CIARCIA'S CIRCUIT CELLAR

BUILD THE POWER I/O SYSTEM

BY STEVE CIARCIA

Controlling the power to the real world with your computer

If you've been reading the Circuit Cellar for any period of time, you've probably noticed that I have a definite prejudice toward computer control. You'll never read about the merits of various DOS (disk operating system) utilities in a Circuit Cellar project, but it is conceivable that you'll come across a computerized, time-of-day-activated dog feeder.

Seriously, though, over the years I've presented a variety of sensors, monitors, and controllers that could turn your computer from a mild-mannered games machine into the HAL 9000 of the neighborhood. These capabilities, be they menacing or beneficial, are directly the result of making the computer aware of the real world.

I define "real world" as conditions that occur external to the computer. A 100-watt table lamp next to your computer is in the real world. The computer is unaware of the lamp's presence because the computer is not connected to the real world. Unless something happens within the address and memory space of the computer, however, it is unaware that anything else exists.

To remedy this condition of ignorance, we must construct an interface that allows the computer to recognize the occurrence of real-world activities and respond accordingly. This real-world interface is a translational device of sorts: the computer sees it as simply another addressable peripheral device, such as a cassette recorder or printer, yet the information communicated comes from the real world or is directed to the control of real-world events, such as turning on the lamp.

In the case of the lamp, the appropriate control element would be an electronic substitute for the mechanical switch to turn the light on and off. However, is the light really on? Of course we can see it, but, unless we provide additional sensing capability, the computer knows only that it has turned the light on, not that it *is* on.

Additional real-world conditions must be monitored to know this with certainty. One way is to physically read the 115 volts (V) AC applied to the bulb or monitor a thermostatic sensor attached to the bulb.

All this might sound absurd, but that is because we take for granted that it is easy to turn a light on and see that it is in fact on. In critical control situations, more than simple on/off activation is required. Fre-

(continued)

Steve Ciarcia (pronounced "see-ARE-see-ah") is an electronics engineer and computer consultant with experience in process control, digital design, nuclear instrumentation, and product development. He is the author of several books about electronics. You can write to him at POB 582, Glastonbury, CT 06033.

quently, as in industrial applications, both the actions and the results must be monitored to produce reliable control conditions. (Open-loop control systems such as the BSR X-10 are generally unsuitable for industrial applications.)

One of the prime components of any real-world interface is discrete-bit AC/DC power input and output, which is on/off control and monitoring of 115-V AC or 5- to 48-V DC devices. With an AC/DC power I/O (input/output) interface, we can control and monitor motors, lights, high-voltage AC systems, and process-control and monitoring devices.

This month's project is a discussion of the design and construction of an AC/DC power I/O (power I/O hereafter) interface with particular emphasis on the internal configuration of the solid-state relays (SSRs) and receivers. The emphasis is on an industrial quality-control interface that meets the needs of both the experimenter and the industrialist. Beyond the homebrew relays and receivers that you can build, I'll describe a true closed-loop power I/O control system using the Circuit Cellar Z8-based computer system (which I've described and used numerous times) and commercially available components.

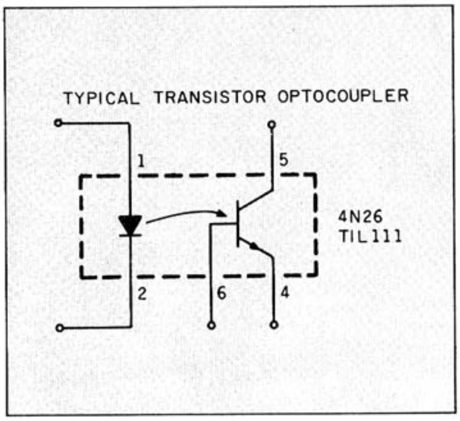

Figure 1: A *typical transistor optocoupler.*

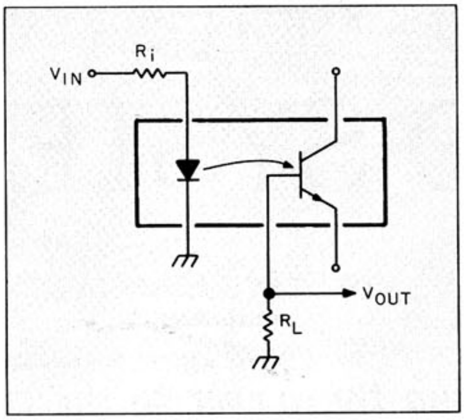

Figure 2: *Figure 1 used as a high-speed diode-diode optocoupler.*

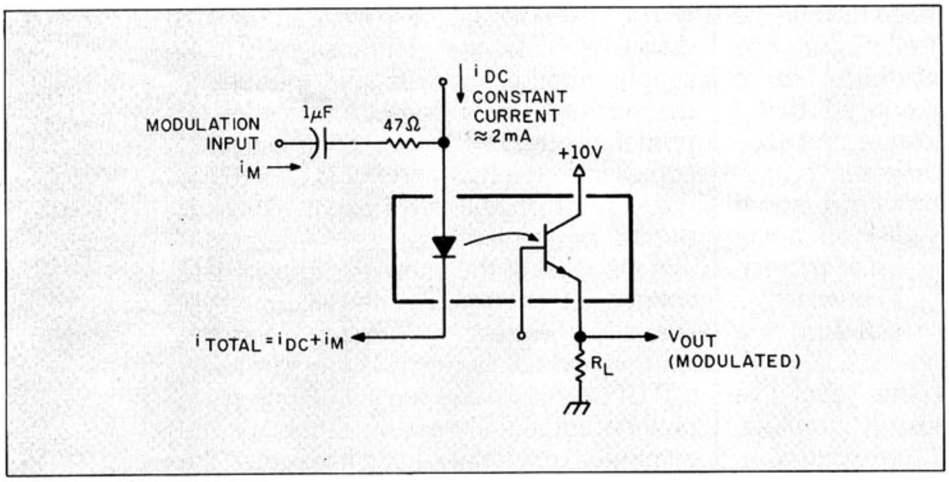

Figure 3: *Figure 1 used as an analog-signal optocoupler.*

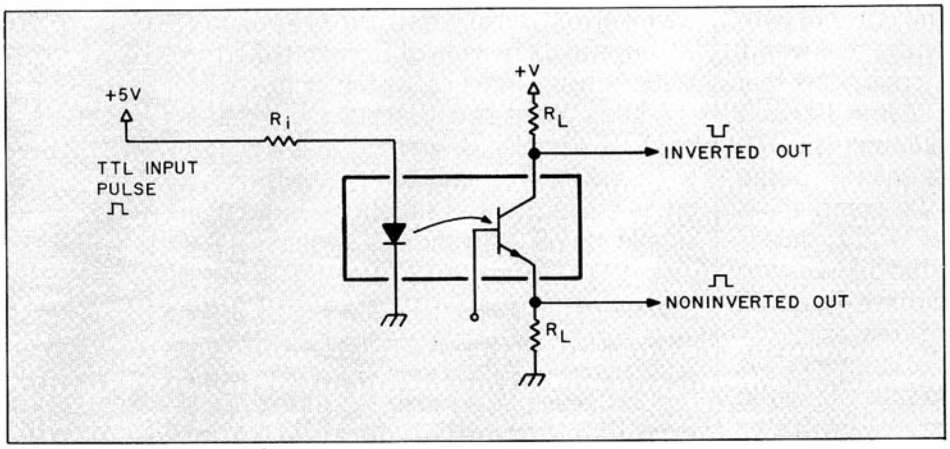

Figure 4: *Figure 1 used as a digital-signal optocoupler.*

A DISCRETE-BIT INTERFACE

Generally speaking, most computers are parallel in function. If you are using an Apple II or IBM PC, communication between the processor and its peripheral devices is handled 8 bits at a time presented in parallel. A parallel printer, for example, receives its character data as a 7- or 8-bit parallel word and sends its status and operating conditions back in a similar manner.

Most peripheral devices use all 8 bits at a time because they are most often communicating 8-bit data or ASCII (American Standard Code for Information Interchange) characters. Externally connected devices such as a light and a thermostatically controlled switch are single-bit devices. However, since the internal function of the computer is word rather than bit wide, each bit of the 8-bit word is used to separately receive or control an external event. If it were a 16-bit computer such as the 68000, each word would have 16 discrete bits of I/O.

Within the computer, one or more memory locations (or I/O port locations if it's a Z80 or 8080) are set aside and the addresses decoded as parallel I/O ports. If configured for output, each bit in a port is then connected to a discrete module that converts the TTL (transistor-transistor logic) level presented to it to a high/low, on/off voltage-level output. If the module is for AC control, it will convert a TTL high to 115 V AC and a TTL low to 0 V AC. DC output modules function as simple contact closures with the voltages dependent upon your proposed application.

When the addressed location is an input port, each bit is attached to an input receiver that converts a high-voltage input level to a TTL logic 1 and a low-voltage input level to a logic 0. The exact range and switch point of the module have to be selected for the application, and there are differences depending on whether the applied voltage is AC or DC.

With a single parallel I/O location (port), eight separate devices can be controlled and eight discrete events monitored. To properly coordinate the activity, bit rather than word manipulation becomes essential.

The I/O modules provide level conversion and isolation between the computer and the external device. Depending upon the components employed, I/O interfacing need not be a prodigious task.

ISOLATION IS THE KEY

The most important factor in I/O interfacing, especially with AC line voltages, is isolation. The computer you are using most likely operates on 5 V. If 115 V AC is applied to an unisolated input port, you are definitely going to produce smoke! High-voltage inputs must be safely converted to 5 V, and output devices must have no way to inadvertently feed 115 V AC back into the computer.

The simplest isolation device is the electromechanical relay. You can easily attach a reed relay to each output bit and the isolated contacts used to switch the AC line. Similarly, you can connect the external voltage to a relay whose contacts are attached to an input bit. When the input level is high enough (determined by series resistors), the contacts close and the computer senses the condition.

