There's been a quiet revolution in the use and manufacture of electronic musical instruments, and it's all due to a new communications standard that appears as a few innocent sockets on the rear panel.

by ROB EVANS

At some time or other, most readers will have been frustrated by a lack of standardization when trying to interface two devices. From that special thick toasting bread not fitting in the pop-up toaster, to a new computer terminal t[illegible]refuses to talk to the mainframe sy[illegible]m; incompatibility abounds.

The music industry has been no exception over the years, the manufacturers rigidly sticking to their own particular standards for interconnecting electronic instruments. Musicians were forced to use various oddball adaptor boxes, or purchase all of their equipment from one company — hardly satisfactory solutions.

This problem was tackled on an international scale in about 1981, when the U.S. firm Sequential Circuits Incorporated proposed a universal communication standard for synthesizers. The ensuing years saw counter proposals from the Japanese companies, and worldwide discussion in an attempt to find a versatile standard, that would not become quickly obsolete.

The result of this remarkable co-[illegible]ation between manufacturers is the Musical Instrument Digital Interface standard, or as it is more commonly known; MIDI. More remarkable still, is the thorough manner in which this system has been implemented into the current generation of electronic musical instruments. From home organs to home computers, MIDI can be found in virtually any microprocessor based equipment designed to be associated with music.

What is MIDI?

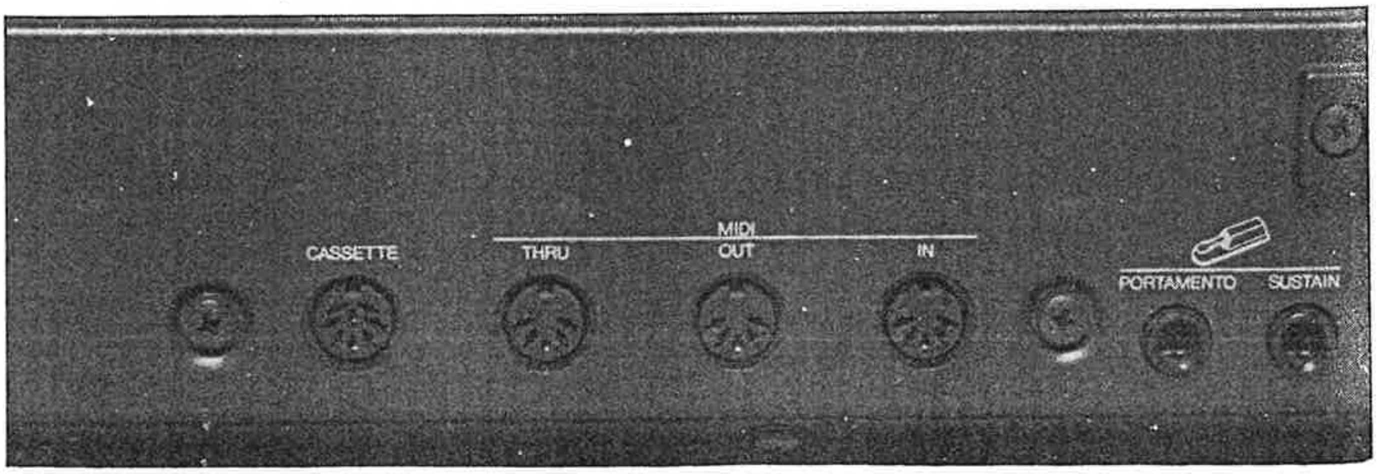
MIDI is basically a communications link. This is probably the understatement of the year, considering the effect it has had on the music industry. But leaving the musical implications aside for the moment, MIDI can be seen simply as the hardware and software standards that are used for communication between computer-based musical instruments.

These standards are defined in the MIDI 1.0 specification, which has been developed by the International MIDI Association (IMA). This publication lists the various codes that apply to standard musical performance events (e.g.,"note on"), and the electronic manner in which these codes are transferred.

