Sociable Objects Workshop

Instructor: Rob Faludi

Plan for Today

- Doorbell Projects: review
- I/O Mode
- I/O Demo
- 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?

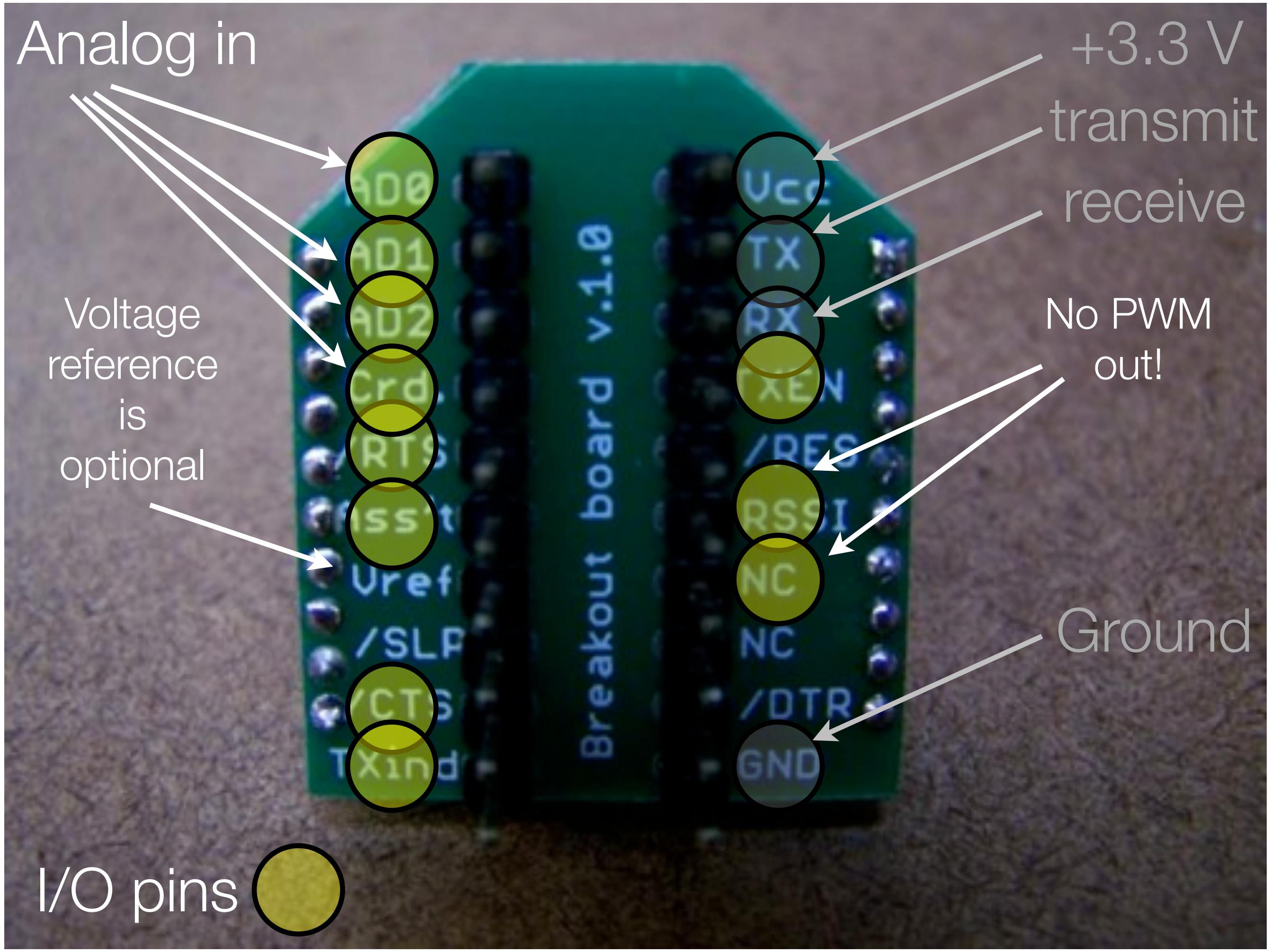
I/O Intro: ZigBee

- 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



Analog in

+3.3 V

transmit

receive

Voltage reference is optional

No PWM out!

Ground

I/O pins ●

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

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 <u>Output</u>, low to start with
- ATDx5   Digital <u>Output</u>, high to start with
    - ...so ATD32 would set digital pin 3 to analog input mode