There is nothing wrong with using relays. For many years, this was the only method available, and it still works, to a point. However, relays are large, expensive, slow, electrically noisy, and subject to wear. They have been replaced with solid-state optoelectronic components that are small,

(continued)

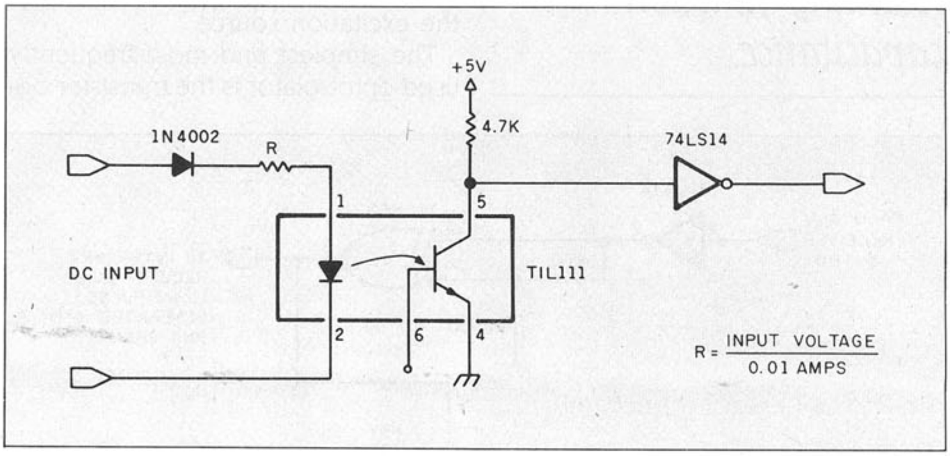

Figure 5: A DC *input receiver.*

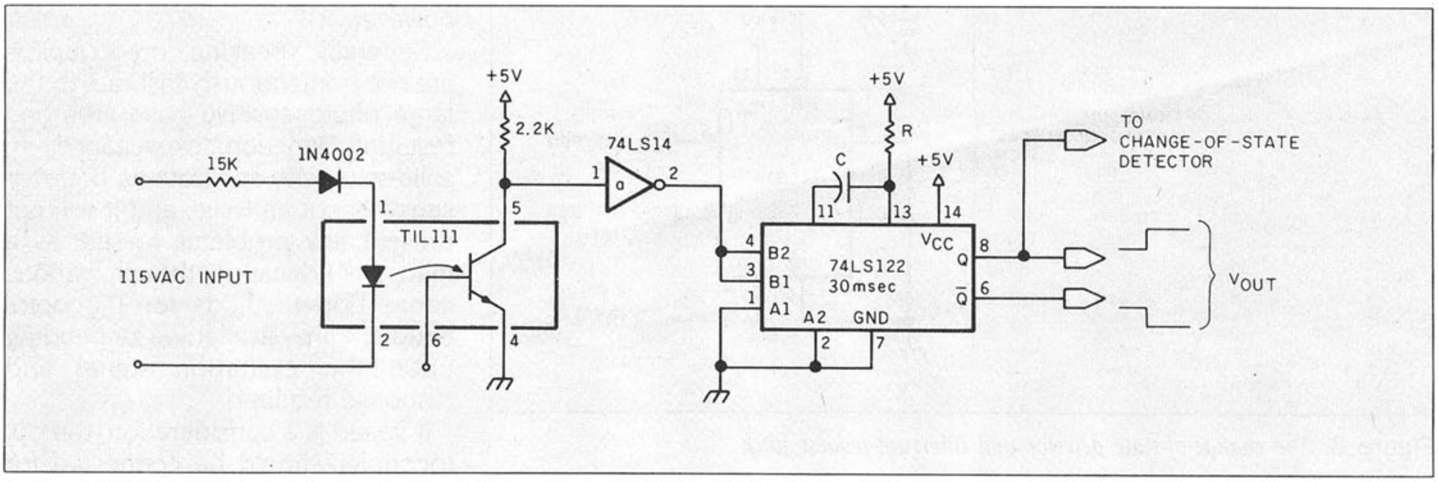

Figure 6: A *half-wave* AC *input receiver with constant-level output.*

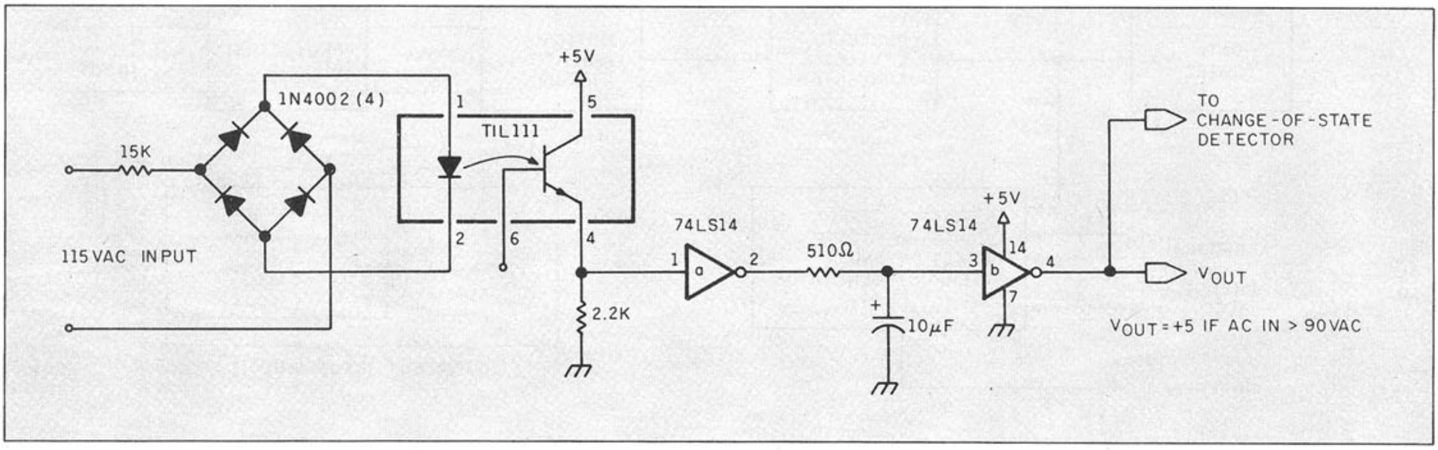

Figure 7: A *full-wave* AC *input receiver with constant-level output.*

Generally speaking, optocouplers are not tremendously fast due to the large photosensitive base area and the resulting junction capacitance.

inexpensive, fast, optionally noiseless, and have no wear in proper use.

INSIDE AN OPTOCOUPLER

The essential ingredient in this project is the optoisolator and its use as an I/O control device for AC and DC voltages. With it, we can configure high-performance solid-state relays and solid-state input receivers. The exact configuration, as I will explain, depends upon the application and the excitation source.

The simplest and most frequently used optoisolator is the transistor optocoupler. It consists of a GaAs (gallium arsenide) infrared LED (light-emitting diode) and a silicon NPN phototransistor separated by a glass partition. The thickness of the glass determines the isolation level of the component. A typical isolation value is 1500 V. This means that the potential difference between the input and output sides of the optocoupler must be less than 1500 V, or it will break down and expose the computer to hazardous voltages. (While this seems unusually high, remember that these relays often switch inductive loads that produce high-voltage transients. Proper "snubbing" and transient suppression must be employed or these limits can be exceeded.)

Figure 1 is the typical transistor optocoupler. A current is applied to the LED, which induces a base current in the phototransistor proportional to the light radiated by the LED. This in turn allows current to flow between the collector and emitter of the transistor. A typical LED current is 10 to 50 milliamperes (mA). A 10-mA current greatly extends component life, however.

Generally speaking, optocouplers are not tremendously fast due to the large photosensitive base area and resulting junction capacitance. In solid-state-relay applications, however, speed is not an issue, and it will not present any problems for us. As a matter of education, though, various connection methods for the optocoupler are available, depending upon the excitation signal and response required.

