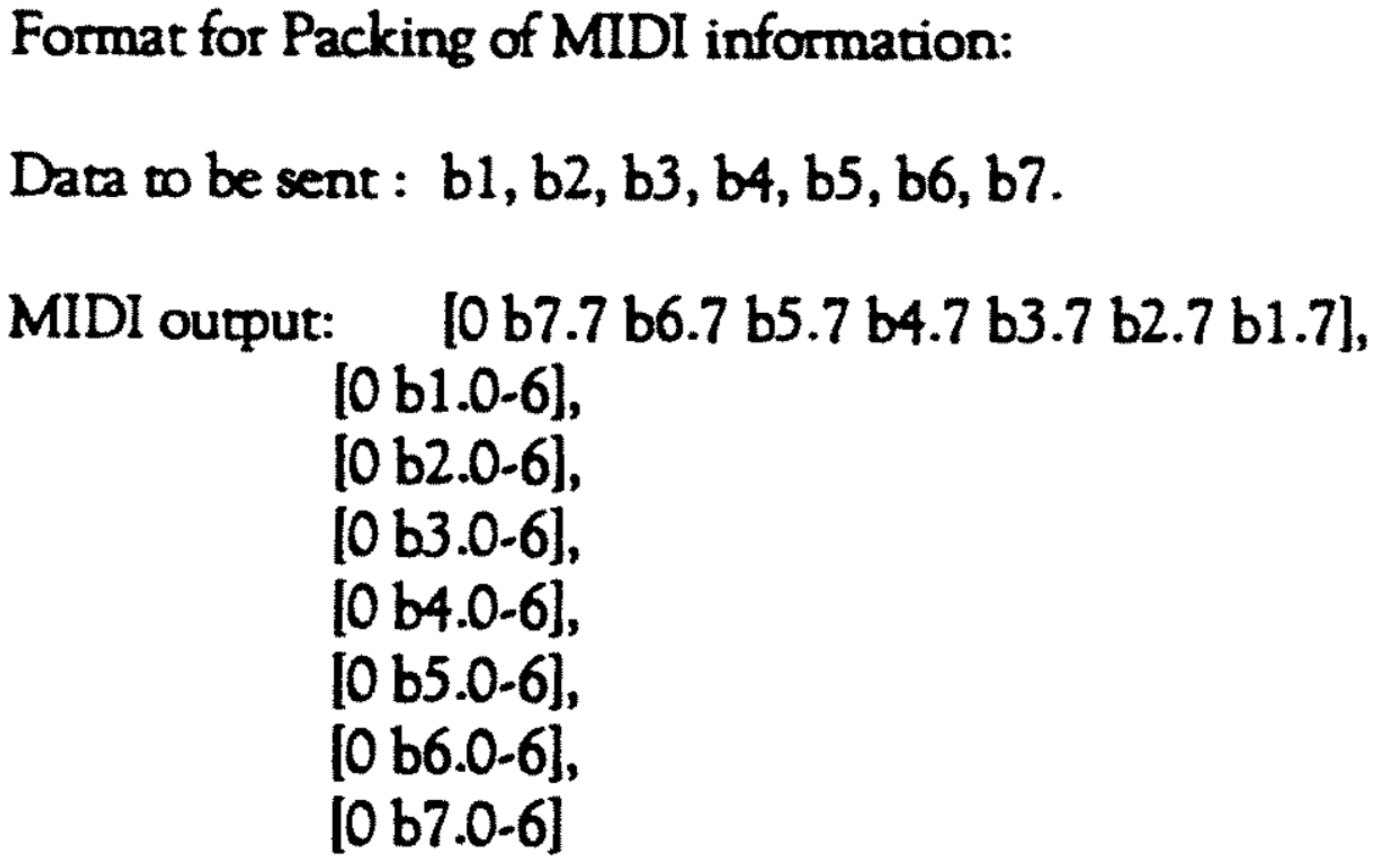
An interesting way to pack bytes into a MIDI SysEx message

December 21, 2021

Here’s a little thought experiment: how would you transmit 8-bit bytes via messages that let you encode only 7 bits at a time?

The engineers at DigiTech handled this exact problem on their DHP-55 effects unit in the 1990s. They needed to transmit 8-bit bytes to the unit via a MIDI SysEx message. Every byte in a SysEx message must have its most significant bit set to 0, so there are only 7 bits available per byte for sending data.