I/O Demo

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;
    }
 }
```

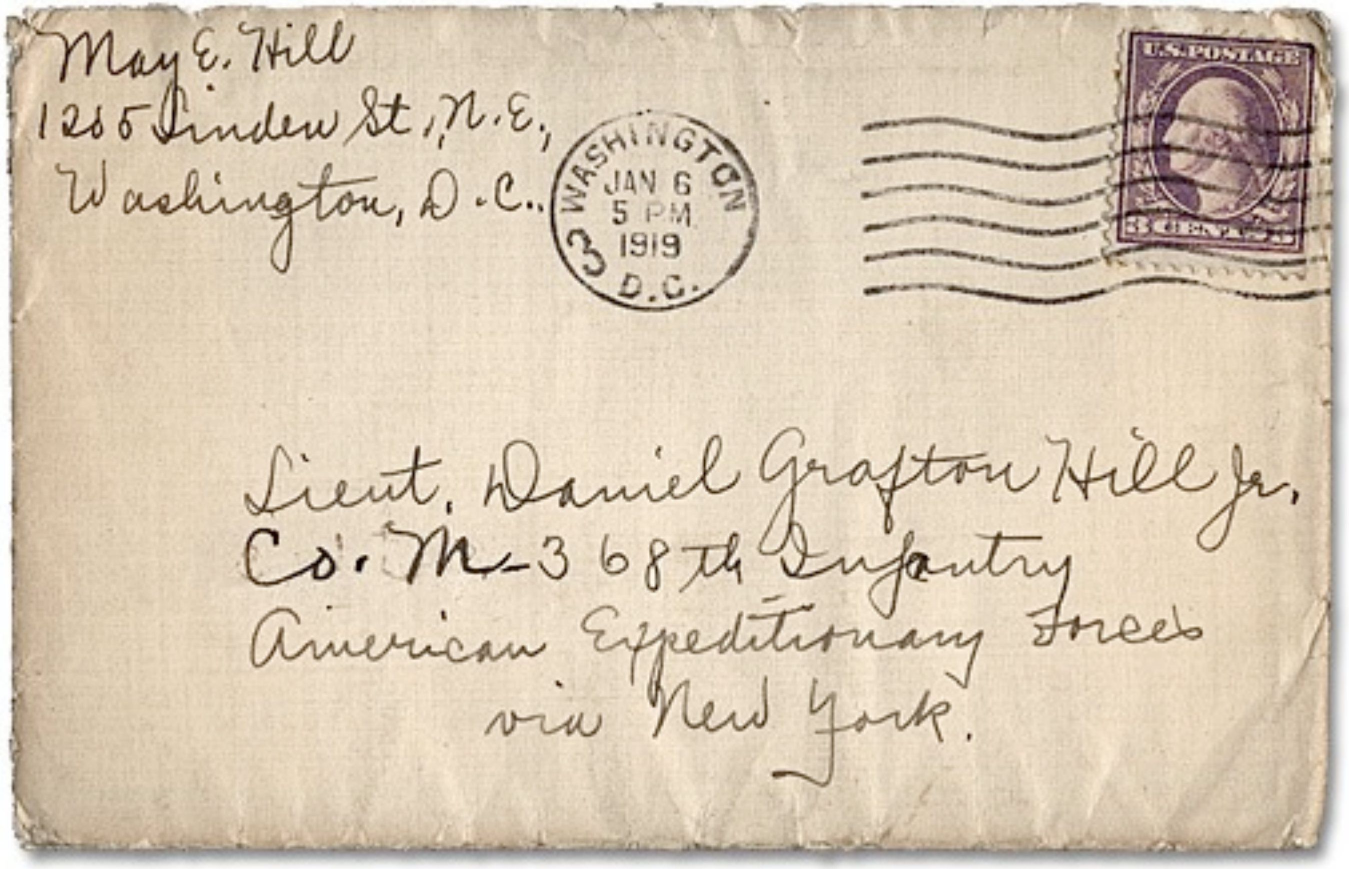
- With a library you just write:

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

API Mode Details

Envelope Has:

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



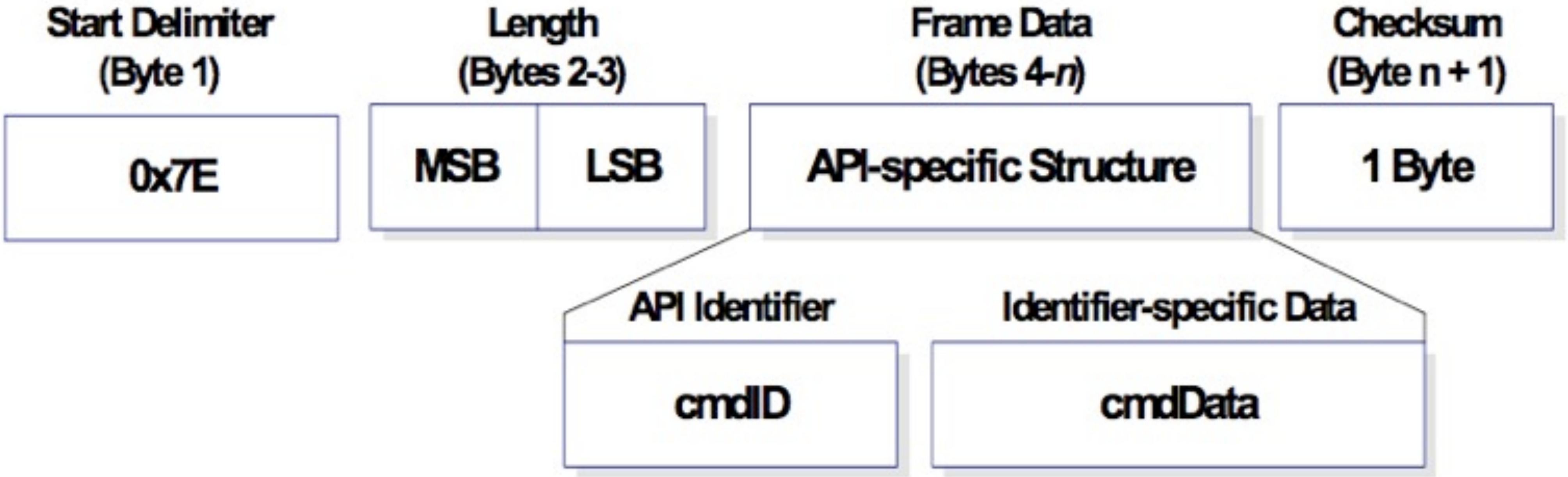
May E. Hill
1205 Linden St, N.E.,
Washington, D.C.

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 Delimiter (Byte 1)

0x7E

Length (Bytes 2-3)

MSB | LSB

Frame Data (Bytes 4-*n*)

API-specific Structure

Checksum (Byte n + 1)

1 Byte

API Identifier

cmdID

Identifier-specific Data

cmdData

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.

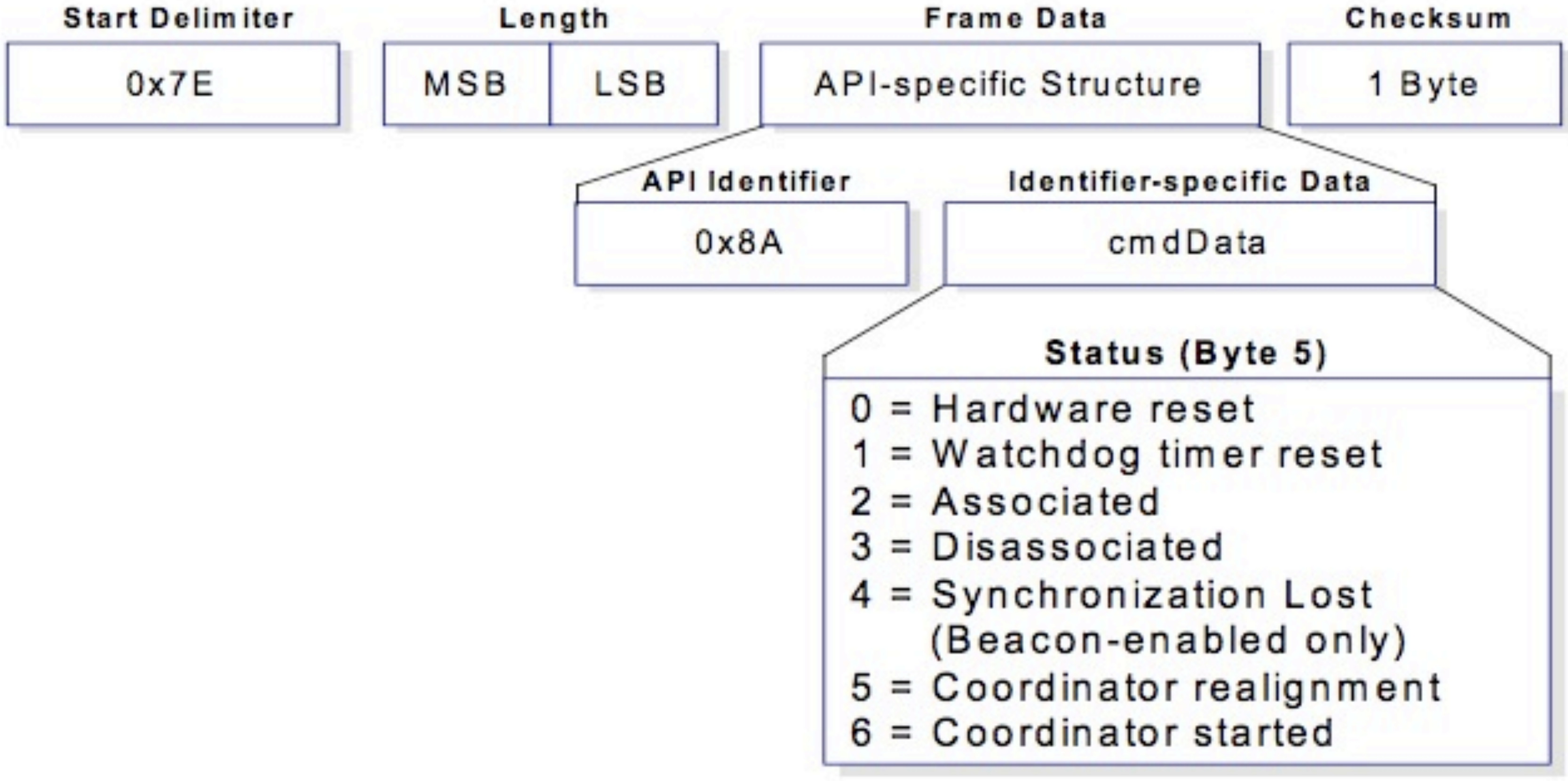
```
 // PROCESSING VERSION:
int localChecksum = (API_ID + addrMSB + addrLSB + RSSI + options + dataSum);

