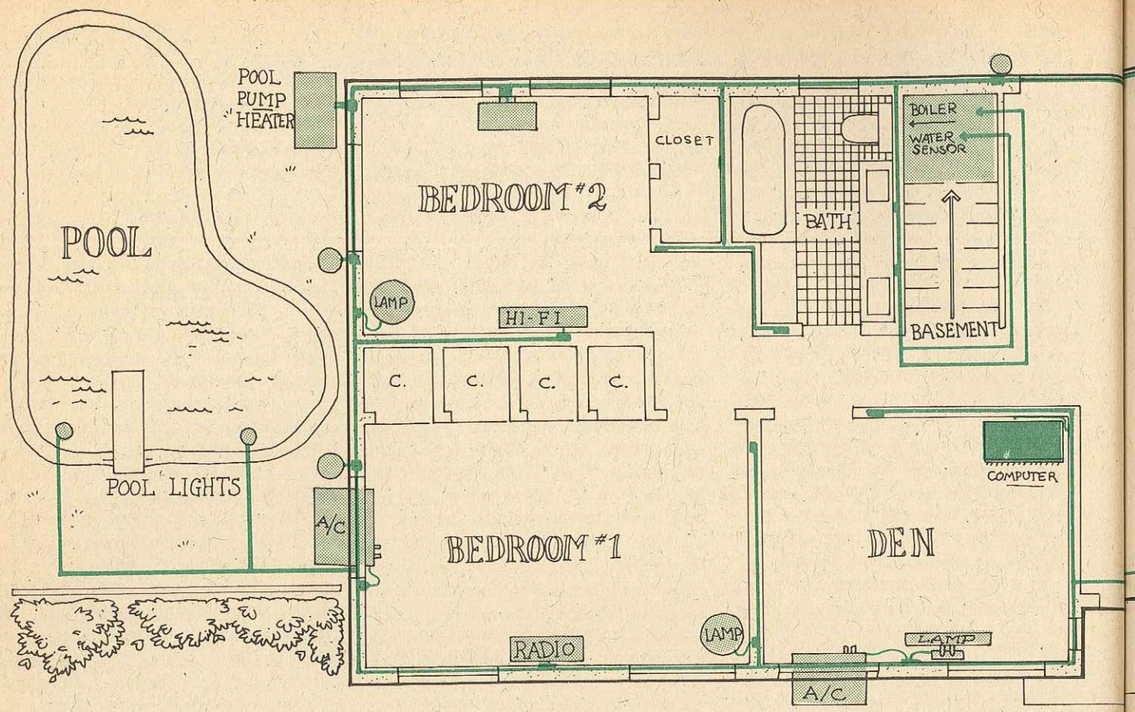


Using Existing House Wiring

BY DAN SOKOL, GARY MUHONEN, AND JOEL MILLER

SOME HOBBYISTS with their own computers at home, use them to play sophisticated games. Others use them for "number crunching." Still others use them simply to learn more about working with microcomputers. Where many computer owners fail to make use of their machines is in the control of electrical appliances in their homes. With the recent introduction of several "controller" boards, in which the computer can activate a power switch, such as a relay or SCR, under program control, the computer's role in the home will undoubtedly change. However, there still remains the frustrating task of wiring the output lines of the computer to the controlled appliances in other rooms.

The Intelligent Remote Controller described here makes room-to-room control wiring a relatively simple matter. With a controller board plugged into any Altair 8800/S-100 bus system, a special ac adapter is connected to the controller board and plugged into the ac line. Commands given by the computer program are sent via the controller to the ac adapter, which impresses the digital waveform on the ac line at the wall receptacle. Hence, instead of running cable all through your house, you simply take advantage of the already existing house wiring to route signals to various remote appliances.

Special dual-channel remotes, which can be connected to any wall outlet in the premises for both power and reception of the digital control signals, are used for the actual power control. Each remote has two conventional, separately controlled, ac sockets that can accommodate any electrical appliance rated at 500 watts or less. Of course, the output circuits can be modified to handle higher-power appliances.

The remotes (up to 64 with this system) constantly monitor the ac line for commands intended for them. When a command for a particular remote is detected, it controls one or both of the appliances plugged into it, turning on or off the power. The remote then "reports" back to the controller on the status of the selected device.

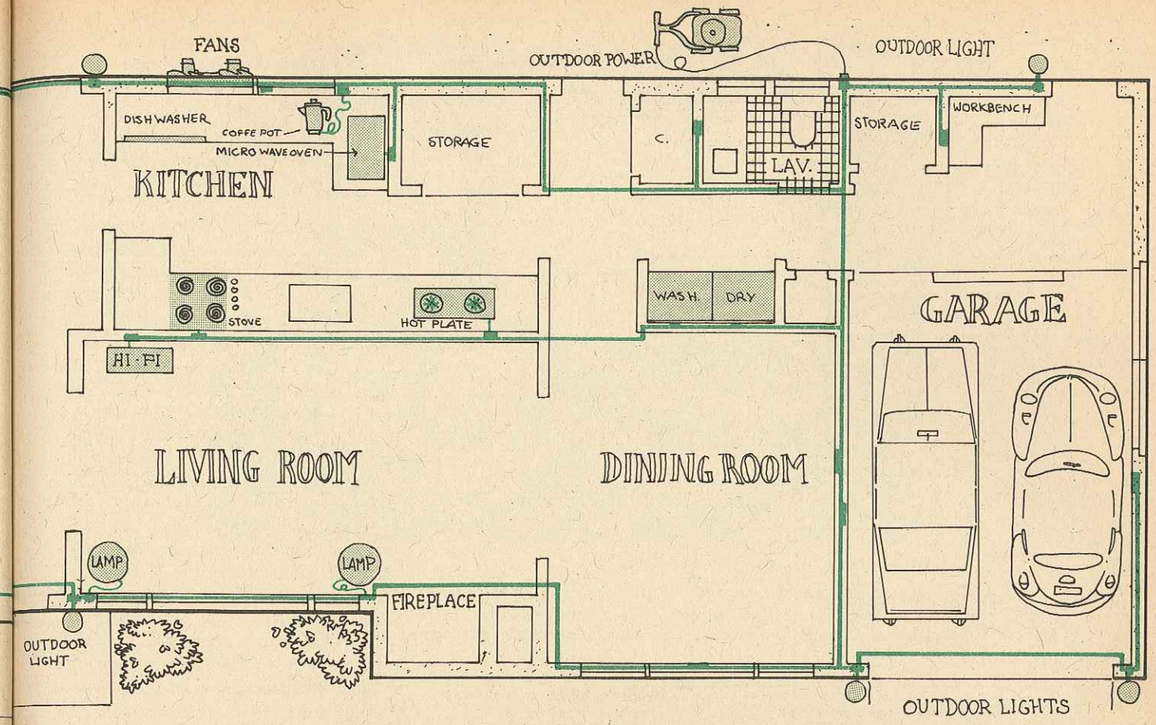
In this first article, we cover the controller circuit (Fig. 1). There are two major sections in this device—the controller itself and a self-calibration circuit that is used for setting up the remotes.

The power supply for the controller board is shown in Fig. 2. It is typical of most computer bus boards.

About the Circuit. The controller occupies three I/O ports. These ports are assigned by board jumpers, with the input and output sharing the same address and the status port being the input port minus one.

In this bidirectional system, the user can "poll" each remote to determine its status. Two ports are used for both writing data to and receiving data from the remotes. Since decoding circuitry is built into each remote, up to 64 remotes can be controlled by the system.

The filter/amplifier/limiter circuit that is shown in Fig. 3 accepts an input from the ac adapter, passes only that portion of the signal above 20 kHz, and conditions it for use by the following data-



for Computer Remote Control

PART I

recovery PLL (phase-locked loop). Although the amplifier's gain is set at 5, its output is diode-clipped at 0.7 volt to prevent the PLL from being overdriven and eliminate false triggering. Transistor *Q4* acts as a switch that shuts down the amplifier during calibration and during the transmission of data.

The data-recovery and clock-generator circuit consisting of *IC4, IC6,* and *IC8* recovers the transmitted data and generates the transmit frequency. When data is present, the locked output from pin 8 of the PLL outputs the data, which is sent to the UART receiver.

To generate the transmit frequency, the output of the free-running vco in the PLL is buffered by *IC6* and used as the transmit frequency by AND'ing it with the UART data before the data is sent to the ac line. In addition, this frequency is divided by 16 by *IC8* to generate the clock for the UART and the reference clock for the self-calibration circuit.

