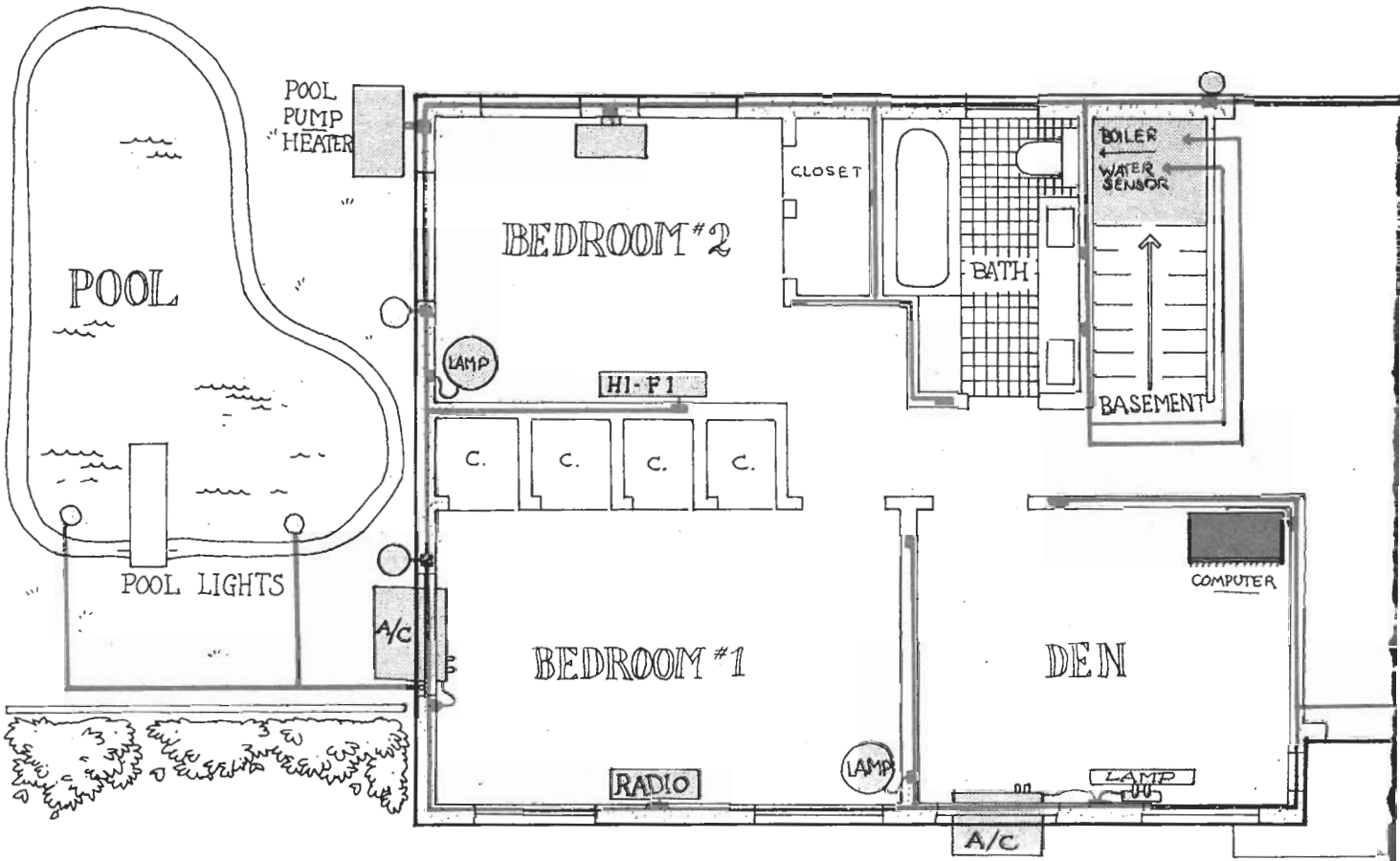


Using Existing House Wiring

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.

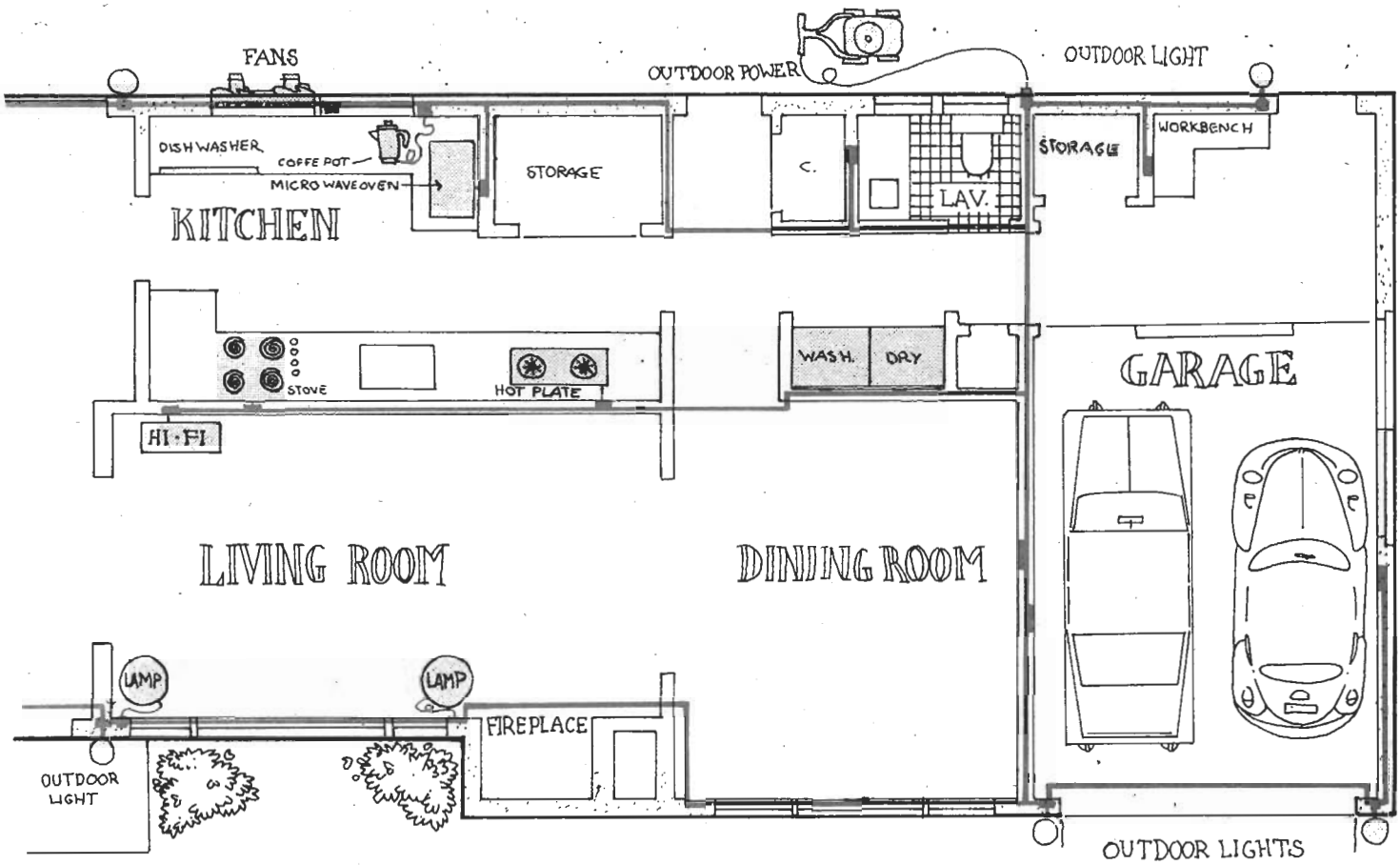
In this first article, we cover the controller circuit (Fig. 1). There are two major sections in this device—the controller itself and a self-calibration circuit that is used for setting up the remotes.

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

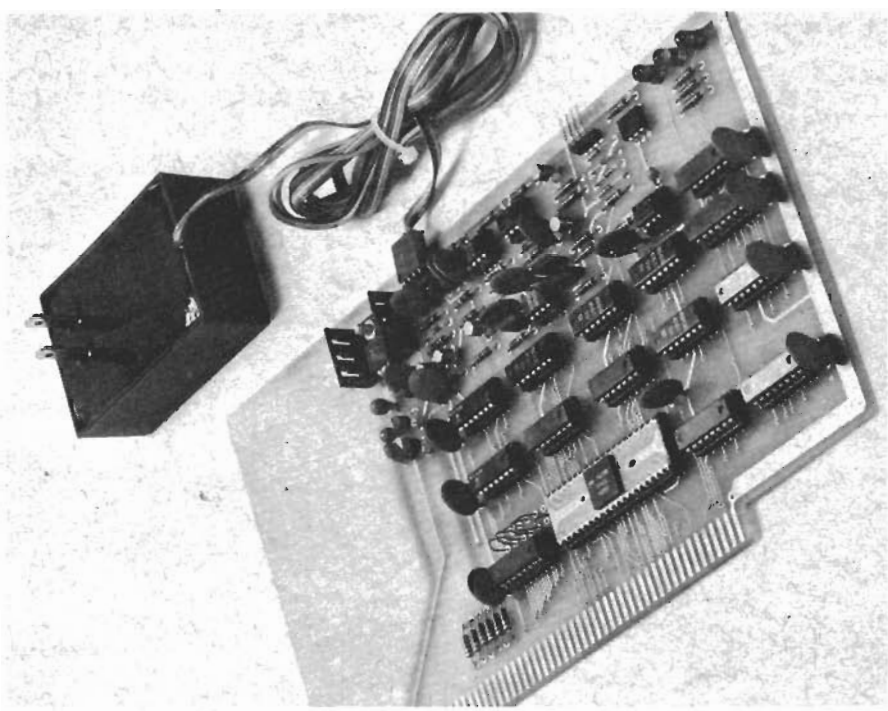


Photo of prototype controller board with adapter plug into ac line.

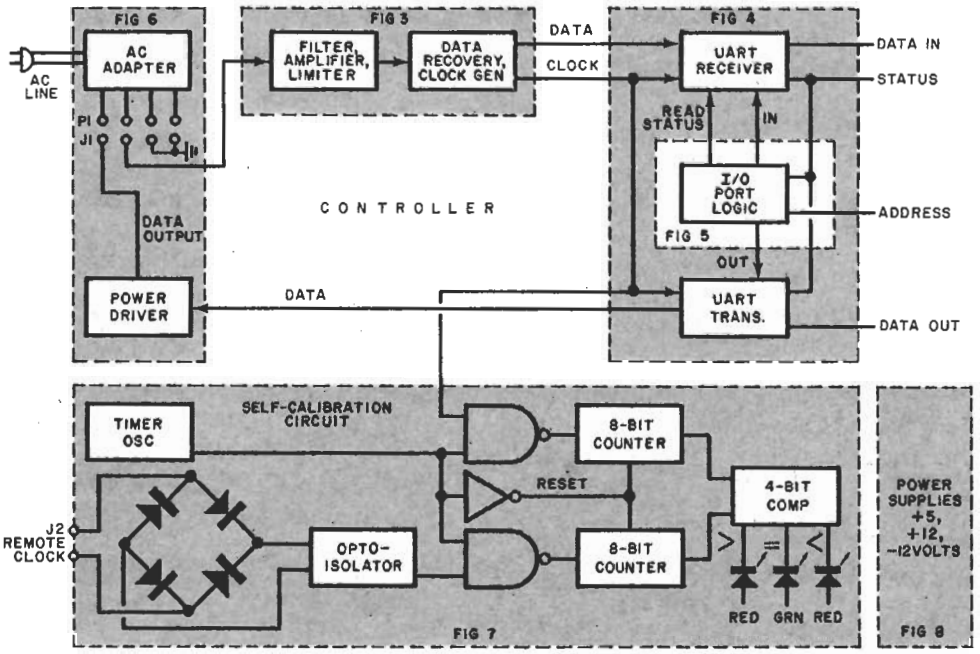


Fig. 1. Overall logic of controller that also includes self-calibration circuit for remote oscillator alignment. Sections of the controller are referred to by figure numbers in which complete schematics are given.

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.

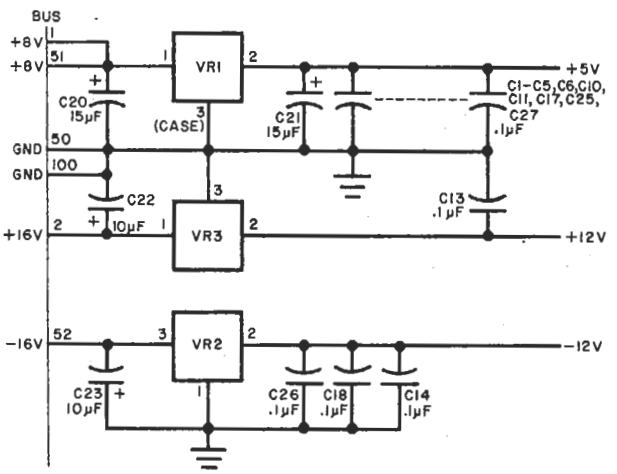


Fig. 2. Schematic of controller power supply.

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.