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

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

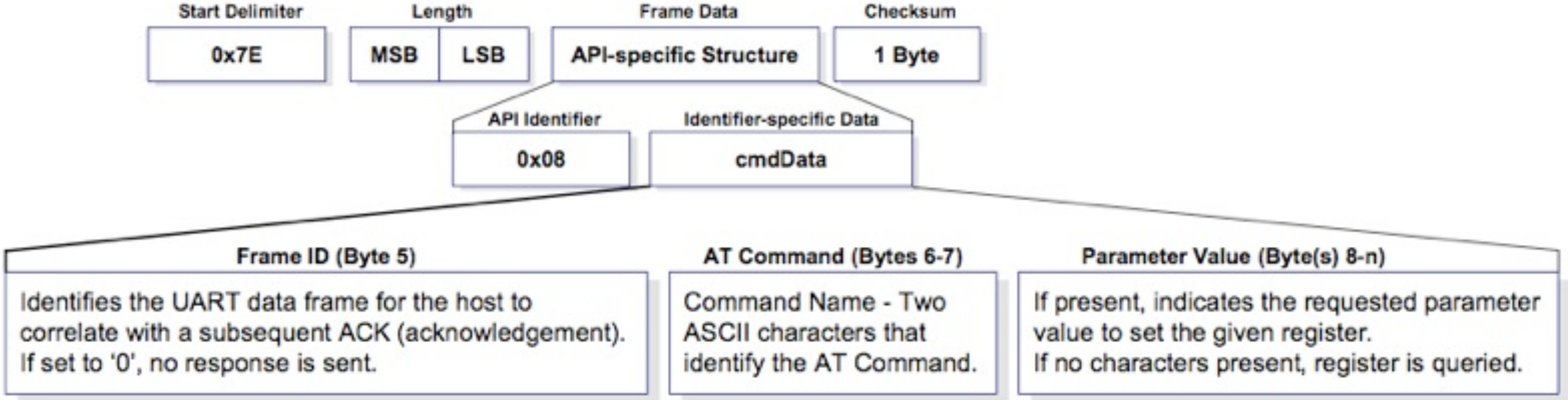
Many Kinds of Envelopes

Modem Status: ZigBee



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



Identifies the UART data frame for the host to correlate with a subsequent ACK (acknowledgement). If set to '0', no response is sent.

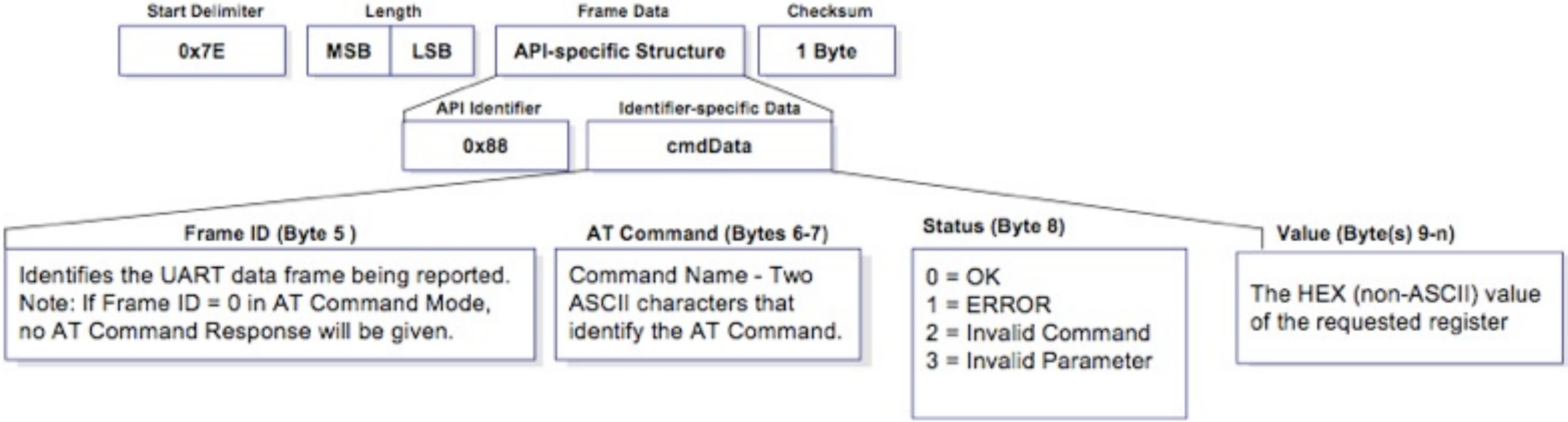
Command Name - Two ASCII characters that identify the AT Command.

If present, indicates the requested parameter value to set the given register. If no characters present, register is queried.