If speed is a consideration, the optocoupler should be connected for

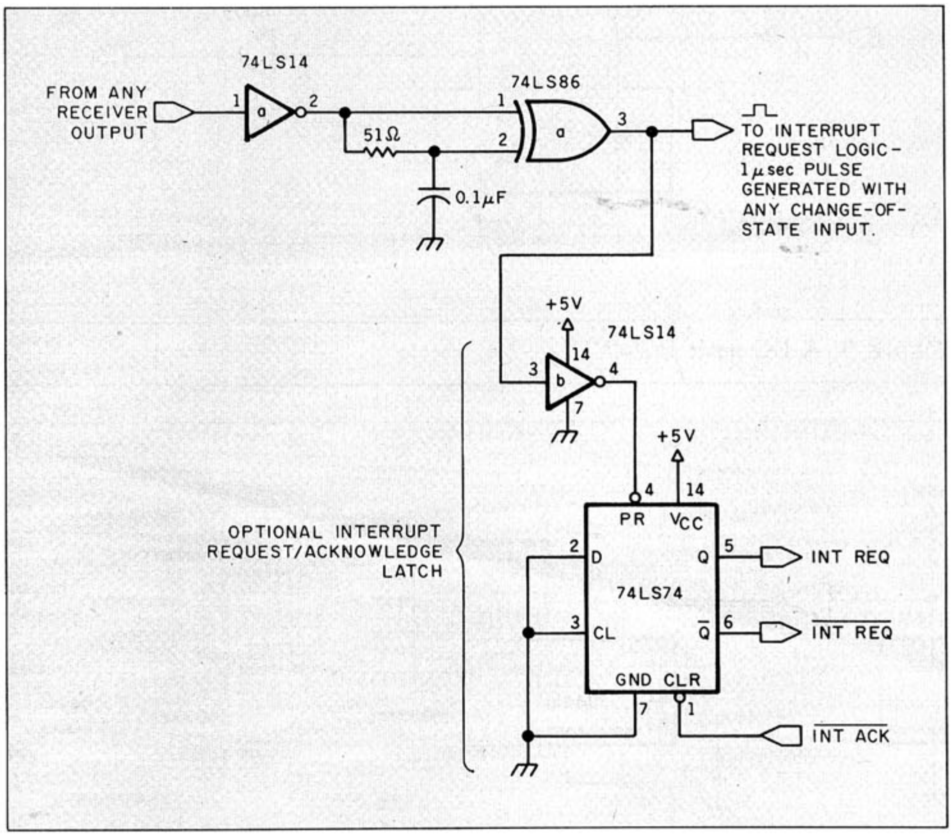

Figure 8: *The change-of-state detector and interrupt-request latch.*

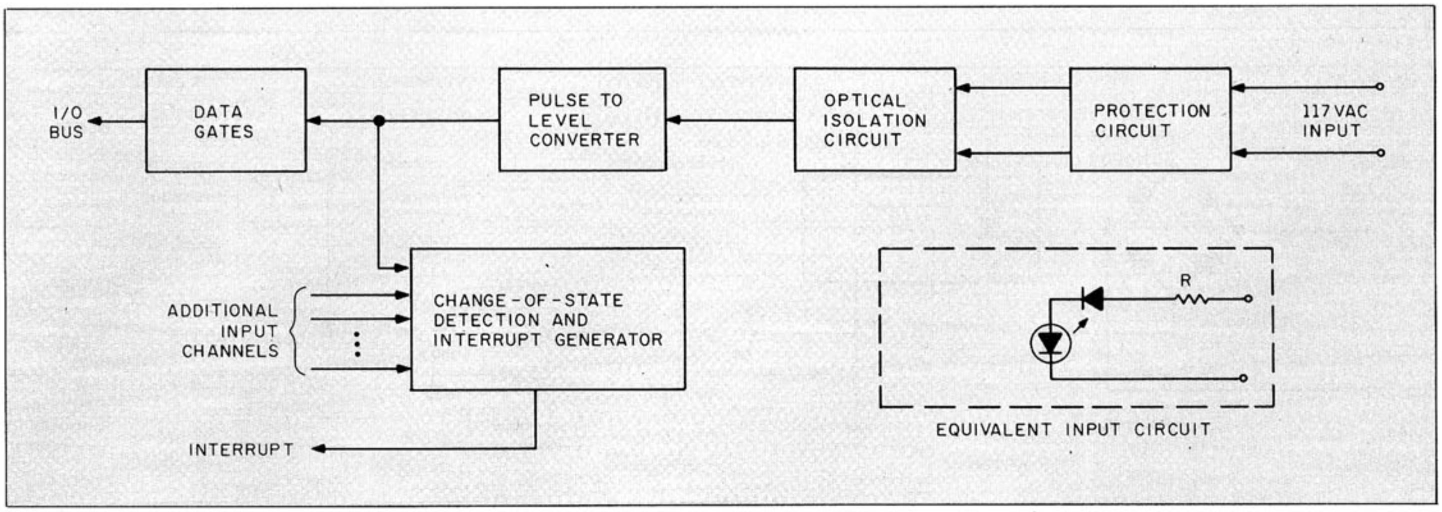

Figure 9: *The block diagram of an optoisolated input circuit.*

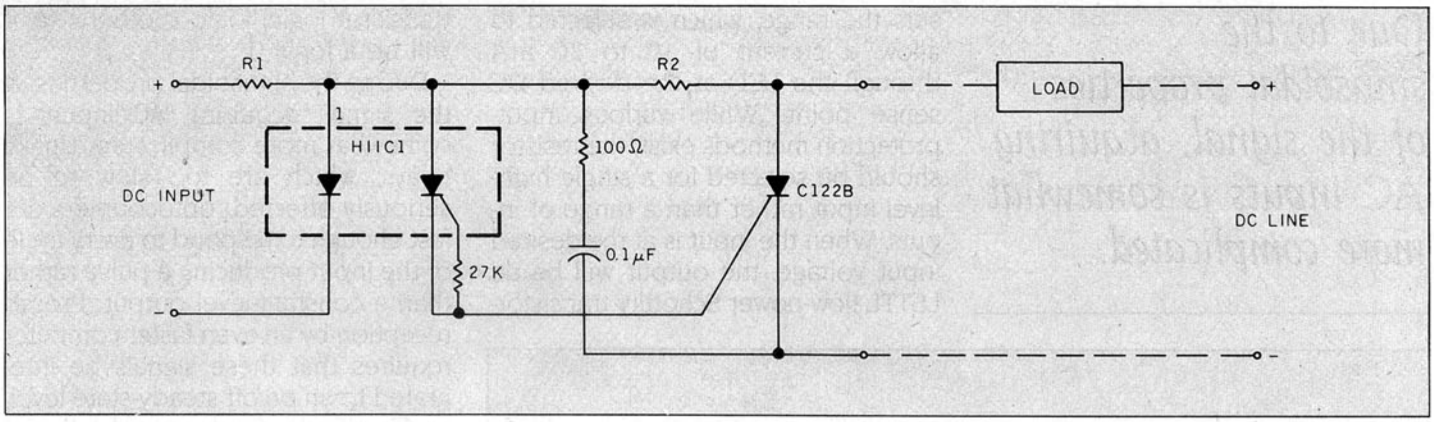

Figure 10: A *simple* DC *output module.*

Table 1: *Resistor values for figure* 10.

DC INPUT VOLTAGE	R1	R2
5	390	200
12	1.1k	200
24	2.4k	470
48	4.7k	1k
120	12k	2.2k

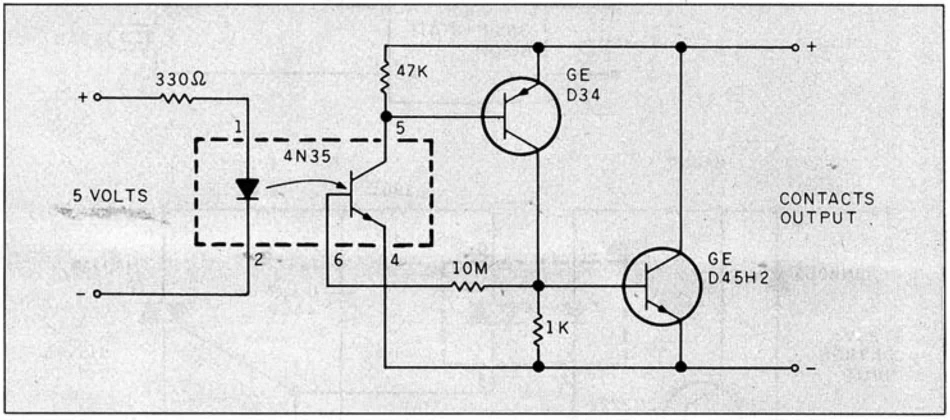

Figure 11: A *normally open* DC *output module.*

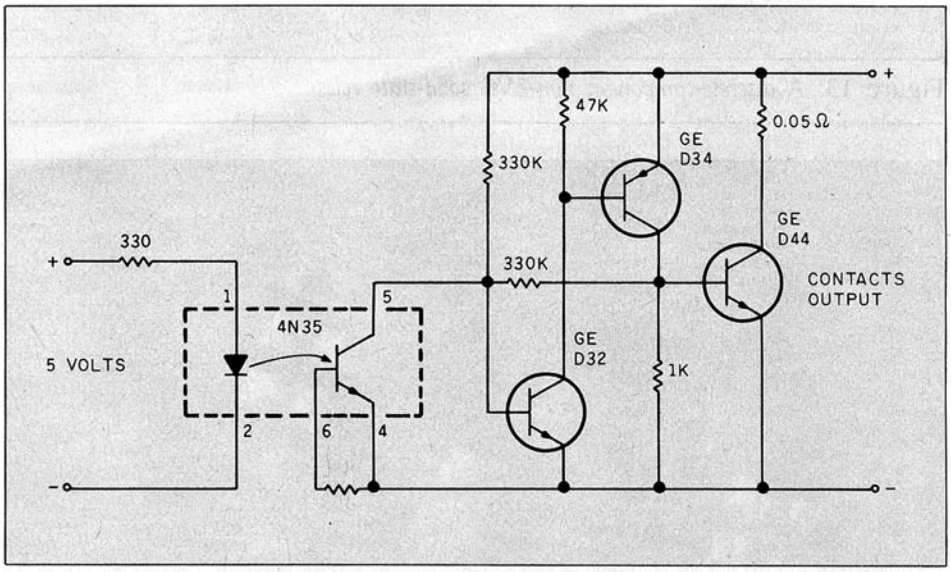

Figure 12: A *normally closed* DC *output module.*

diode-diode operation, as in figure 2. The output signal is directly received at the base connection. Typical response time is 2 to 5 microseconds (μs) as a diode-transistor coupler but only 50 to 100 nanoseconds as a diode-diode coupler. The one disadvantage is the much lower output current, which must be amplified.

While most experimenters think of optocouplers as digital devices, a transistor optoisolator can also be used with analog signals, as shown in figure 3. A constant bias current is applied to the LED to turn the transistor on enough to be within its linear range. Next, an analog input signal (modulation voltage) is also applied to the LED, which varies its light output proportionally to the modulated input. The emitter current in the phototransistor similarly follows this variation.

Most optocouplers are used for digital isolation, as shown in figure 4. A 10-mA LED current simply turns the transistor on or off with the inverted and noninverted responses available at the collector and emitter, respectively.

While the transistor optocoupler is the one I've chosen to describe, optocouplers are available with transistors, silicon-controlled rectifiers (SCRs), and Triacs as output devices. In some applications, the latter devices are more appropriate.

DISCRETE-INPUT AC/DC RECEIVERS

The discrete-input DC receiver is by far the easiest module to construct. As demonstrated in figure 5, it is nothing more than an LED and a current-limiting resistor. I have added a series-blocking diode to protect the optocoupler from reverse connection and a 74LS14 Schmitt trigger to provide cleanly switched levels to the computer.

DC input receivers are generally preset as 5-, 12-, 24-, 36-, or 48-V detectors. The series-input resistor

Due to the sinusoidal properties of the signal, acquiring AC inputs is somewhat more complicated.

sets the range, which is selected to allow a current of 10 to 20 mA through the LED at the desired DC sense point. While various input-protection methods exist, the resistor should be selected for a single high-level input rather than a range of inputs. When the input is at the desired input voltage, the output will be an LSTTL (low-power Schottky transistor-transistor logic) logic 1; otherwise, it will be a logic 0.

Due to the sinusoidal properties of the signal, acquiring AC inputs is somewhat more complicated. Unlike relays, which are too slow to be seriously affected, optocouplers are fast enough to respond to every cycle of the input producing a pulse rather than a constant-level output. Proper reception by an even faster computer requires that these signals be integrated to an on/off steady-state level, as described in figures 6 and 7. Figure 6 is a half-wave detector that uses a 30-millisecond (ms) retriggerable one-shot; figure 7 is a full-wave detector that employs a simple RC (resistor-capacitor) circuit to integrate the pulse output. In either case, a logic 1 output signifies that a 115-V AC input is present.

