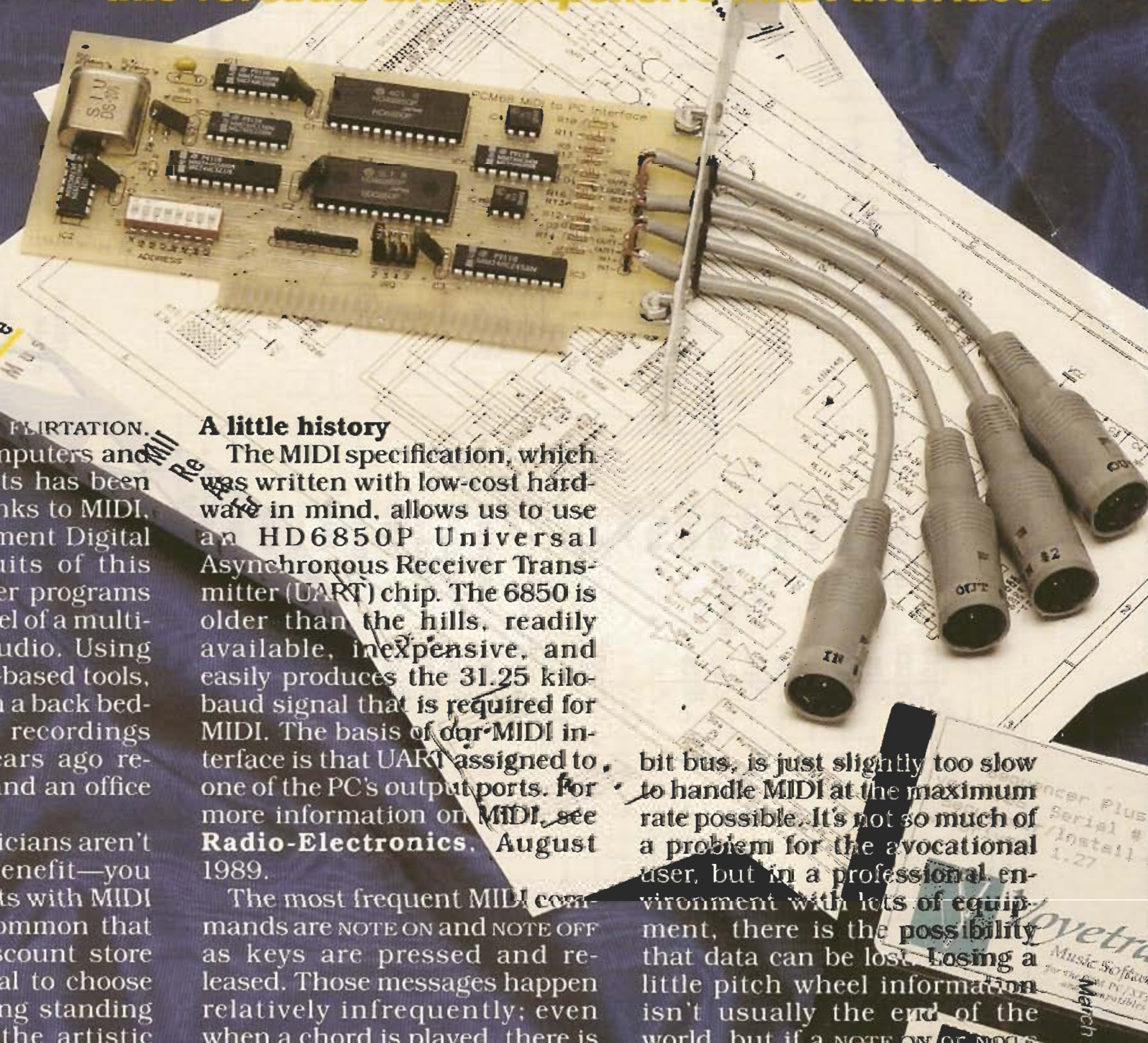
BUILD THIS MIDI INTERFACE FOR YOUR PC

Bring the world of music to your PC with this versatile and inexpensive MIDI interface.



JOHN SIMONTON

AFTER A LONG-TERM FLIRTATION, the marriage of computers and musical instruments has been consummated, thanks to MIDI, the Musical Instrument Digital Interface. The fruits of this union are sequencer programs with the look and feel of a multitrack recording studio. Using those and other PC-based tools, a single musician in a back bedroom can produce recordings that only a few years ago required orchestras and an office staff.

