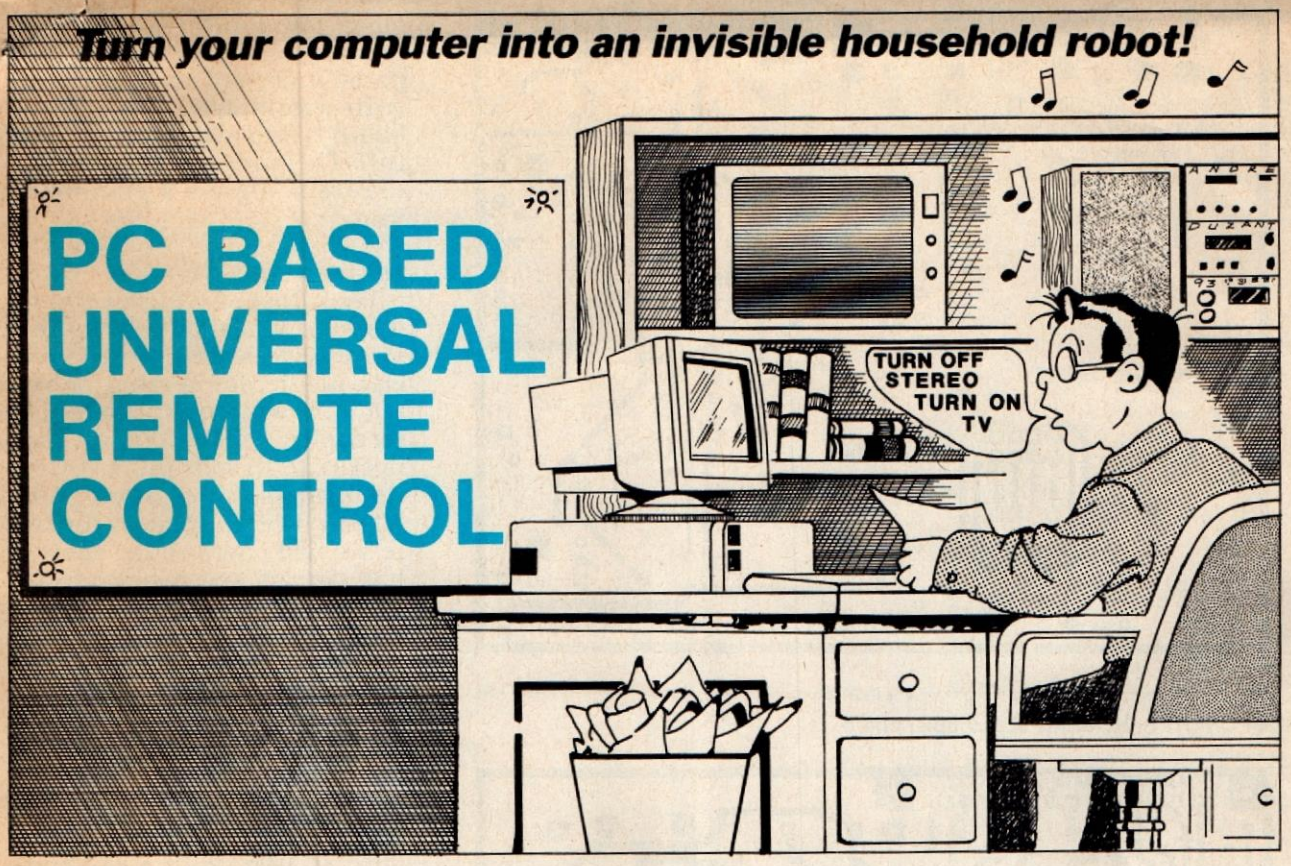
Turn your computer into an invisible household robot!

PC BASED UNIVERSAL REMOTE CONTROL



JON BEK

INVISIBOT IS THE NAME OF THE AUthor's home-control system. It stands for Invisible Robot. Invisibot started off by combining X-10 control with a voice-recognition and synthesis system. That gave the author the capability to say things like *Please dim the lights*, and have Invisibot respond accordingly. (Note that our sidebar contains detailed information on the X-10 and voice-control parts of the system.)

