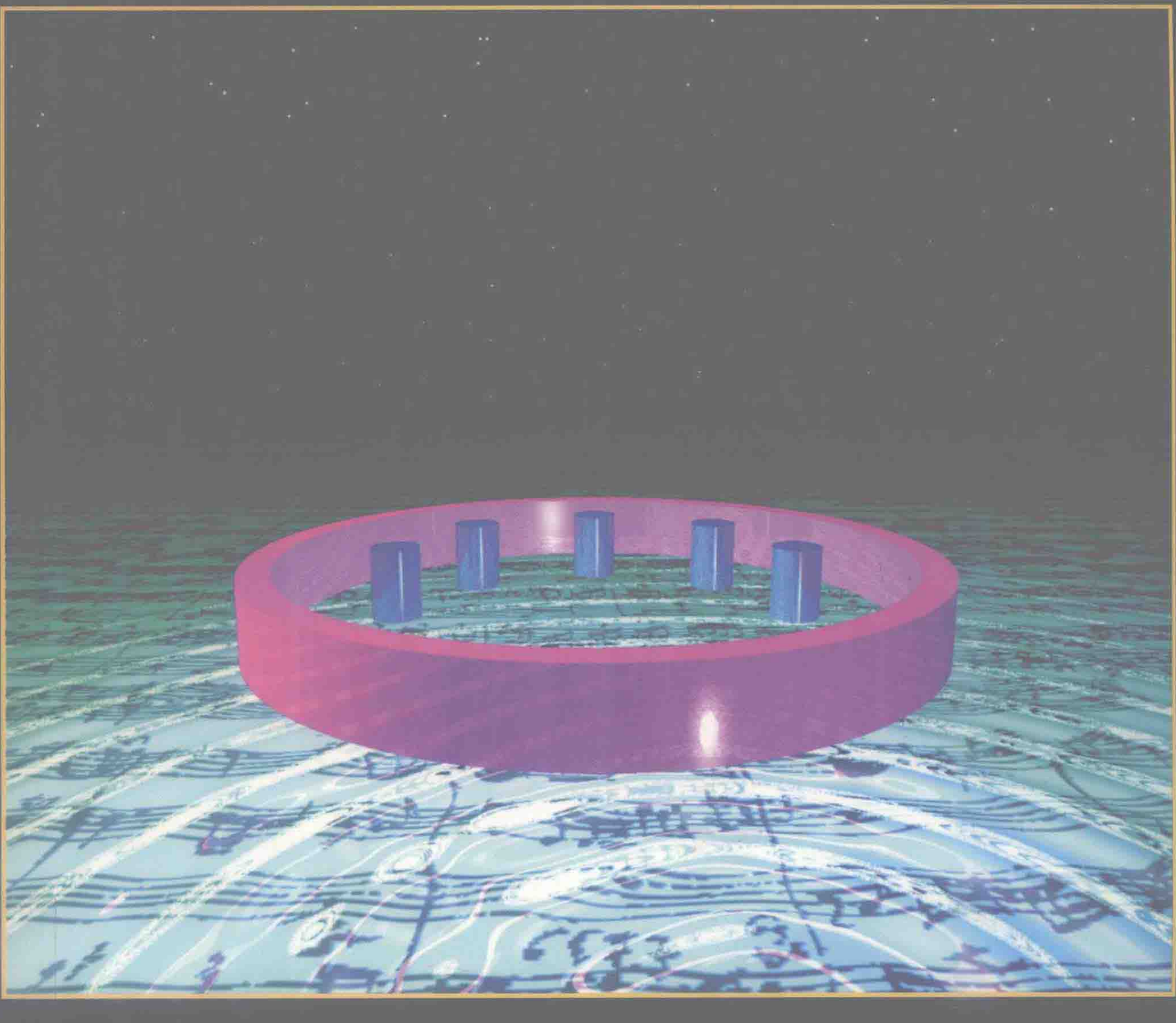
INCLUDES GENERAL MIDI

EWLY REVISED AND EXPANDED EDITION

THE MIDI COMPANION

The Complete Guide to Using MIDI Synthesizers, Samplers, Sound Cards, Sequencers, Computers and More!

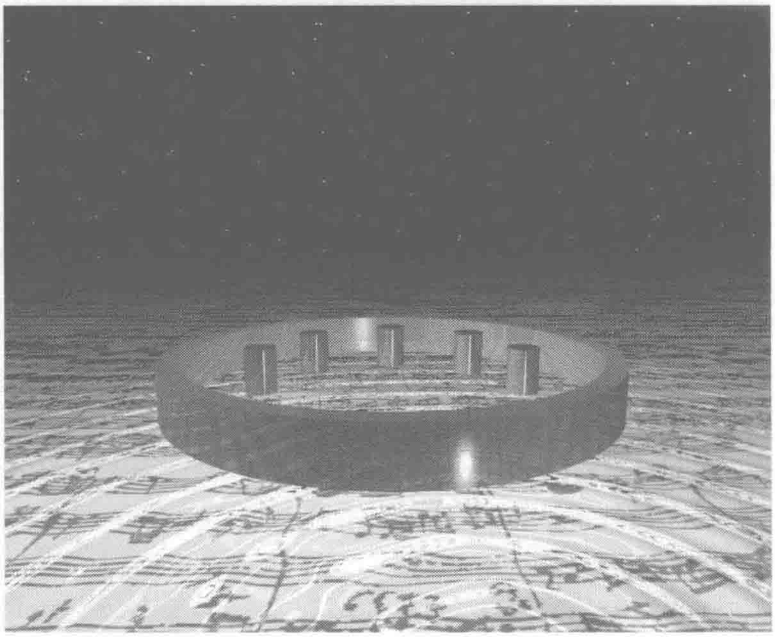
JEFFREY RONA



HAL•LEONARD

THE MIDI COMPANION

by JEFFREY RONA



Edited by RONNY S. SCHIFF
Associate Editor SCOTT R. WILKINSON
Graphic Design, Illustrations and Diagrams by WEST DESKTOP GRAPHICS
Cover Graphic by LACHLAN WESTFALL and JEFFREY RONA

Library of Congress Cataloging-in-Publication Data
Rona, Jeffrey C. (Jeffrey Carl), 1957-
The MIDI Companion (formerly: MIDI, the Ins, Outs and Thrus)
1. MIDI (Standard) 2. Computer sound processng.
I. Schiff, Ronny. II. Title. III. The MIDI Companion
MT723.R66 1987 789.9'9 86-27746
ISBN 0-88188-560-6
1994 ISBN 0-7935-3077-6

HAL•LEONARD™
CORPORATION
7777 W. BLUEMOUND RD. P.O. BOX 13819 MILWAUKEE, WI 53213

Table of Contents

Chapter 1: **Beginnings** **5**
The events and needs in the world of music technology that led up to the creation of MIDI.

