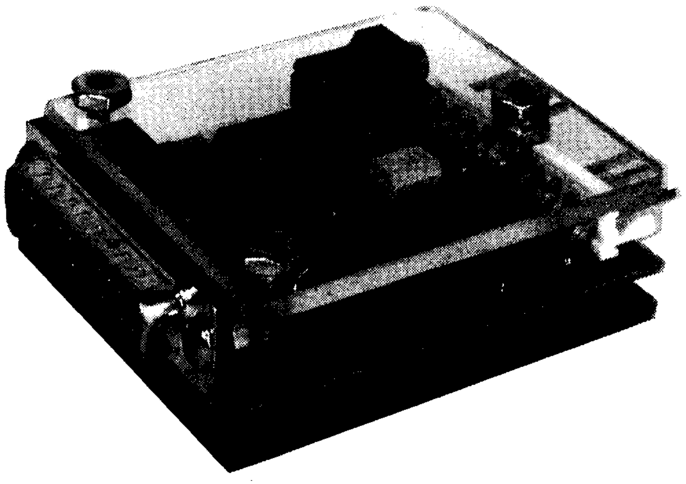
AN AC PROPORTIONAL VOLTAGE CONTROLLER FOR YOUR PC

NEIL BUNGARD



Use YOUR PC to control appliance voltage levels.

While it is easy to control appliances from a PC, that type of control usually has one of two conditions: on or off. There are many applications where it would be desirable to proportionally control the voltage level of an appliance that uses 11 0-volt AC service. Light dimming is one obvious application. Other uses include motor-speed control; heater control; and the ability to easily create an inexpensive, programmable DC power source.

In this article, we're going to show you how to use the parallel port of a PC, a few optical-coupling devices, and a Microchip PIC microcontroller to control the conduction angle of a 1 10-volt AC source, and hence to proportionally control a device connected to the PC's parallel port.

Controlling AC. Let's begin by looking at a 60-Hz 11 O-volt AC sine wave and talk about the concept of con- *duction angle.* Figure 1 A is the circuit that will be used for this discussion. We'll assume that S1 is an electronic switch that can be activated instantly at any time. If S1 is closed and remains closed, Fig. 1B would represent a single cycle of the signal that would be seen by load resistor R1 . In that case, the load draws current during the entire cycle and would have a conduction angle of 360 degrees. In Fig. IC, S1 does not close until the sinewave reaches 90 degrees in the cycle. As the signal passes through zero at 180 degrees, S1 is opened, stopping the flow of current to R1 for the next 90 degrees. At 270 degrees, S1 closes again and current once again flows through R1 until the zero crossing at 360 degrees. The signal in Fig. 1 C has a conduction angle of 180 degrees. Since, in general, the conduction angle is the number of degrees that current passes through the load in each cycle, that means that the amountbf current seen by R1 would be half the amount that would otherwise have passed through it.

In general, that is the way that an ordinary wall-mounted light dimmer works. The physical position of the light-dimmer knob sets the conduction angle at which conduction will begin after each zero crossing. Additionally, when the sinewave passes through zero, conduction is automatically stopped. Of course, that example controls the conduction angle mechanically. Such an arrangement is fine for open-loop control applications where you set the control once and leave it alone. For many control applications, however, you are not interested in just setting a conduction angle and letting the process run. You will want to monitor the process and dynamically adjust the conduction angle to keep the process operating at a preset level.

To dynamically control the conduction angle, you will need to know precisely what the angle is and be able to make adjustments accordingly. That is done by finding when the sinewave passes through zero volts. There are many circuits for determining the zero-crossing point. If you need a high degree of precision, op-amps can be used to get within a few microvolts. Most applications, however, do not need that level of precision and a 'quick and dirty" way of determining the zero-crossing point with a TTL-compatible output signal is to use an opto-coupler.

The opto-coupler circuit shown in Fig. 2A will work well but has some inherent shortcomings. Current-limiting resistor R1 must limit the forward current through the opto-coupler's diode and the external clamping diode to within the maximum limits for the devices at the peak voltage, which is about 156 volts when using wall current. The peak-inverse-voltage ratings of the diodes should also be at least 160 volts.