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

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)

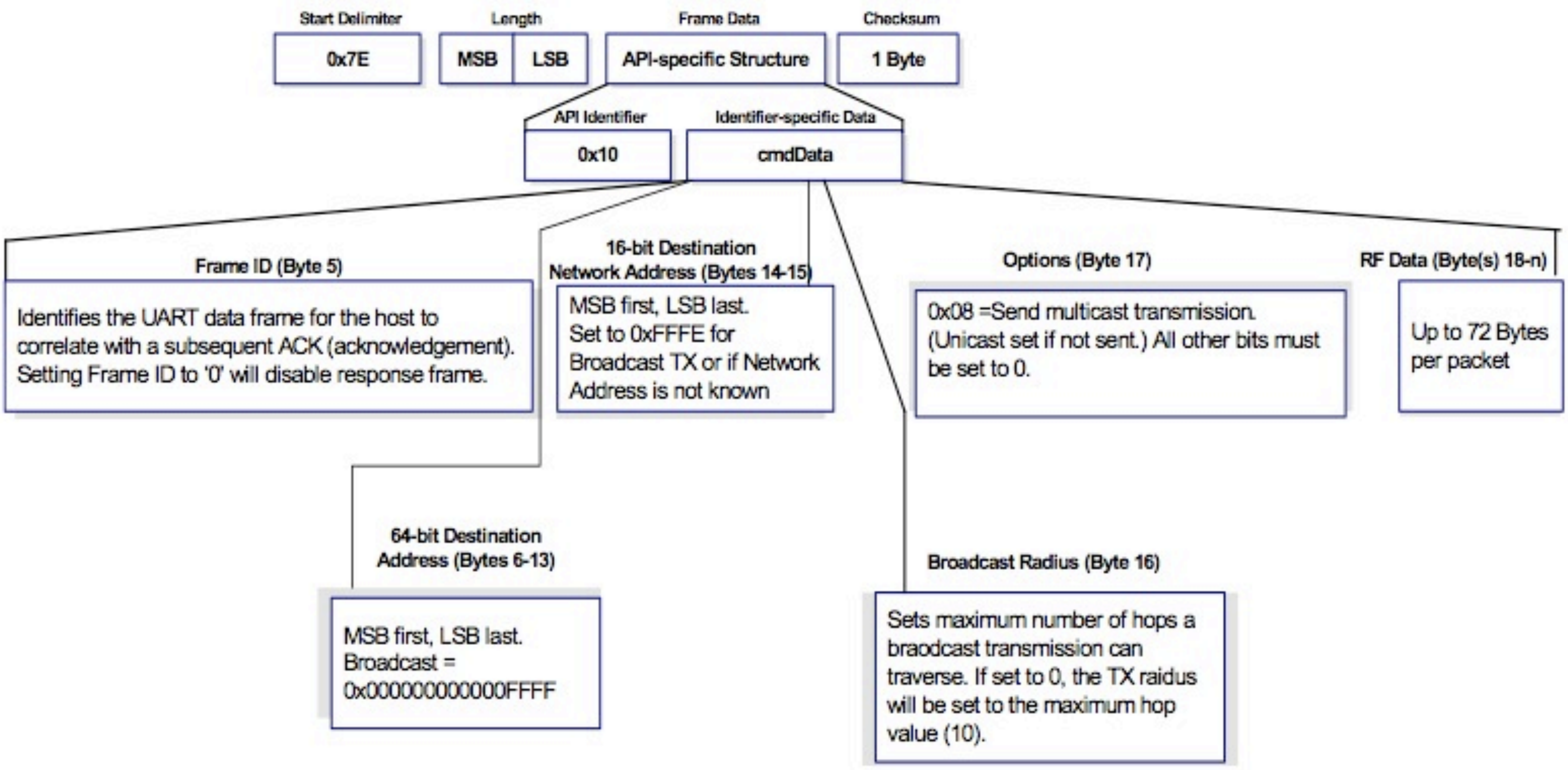
0 = OK
1 = ERROR
2 = Invalid Command
3 = Invalid Parameter

Value (Byte(s) 9-n)

The HEX (non-ASCII) value of the requested register

TX (Transmit) Request

- Remember that this is a request. Results can be checked by Frame ID



Start Delimiter: 0x7E

Length: MSB | LSB

Frame Data: API-specific Structure

Checksum: 1 Byte

API Identifier: 0x10

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.

16-bit Destination Network Address (Bytes 14-15)

MSB first, LSB last.
Set to 0xFFFE for Broadcast TX or if Network Address is not known

Options (Byte 17)

0x08 =Send multicast transmission. (Unicast set if not sent.) All other bits must be set to 0.

RF Data (Byte(s) 18-n)

Up to 72 Bytes per packet

64-bit Destination Address (Bytes 6-13)

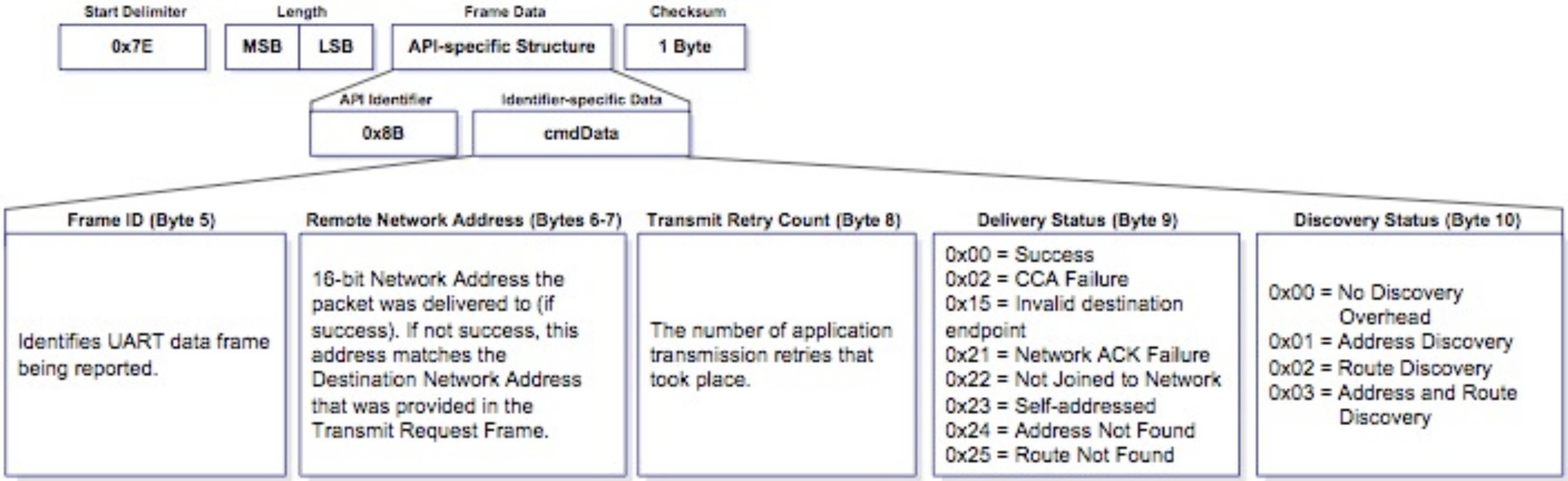
MSB first, LSB last.
Broadcast =
0x000000000000FFFF

Broadcast Radius (Byte 16)

Sets maximum number of hops a braodcast transmission can traverse. If set to 0, the TX raidus will be set to the maximum hop value (10).