During the receive cycle, UART *IC18* is clocked by the frequency of the vco so that the vco in the receiver locks onto

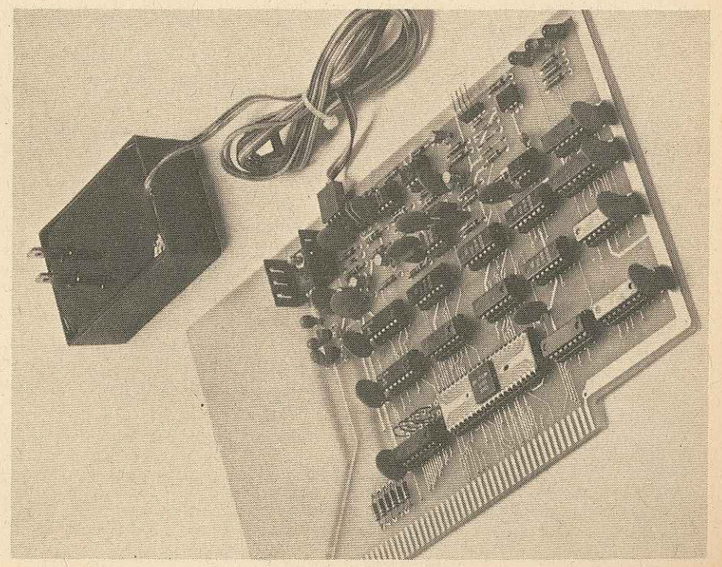


Photo of prototype controller board with adapter plug into ac line.

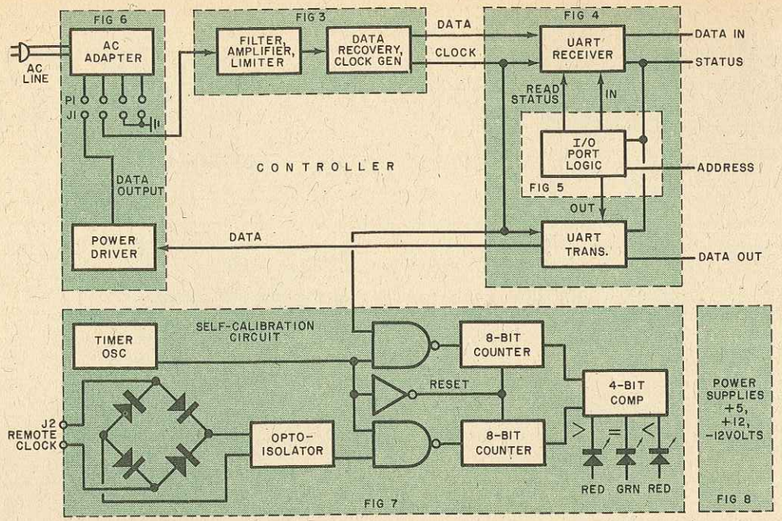


Fig. 1. Overall logic of controller that also includes self-calibration circuit for remote oscillator alignment. Sections of the controller are referred to by figure numbers in which complete schematics are given.

PARTS LIST

C1 through C5, C10 through C14,C17,C18, C19,C25,C26—0.1-μF disc capacitor
C6—0.1-μF disc capacitor
C7—0.0056-μF disc capacitor
C8—0.39-μF disc capacitor
C9—0.01-μF capacitor
C15,C16—0.001-μ F disc capacitor
C20,C21—15-μF, 15-V tantalum capacitor
C22,C23—10-μF, 25-V tantalum capacitor
C24—1-μF, 35-V tantalum capacitor
C27—.1-μF capacitor
D1 through D7—1N4148 diode
*IC1,IC2—NE535V op amp
IC3—MCT-2 optoisolator
IC4—567 phase-locked loop
IC5—555 timer
IC6,IC13—74LS04 hex inverter
IC7,IC12—74LS32 quad 2-input OR gate
IC8,IC9,IC10,IC15,IC16—74LS93 4-bit counter
IC11—74LS85 4-bit magnitude comparator
IC14—74LS132 quad 2-input NAND Schmitt trigger
IC17—8131 6-bit comparator
IC18—TR1802 UART
IC19,IC20—74367 tri-state hex buffer
J1,J2—4-pin right-angle jack (Molex)
LED1,LED3—Discrete red LED
LED2—Discrete green light-emitting diode.
Q1,Q2,Q4—2N2907 transistor
Q3,Q5—2N2222 transistor
The following resistors are ¼-W, 10% tolerance:
R1 through R7,R9,R19—2200 ohms
R8,R12,R13,R18,R29—1000 ohms
R10—390 ohms
R11—10 ohms
R14,R15,R22 through R25, R34—3300 ohms
R16—3900 ohms
R17—15,000 ohms
R20,R21—10,000 ohms
R26,R30,R31,R32—100 ohms
R27—4700 ohms
R28—100,000 ohms
R33—27,000 ohms
RV1,RV2—V33MA1A voltage regulator (GE)
VR1—7805 5-volt regulator IC
VR2—79L12 12-volt regulator IC
VR3—78L12 12-volt regulator IC
Misc.—Printed circuit board; sockets for IC's; heat sink and mounting hardware for VR1; interface adapter No. ACD-1; wire; etc.
Note: The following is available from Mountain Hardware, Inc., P.O. Box 1133, Ben Lomond, CA 95005 (Tel.: 408-336-2495): complete controller kit, including ac interface module for $149.
*IC's are identified by letter "U" in parts placement guide in Fig. 8.

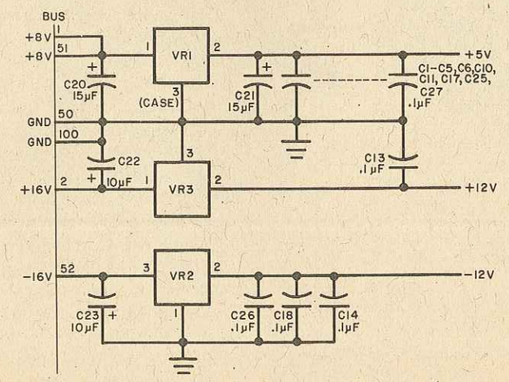


Fig. 2. Schematic of controller power supply.

the frequency of the vco in the transmitter and provides a stable source of the same frequency to the UART. This eliminates the need for expensive crystal oscillators and divider circuits.

The heart of the controller is the UART, shown schematically in Fig. 4. This circuit receives, transmits, and formats data that is sent between the computer and controller. The controller and each remote have their own UART's. Since the UART outputs are tri-state, both the status and the data information can be AND'ed to the same bus.

When power is first applied, the UART is reset by the POC (power-on clear) signal on bus connector 99 after passing

through inverter *IC13*. The UART can be programmed to deal with 5-, 6-, 7-, or 8-bit words, can be set for odd or even or no parity, or the number of stop bits can be set to 1, 1½, or 2. In this circuit, the UART is set for eight data bits, odd parity, and two stop bits.

The transmitter portion of the UART removes the parallel data from the bus and transmits it serially to the power driver circuit. When transmitting a signal, TEOC (transmit end of character) from *IC18*, pin 24's signal is inverted and used to disable amplifier *Q5* in Fig. 3 to stabilize the vco.

The receiver portion of the UART accepts serial data from the PLL, converts it into parallel data, and checks for possible errors. The parallel output of the UART receiver is passed to the bus via tri-state buffers *IC19* and *IC20*.

The receiver section constantly checks its serial input line for a start bit, defined as a mark-to-space transition. When it receives this signal, it waits for a period of time equal to a half-bit period. Then it checks to see if the space is still there to determine if it is a valid start bit. If the start bit is not valid, the UART resumes searching. If the bit is valid, the next 10 bits are clocked into an internal shift register. The start and parity bits are removed before transferring the 8-bit data word to the output holding register. Finally, the UART sets a status flag when readout data is available and when an error is detected.

The three error flags are: receive parity error, receive framing error, and receive overrun error. A receive parity error bit of 1 indicates that the data word in the holding register was received with a parity error. If the receive framing error bit is a 1, the word in the register did not have the correct number of bits. If the receive overrun error is a 1, the new word has overwritten the word previously stored in the register before the old word was read out, indicating that this word has been lost.