Another consideration is that the opto-coupler's output waveform will be non-symmetrical. That is because the input signal on the opto-coupler's diode must exceed the diode's forward-bias turn-on voltage before the diode conducts. That phenomenon has been exaggerated in Fig. 2B to illustrate the concept. Since the actual turn-on voltage is slightly above zero volts, the output will not have a perfect

50% duty cycle. Another problem that is related to the turn-on-voltage condition-and is much subtler-is that the diode's turn-on and turn-off voltages are not exactly the same. This principle, called hysteresis, is exhibited by most electronic components. It is especially noticeable in devices that monitor or control threshold activities such as zero crossing. The symmetry and hystersis effects can be compensated for; we will see how to use software to compensate for those conditions later in this article.

Now that we are able to detect the zero-crossing portion of an AC cycle with a TTL output signal, how do we control the current to a load with a desired conduction angle? A general solution to that problem is shown in Fig. 3. A PIC microcontroller monitors both the zero-crossing signal from the opto-coupler and an 8-bit data byte from the PC's parallel port, which represents the desired conduction angle. By comparing the two pieces of input data, the PIC decides when to trigger the electronic switch. The electronic

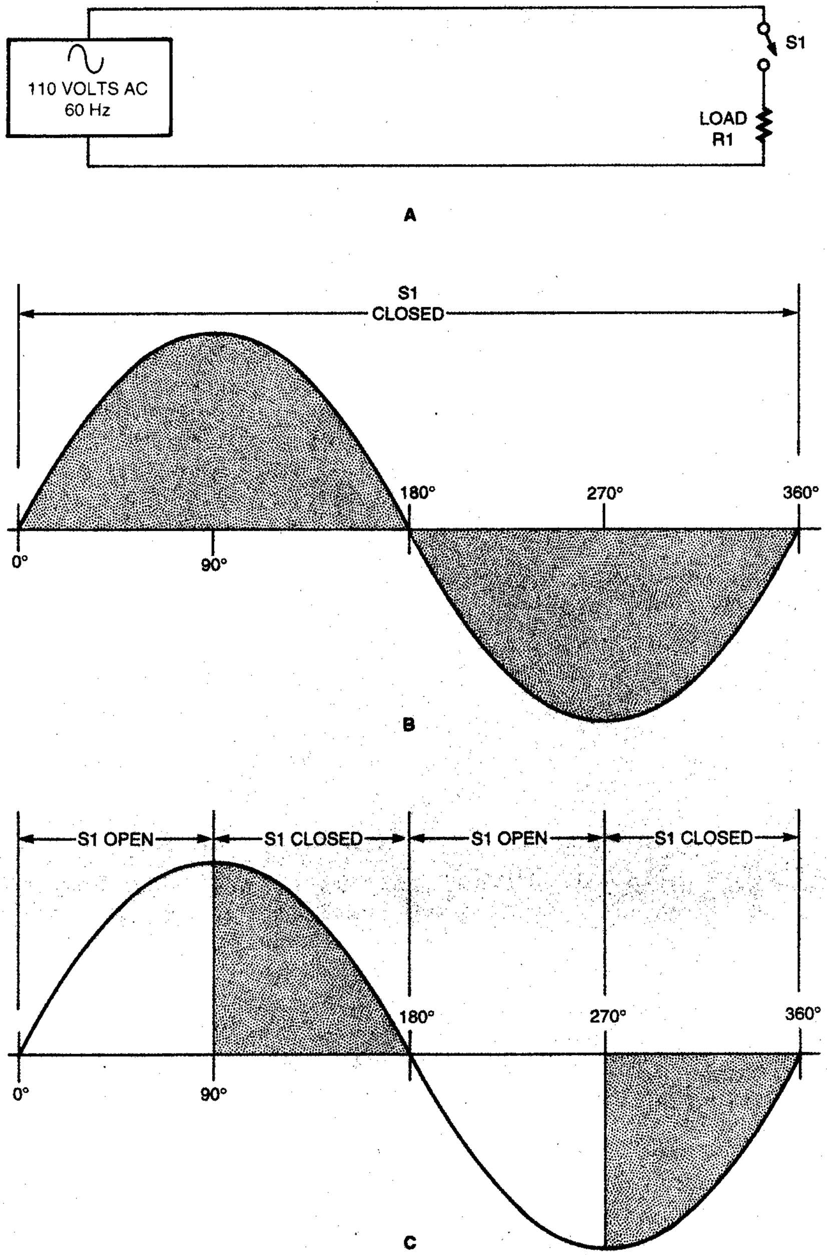


Fig. 1. Varying the AC current to a load is simply a matter of opening and closing a switch at the right time (A). If the switch is left closed for the entire cycle, the load will see full power (B). However, if the switch is opened and closed at the same points in the cycle, the apparent voltage applied to the load will be averaged out. In example (C), the load will only see 50% of full power.

switch we'll use in our controller circuit is actually a Triac that is triggered by the PIC. The sidebar, "Triacs, Diacs, and Control", explains the basics of how four-layer semiconductors are designed and used.