TX Status (Results)

- See if your message was transmitted or not
- Use your Frame ID to see which message is being described



Frame ID (Byte 5)

Identifies UART data frame being reported.

Remote Network Address (Bytes 6-7)

16-bit Network Address the packet was delivered to (if success). If not success, this address matches the Destination Network Address that was provided in the Transmit Request Frame.

Transmit Retry Count (Byte 8)

The number of application transmission retries that took place.

Delivery Status (Byte 9)

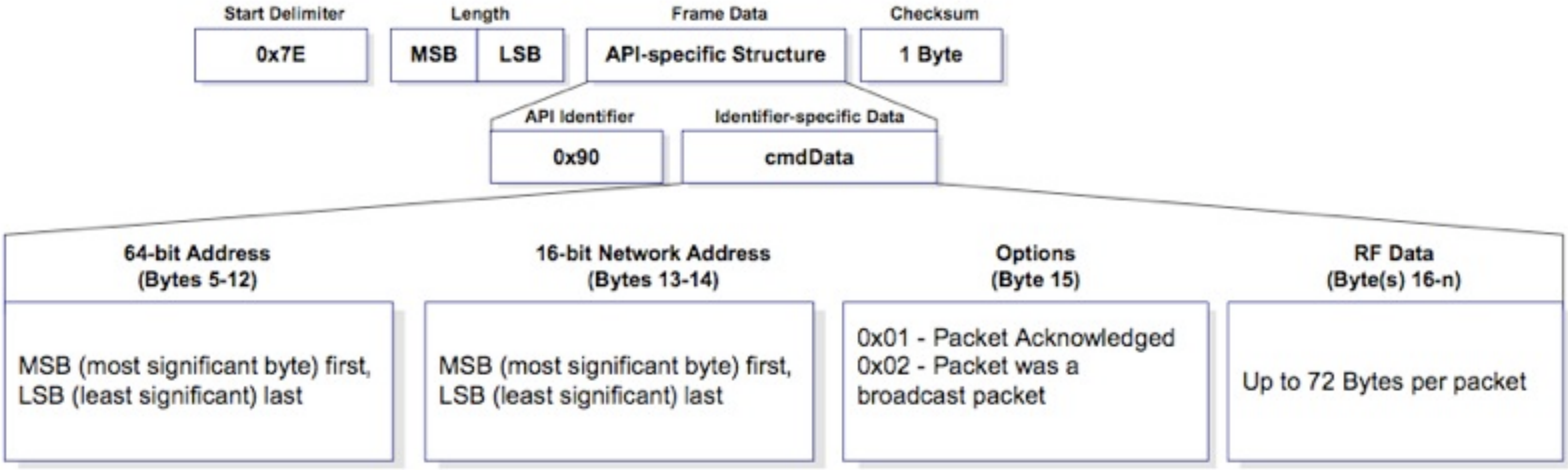
0x00 = Success
0x02 = CCA Failure
0x15 = Invalid destination endpoint
0x21 = Network ACK Failure
0x22 = Not Joined to Network
0x23 = Self-addressed
0x24 = Address Not Found
0x25 = Route Not Found

Discovery Status (Byte 10)

0x00 = No Discovery Overhead
0x01 = Address Discovery
0x02 = Route Discovery
0x03 = Address and Route Discovery

RX Packet

- Maximum of 72 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

0x90

Identifier-specific Data

cmdData

64-bit Address (Bytes 5-12)

MSB (most significant byte) first, LSB (least significant) last

16-bit Network Address (Bytes 13-14)

MSB (most significant byte) first, LSB (least significant) last

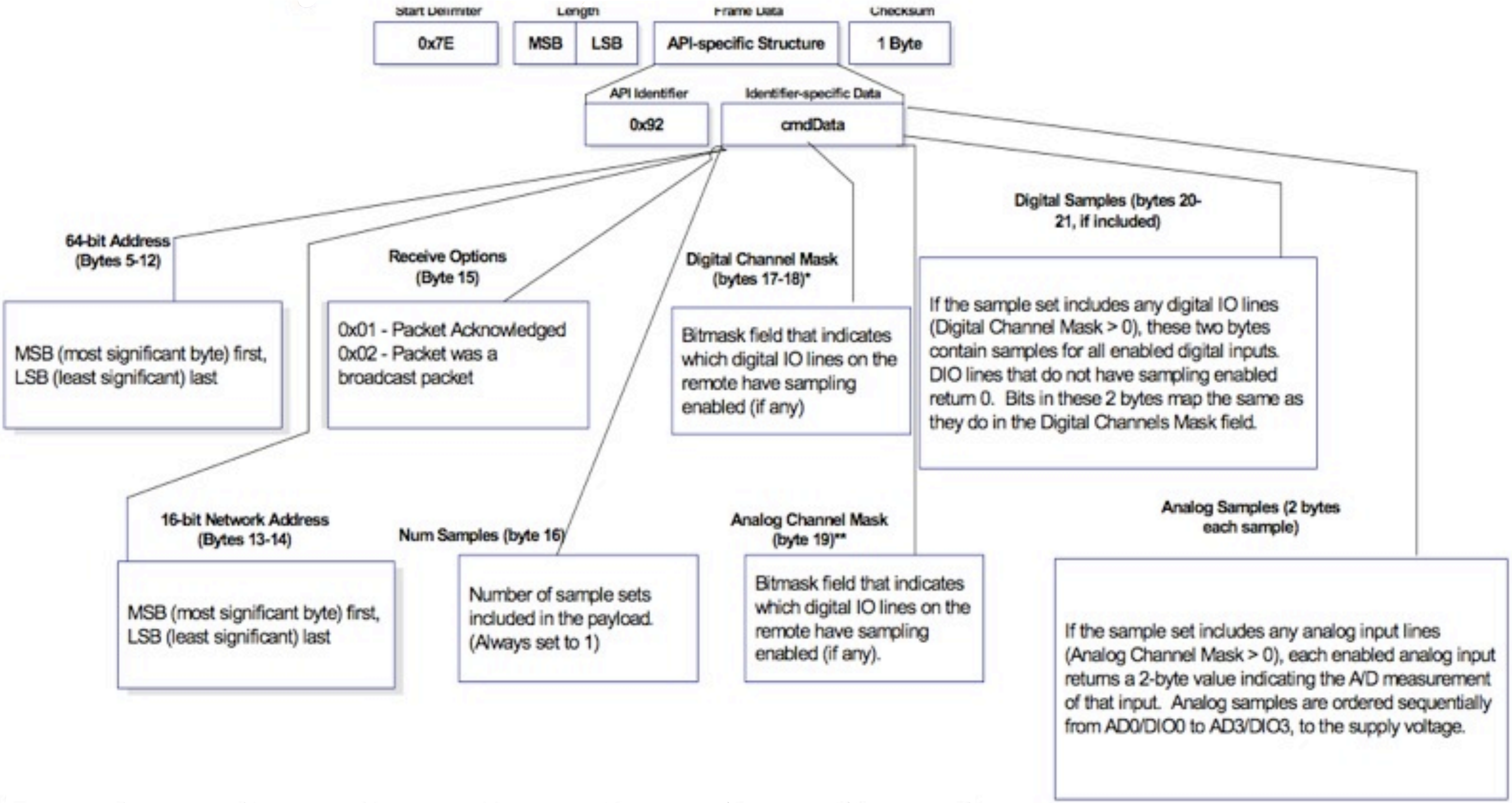
Options (Byte 15)