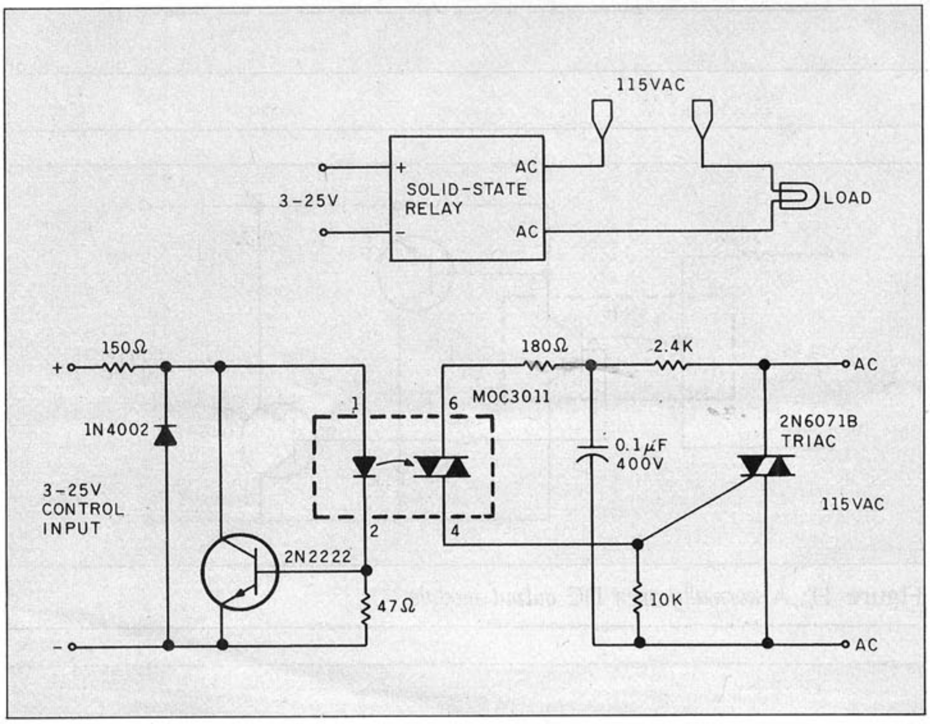

Figure 13: *A discrete-component non-ZVS solid-state relay.*

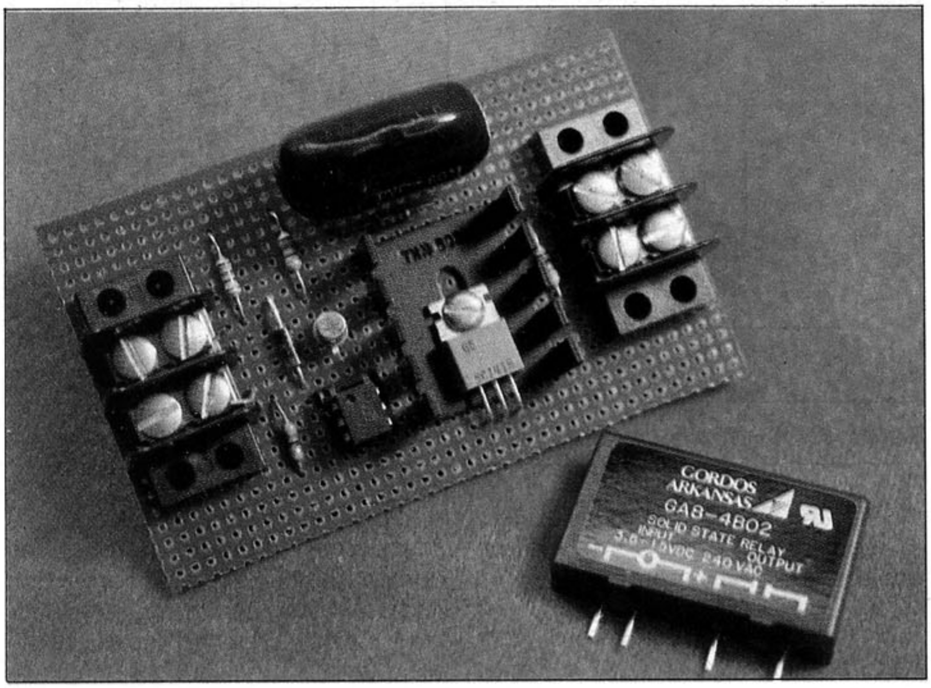

Photo 1: *The circuit board is a prototype of a typical non-ZVS solid-state switch like the one described in figure 13. A simpler alternative is to use the solid-state relay shown to the right of the board.*

CHANGE-OF-STATE DETECTION

One infrequently mentioned but important issue regarding discrete-input receivers is change-of-state detection. "Change of state" means simply that a receiver has changed its input level since the last time you looked at it. This might seem trivial if you have only one input module but is quite essential if you are monitoring 64 devices.

The change-of-state condition can be determined and indicated either through hardware logic or software programming. Figure 8 is a hardware change-of-state detector. Whenever the input module's level goes from 0 to 1, or 1 to 0, a 1-μs pulse is generated at the output of the 74LS86 exclusive OR gate. This signal can be used to directly interrupt the processor or set a change-of-state flip-flop, as shown. The flip-flop retains its set condition until reset by the interrupt program. If eight input modules are used, there would be eight sets of this hardware with the outputs combined to generate a single "someone changed" interrupt. Figure 9 diagrams this approach. The advantages of the hardware change-of-state detection are that it is transparent to the user and requires little processor overhead.

An alternative approach is to scan the inputs in software periodically and compare the old and new readings to find changed states. In sophisticated control systems, a background interrupt routine periodically scans the input channels. Any changed states are represented as a byte in a table available to the application program. More on this technique later in our real application.

DC Power Output Control Devices

As previously mentioned, mechanical relays have been and can still be used in power-control applications. In new designs, however, the cost-effective approach is to use SSRs.

Solid-state relays come in a variety of flavors, depending upon the application. Unlike mechanical relays, which are nonpolarized, SSRs can be either polarized or nonpolarized. DC SSRs are normally polarized; AC SSRs are not.

Figures 10, 11, and 12 illustrate three kinds of DC output control modules. While they technically are SSRs, polarized switches such as DC output control modules are quite different in component configuration and are generally referred to as DC output units rather than SSRs. Figure 10 is a very simple DC module using an H11C1 photo SCR, which in turn triggers a higher-current SCR. Because it uses an SCR, this type of circuit simulates a latched-output relay. A voltage level applied to the LED input turns on the SCR and allows current to flow through it. The amount of voltage that will turn on the SCR is determined by R1 and R2, as shown in table 1. Both the LED input signal and the external load current must be removed to turn off the SCR.

This seems to be an absurd situation if the purpose of the DC output module is in fact to control the external current. Because of this, SCR relay devices are not normally used for DC resistive loads and are reserved in-

(continued)

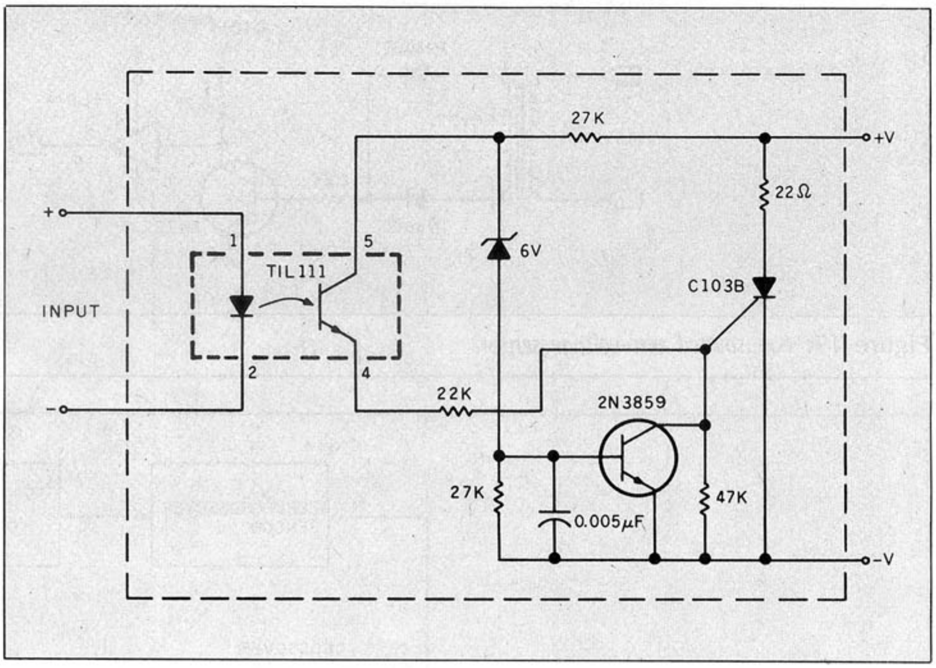

Figure 14: A *normally open* ZVS *circuit.*

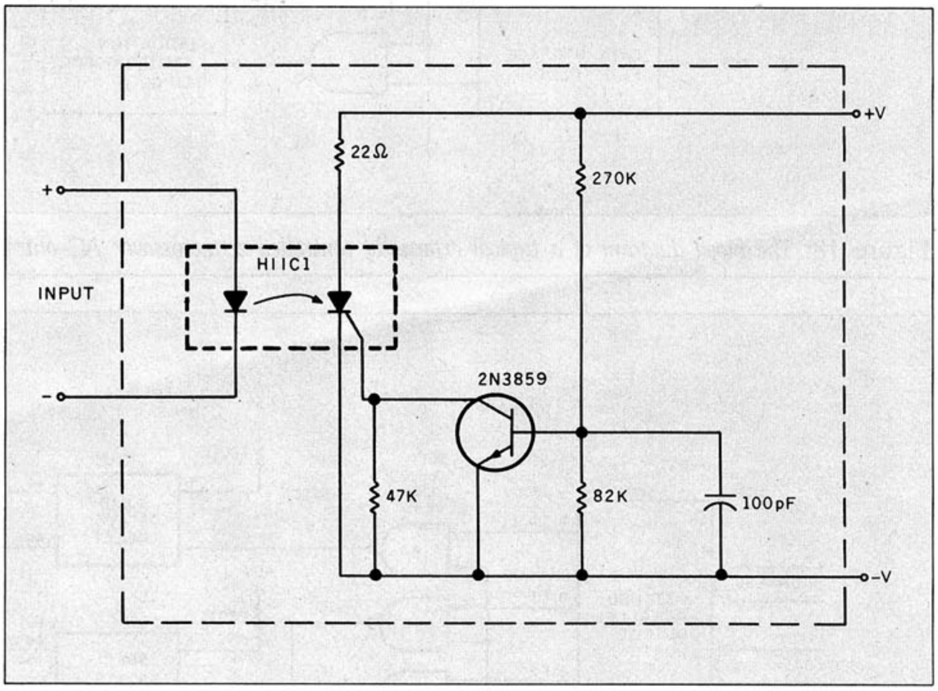

Figure 15: A*nother normally open* ZVS *circuit.*

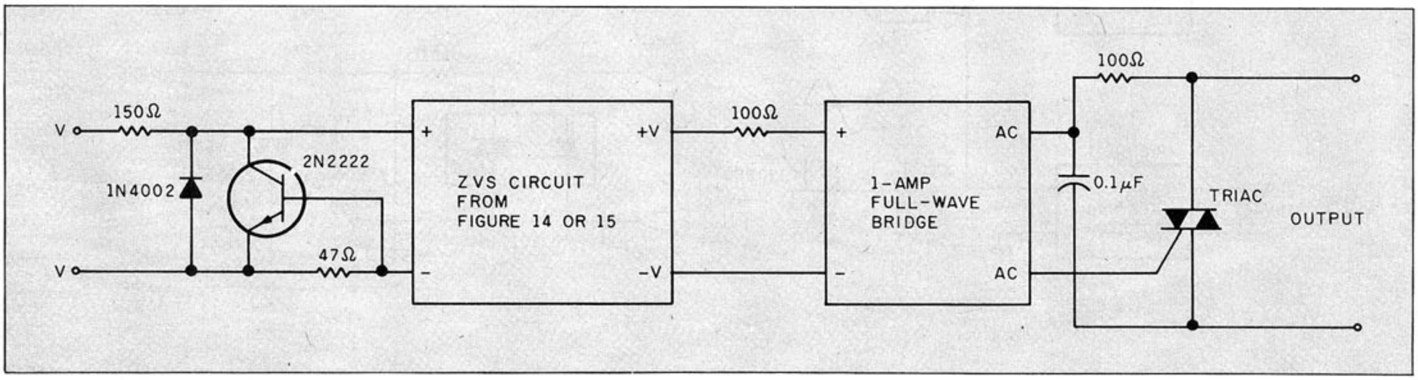

Figure 16: A ZVS *stand-alone* SSR.

