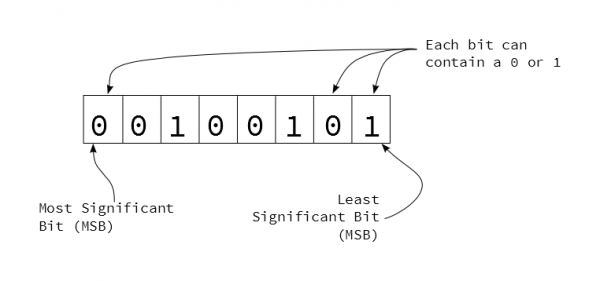
### **https://learn.sparkfun.com/tutorials/midi-tutorial/all**

### 

### **Status or Data?**

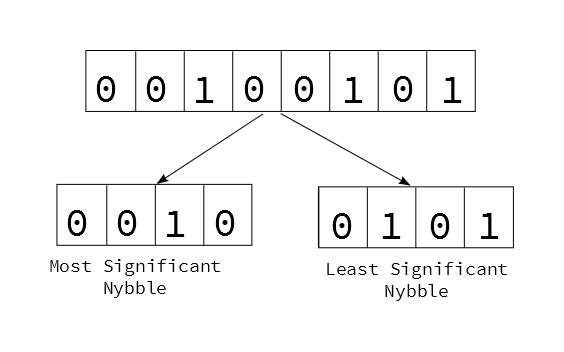
In the previous section, we discussed the UART configuration for MIDI – 31,250 bits per second, 8 data bits per byte, and 1 stop bit. MIDI uses those 8 data bits to the fullest extent!

Within a byte, each bit may be either a 0 or 1.



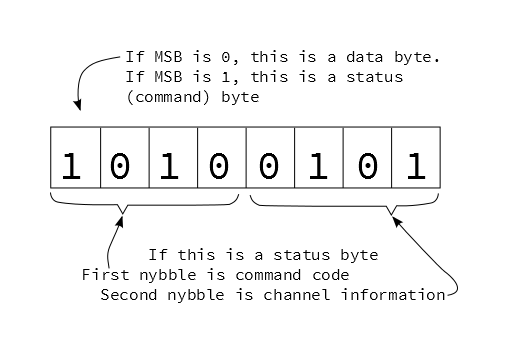
*One Eight-Bit Byte*

Some bytes are further divided into nybbles, or 4-bit chunks. Each nybble is identified by its position within the byte. When written in hexidecimal, each nybble is represented by a character. The left hand character (made up of the higher-value bits) is known as the most-significant nybble, and the right-hand character is known as the least-significant nybble.



*Byte divided into nybbles*

Bytes of MIDI messages are divided into 2 major categories, based on the setting of the most significant bit.



*Status Byte Dissection*

If the first bit is high (values between 0x80 and 0xff), it denotes a status byte. Status bytes are the commands of the MIDI stream.

* Status bytes are further subdivided by nybble. The first nybble specifies the command type, and the second nybble specifies which the channel the command applies to.
* There are only eight bit combinations with the MSB set (0x8 to 0xf), therefore there are eight possible status commands.
* The 4 bits in the channel nybble are all usable, giving MIDI 16 possible channels.

If the first bit is low (values between 0x00 and 0x7f), it is a data byte, indicating parameters that correspond to a previous status byte. Because the MSB must be zero (otherwise they’d become status bytes), the data is limited to 7-bits, or the range from 0 to 127 (0x0 to 0x7f).

### **Statuses & Corresponding Data**

Let’s look at the status bytes. We’ll start with a list, then explore each in the following sections.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MIDI Status Messages | | | | | |
| Message Type | MS Nybble | LS Nybble | Number of  Data Bytes | Data Byte 1 | Data Byte 2 |
| Note Off | 0x8 | Channel | 2 | Note Number | Velocity |
| Note On | 0x9 | Channel | 2 | Note Number | Velocity |
| Polyphonic Pressure | 0xA | Channel | 2 | Note Number | Pressure |
| Control Change | 0xB | Channel | 2 | Controller Number | Value |
| Program Change | 0xC | Channel | 1 | Program Number | -none- |
| Channel Pressure | 0xD | Channel | 1 | Pressure | -none- |
| Pitch Bend | 0xE | Channel | 2 | Bend LSB  (7-bits) | Bend MSB  (7-bits) |
| System | 0xF | further specification | variable | variable | variable |

The messages with the channel number in the second nybble of the status byte are known as channel messages. Channels are often used to separate individual instruments – channel one could be a piano, two a bass, and so on. This allows a single MIDI connection to carry information for multiple destinations simultaneously. Each sound would be played by sending messages with the apprppriate value in the channel nybble.

Channel numbering leads to some confusion. With 4 bits, there are 16 possible binary values, 0 through 15 (0x0 through 0xF). Since most people start counting at one rather than zero, MIDI devices commonly (but not always) internally add an offset of one to the binary value, resulting in the range 1 to 16. If you're having trouble getting a system to communicate, you might try adjusting channels up or down by one.

Since they’re very useful, and easy to implement, we’re going to start with two of the most common types of messages: Note On/Off and System Realtime.

#### **Note Off (0x80), Note On (0x90)**

The meaning of Note On and Off messages is reasonably obvious. When a key on a keyboard is pressed, it sends a Note On message, and it sends a Note Off when the key is released. On and Off messages are also sent by other types of controllers, such as drum pads, and MIDI wind instruments.

When a synthesizer receives a Note On, it starts generating sound; the Note Off instructs it to stop.

Note On and Off are each comprised of three bytes