Basically, MIDI is an *asynchronous* serial data link, not unlike the RS232 standard that may be familiar to many readers (see EA Reference Notebook; October 1987 issue). Each digital bit of the information code is sent one after the other, as a data "stream" along a single circuit. These bits are arranged in groups of eight to form the code word, and are preceded by a "start" bit and followed by a "stop" bit or rest period (see Fig.1).

The start and stop bits of the MIDI format are necessary due to the asynchronous, or intermittent nature of the data. A start bit (always a space or low logic level) announces the arrival of a code word, triggering the receiver timing circuits. Whereas the stop bit (always a mark or high logic level) allows the receiving circuitry to ready itself for the next code word. This next word could arrive at any time depending on the density of the transmitted data, or in the practical sense, the intensity of the musical information. Indeed, a couple of "stabbed" chords and manipulation of the pitch bend facility on a synthesizer will cause quite a flurry of activity on the MIDI bus.

To reduce timing delays between master and slave instruments, MIDI data is transferred at a rate of 31.25k bits per second (bps). For a data word with a total of 10 bits (one start, eight data, and one stop), this gives 320us for each word. A parallel system of data transfer, where all bits of a word are sent simultaneously would obviously be faster, but could not be justified due to



Typical rear panel of a MIDI equipped synthesizer, showing the MIDI connectors.

the expense
wire system
days, the R
14-pin DCI
Bus) stand
ments. Alt
well, the ra
ished its car

MIDI har

MIDI de
via a shielc
nated in 1
connectors.
floating ("g
isolated by
(see Fig.2).
safety of t
removes tl
hum and
duced in th

When a
a MIDI "i
tablished i
conversely.
rest perioc
rangement
loop practi
a logical h
dication. T
tive" (cur
mitted me
unaware c
ods. This
tem with
breaking a

The MI
a MIDI
an exact r
the MID
optional,
'included
chain" a
series.

MIDI d

The ac
MIDI sy
byte forn
the trans
is termec
the type
will tell
deal with
called (y
As fur
bytes h
(MSB) s
have the

MIDI
apply to
tiple sy
Channel
ent in t
bits, wi
types of

the expense and complexity of a multi-wire system. In the chaotic pre-MIDI days, the Roland Corporation had the 14-pin DCB (Digital Communication Bus) standard fitted to their instruments. Although this system worked well, the rapid acceptance of MIDI finished its career in one quick blow!

MIDI hardware

MIDI devices are connected together via a shielded two wire system, terminated in 180 degree, 5-pin DIN type connectors. The receiver circuitry is floating ("ground" not connected), and isolated by a high speed opto-isolator (see Fig.2). This technique ensures the safety of the receiver electronics, and removes the possibility of earth loop hum and other interference being induced in the system.

When a MIDI "out" is connected to MIDI "in", a 5mA current loop is established if a low logic level is present; conversely, no current flows during a rest period (high logic level). This arrangement is opposite to normal current loop practice where current flows during a logical high, providing line failure indication. The MIDI system is only "active" (current flowing) during a transmitted message, each instrument being unaware of the other during rest periods. This enables re-patching of the system without the havoc created by breaking a current loop.

The MIDI 1.0 document also specifies a MIDI "thru" facility, which delivers an exact replica of the signal received at the MIDI "in" connection. Although optional, this facility is almost always included to allow the user to "daisy chain" a number of instruments in series.

MIDI data codes

The actual codes transmitted in the MIDI system generally have a multi-byte format, due to the large range of the transmitted variables. The first byte is termed the *Status byte*, and defines the type of message being sent. This will tell the receiver software how to deal with the following bytes, which are called (you guessed it!) the *Data* bytes. As further identification, the Status bytes have the most significant bit (MSB) set to 1, whereas the Data bytes have the MSB set to 0.

MIDI messages may be configured to apply to a specific instrument of a multiple system, by means of the MIDI Channel number. This number is inherent in the Status byte as the last four bits, with a range of 1 to 16. These types of transmissions are loosely called *Channel* messages, while general information transmitted for all channels are designated *System* messages.