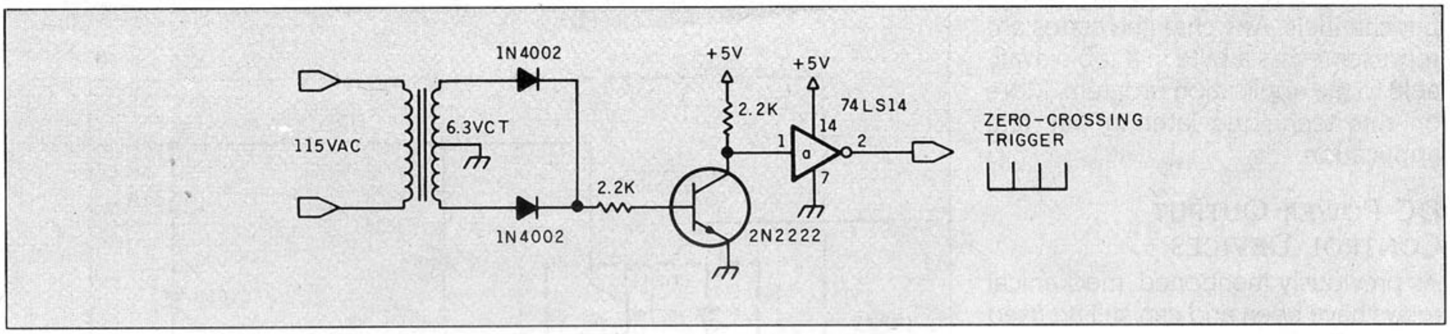

Figure 17: *An isolated zero-voltage sensor.*

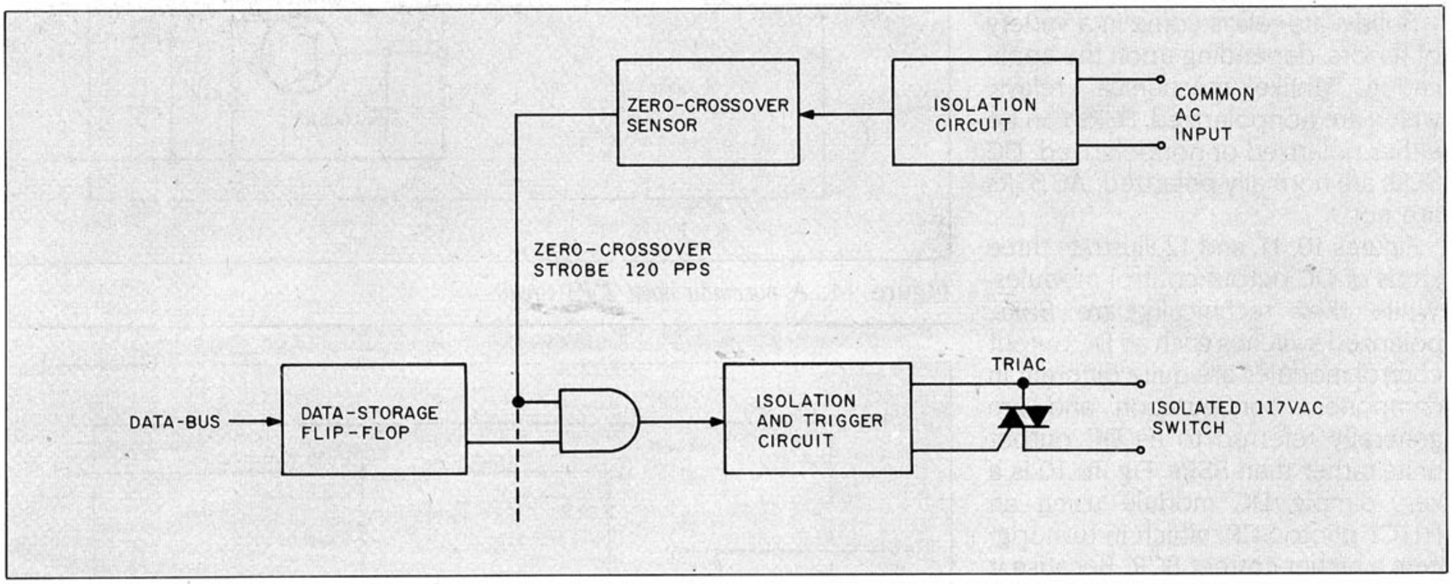

Figure 18: *The block diagram of a typical externally controlled zero-crossover* AC *output circuit.*

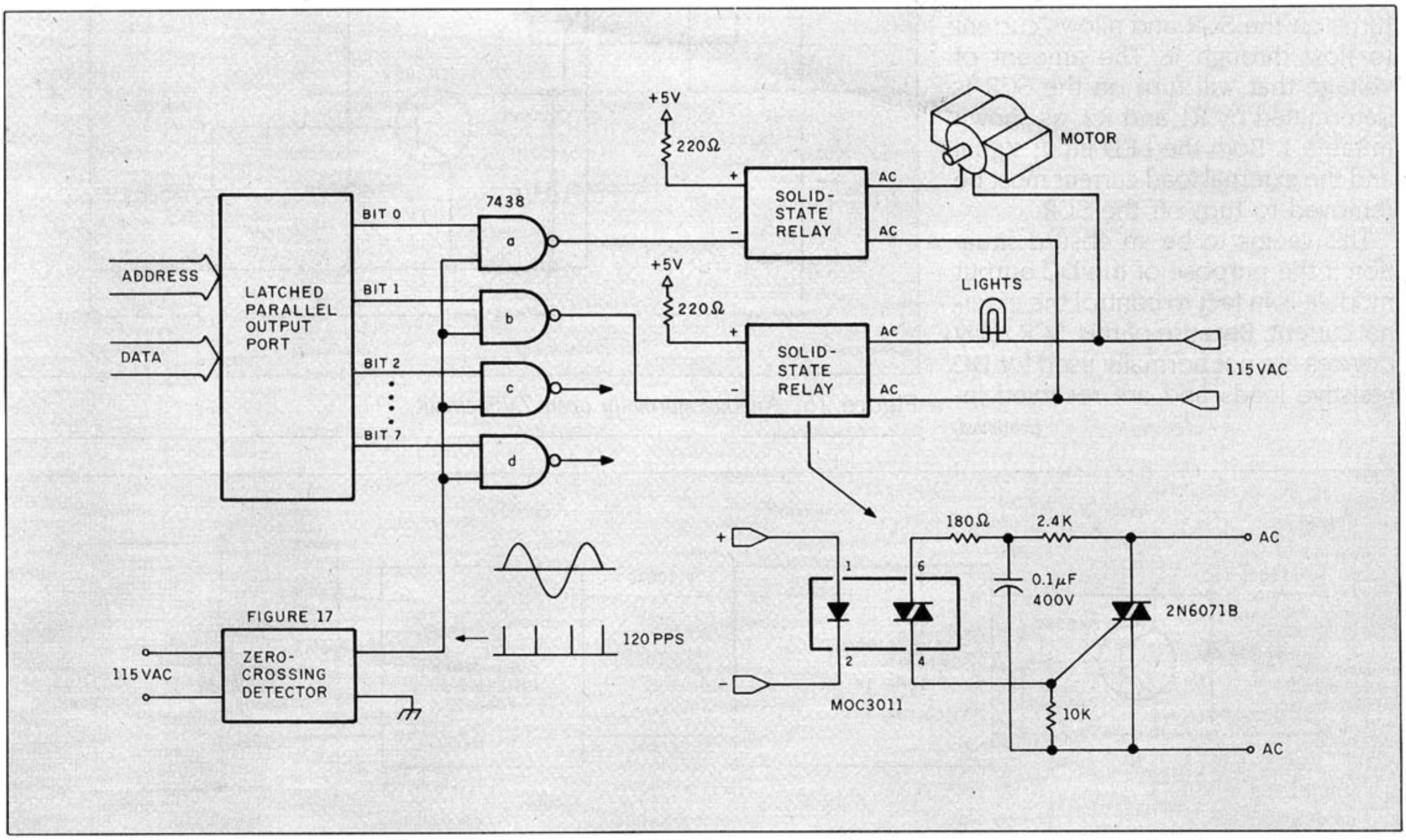

Figure 19: *The typical computer-controlled* SSR *output with* ZVS.

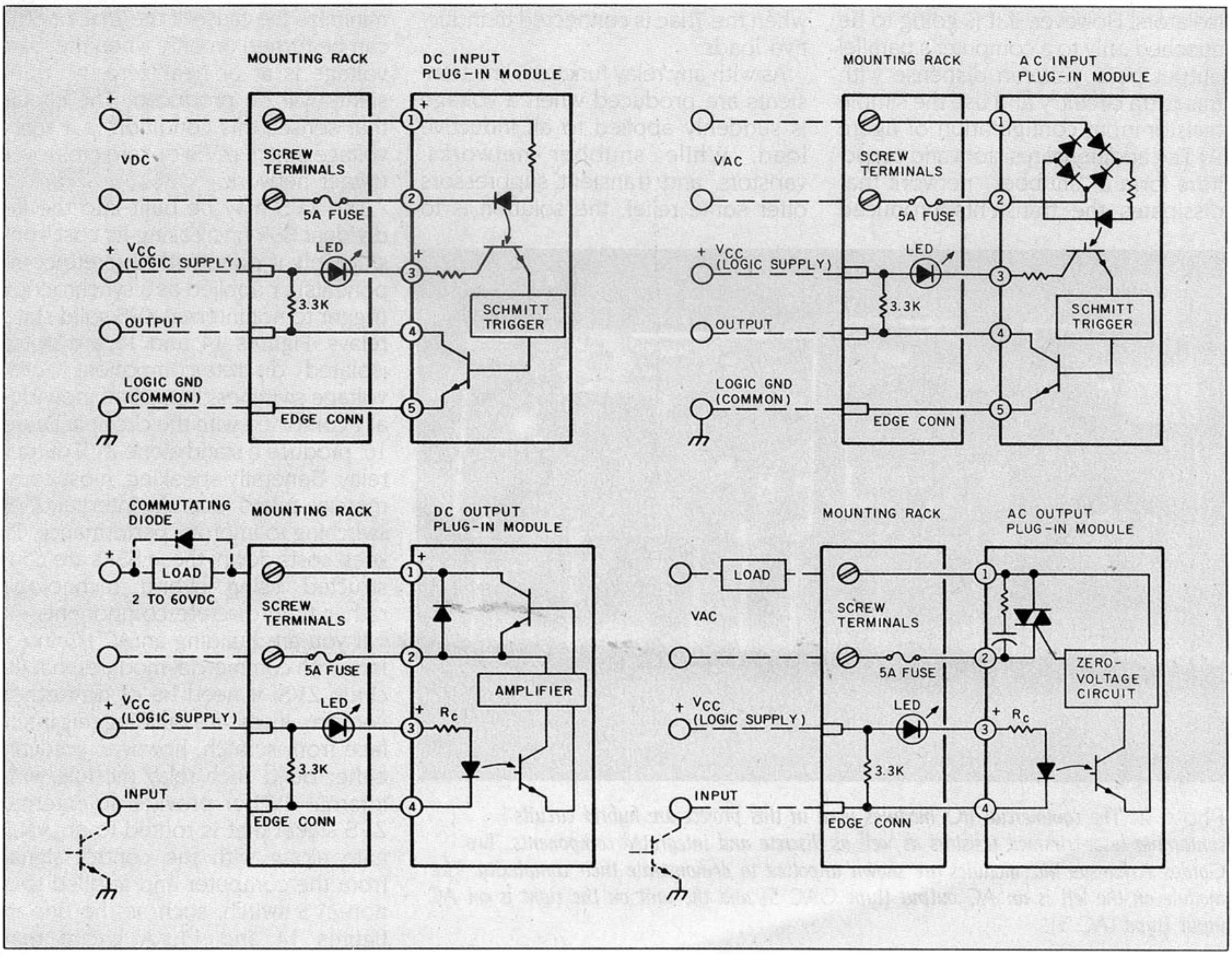

Figure 20: *Mounting-connection diagrams of four Gordos Arkansas power I/O interface modules.*