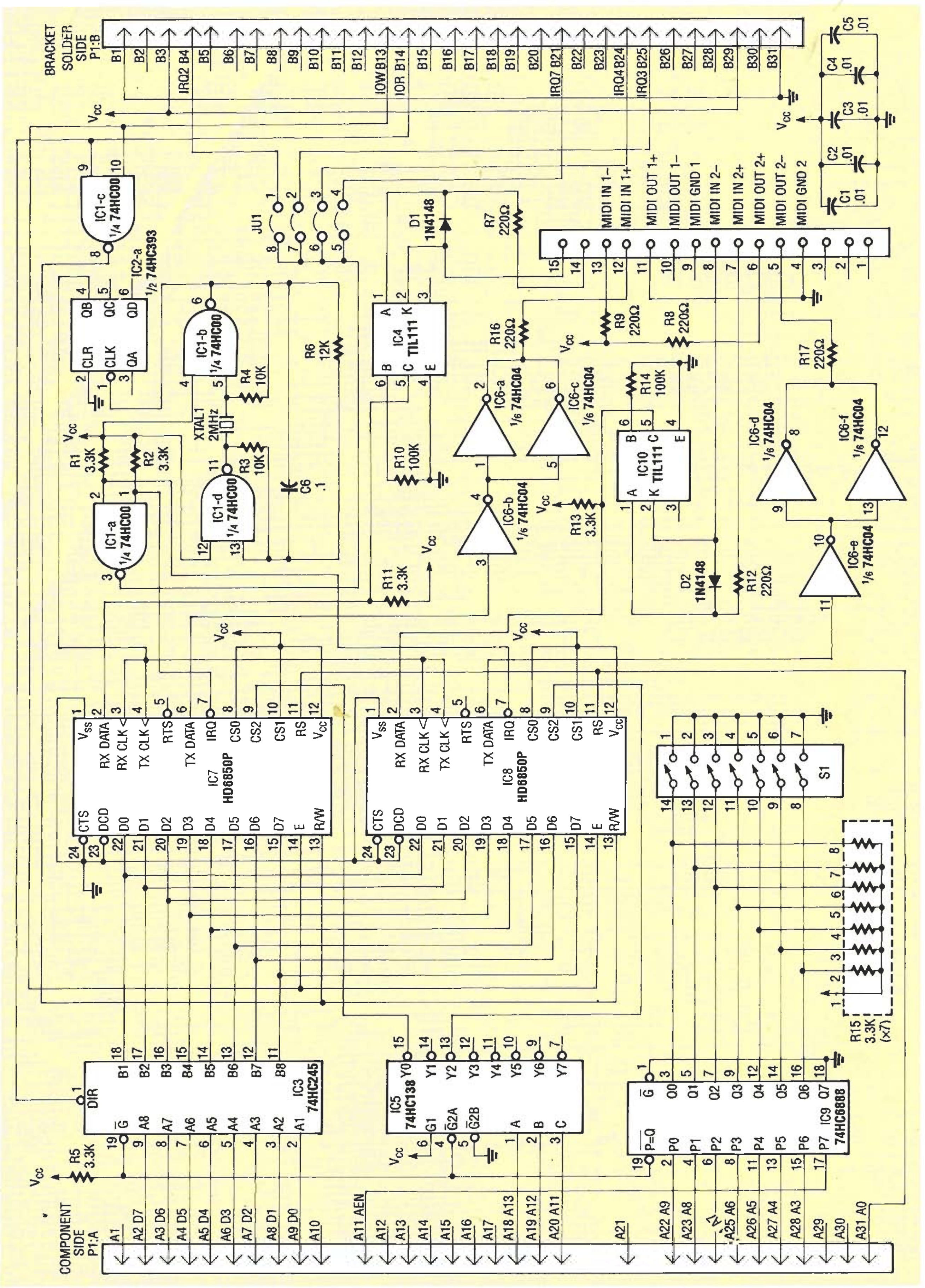
Professional musicians aren't the only ones to benefit—you can too. Instruments with MIDI interfaces are so common that even your local discount store probably has several to choose from. The only thing standing between you and the artistic gratification of writing and performing like the philharmonic is the interface from your PC to the MIDI world, so lets take care of that little detail right now. We'll show you how to build a MIDI interface that is both low-cost for the beginner and upgradable for the serious user. With little further delay we begin, but first...

A little history

The MIDI specification, which was written with low-cost hardware in mind, allows us to use an HD6850P Universal Asynchronous Receiver Transmitter (UART) chip. The 6850 is older than the hills, readily available, inexpensive, and easily produces the 31.25 kilobaud signal that is required for MIDI. The basis of our MIDI interface is that UART assigned to one of the PC's output ports. For more information on MIDI, see **Radio-Electronics**, August 1989.

The most frequent MIDI commands are NOTE ON and NOTE OFF as keys are pressed and released. Those messages happen relatively infrequently; even when a chord is played, there is generally a slight delay between the individual notes. But the protocol also has provisions for continuous controllers such as pitch wheels and foot pedals, and the messages that signal that kind of activity can really spew out data. With nothing more than a UART for an interface, the original IBM PC, with its 4.77-MHz 8088 MPU and 8 bit bus, is just slightly too slow to handle MIDI at the maximum rate possible. It's not so much of a problem for the avocational user, but in a professional environment with lots of equipment, there is the possibility that data can be lost. Losing a little pitch wheel information isn't usually the end of the world, but if a NOTE ON or NOTE OFF command comes in the middle of it, missing that data is a very big deal.