0x01 - Packet Acknowledged
0x02 - Packet was a broadcast packet

RF Data (Byte(s) 16-n)

Up to 72 Bytes per packet

I/O RX Packet



Start Delimiter

0x7E

Length

MSB

LSB

Frame Data

API-specific Structure

Checksum

1 Byte

API Identifier

0x92

Identifier-specific Data

cmdData

64-bit Address (Bytes 5-12)

MSB (most significant byte) first, LSB (least significant) last

Receive Options (Byte 15)

0x01 - Packet Acknowledged
0x02 - Packet was a broadcast packet

Digital Channel Mask (bytes 17-18)*

Bitmask field that indicates which digital IO lines on the remote have sampling enabled (if any)

Digital Samples (bytes 20-21, if included)

If the sample set includes any digital IO lines (Digital Channel Mask > 0), these two bytes contain samples for all enabled digital inputs. DIO lines that do not have sampling enabled return 0. Bits in these 2 bytes map the same as they do in the Digital Channels Mask field.

16-bit Network Address (Bytes 13-14)

MSB (most significant byte) first, LSB (least significant) last

Num Samples (byte 16)

Number of sample sets included in the payload. (Always set to 1)

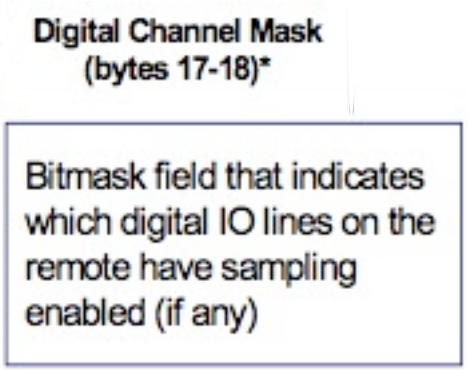
Analog Channel Mask (byte 19)**

Bitmask field that indicates which digital IO lines on the remote have sampling enabled (if any).

Analog Samples (2 bytes each sample)

If the sample set includes any analog input lines (Analog Channel Mask > 0), each enabled analog input returns a 2-byte value indicating the A/D measurement of that input. Analog samples are ordered sequentially from AD0/DIO0 to AD3/DIO3, to the supply voltage.

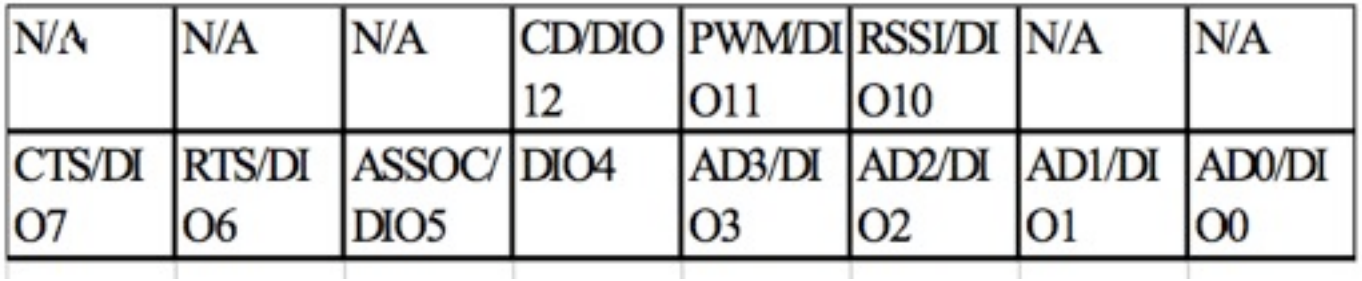
I/O Digital Channel Mask and Digital Data



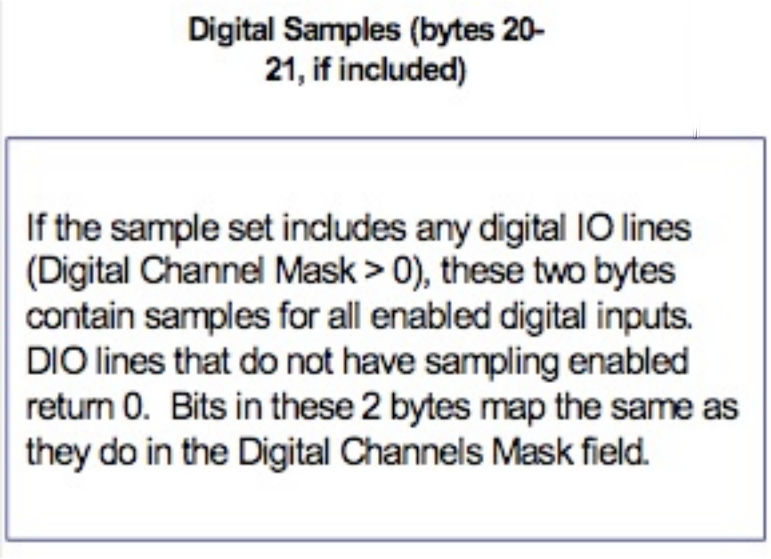
Digital Channel Mask (bytes 17-18)*

Bitmask field that indicates which digital IO lines on the remote have sampling enabled (if any)