Chapter 2: **The Language Of Machines** **9**
The technology of MIDI comes from computers. The ways that the two are related, and the underlying techniques used to send digital data, such as MIDI.

Chapter 3: **What MIDI Sends–A Musical Breakdown** **15**
The musical elements of MIDI, based on the elements of a musician's performance.

Chapter 4: **The MIDI Studio** **19**
The components of a MIDI studio. A look at everything that goes into a personal music system for the home or professional studio.

Chapter 5: **The Basics** **25**
How the technology of computers and digital codes make MIDI work.

Chapter 6: **An Overview Of MIDI** **29**
The name and function of every MIDI command, grouped in tables by their musical functions.

Chapter 7: **Channel Voice Messages** **33**
The application of the most important of the MIDI messages–the ones used to indicate the playing of notes.

Chapter 8: **MIDI Controllers** **39**
The messages MIDI uses to indicate when pedals, levers, or wheels are employed for musical expression on an instrument are listed and explained in greater detail.

Chapter 9: **Channel Mode Messages** **42**
MIDI Mode messages alter the way an instrument responds to incoming MIDI messages. A discussion of the four basic MIDI Modes.

Chapter 10: **Real Time Messages** **45**
The messages used to synchronize MIDI instruments, such as sequencers and drum machines. These codes allow groups of instruments to function as an integrated system.

Chapter 11: **System Common Messages** **51**
A number of MIDI codes are reserved to enhance the operation of other MIDI functions. These messages assist MIDI in linking parts of a system together. Plus, information on MIDI Mode messages.

Foreword

Hmm, another new book on MIDI? In the first decade of the Musical Instrument Digital Interface, there have been a tremendous number of tomes published on the subject. Some were about as exciting as excerpts from the phone book, focusing too much on the technical, propeller-head stuff—least significant bytes, sys-ex packets, microsecond flurblblobs, and the like, and not enough on the important part of what MIDI actually *does*. It would seem that the authors of some of those lexicons forgot that MIDI is *used by musicians* to make music.

Jeff Rona, whose involvement with MIDI goes back to its introduction, details the technical side of MIDI in all its code crunching glory, but he does it in a way that puts things in musical perspective. It's an approach that should help you to connect the technological dots with an un-intimidating and user-friendly experience, even for the technophobic music-minded MIDI neophytes among you.

Enjoy,

Dominic Milano
Editor
Keyboard Magazine

1

Beginnings

The way music is made has been changed forever. MIDI instruments are now the tools of artists from an enormous range of styles and traditions. The quality, and perhaps even the quantity, of music has grown as a result of the MIDI phenomenon. First, a little history:

The 'sixties and 'seventies were an explosive time for the creation of new musical instruments. In addition to the blossoming use of electric guitars and new keyboards such as electric organs and pianos, a whole new breed of musical instrument was beginning to appear on albums and in concerts—the electronic synthesizer. These large, odd looking and odder sounding machines were based on simple *analog* electronics. They used electrical voltages to create and control sounds. Higher voltages would make higher notes and lower voltages made lower notes. Several small companies began to make instruments, all based on the concept of control voltage (CV). Short electrical cables called *patch chords* would feed the control voltages around these instruments to manipulate the sound's character and shape.

For musicians who wanted to play from a standard organ-like keyboard, special CV keyboards were built to control the rest of the instrument. These early synthesizers could play only a single note (*monophonic*) at a time. To get more musical lines, you either had to buy more synthesizers, or record parts onto tape. These synthesizers were difficult to set up, use and maintain, but they gave those musicians something they could get no other way—fresh new sounds.

The *monophonic* (single note) Moog and ARP brands of synthesizers were already bending quite a few ears by the mid 'seventies with bands such as ELP, Genesis and others, when the Oberheim company introduced the first commercial *polyphonic* (able to play several notes at a time) keyboard synthesizer. Relative to its unwieldy predecessors, it was simple to use, had a built-in keyboard, was able to play four notes at a time (a 400% improvement!), and had a simple array of knobs and switches you could manipulate to quickly create rich, wonderful new sounds. It was far more portable and easy to program than most of its predecessors.

Soon after, more easy to use, good-sounding, polyphonic synthesizers began to appear: Sequential Circuits, Yamaha, Moog, Roland, ARP and other companies introduced new models of electronic instruments, all able to play multiple notes simultaneously. Just a few years earlier, what was an expensive, unwieldy and difficult to use machine, was becoming a popular instrument with a growing crowd of diverse musicians.

After polyphony, perhaps the next most important advance in early synthesizer technology was the incorporation of programmable *memory* into instruments. All polyphonic synthesizers have a small built-in computer that "looks" at each key on the keyboard to see if it has been pressed, and then passes those notes on to the available *oscillators* (which are the special electronic circuits in a synthesizer that make the actual sounds). That small computer could also help store and recall sounds created by the user into the synthesizer's built-in memory (like taking a snapshot of all the knobs and buttons on the instrument). This opened up a whole new world for live performance.

Prior to programmable memory, the reason that people like Keith Emerson and Rick Wakeman had such extravagant keyboard setups on stage was that each of the instruments could only be set-up to produce a single sound per show. Hours of preparation were needed to *patch* together the sounds and the different instruments. When memory came along, it allowed a single synthesizer to be used for several different sounds during a live show or recording session, by simply pressing a single button.

Figure 1-1 Keith Emerson

Adding memory to the synthesizer made it many times more useful. But many early synthesizers—like many cars—had personalities of their own. Some got wonderful, thick brass. Others were more adept at woodwinds, or strings, or bells, or sound effects, or pianos, or colorful tropical birds, or the laugh of small friendly aliens. What was needed next was a way to combine the best of each instrument into a single, useful musical system.

A technique that some early synthesizer players adopted to create new sounds was to play the same part on two keyboards at the same time, one hand on each instrument. A keyboardist could then use each instrument to its best advantage: strings from the "string synth," brass from the "brass synth," and so on. This was an awkward technique at best, and one's polyphony was limited to the number of fingers on one hand, typically five.

Rock musicians such as Keith Emerson, shown in **Figure 1-1**, and Rick Wakeman became famous for the enormous stacks of electronic keyboards they would stand in front of and play. Joe Zawinul, of the 'seventies jazz group Weather Report, developed a unique technique for playing on two keyboards simultaneously. He placed himself between a pair of ARP 2600 synthesizers, one of which had its keyboard electronically reversed, going from high notes on the left to low notes on the right (**Figure 1-2**).

All these elaborate measures were designed to accomplish one thing—getting the most from these great new instruments. The layering of sounds upon sounds became an important tool, almost like a trademark sound for some of these and other artists.

Then, in 1979, came the next big step: some new keyboards were coming equipped with computer interface plugs on the back. Instruments from the Oberheim, Rhodes and Roland companies could, for the first time, be connected to another of the same model of synthesizer. For example, an Oberheim OBX synthesizer could be connected to other OBXs. When you played on the keyboard of one, both would play whatever sound was programmed. This was an improvement for performers, since sounds could now be layered on top of each other while playing a single keyboard, but it didn't answer the big question of how to connect *different* instruments from *different* brands together for unique combinations.