The first widely accepted MIDI interface to overcome that problem was the MPU-401 made by Roland. It has its own dedicated processor and memory, and gets around the lost-data problem by maintaining a first-in-first-out (FIFO) buffer for the MIDI data



34 **FIG. 1—TWO 6850 UART's are the heart of our MIDI interface. The circuit can be built in either a single-port version for the beginner or dual-port for the more experienced.**

received. It also does some tricks such as data filtering and providing a metronome.

All well and good, but the introduction of the first XT-class PC made an intelligent interface unnecessary because the computer alone was fast enough to handle the throughput. As faster and more powerful PC's have become available, the need for an intelligent interface has decreased until finally the MPU-401 has technically become a bottle-neck in the system (though you would never notice that the interface is slowing things down).

What *has* turned out to be a noticeable problem to serious users is the bandwidth limitation of the MIDI channel itself. The solution to that problem has been multiple, separate In and Out jacks, equivalent to multiple interface cards. And here the MPU-401 runs into serious problems. Synchronizing multiple 401's is not trivial—and worse than that, each has to have its own slot and dedicated interrupt. Interrupts and slots, is there anything more precious in the PC world? You can switch MPU-401's into a "dumb" mode so that several of them share an interrupt. But then you have expensive UART's and serious slot depletion.

So, having faster computers and the need for lower-cost multi-port interfaces has started a resurgence of interest in UART's, and all software publishers support them. De-facto standards being what they are, the MPU-401 "standard" continues to hang on. But meanwhile, really hip "power users" are stepping back in time to just the sort of card that we'll come up with here.

PARTS LIST

All resistors are ⅛-watt, 5%, unless otherwise noted.
R1, R2, R5, R11, R13*—3300 ohms
R3, R4—10,000 ohms
R6—12,000 ohms
R7, R8*, R9, R12*, R16, R17*—220 ohms
R10, R14*—100,000 ohms
R15—3300 ohms × 7, 8-pin SIP
Capacitors
C1–C5—0.01 µF, Mylar
C6—0.1 µF, Mylar
Semiconductors
IC1—74HC00 quad NAND gate
IC2—74HC393 dual 4-bit binary counter
IC3—74HC245 octal transceiver
IC4, IC10*—TIL111 opto-coupler
IC5—74HC138 3-to-8 line decoder
IC6—74HC04 hex inverter
IC7, IC8*—6850 UART
IC9—74HC688 8-bit magnitude comparator
D1, D2*—1N4148 silicon diode
Other components
J1, J2, J3*, J4*—inline 5-pin female DIN connector
JU1—8-pin IDC header w/jumper blocks
S1—7-position DIP switch
XTAL1—2 MHz crystal
Note: Components marked with a "*" are optional for a second MIDI in/out.
Miscellaneous: Printed circuit board, hold-down bracket, wire, solder, etc.
Note: The following are available from PAIA Electronics, Inc., 3200 Teakwood Ln., Edmond, OK 73013 (405) 340-6300:
- **PCM68pc double-sided, plated thru, PC board and bracket—$24.95**
- **PCM68k kit of all parts needed to build the single-port interface, including PC board, bracket, wire, MIDI jacks, etc.—$49.95**
- **PCM68ex expander parts for adding second MIDI port—$14.95**
- **MIDIpac MIDI starter set consisting of PCM68k interface kit and Voyetra SPJr. 64-track polyphonic sequencer software for PC/clones—$99.00**

Add $3.00 P&H per order.

Design analysis

The 6850 UART's, which are the heart of our MIDI interface, are shown as IC7 and, optionally, IC8 in the schematic Fig. 1. The circuit can be built in either a single-port version for the beginner or dual-port version for the more demanding user. Much of the rest of the circuitry is concerned with decoding addresses and managing control lines on the PC's slot.

The lowest-order address line (A0) directly drives the RS (REGISTER SELECT) pins (pin 11) of the UART's to allow selection of either Status/Control or Data registers internal to the chip. We'll look at what those registers do when we test the interface. The next two address lines (A1 and A2) are not used, so each chip occupies 8 bytes of space consisting of four 2-byte chunks which overlay one another.

The next seven address lines (A3–A9) are routed to one set of inputs on IC9, a 74HC688 8-bit magnitude comparator. The other "side" of IC9 connects to DIP switch S1 and seven pull-up resistors in SIP-network R15. When the pattern of bits from the address bus matches the pattern of bits set by S1, pin 19 is pulled low. Notice that the 8th input to IC9 (Q7, pin 18) is grounded on one side and connects to the slot's AEN on the other (P7, pin 17). An address match will be valid only when AEN (ADDRESS ENABLE) is low, indicating that it's not the DMA controller that has the bus.