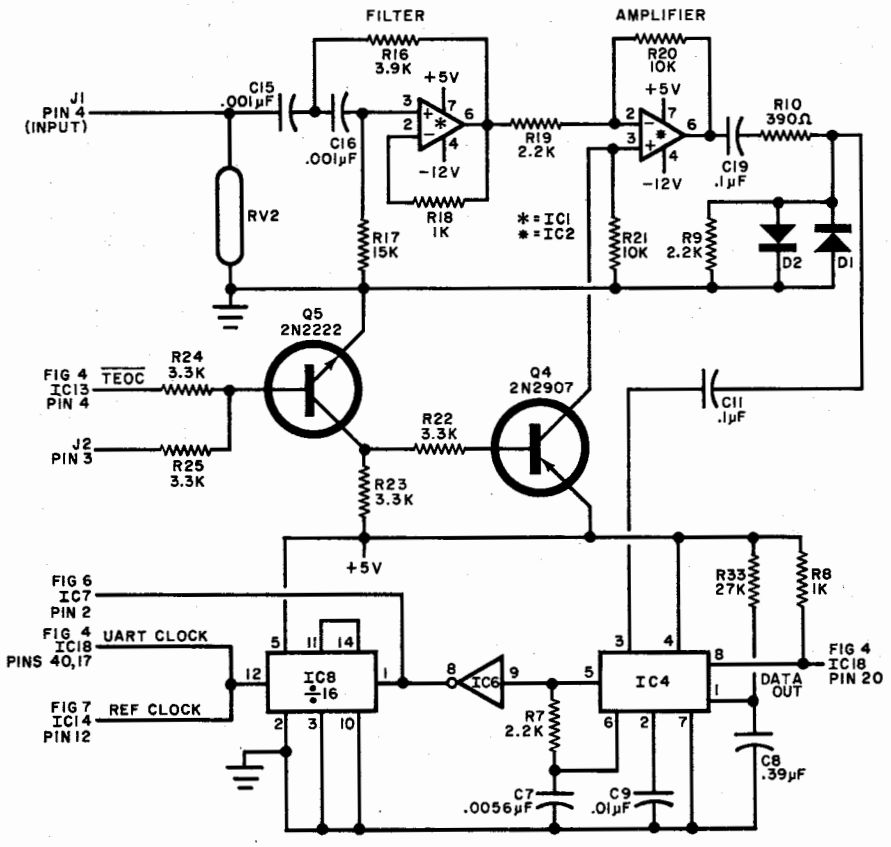
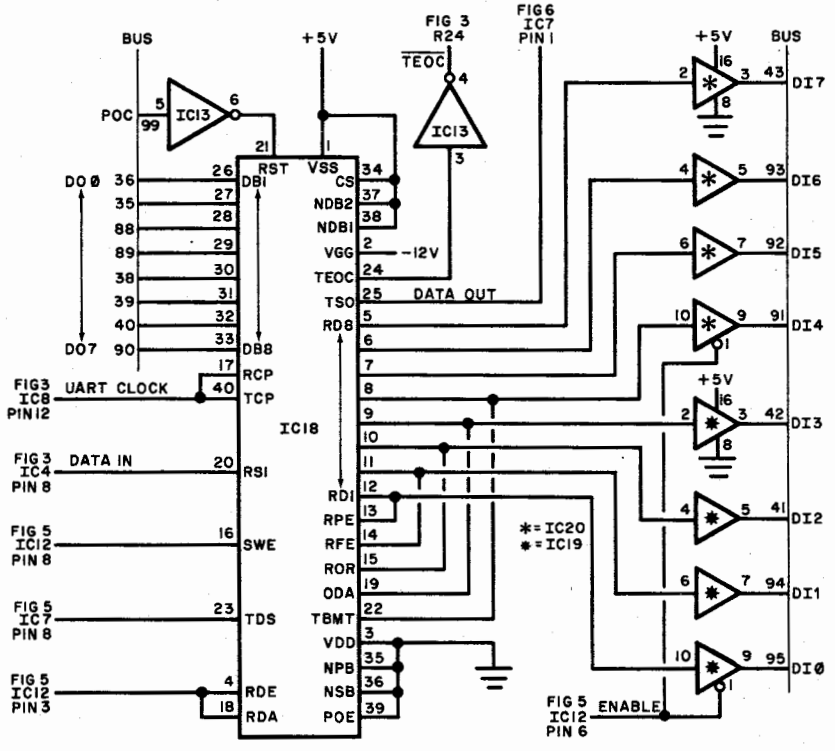


Fig. 3. Filter, amplifier, limiter, and data and clock recovery.

Fig. 4. UART connection between controller and computer.



Two other status bits are available: output data available (ODA) and transmit buffer empty (TBE). When ODA is a 1, data is available at the receiver's holding register. When TBE is a 0, the transmitter is busy.

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

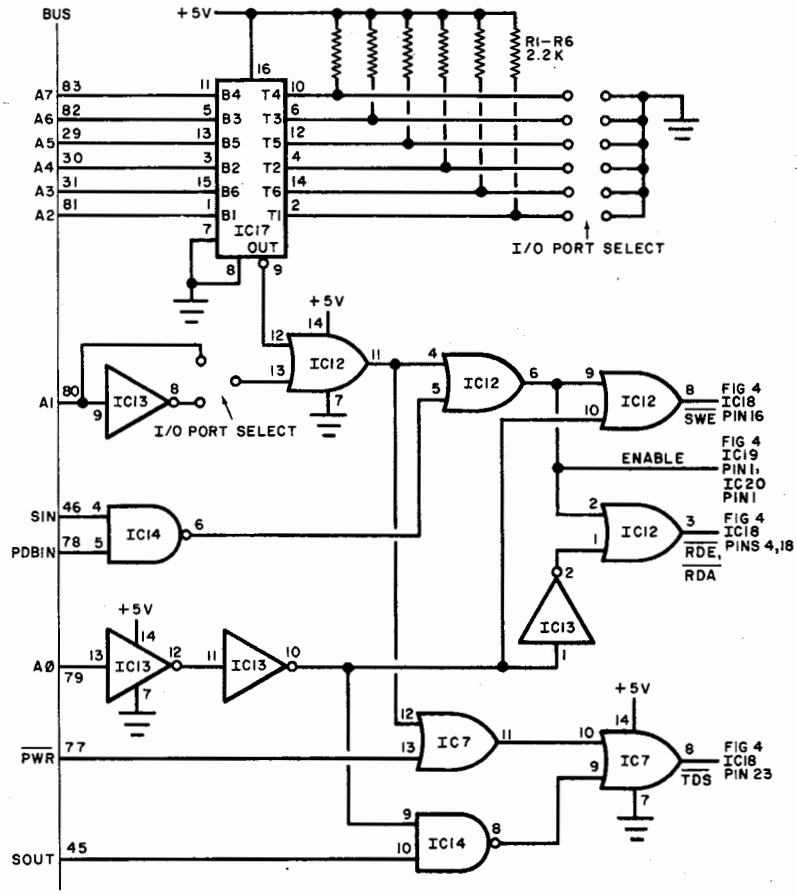


Fig. 5. The I/O port selection is made by choosing the jumper arrangement for the selected port.

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 frequen-

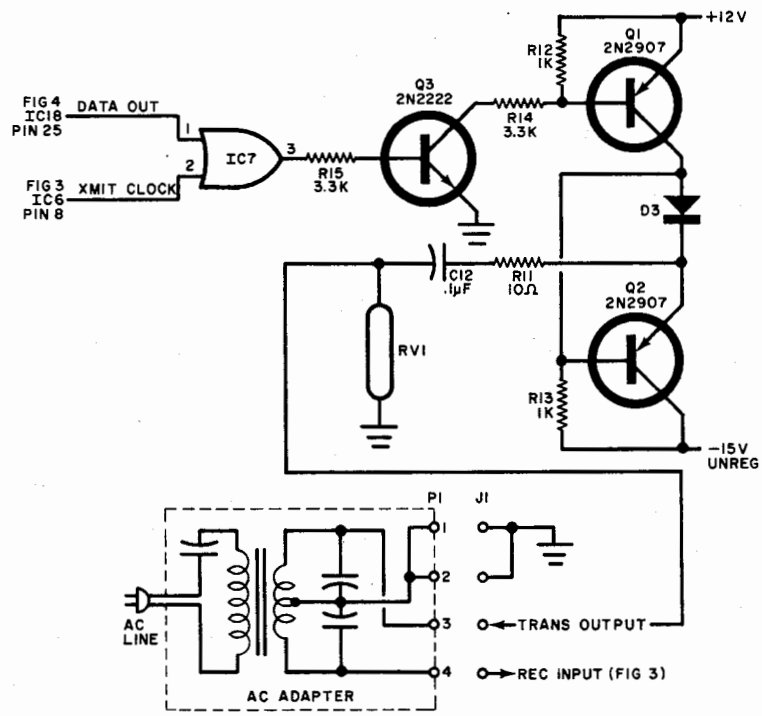


Fig. 6. Power driver accepts UART data and clock and delivers high-level signal to ac adapter.

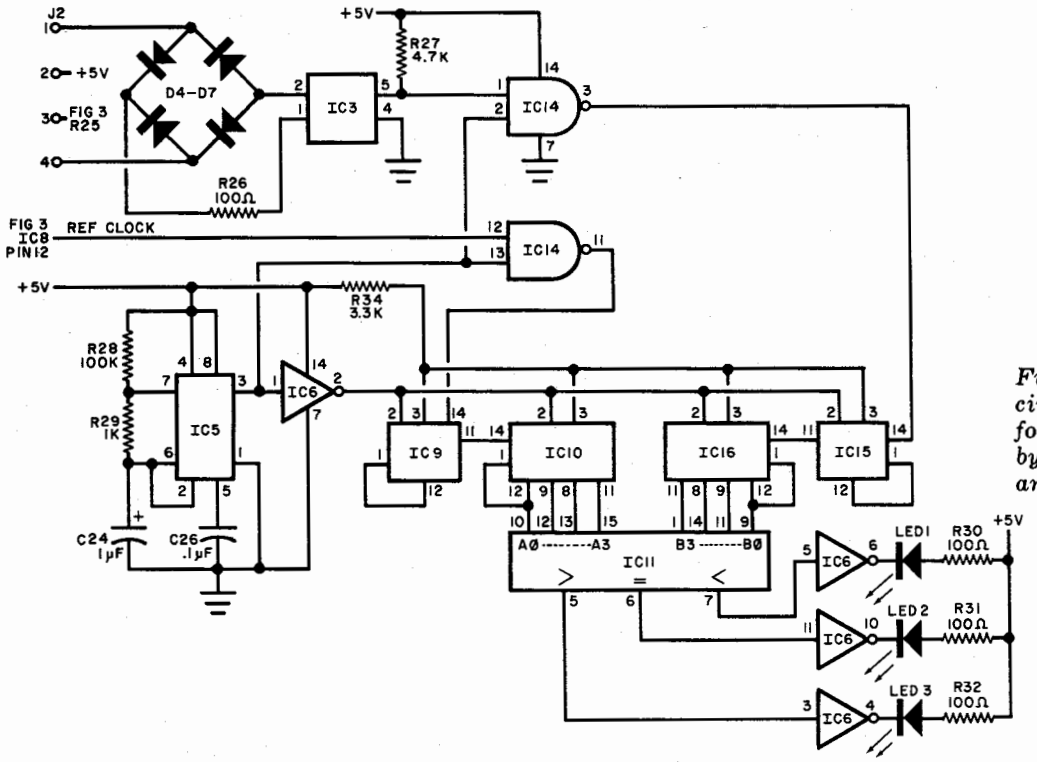
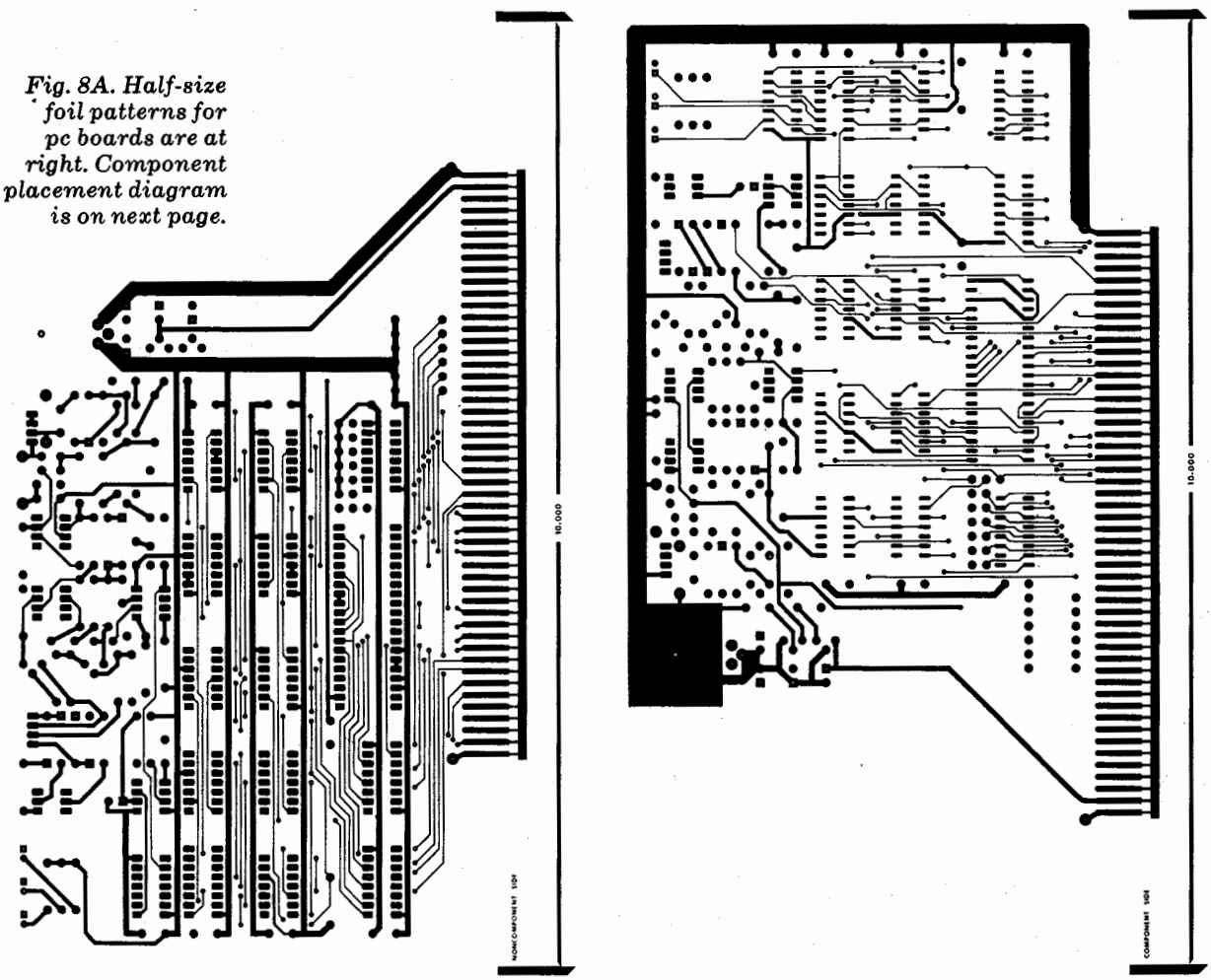


Fig. 7. Self-calibration circuit eliminates need for a frequency counter by comparing local and remote clocks.

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



cy. The two LED's indicate in which direction the remote vco differs (less or greater than) from the controller vco.

