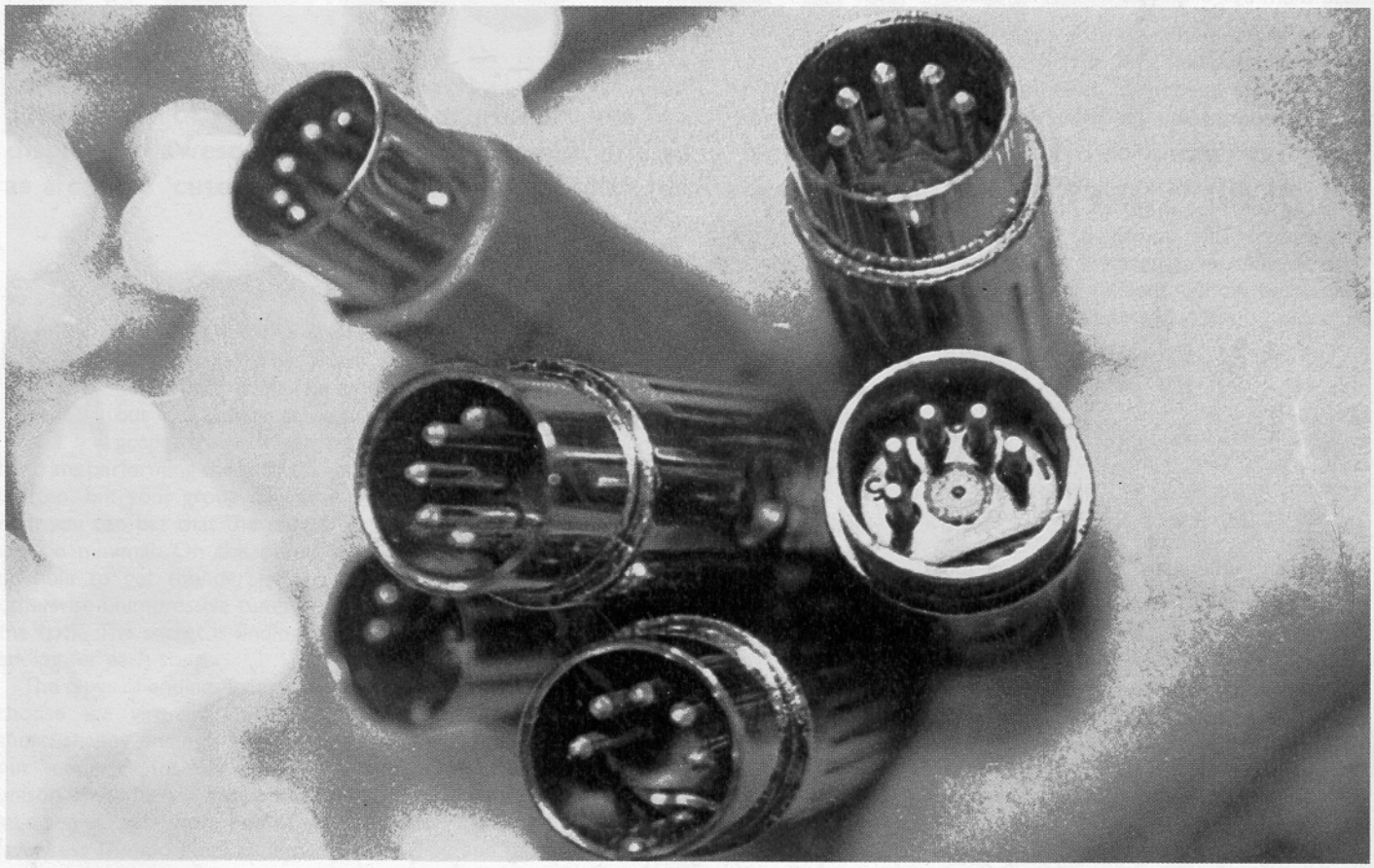
MIDI *matters*



Photography *Alan Beeson*

Just when you thought we'd exhausted all of the possibilities for MIDI messages, we found some more. Yeah, life's tough but communication is important, so listen up!

Text by Norman Weinberg.

OKAY, SO YOU'RE convinced. Tomorrow you're off to the local music store to get MIDIfied. The problem is that you don't have all the money in the world, but still want to get a hip system. If you're looking for a sound generator that will give you flexibility, a wide variety of sounds, and few headaches, then 'multitimbral' is the way to go.

Timbre is a musical word which means sound color, and multitimbral is a term which is used to describe a synth's ability to play more than one timbre at a time. All drum machines can be thought of as multitimbral because they are capable of playing several different sounds at the same time. When you are playing a bass drum, snare drum, and high hat, along with a few toms, you are multitimbral. Some keyboard synths can be multitimbral by using two different techniques. One might be to use a 'split keyboard', where you could have a bass guitar sound on the lower octaves and a flute sound on the upper keys, for example – although any two timbres can be combined. The second method of getting different sounds at one time involves a special use of MIDI Mode Four.