The one thing he wanted Invisibot to do that wasn't available "off-the-shelf" was to control TV's, VCR's, and other IR-controlled consumer-electronics gear. Even though we all have several remotes to control these toys, they always seem to hide under seat cushions, in magazines, and other unlikely places. When we do manage to locate a remote, it's usually the wrong one, or the batteries are dead. We're sure you've experienced the same problem.

Of course, there are the "smart" units that can control several devices, even ones from different manufacturers, but the thought of buying yet one more remote holds limited appeal. Invisibot was created to eliminate the need to push any remote-control buttons again.

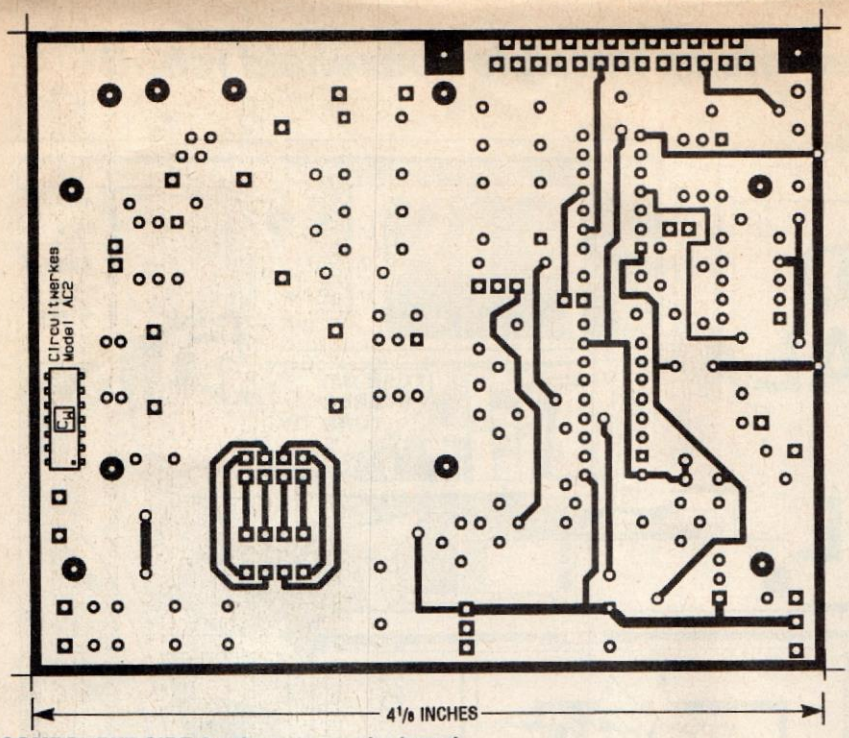
Universal remotes

There are two types of universal remote controls: learning and preprogrammed. Typically both can remember the codes for several devices. In addition to its infrared transmitter, the "learning" control also has an IR receiver that it uses to learn how to control your devices. You put the universal remote into a "learn" mode, point your other remote at it, and press a button. The universal remote then memorizes the pattern it receives. That pattern can be assigned to a button on the learning remote so that each time you press that button in the future, the remote will send the corresponding pattern.