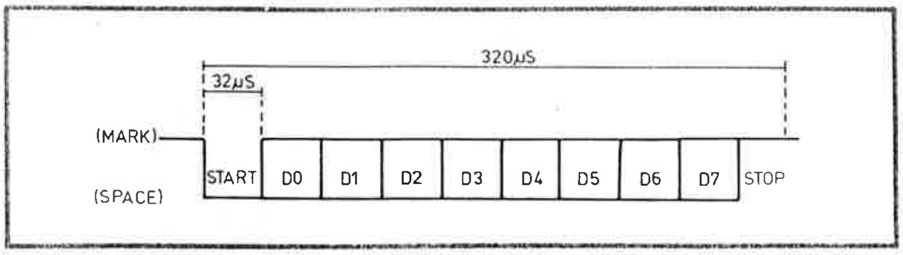


Fig.1: The MIDI data word format. MIDI data is transmitted at 31.25 kilobits per second.

A common example of a MIDI Channel message is the simple transfer of note information. When one MIDI keyboard instrument is driving another, depressing a note key on the master keyboard will induce the appropriate sound from the slave. The 3-byte message sent in this situation is configured as follows:

1001nnnn 0kkkkkkk 0vvvvvvv

Where: 1001 is the code for "note on". : nnnn is the MIDI channel number (range:0-15 for channels 1-16). : kkkkkkk is the note (range:0-127). : vvvvvvv is the key velocity (range:0-127).

In a synthesizer, the key velocity code is derived from the time taken for the key to be depressed its full distance. With logarithmic scaling, this time is roughly proportional to how hard the key has been played. Therefore the key velocity data may be used to control the resulting note volume, providing player dynamics. The code range is 1-127, defaulting to 64 for instruments without this facility. Code 0 indicates "note off" (see MIDI code table).

If a Middle C note (code 60) was depressed with a moderate velocity (say, code 64), and the instrument programmed to send on MIDI channel 5 (code 4), the resulting MIDI message would be:

10010100 0011100 01000000

An example of a System message is the MIDI Clock code, which is sent by a master sequencer (or computer) to synchronize a time related slave device, such as another sequencer or drum machine. This message is sent regularly at a rate of 24 clocks per quarter note, and is not MIDI channel specific. Of course, a clock has no relevance to a standard synthesizer, and will be ignored. As stipulated by the MIDI standard, the one-byte clock transmission will appear as: 11111000.

Although the MIDI 1.0 document is an absolute specification, it has a mes-

SUMMARY OF MIDI STATUS BYTES

Status D7 . . . D0	No. of Data Bytes	Description
Channel Voice Messages		
1000nnnn	2	Note off any event
1001nnnn	2	Note on any event (velocity 0=note off)
1010nnnn	2	Polyphonic key pressure/after touch
1011nnnn	2	Control change
1100nnnn	1	Program change
1100nnnn	1	Channel pressure/after touch
1110nnnn	2	Pitch wheel change
Channel Mode Messages		
1011nnnn	2	Selects channel mode
System Messages		
11110000	****	System Exclusive
11110sss	0 to 2	System Common
11111ttt	2	System Real Time

Notes
nnnn: channel code = channel number –1
e.g., 0000 is channel 1, 0001 is channel 2, e.t.c.,
*****: any number of bytes, followed by End of System Exclusive flag (11110111).*
sss: 1 to 7
ttt: 0 to 7

sage format that allows for data that is exclusive to a particular manufacturer. This is called System Exclusive information, and allows manufacturers to capitalise on the common operating system of their instruments. For example, the entire memory of one unit may be transferred to another in a couple of seconds (MIDI memory dump). This information could not be interpreted by an instrument from another company, and therefore has a two-byte "preamble" containing the manufacturers identification number. The actual data may be of any length, but must be followed by a one-byte System Exclusive End code. This system removes any temptation for manufacturers to compromise MIDI compatibility by the use of non-standard codes.

