Command Formatting & Syntax

Build MIDI Messages One Byte at a Time

Most MIDI messages are sequences of 2 or 3 8-bit bytes. The largest hurdle when integrating the DecaBox with Crestron / AMX / Control4 equipment is properly formatting these messages. An excellent, detailed description of their syntax can be found here:

- https://www.midi.org/specifications/item/table-1-summary-of-midi-message
- https://www.midi.org/specifications-old/item/table-3-control-change-messages-data-bytes-2

> In our documentation, we use this format to describe raw byte data: `$AA $BB $CC` . The $ sign signifies hexadecimal values, which means that AA, BB, etc are in the range [00 FF] hex, which maps to [0 255] decimal. Spaces are inserted between bytes in this documentation merely for convenience in reading.

MIDI messages are zero based and *generally* bytes 2 and 3 have a maximum value of $7F, or 127 decimal. A simple online decimal <-> hex converter can be found at

https://www.binaryhexconverter.com/decimal-to-hex-converter

Here are sample MIDI messages:

```
$90 $01 $7F   Note on, MIDI channel 1, note #2, full velocity $8F $07 $40   Note off, MIDI
channel 16, note #8, 50% velocity
$B1 $03 $10   MIDI CC #4 (foot controller), ~30% intensity, MIDI channel 2
```

(The MIDI channel is the lower nibble of the first byte. That is, $BX means MIDI CC message and X can vary between [$0 $F]. Hex $0 means MIDI channel 1, $1 means MIDI channel 2, etc. $F is channel 16.)

> It’s vital to note that these MIDI messages must be transmitted as ***raw hex bytes*** rather than a group of ASCII characters. Occasionally we entertain tech support calls where we learn that a control system is sending out a nine-byte *string* instead. Obviously, this won't generate the expected results.

> "$" + "9" + "0" + "$" + "0" + "1" + "$" + "7" + "F" **is *not* the same** as $90 $01 $7F.

On most control platforms, it's necessary to 'escape' these byte values so that they are transmitted as raw data, rather than as a string of ASCII characters. For example, Crestron uses the characters **\x** to signify a single byte:

```
\xAA\xBB\xCC
\x90\x01\x7F <--- Example 1
\x8F\x07\x40 <--- Example 2
\xB1\x03\x14 <--- Example 3
```

MIDI SYSEX messages can be nearly any length but are framed by the bytes $F0 and $F7. An example message might look like this in Crestron format:

```
\xF0\x01\x02\x03\x04\x05\x06\xF7
```

> One way to quickly troubleshoot this system is to connect a MIDI cable from the DecaBox's 'MIDI Out' to 'MIDI In' connectors. This gives loopback on the serial side and makes it easy to confirm that the correct messages are being sent. We've had more than a few support calls where what a control system claimed to be sending, and what was actually going out the door, were wildly different. At least one of those calls uncovered serious system-level bugs, which led to a massive firmware revision by a well-known equipment manufacturer who shall not be named.

Revision #10
Created Tue, Oct 29, 2019 6:56 PM by ESINC
Updated Thu, Apr 16, 2020 1:48 PM by ESINC