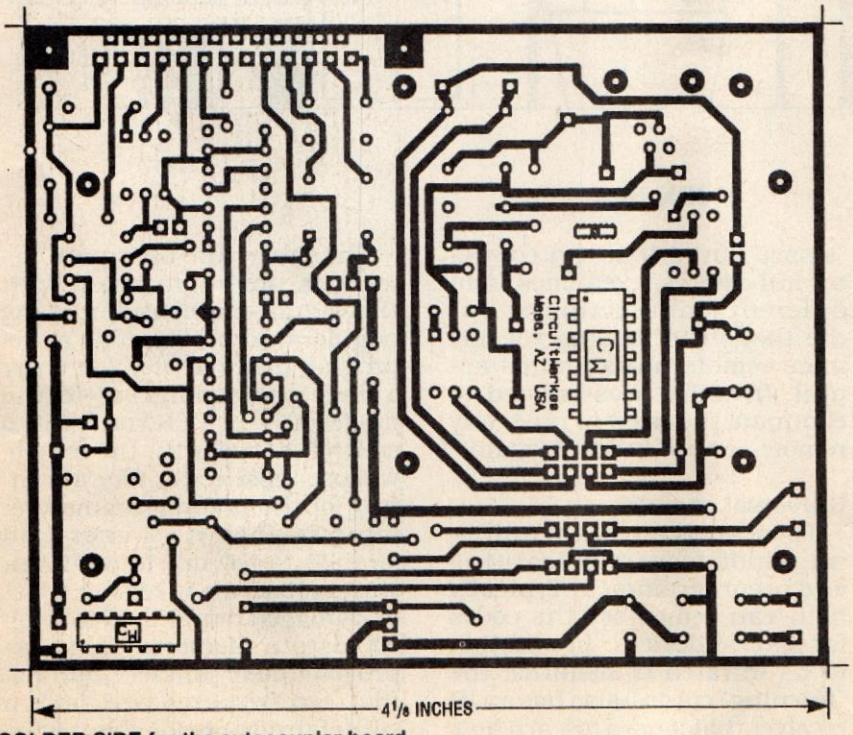
The other type of "smart" remote is preprogrammed with the signal patterns for many popular devices. To set up a preprogrammed remote, you enter a code corresponding to the model of TV or VCR you want to control; henceforth, the remote will use those codes. The advantage of the preprogrammed remote is that it's faster and simpler to set up; in addition, you can use one to replace a lost or damaged original. This project uses a Memorex AV-4 preprogrammed "smart" remote, that can be purchased from a corner drugstore for about $40.

Rows and columns

The AV-4 consists of one IC, a few resistors, several diodes, the infrared transmitter LED's, a capacitor, and a simple row-by-column, or matrix keyboard. Picture a tic-tac-toe board with columns labeled 1, 2, 3, . . . and rows labeled A, B, C, ... Each button on the keyboard corresponds to one letter/number



COMPONENT SIDE for the autocoupler board.



SOLDER SIDE for the autocoupler board.

and a piece of 8-conductor ribbon cable can be used to attach a stand-alone MPC-2 board (see **Electronics Now**, May 1993) to the autocoupler board, and then you would just leave out the MPC-2 parts on the autocoupler board. An 8-pin header can also be used to mount the MPC-2 board as a daughter board on top of the autocoupler board. Otherwise, you can mount all of the parts on the autocoupler board.

Solder the components on the board following the Parts-Placement diagram in Fig. 2.

Voltage-regulator IC7 should have a heat sink attached to it. Be sure to apply heat-sink compound on the back of the regulator before attaching the heat sink. The indicator LED's can either be board-mounted or panel-mounted and connected with wire jumpers. When the board is finished, insert the IC's into their respective sockets. Figure 3 shows a completed unit.

Initial testing

Verify that all IC's are properly installed and that polarity-sensitive devices are oriented properly before connecting power, audio, or a phone line to the autocoupler. Then, you can connect 12 to 18 volts (AC or DC) to the board's power-input pads. When power is applied, the two relays will energize for about a half second, and then drop out. That indicates the initial operation of the one-shot and latch circuits. If the unit is connected to a phone line when powered up, it will seize the line and hold it until a line current zero-crossing occurs, at which point the unit will drop the line.

Almost every modern central office in this country generates a zero-crossing in the telephone-line battery voltage less than a minute after dial tone is applied to the line—if no activity is detected in that period of time. That also occurs after the calling party hangs up. If your local central office is one of the rare systems that does not support that signaling, you might have to include a preset timer or, preferably, a dial-tone detector that will hang up your autocoupler automatically.

If the initial power-up occurs as has been described, you are ready to test the device on the phone line. Remember that the phone line carries voltages that are high enough to give you a nasty shock if you happen to be touching the tip and ring conductors on the bottom of the MPC-2 coupler section during a ring.