The above examples show only a few of the many messages available; the MIDI specification has suitable codes for virtually any music related event. A full listing and discussion of these MIDI messages is available in the MIDI 1.0 document, although a summary of Status bytes appears on page 123.

MIDI applications

Electronics has infiltrated most aspects of life, and musical instruments are no exception. A standard system that enables these common devices to communicate opens new doors for the contemporary musician.

By simply connecting the MIDI output of an elected master instrument to the MIDI input of a slave, the communication link is established if their channel numbers match. Additional instruments may then be arranged in "daisy chain" fashion by connecting a slave instrument's "thru" output to the following slave's "in" socket (see Fig.3). This basic configuration enables a musician to easily create unique sound textures and huge "walls of sound", without leaving the comfortable piano stool!

In fact, the whole MIDI protocol is very "user-friendly", requiring few programming steps to establish a large MIDI network. This is because a slave instrument simply accepts and processes data as it appears at the MIDI "in" port, even if its keyboard is being played at the same time. In effect, a MIDI equipped synthesizer reacts to MIDI messages as if they had originated from its own control panel or keyboard.

Similarly, the MIDI "out" socket is always active, sending messages as the keyboard and controls are used.

Synthesizers are now becoming available in a "keyboardless" form, virtually identical to the version with a keyboard, but produced in a rack mount cabinet. These units are called Expander Modules, and are more cost and space efficient. To upgrade a system, the user simply connects the expander module to the master keyboard via a MIDI link, remotely controlling all performance aspects.

Currently available modules often have multi-timbral capabilities, which is ideal for sequencing applications (see Glossary). An example of this is an eight-voice polyphonic instrument (eight separate sound generators), where each voice may be assigned to a different sound and MIDI channel. The sequencer is then able to control eight independent, monophonic sound sources from the one unit, via eight MIDI channels.

A typical example of a complete MIDI system is shown in Fig.4, although many other configurations are possible. It consists of a master keyboard, a number of expander modules, a drum machine, and a sequencer.

In such an arrangement the sequencer is the centre of activity, recording MIDI messages sent from the master keyboard, and playing them back to the entire system. As the messages for each section of a composition are recorded into the sequencer, a MIDI channel may be nominated, which will then be inherent in this information (see MIDI codes). When played back, the slave modules will "look" for messages corresponding to their set channel, therefore responding to different sections of the composition.

The sequencer will also transmit regular MIDI clock codes in sympathy with its internal tempo clock. This informa-