Using an 8-bit word to represent the desired conduction angle lets us control the conduction angle of the AC waveform in 256 discrete steps. That is probably more than enough resolution for most applications and appears continuous when being used in light-dimmer or heater-control applications. The sidebar, "The Parallel Port," reviews the operation of the PC parallel port and how it interfaces to the outside world.

Circuit Description. Bringing all of the concepts we've discussed so far together yields our project, called the Parallel-Port Controller. Its schematic drawing is shown in Fig. 4. Hardware operation for the entire circuit should be familiar based on the previous discussions.

Wall-socket current is applied to

PARTS LIST FOR THE PARALLEL-PORT CONTROLLER

SEMICONDUCTORS

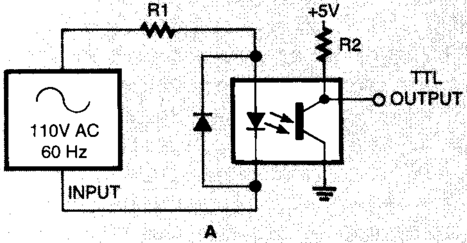
IC1-PIC14C544 microcontroller, integrated circuit
IC2-H1 1AX optoisolatar, transistor-based, integrated circuit
IC3--MOC3010 optoisolator, Triac output, integrated circuit
TR1-2N6347 Triac,
D1--1N4004 silicon diode
D2-lN523 l Zener diode

RESISTORS

(All resistors are ¼-watt, 5% units unless otherwise noted)
Rl, R5-100-ohm
R2-1 OOO-ohm
R3-11,000-ohm, ½-watt
R4-l8O-ohm

ADDITIONAL PARTS AND MATERIALS

C1--O.Ol-µF capacitor, ceramic-disc
Jl, J2-4-pin right-angle connector (Molex 26-60-5040 or similar)
J3---DB25 right-angle male connector (Mouser 152-3325 or similar)
J4---Co-axial power connector, male (Mouser 161-3112 or similar)
RES1--8-MHz ceramic resonator
Wall transformer, Plexiglas covers, hardware, etc.



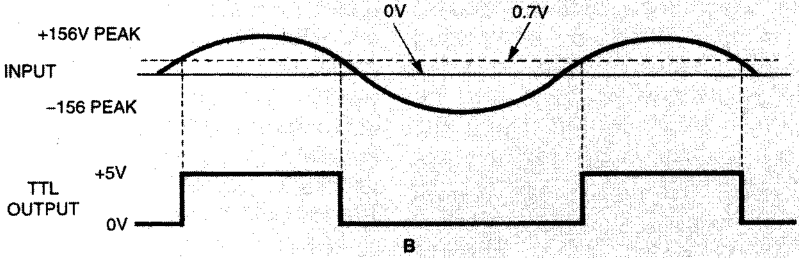


Fig. 2. It is very easy to sense the zero-crossing point of an AC cycle with a simple optoisolator cir-cuit (A). Unfortunately, the digital output of such a circuit will not be a perfect squarewave because of the voltage drop across the internal photodiode (B).

J2. The AC waveform is applied to IC2 through R3 with D1 providing a reverse path as discussed before. The TTL squarewave is applied to pin 17 of IC1.

When the PIC program decides that the output should be turned on, an output signal from pin 18 of IC1 activates the photodiode of IC3. Current for the diode is supplied by RI. The output of IC3 is applied to the gate of TR1, which completes a path between the main connections of J1 and J2. The load being controlled is connected to J1 .