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

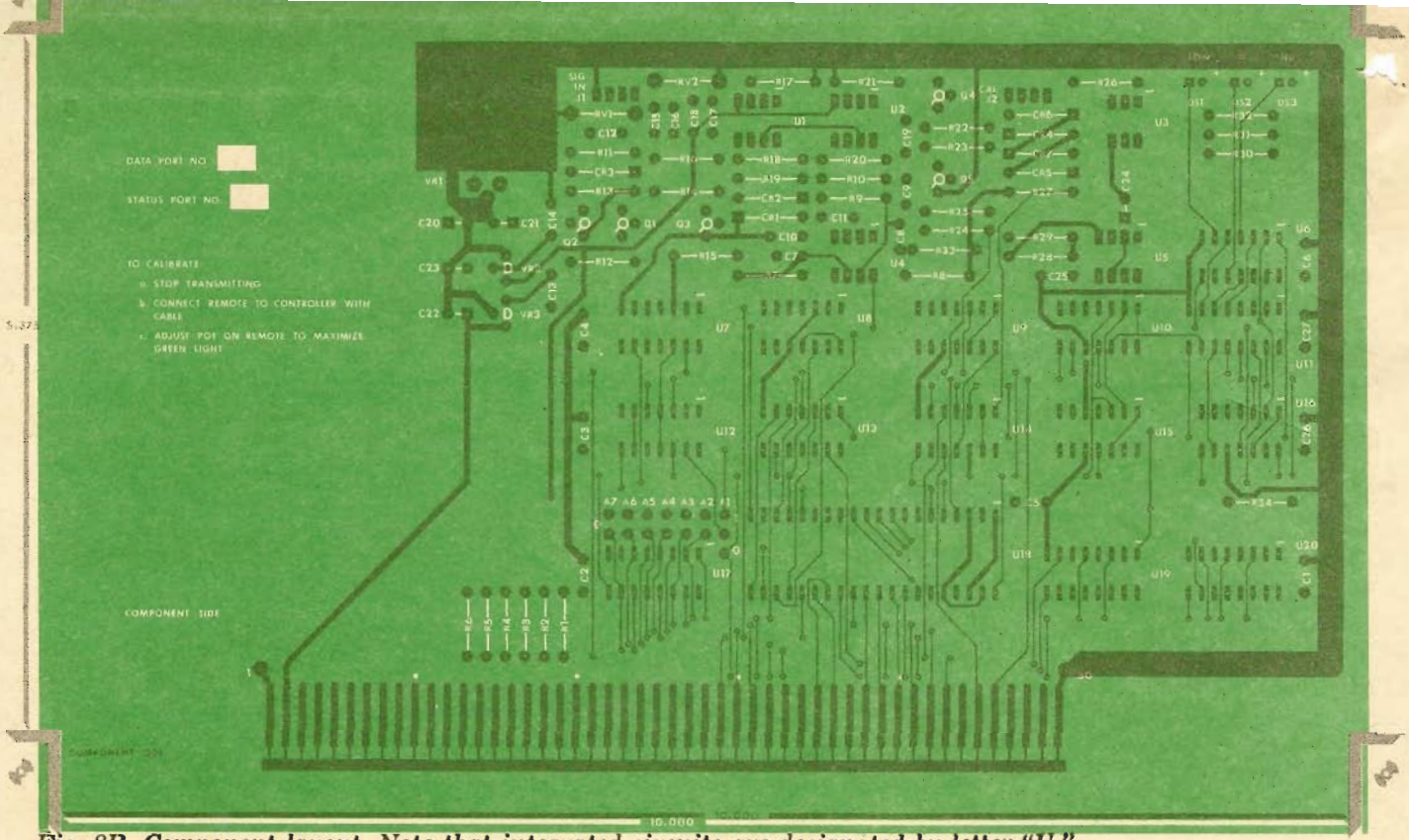


Fig. 8B. Component layout. Note that integrated circuits are designated by letter "U."

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. ◇

Using Existing House Wiring for Computer Remote Control

PART 2

How to build a typical remote.

BY DAN SOKOL, GARY MUHONEN, AND JOEL MILLER

LAST MONTH, we described the theory and construction of an Intelligent Remote Controller that utilizes a building's standard ac wiring for communicating between a computer and appliances. In this concluding part, we cover the details of a typical two-channel remote unit (sometimes called just a "remote") and discuss some software to get the composite system "up and running."

The basic block diagram of a remote is shown in Fig. 1. Note that many sections of the remote resemble their counterparts in the controller because both devices can send and receive data over an ac power line.

How It Works. The user determines which remote he wishes to communicate with and what command he wishes to issue. For example, if he wants to toggle remote 41, a 233 must be outputted to the controller output port. The computer then executes the assembly language command OUT 5. (5 is the num-

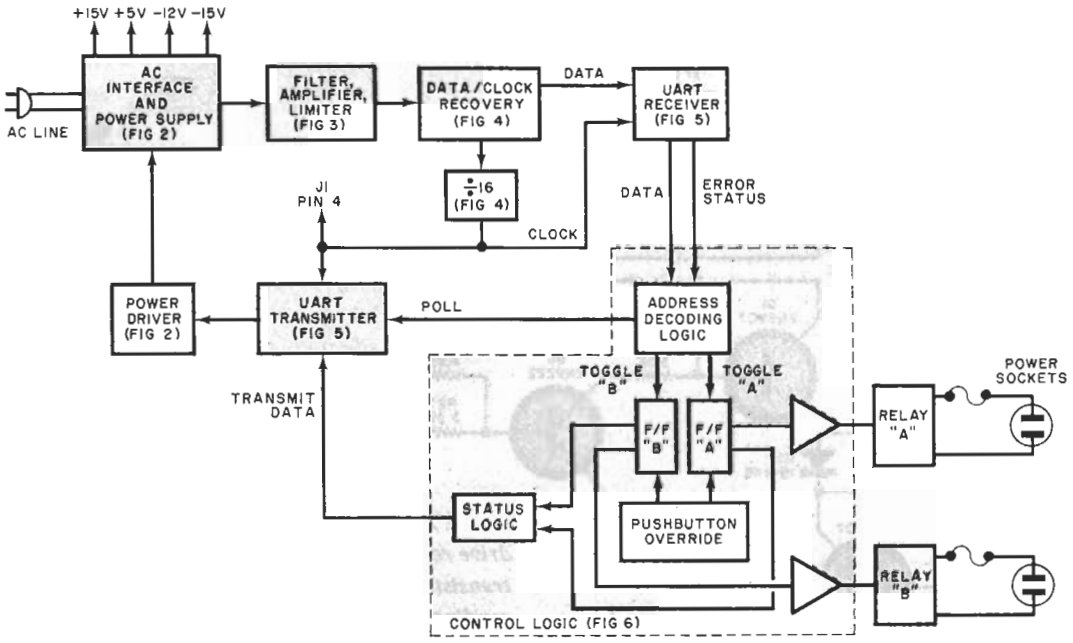


Fig. 1. Block diagram shows similarity of remote to last month's main controller.

ber of the output port while 233 is data.) The I/O port decoding logic on the controller board determines that the controller is being addressed with an output instruction. The controller UART transmitter then reads the data bus, formats the word, and sends it out to the power driver as a serial stream of data bits.

The power driver impresses the signal on the ac line via the ac interface adapter. The data appears on the ac line as a digitally modulated signal at about 50 kilohertz.

All the remotes are constantly monitoring the ac line for possible commands. Each remote contains two independent channels, each capable of controlling one external device plugged into its power socket. This means that each remote is assigned two sequential addresses (selected by the user by putting jumpers on the remote board).

The signal received by the remote is coupled through an ac interface adapter tuned to 50 kHz. A high-pass filter (rolling off at 6 dB per octave below 20 kHz) removes the 60-Hz line frequency and all its relevant harmonics. The filtered output is amplified and used to drive a phase-locked loop (PLL). There, the vco output from the loop is divided by 16 and used as the clock for the internal UART. The received data is recovered at the lock output of the PLL, and this signal is used as the input to the UART receiver.

When the receiver detects a data word, that word appears on its eight parallel output lines, along with error and flag information. The address and decode logic then determines whether or not that word is intended for that remote.

The three valid outputs from the address and decode logic are toggle-A, toggle-B, or poll. The latter is actually two commands—poll-A or poll-B—and the

PARTS LIST

C1,C2, C14, C15—0.1-μF, 200-V capacitor
C3—0.015-μF capacitor
C4,C5—0.001-μF capacitor
C6 through C10, C16, C17, C22 through C27, C34—0.1-μF, 25-V capacitor
C11—0.39-μF capacitor
C12, C18, C19, C28 through C33—0.01-μF, 200-V capacitor
C20, C21—470-μF, 25-V electrolytic
D1 through D5, D10, D11—1N4148
D6 through D9—1N4001
F1—¼-A fuse and holder
F2,F3—5-A fuse and holder
IC1—TR1602 UART
IC2, IC3—4069 CMOS hex inverter
IC4,IC8—4001 quad 2-input NOR gate
IC5—4011 quad 2-input NAND gate
IC6—74C107 dual JK flip-flop
IC7—74C30 8-input NAND gate
IC9—74LS93 4-bit binary counter
IC10, IC11—NE535 op amp
IC12—NE567 PLL tone decoder
K1, K2—Spdt, 5-A contact-rating relay (Stancor MS64-931 or similar)
Q1, Q2, Q4—2N2907 transistor
Q3, Q5, Q6, Q7—2N2222 transistor
Following resistors are ¼-watt, 5% unless otherwise noted:
R1—15,000 ohms
R2—3900 ohms
R3, R13, R17, R18, R19, R23, R24—1000 ohms
R4, R11—2200 ohms
R5, R6—10,000 ohms
R7, R8, R9, R20, R21, R22—3300 ohms
R10—390 ohms
R12—27,000 ohms
R14—1800 ohms
R15—1000-ohm, 10-turn trimmer potentiometer
R16—10 ohms
R25—200 ohms
R26, R28, R30—100,000 ohms
R27, R29—270,000 ohms
RV1, RV2,—V33MA1A varistor (GE)
SΦ,S1—Spst normally open, pushbutton switch
T1—Coupling transformer (see Note)
T2—25-V CT 180-mA transformer
VR1—7805 5-volt regulator
VR2—79L12 −12-volt regulator
Misc.—In-line fuseholders (3), 117-volt, chassis-mount ac sockets (2), line cord, suitable enclosure, mounting hardware, etc.
Note: The following are available from Mountain Hardware, Inc., P.O. Box 1133, Ben Lomond, CA 95005 (Tel: 408-336-2495): T1 (MH-T1) for $6.00; complete kit for one dual-channel remote including walnut case for $99.
Diodes are identified by letters "CR" and IC's by letter "U" in parts placement guide in Fig. 7.

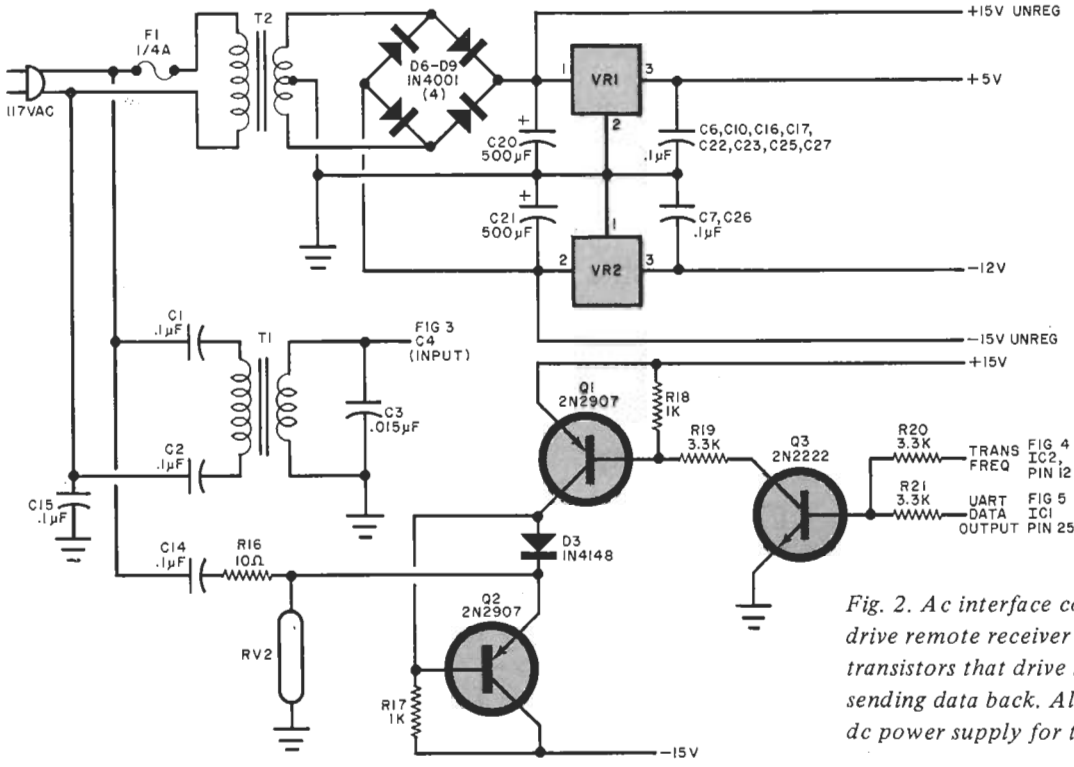


Fig. 2. Ac interface consists of T1 to drive remote receiver and three transistors that drive the ac line for sending data back. Also shown is the dc power supply for the remote.

status logic determines which of the two is acted upon.

A toggle command causes one of the two flip-flops to change states. This opens or closes a relay associated with that channel and controls the external device connected to that socket.