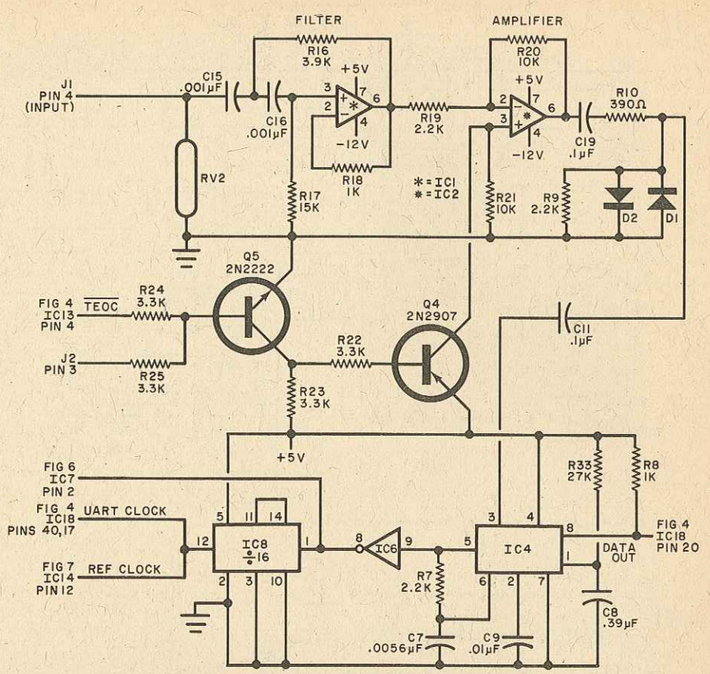


Fig. 3. Filter, amplifier, limiter, and data and clock recovery.

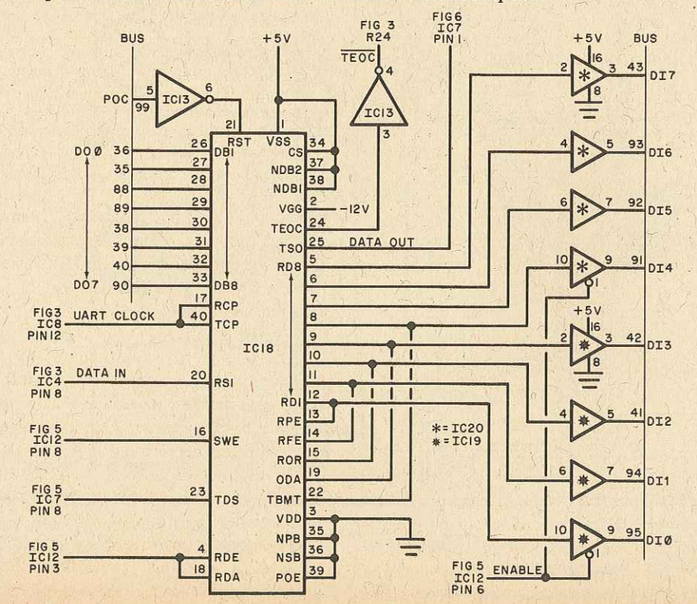


Fig. 4. UART connection betwen controller and computer.

The I/O port decoder shown in Fig. 5 determines if the computer is communicating with the controller and prepares the controller for transmitting or receiving data. The output of this circuit causes the controller to place data on or read data from the computer bus.

The circuit acknowledges three commands internal to the controller: read status, read the UART receiver's holding register into register A of the computer, and transfer register A data into the UART buffer and begin the transmit cycle. These internal commands are related to system software commands IN and OUT (the assembly language mnemonics for communicating between the computer and controller). Integrated circuit *IC17* and its associated logic determine the I/O port selection, while the remaining integrated circuits in Fig. 5 decode the command from the computer controller.

The power driver, shown schematically in Fig. 6, provides sufficient drive for

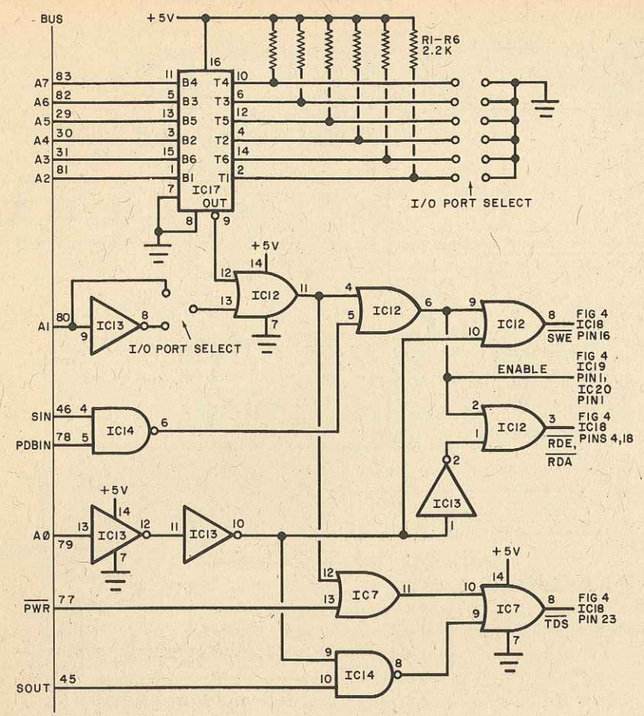


Fig. 5. The I/O port selection is made by choosing the jumper arrangement for the selected port.

the signal to ensure adequate reception at the remote receiver. The driver AND's the transmit frequency with the UART transmitter's serial output (TSO). The circuit then converts the TTL-level input signal into ±15-volt levels. The resulting signal is entered into the ac line via the ac interface adapter, which consists of a package that contains three capacitors and a tuned transformer that is resonant at 50 kHz. The adapter is connected to the controller via a four-conductor cable to connector *J1*.

For the system to function properly, the free-running vco frequencies must be within 4% of each other. If they are not, receiver overrun errors result in incorrect data. The self-calibration circuit shown in Fig. 7 is used to adjust the remote vco. The vco in the controller is not adjustable; it is used as the "reference" for the system. The self-calibration circuit visually indicates whether the remote vco is running faster, slower, or at the same rate as the controller's vco. This circuit also eliminates the need for a relatively expensive frequency counter to check both oscillator frequencies.

The UART clock on the controller board is used as the reference frequency, and the UART clock from the remote is connected to the controller via *J2*. The remote is coupled through optical isolator *IC3* to keep any line voltage from appearing on the controller board.

The signal from *IC3* is shaped and gated by *IC14* and then passed to counter *IC15*. Free-running oscillator *IC5* provides control for *IC14* and generates the reset pulse for the counters. This oscillator is set for a 1% duty cycle and provides a "window" to enable the reference clock (and its equivalent from the remote) in two eight-bit counters (*IC9, IC10, IC15,* and *IC16*). The counters are arranged as two eight-bit counter chains, and the long period of 99% of the *IC5* output is the window that allows the counter to operate, while the short 1% period pulse resets both chains. The four most significant bits from each counter are compared in four-bit comparator *IC11*.

The outputs of the comparator are inverted and buffered by portions of *IC6* and are used to drive three LED's. On the "less than" or "greater than" outputs, red-colored *LED1* and *LED3* glow. When the output is "equals," green-colored *LED2* glows.

During calibration (described in Part II), a cable is connected between the controller and remote. It disables the analog sections and provides a signal path between the two boards. The analog section must be disabled to remove jitter from the vco's.

As the vco control potentiometer on the remote vco is adjusted, the period of time that the green LED glows becomes longer and longer, indicating that the two vco's are running at the same frequency. The two LED's indicate in which direction the remote vco differs (less or greater than) from the controller vco.

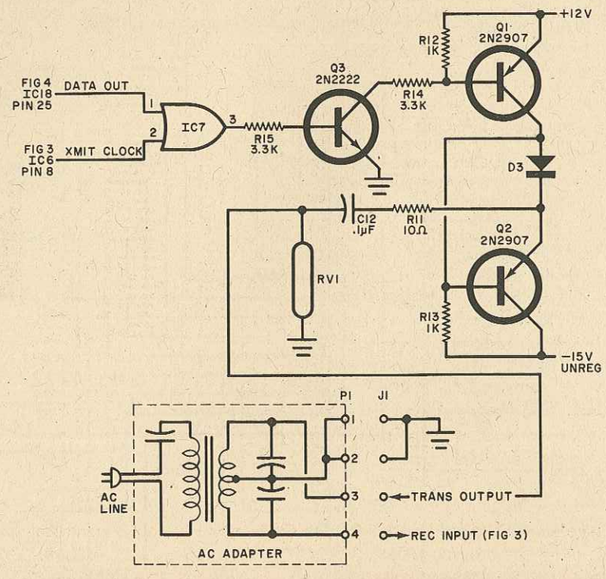


Fig. 6. Power driver accepts UART data and clock and delivers high-level signal to ac adapter.