One person who took matters into his own hands was jazz musician Herbie Hancock. Newly enthralled with the technology of synthesizers, he spent a small fortune to have many of his electronic instruments custom modified to connect with each other, allowing him to mix and match sounds any way he wished. For the first time, instruments of different makes were connected with each other by means of a common, though custom, digital connection.

More and more rock and jazz musicians were approaching the instrument makers to try and get their own equipment to interconnect. In addition, the first digital *sequencers* (a device that records and replays back a performance on an electronic musical instrument) were starting to show up. These sequencers, such as the Roland MC-4 Micro-Composer (*see* **Figure 1-3**) and the Oberheim DSX, were yet another reason to want compatibility between products from the different instrument makers. It would be possible for one person to sequence and play back all the parts of a song on a group of synthesizers. The

Figure 1-2 Joe Zawinul of the Group Weather Report

Roland Micro-Composer was a primitive, four-track sequencer that produced either control voltages, used extensively for controlling earlier analog synthesizers, or used a special "Roland only" digital connector for some of their newer instruments. Oberheim's sequencer was quite a bit more sophisticated, but was limited to use only with their own OBX model synthesizers.

Figure 1-3 The Roland MC-4 MicroComposer

TIME FOR A CHANGE

Time was ripe for a change to occur in the musical instrument industry by the early 'eighties. Synthesizers were no longer a *techno-oddity*, and sales of instruments to the mass market musicians, as well as to professionals, were growing quickly. There were more companies involved now from Japan, the U.S. and Europe. The diversity of keyboards, drum machines, sequencers, and other musical devices was growing rapidly. To move up another notch in technology and accessibility, the synthesizer industry needed to take a lesson in compatibility from the computer industry.

Computer makers have long depended on certain technical standards to ensure compatibility between computers and other devices. For example, the *modem* is a device that lets computers exchange information over telephone lines. It makes no difference what the makes, models or cost of the interconnected computers are. Now, millions of people on thousands of different computers all speak the same "language" because their equipment was designed using the same technical standard for modems. Other examples of computer standards are disk drives, printers, cables, memory chips, and many types of software. Compatibility strengthened the new personal computer industry and was a major factor in its amazing success. There are many examples of technical standards that allow devices from different companies to work together. Look at the success of items such as VCRs, video cameras and tapes, cassette machines and tapes, stereo equipment and a host of everyday gizmos.

Twice a year, the members of the National Association of Music Merchandisers (NAMM) hold a huge convention to show new musical products and find new ways to market musical instruments and accessories. During one of these shows in 1982, a meeting of a small group of synthesizer manufacturers took place at the request of Dave Smith, President of Sequential Circuits, a popular synthesizer company of that time. Engineers from many of the major synthesizer companies were in attendance. They discussed a proposal for the adoption of a *universal standard for the transmitting and receiving of musical performance information between all types of electronic musical instruments*. The original proposal was called UMI, for *Universal Musical Interface*.

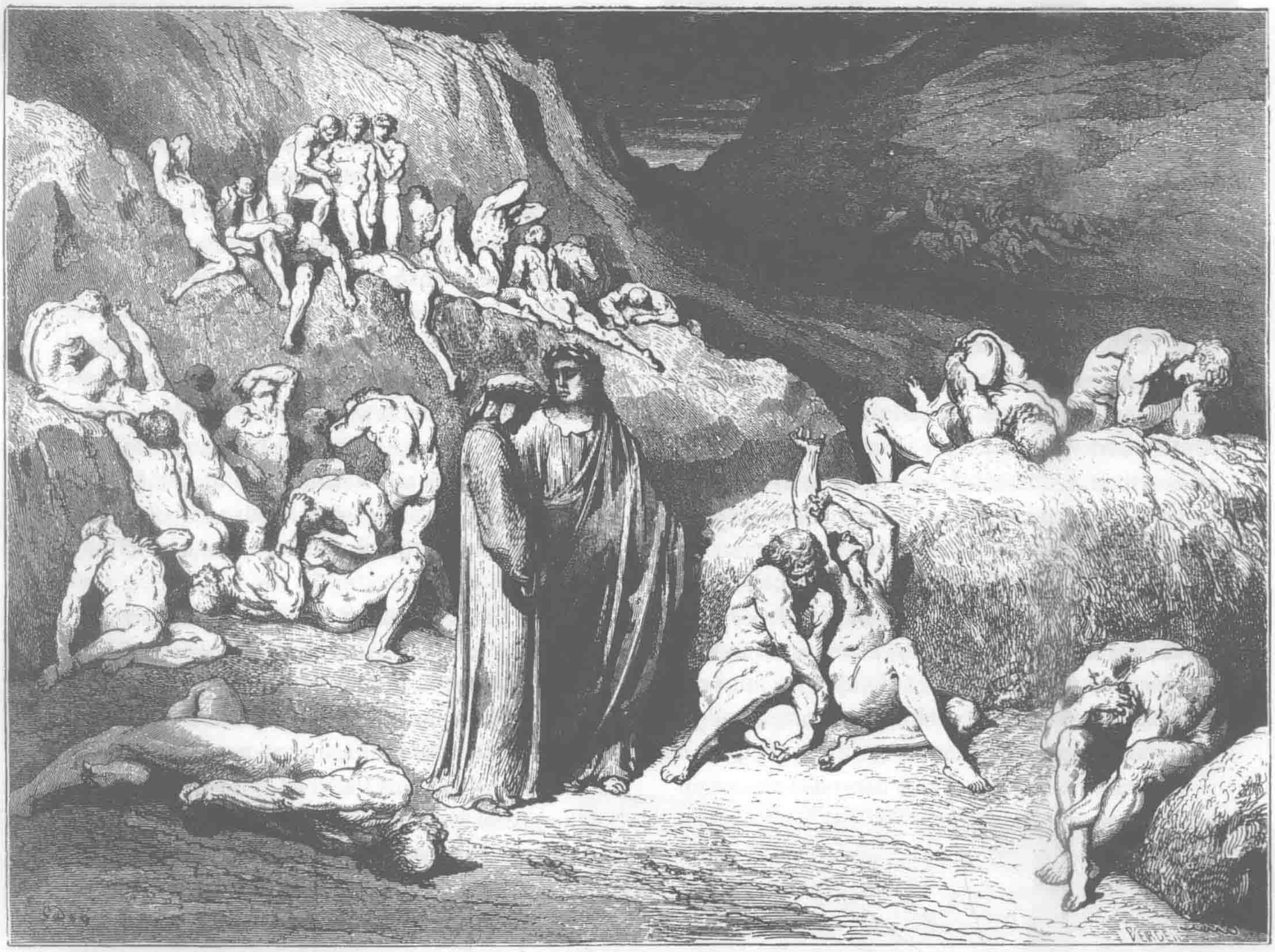
The original proposal went through a significant number of revisions before being renamed and becoming the *Musical Instrument Digital Interface*, or MIDI standard. Several prominent Japanese musical instrument companies became involved in engineering the final version. It was a truly international cooperative venture. Finally, in 1983, Sequential Circuits from the U.S. and Roland from Japan introduced the first keyboards with MIDI, soon followed by virtually every other synthesizer company in the world!