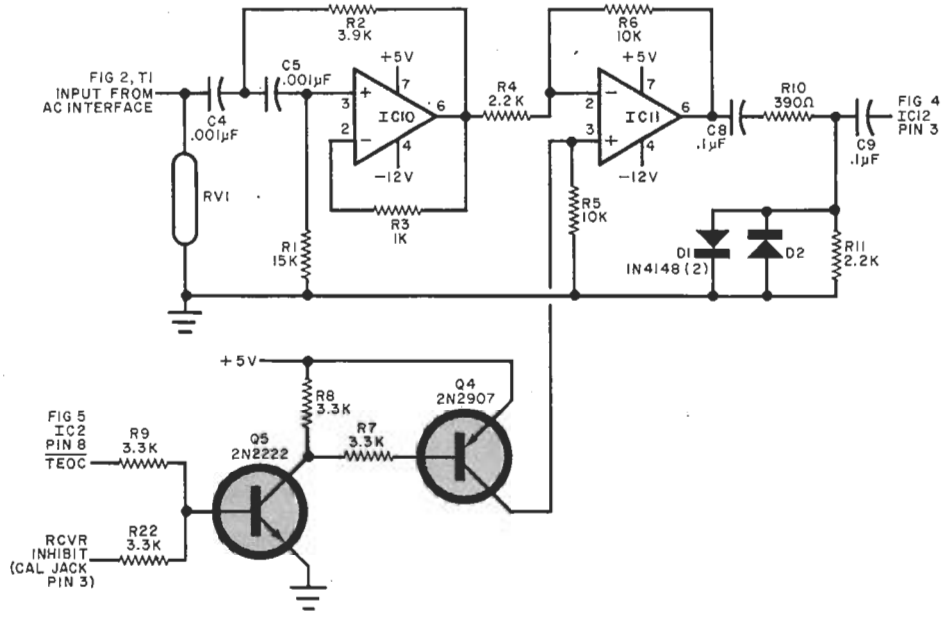
A poll command causes the status logic to place a word into the UART transmitter buffer in accordance with the following format. The first five bits of the data word contain the address of the remote channel being polled. The sixth bit contains the status of the remote device (on or off), while the seventh bit is set to zero to inform the system that a remote is responding to the controller. This indicates to all other remotes that the digital word on the ac line is not a command. The word is then formatted by the UART transmitter and sent via the ac interface to the power line.

AC Interface and Power Supply. This circuit (Fig. 2) forms the power supply to the electronic system and provides the interface between the digital receiver, the transmitter, and the ac line. Transformer *T2* and its associated components provide regulated +5 and −12 volts. Other components provide the unregulated ±15 V required by the various circuits.

Transformer *T1*, resonant at 50 kHz, provides the actual interface and isolation from the ac line.

Filter, Amplifier, Limiter. This circuit (Fig. 3) operates in exactly the same way as its companion circuit in the controller described last month. See the December issue for details.

Next month, Part 3 will conclude this article with the final circuit discussions, construction and software.



Using Existing House Wiring For Computer Remote Control PART 3

Construction and Software

BY DAN SOKOL, GARY MUHONEN, AND JOEL MILLER

THIS concludes the series of articles on computer remote control.

Data Recovery and Clock Generator. This circuit (Fig. 4) is also similar to its counterpart in the controller except that, in this case, the frequency-adjust potentiometer, *R15,* is a 10-turn potentiometer that is used for accurately synchronizing the frequencies of the controller and the remote.

UART. The data received from the controller is decoded and "unformated" in this circuit (Fig. 5). Data is also put into the proper format to be sent to the controller. The receiver section of the UART accepts the serial input data from the phase-locked loop (*IC12* in Fig. 4), and converts it to parallel data and status information.

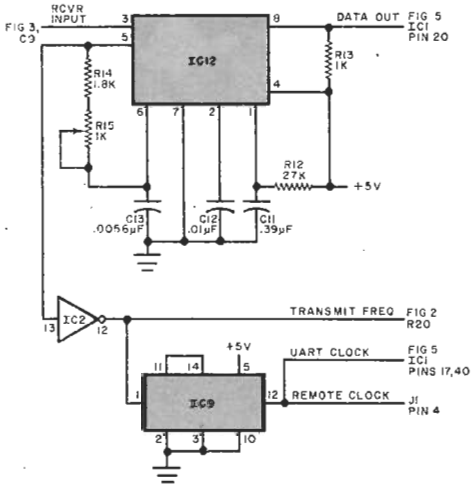


Fig. 4. Data-clock recovery is made by a PLL that delivers data output and a clock signal. The latter is divided by 16 for use in the UART.

The status information is used by the address and decode logic (Fig. 6) to indicate when data is available and if any errors occurred. Data at the receiver output is looped back to provide the first six bits of the transmit data word. The seventh and eighth bits of the transmit data word are originated by the address and command decode logic. The transmit side of the UART responds with data to the controller when the address and command logic gets a poll command.

The data word sent from the computer through the controller has a specific meaning to the remote. The first five bits (Table I) contain the address of the remote to be controlled while the sixth and seventh bits contain the command information. If the seventh bit is a zero, all remotes (up to 32 in the system) ignore the word. However, if the seventh bit is a one, the word is defined as a command to the remote whose address is contained in the first five bits. The sixth bit contains the actual command; and if it is a one, it toggles the remote channel addressed. If the sixth bit is a zero, the remote responds with poll information that informs the computer of its status (on or off). Bit 6 of the transmitted word contains the on or off information about the remote being polled (1 is on, 0 is off). Bit 7 is always a 0 during a poll.

Address, Command Logic. In the circuit shown in Fig. 6, the incoming data word is compared with that formed by the user-selected address jumpers to determine that it, and no other remote, is being addressed. The circuit then decodes one of the four possible commands and executes the decoded information. In *IC3* and *IC7*, the address is decoded and checked for errors, while *IC4* and *IC5* decode the specific command. Flip-flop *IC6* controls the state of outputs A and B, while portions of *IC8* provide the transmit side of the UART with correct poll information on the status of each side of the remote—circuits A and/or B.

Relay drivers *Q6* and *Q7* convert the outputs of the CMOS circuits to a sufficient power level.

Construction. Due to the complexity of the circuit, it is best to use a double-sided pc board as shown in Fig. 7. Note that, on the component layout guide, diodes are designated "CR" instead of "D" and integrated circuits are "U" in-

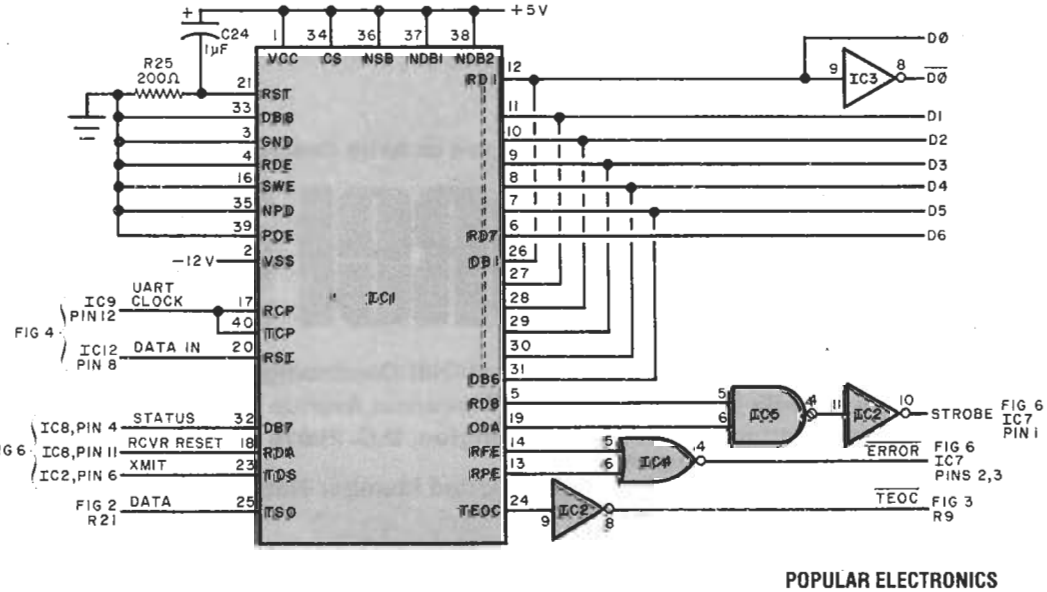


Fig. 5. UART (IC1) decodes data received by remote and formats it to be sent to controller. It also provides interface signals for other parts of circuit.

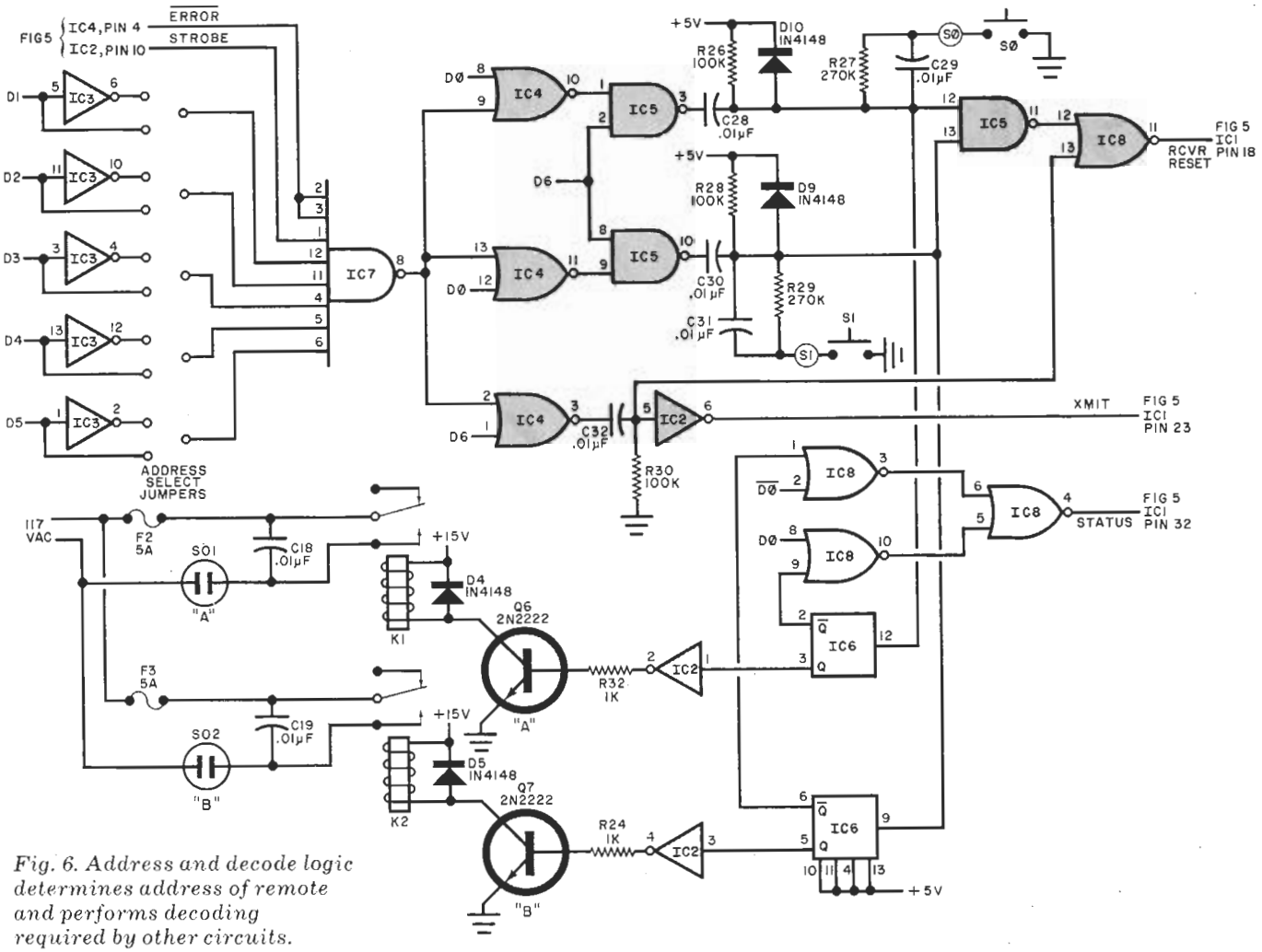


Fig. 6. Address and decode logic determines address of remote and performs decoding required by other circuits.

TABLE I

Information available at status port:

	MSB						LSB
Bit	7	6	5	4	3	2	1 0
	not	T	O	R	R	R	
Use	used	B	D	O	F	P	
	(always=1)	E	A	R	E	E	
Decimal	224	16	8	4	2	1	
Octal	340	40	10	4	2	1	
Hex	EO	10	8	4	2	1	

RPE = receive parity error. If this bit is a 1, then the character at the input port was received with a parity error. This bit clears when a word is received without error.

RFE = receive framing error. If this bit is a 1, then the character at the input port did not have the correct number of bits when it was received. This bit clears when a word is received without error.

ROR = receiver overrun error. If this bit is a 1, then the character at the input has overwritten the previous word (that is, the previous word was not read out prior to receiving this word).

ODA = output data available. When this bit is a 1, there is a character waiting to be read at the input data port. This bit clears when the input port is addressed.

TBE = transmitter buffer empty. This bit is a 0 during the time that the output port is busy. When it is a 1, data can be presented to the output port.