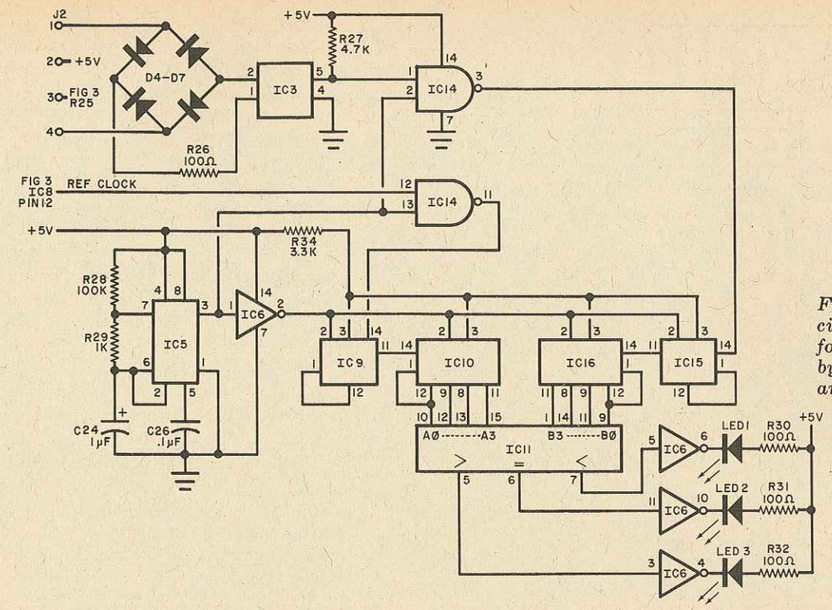
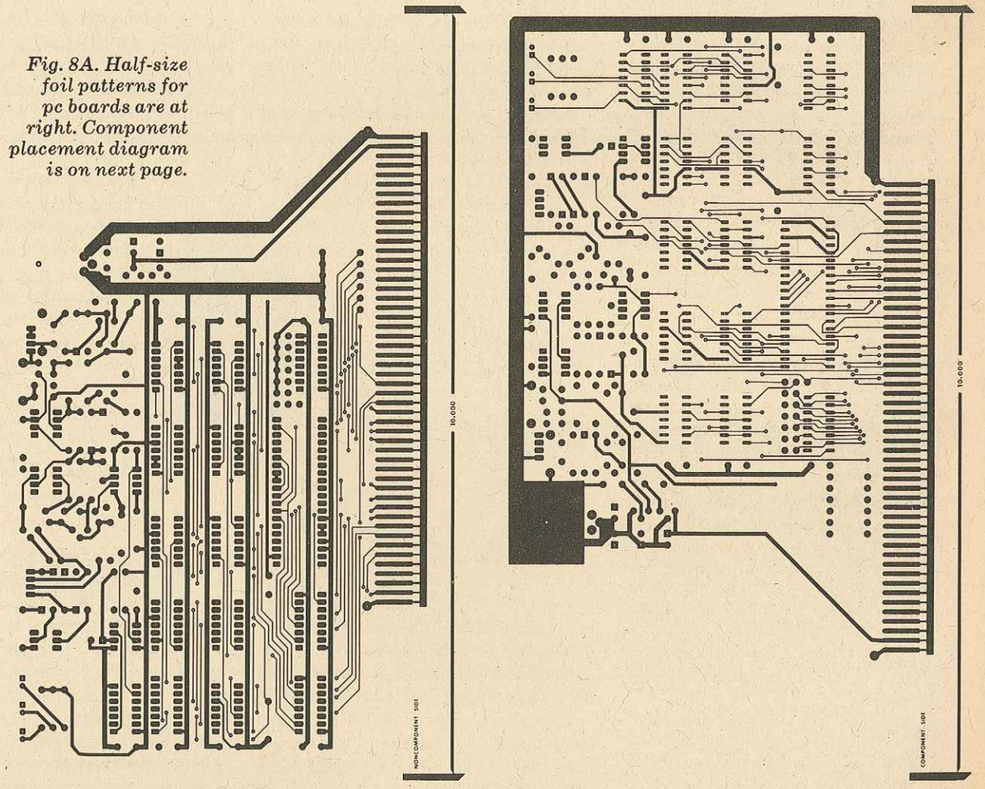


Fig. 7. Self-calibration circuit eliminates need for a frequency counter by comparing local and remote clocks.

Fig. 8A. Half-size foil patterns for pc boards are at right. Component placement diagram is on next page.



Construction. The only practical way of assembling the controller part of the system is on a double-sided printed circuit board. The etching-and-drilling and component-placement guides for the board are shown in Fig. 8. Sockets are recommended for all IC's. However, the transistors, voltage regulators *VR2* and *VR3*, and optoisolator *IC3* can be mounted directly on the board. Main 5-volt regulator *VR1* is installed with the

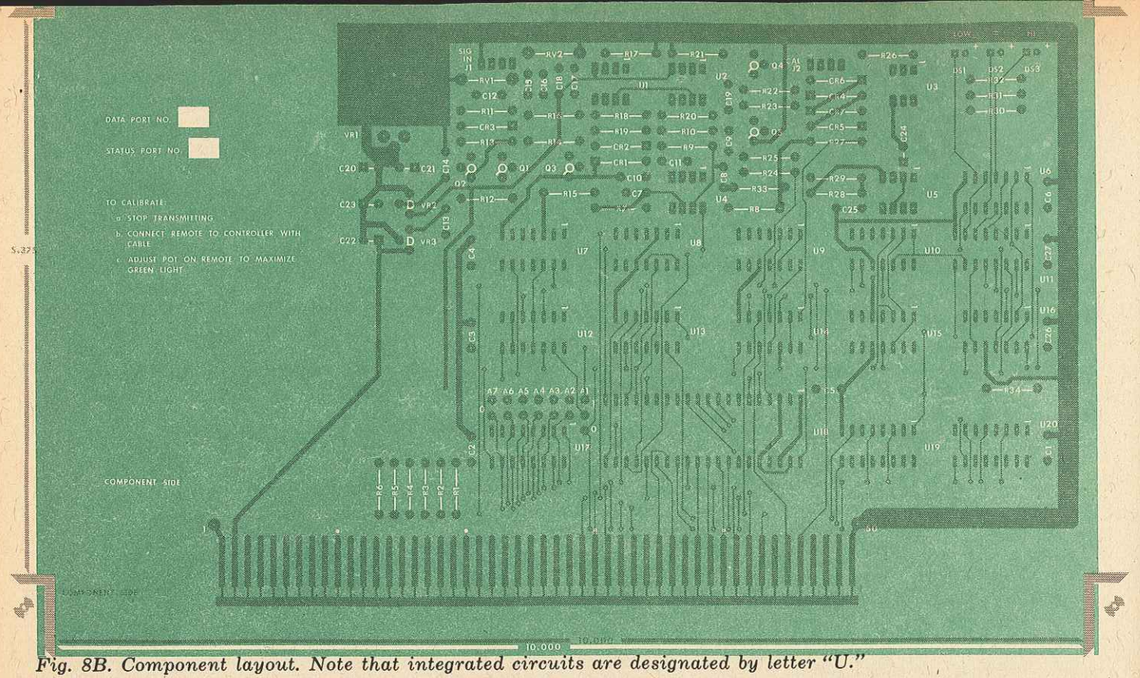


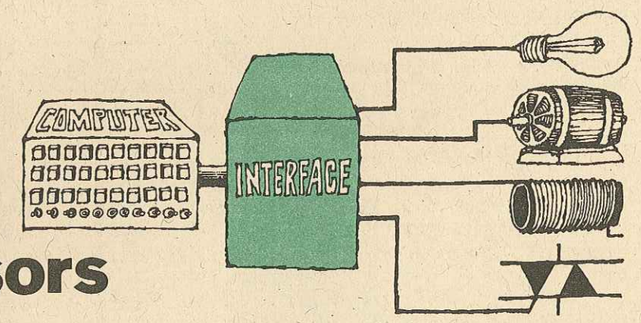
Fig. 8B. Component layout. Note that integrated circuits are designated by letter "U."

usual heat-sink and mounting hardware.

