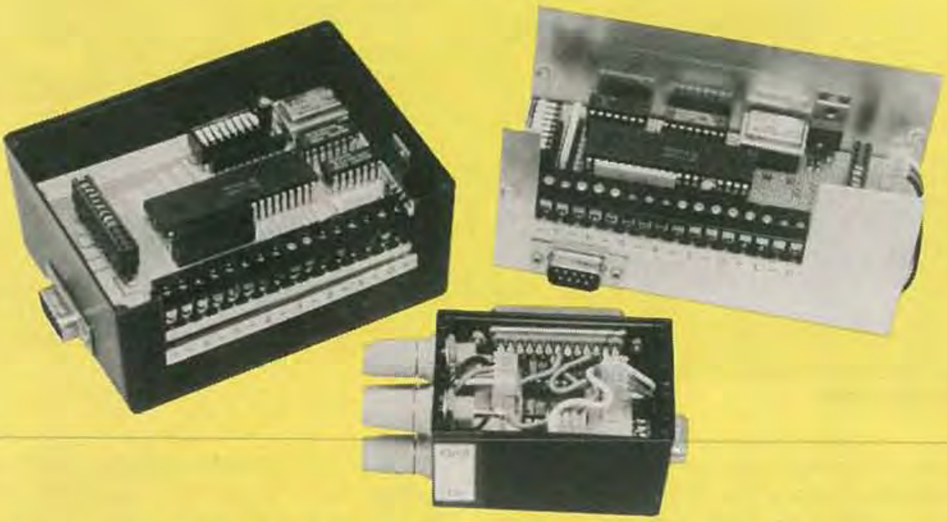
RS-232



MONITOR/CONTROL SYSTEM

Control the environment with your PC and our simple interface.

STEVEN J. FRICKEY

PC's aren't just for word processing, spreadsheet analysis, and database management. In fact, when a PC can collect data from remote locations it can make decisions based on that data, so it becomes a powerful tool for controlling the environment. The problem is that special I/O cards might be required, and they typically cost hundreds of dollars. Also, I/O boards usually require installation within the PC, taking up yet another slot.

In this article we will describe the hardware and software of an I/O control system that can be implemented for less than $50.00, will interface to any personal computer through an RS-232 port, is modular, and has full duplex operation for both input and output.

The heart of the system is a little-known special-purpose IC made by Motorola, the MC14469. The MC14469 is an addressable asynchronous receiver/transmitter that is especially well-suited for remote data collection and control.

The control software is written in Microsoft "C" for the IBM PC and compatibles. Adapting the software to other compilers and computers should be easy.

System overview

Figure 1 shows an overview of the system. It's composed of a PC, control software, a combination RS-232 interface and power-distribution center, and one or more control nodes connected in parallel over a four-conductor bus. The conductors carry power and ground, and the transmit and receive signals.

A control node is shown in Fig. 2. Each node has a unique 7-bit address that is set via DIP-switch S1, which connects to the seven address lines (A0–A6) of IC3.

To communicate with a node, the software on the host PC must first transmit an address byte, over the common receive line (RI). Each node on the bus then compares the received address against its own address, which is set by the DIP switch. If the values match, then that node will accept the control byte that follows

The control byte is latched until a new address and control byte are received by the node. The control and address bytes are distinguished by the value of the most significant bit.

The control data may be used in conjunction with two other MC14469 control signals to direct the activity of the node. The other control signals are Valid Address Pulse (VAP), which is generated after a valid address is detected, and Command Strobe (CS), which is generated after a valid control byte, has been received.