MSB							LSB
7	6	5	4	3	2	1	0
P	C	address					
O	T	0–63					
L	R						
L	L						

The first six bits contain the address of the remote being contacted. The poll and control bits will determine how the data is interpreted as follows:

Bit	7	6
toggle this remote	1	1
poll this remote	1	0
ignore this data	0	x

(x = don't care)

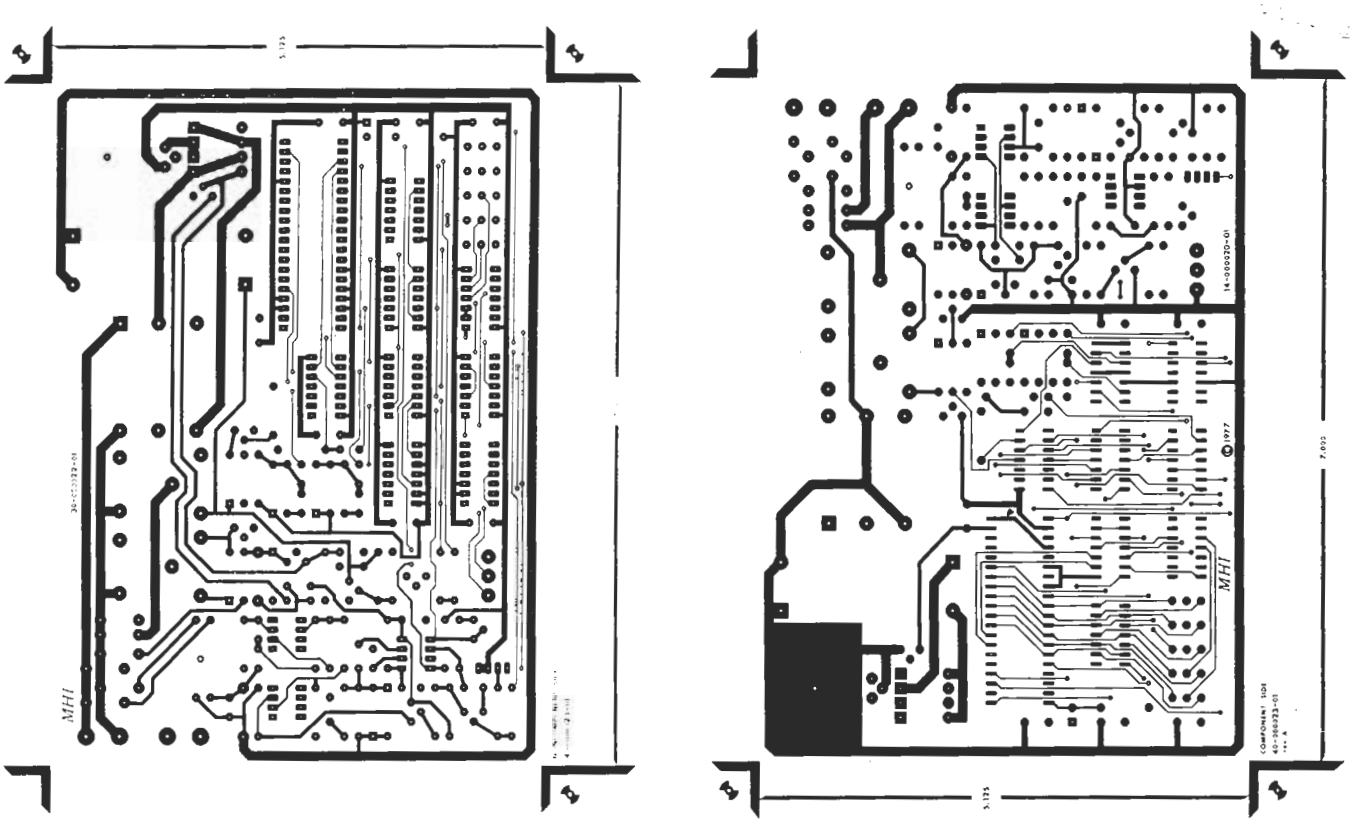
A toggle command will cause the remote to turn on (or off) depending on its previous state. For example, to toggle remote 41 (decimal) output 233 (decimal) to the controller's output port.

stead of "IC." Sockets may be used for all IC's. Regulator *VR1* is mounted with a conventional heat sink and *VR2* can be mounted directly on the board with the seven transistors. Observe the polarity of the capacitors and diodes and make sure of the orientation of the IC's before installation. Note also that the conductive pot covering transformer *T1* should be electrically isolated from the foil traces beneath it by means of an insulating mica washer.

External wiring is made in accordance with Fig. 8, which shows the connections to be made to the two manual override pushbutton switches and the two sockets to be controlled. These parts are mounted on the rear apron of the selected chassis.

The pc board can be installed in any convenient chassis. If a metal chassis is used, be sure the pc board and other components are well insulated from the metal structure. Keep in mind that there is 117 volts ac on the pc board.

Software. The Intelligent Remote



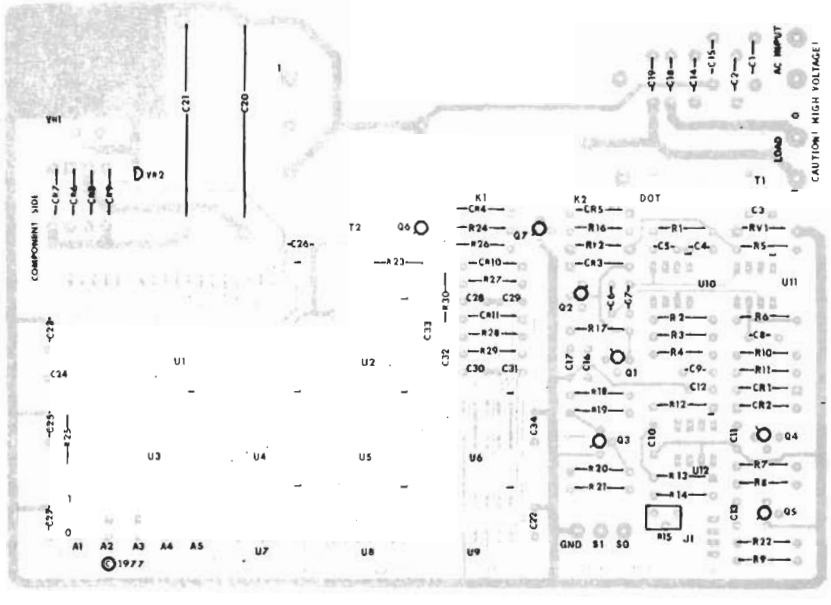


Fig. 7. Etching and drilling guides for the double-sided pc board are shown above half size. Component layout is shown at the left.

Controller is software-oriented to give the user broad flexibility in use. The software set is in two parts: (1) a group of subroutines designed to provide a format for the user to develop software particular to his application, and (2) a program to determine the background error rate and eliminate it. Both programs are written in BASIC for ease of use and are shown in Table II.

Subroutine 1 is a loop that waits for the transmit buffer to clear (TMBT = 1). Several assumptions are made, and the user may have to change these depending on where he has placed his board on the I/O map. These assumptions are: (a) the data input port is jumpered for 5; (b) the data output port is jumpered for 5; (c) the status port is jumpered for 4; (d) the remote is jumpered for addresses 52 and 53.

Subroutine 1 inputs the data at the status port and masks out all but TMBT (which is equal to 16). When TMBT is true, it returns to the main program. If TMBT is false, it remains in the loop to continue the search for a true TMBT.

Subroutine 2 is the polling subroutine that contains a series of conditional loops that wait for a valid response from a specific remote. The following variables in subroutine 2 have these meanings: (a) E is the main error flag and, if this routine returns to the main program with E = 1, then there was an error that could not be corrected within the constraints of this subroutine; (b) P contains the data that is transmitted to the remote plus 128 (bit-7 = 0); (c) X is the data in the status port that is updated during the subroutine; (d) D is the data received from the remote. It is valid when the sub-

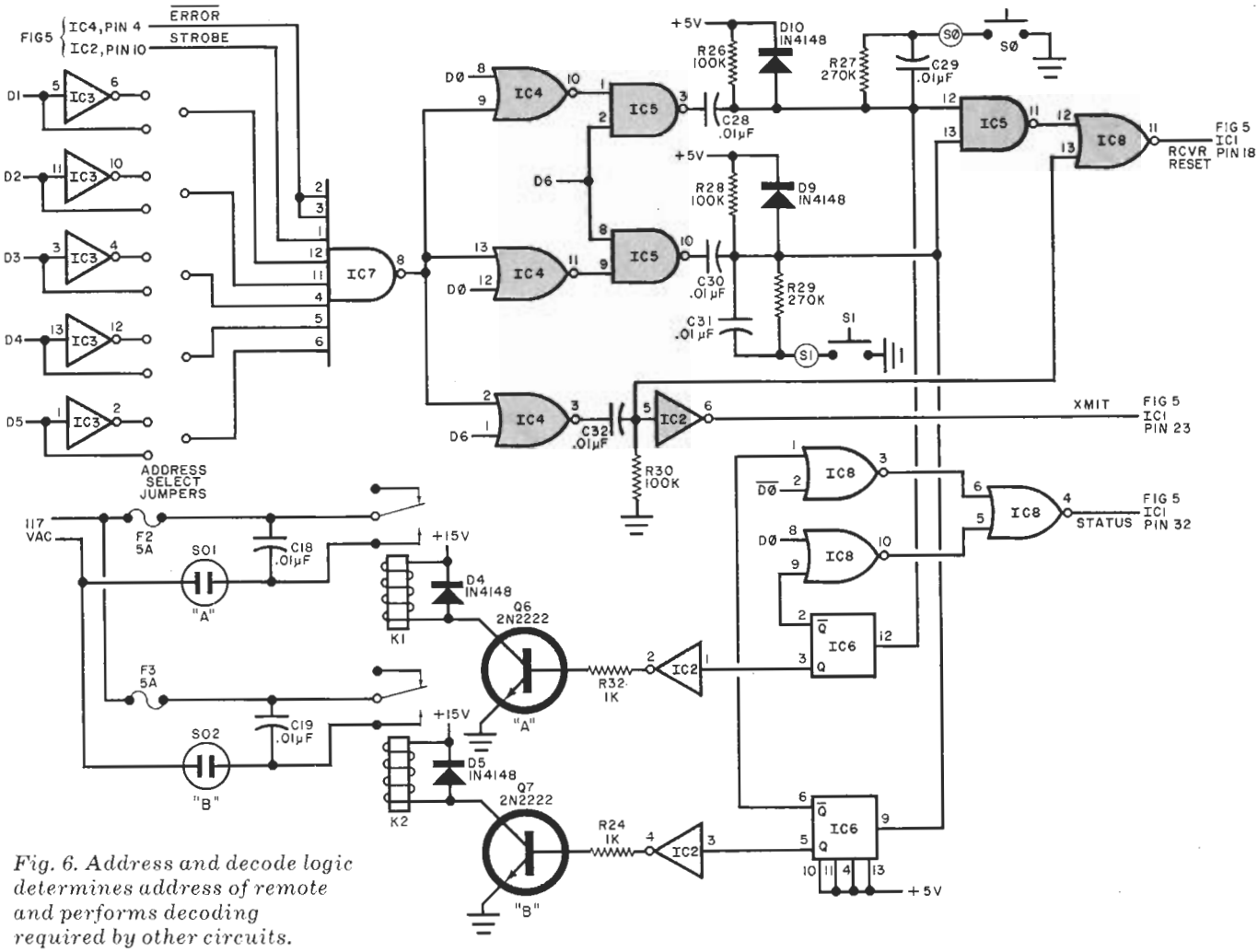


Fig. 6. Address and decode logic determines address of remote and performs decoding required by other circuits.

TABLE I

Information available at status port:

	MSB							LSB
Bit	7	6	5	4	3	2	1	0
	not		T	O	R	R	R	
Use	used		B	D	O	F	P	
	(always=1)		E	A	R	E	E	
Decimal	224		16	8	4	2	1	
Octal	340		40	10	4	2	1	
Hex	E	O	10	8	4	2	1	

RPE = receive parity error. If this bit is a 1, then the character at the input port was received with a parity error. This bit clears when a word is received without error.

RFE = receive framing error. If this bit is a 1, then the character at the input port did not have the correct number of bits when it was received. This bit clears when a word is received without error.

ROR = receiver overrun error. If this bit is a 1, then the character at the input has overwritten the previous word (that is, the previous word was not read out prior to receiving this word).

ODA = output data available. When this bit is a 1, there is a character waiting to be read at the input data port. This bit clears when the input port is addressed.

TBE = transmitter buffer empty. This bit is a 0 during the time that the output port is busy. When it is a 1, data can be presented to the output port.

MSB							LSB
7	6	5	4	3	2	1	0
P	C	address					
O	T	0–63					
L	R						
L	L						

The first six bits contain the address of the remote being contacted. The poll and control bits will determine how the data is interpreted as follows:

	Bit 7	6
toggle this remote	1	1
poll this remote	1	0
ignore this data	0	x

(x = don't care)

A toggle command will cause the remote to turn on (or off) depending on its previous state. For example, to toggle remote 41 (decimal) output 233 (decimal) to the controller's output port.

stead of "IC." Sockets may be used for all IC's. Regulator *VR1* is mounted with a conventional heat sink and *VR2* can be mounted directly on the board with the seven transistors. Observe the polarity of the capacitors and diodes and make sure of the orientation of the IC's before installation. Note also that the conductive pot covering transformer *T1* should be electrically isolated from the foil traces beneath it by means of an insulating mica washer.

External wiring is made in accordance with Fig. 8, which shows the connections to be made to the two manual override pushbutton switches and the two sockets to be controlled. These parts are mounted on the rear apron of the selected chassis.

The pc board can be installed in any convenient chassis. If a metal chassis is used, be sure the pc board and other components are well insulated from the metal structure. Keep in mind that there is 117 volts ac on the pc board.

Software. The Intelligent Remote

routine returns to the main program, if and only if, E = 0.

Variables C and C1 are the conditional counters that determine how many times the controller is allowed to try for a successful poll of the remote. These variables are absolutely necessary otherwise a failure in the remote (for example, a remote not connected to the power line) would keep the program in the loop and hang up the system.

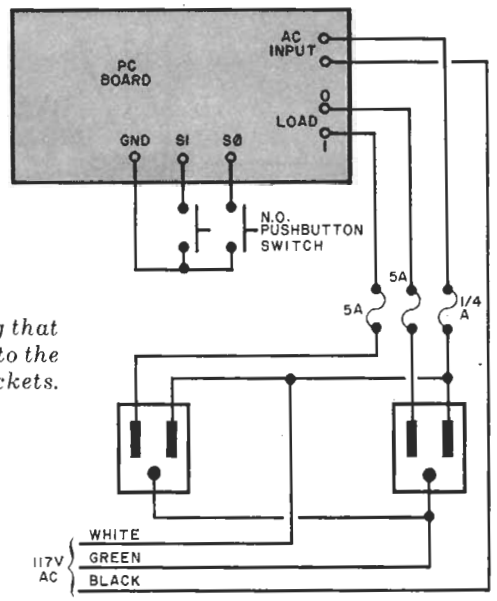


Fig. 8. External wiring that connects the pc board to the ac line and controlled sockets.