An address match does two things. It strobes IC3, a 74HC245 transceiver, which routes data either from the slot to the card or from the card to the slot depending on the direction selected by the $\overline{\text{IOR}}$ (I/O READ) line (which connects to the DIR pin (pin 1) of the chip). It also enables IC5, a 74HC138 3-to-8 line decoder which does the final address decoding for UART selection. UART IC7 is selected when A11–A13 match the pattern 0h, and IC8 is selected when the pattern is 2h. $\overline{\text{IOW}}$ (I/O WRITE) ties to the R/W pins of the UART's to select either a read or write to the chips.

The $\overline{\text{IRQ}}$ (INTERRUPT REQUEST) output pins of the two UART's are pulled up by R1 and R2 and combined by NAND gate IC1-c so that an interrupt request by either of them shows up on a line which can be routed to IRQ2, IRQ3, IRQ4, or IRQ7 depending on the placement of jumper JU1. Subsequently, the software will poll the two UART's to determine which of them generated the interrupt.

A 500-kHz transmit and receive clock starts out with the oscillator formed by the two NAND gates IC1-b and IC1-d. The frequency of the oscillator is set to 2 MHz by crystal XTAL1 which is then divided by 4 in one section of the 74HC393 dual 4-bit binary counter IC2.

Two TIL111 opto-couplers (IC4 and IC10) are used on the MIDI inputs to prevent grounds though the path. A continuous ground on the MIDI inputs would likely duplicate a similar ground at the audio inputs and outputs, leading to circulating ground currents and noisy audio. The TIL111's are not the fastest opto's in the world, but are more than fast enough for this application and less expensive than their faster brothers. On the MIDI outputs, two inverter stages from IC6 are paralleled to increase drive current capabilities.

Assembly

Because things run fairly slowly on slot I/O operations of even the fastest PC, there are no extraordinarily high frequencies involved on the PCM68. That means that prototyping boards and wire-wrap can be used to put together a card if you like. Be careful, though; 0.8-inch spacing between slots in a PC doesn't leave a lot of room for wire-wrap pins. And a misplaced conductor can cause a lot of damage in no time at all. It should go without saying that the shortest possible wires should be used to get a signal from the card edge to the IC it connects to.

Of course it's always best to use a PC board for any project, and you can either make your own from the foil patterns we've provided, or purchase a ready-made board from the source mentioned in the parts list.

A parts-placement diagram is shown in Fig. 2. IC sockets can be used, but certainly are not necessary. Note that if you've elected to put together the single MIDI in/out configuration, you can leave out the following parts: IC8, IC10, D2, R8, R12–R14, and R17.

Since PC-slot access holes weren't designed with MIDI in mind, they typically aren't wide enough for DIN connectors to peek through. Current practice

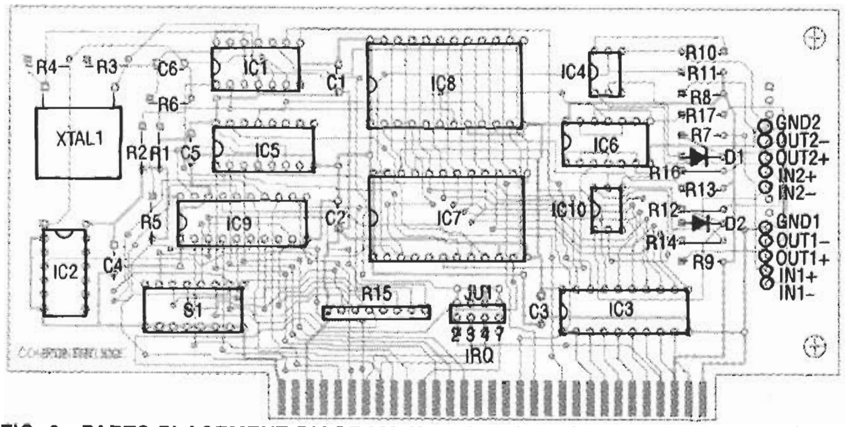
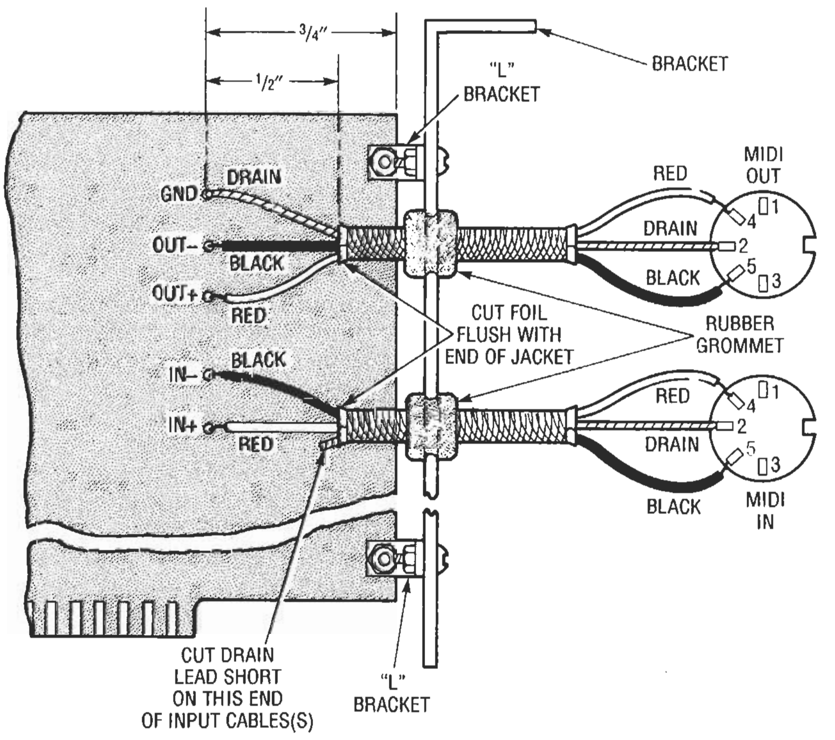