Data transmission back to the host PC is initiated by toggling the SEND input (pin 30) from low to high. The

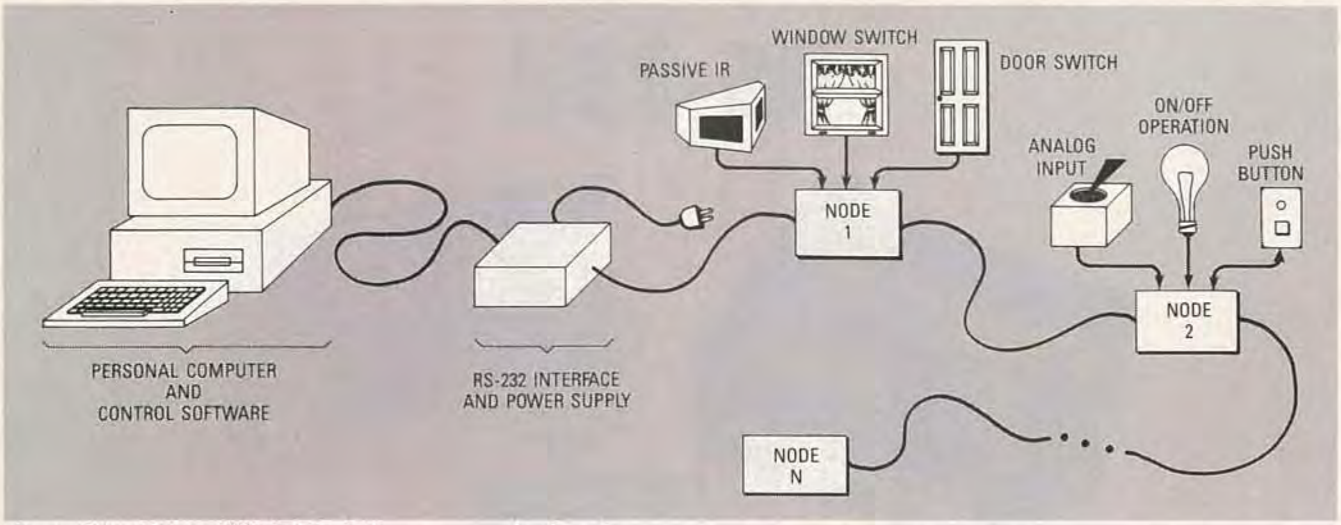


FIG. 1—THE RS-232 INTERFACE buffers communications between the host PC and all nodes.

data that is present on the sixteen input pins (ID0–ID7 and S0–S7) will be transmitted back to the host after SEND is toggled. Data is sent one byte at a time; we'll discuss the details shortly.

After receiving the data from the selected node, the host software could compare that value against the previous value from the same node, perform some action based on the comparison, and then continue on, polling the next node.

By creating an appropriate interface between external devices and any given node, the software can be tailored to a number of monitor and control situations. For example, a number of inputs could be connected to door and window switches. If one of those switches were opened before a master switch, an alarm might be sounded.

Node circuit

Connector J3 provides eight pulled-up input lines (S0–S7) that may be driven by reed switches, pushbuttons, mercury switches, tilt switches, relays, and other mechanical-switching devices. That connector also provides eight ground lines for attaching lead wires.

Connector J4 provides access to the seven output-control lines (C0–C6) of the MC14469, eight more input lines (ID0–ID7), and various control signals. To use the node in its basic configuration, jumpers should be installed across pins 9 and 10, 11 and 12, and so on, through pins 25 and 26. Later on, we'll show how J4 can be used to interface an 8-bit A/D converter to a node.

As shown in Fig. 2, the four-conductor bus runs straight through each node from J1 to J2. One line is for ground, another for +12-volts DC, one for the common transmit line (TRO), and another for the common receive line (RI). The overall length of the bus (from the RS-232 interface to the last node on the bus) depends on the degree of electrical noise in the operating environment. The author has successfully operated three nodes, using 20-gauge unshielded cable, at a cumulative length of 200 feet.

With a seven-bit address, the possible number of nodes in a system is 127, but that is not a practical limit. Realistically, the number of nodes is limited by the amount of current supplied by the +12-volt power source. Each node (with no expansion circuitry) draws 50 mA.

IC4, a 7805 voltage regulator, drops the +12-volt bus voltage to +5 volts for powering the logic circuitry. Because the +12-volt line is also available at J4, you can attach off-the-shelf alarm-system components, such as passive infrared detectors, buzzers, beepers, etc.; all of which typically operate at +12 volts.