TABLE II

Subroutine 1

```
1000 X=INP(4) : IF (X AND 16)=16 THEN
     RETURN
1010 GOTO 1000
```

Subroutine 2

```
5000 C=0 : C1=0 : E=0
5010 OUT 5,P : GOSUB 1000
5015 GOSUB 8000
5020 X=INP(4) : IF (X AND 8)=8 THEN
     GOTO 5200
5030 C=C+1 : IF C > 5 THEN GOTO 5100
5040 GOTO 5020
5100 C1=C1+1 : IF C1 > 5 THEN GOTO
     5150
5110 GOTO 5010
5150 REM you can put an error flagging
     routine here
5160 E=1 : RETURN
5200 D=INP(5)
5210 IF(X AND 7) > 0 THEN GOTO 5100
5230 IF (D AND 63) <> (P AND 63) THEN
     GOTO 5010
5240 RETURN
```

Subroutine 3

```
8000 REM time waster
8010 FOR N=1 TO 15
8020 N1=N1+1
8030 NEXT N : RETURN
```

Main Program

```
 10 DIM R(2),A(2)
 20 A(1)=52 : A(2)=53
 30 FOR I=1 TO 2
 40 R(I)=A(I)+128+64
 45 P=R(I)-64
 50 GOSUB 5000 : REM Call the polling
    routine
 55 Z=Z+E
 60 T1=(D AND 64)
 70 IF E=1 THEN GOTO 50
 80 OUT 5,R(I) : GOSUB 1000
 90 GOSUB 8000 : REM time waster
100 GOSUB 5000
105 Z=Z+E
106 IF E=1 THEN GOTO 150
110 T2=(D AND 64)
120 IF T2=T1 THEN GOTO 80
130 CO +CO+1 : REM CO counts the num-
    ber of times through the loop
135 IF CO/25 <> INT(CO/25) THEN
    GOTO 150
140 PRINT "CYCLES =";CO;" ERRORS
    =";Z;" % =";(Z/CO)*100
150 NEXT : GOTO 30
```

Armed with this knowledge, the user can change the conditional loops until the point of zero errors is reached. Typical error rates with only one pass are 5 to 8%. With this as a background error rate, four passes will make the error rate less than 0.01%.

Errors induced by noise from the ac line are a fact of life. Fortunately, the computer can be taught to recognize and correct errors in transmission. If an error is detected by a remote, it ignores the command. If an error rate exists in response to a poll, it is easily detected.

For example, first test bits 1, 2, and 3 of the status port. If any of these three bits is a 1, then an error has been detected. Read the input port to clear the RDA bit, but ignore the data. If all three bits are 0, then compare bits 0 through 5 with the address polled. They must be the same. If not, re-poll the remote. If the bits are the same, read the poll bit to determine the actual status of the remote.

The actual error rate varies according to operating conditions. The conditions affecting the error rate are as follows: (a) Distance from the transmitter—the farther apart the controller and the remote, the weaker the signal. (b) Many residences are wired with 220-volt, 3-phase power, which means that there are two 117-volt circuits available. The transmitted signal can be detected on the other phase, but greatly attenuated. A 0.01-μF, 600-volt capacitor across the 220-volt line will correct this problem. (c) High-amplitude, wideband noise, generated by older brush-type ac motors, can cause problems. If you can't replace the motors, then you will have to live with the problems. (d) Impulse noise caused by high-current inductive devices (refrigerators, air conditioners, etc.) when they turn on and off is a random factor that can produce single-bit errors. Fortunately, this type of noise is just as rapidly attenuated as the useful digital signal. (e) Triac noise, usually produced by poorly designed light dimmers, can raise the error rate.

The variables in the main program are as follows: (a) A(I) is an array that contains the addresses of all remotes; (b) R(I) is the toggle command for the remote channel and is equal to A(I) plus 128 plus 64; (c) P is the poll command for each remote and is equal to R(I) minus 64; (d) T1 and T2 contain the poll status of the remote before and after it has been toggled; (e) D is the data from the remote; (f) Z is the total number of errors that have been detected. ◇

POWER PHASE NOTES

A minor error in the "Computer Remote Control Project" (February 1978) requires clarification. It is stated that many residences are wired for 220-volt, three-phase power. This is not correct. Most U.S. homes are wired with 120/240-volt, *single*-phase power and would require the 0.01-μF capacitor as stated. If a three-phase wye system is provided, two capacitors would be required to bridge all three hot lines. In this case, the phase-to-phase potential is 208 and 120 volts to ground and loads on opposite phases are being controlled. If a delta system is employed, the voltage on the "high leg" would be in excess of 208 volts to ground; naturally, this leg would not be used for switching purposes in this project. —*Gary R. Knight, Tampa, FL.*

AUTOMATE YOUR HOME

Tailor this programmable control system to do the job

NOEL NYMAN

YOUR HOUSE CONTROLLED BY COMPUTER! What was science fiction a few years ago is a practical reality today . . . or is it? There are several problems you'll encounter if you computerize your home. This article deals with a system that tries to overcome some of them.

Connecting a computer to house electrical systems requires special interfacing circuitry. The computer needs inputs to tell it what's happening in the outside world. Also, its outputs must be connected to the devices you wish to control. The input and output circuits should isolate the computer from the house line voltage, and should also allow the computer and the people living in the house to control the same devices without disturbing each other.

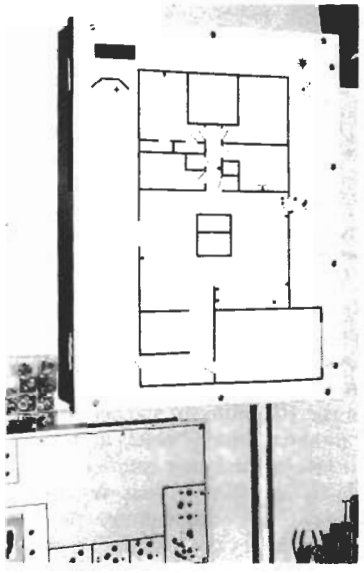
There are ready-made interfaces and kits that do this. Most of them use modulated radio waves carried over the house wiring. Such circuits are expensive and complex. The assumption is that most people don't want to "run wires all over the house." While this may be true of many homeowners, electronics activists enjoy spending time on their projects. Running wires does take time, but it also drastically reduces the cost of interfacing. The noise problems that sometimes plague even sophisticated RF systems are eliminated, and the interface we describe here allows both computer and human control of the same device without disturbing either living patterns or computer programs.

Another set of problems concerns the computer itself. Hobbyist machines are usually short on memory. We all need more memory than we have, and even a simple program for a few appliances takes an important part of that limited space. The computer must run all day every day without shutdown to maintain control. To time electrical systems, a real-time clock is needed. This is an expensive option in machines designed for it and difficult to add to others. And the biggest problem is very basic to many of us . . . we don't own a computer yet!

The system described here is a dedicated control. Rather than tie up a computer, the control circuits are designed around discrete logic. The system is "programmed" using wire and a patch panel rather than software. This drops the cost dramatically, especially if surplus parts are used. The interface uses simple circuits with few parts. Even if all new components are used, each device-controlling circuit can be built for under $10. If you don't own a computer, you might want to build an entire system similar to the one described here. If you do own a computer, the interface circuits may help you expand your machine to control your house.

A basic part of the control logic is the real-time clock. Clock IC's are readily available; but to control electrical devices you need separate outputs for hours, minutes and, if used, seconds. These outputs aren't readily available from conventional clock IC's. I built up a clock using counters and decoders. For reasons that will be explained later on, I used a 24-hour format and I needed only hour and minute outputs.

Figure 1 is the block diagram of the house-control system. A divide-by-60 counter is clocked by 1-second pulses to count 60 seconds. A divide-by-10 counter clocks a divide-by-6 counter to get the divide-by-60. Any other arrangement can be used that achieves the same result; for example, a single-IC programmable counter.



FRONT PANEL of logic system has house layout. LED's show status at different points.

I got the 1-second pulse from the flashing colon on a digital clock display, and used a resistor voltage-divider network to drop the pulse to the proper voltage level. Another approach would be to use a low-voltage transformer and Schmitt triggers to provide 60-Hz pulses and a divide-by-60 counter to get 1-second pulses. Both methods provide power-line accuracy that is adequate. A DPST switch changes the real-time clock input to a free-running astable with adjustable frequency that provides setting pulses.

The seconds counter clocks another divide-by-60 that counts minutes. Since I wanted control that is accurate to individual minutes, I used decoders for the outputs of both sections of this counter. The first section provides a logic 1 or logic high for each minute as it's counted (the 7442 decoder outputs go low or to a

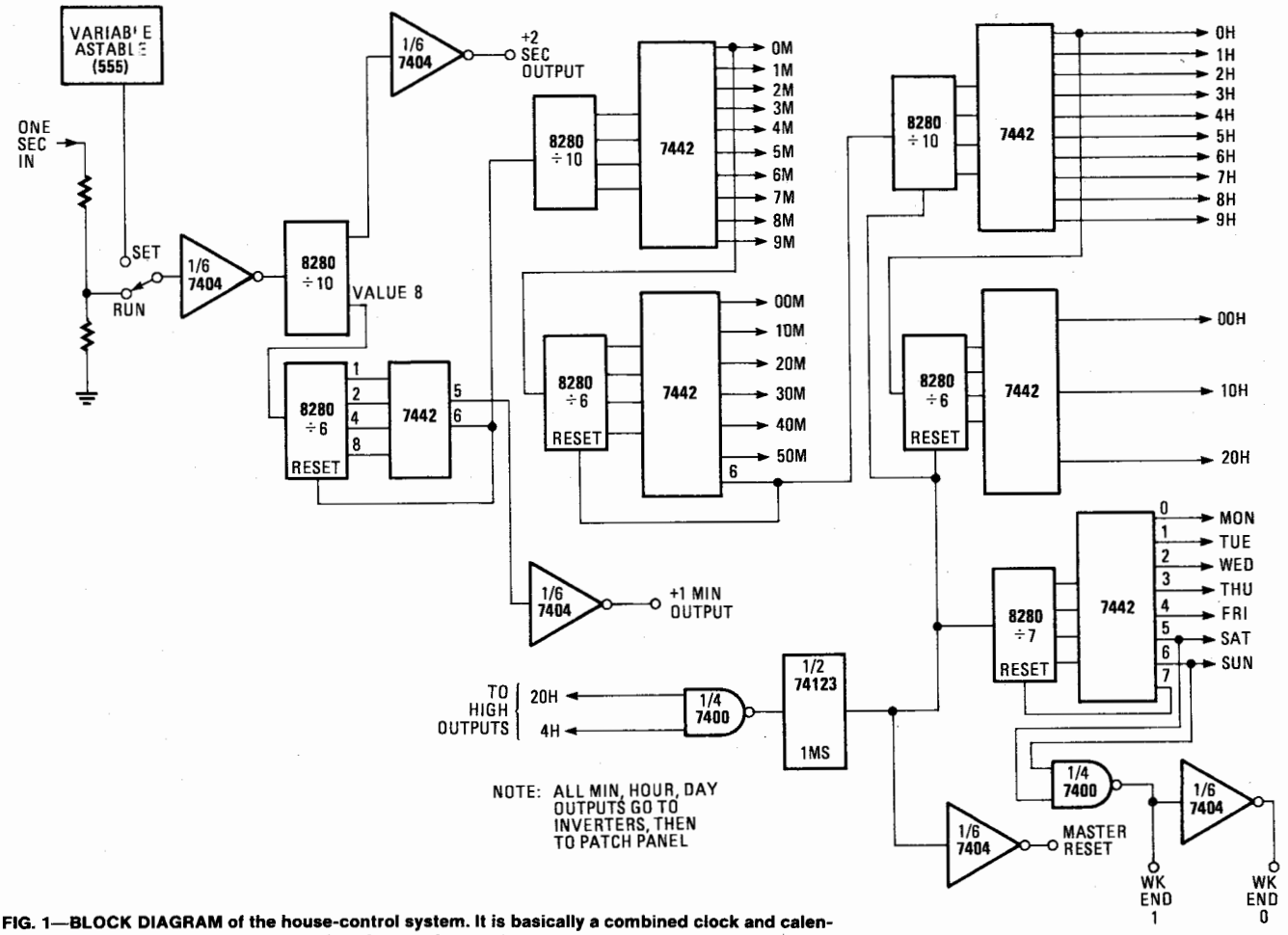


FIG. 1—BLOCK DIAGRAM of the house-control system. It is basically a combined clock and calendar providing logic outputs for seconds, minutes, days and hours.

logic zero that is inverted to a logic 1). When the counter resets at a count of 10, it clocks the divide-by-6 counter to count 10 minute intervals. The decoder provides the 10's minutes signals. When the 10's minutes counter resets at a count of 60, it clocks the hours counter. This is a divide-by-24 counter that is otherwise similar to the minutes counter. At a count of 24, the hours counter fires a one-shot that resets the hours counter, clocks a divide-by-7 counter and provides a master reset pulse that can be used by other circuits. The divide-by-7 counter is decoded and provides day-of-the-week outputs.

Figure 1 shows that the SAT and SUN outputs are combined into WEEKEND zero and WEEKEND 1 outputs. To provide a pulse every minute is handy, and the "50" output of the seconds decoder was used to provide a long pulse width. Pulses with 2-second periods are handy for flashing-alarms and this signal, labeled +2 SEC, comes off the seconds counter.

The outputs of the minutes, hours and days counters are connected to LED's (not shown) as a visual indication to help in setting the clock. An inverter changes each signal to a logic 1, and each signal is connected to a patch panel.

I used a surplus wire-wrap board for this project, with the IC sockets forming the patch panel. Small-gauge wire fits nicely into the individual socket holes. The logic-circuit inputs are also connected to sockets, and "programming" the system is a simple matter of inserting wires into the appropriate sockets. If you plan never to change the programming, you could omit this procedure and wire the clock directly to the logic.

The simplest form of computerized house control turns an electrical device on and off once every 24 hours without any need for human control. Let's take for example a hot-water heater. As an energy-saving measure you want to turn the heater off late at night when hot water isn't needed and turn it on in the morning early enough for a hot shower.

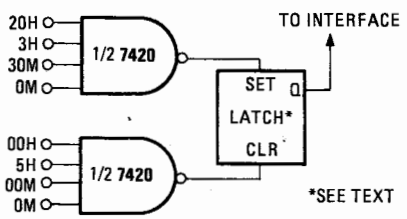


FIG. 2—LOGIC to control water heater. Top gate turns it on, lower gate turns it off.

Figure 2 shows the logic to control this. Two four-input NAND gates have their inputs connected through the patch panel to the clock. When all four inputs of one NAND gate go high, a latch is set, and the

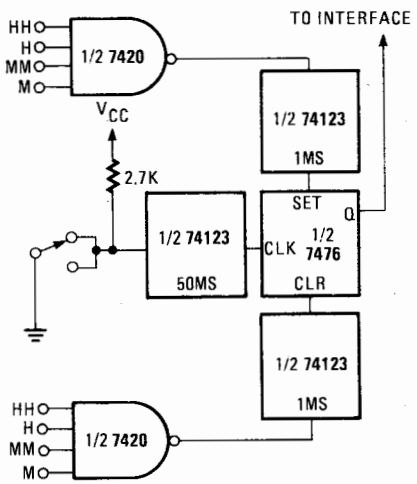


FIG. 3—FOR MANUAL OVER-RIDE, one-shots are used between gates and clocked latch.