This is how they describe their approach in the DHP-55 manual:



Format for Packing of MIDI information:

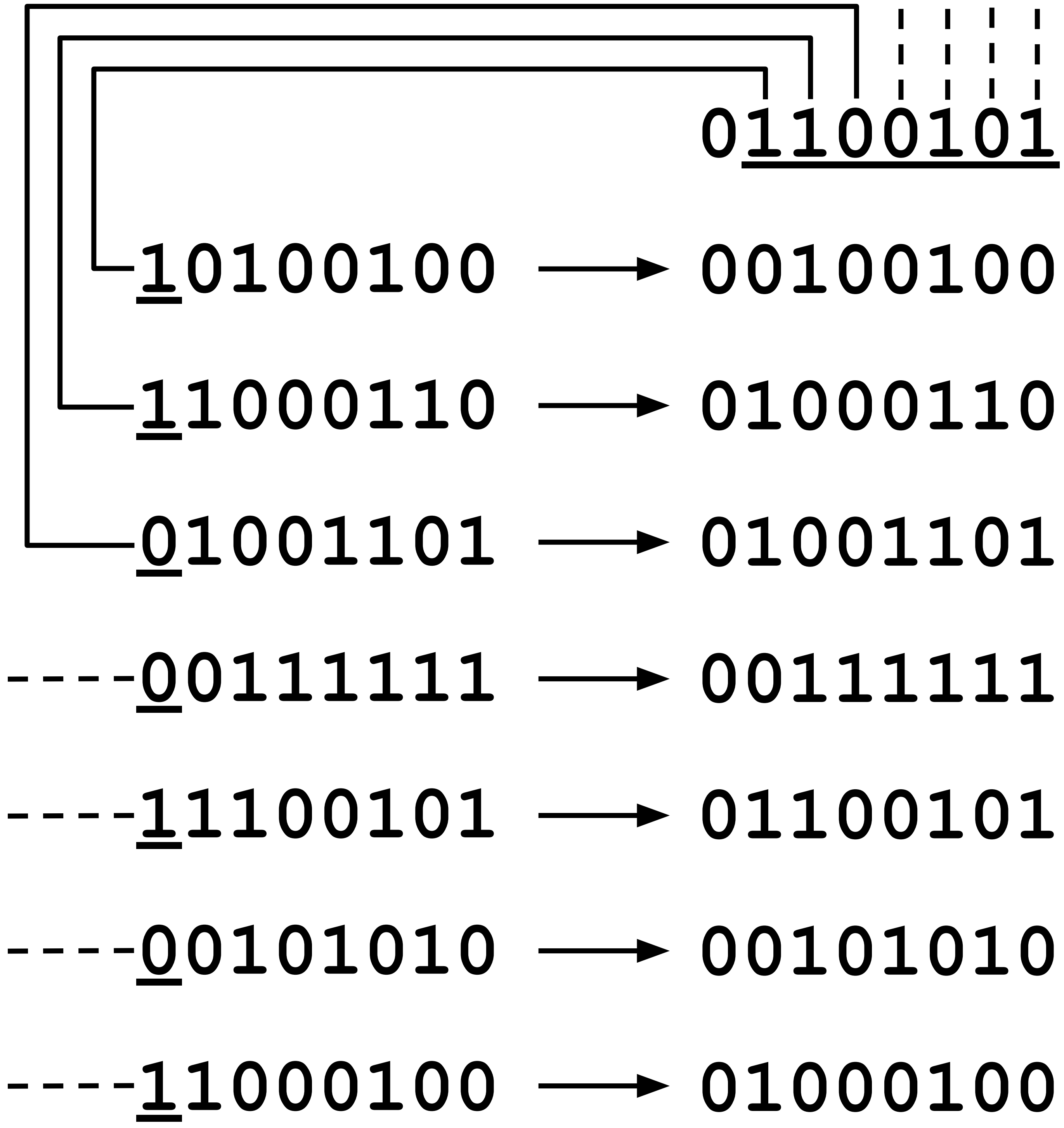
Data to be sent : b1, b2, b3, b4, b5, b6, b7.

```
MIDI output:     [0 b7.7 b6.7 b5.7 b4.7 b3.7 b2.7 b1.7],
             [0 b1.0-6],
             [0 b2.0-6],
             [0 b3.0-6],
             [0 b4.0-6],
             [0 b5.0-6],
             [0 b6.0-6],
             [0 b7.0-6]
```

(For clarity, I'm going to call the 8-bit bytes we want to transmit the "real" bytes and the bytes with 7 bits available that we have to use the "send" bytes.)

The engineers chose to do the following: pack 7 "real" bytes into 8 "send" bytes at a time. Take the most significant bit of each of the "real" bytes and put them into the first "send" byte. Then take the remaining 7 bits of each of the "real" bytes and put them in each of the remaining "send" bytes.

I've illustrated an example of this packing below:



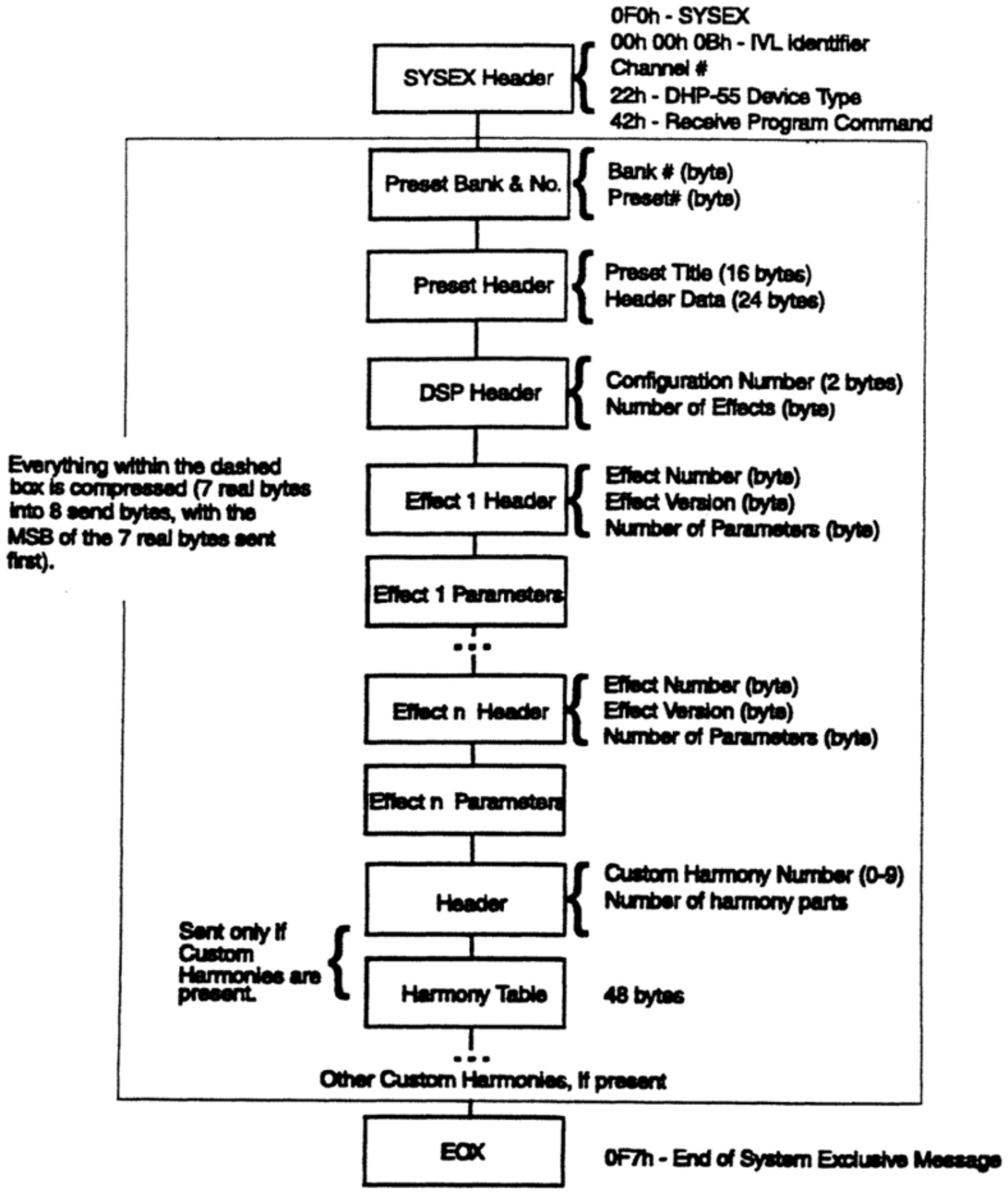
- **Left**: The 7 (8-bit) "real" bytes we want to transmit
- **Right**: The 8 "send" bytes. The first byte contains 0 and the most significant bit of each of the 7 "real" bytes. The following 7 "send" bytes contain 0 and the 7 remaining bits of each of the 7 "real" bytes.

I find this a pretty interesting solution to the problem. It's not my first instinct – I would have just mashed the bytes together instead of sending the most significant bits separately.

So why the engineers at DigiTech separate the most significant bits? One reason might be that it makes hex dumps a lot easier to read.

If you take a look at the manual, you can see that much of the data that's being transmitted is small integers (less than 128) where the most significant bit is already 0. This means that many of the "send" bytes will have identical values to those in the "real" bytes.

Bank number, preset number, number of effects, effect number, number of parameters, effect version – all of these will likely be less than 128.


0F0h - SYSEX
00h 00h 0Bh - IVL identifier
Channel #
22h - DHP-55 Device Type
42h - Receive Program Command
SYSEX Header
Preset Bank & No.
Bank # (byte)
Preset# (byte)
Preset Header
Preset Title (16 bytes)
Header Data (24 bytes)
DSP Header
Configuration Number (2 bytes)
Number of Effects (byte)
Everything within the dashed box is compressed (7 real bytes into 8 send bytes, with the MSB of the 7 real bytes sent first).
Effect 1 Header
Effect Number (byte)
Effect Version (byte)
Number of Parameters (byte)
Effect 1 Parameters
Effect n Header
Effect Number (byte)
Effect Version (byte)
Number of Parameters (byte)
Effect n Parameters
Header
Custom Harmony Number (0-9)
Number of harmony parts
Sent only if Custom Harmonies are present.
Harmony Table
48 bytes
Other Custom Harmonies, if present
EOX
0F7h - End of System Exclusive Message

If the DigiTech engineers had gone with my approach, "send" bytes would not resemble "real" bytes, and they wouldn't be able to use hex dumps for spot checks.

If you know another reason, please reach out!