The MC14469's baud-rate clock can be generated internally across pins 1 and 2, or an external clock can be fed directly to pin 1. The maximum baud rate (4800) is re-

NODE-CIRCUIT PARTS LIST

IC1—74ALS161, synchronous 4-bit counter
IC2—74ALS05, open-collector hex inverter
IC3—MC14469, addressable asynchronous receiver/transmitter
IC4—7805, 5-volt regulator
IC5—4-MHz TTL crystal oscillator
RP1, RP2—4700 ohms, 10-pin SIP
S1—8-position DIP switch.
C1—0.33 µF, 12 volts, tantalum
C2—22 µF, 25 volts, electrolytic
C3—0.1 µF, monolithic ceramic
J1—9-pin D, female
J2—9-pin D, male
J3—16-pin, PC-mount, screw terminal block
J4—26-pin dual-row header strip

RS-232 INTERFACE PARTS LIST

IC1—1488, quad RS-232 line driver
IC2—1489, quad RS-232 line receiver
R1—4700 ohms, ¼ watt, 10%
J1—25-pin D, male
J2—9-pin D, male
J3—4-pin power connector
Miscellaneous: Power supply with ±12- and +5-volt outputs, cases, interconnecting cables, etc.
Note: A double-sided PC board with plated-through holes is available for $16, the MC14469 is available for $15 (including spec sheet and application note), and a floppy with the source code is $5 from Steven J. Frickey, 3661 N. Lena, Boise, ID 83704. All orders add $2 for shipping and handling. Also, the author has also developed control software for the Amiga; contact him at the address above for more information.

stricted by the +5-volt supply. The required clock rate is 64 times the baud rate, or in this case, 307.2 kHz.

Because that's a non-standard frequency, the circuit uses a readily available 4-MHz TTL clock oscillator (IC5), a 74ALS05 (IC2) open-collector inverter, and a 74ALS161 (IC1) four-bit counter to divide the 4-MHz signal by 13, thereby providing a 307.69-kHz signal. The communications protocol is fixed at one start bit, eight data bits, an even parity bit, and one stop bit. So at 307.69 kHz, the maximum sampling time error over the entire 11 bits is 35.7 μs, well within one-half of a data bit period, which is 104 μs at 4800 baud.

A second gate on the 74ALS05 (IC2-d) inverts the serial data from IC3 and drives the common transmit line (TRO). The pull-up resistor for IC2-d is actually located in the RS-232 interface circuit (shown in Fig. 3 as R1). The open-collector outputs of all nodes are pulled up by that resistor, which makes it a wired-OR circuit.

A local reset is generated by each node at power up by an RC circuit consisting of 22-μF capacitor C2 and a 4.7K resistor inside RP1. The reset signal is also provided at J4, should your expansion circuitry require access to that signal.

The 7-bit address for each node is set on pins 4 through 10 (A0–A6) of IC3. Table 1 shows the relationship between switch settings and node numbers.

The voltage supplied to IC3 can range from 4.5–8.0 volts. At five volts, the output drive current of each pin (I_{OH}) is typically 0.35 mA, providing a fan-out capacity of 17 ALS devices. The output-high voltage (V_{OH}) is typically 5.0; the low voltage (V_{OL}) is typically 0.0. The input high voltage is typically 2.75; the input-low voltage is typically 2.25. For more information on the MC14469, consult *CMOS/NMOS Special Functions Data,* Motorola Inc., 1984, and Application Note AN806A, *Operation Of The MC14469,* Motorola Inc., 1984.