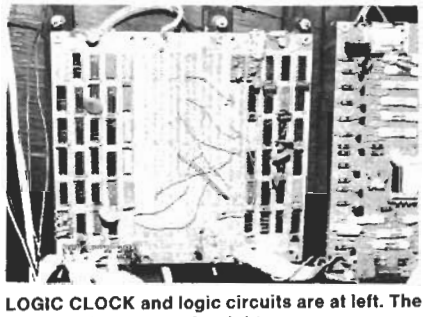
latch output turns off the heater via an interface (described later). When all four inputs of the other NAND gates go high, the latch is cleared and the heater turns on. If you want the heater to go off at 11:30 pm (2330 hours) and on at 5 am (0500 hours) the inputs are patched to the clock signals, as shown in Fig. 2. Using a 24-hour format eliminates the need for an AM/PM input for each gate. If you want to time accurately to the second, you need six-input NAND gates. If timing to a 10-minute interval is OK (as

it usually is for these applications), three-input NAND gates will suffice. The fewer the inputs, the fewer total IC's you'll need. The latch can be formed either from part of a flip-flop or from two cross-coupled NOR gates.

Most household devices are controlled by humans, however, and some provision must be made for people to override the logic circuits. When you start to computerize your home, you may find that selling the idea to your family is the hardest part of the project. Your spouse may have enjoyed the stereo amplifier and thought the electronic dice game was cute, but may be very reluctant to have you tinker with the house lights and electrical appliances. Therefore, anything you can do to make these controlled devices behave normally when operated by people will help.

To allow human control, several gates are added to the basic circuit, see Fig. 3. The input NAND gates now set and clear the latch via one-shots. The latch has been changed from an R–S flip-flop to a clocked flip-flop, and a switch and a one-shot clock the latch. The output goes to an interface as before. The NAND gates set and clear the latch; but any time the clock one-shot is fired by the switch, the latch changes state. This means that people can turn the controlled device on or off regardless of what part of the program the logic is in. Clocking the latch does not change the logic cycling, and the device still turns on and off as programmed.

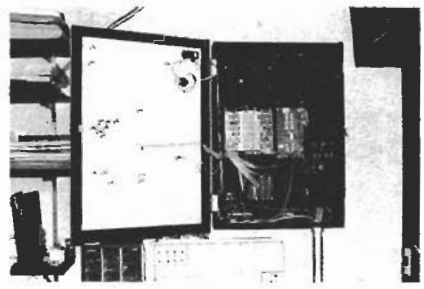
If you use a computer instead of IC logic for your system, you can replace the NAND gate outputs with computer outputs. The one-shots for the set and clear signals are used because people can't clock the latch while the set or clear inputs are low. Not using the one-shots



LOGIC CLOCK and logic circuits are at left. The interface board is at far right.

means there would be two 1-minute periods each day when the device couldn't be human-controlled. The one-shot on the clock input has a long period (50ms) and prevents extra triggering impulses resulting from switch-contact bounce. The other one-shots have shorter periods—about 1 ms. Only a brief switch pulse is needed on the one-shot that clocks the latch.

Most house lights and switched outlets use wall SPST toggle switches. Lights that are turned on or off from two locations use SPDT switches (also called three-way switches) that look the same



DISPLAY PANEL is hinged to permit access to logic, patch and interface boards.

except for the ON and OFF labels. These SPDT's are break-before-make or non-shorting switches. As they are thrown, both circuits are open briefly. You can replace the existing wall SPST switch with an SPDT switch and clock the latch without running additional wires to the wall box. The cable that goes to the existing switch is located and cut at a convenient point. The SPDT switch is installed with one wire going to the common connection (a black screw), and the other going to *both* of the other connections. **Turn the AC power off before working on the house circuits. Follow local electrical codes in terminating the unused end of the cable.**

The wire spliced to the switch side of the cut cable carries logic-level voltage and can be small-gauge wire. The input of the one-shot is pulled high by a resistor and grounded through the SPDT switch. When the switch is thrown the ground circuit opens briefly, the one-shot input is triggered when it is pulled high by the resistor, and the latch clocks. When a person operates the switch, the wall system seems normal; throwing the switch changes the on/off state of the device, yet the logic still has control. If two or more on/off cycles are needed, more four-input NAND gates are used with their outputs OR'ed to the one-shots.

Other applications

There are other useful inputs besides time and wall switches. You may want to trigger circuits when doors are opened or closed. Commercial magnetic-reed switches are available for this purpose, but it is easy to make your own by removing the switch section from commercial magnetic-reed relays. If you can remove the moldings from around doors and windows, you can mount the switches behind them and run the wires where they won't

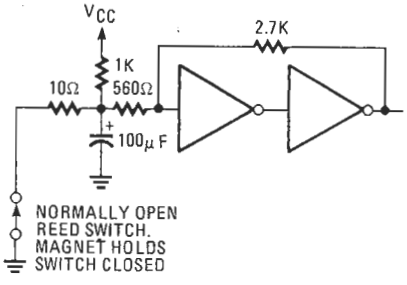


FIG. 4—MAGNETIC SWITCHES trigger the control logic when doors are opened or closed.

be seen. The magnets controlling the switches can be mortised into the door or window so that the switches are completely invisible. Figure 4 shows a typical input circuit for a magnetic switch. The capacitor and resistors condition the input, and the resistor feedback across the inverters provides a snap action or Schmitt trigger. The inverters outputs can provide OPEN zero and OPEN 1 signals to the patch panel.

A light sensor can be used to sense that a lamp is on without having to install extra wires to the lamp itself, (a typical circuit is shown in Fig. 5) A photoDarlington transistor (for example, an FPT 120 625) is mounted near the floor and aimed toward the lamp being sensed.

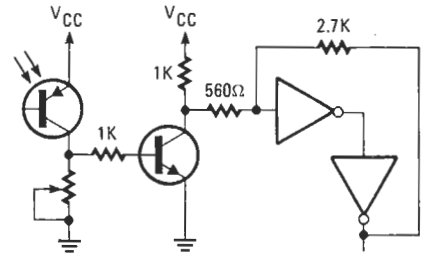


FIG. 5—LIGHT SENSOR may be a phototransistor or a sensitive photo Darlington.

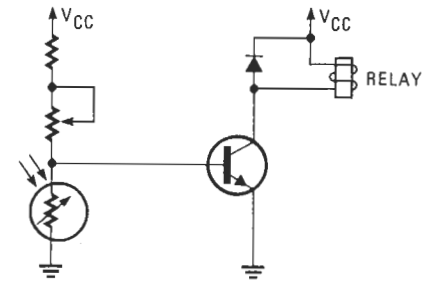


FIG. 6—PHOTORESISTIVE CELL has enough sensitivity for use in dusk/dawn sensor.

A dawn/dusk signal is also useful. However, the Fig. 5 circuit may be too sensitive for this input. Figure 6 shows a simple photoresistive cell that is mounted outside the house facing west. The resistors in series with the cell are selected to bias-off the transistor during the day. As the sky darkens, the photocell resistance goes up until the transistor conducts sufficiently to energize the relay. The variable resistor is adjusted to trigger the relay at the desired darkness level.

Circuit applications

Now, for some actual circuits that I use. In my household we must take our garbage cans out on Thursday mornings. It was easy for us to forget to do this, and became a good logic-circuit application. Figure 7 shows the circuit I used. A three-input NAND gate (IC1) receives input signals from DOOR OPEN 1, a clocked-latch Q output and the real-time-clock THURS output. On any day but Thursday, the THURS output is low and nothing happens. At midnight on Wednesday, THURS goes high. At 4 am, 4H and zeroH are both high, output IC3

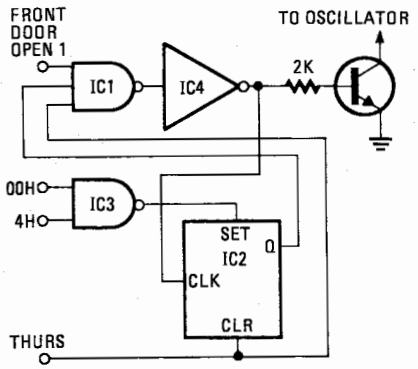


FIG. 7—GARBAGE DAY circuit sounds alarm as you start out on day garbage is collected.

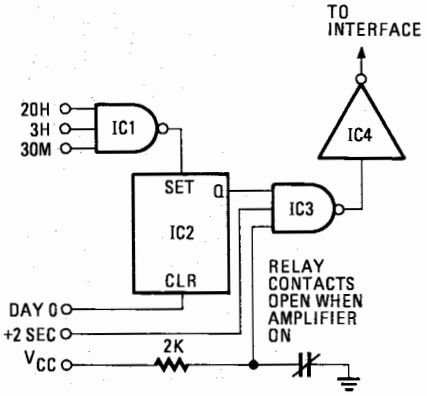


FIG. 8—STEREO ON reminder flashes bedside lamp if stereo is advertently left on.

goes low and sets IC2. With the latch set, IC2's Q output goes high. Two inputs to IC1 are now high.

When the door is first opened in the morning, the third IC1 input goes high; its output goes low, is inverted by IC4 to bias-on a transistor that powers an oscillator outside the front door. The sound of the oscillator reminds us to take out the garbage. The clocked input to the latch is used to prevent the oscillator from being activated all day Thursday whenever the door is opened. The clock triggers on a negative-going pulse edge so the high-going inverter output that turns on the oscillator doesn't affect the latch. When the front door is closed, the inverter output goes low, turning the oscillator off and clocking the latch. The Q latch output goes low and pulls one of IC1's inputs low, preventing further circuit action. At midnight the THURS signal goes low, holding the latch in a cleared state and the IC1's THURS input low for the next six days.

Here's another circuit I devised: I have remote speakers in my workshop, and I used to leave my stereo system on inadvertently. Figure 8 shows a circuit that flashes a night light in the master bedroom if the stereo is still on after 11:30 pm. IC1 sets a latch at 11:30 pm (23.30) and a low-going signal at dawn clears the latch. The Q output of IC2 drives one input of IC3 high. Another input of IC3 goes high when the stereo is on. This is accomplished by using a 120 VAC relay located at the amplifier. The relay is plugged into the switched outlet on the amplifier, but any appliance can provide an "on" signal by using a relay operated by the appliance's on/off switch. The wires running from the relay to the logic circuit can be small-gauge wires since they carry only low voltage.

With the "stereo-on" input high and the latch input high, IC3's output goes low and high alternately with the +2SEC alternations. The NAND gate output is inverted and interfaced to the night light.

The most complex circuit, shown in Fig. 9, in my system controls the porch light and a lamp near the front door that is operated via a wall-switched floor outlet. Although the circuit may seem very involved at first, you can use its compo-

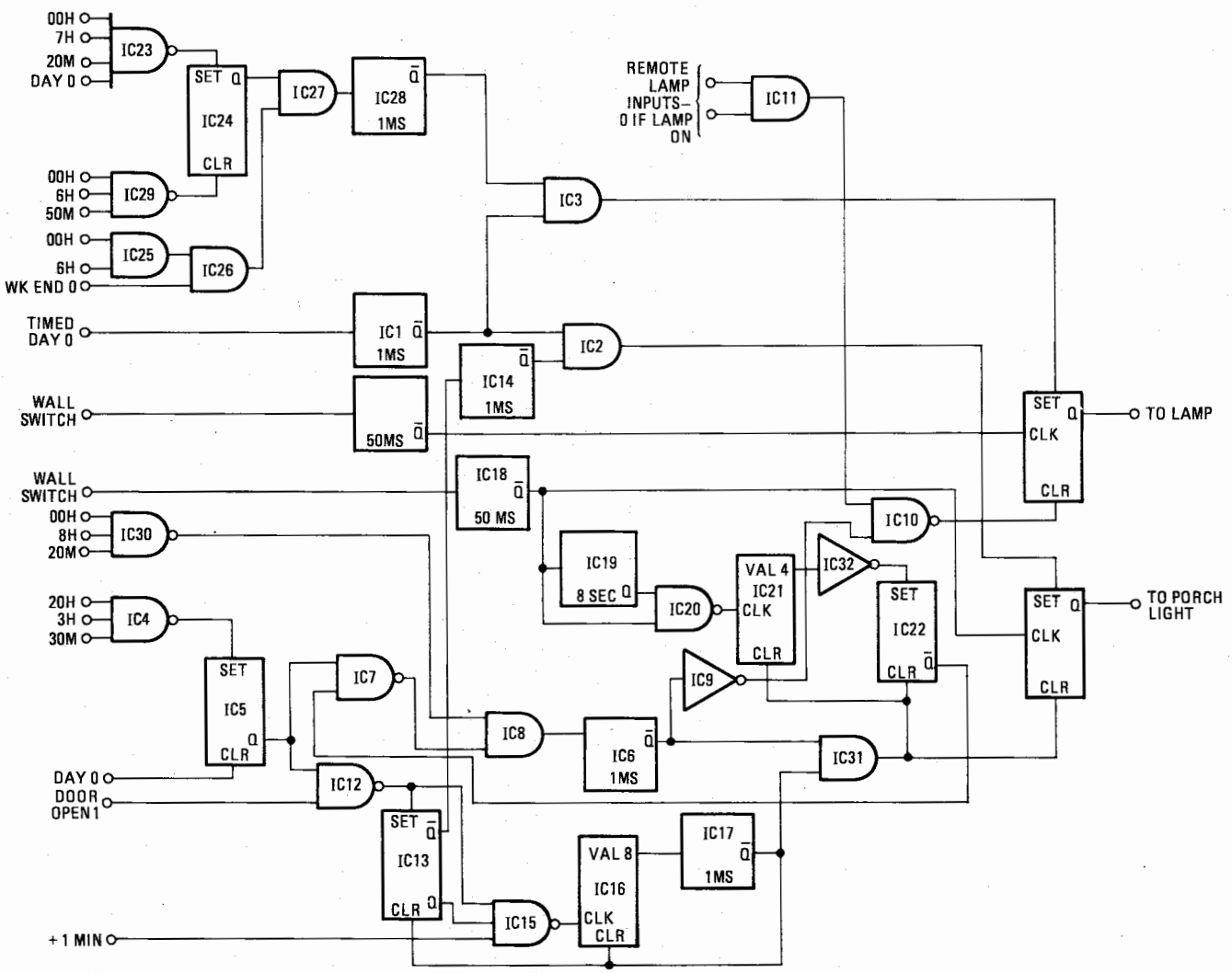


FIG. 9—COMPLEX CONTROL turns porch lamp, and one other lamp, on at 11:30 pm and off at dawn. Circuit is not fooled by shadows or bright light flashes.

nents or develop ideas from it to achieve almost any kind of sophisticated control for a house electrical system. This circuit grew one stage at a time and its construction may be more involved than necessary for your particular application. I suggest you examine it carefully to see how you could save IC's in your own system.

