DecaBox RS-232 MIDI Bidirectional Conversion

Dead simple interfacing between control systems and MIDI gear.

- Overview
- Connections
- Command Formatting & Syntax
- Purchasing

Overview

This useful firmware revision was first commissioned by the production crew of a long-running (nearly 30 years) game show televised in the USA. One of the show's main set pieces was controlled by custom software running under DOS. However, the DOS system needed to be updated, as spare parts were no longer available.

It has since been used worldwide, often by integrators who want to recall presets on audio consoles via MIDI, but automated through a Crestron / AMX / Control / Savant platform.

Operation is very simple: each byte received by the RS-232 port is instantly transmitted via the MIDI

out connector. And every byte received by the MIDI in jack is transferred to the RS-232 port.

The system is completely transparent to all data flowing through it. All that changes is the baud rate (38,400 <–> 31,250) and the signal type (true RS-232 voltages vs MIDI's current loop).

Full duplex communication is easily possible if required.

Connections

Required connections are:

- DC 9V, 300 mA, center positive. This adapter is supplied with the DecaBox.
- MIDI In & Out on standard DIN-5 circular jacks
- RS-232 cable, DB9-M on at least one end. RS-232 baud rate is 38,400 8N1 (8 data bits, no parity, 1 stop bit).

The DecaBox has three LEDs, one on each of the user interface pushbuttons. For this firmware build, the left LED flickers when data passes (in either direction) across the RS-232 port. The right LED likewise flashes in the presence of MIDI data, either in or out. In general, these two LEDs will appear to illuminate simultaneously. However, in our firmware they are controlled by a parallel set of functions.

In this firmware build, the three pushbuttons on the front panel have no specific function.

Buffering: MIDI vs RS-232

MIDI data is sent and received at 31,250 bits per second. On the serial side, our baud rate is 38,400, which is approximately a 20% difference. For this reason, MIDI data may be send to RS-232 equipment *at line speed.* DecaBox can process incoming MIDI transparently and without any delay. However, the reverse is not true. DecaBox contains an internal 128 byte buffer which stores incoming serial data and queues it for MIDI transmission. For short messages containing only tens of bytes, the system will send out the MIDI data as quickly as possible. Large, tightly spaced blocks of serial data may cause internal buffer overruns and data loss. For this reason, it's best to transmit a few tens of bytes at a time, then pause to allow the queue to clear.

For reference, a three-byte MIDI message such as 'note on' or 'note off' takes ~ 1 mS to transmit on the MIDI port.

The RS-232 connection on the DecaBox is identical to that of a standard PC or USB / serial adapter:

- Pub 2 Data Receive
- Pin 3 Data Transmit
- Pin 5 Ground

If the DecaBox is connected to a PC for testing or troubleshooting, a null modem cable is *required.* Pins 2 and 3 must swap to allow proper communication between two devices with identical serial pinouts.

> Pinouts and connections to other equipment may vary; consult user manuals for best results.

> The DecaBox USB port is used for firmware updates *only.* It neither transmits nor receives MIDI or serial data.

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.

Purchasing

If you've landed here via a web search, don't already own one of these systems, but would like to add one to your collection, our online store is here:

www.response-box.com/gear/shop

And our main site is here, which includes links to distributors, etc:

www.response-box.com/gear