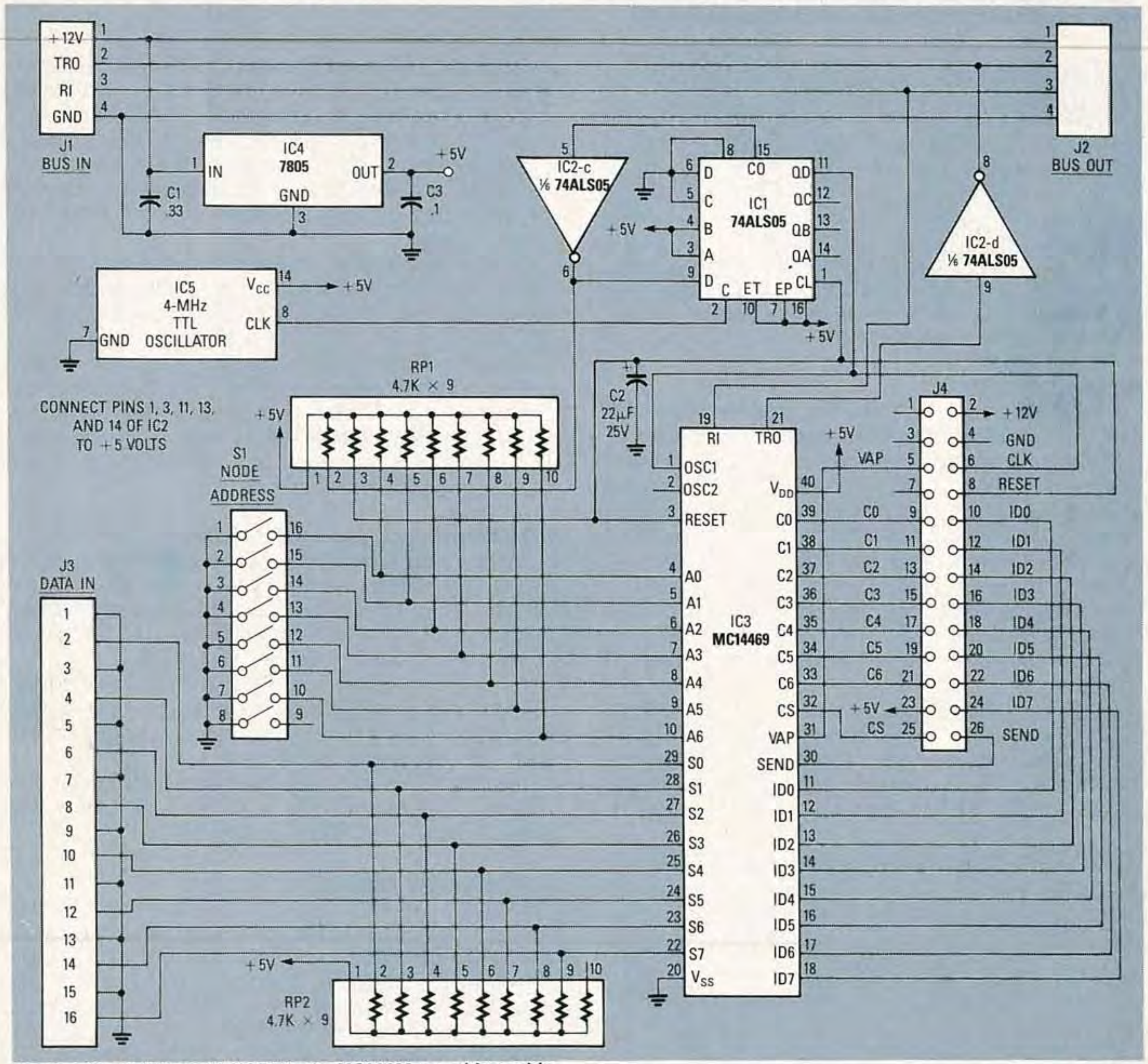


FIG. 2—THE HEART OF A NODE is the MC14469, an addressable UART. When a node is addressed, data present on pins 11–18 and 22–29 is transmitted to the host. The address is set on pins 4–10.