Figure 9 shows the complete porch light and lamp circuit. Using the SPDT/one-shot technique described earlier, the wall switches still allow human control of the light and lamp. The first logic control turned the light and lamp on at dusk and off at 23:30. To avoid false triggering, the dusk sensor described earlier is connected through inverters (providing DAY 1 and DAY ZERO signals) to a counter; see the circuit shown in Fig. 10. The counter is clocked by the +1 MIN signal and after 8 minutes, sets a latch that provides a TIMED DAY zero signal that goes high 8 minutes after dusk. Dawn or the DAY zero signal clears the latch so that dark periods of less than 8 minutes or cloud shadows don't turn on the lights.

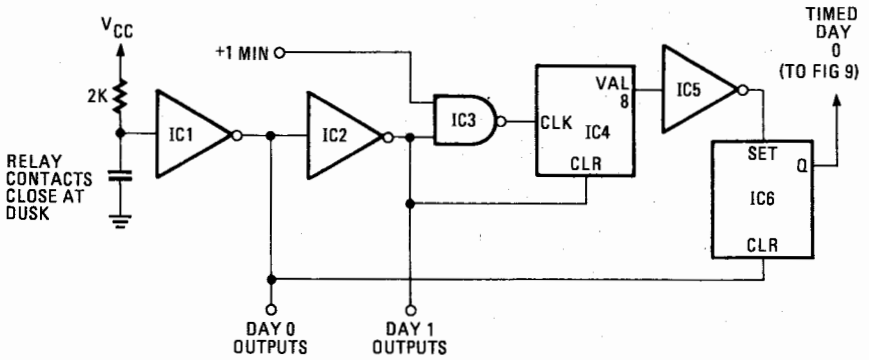


FIG. 10—THE LOGIC DELAY circuit prevents the porch light from coming on until 8 minutes after dusk. Shadows or dark periods shorter than 8 minutes don't turn on the lights.

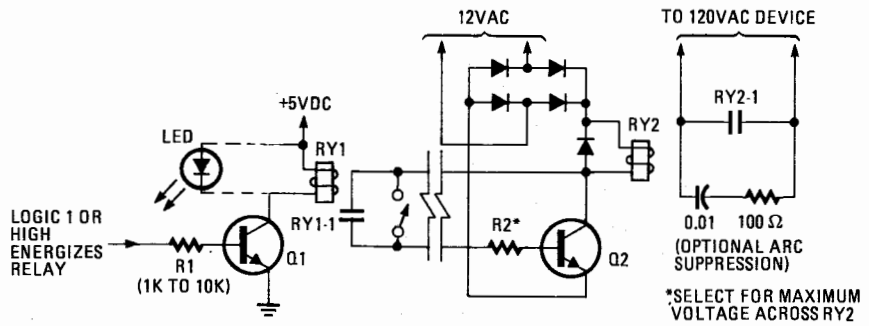


FIG. 11—HOW INTERFACE RELAYS provide total isolation between the logic voltages and the higher voltages used to power the control devices or stages.

The TIMED DAY 0 signal sets the porch-light latch and the lamp latch through one-shot IC1 and AND gates IC2 and IC3 (Fig. 9). In the descriptions that follow, we'll assume that any other gate inputs are as they must be to allow the gates to function as described. AT 23:30, IC4 sets latch IC5, and its Q output fires one-shot IC6 through IC7 and IC8 to clear the porch-light latch through IC31 and the lamp latch through IC9 and IC10.

In our home whenever we have visitors in the front room at 23:30, since it would be rude to have the indoor lamp turn off, I added another circuit to prevent this. One of two other lamps are usually on in addition to the front door's logic-controlled light. One of these lamps is wall-switch controlled, and I placed a relay in parallel with it to provide an "on" signal to the control logic. The other lamp is of a more conventional design. Instead of adding a relay and wires to the lamp, I used the remote light sensor with a photoDarlington input. If either of these lamps is on, one IC11 input is low. This makes one IC10 input low and prevents the 23:30 one-shot pulse from passing through IC10 to clear the lamp latch. The lamp stays on at 23:30 even though the porch light goes off.

When the guests leave, it's convenient to have the porch light come on, stay on until they reach their cars, and then turn off.

To accomplish this, the IC5 Q output sends one input of IC12 high. The other input goes high with FRONT DOOR OPEN 1. The IC12 output sets latch IC13 and fires one-shot IC14 so that the porch-light latch is set through IC2. The porch light then comes on. When the front door is closed, IC12's output goes high. The IC13 latch is still set and IC15 will go low once-per-minute with the +1 MIN input. This output clocks counter IC16, which counts minutes. After 8 minutes the counter fires one-shot IC17. This clears the porch-light latch through IC31 and clears IC13 and IC16. The porch light stays on for 8 minutes after the door is closed and then turns off.

I also wanted to be able to leave the two lights on past 23:30 in case someone stayed out late. Ideally, this circuit should be enabled easily without additional switches. I decided to use the porch-light switch. When the porch-light wall-switch one-shot IC18 clocks the porch-light latch, it also fires one-shot IC19, which has a period of about 8 seconds. When IC19's output Q is high, it allows the IC18 pulses to pass through IC20 to a counter, IC21. If the switch is thrown four or more times during the 8-second period, IC21 sets latch IC22. The latch $\overline{Q}$ output controls the path of the 23:30 pulse. With one input of IC7 low, its output stays high at 23:30, and the porch light and lamp stay on. When the front door is opened and then closed, the 8-minute counter described earlier cycles and clears the porch-light latch again. However, this pulse also clears counter IC21 and latch IC22, and places latch-output $\overline{Q}$ high. Latch IC5 output Q is still high, so IC7 goes low and fires one-shot IC6, which also clears the lamp latch. Eight minutes after someone returns home and closes the door, the porch light and lamp both go off.

Memory circuit

The final circuit (see Fig. 9) is a memory circuit that is used to retain information from the previous day. During the dark winter mornings, when I go to work, I like to keep the front room lamp on when I leave. On weekends or mornings when it's light outside, the lamp doesn't have to come on at all. In the memory circuit, IC23 sets latch IC24 if it is not yet dawn at 7:20 am (DAY ZERO is still high). Nothing else occurs that day, but at 6 am the following day, IC25 output goes high. If the day is a weekday (WEEKEND ZERO isn't low), IC26 goes high and, through IC27, fires one-shot IC28, which sets the lamp latch. At 6:50 am every morning, latch IC24 is cleared by IC29 so the lamp only goes on at 6 am if it had been dark at 7:20 the previous day. Then, IC30 fires one-shot IC6 to turn the lamp off at 8:20 am; and IC30 acts as a master backup circuit to the entire turnoff system. If the porch light was enabled to stay on past 23:30 but the front door wasn't opened all night, IC30 will turn off the porch light too.

Relays

The circuits I chose to interface to the 120 VAC house wiring use relays. Opto-isolators would work as well for the logic-voltage sections of the interface. Magnetic-reed relays that operate on low voltage draw little current and are available from surplus outlets or by mail order. Relays offer total isolation between the logic voltages and the higher voltages used in the following stages. Using reed relays involves only a few components (see Fig. 11). A small-signal NPN transistor, Q1, provides a ground path for the relay coil. When the logic-level input signal goes high, it biases Q1 on through

a current-limiting transistor and energizes relay RY1. Don't forget to include the diode across the relay coil to limit back EMF surges. The LED shown in Fig. 11 is optional for a display of which logic systems are ON.

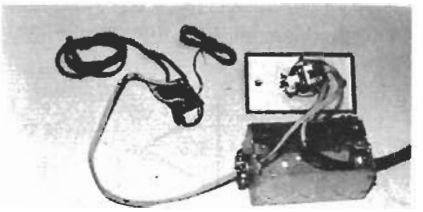
Reed relay contacts have a current rating that is much too low to operate 120-VAC systems directly. Having the 120 VAC control in a central location means running grounded NM house cable from many locations; this adds to the project cost. To avoid this I used a second relay at or near the controlled device. Using a transformer with a high-current secondary, I ran a 12 VAC line wherever necessary throughout the house. Near the controlled device I installed the second relay, RY2, which is energized by Q2 when the logic-controlled relay RY1 contacts close.

You may need a small filter capacitor across the bridge output. It may be less expensive to use four separate diodes than a commercial bridge assembly. A supply of 50 PIV at 1 A or more is fine for these diodes. Although it is possible to use any 12 VDC relay whose contacts will handle 120 VAC with enough current rating, I found that 24 VAC relays work very nicely on 12 VDC. These relays run cooler and are easier to obtain on the surplus market. The ones I used draw 0.25 A at 12 VDC. I used surplus power transistors rated at about 20 watts to insure that they would run cool. This part of the interface may not be easy to reach once installed, and overrating component values will minimize failures.

Relay RY2 will probably have several poles of contacts; use all of them in parallel to reduce on contact arcing. You may want to use a 0.01-μF, ceramic disc and 100-ohm, ½-watt resistor in series across the relay contacts to reduce both arcing and electrical noise. Then, decide whether the device will be on or off most of the time. If the device will usually be off, wire it to the normally open contacts; energizing the power relay will turn the device on. If the device is on most of the time, wire the contacts normally closed. The logic signal will energize the relay to turn the device off. You can use this circuit to control the hot water heater.

The 12 VAC supply should run through relatively heavy wire—16-gauge or 14-gauge lamp cord is recommended. Although local electrical codes may cover such low-voltage wiring, the wires can usually be run in any convenient manner without using electrical boxes, etc. The connections to the power relay, however, are 120 VAC and *must* conform with electrical codes. In general, this means that the relay must be mounted in an approved electrical box that may need to be properly grounded. There may be restrictions on how the wire can be run from the relay to the controlled device; check your local codes. You may want to place your circuitry inside an external cabinet near the controlled device. The cabinet would have its power cord plugged into a wall receptacle, plus a convenience outlet to control the device. This would avoid most code restrictions.

If you've never done any house wiring or if you plan to remove drywall or plaster to run new wires, check out do-it-yourself books. These booklets are available in many hardware, drug and grocery stores, and are geared to the novice. You'll find the circuit descriptions are oversimplified, but the books do contain step-by-step ideas on how to run new wires. Running wires under or over the rooms is obvious if you have an attic or a basement. But it may not have occurred to you to run wires behind baseboards and through walls using easily patched small holes. I found these do-it-youself booklets invaluable, and so far I've installed my system using over ten 120 VAC interfaces without having to make any noticeable changes in the living areas.



CONTROL RELAY used as interface to line-operated devices is in approved enclosure.

The logic used

The logic circuits I used are TTL, but CMOS circuits should work equally well. I used TTL circuitry because I had the IC's on hand. A system as complex as this has a high IC count, and TTL circuits will require a power supply of several amps at a tightly regulated 5 volts. Fan-out to the patch panel has to be considered with TTL circuits too, so CMOS IC's might be a better choice for a large system.

Helpful hints

If you do choose TTL IC's, here are some tips that might be helpful. I used 74123's for the one-shots instead of the more popular 555. You'll find some one-shots triggered by high-going pulses and others by low-going pulses. You'll also find that both the Q and $\overline{Q}$ outputs of one-shots are used. This is easy with the 74123 and requires no more socket space per one-shot than the 555.

I also had some 8280 counters, which, unlike the more common 7490's, are loadable counters. An 8280 works fine in these circuits but, my counter had the alarming habit of resetting to 9999 instead of zero, and occasionally it would reset at a count of 5 instead of 10. I found that tying the LOAD ENABLE and all LOAD inputs to V_{cc} corrected the first problem, and a 0.01-μF ceramic disc at the socket from the clock input to ground cured the second problem.

When you design your control system, keep track of the unused gates on the IC's. If you need an inverter for example, you can use an unused NAND or NOR gate from another IC. Use ceramic discs across the supply lines at every other socket—*minimum*. Supply lines should be of heavy gauge wire. Use a surplus wire-wrap board or perforated board and wire-wrap sockets if at all possible, even if you have to invest in wire-wrap equipment. The tools are relatively low-cost, and it is easier to change your designs as you go along. If you must use a printed circuit board, breadboard everything first and construct the system one circuit at a time. It is helpful to use 0.01-μF discs on all the input signals coming into the logic board. These capacitors to either ground or to V_{cc} help shunt noise pulses.

Document **ALL** your work. Use color-coded wire if possible, and label wires if necessary. Identify every wire and component on paper. Years from now you may have to troubleshoot your system and writing everything down will save many hours of tracing circuits and wire runs.

A wall-mounted LED display isn't necessary, although it looks impressive. It *is* helpful to have LED's indicating the outputs during the troubleshooting setup for isolating problems. You can use switches to simulate the various inputs such as DUSK and DOOR OPEN in check-outs or troubleshooting. Switches wired across the logic relay contacts of RY1, as shown in Fig. 11, can be used to operate controlled devices manually if the boards are powered down for any reason.

Other circuit ideas

Here are some circuit ideas to add to your basic system:

1. Add a crystal-controlled clock and battery power to maintain timing in case of power failure.

2. Use door and window monitors to warn you as you leave the house that some entries are open or unlocked.

3. Add smoke and fire detectors; Use commercial detectors, but have each detector sound a unique series of pulses, and trigger the other alarms with the same sound so you know immediately where the fire is located.

You can also use the logic control to protect your house against burglary when it is unoccupied by turning the house lights on and off automatically. These circuits should be set to operating in somewhat random fashion over several days. They should start operating before dusk as during normal operation when the house is occupied. Regardless of what time the sequences start, they should end near the time your lights normally go out. And, as before, system should allow for human control. The possibilities for using logic circuits to control your house are limitless. What further uses can you think of? **R-E**