As you are installing the components on the board, pay careful attention to the polarization of diodes and capacitors. Install the IC's last, double-checking the pin-1 identifiers on each to be certain that they are installed properly in their respective sockets.

Coming Up. Next Month, in Part II of this article, we will cover the remote receivers to be used around the house. We will also detail the calibration procedure and present some sample software to use with the system. ◇

How To Interface Microprocessors



BY RALPH TENNY

A MICROPROCESSOR is a relatively complex device. Therefore, interfacing one with peripheral equipment may sometimes present a problem. Just as in any electronic interface, the solution lies in understanding how each side of the interface works and then selecting components and techniques to connect the two smoothly.

The microprocessor communicates with the outside world through three groups of signals as shown in Fig. 1. The address bus usually has between 12 and 16 lines. The data bus has 8, and there can be 1 to 12 control lines.

The internal operation of a processor is based on time—from an accurate oscillator called a clock. Some processors also require two clock signals (ϕ1 and ϕ2, where ϕ means phase) slightly displaced in time. They usually have different time durations, but do not overlap.

Typical machine cycles of operation are shown in Fig. 2 with the input shown in Fig. 2A and the output in Fig. 2B. Note that each machine cycle is divided into a number of time intervals. In each case, the ADDRESS data is sent out during the middle of interval I1 and holds steady until the middle of I4.

For the input or read cycle, the DATA INPUT strobe is high during I2 and drops during I3. For the output or write cycle, the WRITE strobe is low for most of I3. Each of the I1 through I4 time intervals is about 0.5 microsecond, which means that a read or write cycle will occur every 2 μs or 500,000 times/s.

During the read cycle, the processor is *asking* for data and during the write cycle, the processor is *sending* data. If there is to be communication between the processor and any other equipment, then some circuit must be "listening" for the data being sent or some circuit must

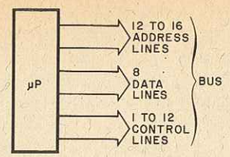


Fig. 1. Address, data, and control line bus allows microprocessor to communicate with outside world.

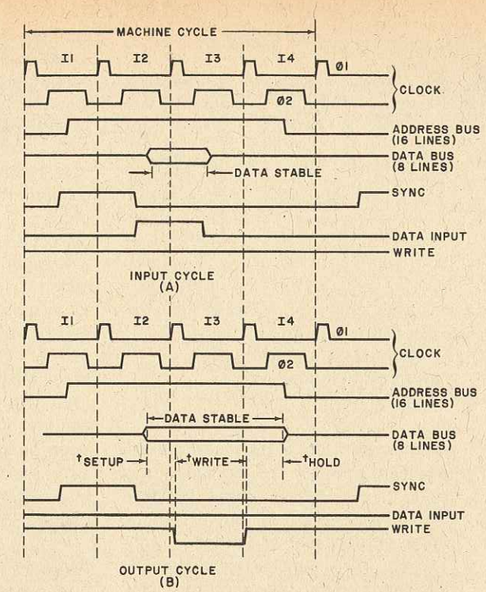


Fig. 2. Correct timing is secret of computer operation. Above are input cycle (A) and output cycle (B).

be able to furnish the data being requested. The processor may not know when its output is not received; but if the data it is requesting is not available, it may stop its operation. This is because part of the data input may include instructions for further operations.

To avoid chaos, one and only one device can send data to the processor during the input cycle. This device is selected by a unique address code that permits only the addressed device to "listen" to the data bus. A control signal (sometimes called a "handshake") tells the addressed device what to do with the data appearing on the data bus. If the three signals—data, address, and control—are to work properly, they must be coordinated in time and this is done by sending the common clock signal through the bus.

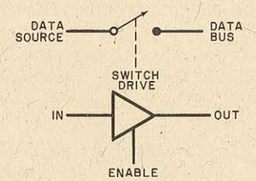


Fig. 3. Three-state buffer is like open switch when enabled.

In summary, successful data transfer between a processor and its associated elements requires three conditions: a unique address, control signals to enable the device being addressed, and means to disconnect data sources from the data bus when they are not specifically requested. Most processors will directly address 2^{16} (65536) different locations since they have 16 address lines.

The most common connect/disconnect system is the three-state buffer whose basic concept and logic diagram are shown in Fig. 3. Such a three-state buffer simulates an electronic switch that is closed only when the enable input is driven by the control signal. When it is not enabled, the output of the buffer is isolated from the internal circuits.

Timing is very critical for data transfer. Note the area marked DATA STABLE in Fig. 2A. The exact timing for data handling varies with different processors but the principle is the same: data must be available and stable for some minimum time and must remain stable for a short time after the three-state enable signal decays. This condition is usually met by using the enable (or DATA INPUT) signal to drive the three-state lines.

The processor output cycle is shown in Fig. 2B. The major difference between the input and output cycles is that, during the output cycle, the WRITE line is low for most of I3 and the DATA INPUT line remains low. Note that the output data from the processor (t_{write}) is available for only about 0.6 μs or less. This means that the IC's used must be able to "remember" the data that appears for such a short time.

Memories. IC memories—from simple flip-flops to RAM's—acquire data in one of two ways. Latches and flip-flops (for example the 74279 latch and the 7474 flip-flop) store their input data on either the positive- or negative-going pulses. In contrast, latches like the 7475, 7477, and 74100 store whatever data is at their D-inputs whenever their enable inputs go high. This is a somewhat subtle distinction and the user must be familiar with the various devices and their performance characteristics. In further contrast, changing data on the D-input of a 7474 will cause no output change until its clock input is driven high. The 7475, 7477, and 74100 outputs follow their D-input as long as their enable inputs are high. Finally, the 74279 latch requires alternate negative-going pulses on the S and R inputs to change the output.

Thus, the 7474 and 74279 devices can be considered to be strobed, or clocked memories, while the 7475's are gated-entry devices. Similar distinctions can be made with CMOS devices, and a careful study of the data sheets will be required to understand each device's operation.

Buffering plays an important part in interfacing a processor with any other device. One common output specification for address and data bus drives for many processors is one TTL load and 130 pF of capacitance. Therefore, if the processor is called upon to handle a number of external devices, some form of buffering must be used to prevent overload of the lines.

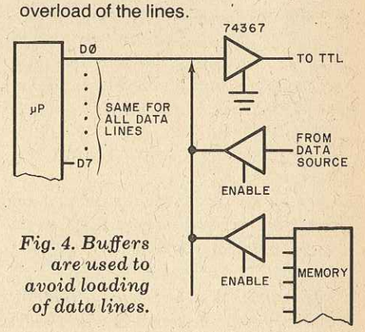


Fig. 4. Buffers are used to avoid loading of data lines.

For the address lines, a device similar to the 74365 or 74367 is recommended, while the data lines are buffered as shown in Fig. 4. If a number of TTL devices is to be driven, then the outgoing lines will also need buffering. Note that the memory lines are also buffered because of the TTL buffer load on the data line. Some medium-size systems use low-power Schottky TTL which has one fourth the loading of a standard TTL, but will drive five standard TTL loads.

Timing. In discussing timing in interfacing, we will refer to Fig. 2B and use a "worst-case" analysis. That is, we will decide which device specification is the

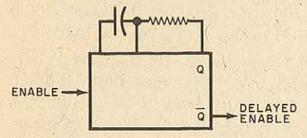
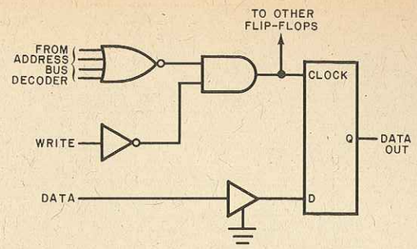


Fig. 5. Enable signal (below) can be delayed by one-shot set for any time delay.



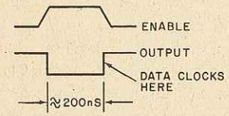


Fig. 6. Circuit above shows how one bit of data can be abstracted from data bus when data, address and handshake signals appear at the same time.

most likely to produce a failure and then be sure the selected part will work.