Within three years after MIDI's introduction, almost no electronic instrument was made in the world that didn't have a MIDI plug on it. It became a true universal standard. To this day there is no competition to MIDI for connecting all types of electronic musical instruments together or for creating personal musical systems. Like computers, MIDI is used by millions of people for thousands of applications. It's also being used in fields other than just music, such as theatrical lighting, computer games and recording studio automation.

From the beginning, the MIDI standard was designed with room for growth and improvement. Since its start, new features have been added, while others have been defined more clearly. A great deal of room was left for expansion without sacrificing the main power of MIDI—simplicity and compatibility with all other existing MIDI instruments.

This book is a guide for the musician, performer, producer, composer, engineer, computer enthusiast, student, or anyone wanting to get a good understanding of how MIDI works, and how to work with MIDI. It will assist you in learning the nuts and bolts of MIDI technology. The more you understand how MIDI operates, the easier it is to use the musical tools it provides. You will develop an understanding of how a MIDI system can be put together quickly and easily for any occasion.

You will gain the knowledge needed to make sound (pun intended) purchases of MIDI and MIDI-related equipment. You will learn how to get the most out of any musical situation that calls for using electronic musical instruments. Examples of many different MIDI systems are shown to help with the creation of the right music system for your needs and budget. You will probably be surprised at just how simple MIDI is to understand and use. As your knowledge about MIDI increases, you will see the wonderful possibilities available to you from the technology of music.



Synthesizer engineers see the need for MIDI.

2

The Language of Machines

Telephones and cable television use electricity to send information over wires. MIDI sends information over wires too, though in a slightly different way. MIDI uses the same technology as computers to relay information from one machine to another.

When you speak on the telephone, the microphone in your phone's handset converts the sound of your voice into electricity. The electricity travels over a single wire to the telephone on the other end of the line, which converts that electricity back into sound through the speaker in the handset. A second wire sends sound from the other phone back to yours. This is a simple circuit, *but its simplicity makes it no less powerful!*

Cable television sends even more information over a wire. Dozens of channels of information (in the form of pictures and sound) travel from the local cable company to your house over a single cable. Using the tuner on your TV or cable box, you select the one channel you want to watch. Even though all the other programs currently running are coming into your home via the cable, you are tuning them out while allowing your favorite show to appear on the TV screen. This is an example of *multichannel* transmission.

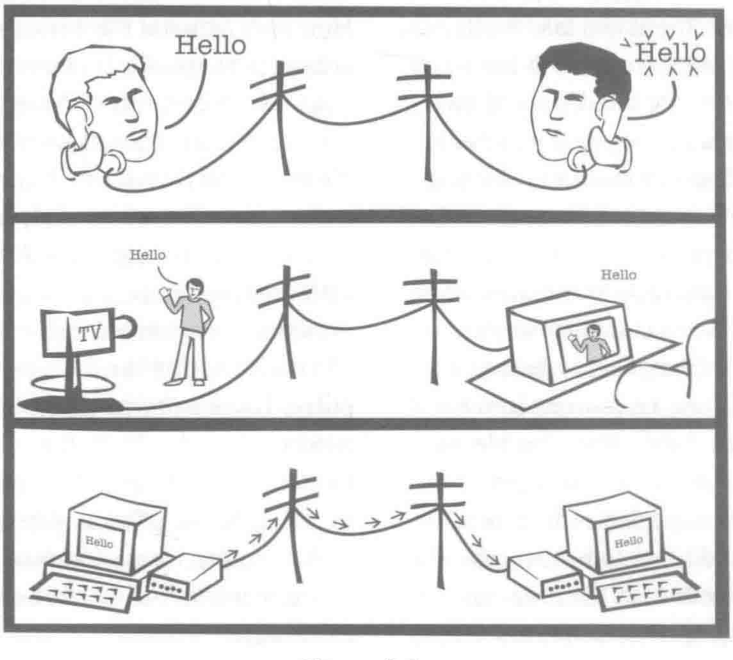


Figure 2-1
Information can be sent over wires in a number of ways.

Computers can also be connected together in order to send many kinds of data, but they do so differently (they don't have voices or game shows). One method is by using a *modem,* which connects computers via telephone lines, allowing computer users to exchange electronic mail, programs, pictures, play games, type messages, or other types of information. Unlike telephones or TV, computers communicate *digitally*, which means that they use numbers to represent everything. Computers communicate with each other via numbers. MIDI is digital too. Remember, MIDI stands for Musical Instrument *Digital* Interface.

If you already play or use synthesizers, drum machines, sequencers, digital effects, or most any electronic music equipment, you already know how to work with computers, because that's exactly what you are doing. The personal computer is but one type of computer. With synthesizers, you are simply working with a *different type of computer* in a different way—instead of producing mailing lists, you are making sounds and music.

This all leads up to a brief explanation of a few key parts of computer technology that apply directly to MIDI. *MIDI is the sending and receiving of information between two computers*, not computers that do spreadsheets, but the kind that make *music*. Let's understand what digital means a little better.

INSIDE THE MIND OF A MACHINE

If you could see into a computer's inner workings, here's what you would find?

Computers, and all so-called "intelligent machines," are not necessarily so complex, or so smart. Computer processors (called *Central Processing Units*, or *CPUs*) are found in a wide range of everyday items from desktop personal computers (PCs), to banking machines, calculators, auto-dial telephones, video games, VCRs, and yes, MIDI synthesizers and other electronic musical instruments. All computer-based machines have the basic elements shown in **Figure 2-2**.

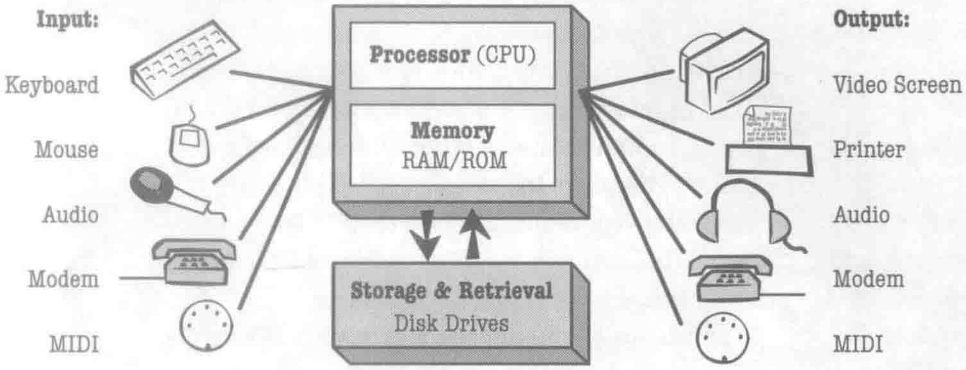


Figure 2-2
A computer system is made up of many parts.