An 8-bit value for the amount of delay is applied to J3 from a PC's parallel port. Power for the circuit is connected to J4 and regulated by D2

TABLE 1– SPP MODE PHYSICAL AND LOGICAL PIN ASSIGNMENTS

Pin	Signal	In/Out	Register	Inverted
	Strobe	In/Out	control	Yes
2	Data 0	Out	Data	NO
3	Data 1	Out	Data	No
4	Data 2	out	Data	NO
5	Data 3	Out	Data	NO
6	Data 4	Out	Data	No
7	Data 5	Out	Data	No
a	Data 6	out	Data	No
9	Data 7	out	Rata	No
10	Ack	in	status	No
11	Busy	In	Status	Yes
12	Paper	Out / End In	Status	No
13	Select	In	Status	No
14	Auto Linefeed	In/Out	Control	Yes
15	Error/Fault	In	status	N o
16	Initialize	In/Out	Control	No
17	Select Printer	In/Out	Control	Yes
18	Ground	Gnd		
19	Ground	Gnd		
20	Ground	Gnd		
21	Ground	Gnd		
22	Ground	Gnd		
23	Ground	Gnd		
24	Ground	Gnd		
25	Ground	Gnd		

Software. Overall operation of the Parallel-Port Controller is really a function of the program running in IC1 , Compensation for the non-symmetry and hysteresis problems mentioned before are accomplished by "tweaking" certain variables in the program.

The block diagram of the PIC program is shown in Fig. 5. After initializing the variables that the program uses, the PIC waits for a positive-going zero crossing to occur. At that time, a value from the parallel port is read and used to determine when to trigger the Triac. Once the Triac has been pulsed on, the program waits for the negative-going zero crossing. When that occurs, the current parallel-port value is again obtained and the precise turn-on time for the negative half of the cycle is calculated. Once the Triac is triggered on for the negative half cycle, the program loops back to the top and the process is repeated.

A study of the source code will show the inner workings of The Parallel-Port Controller. The source code is available at the Gernsback FTP site (ftp.gernsback.com/pub/EN/ppc.txt).

After initialization of the program variables, a starting point is established and the "wait_for_pos" routine begins looking for a positive-going zero crossing. When the zero crossing is detected, the PIC moves a value from the parallel port into the variable **"temp1** ". At this point, a check is made to see if the Triac should be full on (temp1=0) or full off (temp1=255); if so. the appropriate action is performed, otherwise a call is made to a subroutine called "dwell".

The "dwell" subroutine does two things. First, it executes a timing loop that compensates for the premature zero crossing caused by the forward voltage drop of the detection diode. At the same time, the loop compensates for the hysteresis effect. The values for the compensation loop were found by testing several diodes and looking at their responses on an oscilloscope. Once an average time delay until the actual zero crossing occurs was found, compensation for the difference was done by manipulating

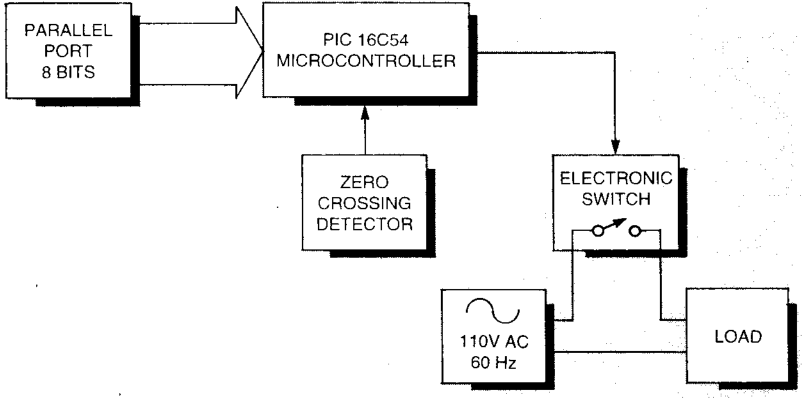


Fig. 3. Here's how to use a PIC 16C54 microcontroller to control the delay between the zero crossing of the AC cycle and switching on an output. An 8-bit number from the parallel port tells the PIC how long to wait.

the loop variables “await_cntrl " and “await_cntr2”.

The second thing that dwell does is to use the value obtained from the parallel port in a second timing loop that decides when to turn on the Triac. The larger the parallel-port value, the longer the loop runs before it times out. A longer delay will fire the Triac later in the cycle resulting in less average current through the load.