*



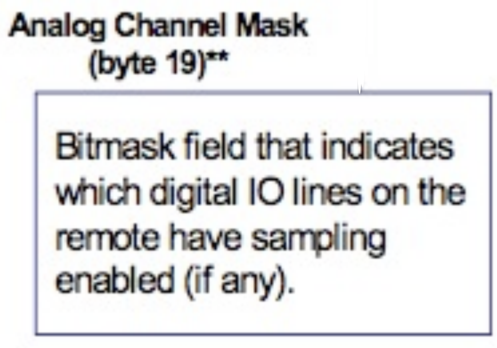
N/A	N/A	N/A	CD/DIO12	PWM/DIO11	RSSI/DIO10	N/A	N/A
CTS/DIO7	RTS/DIO6	ASSOC/DIO5	DIO4	AD3/DIO3	AD2/DIO2	AD1/DIO1	AD0/DIO0



Digital Samples (bytes 20-21, if included)

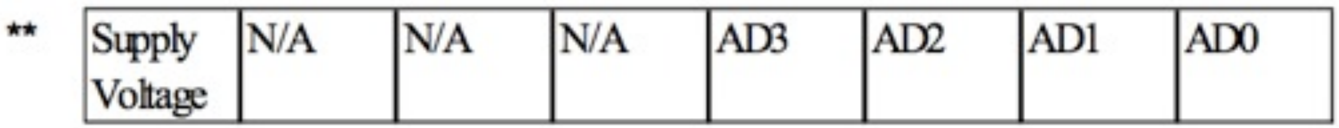
If the sample set includes any digital IO lines (Digital Channel Mask > 0), these two bytes contain samples for all enabled digital inputs. DIO lines that do not have sampling enabled return 0. Bits in these 2 bytes map the same as they do in the Digital Channels Mask field.

I/O Analog Channel Mask and Analog Samples



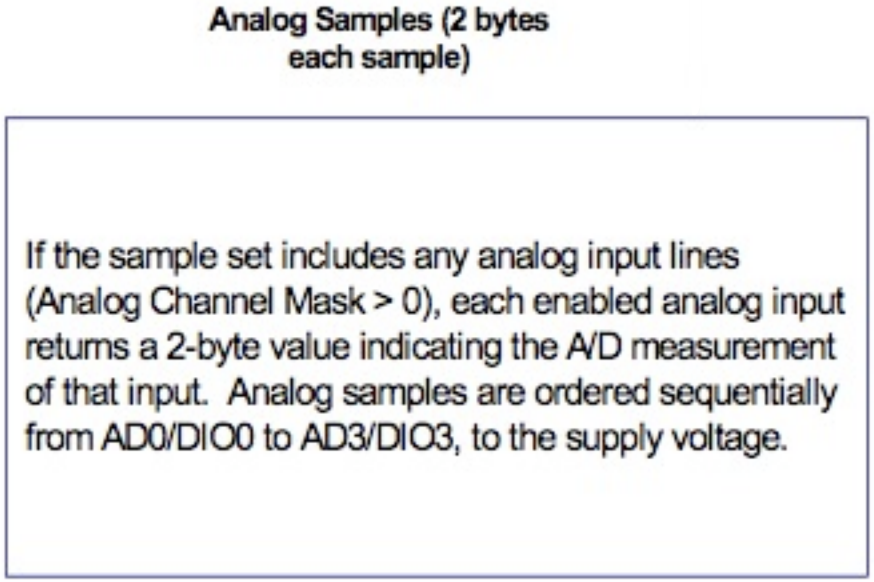
Analog Channel Mask (byte 19)**

Bitmask field that indicates which digital IO lines on the remote have sampling enabled (if any).



**

Supply Voltage	N/A	N/A	N/A	AD3	AD2	AD1	AD0



Analog Samples (2 bytes each sample)

If the sample set includes any analog input lines (Analog Channel Mask > 0), each enabled analog input returns a 2-byte value indicating the A/D measurement of that input. Analog samples are ordered sequentially from AD0/DIO0 to AD3/DIO3, to the supply voltage.

I/O Structure Reviewed

- Num Samples (1 byte)
- Digital Channel Mask (2 bytes)
- Analog Channel Mask (1 byte)
- Two bytes of digital data IF ANY DIGITAL CHANNELS ENABLED followed by...
- ...two bytes for EACH analog channel enabled...

  - Q: How many bytes ATD02 ATD12 ATD23?