A computer takes information from the outside world through its *input*. It then organizes it, modifies it (if called to by its *software* program), and stores it within its memory. Input can take many possible forms, depending on the nature of the machine and the desired tasks at hand. Typing on a computer keyboard, using a mouse, modem or any of the devices in **Figure 2-2** are all examples of input.

A PC displays information on its video screen with letters, words, numbers, colors or pictures, but it holds that information internally as *numbers in its memory*. That is the only way a computer stores information. In order for information to go into a computer, it must be *digitized,* i.e., turned into the numbers a computer understands. For example, as you type on a computer keyboard, the computer is converting each letter into a specific code in order to store it in its memory. In addition to text, computers can receive pictures or sounds with the proper hardware and digitize them into groups of numbers.

Once inside, these numerical codes are used by the computer to represent and store all the information they have received. Each letter of the alphabet is a number. So is each color. More complex pictures and graphics are groups of numeric information in the computer's memory. A computer program (*software*) is a series of actions that the CPU can carry out on the information stored in the computer's memory. These actions, too, are stored as digital codes.

So, numbers are the language of machines. In order for people to understand the information inside any computer, the codes must be translated back into some language we humans understand—letters, words, pictures or sounds. That's the *output*. For example, if you wish to retrieve the information that you typed on the computer's keyboard earlier, the computer must first translate the digital codes back into the letters and numbers you typed in order to display them on its screen.

A CD player is really a specialized kind of computer. The CD disc is the input. The computer gets huge streams of numbers that are encoded on the disc and outputs them as sounds to your amplifier and speakers. A computer game is little more than pictures and a set of complex rules, all represented in the machine as numeric codes, and translated on the screen as colorful graphics and actions. In the world of computers, numbers can do just about anything.

STORING INFORMATION

How does a digital machine store a number? The answer is in small bits of electricity in the memory chips of the computer. These memory chips are called *RAM*, which stands for Random Access Memory, and they store all the information put into a computer. These bits of electricity are called, appropriately enough, *bits*. A bit is like a microscopic little switch—it is either ON or OFF. Bits are grouped and arranged inside RAM in patterns that, when added together, become numbers to the computer. Those patterns create a code that only computers can understand. It is the role of the computer to convert those patterns of electricity into information that humans can understand.

A computer doesn't look at each of the bits in its memory individually, but in uniform small groups called *bytes*. A byte represents a single numerical value to the computer, and is usually made up of 8, 16 or more bits. It's the computer's job (with the help of software) to convert those bytes into all kinds of information—including letters, numbers, symbols, graphics, sounds or commands. Musical instruments use bytes to represent program settings, front panel knob and button settings, key presses, rhythmic patterns in a drum machine and anything else the synthesizer will do or remember.

The more bits used in a byte, the bigger the range of possible numerical values it can represent. Every computer or other digital device is designed to recognize bytes of a particular bit size. More is better, but also more complex and expensive. That's one reason why there are more *and* less powerful machines available.

A computer's RAM memory is considered *volatile*, meaning it holds information only while the machine is on. If information needs to be kept after the machine is shut off, then it is necessary for another means of memory storage and retrieval to be included, such as a disk drive or memory card.

DANCING BIT TO BIT

When any two computers are connected together, either with a modem or by a more direct link, there is no need to translate the information into "human recognizable form." It can remain in digital form from output to input. A computer communicates with other computers by sending each of its stored bits electrically over a wire. A receiving computer will then store the bits in its own memory to re-form the information into numbers.

However, in order for computers to share information, a standard form of *transmission* is essential. Telephones everywhere all use the same kind of wiring, dialing system, voltages, microphones, and speakers. As a result, any phone can dial up any other phone. The same is true with computer modems. They are all compatible with each other. However, there are also other more direct ways of linking computers and other digital devices without modems or phones. It can take quite a few bits to say something important. It takes about forty bits for a computer to just to send the word "HELLO."

Some computer systems can send information over a multi-wired cable several bits, or a whole byte at a time. This technique is called *parallel transmission* and it works something like **Figure 2-3**.

Parallel transmission schemes use cables and connectors with several wires in them. Each bit of a number is sent over one of the wires in the cable. This is the most efficient way to send data between two computers.

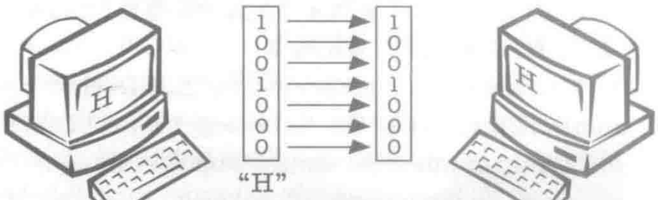


Figure 2-3
Parallel transmission sends data over several wires.

Serial transmission is another technique for sending information from computer to computer. Instead of sending several bits of information simultaneously over several wires, the bits leave the computer in a single file line over a single wire:

At the other end of the wire, the receiver picks up the bits and reassembles them into the byte of information, in this case (**Figure 2-4**) the numerical code for the letter "H." It's called *serial* transmission since the bits are moving in a series (one after another) through the wire.

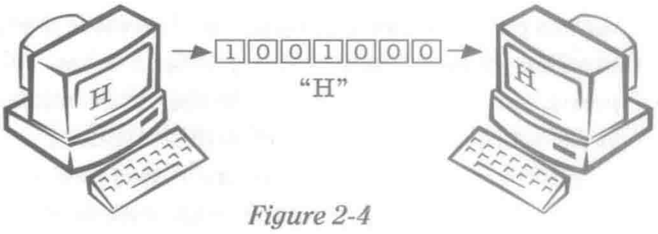


Figure 2-4
Serial transmission sends all the data over a single wire

Computers, or other computer-based devices that can be linked to share information, use one of these two methods of digital communication—serial or parallel. There are advantages and drawbacks to each: While parallel transmission is more efficient and generally faster, the input and output hardware and the cables are more expensive. Longer parallel computer cables unfortunately can also act as very good radio antennae, inadvertently sending or receiving radio interference that can disrupt any nearby devices such as TVs, VCRs, radios or amplifiers. In turn, those machines can also radiate interference into a parallel cable and disrupt the data as it travels between devices. Parallel cables cannot be longer than just a few feet before the data starts to become unreliable and inaccurate.

On the other hand, serial input and output, while a bit slower due to its method of sending numbers one bit at a time, has several advantages: It is significantly less expensive to implement, the cables can be much longer, and there is no radio noise interference problem.