When the second loop times out, program execution resumes at the “wait_for_pos” routine where the Triac is actually turned on by clearing and setting RA.1 Once the Triac is turned on, the “wait_for_neg” loop looks for a negative-going zero crossing. When that condition is found, dwell1 is called again, using the parallel-port value to determine when to activate the Triac during the negative half cycle. Notice that compensation for the diode drop and the hystersis is missing in the positive-going zero-crossing case. That is because the detection occurs after the actual zero crossing. Since it is very difficult to travel into the past,

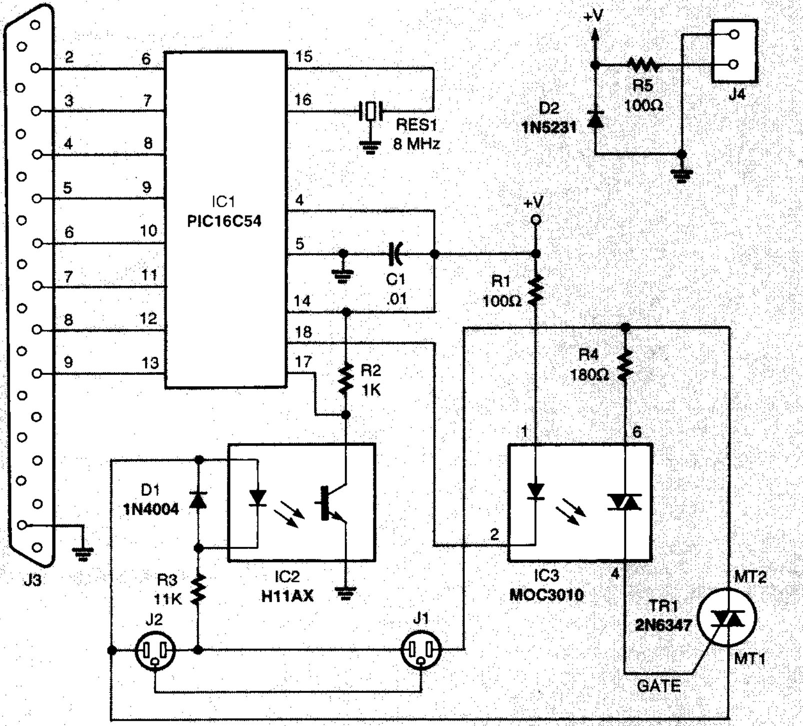


Fig. 4. The Parallel-Port Controller's schematic shows how simple the hardware is. Note that th output is also optoisolated by a Triac-output isolator.

the program code has been kept simple by ignoring that section of the waveform. The result is a small loss of control in terms of the entire cycle time. However, that presents no problem for most control applications. If that loss of precision is troublesome in your particular application, you can always replace the detection circuit (IC2) with an op-amp circuit to detect true zero crossing and regain the lost precision Of course, sections of the program would have to be modified to ignore hysterisis and symmetry compensation; those types of modifications are beyond the scope of this article.

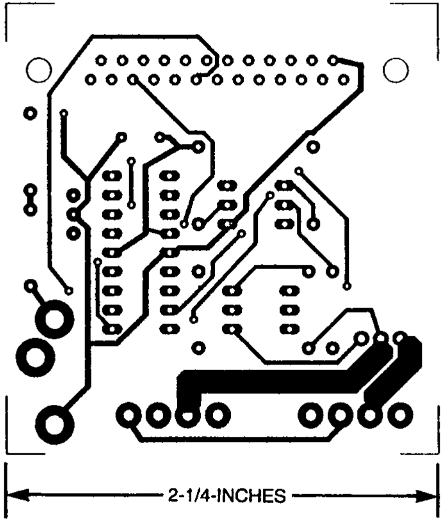
Construction. While the Parallel-Port Controller can be built on a perfboard using standard construction techniques, a PC board is recommended because of the high voltages involved. Any error in construction can damage equipment or

TRIACS, DIACS AND CONTROL

Both Triacs and diacs are four-layer semiconductor devices that can conduct current in either direction, The diac has two terminals. It is normally in an “off” state until a certain voltage level (the breakover voltage) is reached. At that point, the diac begins conducting current in the direction of the voltage polarity that is being applied across It.

The Triac, on the other hand, is like a diac with a third gate terminal. The Triac can be turned on by a pulse of gate current; there is no fixed breakover voltage needed. The breakover voltage in the Triac actually decreases with an increase in the gate current. The Triac can be thought of as two silicon-controlted rectifiers (SCRs) connected in inverse parallel with a common gate terminal. With that arrangement, the Triac can conduct currem in either direction when it is triggered on, Like the diac, the current direction depends on the polarity of the voltage across the Triac’s main terminals, Like the SCR, the Triac turns off when the anode current drops below a specified value, called the holding current The only way to turn off a Triac is to reduce the anode current to a sufficiently low level.