FIG. 2—PARTS-PLACEMENT DIAGRAM. If you've elected to put together the single MIDI in/out configuration, you can leave out IC8, IC10, D2, R8, R12–R14, and R17.



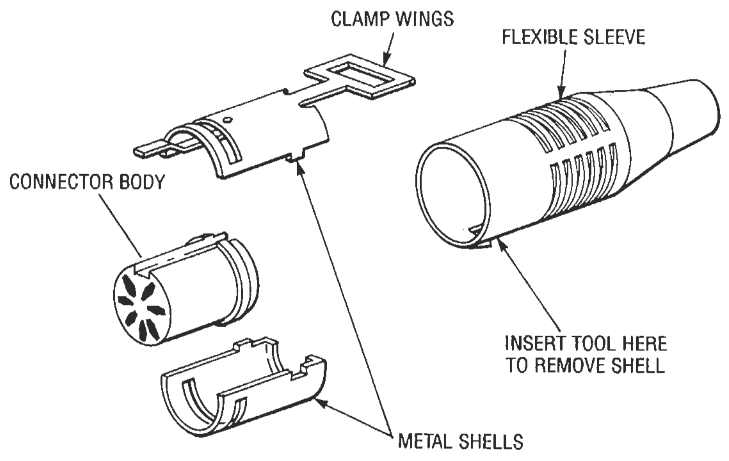


FIG. 3—PC-SLOT ACCESS HOLES aren't wide enough for DIN connectors, so we hang in-line female connectors out on pig-tails. Labeling the connectors will make it easier to tell them apart.

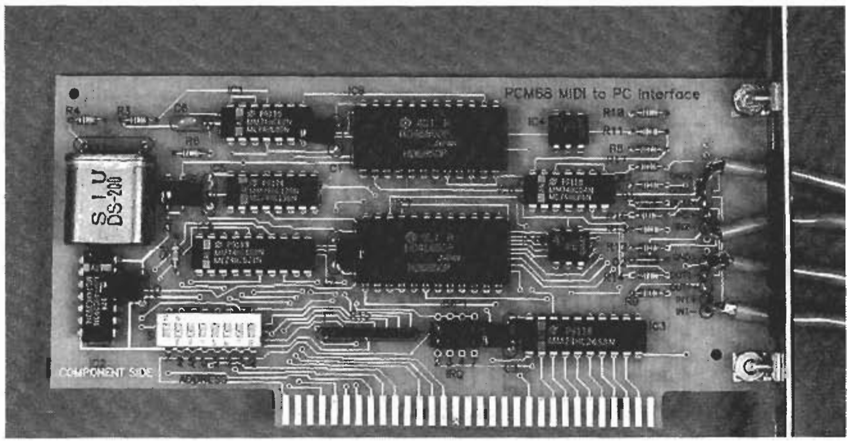


FIG. 4—THE COMPLETED PROTOTYPE CARD. The MIDI jacks are wired directly to the circuit board using shielded twisted pair.