stead for commutating loads such as motors. With a DC commutating motor, the output current is interrupted many times a second as the motor shaft turns, allowing the SCR to turn off when the LED is extinguished.

All other DC control applications rely on transistor control elements, which exhibit fewer peculiarities but involve more components. Figures 11 and 12 demonstrate two typical 25-V DC output-module designs. Figure 11 is configured to have a normally open output; figure 12 has a normally closed output. These units are nonlatching and can be turned on or off in direct response to the logic levels from a parallel output port.

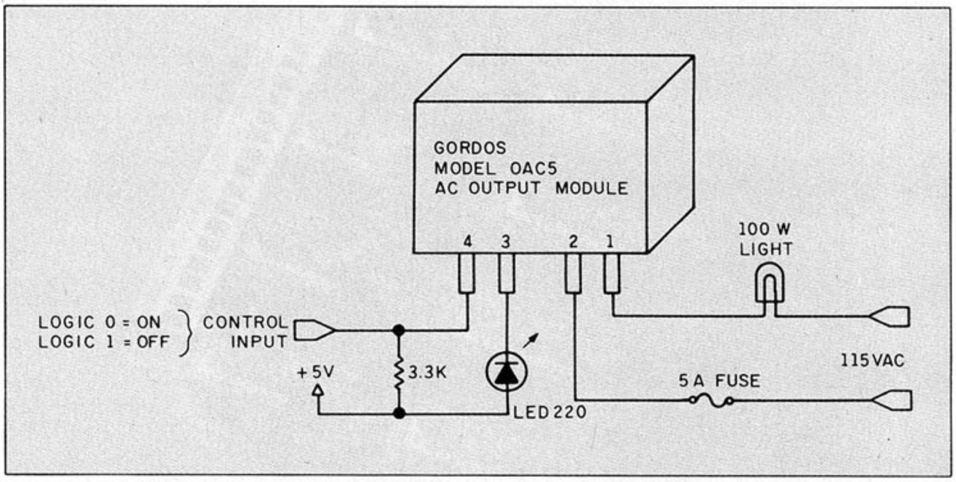

Figure 21: *The connections for a Gordos AC output module.*

AC POWER OUTPUT CONTROL DEVICES

When we use the term "solid-state relay," we are generally talking about AC power output devices. These SSRs are nonpolarized and intended for use only with AC loads.

Figure 13 and photo 1 show the circuit of a general-purpose off-the-shelf component-configured SSR module suitable for control of lights and light-load appliances. The circuit employs an MOC3011 photoisolated Triac that in turn controls a power-output Triac. Input-protection circuitry has been added to the LED side of the module so that it can be used within a 3- to 25-V input range. (This input circuit can be added to any of the optoisolators. However, if it is going to be attached only to a computer's parallel output port, you can dispense with this extra circuitry and use the simple resistor-input configuration of figure 9.) The additional resistors and capacitors form a "snubber" network that dissipates the transients produced when the Triac is connected to inductive loads.

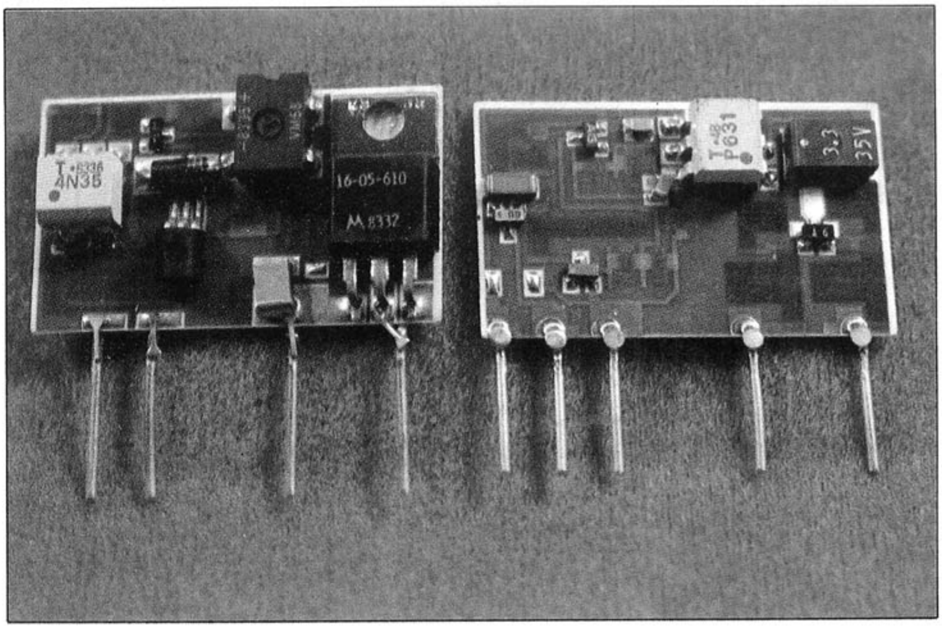

Photo 2: *The commercial I/O modules used in this project are hybrid circuits containing laser-trimmed resistors as well as discrete and integrated components. Two Gordos Arkansas Inc. modules are shown unpotted to demonstrate their complexity. The module on the left is an AC output (type OAC 5) and the unit on the right is an AC input (type IAC 5).*

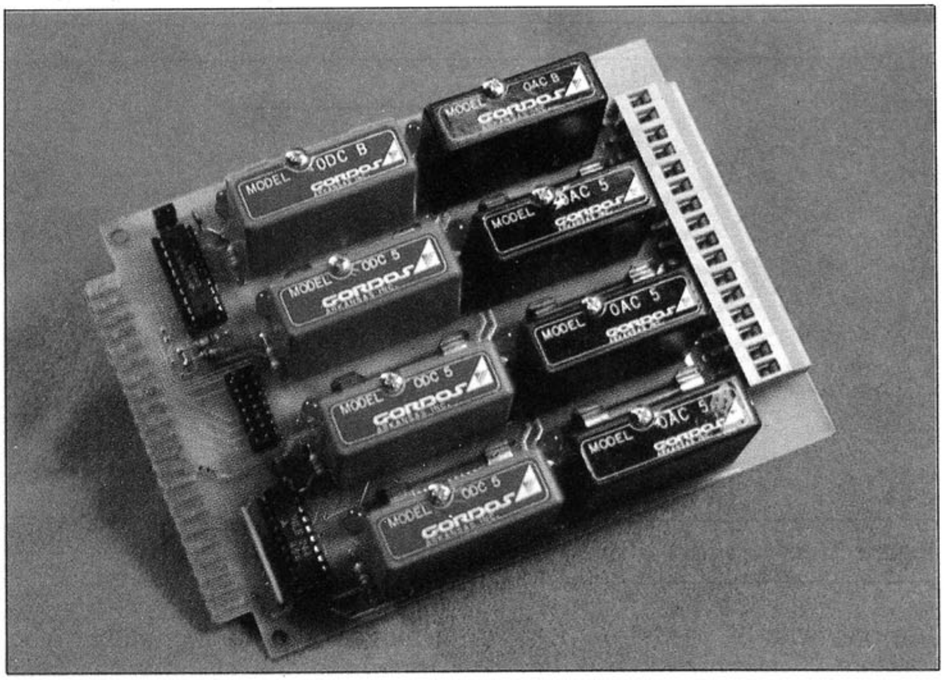

Photo 3: *Eight of these I/O modules can be mounted together on a single board. Individual boards can be separately configured as AC-DC output or AC-DC input. The Circuit Cellar Z8 Power I/O prototype board in the photo has four AC outputs (black modules) and four DC outputs (red modules).*

As with any relay function, line transients are produced when a voltage is suddenly applied to an inductive load. While snubber networks, varistors, and transient suppressors offer some relief, the solution is to minimize the cause. If the Triac or SCR can be turned on only when the load voltage is at or near zero, no transients will be produced. The circuit that senses this condition is a zero-voltage switch (ZVS) or zero-crossover trigger network.

The ZVS may be built into the individual SSR (increasing its cost considerably if done with discrete components) or applied as a synchronous trigger to noninternal ZVS solid-state relays. Figures 14 and 15 are opto-isolated discrete-component zero-voltage switches that, when individually combined with the circuit in figure 16, produce a stand-alone ZVS output relay. Generally speaking, most commercial relays contain internal ZVS switching to improve performance. To keep costs down, these SSRs are constructed using hybrid technology rather than discrete components.

If you are building an AC I/O interface with commercial modules that include ZVS, it need be of no further concern. If you are building this interface from scratch, however, you can either build each relay module with internal ZVS or provide an external ZVS signal that is routed to an AND gate along with the control signal from the computer and applied to a non-ZVS switch, such as the one in figures 14 and 15. A circuit that detects zero crossing is demonstrated in figure 17, and a block diagram of this synchronous switching concept is presented in figure 18.

Figure 19 is the circuit for a computer-controlled AC output interface using the devices I've described thus far. To turn on the individual output channels, you merely set a logic 1 output at that bit position. This is most easily accomplished with an OUT command in BASIC. Once the bit is set, the SSR will turn on at the next zero crossing of the AC line.

Not Through Yet

Ordinarily, the project would end here. I've demonstrated how to build the I/O modules. A two-line BASIC program with INP and OUT instructions is all that it takes to control them. Unfortunately, knowing how to build a solid-state relay is different from assembling a practical control system.