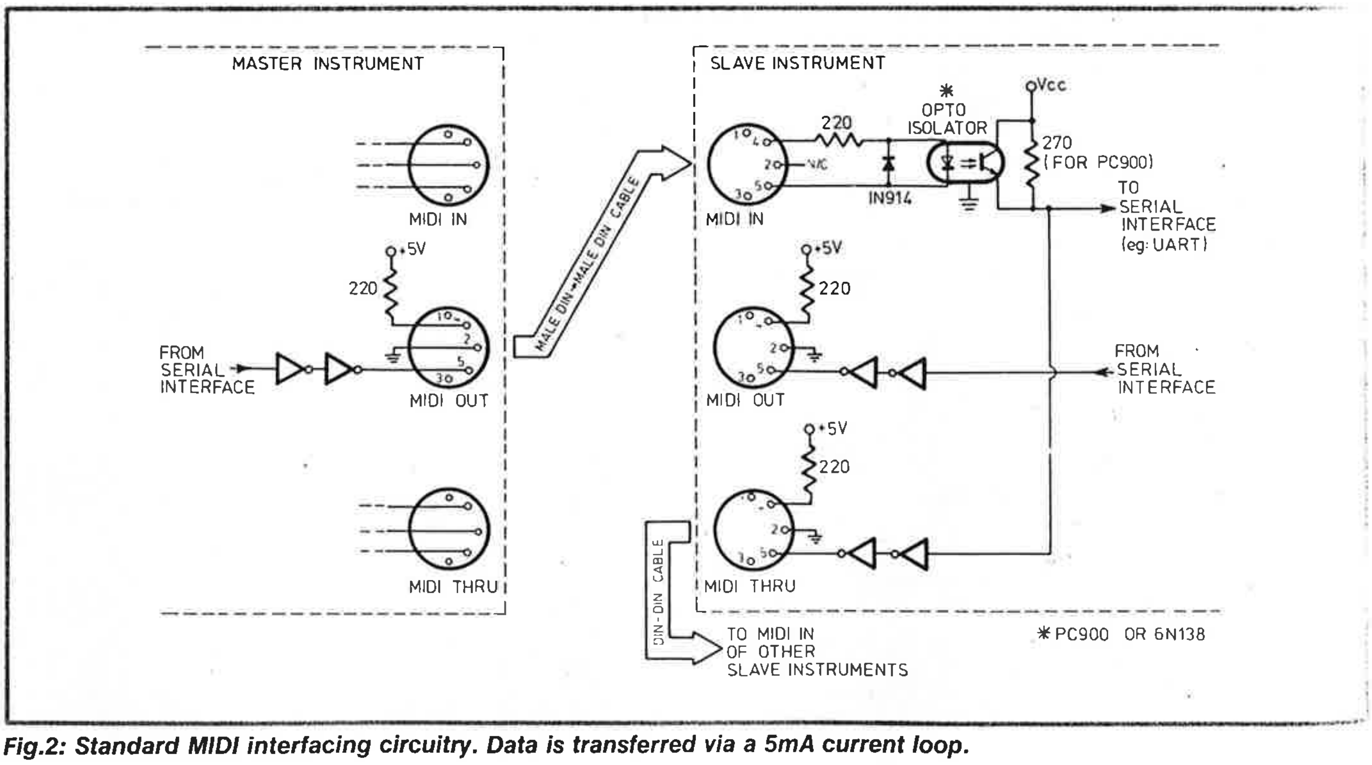


Fig.2: Standard MIDI interfacing circuitry. Data is transferred via a 5mA current loop.

tion is interleaved with all other messages, and interpreted by the drum machine as its tempo clock, synchronising the whole system.

Such a system is quite powerful, although still rather primitive when compared to the facilities of some contemporary recording studios. The studio MIDI system may reach as far as the ubiquitous mixing console, where channels and outputs are programmed to mute under the control of the master sequencer.

Even recently produced signal processing equipment, such as delay and reverb units carry substantial MIDI facilities. By interfacing them in a MIDI system, many of the parameters may be altered on call, or as part of a sequence. The proverbial one man (or woman) band takes on a new meaning with the MIDI protocol!

Computers and MIDI

Without computers there is no MIDI, for every device that communicates via this system is microprocessor based. The common Personal Computer is obviously in this category, and therefore should be able to exchange information with MIDI instruments. This fact has been exploited by a number of small companies, who are now producing MIDI conversion kits and appropriate software for some popular computers. Amongst others, software is available in Australia for the Commodore 64, the Apple II, Apple Macintosh, and IBM PC or compatibles.

A home computer coupled to a MIDI interface can be a powerful music composition tool and sequencer. Music may be composed, displayed, edited, played and printed with one composite software package on a common PC. In fact, the implemented software will generally set the limits of its performance. Conversely, a manufacturer's dedicated sequencer is often more user-friendly and reliable, particularly when compared to a home developed interface and software package (this is based on personal experience!).

Designing MIDI interface hardware is assisted by the slightly odd MIDI baud rate of 31.25kbps. This rate is based on an even division of common computer clock rates (1MHz divided by 32, 2MHz divided by 64, and so on). This enables the processor clock to be divided easily by the internal dividers of a serial interface chip (e.g., Motorola 6850 ACIA). Some computers already have a suitable serial port, and with the appropriate clock division, are capable of transmitting MIDI information directly.