Triacs are primarily used to control average power to a load by phase control. In that method, the Triac is not triggered until a certain amount of time after the zero-crossing point of an AC waveform for both the positive and negative portions of the cycle.



Here's the foil pattern for the solder side of the Parallel-Port Controller.

TABLE 2-SOFTWARE REGISTERS

Offset	Name	Read/Write	Bit No.	Properties
Base +0	Data Port	Write (Read/Write if port is bidirectional)	Bit 7	*Data 7*
			Bit 6	*Data 6*
			Bit 5	*Data 5*
			Bit 4	*Data 4*
			Bit 3	*Data 3*
			Bit 2	*Dafa 2*
			Bit 1	*Data 1*
			Bit 0	*Data 0*
Base +1	Status Port	Read Only	Bit 7	*Busy*
			Bit 6	*Ack*
			Bit 5	*Paper Out/End*
			Bit 4	*Select*
			Bit 3	*Error/Fault*
			Bit 2	*IRQ* (Active Low)
			Bit 1	Reserved
			Bit 0	Reserved
Base +2	Control Port	Read/Write	Bit 7	Unused
			Bit 6	Unused
			Bit 5	Enable Bi-Directional Port
			Bit 4	Enable IRQ Via Ack Line
			Bit 3	Select *Printer*
			Bit 2	*Initialize*
			Bit 1	*Auto Linefeed*
			Bit 0	*Strobe*

Note: Italics indicates external signals

TABLE 3- LOGICAL PARALLEL PORT ADDRESSES

Address	Notes:
3BCh - 3BFh	Used for Parallel Ports which were incorporated on to Video Card- Doesn’t Support ECP Addresses
378h - 37Fh	Standard Address For LPT 1
278h - 27Fh	Standard Address For LPT 2
300h - 340h	Used for Prototype Board Development (64 individual Addresses)

result in severe injury. A double-sided foil pattern has been included here.

Start by cutting a pair of plastic covers that are the same size as the PC board. Drill three holes in the covers large enough for a 6-32 screw to pass through. Two of them should match the mounting holes for J3; the third should be drilled between J1 and J2 such that a 6-32

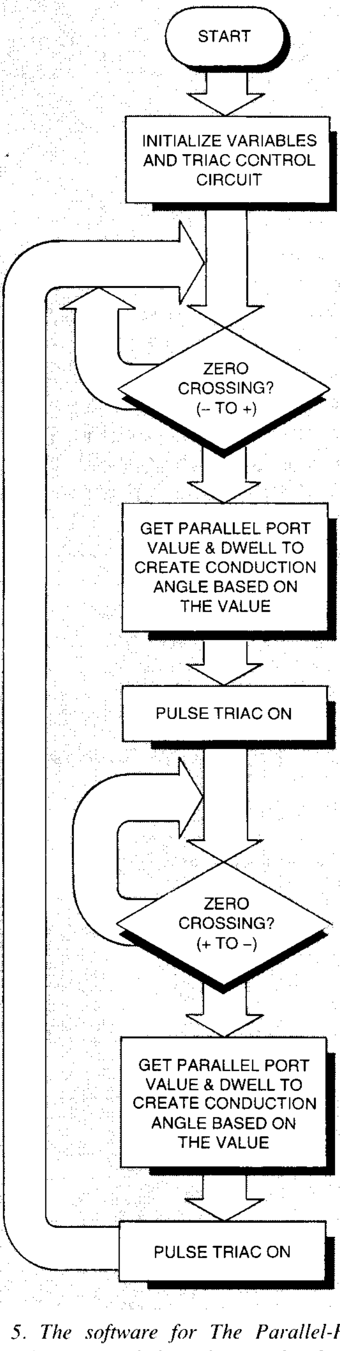
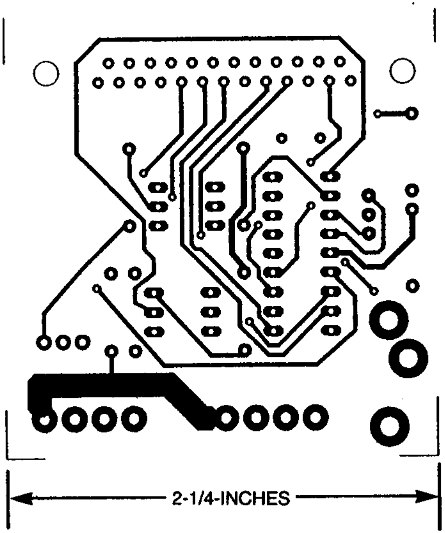