Rather than leaving these practical matters as an exercise for you, I'd like to describe the 64-channel power I/O system I ultimately configured. The description, though somewhat complex in detail, is intended to provide a basic understanding of the system software necessary to implement a reliable high-performance industrial-grade closed-loop control system. Utilizing the basic concept but substituting commercially available power I/O modules and a dedicated computer, a rather sophisticated programmable power I/O control system can be configured.

The commercial modules I chose are made by Gordos Arkansas Inc. Shown in photo 2, these potted modules are designed using thick-film hybrid technology for high-density packaging. Figure 20 is a diagram of the contents and connections of four typical Gordos power I/O interface modules. Figure 21 demonstrates the physical connections for an AC output module.

The computer I chose is the Z8 system/controller, which I've used in many Circuit Cellar projects. Based on a project presented in July and August 1981, the Z8 system/controller is a 4- by 4½-inch single-board computer with on-board tiny BASIC or FORTH, 6K bytes of RAM (random-access read/write memory) or EPROM (erasable programmable read-only memory), two parallel ports, and one serial port. To the Z8 computer, I've added the Micromint BCC33 memory and parallel I/O expansion board, which adds 8K bytes of memory, three parallel ports, and a cassette-storage interface, to interface to the power I/O modules.

(If you have either built the original Z8 computer/controller and wish to update it to the system/controller configuration or would like to build the memory and I/O expansion board for your existing system, just send me a preaddressed 9- by 12-inch envelope with $2.30 postage [overseas, just send name and address and $4 in international mail coupons] and I will send you the schematics and manuals for the two boards.)

The three parallel ports on the expansion board are configured as an I/O bus with input, output, and control capability. The power I/O modules are separated by function (input or output) and arranged eight modules per I/O card. Both AC and DC modules may be on one card, but only if they are all the same function. Up to 16 boards (64 input and 64 output modules, addressed as input boards 0 through 7 and output boards 0 through 7) can be accommodated with a single BCC33 expansion board. (Eight expansion boards can be put in the system if you are trying to control a small city.)

The computer communicates with the I/O cards through the expansion-board parallel ports. Port A functions as an 8-bit input bus, port B as an 8-bit output bus, and port C as the control lines for the individual I/O cards. Each power I/O card has a set of eight two-position jumpers, a 74LS374 output latch, and a 74LS244 input buffer (see photo 3 and the schematic in figure 22). Photo 4 shows some of the cards mounted in a card cage.

A single jumper selects board address and function. The eight output lines of port C are attached to the center position of the eight jumpers (boards 0 through 7). Only one of these lines is active low at a time; all others are at logic 1. The line that is low enables the power I/O card jumpered to it. Within that enabled card, a jumper installed to the center and left side (O) will enable the LS373. If installed between the center and the right side (I), it selects the LS244. A second jumper Tri-states the LS373 when the board is configured for input. If the jumper were in the #31 position, this would be addressed by bit 3 on port C, and it would be an input-only card.

> *The process of interfacing with the power I/O cards is relatively simple and can be accomplished directly in BASIC if speed is not a critical factor.*

Figure 23 shows a detailed block diagram of the power I/O system. It can be all AC input, DC input, AC output, DC output, or a mixture (in groups of eight similar functions). When you want to set the eight output modules on board 2, you merely set the bit pattern on port B and then strobe bit 1 on port C (board 2 enable line) to latch that data into the LS373. Conversely, to read the eight input channels on board 3, you would set bit 3 of port C low, read and store the data input to port A, and set bit 3 of port C high again.

The process of interfacing with the power I/O cards is relatively simple and can be accomplished directly in

(continued)

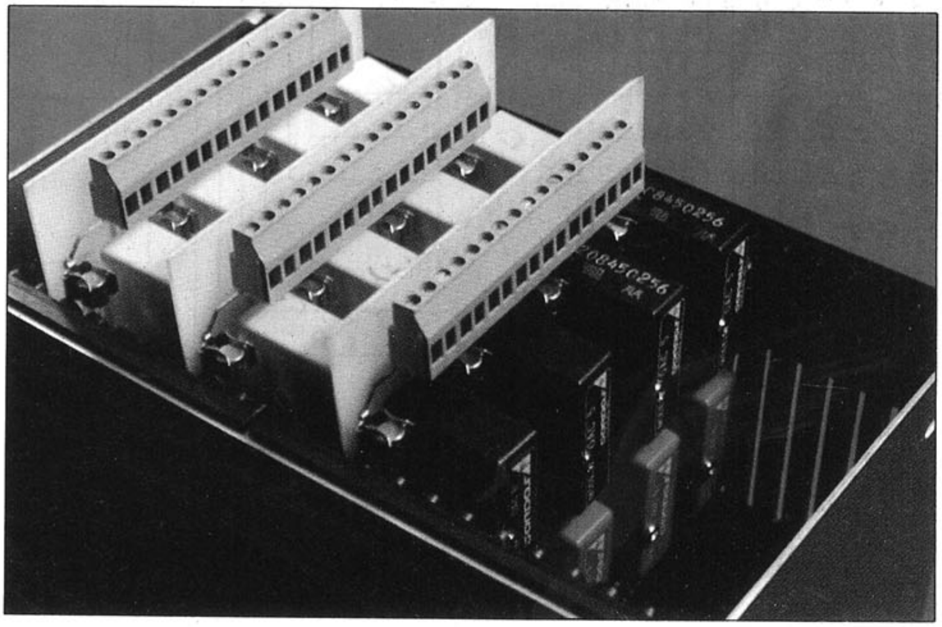

Photo 4: *Up to 16 boards (8 input and 8 output) can be supported from each I/O expansion board in a Z8 system. Up to 4 expansion boards can be mounted in a card cage. The photo shows 3 expansion boards installed. The green connector protruding out of the cage is for external wiring connections.*

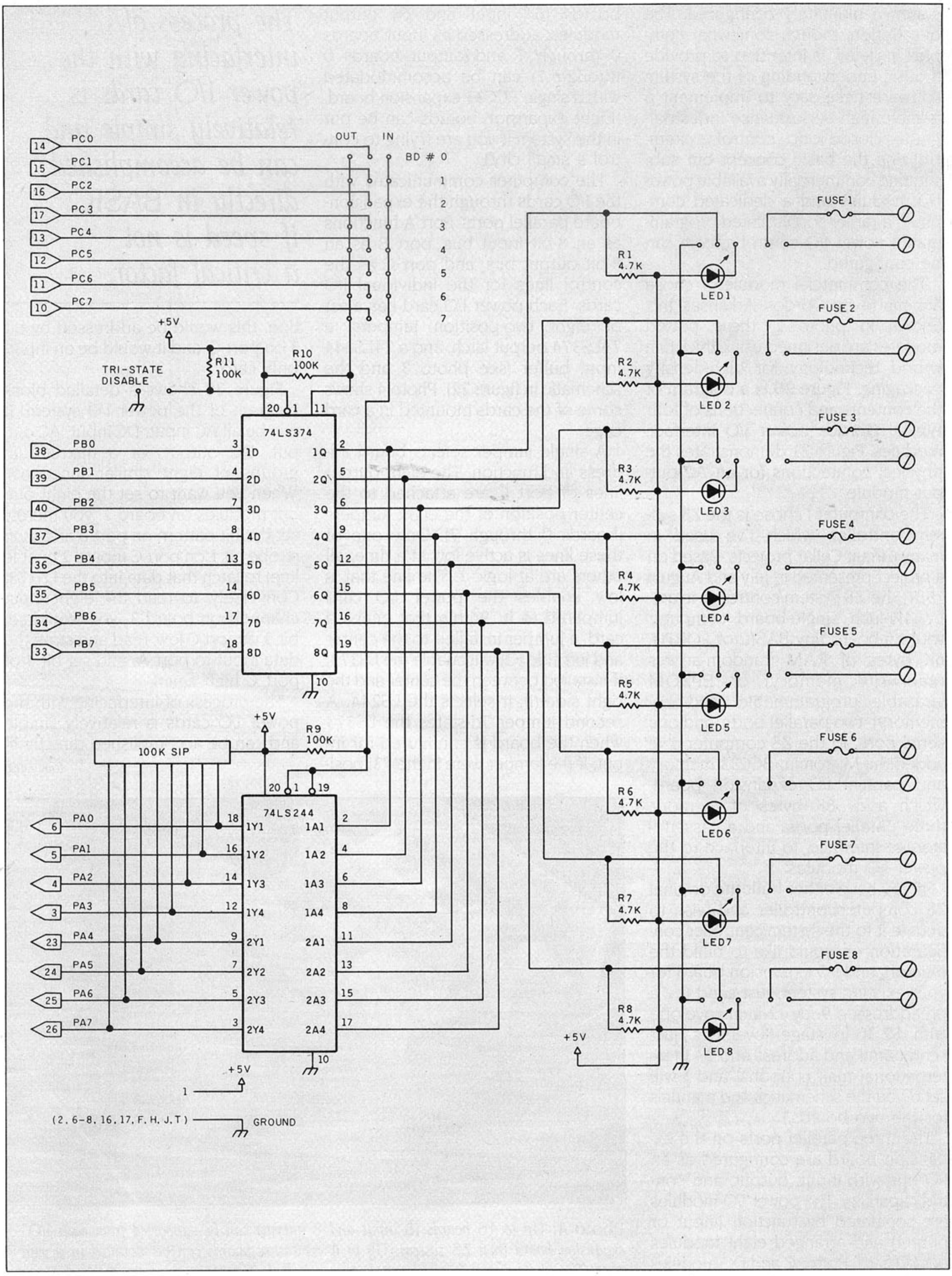

Figure 22: *The schematic of the Z8 power I/O card, as shown in photo 3.*

BASIC if speed is not critical. With 8 or 10 modules, it is not a problem to scan and record change of state, read the real-time clock, and still meet the requirements of the application.

While the simplified hardware for the power I/O system is important, it takes more to produce an industrial-grade control system. It is counterproductive to run time-consuming, repetitive tasks in BASIC that can be done more quickly in assembly language. For that reason, I've added a set of interrupt-driven utilities that greatly simplifies the interaction between user and power I/O system and allows the use of BASIC (unless you prefer assembly language), even with 64 active I/O channels.

These Z8 assembly-language routines, flowcharted in figures 24 and 25, operate as background tasks to any user application programs and are completely transparent. In addition to real-time clock functions, they allow the user to interact with the I/O system through a table of 64 input and output values rather than setting and reading expansion ports. To turn output channel 16 on, we simply load a value greater than 0 into table location 16. To turn output channel 1 off, we load 0 into table location 1.

Conversely, all inputs are continuously scanned and the present values loaded into a similar channel table for examination. In addition to the present value, a separate indication of change of state by board and channel number is also produced. The change-of-state indication is maintained until the user reads the affected channels. The result is a simple BASIC single-byte read-and-compare to find any input channels that have changed and a single-byte write to make a corresponding control output.