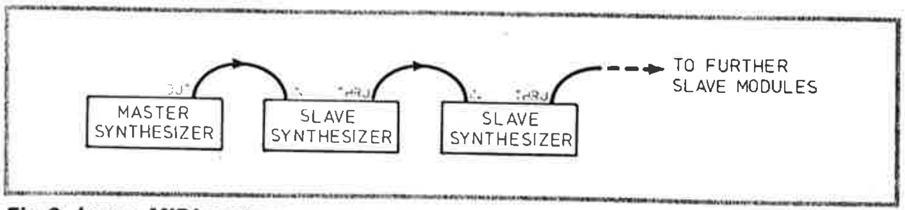


Fig.3: Large MIDI systems may be implemented by simply "daisy chaining" a number of instruments in series.

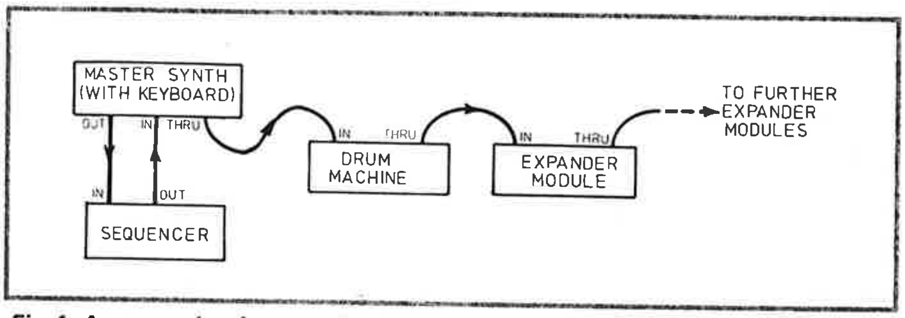


Fig.4: An example of a complete MIDI system capable of programming and playing entire musical compositions.

The future

As early as 1984, some manufacturers were critical of perceivable delays encountered when transferring MIDI note data. These delays creep into the proceedings when a large number of notes are sent to a number of instruments, along the same line. At 320us per word and 3 words per MIDI note message, it takes about 1ms to transfer a single note. If the master instrument is sending different 5-note chords to each slave instrument (not an unusual situation), we already have about a 15ms delay. This time can easily increase if continuous multi-byte information, such as Pitch Bend messages are sent simultaneously. Although these delays are not a problem for a real-time performance (that is, a human performer actually *playing* the master keyboard!), large sequencer-based systems can be affected.

In extreme cases of high information density, the data crowding reaches the extent of effectively "saturating" the MIDI bus. Not surprisingly, the whole system grinds to a halt. Some of the more expensive sequencers solve this problem by providing a separate serial interface and MIDI out socket, for different MIDI channels. This means that each output socket will transmit MIDI information for only one channel, and is connected to only one slave instrument.

The IMA people may have envisaged these problems when developing the MIDI 1.0 specification, for it states that successive messages of the same type may be sent without a Status byte. If notes are turned off by sending a note-on command of zero velocity, then the note-on Status need only be sent at the beginning of a note data stream. What was 3 bytes for note-on and 3 bytes for note-off, becomes 2 bytes for each. Not much difference you might say, but it could eliminate the proverbial straw!

In general use, these crowding and delay problems rarely appear and MIDI has an ensured future. A whole industry has emerged to supply contemporary musicians with MIDI software, interfaces, and general MIDI manipulation equipment.

The MIDI explosion has even prompted the development of new musical instruments, such as MIDI transmitting devices based on percussion, wind, and stringed instruments. From the performance aspect, a musician no longer has to hide behind a bank of instruments (although he may want to!). All that is required to run a large number of MIDI equipped instruments is one versatile MIDI control device.

GLOSSARY

SYNTHESIZER: Electronic musical instrument capable of synthesizing naturally occurring sounds, or constructing unique timbres.
SEQUENCER: Machine with a memory system able to record, edit, and playback the electronic codes for musical time related events (notes etc.).
DRUM MACHINE: A dedicated sequencer arranged to control internally generated drum or percussion sounds.