START
INITIALIZE VARIABLES AND TRIAC CONTROL CIRCUIT
ZERO CROSSING? (– TO +)
GET PARALLEL PORT VALUE & DWELL TO CREATE CONDUCTION ANGLE BASED ON THE VALUE
PULSE TRIAC ON
ZERO CROSSING? (+ TO –)
GET PARALLEL PORT VALUE & DWELL TO CREATE CONDUCTION ANGLE BASED ON THE VALUE
PULSE TRIAC ON

Fig. 5. The software for The Parallel-Port Controller is a simple loop that watches for the zero-crossing indicator, reads the delay time from the parallel port, and calculates how long to wait before turning the Triac on. The calculation also takes the delay effects of the sensing circuit into account.



Here's the foil pattern for the component side of the Parallel-Port Controller.

THE PARALLEL PORT

Prior to 1994, the interface to parallel ports was not formally standardized other than the information released by IBM when the first PC came out in the early 1980s. That lead to some frustration when trying to develop devices that could be used universally on different computers. In 1994, a formal standard, IEEE 1284-1994, was developed that defined the electrical, physical, and logical standards for the parallel port, That standardization efiminated many of the problems connected with designing hardware that could be used on any machine reliably. However, even today you will find hardware vendors that do not adhere to the IEEE 1284 standard-still making interfacing to a particular parallel port a process full of testing and guesswork.

The IEEE 1284-1994 Standard defines 5 modes of operation:

1. Compatibility Mode
2. Nibble Mode
3. Byte Mode
4. EPP (Enhanced Parallel Part) Mode
5. ECP (Extended Capabilities Port) Mode

The idea was to create a new standard that improved performance (speed and control) over the existing parallel-port specification, while at the same time maintaining backward comparability The compatibility, nibble, and byte modes can use the original hardware without any redesigning needed; they are collectively referred to as the SPP (standard parallel port) mode. The EPP and ECP modes are improved performance modes *that* imply backward comparability with the earlier SPP standard.

The compatibility mode is also referred to as the "Centronics" (named after the printer manufacturer that devised the hardware connector and pinout specifications) mode and is a write-only specification. In that mode, data can be transferred from the computer to a device at a maximum rate of 150 bytes per second. The nibble and byte modes are used for read operations and transfer data from external devices to the computer at a speed similar to the compatibility mode. All three of those standards use software handshaking techniques, which is responsible for their relatively low transfer rates. The EPP and ECP modes rely on additional hardware that is used for handshaking. That means that fewer I/O instructions are needed to transfer the same amount of data-increasing the data-transfer rate up to about two megabyte per second. The ECP has the additional advantages of direct-memory access (DMA) capability, which eliminates the I/O bottleneck and gives the port direct access to the microprocessor bus. In addition, ECP mode utilizes *run-length-encoded* (RLE) data compression to improve the overall speed of data transfer.

The connector standard specified by IEEE 1284 will work physically with SPP-, EPP, and ECP-mode devices, However, the logical pin assignments are different for the three modes, and are backward compatible from the ECP mode. In other words, SPP-mode devices will work with SPP-, EPC-, and EPP-mode interface boards. The EPP-mode devices will work with both EPP- and ECP-mode interface boards, and the ECP-mode devices only work with ECP-mode interface boards.

nut will not touch any of the PC board traces.

Microcontroller IC1 will have to be programmed before installing it in the circuit. Note that object code has not been supplied-only source code. That source code will have to be "compiled" into the numbers that IC1 will recognize as machine-code instructions. While it is tempting to "tinker" with the program, it is a good idea to compile the original source code first. Once the Parallel-Port Controller is working, modifications will no doubt suggest themselves to the person who has the knowledge and ability to rewrite the program. Again, those topics are beyond the scope of this article.

Follow the parts-placement diagram in Fig. 6. Before mounting J1 and J2, it's a good idea to "key" the two connectors in order to prevent anyone from accidentally attaching the AC voltage source to the wrong connector, Clip the fourth pin from J1 and the third pin from J2.

An important consideration is that all of the components be below the height of J3. When mounting TR1, bend the leads over so that the component lies flat on the PC board. Sockets can be used for the integrated circuits; their use will not make the height of the components excessive as long as low-profile sockets are used.