We will use a D flip-flop such as the 7474 (TTL) or 4013 (CMOS) to "remember" the data. Data set-up time (time the data has to be stable on the D-input before the rising edge of the clock pulse) for the 7474 is a minimum of 20 ns going from logic 0 to logic 1. For the CD4013, data set-up time is 20 ns typically and 50 ns maximum. Data hold time (time the data has to remain stable after the clock pulse edge) for the 7474 is a minimum of 5 ns going from logic 0 to logic 1. Propagation delay (time it takes data to pass through the flip-flop after the clock enters) from the clock pulse edge going from 0 to 1 for the 7474 is 10 ns (min.), 14 ns (typ.), and 25 ns (max.) Going from 1 to 0, it is 10 ns (min.), 20 ns (typ.), and 40 ns (max.). For the CD4013, propagation delay is 150 ns (typ.) and 300 ns (max.).

In a typical processor, the data set-up time when the WRITE line goes down ($t_{set\text{-}up}$) is 140 ns minimum. Data hold time after the trailing edge of the WRITE pulse (t_{hold}) is also 140 ns minimum. The WRITE pulse (t_{write}) is 500 ns min.

Since the maximum set-up time for either flip-flop is 50 ns, either edge of the WRITE pulse could be used to store data. Note the worst-case values: the minimum time for the processor and the maximum time for the flip-flops. Input timing for the same processor is almost handled automatically if the input strobe enables the three-state devices.

Since some processors have very tight timing on the data output bus, a delayed enable may be needed. The one-shot, shown in Fig. 5 will trigger on the leading edge of the enable signal; and, if the D flip-flop triggers on the rising edge of the one-shot output, it will now have the proper delay (set by the RC network).

A circuit that captures a data bit from the processor data bus is shown in Fig. 6. A NOR gate receives the correct decoded address signals, while the data is buffered by a permanently enabled buffer. The WRITE strobe is inverted. Although only one bit is recorded by this circuit, seven more can be clocked by the AND gate to capture the full 8-bit word when the address is entered.

The inverse function, inserting data onto the data bus is shown in Fig. 7. One bit stored in the flip-flop is sent to the processor (via the data bus) when the correct address is received.

Another important facet of input interfacing is the reset of the input data. Once a computer has "read" an input, it has no way to tell when that point is next sampled if the data then present is new data or the same as previously sampled. Therefore, the processor must either reset the data latch after the data has been read out, or must continuously sample the input line until the data changes state. Then the computer can interpret the data changes as valid.

Sample Interface. The most common man-machine interface element is a basic switch. Three ways to use a switch and a flip-flop to input data to a processor are shown in Fig. 8. In each case, a 7474 or 4013 will work; and the three examples show how different system responses can be obtained by setting the flip-flop output to logic 1 by various means. In each case, the immediate response to the switch closure is the same, but the effect on the processor system is different. An example of each type debouncing is shown in Fig. 9.

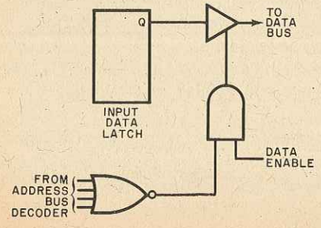


Fig. 7. With simultaneous access and enable signals, one bit of data can be passed to data bus via the buffer.

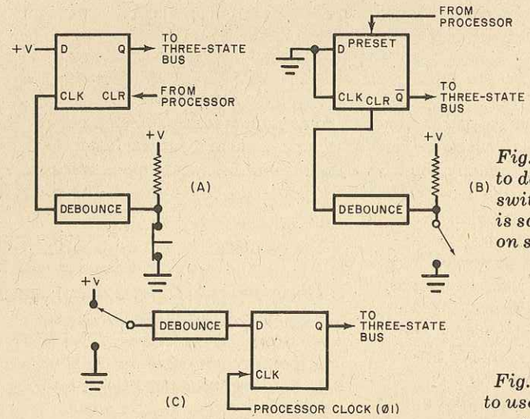


Fig. 8. Three ways, left, to debounce mechanical switch. In all, response is same, but effect on system is different.

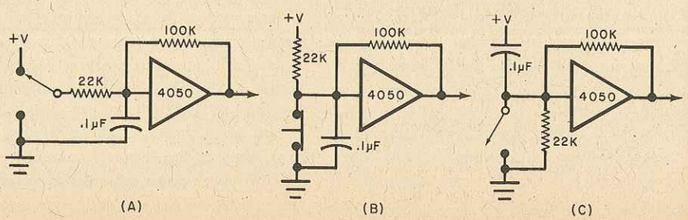


Fig. 9. Below are 3 ways to use CMOS to debounce a mechanical switch.

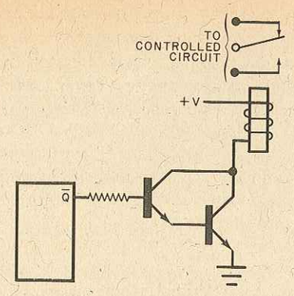


Fig. 10. Output bit can be used to turn on a power transistor and control relay.

In Fig. 8A, the switch drives the clock input high, which causes the Q output to go high. The processor can reset Q to a logic 0 through the clear input and the flip-flop is then ready to respond to another input. Figure 8B shows the clear input being driven, with the output taken from the not-Q and the flip-flop reset accomplished by the preset input. Note that the clear function overrides all other flip-flop inputs so that it will not reset until the switch opens. In Fig. 8C, the flip-flop samples the switch position using the processor clock. The Q output will then track the switch position. If the processor should reset the flip-flop, the Q output would still reflect the switch position after the next clock pulse. Note that the processor clock synchronizes the data entry to the system so that an input can never change while the processor is "reading" the data line.

Control Circuits. If a processor is to perform some useful work, it may have to control large amounts of power. Since its output may be the relatively low current of a flip-flop, some means must be found of controlling higher power. One method is to insert a relay as shown in Fig. 10. The use of Darlington transistors can be extended so that very high-power relays can be controlled. A power semiconductor such as an SCR or triac can be used instead of the relay. The circuit shown in Fig. 11 applies power to the load only at power-line zero crossings to eliminate r-f interference and line transients.

In Fig. 11, when *Q1* is turned on by the reed relay, there is no gate drive for *Q2*. When *Q1* is off, the gate drive for *Q2* is through *R1*. Note also that *R1* should be capable of handling the full line power while passing adequate current to trigger *Q2*.

In general, dc loads can be handled in the same way as ac loads except that suitable power transistors are used in-

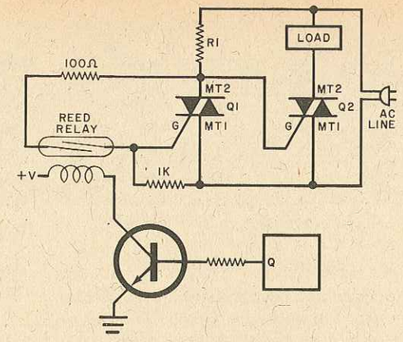


Fig. 11. Using output bit to turn on a triac eliminates RFI during switching operation.

stead of triacs. As long as the dc operating voltages are derived from transformer-powered supplies, the major precautions to be observed are proper voltage insulation, and adequate heatsinking for the power semiconductors.

Motor Controls. In computer control of motor speed, there are two basic methods which can be used: open loop or closed loop. A simple example of the former merely involves setting a supply voltage for the motor and using the resulting motor speed. Depending on how the mechanical load varies, this method can allow motor speed to vary 10 percent or more.

Closed-loop control involves the continuous sampling of motor speed and setting a voltage (or other signal) to obtain the desired speed. Such control usually involves current sampling; and, if the motor or mechanism it is driving becomes jammed, closed-loop control attempts to drive the motor faster. As a result, either the motor, the power supply, or both, can be damaged. The solution to this problem lies beyond the scope of this article, however.

Sensing motor speed can be done in one of a number of ways. The simplest is accomplished by the circuit shown in Fig. 12A. A series of pulses from the motor drives a counter that is coupled to the processor through three-state buffers. The processor periodically reads the counter, resets it, and compares it with the count required by the program.