SENDING INFORMATION THE MIDI WAY

So what does all of this have to do with MIDI and musical instruments? The technology behind PCs and MIDI instruments is the same. Computers are composed of a microprocessor, digital memory, and some means of sending and receiving information with the outside world. Sending and receiving can either be between the computer and you, or between the computer and another computer. For example, a button pressed on the front panel of a synthesizer will cue the instrument's internal computer to do something such as change to a different sound or alter the current sound. The computer can display the new sound's name or changed parameter on the synthesizer's front panel. Recalling a stored sound (also called a *program* or *patch*) causes the synthesizer to look inside a particular part of its memory to get the various parameters of the sound, and then move them to an area to be played or changed. When a key is pressed on the keyboard of a synthesizer, the computer will interpret that to mean "play this note now!"

It is the microcomputer chip that makes modern electronic instruments sound so vivid, while so easy to use, compared with their pre-digital predecessors. Using computer technology does not automatically make things more complex, in fact it makes working with instruments much simpler. Without the memory and CPU chips that go inside instruments, it would be impossible to retrieve a stored sound at the touch of a button, as current instruments can do. It would be impossible to have a drum machine or a sequencer, or to "sample" sounds, or to have a graphic visual display on an instrument to assist in the creating of new sounds. Most kinds of sound processing such as delays and reverbs could not be produced. MIDI would not exist.

MIDI, which is a *Digital Interface* (interface is a fancy word for *connection*), must use some standard means of moving musical "data" from one instrument to another, just as computers send words and other information back and forth.

With the knowledge of parallel and serial transmission technology in mind, the creators of MIDI had an easy time deciding which kind of transmission method to use. The disadvantages of parallel outweighed its advantages, and so they chose the simpler, less expensive but more reliable, serial transmission technology.

Let's see that in big letters:

MIDI USES SERIAL TRANSMISSION

To keep things simple and inexpensive (so everyone could get their hands on it), the creators of MIDI chose a simple, readily-available five pin plug to put on all MIDI-compatible instruments and cables, called a *DIN plug*. It had been used previously for many applications in audio and video, but none involving electronic musical instruments. Thus, it would not be confused with other connectors found in music studios, such as audio or electrical cables, but people could buy them at their nearby electronics store. The MIDI connectors found on every MIDI instrument and cable look like this:

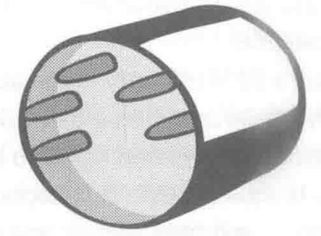


Figure 2-5
A standard MIDI Connector with 5 pins

The plugs on the instruments are "female" with five small holes, while the cables are "male" with 5 matching pins on both ends. On all MIDI plugs and cables, opposite the five little connectors, is a small notch that helps you align the small pins on the cable with the corresponding holes on the plug. MIDI cables can be found in many lengths, from 1 foot up to about 50 feet (15 meters), which handle most needs.

Since serial data transmission needs only a single wire to send information from machine to machine, why are there five wires in a MIDI cable?

Not all the pins and wires are used:

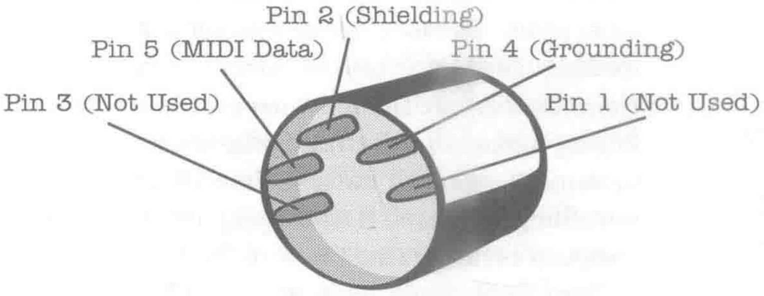


Figure 2-6
Not all pins are used on a MIDI cable

- Pin 1 and Pin 3 are not used at all. In most MIDI cables, they are not even connected to the wires.
- Pin 2 is used as electrical shielding. That means it is attached to a wire that is wrapped around all

the other wires in the cable. This important feature helps prevent the cable from transmitting or receiving any kind of electrical or radio interference that might ruin the data as it travels down the cable.

- Pin 4 is a grounding wire. Grounding is a part of most electric circuits to ensure that the electricity flows in the proper direction.
- Pin 5 is the only real sender of MIDI information!!

Since MIDI uses a single wire in the cable to send information, the musical data that MIDI sends travels in only one direction over a single cable. However, MIDI was devised to allow information to go both directions between two instruments, by simply using two cables. At the same time, MIDI can also pass data on to a third, fourth, fifth instrument, or as many synthesizers as you can afford. To accomplish this, it was decided to have three different MIDI connectors on each instrument:

- One to receive data IN,
- One to send the data OUT,
- One to pass incoming data on THROUGH (spelled "THRU" in the MIDI world) to another MIDI instrument.

Here is what a typical MIDI instrument's MIDI connectors (also called "ports") look like:

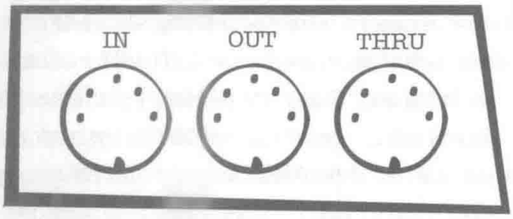


Figure 2-7
Most MIDI devices have these three connectors

Three plugs—IN, OUT, and THRU—are located on the backs of virtually every MIDI instrument or device. To prevent the possible electrical problems that can cause hum in an audio system, all MIDI ports are *opto-isolated,* which means they convert electricity to light and back to electricity at each port. This keeps each instrument electrically isolated from the others. Understanding what each connector does lets you see the logic behind putting a MIDI system together.

MIDI OUT

Perhaps the most important concept to understand is that ***MIDI does not transmit sound*** over wires the way audio components in a stereo system do. Instead, it sends digital codes that *represent what is being played on the instrument*. As you play on a MIDI keyboard, your *performance* is examined by the computer in the instrument. Your actions are then translated by the instrument's computer into a stream of MIDI codes. That information is sent out the instrument's MIDI OUT port to other synthesizers that reproduce the performance, but using sounds of their own.

MIDI IN

MIDI keyboards can be viewed as being two machines in one: One is the computer processor that monitors the keyboard, program memory, front panel displays and MIDI ports. The other part, under the control of the on-board computer, is the electronics that actually make the sounds.

The MIDI IN port receives incoming MIDI information and sends it to the instrument's computer. Once there, it is analyzed and acted upon the same as a performance *on* the instrument itself, such as pressing a key to play a note. There is little or no difference to the sound-making part of a synthesizer, whether the command to play a note comes from a key press on the instrument itself, or comes as a command via the MIDI IN port.