The plastic shields are held in place with screws and nuts. Start by placing the screws through the shield that will protect the solder side of the PC board. Use nuts to tighten the screws in place. Next, place the PC board over the screws. Run another nut down the screw that is between J1 and J2. The height of the nut should match the height of J3. Place the upper shield over the screws and tighten it down with three additional nuts. The whole unit will become a solid assembly.

Testing. Checking the Parallel-Port Controller can be done without writing a single line of code by using the old MS-DOS utility DEBUG. That programming utility still exists in today's operating systems; it can be found in the WINDOWS\COMMAND folder under Windows 95 or Windows 98.

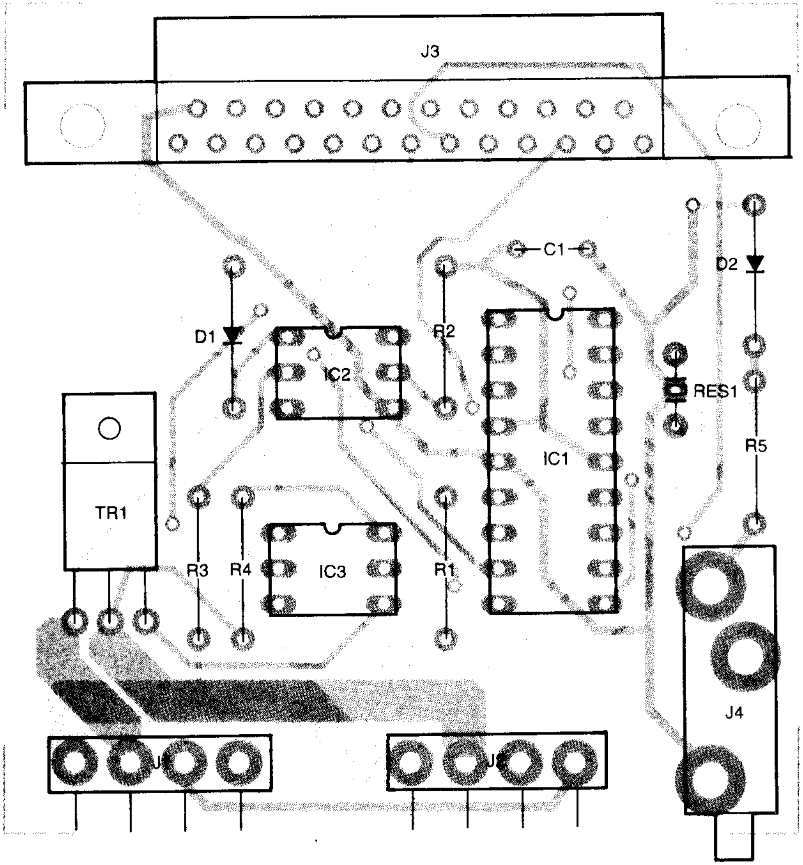


Fig. 6. The Parallel-Port Controller is a very simple device to build, thanks to the use of a pre-programmed microcontroller.

To test the Parallel-Port Controller, plug the controller directly into the PC's parallel port (LPT1) or use a 25-pin extension cable between the controller and the PC. Using an extension cable has the advantage of making the controller more accessible, which can be a big help if any troubleshooting is needed

Connect a common lamp to J 1; be sure that the lamp switch is on and that the bulb is good. Carefully connect a 110-volt AC cable to J2 and to a 110-volt outlet. Start DEBUG by going to a DOS prompt (or opening a DOS window). Change to a directory that contains the debug program and type DEBUG at the DOS prompt. Debug will respond to the command with a prompt of its own (-). That is known as the *debug prompt* (as opposed to the more familiar DOS prompt). At the debug prompt, you will type a simple output port command to send hexadecimal values between 00 and FF to LPT1. The command to turn the lamp full on is:

```
o 378,0
```

To turn the lamp full off again is:

```
0 378,ff
```

The command starts with the letter "o" followed by a space and the address of the printer port. If you are using LPT1, the address is 378; for LPT2, it is 278. Separate the data from the address with a comma. Any number between 00 and ff (hex) in the output port command will vary the intensity of the lamp between full on and full off. Try several values; remember that the higher the value, the dimmer the lamp.

With the Parallel-Port Controller working, you can now control the world from your desktop...or at least a few handy appliances!