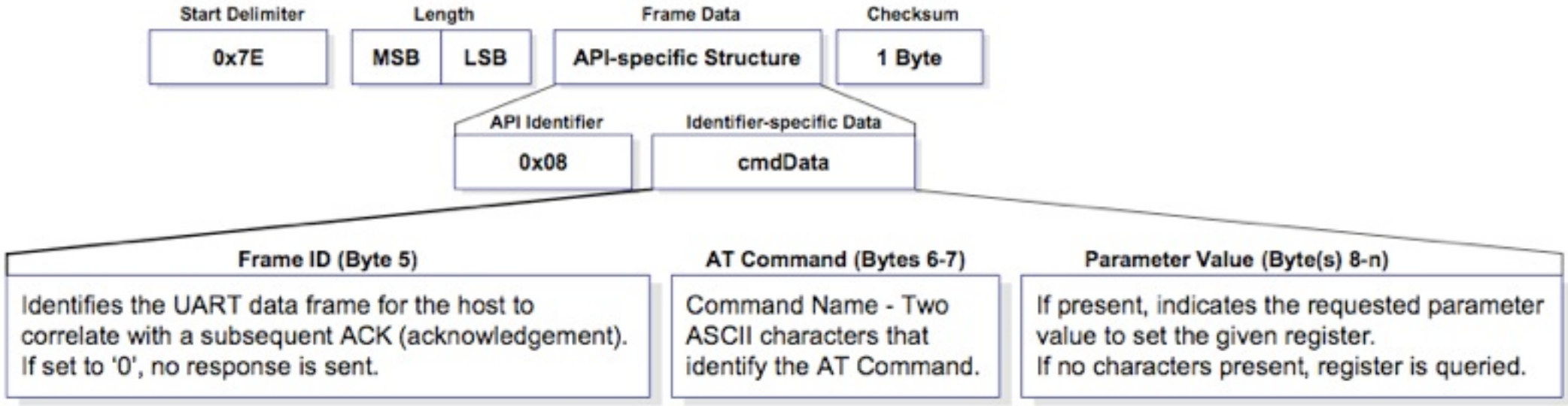
```
 // PROCESSING VERSION:
int localChecksum = (API_ID + addrMSB + addrLSB + RSSI + options + dataSum);

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

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

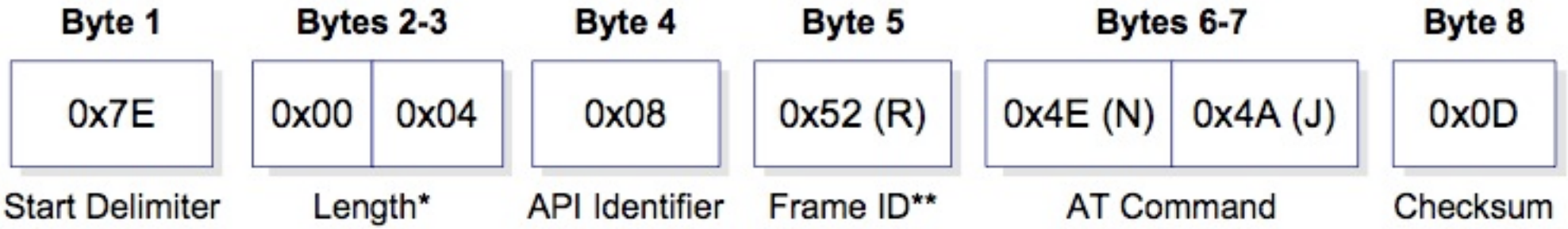
Many Kinds of Envelopes

Modem Status: ZigBee



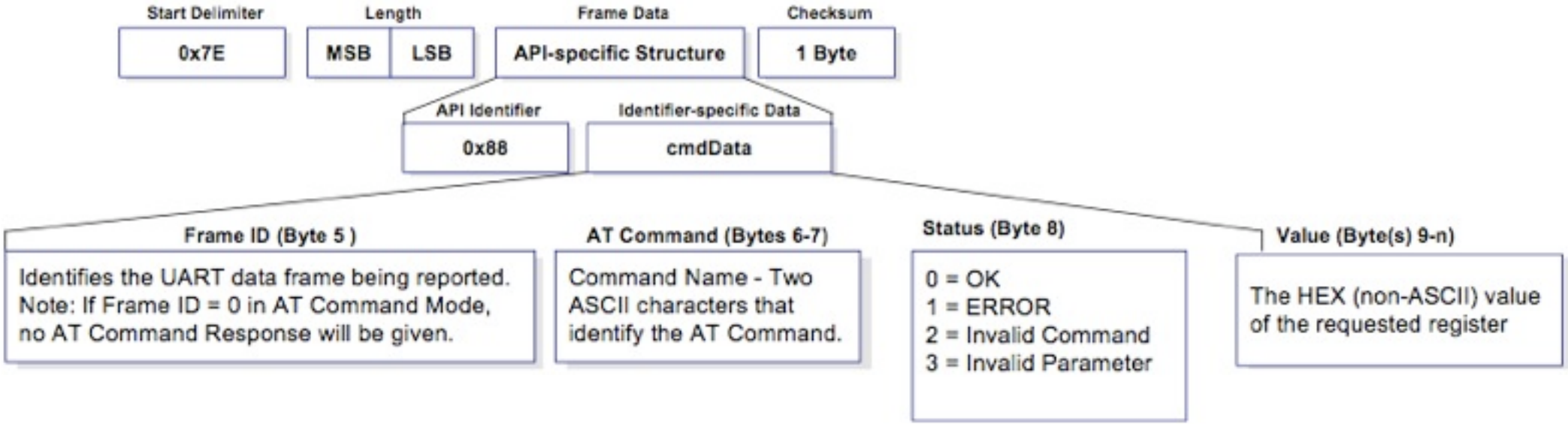
AT Command


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.


Byte 1
0x7E
Start Delimiter
Bytes 2-3
0x00
0x04
Length*
Byte 4
0x08
API Identifier
Byte 5
0x52 (R)
Frame ID**
Bytes 6-7
0x4E (N)
0x4A (J)
AT Command
Byte 8
0x0D
Checksum

AT Response

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



Readings and Assignments

- Readings
    - XBee ZB Manual: API mode and I/O mode sections
- Assignments
    - Complete Doorbells
    - Romance Light Sensor
    - Romance Light with Feedback