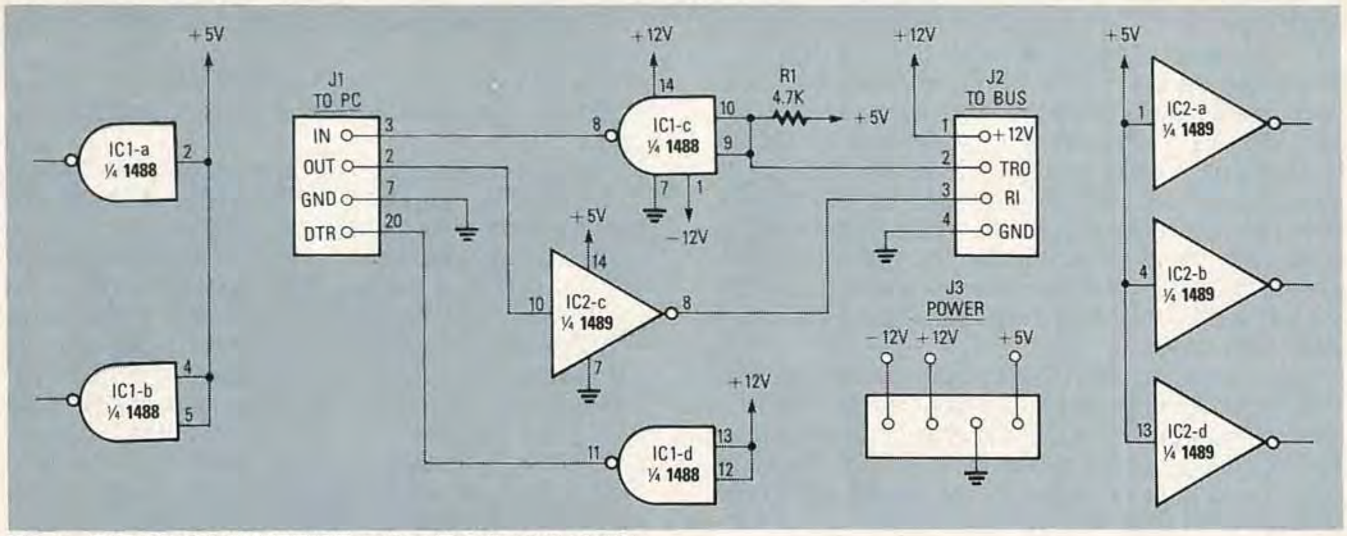


FIG. 3—THE RS-232 INTERFACE routes 12-volt power to the nodes, and buffers data between the nodes and the host PC.

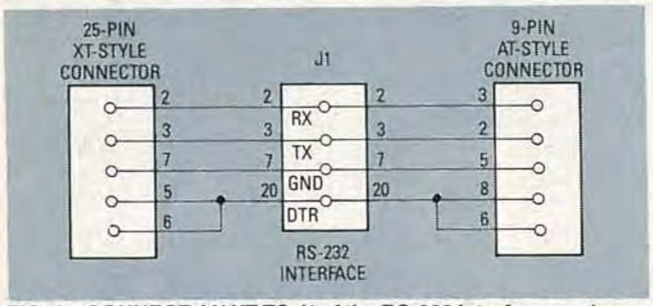


FIG. 4—CONNECT AN XT TO J1 of the RS-232 interface as shown at left, and to an AT as shown at right.

Node operation

The communication software first transmits a seven-bit address that is received simultaneously on pin 19 (RI) of all MC14469's in the system. Each node then checks the state of the most significant bit. If it's high, then the remaining seven bits are compared against the address set on A0–A6. If the values are identical, then VAP is generated on pin 31. VAP is not used in the node circuit shown in Fig. 2, but it is used internally by the MC14496 to latch a control byte on output pins 33–39 (C0–C6). Control-byte data is latched only after a valid address has been received, and it remains latched until another address byte is received.

Transmitting data back to the host PC is accomplished by toggling pin 30 (SEND) high. After receiving the SEND pulse, the MC14469 will transmit, via pin 21 (TRO), the data present on pins 11–18 (ID0–ID7), followed by the data on pins 22–29 (S7–S0). The only stipulation is that the rising edge of the SEND pulse must occur within eight bit times after the generation of either VAP or CS. At 4800 baud, eight bit times provides a maximum of 1.667 ms.