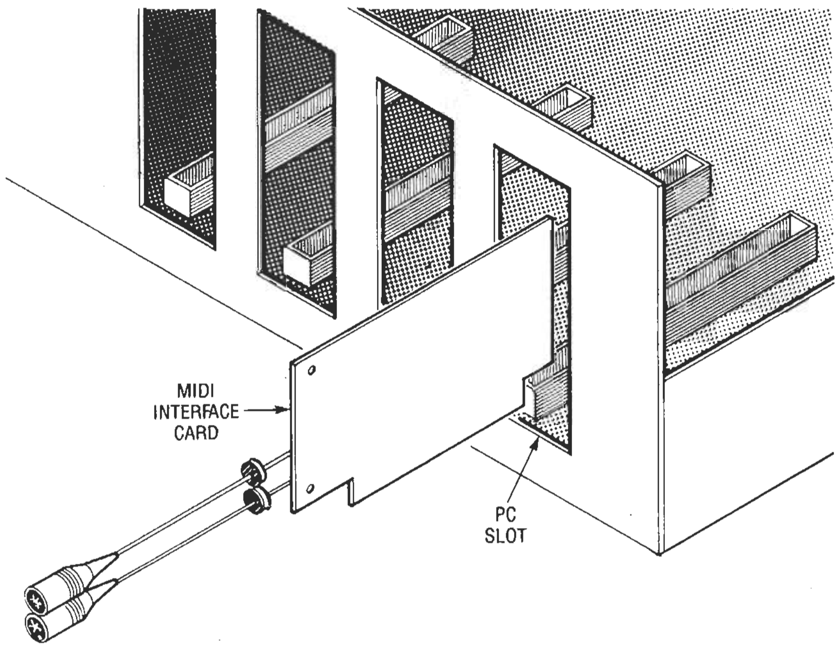


FIG.5—THE CARD FITS through the computer's rear slot.

on many interfaces is simply to hang in-line female connectors out on pig-tails and that's what we'll do here. The circuit board end of those connections can be soldered directly to the circuit board. Use 6-inch lengths of shielded, twisted pair; see Fig. 3. Note that while the shield's drain wire (the wire that connects to the shield itself) is soldered to pin 2 of all the DIN connectors, it connects to the circuit board ground only on the MIDI outputs. Be sure to slide a ¼-inch rubber grommet onto each wire as shown in Fig. 3 before soldering both ends of the wires. It's a good idea to label the connectors, as they are all identical and it will be hard to tell them apart once things are sealed up. Figure 4 shows the completed prototype card.

Installation

When it comes time to install the card in your PC, you'll notice right away that the DIN connectors won't fit through the hole in the back of the case. But by now you've probably also noticed that the interface is somewhat smaller than a usual card, and that's because the card itself is designed to fit through the rear of the PC (see Fig. 5).

A hold-down bracket for the PCM68, shown in Fig. 6, can be fabricated from any material that's handy; ours is bent from 0.040-inch aluminum. After the card is slipped through the access hole in the case, the bracket mounts to the card with two small "L" brackets, and the rubber grommets that you slid on the wires during assembly fit into the notches in the bracket. Finally, push the card down into the selected slot and secure the whole affair with the traditional screw at the top.

Address/interrupt selection

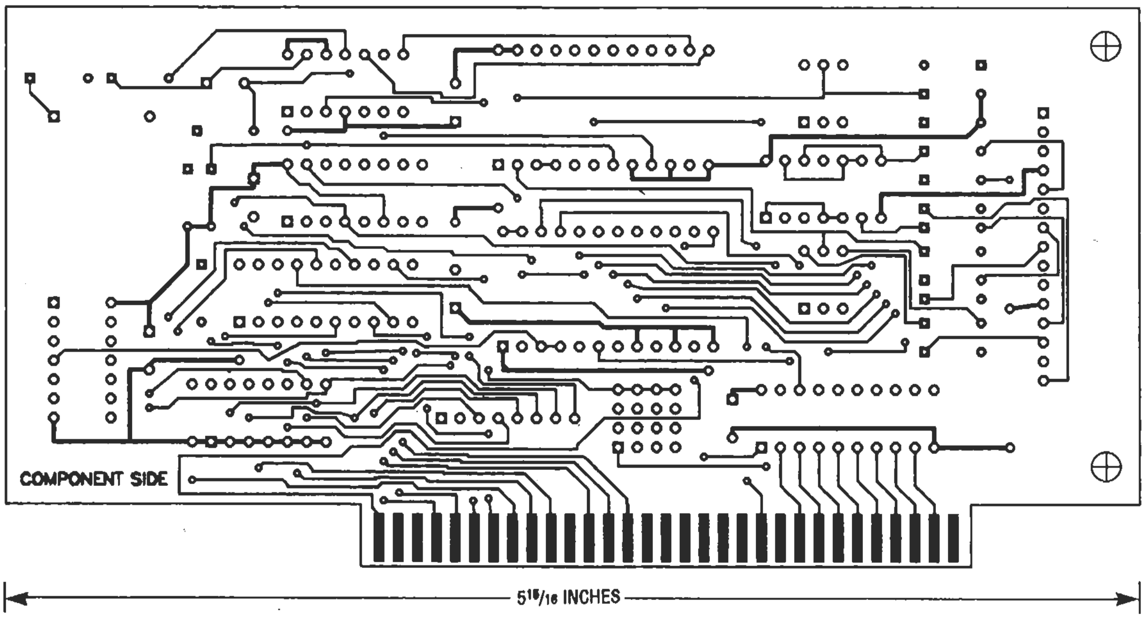
The I/O addresses that the MIDI channels on the PCM68 occupy are selected by DIP-switch S1. It's fairly common for software to default to an address of h330 for the first set of connectors, and Fig. 7 shows the setting of the switches for that situation. If it turns out that there is a conflict, the switches can be set to any address from h0–h3f8. Any software that you select will have some provisions outlined in the owner's manual for changing the port address.