The user can drive seven subroutine calls in dealing with the power I/O system. They are

1. System initialization.
2. Read an input channel's change-of-state flag (1 bit).
3. Read an input channel's data bit and reset change-of-state flag.
4. Set an output-channel data bit (1 on or 0 off).
5. Read an input board's change-of-state flags (8 bits).
6. Read an input board's data bits and reset change-of-state flags.
7. Set an output board's data word (8 bits, 1 on or 0 off).

In addition to these subroutines called by the user, other routines under interrupt control update the clock/calendar values and the power I/O boards and data/change tables.

By the time you read this, the Circuit Cellar could be rewired. But I *might experiment with my new home-control system.*

The completed software fits on a 2716 EPROM on the I/O expansion board. Unfortunately, I don't have room enough here for a complete program listing, but I will send you one if you write to me.

In Conclusion

What started out as a very simple solid-state-relay project got a little carried away. I use all the devices I design, and this is no exception. By the time you read this, the Circuit Cellar could be completely rewired. On the other hand, I might wait a month and experiment a little more with the new Circuit Cellar Home Control System that's in the works. Com-

(continued)

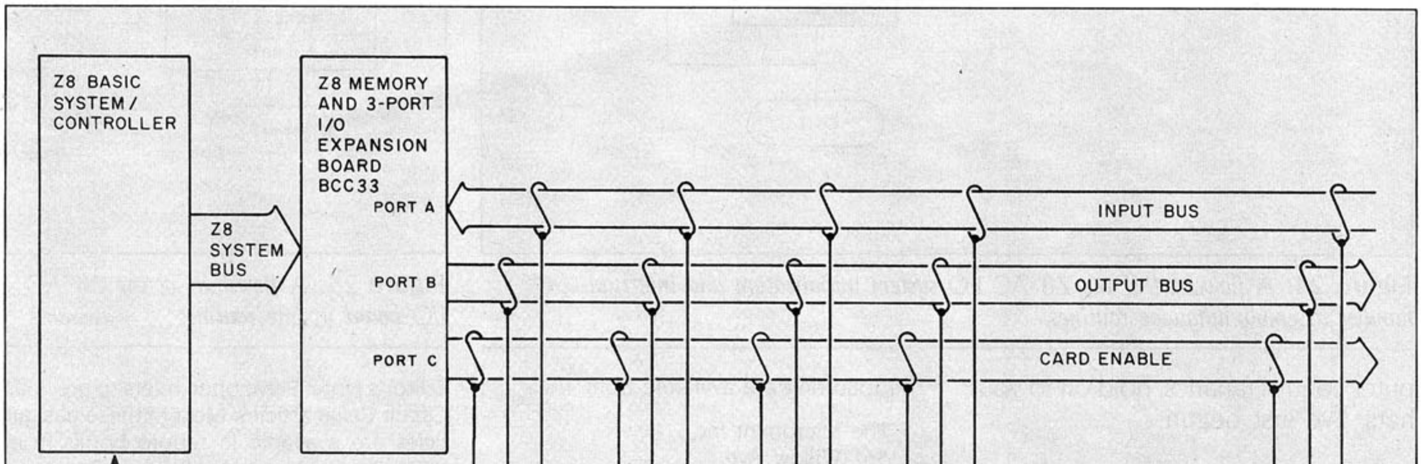

Figure 23: *The block diagram of the Z8 power I/O system.*

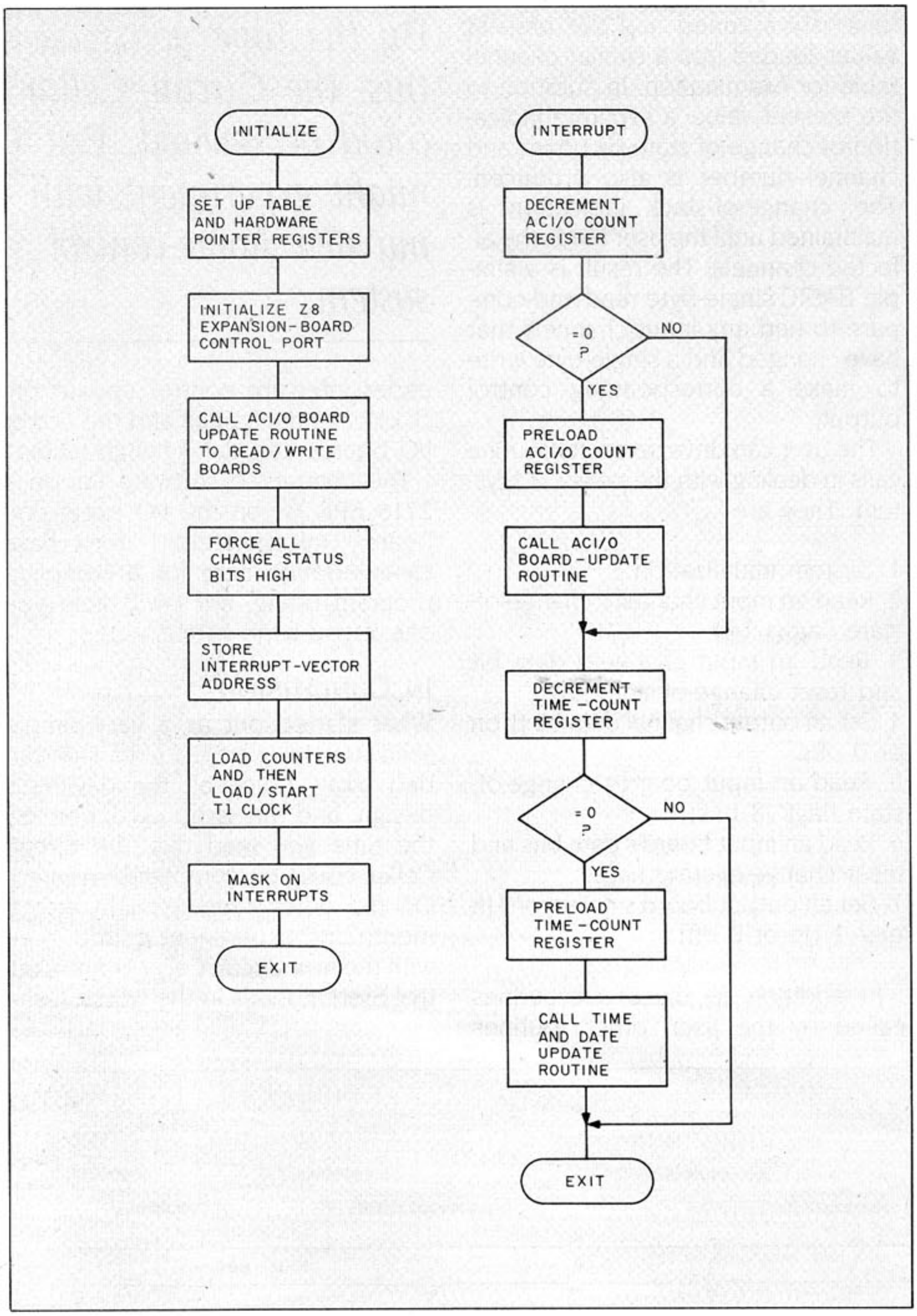

Figure 24: A *flowchart of the* Z8 AC I/O *system initialization and interrupt-handler assembly-language routines.*

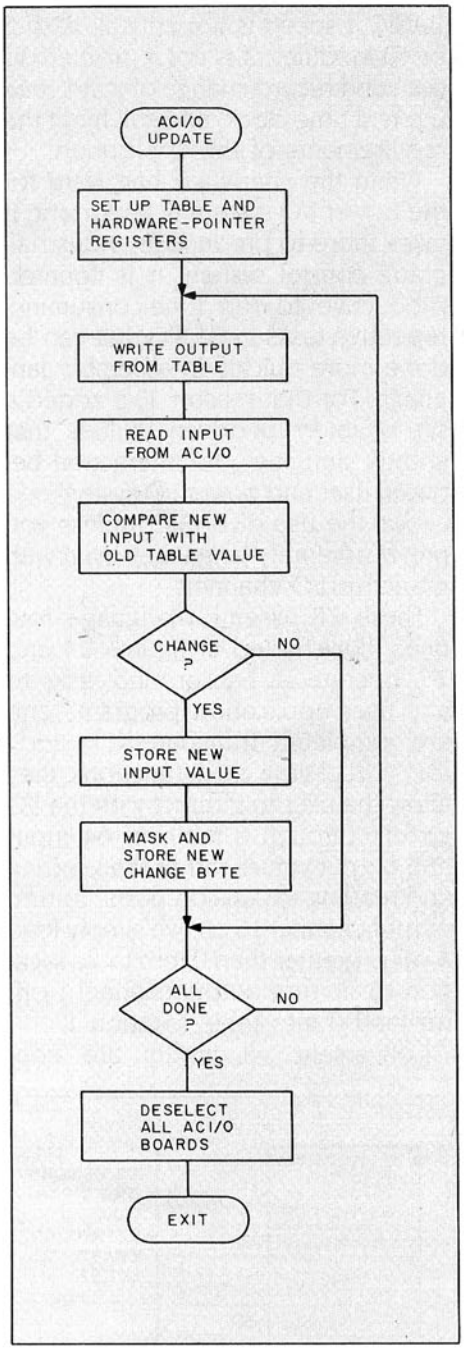

Figure 25: A *flowchart of the* Z8 AC I/O *board update routine.*

puter control fanatics, hold on to your hats, I've just begun.

CIRCUIT CELLAR FEEDBACK

This month's feedback begins on page 375.

NEXT MONTH

Tired of weak power supplies? Tune in next month, when I talk about linear power supplies. ■

Special thanks to Bill Curlew *for his software expertise. Diagrams of Gordos modules reprinted courtesy of Gordos Arkansas Inc.*

The following are available from

The Micromint Inc.
561 Willow Ave.
Cedarhurst, NY 11596
(516) 374-6793 for information
(800) 645-3479 for orders

1. Assembled and tested Z8 power I/O board without power I/O modules $159
2. AC input or output modules, types IAC5, IDC5, OAC5, and ODC5 . . . . $15 each
3. Eight-slot motherboard and card cage usable for I/O boards or Z8 . . . please call
4. Z8 system controller $149

Please include $4 for shipping in the continental United States, $10 elsewhere. New York residents please include 8 percent sales tax.

Editor's Note: Steve often refers to previous Circuit Cellar articles. Most of these past articles are available in reprint books from BYTE Books, McGraw-Hill Book Company, POB 400, Hightstown, NJ 08250.

Ciarcia's Circuit Cellar, Volume I covers articles that appeared in BYTE from September 1977 through November 1978. *Volume* II covers December 1978 through June 1980. *Volume* III covers July 1980 through December 1981. *Volume* IV covers January 1982 through June 1983.

To receive a complete list of Ciarcia's Circuit Cellar project kits, circle 100 on the reader-service inquiry card at the back of the magazine.