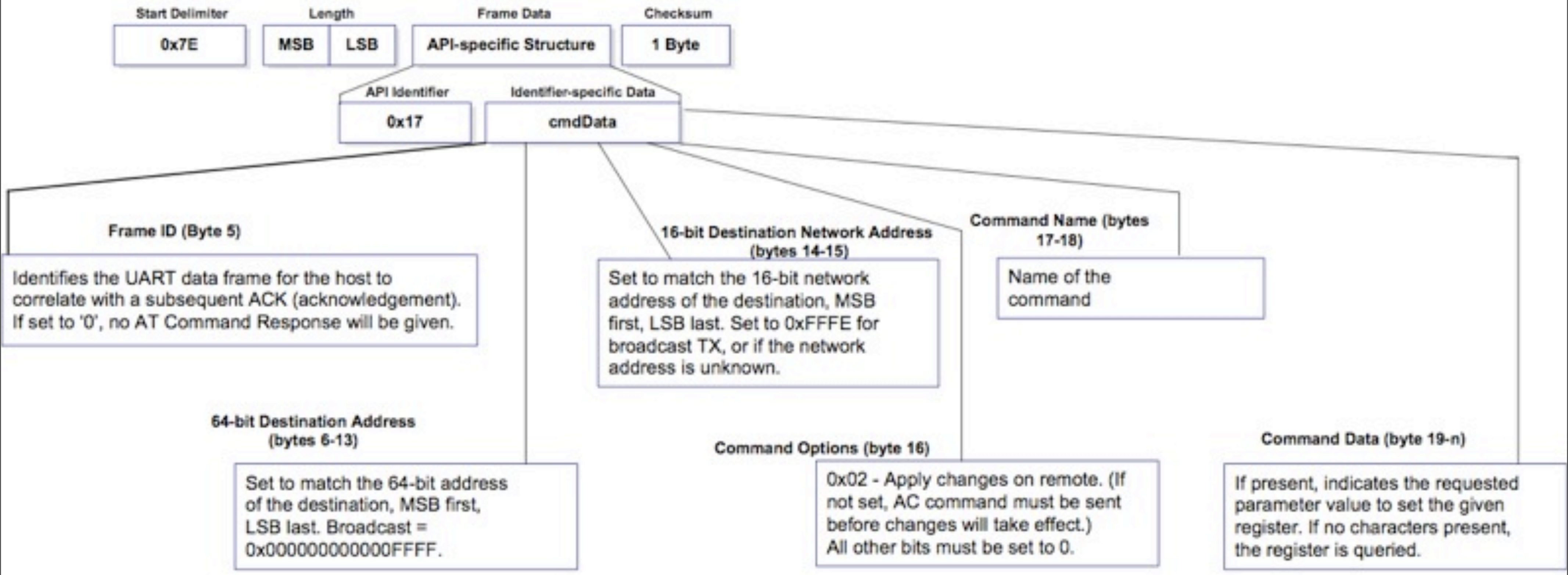
I/O Code: Basic

- Fixed parameters make for easier programming
- Assume we are just reading a single ADC channel:

```
  Arduino Version:
// make sure everything we need is in the buffer
  if (Serial.available() >= 21) {
    // look for the start byte
    if (Serial.read() == 0x7E) {
      // read the variables that we're not using out of the buffer
      for (int i = 0; i<18; i++) {
        byte discard = Serial.read();
      }
     int analogHigh = Serial.read();
     int analogLow = Serial.read();
     analogValue =  analogLow + (analogHigh * 256);
    }
  }
```

Remote AT Command Request

- Send commands across the network



Start Delimiter: 0x7E

Length: MSB | LSB

Frame Data: API-specific Structure

Checksum: 1 Byte

API Identifier: 0x17

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 AT Command Response will be given.

64-bit Destination Address (bytes 6-13)

Set to match the 64-bit address of the destination, MSB first, LSB last. Broadcast = 0x000000000000FFFF.

16-bit Destination Network Address (bytes 14-15)

Set to match the 16-bit network address of the destination, MSB first, LSB last. Set to 0xFFFE for broadcast TX, or if the network address is unknown.

Command Options (byte 16)

0x02 - Apply changes on remote. (If not set, AC command must be sent before changes will take effect.)
All other bits must be set to 0.

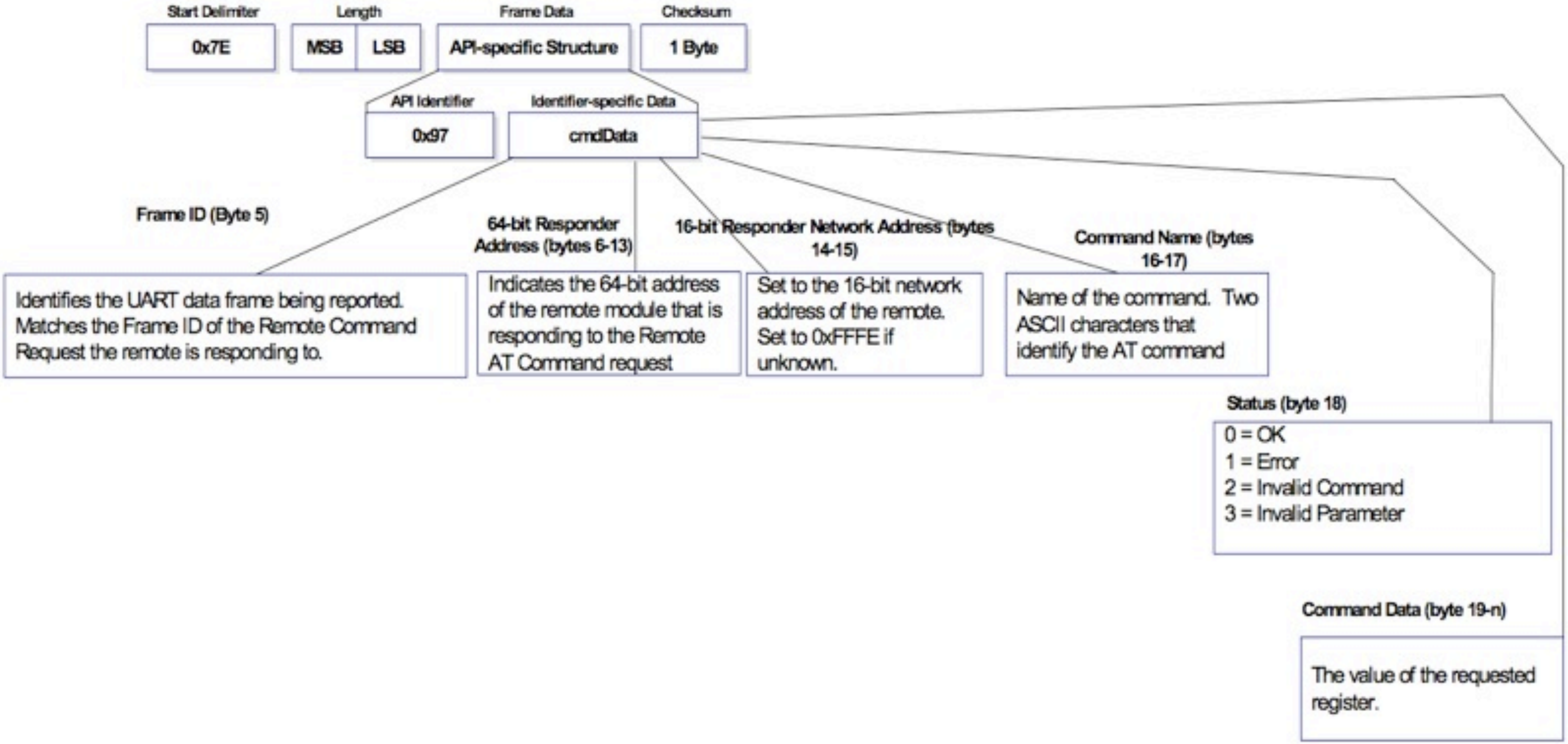
Command Name (bytes 17-18)

Name of the command

Command Data (byte 19-n)

If present, indicates the requested parameter value to set the given register. If no characters present, the register is queried.

Remote AT Command Response



Start Delimiter

0x7E

Length

MSB LSB

Frame Data

API-specific Structure

Checksum

1 Byte

API Identifier

0x97

Identifier-specific Data

cmdData

Frame ID (Byte 5)

Identifies the UART data frame being reported. Matches the Frame ID of the Remote Command Request the remote is responding to.

64-bit Responder Address (bytes 6-13)

Indicates the 64-bit address of the remote module that is responding to the Remote AT Command request

16-bit Responder Network Address (bytes 14-15)

Set to the 16-bit network address of the remote. Set to 0xFFFE if unknown.

Command Name (bytes 16-17)

Name of the command. Two ASCII characters that identify the AT command

Status (byte 18)

0 = OK
1 = Error
2 = Invalid Command
3 = Invalid Parameter

Command Data (byte 19-n)

The value of the requested register.

Readings and Assignments

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