The motor rotation pulses can be generated by either of the two systems shown. In Fig. 12B, a sliver of shiny aluminum tape on a dark shaft allows light to bounce onto a photocell. The cell drives a suitable circuit that shapes the pulses for use by the counter. The slotted-disc approach shown in Fig. 12C also uses a light source and a photocell. Both of these methods are linear with changes in rpm, and the choice of which one to use depends on the amount of resolution required. If the motor speed tends to vary very quickly, the rpm must be sampled very often, so a large number of pulses per revolution is required to make accurate measurements. If the motor shaft operates at high speed, and the load has high inertia, one pulse per revolution may be sufficient. Another speed measuring technique involves the use of a tachometer, which is often a part of a motor and delivers a dc voltage linearly proportional to rpm. An analog-to-digital (A/D) converter must be used to convert the tachometer output into a signal suitable for the processor. The converter must also be furnished with address decoding and three-state bus drive. The advantage of the added complexity is that very close control can be maintained over motor speed. The basic logic approach is shown in Fig. 13A.

Another type of closed-loop control is shown in Fig. 13B. A small dc motor is

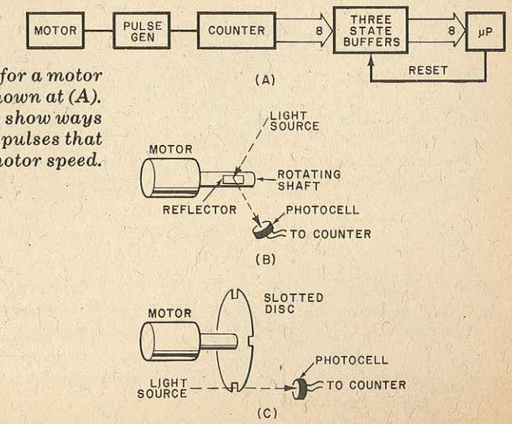


Fig. 12. Circuit for a motor control is shown at (A). Both (B) and (C) show ways to generate pulses that represent motor speed.

used to drive a potentiometer that sets the reference voltage level. If the motor has a large gear-reduction train and if the potentiometer is a multi-turn device, the reference voltage can be set very accurately. Note that, in this scheme, the processor is connected only to the control signals and not the actual power system.

A modern method of producing the necessary accurate reference voltage is shown in Fig. 13C. A D/A converter having an 8-bit resolution (1 part in 256 or 0.4 %) can do the required job. The data latches with address select are necessary to hold the D/A output between changes.

A final type of motor, extensively used with computers, is the stepping motor. It operates by having (typically) two to four drive coils and a rotor with an odd number of poles. When power is applied to a drive coil, the rotor locks in one position. If the alternate drive coil is energized, the first coil is turned off and the rotor increments once and locks. Thus, alternating pulses to the drive coils produce discrete increment rotation. Typically, a rotor may be advanced by 5° or 7.5° per step, which, when combined with a suitable gear train, can produce very fine resolution of mechanical position. Variation in the pulse rate produces excellent control of motor speed.

Other types of sensors that can be used with computers include various forms of limit and proximity detectors, item counters (for conveyor belts), and fire and intrusion detectors. ◇

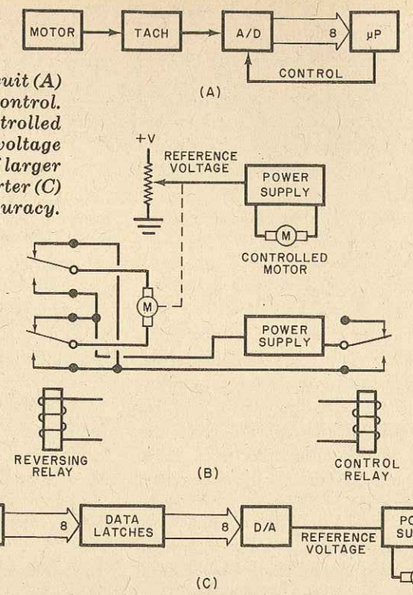
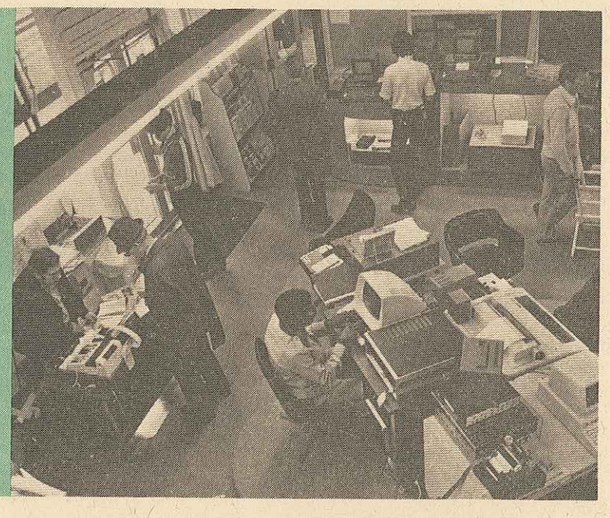


Fig. 13. Tachometer circuit (A) provides close speed control. In (B), small motor, controlled by computer, adjusts voltage to maintain speed of larger motor. Use of D/A converter (C) provides motor speed accuracy.

COMPUTER STORES:

A New Retailing Phenomenon

BY SHERMAN WANTZ

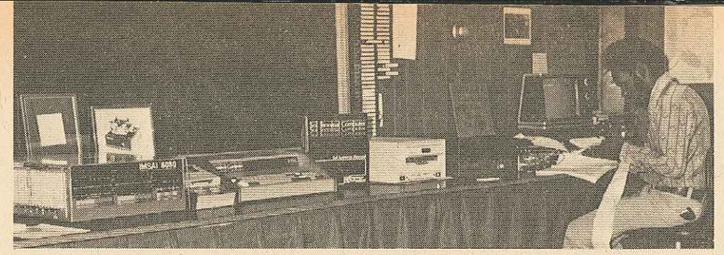


Independent shops like N.Y.'s Computer Mart sell many brands.

YESTERDAY, home computers were a science-fiction fantasy. Tomorrow, you'll probably find them under a heading of their own in the "Yellow Pages." And today there are already more than 900 home computer dealers in this country.

Not all computer dealers are alike, though. They range from part-time operations by computer hobbyists, through small departments in big electronics stores, to full-time specialist computer dealers. One way or another, if you live near a city with 50,000 people or more, you're likely to find a retail computer outlet of some kind nearby.

Finding it may be a problem, though (unless you're in California, which has about 25% of the nation's computer dealers). Most of the specialist stores have limited advertising budgets. And most "Yellow Pages" listings so far lump the home computer stores with the ones serving businesses, under such headings as "Computing Devices" or "Data Processing Equipment."



Equipment is displayed in open in MicroComputer System's Florida store.

Certain key words in the company name will help you identify the home computer stores when you run across them: "Byte," " Computer," "Micro," "Digital" and "Data." Often, too, the names are deliberately un-stuffy, such as "Bits 'n' Bytes," "Kentucky Fried Computers," or "Digital Deli." But not all hobby computer dealers are so readily identified by name alone. Many electronics stores, such as Allied Electronics, Team Electronics and Radio Shack are entering this field.

What You'll Find. Like car dealers and furniture showrooms, computer stores are usually set up to encourage browsing. Most people have never touched a home computer yet and contact is addictive. So you'll seldom find counters separating you from the merchandise. Computers, video terminals, tape readers, keyboards and other equipment are likely to be displayed openly on tables. More often than not, some equipment will be connected and in operation. Ask a salesman if you can try your hand at any of the equipment that's running, or if he can demonstrate what one can do with a computer.

Independent computer stores give you the chance to compare and evaluate many different makes of computer equipment and systems. But not all computer stores are independent. MITS, for example, the company that makes the popular Altair line of microcomputers, has exclusive franchise agreements with dealers who sell no competitive products. Similarly, Heathkit computers are sold exclusively by Heath Electronic Centers and by mail from Heath's main office; Radio Shack stores will handle only the Radio Shack computer line (though Tandy Computer Stores, and probably others, will handle both Radio Shack and competing hardware).

However, computers aren't all you'll find at computer stores. You'll probably see a variety of terminals, for example, from the old standby Teletype to faster and fancier (or more limited, but cheaper) types of printer, plus several models of CRT terminals. You'll find a variety of peripheral and support types of computer equipment, too.

Along the walls of the store you'll certainly see shelves containing books, magazines, newsletters, promotional material, and probably assorted components, hand tools and test gear. If you're unfamiliar with computers, ask a salesman to suggest which magazines and books to buy, and which brochures to take home and study. These publications will supply answers to some of your initial questions (including the ones you may feel a bit shy about asking). More importantly, they'll also suggest a number of new questions you'll want to ask on your next visit.