Of course you remember from earlier articles (you *do* remember, don't you?) that Mode Four is Omni Off/Mono. In other words, the synth will be listening to a single MIDI channel and responding in a monophonic manner (one note at a time). However, an instrument in Mode Four is not limited to listening on just one MIDI channel. It *can* respond monophonically to several different MIDI channels, and of course each MIDI channel would be used to access a different timbre to make it – you guessed it – multitimbral. Not only that, but there is an extension to Mode Four, called 'Multimode', which enables an instrument to respond polyphonically (with different timbres) to several different MIDI channels at once, thus making it multitimbral.

As an example, let's look at one of the least expensive MIDI synthesizers on the market, the Casio CZ101. Sold by mail order for under $300, it has an impressive array of features. While in Mode Four, it can read four different MIDI channels at one time with each channel having its own different timbre. Each MIDI channel is monophonic, but you can play four timbres at the same time. This mode makes the synth act as if it were four individual monophonic instruments. MIDI channel 3 could be playing an electric bass sound while channel 4 plays a woodblock, channel 5 could play an organ sound, and channel 6 might be an explosion effect. Use your trigger to MIDI interface so that you send the pads to different MIDI channels and it's all under control. With the CZ101, you must keep one thing in mind when dealing with this special use of Mode Four: the MIDI channel numbers that you're using must be in consecutive order. You can't have it read channels 4,5,6, and 10 for example.

E-mu Systems' E-Max sampler is a good example of Multimode – only E-mu calls it 'Super Mode'. While in Super Mode, the E-Max will read any or all sixteen different MIDI channels and assign a different preset

(set of timbres) to each one. With this instrument, each channel can also respond in a polyphonic manner! The Ensoniq ESQ1, the Yamaha TX81Z, and the Roland MT32 all have similar abilities.

Channel Mode Messages

WHILE WE ARE on the subject of Mode Four, it might be a good time to tell you about the other Channel Mode Messages that are defined in the MIDI specification. Last time, we discussed all the Channel Voice Messages including Control Change commands. The Channel Mode Messages are a subset of the Control Change message. If you remember, the Control Change status byte was 1011 followed by the channel number. The first data byte that follows this status command is going to determine which of the Channel Mode Messages are being sent. The second data byte either presents more information or is a 'dummy' byte.

Local Control is specified by the data byte of 122 (there are no controllers numbered from 122 to 127 because these controller numbers are reserved for the mode messages). Local control is aptly named, because if local control is turned off, then the microprocessor inside the MIDI unit will ignore messages coming from its own keyboard or pads. This is pretty worthless on a drum machine or a sound generator, but it may come in handy if you own a keyboard synthesizer and you're using it to program a drum machine. With local control off, the synth will continue to send MIDI messages, but will not make any sounds of its own. With local control turned on, you would be able to hear both the drum machine and the synth sounds at the same time.

Perhaps one of the most valuable mode messages is 123, All Notes Off. If you've ever had a problem with stuck notes, then you can appreciate this message. When a synth receives an all notes off command, it will stop playing any notes that are currently sounding. If you use a sequencer, whenever you hit the 'stop' button, the sequencer will typically send one of these commands. However, please note that not all MIDI sound sources will respond to an All Notes Off message.

The next four mode messages determine the MIDI modes themselves. If the first data byte is 124 (binary code of 0111 1100), then the receiving unit will change to Omni Off. A data byte of 125 (0111 1101) indicates Omni On, 126 (0111 1110) is Mono On, and a data byte of 127 (0111 1111) is Poly On. Remember, that Omni is either on or off, but Poly On and Mono On are mutually exclusive (turning one on turns the other off). Example Number 1 shows all of the different Channel Mode Messages that are defined by the MIDI specification.

Example No. 1

Channel Mode Messages

Status Byte	First Data Byte	Second Data Byte
Program Change 1100 nnnn	Local Conrol 0111 1010	On / Off Value
Program Change 1100 nnnn	All Notes Off 0111 1011	Dummy Byte 0000 0000
Program Change 1100 nnnn	Omni Off 0111 1100	Dummy Byte 0000 0000
Program Change 1100 nnnn	Omni On 0111 1101	Dummy Byte 0000 0000
Program Change 1100 nnnn	Mono On 0111 1110	Number of Mono Channels
Program Change 1100 nnnn	Poly On 0111 1111	Dummy Byte 0000 0000

"nnnn" represents the four bits that determine the MIDI channel. Dummy bytes are required so that each complete MIDI mode message is three bytes long.