If you've built the interface with two ports, the second pair of inputs and outputs will be at the base address set by S1 plus 1000h. If the first port is at 330h, the second port will be at 1330h.

For interrupts, IRQ2 is normal, and placing a jumper at that location on JU1 will set the card that way (see Fig. 8). If there's a conflict, the jumper can be set to send interrupts to IRQ4 (COM2), IRQ3 (COM1), and IRQ7 (LPT). That's about the order you should try them in if you have to search for one that's unused. Your software will probably default to IRQ2 (or possibly IRQ9 which is re-directed to IRQ2 in AT's) and will definitely have some provisions for changing the default if needed.

Testing

Your software will have a complete test of the interface, but we'll do some simple tests here that will give you a feel for what's going on and confirm that things are working properly. In-



COMPONENT SIDE of the MIDI interface card.

terface tests check to see that data sent from the output side appear at the input side, so the first step is to connect the input to the output with a MIDI jumper cable.

We'll use DOS's handy do-all tool, DEBUG, to directly control the UART and see that it can talk to itself—one of the few cases where that is an indication of sanity. After making sure that DEBUG exists somewhere in the path of your system, invoke it by simply typing "debug" at the DOS prompt. DEBUG responds with its own "-" prompt.

TABLE 1

```
>debug
-o 330 03            This resets the UART
-o 330 15            /16, 8 data bits, 1 stop bit
-o 331 aa            Write hAA to the output data register
-i 331               Read the input data register
AA                   Alright, a response! Now what?
```

A 6850 UART has four internal registers; two read-only and two write-only. When the UART's RS (A0) line is low, a write operation puts data into the Control Register which sets the chip's operating parameters and a read brings back the contents of a Status Register which is information on whether the transmit and receive registers

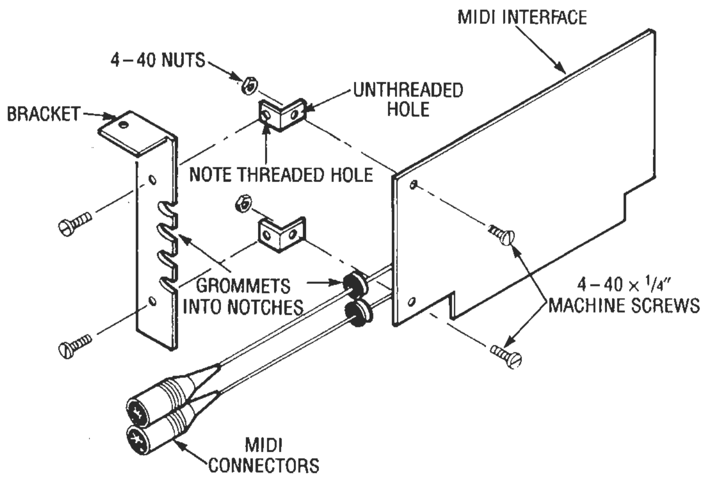


FIG. 6—A HOLD-DOWN BRACKET was fabricated from 0.040-inch aluminum. After the card is slipped through the access hole in the case, the bracket mounts to the card with two small "L" brackets.