Some of the most important merchandise the computer store has to offer is the least impressive looking: software. It is what computer people call the programs without which the hardware wouldn't work. The availability of programs is one of the main factors to consider when buying a computer system. A computer store can help you find out what software is available for your present or prospective system. Even more important, they can help you make the small but vital changes to the programs that may be needed to make it run on your particular system. What's more they can let you try out programs on the store's equipment to see if that software will suit your needs at home.

Your best introduction to computers is to play a game with one. Almost all stores have programs on hand for playing games, from simple Tic-Tac-Toe, through Blackjack, to a sophisticated game called "Startrek," patterned after the popular TV series.

For now, the heaviest emphasis is being placed on the microcomputer's entertainment value; but the availability of more advanced programs and equipment is changing that. Today's hobby computers are being used not only for playing games, but for controlling electro-mechanical devices and business and educational purposes.

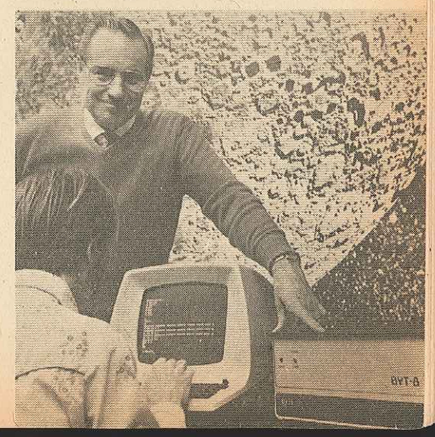
Digital Cracker-Barrel. Like the old country store with its potbellied stove and cracker-barrel, the computer store is serving as a meeting and discussion center. Often, you can learn almost as much from talking with the customers as you can from the salesmen. While computer stores and Startrek attract their share of kids, you'll find a number of computer professionals and serious hobbyists there too. They may be programmers who work for one of the growing number of computer service companies; electronic technicians and engineers; students who've already taken computer courses in high school or college; amateur radio enthusiasts; or businessmen anxious to learn how a microcomputer can relieve them of tedious, routine chores.

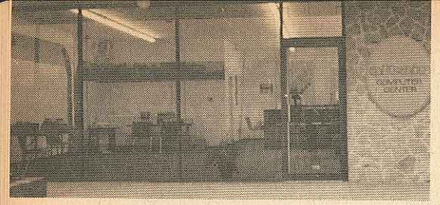
Your presence at the computer store creates a bond between you and the other customers. You'll find it easy to strike up a conversation with one or more of those who are inspecting or operating equipment. They are as anxious to discuss computers as you are. Often, they'll be more knowledgeable about particular aspects of computer hardware or software than the store's own employees. People who already own computer models that you're considering for yourself can prove particularly helpful.

Look for a bulletin board on which the local computer club might announce the time and place of its next meeting. If you don't see such an announcement, ask one of the store's employees about the existence of a club. He should know. If you can find a club, you'll find a lot of the talk at your first few meetings rather hard to follow. But you'll also meet a lot of other hobbyists who'll be glad to explain things to you.

Special Store Services. Because

Demonstrating a microcomputer system in one of Byte Shops' chain of computer stores.





Altair carries mainly MITS gear.

computers are so complex, and so new to most people, computer customers need a lot of special services. And most computer stores provide them.

If you're handy enough to build some of your equipment from kits (and save up to 40 percent in the process), most stores will help you interpret unclear instructions and check out your work when you've finished. If you're unsure about your ability to build a particular kit, the store will often let you look over its construction manual, first, to get an idea of its degree of difficulty.

If you don't want to build a kit, but want an item that's not available in assembled form, many stores have technicians who'll build it for you—for a fee.

Computer stores usually have service facilities where you can take a malfunctioning computer (or the appropriate boards, if you can narrow down the problem) for testing and service. Bring a copy of your program, too; often, computer problems turn out to be in software, not in hardware.

Some technicians don't mind letting you watch and learn as they troubleshoot your system. But remember that you pay for most repairs at an hourly rate. Talking to the technician slows him down, and costs you money.

Stores will generally replace any defective parts in kits they've sold you (but not kits you've bought elsewhere). Servicing completed kits is usually done for the same flat fee or hourly rate as the manufacturer would charge, and saves you shipping time and charges.

Many stores provide consulting services, custom-designed hardware and software, and information on how to modify your system for better performance. More and more stores, in fact, are devoting a lot of attention to providing such services for small businesses (which gives them lots of experience for handling your problems, but may mean the technician or salesman you want to see is out if you just drop by unannounced). For the hobbyist, many stores give low-cost classes in computer and programming fundamentals.

For established customers, many stores will accept phone orders, often shipping out their orders overnight. Many also accept major credit cards.

When it's time to upgrade your system, the store where you bought your computer will usually have add-on module boards and peripherals, or be able to suggest equipment modifications, that will handle your requirements. If your old equipment simply can't be made to handle your new needs, many stores have bulletin boards where you can post your old equipment for sale. A few stores even take trade-ins.

If the store nearest you does not yet offer all of these services, don't be disappointed. The field is growing rapidly, and most stores, still small, must work hard to keep up.

Still, this is the calm before the storm, the lull before the home computer hobby *really* takes off. Someday you may have to take a numbered card and stand in line waiting for a salesman to take our order—as soon as he can free himself from the constantly jangling phone.

Beat the crowds, and begin now to visit the computer stores near you. Compare the lines of equipment each handles. Find out which stores give you the greatest bargains in quality merchandise and the most personal attention. Don't hesitate to ask about the availability of the services mentioned here. (But don't expect to find all of them in any one store, either.) Once you've found a store whose technical experts give you confidence, that's where you should go for help in setting up your own computer system.

And after you have your computer up and running, remember to keep in close touch with your computer store. In this fast-moving hobby, that's where much of the action is. ◇

QUICK HEX-DECIMAL CONVERSIONS

BY RAYMOND J. BELL

CONVERSION from hexadecimal to decimal or vice versa is sometimes required in microcomputers. The table presented here offers a rapid and efficient solution to this problem. It is suitable for integers between 0 and 65,535 (O_{16} to $FFFF_{16}$). It can also be easily expanded.

Here's an example of how to use the table. Say the hexadecimal number, $A7BD_{16}$, is to be converted to decimal. Starting with the right-most digit, D, look at the table's fourth-place digit and read down to D in that column. The decimal equivalent is 13. Repeat for the next digit in the third column. Here, the original number, B, corresponds to 176. Continuing with the next two digits, we read 1792 and 40960, respectively. Add these numbers, and the total is 42941, which is the decimal equivalent of $A7BD_{16}$.

The table can also be used in reverse to convert decimal numbers to hex. To convert 800_{10} to hex, for example, look in the table for the highest entry which does not exceed the number, which is 768. This corresponds to a 3 in the third hex digit. (The fourth digit is 0, so it can be ignored.) Next, 768 is subtracted from 800, yielding a remainder of 32. The highest table entry that does not exceed 32 is 32, which corresponds to a 2 in the second hex digit. Subtracting 32 from 32, the remainder is zero, which means the conversion is complete. (Note: to maintain proper relationship of the hex digits, we put 0 in the first hex digit, giving 320_{16} as the hex equivalent of 800_{16}, *not* 32_{16}, which is 50_{10}.)

The table can be expanded by multiplying the digits of 0 to 15 by the appropriate power of sixteen. To construct the fifth column of the table, multiply 16^5 (65,536) by 0, 1, 2 to 15. ◇

HEX-DECIMAL NUMBER TABLE

1st Place		2nd Place		3rd Place		4th Place	
Hex.	Dec.	Hex	Dec.	Hex.	Dec.	Hex	Dec.
0	0	0	0	0	0	0	0
1	4096	1	256	1	16	1	1
2	8192	2	512	2	32	2	2
3	12288	3	768	3	48	3	3
4	16384	4	1024	4	64	4	4
5	20480	5	1280	5	80	5	5
6	24576	6	1536	6	96	6	6
7	28672	7	1792	7	112	7	7
8	32768	8	2048	8	128	8	8
9	36864	9	2304	9	144	9	9
A	40960	A	2560	A	160	A	10
B	45056	B	2816	B	176	B	11
C	49152	C	3072	C	192	C	12
D	53248	D	3328	D	208	D	13
E	57344	E	3584	E	224	E	14
F	61440	F	3840	F	240	F	15