MIDI THRU

In order to send MIDI data on to other instruments in a chain, a third MIDI connector called THRU duplicates any MIDI messages that come to the MIDI IN port of an instrument. It is a "repeater" of the MIDI data. An important concept to understand when putting together a MIDI-based music system is that *anything played on a keyboard only goes out the MIDI OUT, and not the MIDI THRU.*

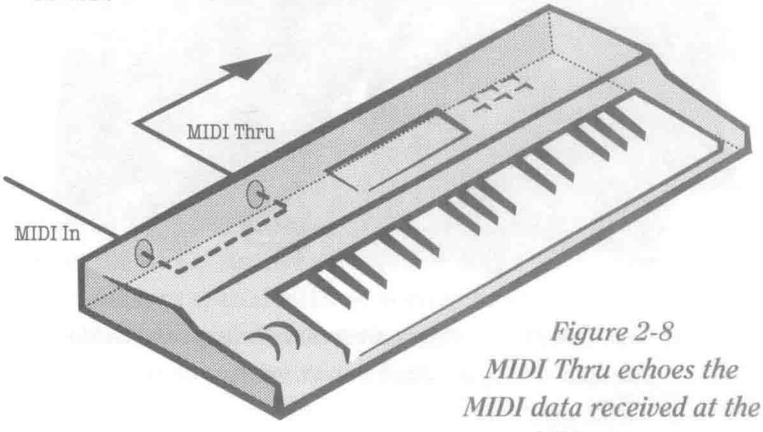


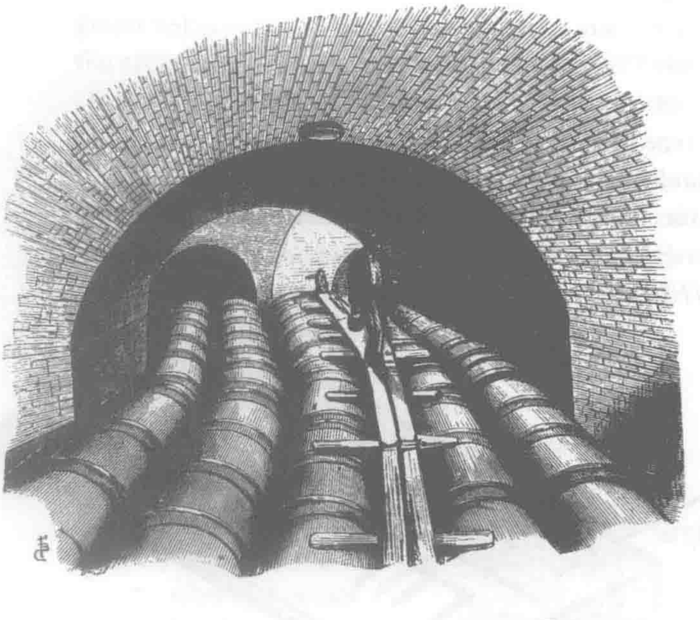
Figure 2-8
MIDI Thru echoes the MIDI data received at the MIDI In port.

Like the symbols in hieroglyphics, the dots and dashes of Morse code, or the tones in a Touch Tone phone, the bytes in a computer are a *code*. MIDI information is a computer code as well. Every time a key is pressed or a pitch wheel is moved, one or more bytes are sent out an instrument's MIDI OUT port. Other synthesizers connected to that sending instrument are looking for those bytes to come over the wire, which are then interpreted back into commands for the synthesizer to obey. Digital codes are *messages* to a computer, just like the words in a phone call or the letters on this page are to a person. Some messages are interpreted as commands ("do something now!") and others as information ("the last note played was a C#")

When computers are connected to share these codes they form a *network*. MIDI is a network for musical instruments.

FAST AND FASTER

MIDI sends information at a rate of 31,250 bits per second. This speed is called a *baud rate*. Sometimes it is referred to as 31.25 Kbaud, which stands for *kilobaud*, meaning thousands of bits per second. Since MIDI is serial, it sends data one bit at a time. All MIDI messages use 8 bits for the information. To guarantee perfect accuracy when MIDI data is being transmitted, two additional bits are used in every byte, bringing the total up to ten bits per MIDI message byte. Which means that MIDI sends about three bytes of data every *millisecond* (one thousandth of a second).



MIDI data moves in a single direction through the cable (as seen through a high power microscope).

The MIDI message for playing a single note has three bytes in it. At the MIDI speed of 31,250 bits per second, it will take .96 milliseconds to send the command to play a note from one instrument to another. To keep things simple, this number is rounded off and called one millisecond. It takes another three bytes–another millisecond–to shut that note off. If it takes one millisecond to turn the note on and one to turn the note off, then MIDI can play approximately 500 notes a second—a lot of notes!

However, the ear is highly sensitive. It can detect if two sounds are played simultaneously or if they are slightly apart in time. How slightly? Many people feel that when there's a gap of less than 20 to 30 milliseconds between two sounds, they are perceived as sounding simultaneously. As the time gap gets longer, the ear will perceive it as two sounds. This is still well outside the time delay factors in MIDI. The data for an eight-note chord will take only about four milliseconds to transmit over MIDI. Even large chords, though transmitted serially, still sound simultaneous. With these numbers, assume that even very complex music sent over MIDI should be heard exactly as it was played.

Some of this depends on the types of sounds being used. Complex music using sounds with fast, clicky attacks might begin to have some noticeable timing delays. If the same music were to use sounds with slower attacks, such as slow string or brass sounds, no delays would be perceived.

From the start, there has been a bit of controversy over MIDI's speed and accuracy. Is it fast enough? Should it be replaced with something faster? For most music systems, it works perfectly. The occasional tiny time lags that may occur in some people's MIDI systems are more often due to the instruments themselves, rather than MIDI's speed and efficiency. The speed of a MIDI instrument's built-in microprocessor to respond to MIDI information, process it and begin the actual note is greater in some instruments than others. Some cars have more horsepower and some synthesizers respond to MIDI faster. Most instruments are amazingly fast at responding to MIDI, and the vast majority of people using MIDI have no problems at all with the speed or timing accuracy of their music systems.

Those three connectors—IN, OUT, and THRU—are the backbone of all MIDI hardware. Understanding these will give you the basis for being able to put together a MIDI system to suit any purpose. Now, look at what is actually communicated between instruments over MIDI.

3

What MIDI Sends – A Musical Breakdown

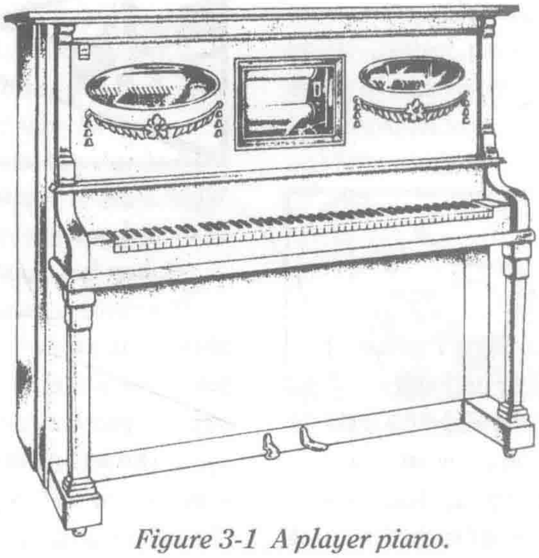


Figure 3-1 A player piano.