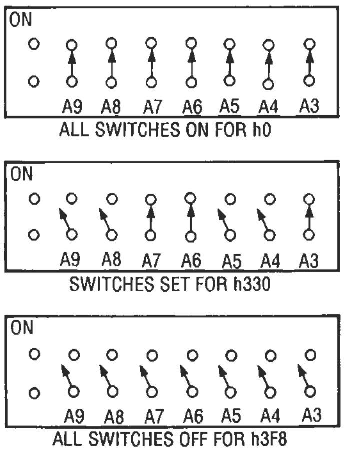
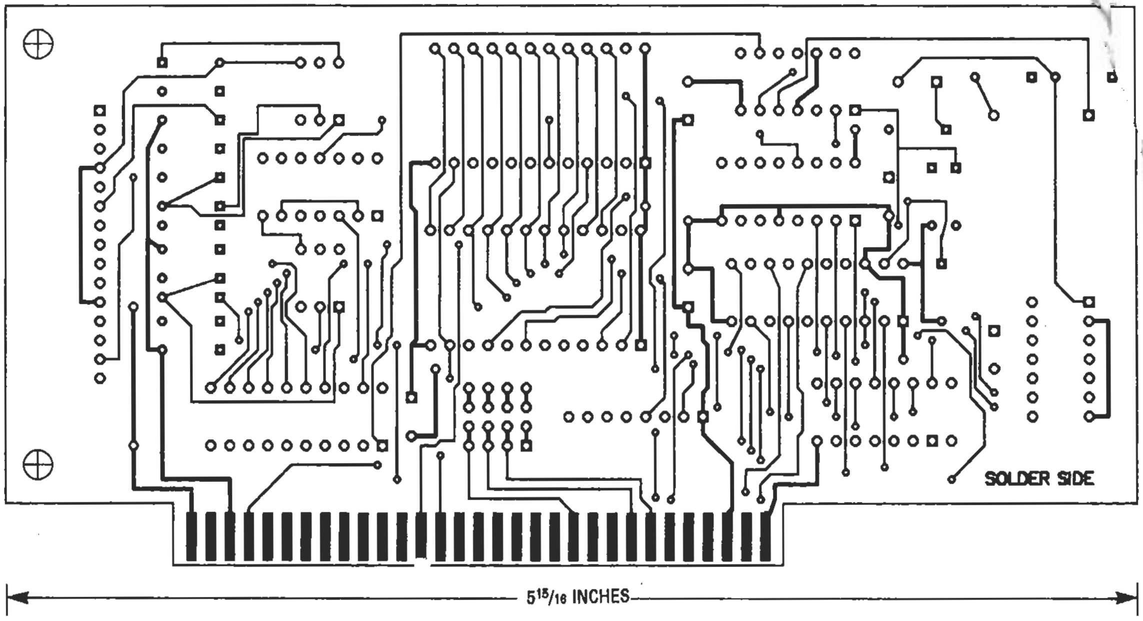


FIG. 7—THE I/O ADDRESSES that the MIDI channels on the PCM68 occupy are selected by DIP-switch S1. Setting the address DIP switches as shown here puts the interface on port h330.



SOLDER SIDE of the MIDI interface card.

are empty or full. One address higher up (RS high) are the Data Registers, and writing to that address sends data out on the serial output, while reading brings back anything which has been received from the serial input.

Unlike some interface chips, a 6850 has no hardware reset line. Instead, a "reset word" is written into the Control Register. The first part of our test will be to do that by entering "o 330 03" from the keyboard. To DEBUG that means to write (output) the datum 03h to the port at address 330h. "03" is the reset word and if you have set up the base address to be other than 330h you will want to change that part of your entry.

The next instruction will be "o 330 15," and that writes a byte to the Control Register, which sets the UART to a mode of 8 data bits and one stop bit and sets the internal frequency divider on the transmit and receive clocks to divide by 16.

Now we output a byte by typing "o 331 aa" which writes the datum hAA to the Data Register of the UART, which in turn sends the data out serially. To see that the data was received, the entry "i 331" reads (inputs) from port h331 and should cause the screen to display "AA." The test process is summarized in Table 1.

The pattern in Table 1 shows how you can write whatever data you like with "o 331 xx" and check to see that the data was received with "i 331." The second port, if you have one, can be checked by writing the reset and setup words to h1330 and writing and reading data to h1331.

Using the interface

After installing your software following its publisher's instructions and fully testing the interface using their tests, you're ready to plug things together and start composing and recording. As you become more involved with MIDI, you'll realize that there are a lot of different ways to hook things up, depending on what you're going to be doing. But the simplest configuration for the beginner is simply to use MIDI patch cords to connect the MIDI output of your keyboard to the MIDI input of the PCM68 and vice versa.

The keyboard that you choose may have its own means of enabling MIDI, such as a slide switch which has a "MIDI" position or something similar. Of course that switch should be set appropriately. More professional instruments might have more exotic capabilities such as re-mapping the keyboard or other controllers onto different MIDI channels, but you'll learn about those things as you go along.

MIDI can be dealt with at a fairly low level for the beginner, yet it offers the capability of becoming as complex as you like. A good place to start learning is "All About MIDI" in the August 1989 issue of **Radio-Electronics**. And for a really well done treatment of MIDI (not only the technical details but also the user side of it), try reading *MIDI for Musicians* by Craig Anderton, published by Amsco Publications. R-E

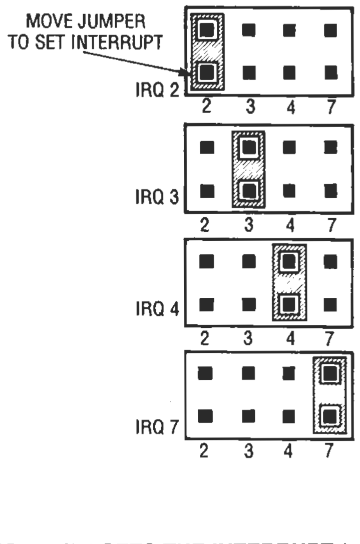


FIG.8—JU1 SETS THE INTERRUPT (see text).