Receipt of a control byte generates a Control Strobe (CS) pulse on pin 32. In our circuit, CS is normally connected to SEND through J4. In this configuration, data will be transmitted to the host as soon as a control byte has been received.

What is the minimum interval between events that this system can detect? The time it takes to transmit and receive data from the same node twice, which works out to (1/4800) × 11 bits/byte × bytes, or about 18 ms.

Realistically, the minimum time is much longer, at least on the order of hundreds of milliseconds, because of the amount of time the software processing takes, especially when relatively slow I/O devices (disk, BIOS video routines, printer) are being accessed. Just don't try to detect more than three events per second.

TABLE 1—NODE ADDRESSES

Node	A6	A5	A4	A3	A2	A1	A0
0	H	H	H	H	H	H	H
1	H	H	H	H	H	H	L
2	H	H	H	H	H	L	H
3	H	H	H	H	H	L	L
. . .							
125	L	L	L	L	L	H	L
126	L	L	L	L	L	L	H
127	L	L	L	L	L	L	L

RS-232 interface and power supply

Figure 3 is the schematic for the RS-232 interface, which uses a 1489 (IC2) for the line receiver and a 1488 (IC1) for the line driver. Pin 2 of J2 is the common transmit line (TRO) that receives data from the open-collector output of each node, and R1 is the pull-up resistor.

Power is supplied to the system via four-pin connector J3. As stated earlier, a single node draws about 50 mA from the +12-volt supply. Low-current sources of +5 and −12 volts are also required.

Figure 4 shows the cable wiring required to connect the RS-232 interface to a 25-pin XT-style port (on the left), and to a 9-pin AT-style port (on the right).

Assembly and testing

Figures 5-*a* and 5-*b* show how to mount the components on the PC boards. The Node board, shown in Fig. 5-*a*, is a double-sided board. You can use the patterns shown in PC Service to build your own, or you can purchase the board from the source mentioned in the Parts List. The pattern for the RS-232 board is also shown in PC Service, but because it is so simple, a commercial product has not been made available.

After you assemble the system, test it using the sample

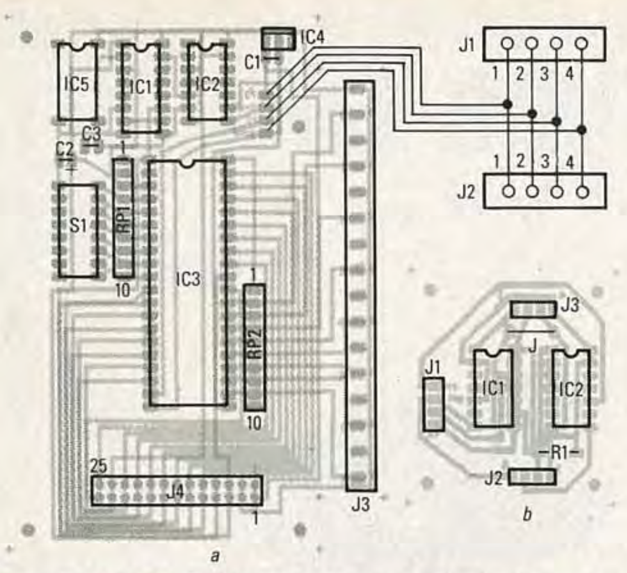


FIG. 5—MOUNT ALL COMPONENTS on the node circuit board as shown in 5-*a* and mount all components on the RS-232 circuit board as shown in 5-*b*.