The old player piano—a marvel of technology in its time: A roll of paper with holes in it passes over a metal bar. There are holes in the metal bar, one for each of the 88 keys on the piano. A vacuum pump inside the piano draws air in through the holes in the bar. When a hole in the paper comes over a hole in the bar, air is allowed in, which then triggers a mechanism to move the hammer onto the appropriate string and strike the note. There was a special recording piano built solely for creating the paper rolls. The player pianos that people bought for their homes were for playback only, not unlike a record or CD player. As far as dynamics or musical nuance were concerned, they didn't exist on the player piano. Each key was struck with exactly the same force, creating a performance that might at best be called forceful, and at worst cacophonous. They were still a hi-tech wonder of their day.

What was "encoded" on the player piano's paper rolls wasn't the actual sound of a piano, but simply what was *played* on the piano. The main idea behind MIDI is the same: to allow what is *performed* on one MIDI instrument to be played on any other MIDI instrument. Your physical actions are analyzed by the instrument's computer and converted into a series of codes that are then sent out of your instrument's MIDI OUT port over a MIDI cable to all other MIDI instruments that are connected to yours. They will play exactly what you are playing, but with their own sounds. In essence, the instrument you are playing, which is called the *master*, controls all the other instruments in the system, which are the *slaves*. It's as if another player was sitting at each instrument and playing it exactly in time with you. MIDI allows for nearly unlimited connection and layering of electronic instrumental sounds.

MIDI doesn't just stop at layering different synthesizers together. It is also used for recording your musical performances for playback later, like a tape recorder, but far more powerful and fun. Instead of recording sounds to a tape, which allows for very little in the way of further editing, MIDI data can be recorded into special MIDI recording devices or computers. This is called *sequencing*. Once recorded into a sequencer, your performance can be edited in any number of ways and replayed back by any other MIDI

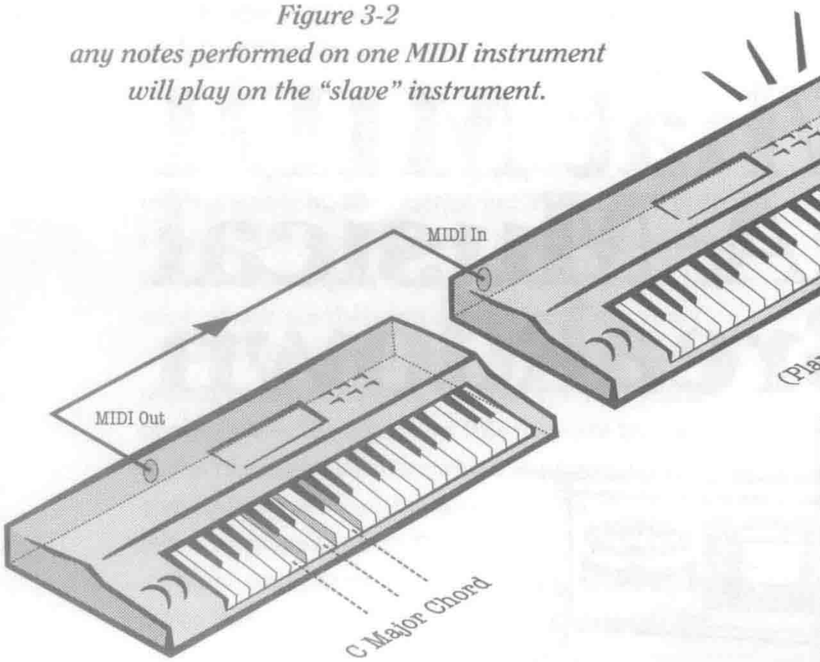


Figure 3-2
any notes performed on one MIDI instrument will play on the "slave" instrument.

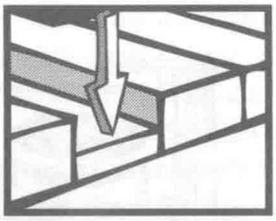
instrument or connected instruments. If you wish, MIDI lets you be your own band or orchestra. You are limited only by your imagination and budget.

The word MIDI has really come to mean two things: The first is hardware—the plugs and cables found on all MIDI instruments used for the transmitting of musical data. But MIDI also refers to the digital codes themselves sent out over the MIDI cables from instrument to instrument.

Each piece of equipment in a MIDI setup plays a specific role in the system. There are transmitters, which send MIDI data, and there are receivers, which get and respond to MIDI data, as shown in **Figure 3-2**. Some instruments do both, while some do just one or the other.

MIDI's original design was based around the keyboard as a means of performance control, but its usefulness goes far beyond that one type of instrument. There are MIDI controllers for guitarists, drummers, woodwind players, and violinists. These devices are played like their traditional models, but their job is to generate MIDI information. MIDI has allowed for the creation of new kinds of instruments that never existed before. MIDI is also used for some non-musical applications as well, such as operating and automating theatrical lighting gear, or automating recording studio and video post-production equipment.

You've seen the hardware side of MIDI. Now here's a look at the software side—the codes that come from a musical performance. These are the main elements of the MIDI code...

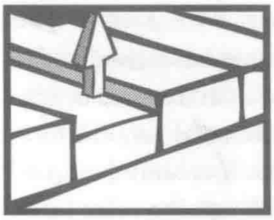


PRESSING A KEY

When a key is pressed on a MIDI controller, such as a keyboard, a digital message is sent out that says:

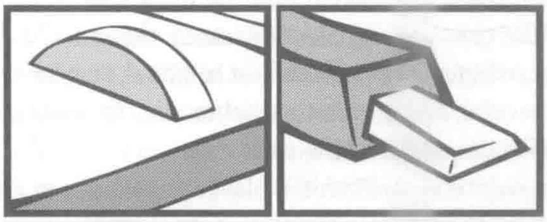
"A key has just been pressed!"

That simple message will command a slave synthesizer to play a note. The message is followed by two more pieces of information: The first says *which* key was pressed (middle C, or A flat below middle C, etc.). The second one indicates *how quickly* that key was pressed (velocity), which tells other instruments the dynamic (loudness or softness) of the note just played.



RELEASING A KEY

MIDI treats the pressing of a key and the releasing of a key as two different events. MIDI instruments don't know the length of time that a key is held down. Every MIDI message is sent out just as the action occurs. The moment a key is pressed, the code indicating that action is sent. The moment a key is released, *another* code is sent to indicate that movement.



WHEELS AND PEDALS

Typical MIDI instruments have one or more knobs, wheels, pedals, or levers to control various musically