System Messages

IN ADDITION TO messages that are sent over specific MIDI channels, there is another classification of messages that are intended for all MIDI devices connected in the system. They are called System Messages and are received by every piece of gear no matter what MIDI channel they are listening to. Because these commands are not channel specific, they all begin with the binary digits of 1111. The next four bits in the status byte determine the actual system message type.

The first class that we will discuss is called System Common Messages. There are five different system common messages currently defined by the MIDI specification.

The Quarter Frame Message (1111 0001) is used only when devices are running under the newly defined MIDI Time Code (this is pretty advanced stuff, so hang on until next month when we discuss different types of timing and synchronization systems). Unless you have a computer and deal with syncing music and sound effects to video, you really don't have much use for this message.

More important to drummers is the Song Position Pointer (1111 0010). Song position pointers are now used in almost every model and brand of drum machine. They perform a simple task which is a gigantic time saver. They locate the exact point within a song down to each individual sixteenth note in over a thousand different measures.

To see how song position pointers are used, imagine that you are programming an eighty measure song, which uses two MIDI drum machines synced together. Without song position pointers, if you want to change something during the last four measures, you must start both machines at the beginning of the song. You have no choice, but to sit back and have a cup of coffee until the last few measures come up. With song position pointers, one MIDI drum machine can tell the other exactly where to begin anywhere in the song, thus saving you many hours of hard work and, perhaps, caffeine poisoning. Just like the pitch wheel change messages, a song position pointer makes use of high resolution data by combining the two data bytes to indicate over 16,000 different positions.

Song Select is another important system common message for drummers. If you often use a computer to control your drum machine, then this message will tell the drum machine to call up a particular song and get it ready to play. Song select messages can indicate any of 128 different songs.

Tune Request (1111 0110) was primarily designed for analog synthesizers. Whenever a device receives this command, it will tune itself to some sort of internal reference (most often A=440 cps). Now that almost everything on the market is digital, and instruments don't drift in and out of tune as much as they used to, this message is all but obsolete. The last system common message is called the End of Exclusive and is discussed below. See Example Number 2 for a chart of all the different System Common Messages.

Example No. 2

System Common Messages

Status Byte	First Data Byte	Second Data Byte
Quarter Frame 1111 0001	Message Type Time Frame	None
Song Position Pointer 1111 0010	LSB Value	MSB Value
Song Select 1111 0011	Song Number	None
Tune Request 1111 0110	None	None
EOX 1111 0111	None	None

System Exclusive Messages

ORIGINALLY, SYSTEM EXCLUSIVE messages were designed to allow a musician access to certain controls within a synthesizer that might not be defined by the MIDI specification. Primarily, system exclusive messages deal with non-performance aspects of music, such as programming the sounds that the synth will perform. As an example, consider a synthesizer that has an envelope generator controlling its amplitude. The MIDI specification says nothing about individual envelope settings. Even though the envelope generator is a type of controller, it is not defined as a controller by MIDI. Why not, you ask? Because different synths have different types of envelope generators and while some may have only four stages called attack, decay, sustain, and release, others may have eight or more. Manufacturers and musicians didn't want to standardize the number of stages an envelope should have. If every aspect and feature of a synthesizer was made standard, all instruments would sound the same! With system exclusive messages, each instrument's individual abilities can still be accessed through MIDI.

System exclusive messages always begin

the same way. They start with a command of IIII 0000 as the status byte. After this status byte comes a manufacturer's ID number. The purpose of the status byte is to tell all MIDI devices connected in the system to get ready for some system exclusive information. The manufacturer's ID identifies a particular brand of device, and maybe even a particular model within a company's product line. Manufacturer ID numbers are assigned by the MMA (MIDI Manufacturers Association) and can be requested by any company that builds MIDI devices. Some examples are: 27 for equipment made by Baldwin, 71 for gear by Akai, 67 for Yamaha equipment, and 68 for Casio. Once this ID number has been sent, only those MIDI devices that recognize their own number will continue listening to the rest of the message. Any other brand or model of instrument in the system will ignore all the messages that follow.

Next in the system exclusive format comes the data itself. This data stream can be anything that the manufacturer wants it to be. It can include everything from the various envelope settings to the tuning of a bass drum sound. The data stream can be very short or extremely long, depending on the amount and type of information that is being communicated. After the data is transmitted, system exclusive messages all end in the same way, with the system command message mentioned earlier called End of Exclusive (often abbreviated as EOX). The EOX byte is IIII 0III and tells all devices connected to the system that they should start paying attention again.