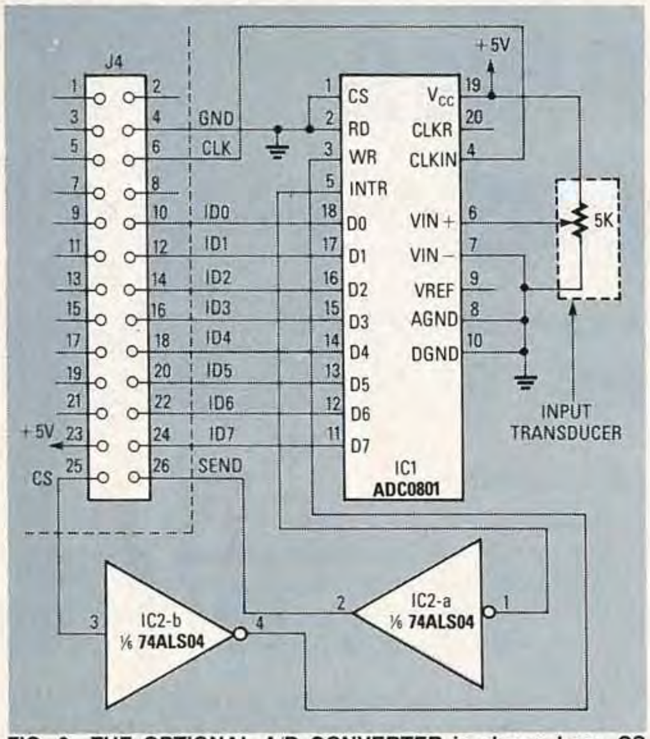


FIG. 6—THE OPTIONAL A/D CONVERTER is shown here. CS from the Node board starts the conversion process; INTR from IC1 here informs the Node that the process is complete.

program that will be discussed shortly. Apply ±12- and +5-volts DC to the RS-232 module, and connect it to your PC and to a single node configured as address 0. Then run the test program. If you receive any error messages (especially a time-out error), check your cabling carefully—the chances are that the RS-232 module hs not been connected to your PC properly.

When the software seems to be running correctly, temporarily short several of J3's even-numbered pins to ground, one at a time. Then terminate the test program according to the directions given on the screen. An ASCII text file called MONITOR.LOG should be present in your current directory. That file should contain a number of messages corresponding to the state of the input lines of J3 at startup, and it should also include messages indicating that it sensed the shorts.

A/D expansion example

Figure 6 shows how to interface an eight-bit analog-to-digital converter (the ADC0801) to a node via connector J4. The component labeled Input Transducer is shown as a 5K potentiometer, but in real life it might be a temperature sensor, a pressure sensor, etc.

In this circuit, CS initiates the analog-to-digital conversion (WR), and the end-of-conversion (INTR) pulse from the ADC initiates data transmission to the host by toggling the MC14469's SEND input.

The ADC uses the 307-kHz node clock. At that rate, a single conversion will take at most 240 μs, which is well within the 1.667-ms time limit between the CS and the SEND pulses.

The software

Because of space limitations, we are unable to print the 600-line C source listing here. However, we will give an overview of how the software works. In addition, both executable files and the full source code have been posted on the RE-BBS (516-293-2283). Download file RS232MON.ARC at 300 or 1200 baud, eight data bits, one stop bit, and no parity. (Source and executable files of an additional program that demonstrates use of the A/D converter is also included.)

The program is a simple event-logging system that continually polls a single node, logs the date, the time, and the input device(s) that changed state since the last time that node was polled. Execute it by typing the name of the program followed by the number of the serial port being used (0 = COM1, 1 = COM2, etc.).

The program communicates directly with the serial port through BIOS interrupt 14h. That means the program can reconfigure the port-communications protocol, read a byte, write a byte, and check the status of the port. Several error conditions can also be determined when using the interrupt. If an error does occur during execution, the program stops and a message is displayed on the screen indicating the type of error.

In the program, each node is represented by a data structure that contains the node address, the initial value of the control byte, a mask value indicating which bit values to respond to, a copy of the last data values returned from the node, and sixteen other fields that correspond to the bit values returned from a node. The sixteen fields contain names that identify what a bit represents, what its *on* state is, and what its *off* state is.

When the program starts, each record is accessed sequentially, and the corresponding node address and control byte are sent. For each node, required functions are initialized, communications checked, and initial conditions logged in a disk file called MONITOR.LOG.

After initialization the program begins to loop, sequentially polling each node and checking the return values against the previous values from the same node. If a new value is different from a previous value, and if those particular bits that indicate a difference are not masked, then the event is logged in the log file with the date and time. Polling continues in that way until the user terminates the program by pressing a key.