How can you use system exclusive commands with your MIDI setup? Have you ever seen a visual editor for a synthesizer? Visual editors use the power and large screen of a computer to program the synthesizer by 'remote control'. Visual editors let the user see all the different programmable values at the same time. Instead of programming the synth from the front panel and pressing nine zillion different buttons to change a particular parameter, the computer can do it for you. When a value is changed on the computer's screen, the proper system exclusive commands are sent over MIDI by the computer, and the new values are received by the synth. There are visual editors for most popular samplers, synthesizers, and tone generators. While there are very few visual editors for drum machines, let's take a look at how a computer programmer might use system exclusive commands to write an editing program for the Yamaha RXII drum machine.

Example Number 3 shows the binary codes that would change the 'Total Volume' control of the RXII (listening to MIDI channel one) to the level of 52. The entire message starts off with the system

Example No. 3

RX11 System Exclusive Messages

MIDI Number	Binary Code	Description
241	1111 0000	System Exclusive Status Byte
67	0100 0011	Yamaha Manufacturer's I.D. Number
16	0001 0000	Sub Status and MIDI Channel Number
3	0000 0011	Parameter Group and Sub Group Number
115	0111 0011	Parameter Number
52	0011 0100	Data Value of 52
248	1111 0111	EOX Message

exclusive status byte. Next is the data byte that says "this message is intended for Yamaha gear only" *(or "this Bud's for you . . ." sorry Norman, I couldn't resist – MM)*. The following data byte specifies the model of the instrument, and a MIDI channel number. It means "Okay, Yamaha stuff; if you are an RXII listening on MIDI channel I, stay tuned. If you're not, then ignore the rest of this message." Being able to specify a MIDI channel can come in handy if you have two or more RXII units and want to send a message to only one of them. To single one out for the message, simply have them listen to two different channels.

The next data byte is a parameter group and sub group number. This particular set of digits is required when working with the RXII. The fifth byte that moves down the cable is the parameter number. Yamaha has defined the number II5 to mean the total volume parameter. The next byte is really the one that sets the value of the parameter (in the case of the example, the total volume). The complete message is over when the EOX command is sent and received over the MIDI cable. Even though II5 specifies the 'Total Volume' on the RXII, other devices made by Yamaha may use different parameter numbers to indicate volume. Keep in mind that system exclusive messages will all be different for each instrument. When you want to really get down into the system exclusive codes, it's time to read the manual!

System Real Time Messages

AS YOUR MIDI system grows, timing will become more and more important. Synchronization comes in many different flavors. There are two types of synchronization available in the MIDI specification. One type, called MIDI sync, makes extensive use of still another classification of system messages called System Real-Time Messages. They work in the domain of 'real-time' (time when the MIDI system is up and running as opposed to 'stopped-time'). Since these commands are included in the MIDI data stream, they are designed to be as short as possible. All real-time commands consist of a single status byte without any data bytes tagging along. This keeps things happening as fast as possible.

The Start (IIII I0I0) message is self-explanatory. If you have one drum machine controlling another, then pushing the start button on the master unit will cause the slave to play from the beginning of the sequence or song. Stop (IIII II00) is another real-time message that can be conveyed through MIDI.

The Continue message (IIII I0II) is very similar to the start command, with one difference. While the start message will ask the slave device to go back to the beginning, this message will ask the slave device to continue playing from where it left off. If a device does not have this command implemented in its controls, it will always have to start back at the beginning of a song.

Active Sensing is an optional message. It has the binary structure of IIII III0, and is used by some companies (notably Yamaha) to insure that all the MIDI connections are constantly in proper working order. An instrument that is programmed to read active sensing commands will wait until the first such message is received. Once it receives this message the first time, it will expect another one every 300 milliseconds. If that time period passes, and another active sensing message isn't picked up, the slave device will turn off all its notes and return to normal operation. This can save a lot of headaches when someone trips over your MIDI cable and the DIN plug is yanked out of the machine.

Another real-time message is System Reset (IIII IIII). This command will ask every device connected in the MIDI system to reset themselves to their normal default (or at least power up) conditions. Be very careful how you use this command, or you could find yourself reprogramming much of the information you have entered. (Because of this, very few products implement System Reset.)

The most important real-time message is called the Timing Clock (IIII 0000). This message keeps all devices in the system in sync with each other. The timing message is sent twenty-four times every quarter note. It doesn't matter what tempo your drum machine is playing or what meter you use for your song, there will always be 24 of these messages for every quarter. If your tempo is quarter note equals 60, then there will be I440 MIDI clocks moving down the cable every second. If your tempo is quarter equals 300, then there will be 7200 MIDI clocks per minute or I20 every second.

Next month, we'll deal with the other timing system included in MIDI, the MIDI Time Code. We'll discuss other types of synchronization such as SMPTE Time Code, FSK, and pulse clocks. We'll also take a long hard look at what sequencers are and what they do. So stay tuned – MIDI channel six, mode three. ®