Connect the RJ-11 jack to your phone line with a standard modular cord, and have someone call your line. As soon as the phone starts ringing, you should see LED2 light up. The coupler should answer at the end of the first ring; when it

continued on page 72

pair. The goal was to devise a circuit that would connect to an existing PC simply and inexpensively, and activate the rows and columns of the keyboard matrix under software control.

A standard parallel port provides a suitable interface. All signals run at five volts, so connecting to the remote is straightforward. In addition, the printer port is directly addressable, even in BASIC, so programming wouldn't be a challenge. The only problem was how to control a 5-by-8 switch matrix using only the eight bits that comprise a standard parallel port. Five CMOS 4051 8-channel analog multiplexers make the job easy.

The 4051's, shown in Fig. 1, have three address inputs (A0, A1, and A2), eight channel inputs (0–7), and an input/output (I/O) pin. Each 4051 internally connects one of its eight channel lines to its I/O pin. The channel selected depends on the combination of signals applied to the input-select lines.

The address inputs are weighted in a binary fashion (A0 = 1, A1 = 2, A2 = 4). You select the desired channel by applying "highs" to the appropriate address inputs. For example, to select input channel 5, you would apply +5 volts to the A0 and A2 address inputs. The three address inputs provide a total of 2^3, or eight combinations, ranging from 0 to 7.

The 4051 has one other input, chip enable, or $\overline{EN}$ for short. When $\overline{EN}$ is brought high, the IC prevents any connection between I/O and the input channels, regardless of the states of the address inputs.

Circuit details

The complete circuit, shown in Fig. 2, consists of five 4051's, some pull-up resistors, and a connector. Note that the eight columns from the remote's keyboard form a "bus" to which the eight channel lines of all five 4051's are connected in parallel. Also note that each of the remote's five rows connects to the I/O line of a different 4051. Thus, all the channel lines will be selected simultaneously, but with only a single I/O line.

The computer's parallel port connects to J1. Of the eight lines, three form a bus that drives the address inputs, and each of the other five lines drives a separate $\overline{EN}$ input on the 4051's. That arrangement presents the same binary input combination to all five 4051's, but as long as the software drives only one of the five $\overline{EN}$ lines high, only a single row-column connection can be made at any one time.

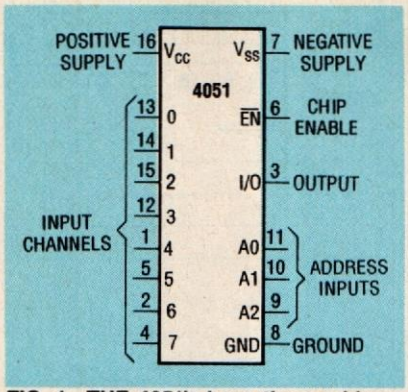


FIG. 1—THE 4051's have three address inputs, eight channel inputs, and an I/O pin. One of the eight channels is connected to I/O pin.

To understand how the circuit works, let's go through an example. Assume first that the PC has set all the $\overline{EN}$ lines high, so all the 4051's are off, and no row-column connections exist. Now assume that we want the computer to "press" the key corresponding to intersection A1 in the matrix. That corresponds to a binary value of 001 at the address inputs. By driving A0 high and the other two (A1 and A2) low, we obtain 001.

With the 4051's still disabled, the PC drives the $\overline{EN}$ line of IC1 low, which turns the device on. That makes the connection between row A and column 1, just as if we'd physically pressed that key.

MY FRIEND, INVISIBOT

I've always been fascinated by robots. Since the first time I saw "Forbidden Planet" as a child, I was certain that someday one of those electronic servants would cater to my every whim. I still don't have anything that looks like Robby the Robot, but I do control many of the lights and appliances in my house with spoken commands—and my invisible robot responds verbally as well!

The primary elements of "Invisibot" came as two off-the-shelf accessories for my PC: the X-10 CP290 computer interface, and the Covox Voice Master Key system.

The X-10 product has been around for years; it consists of a master control unit and one module for each appliance you want to control. The modules can be installed in place of normal light switches in the wall, or plugged in between the AC power outlet and the device being controlled.

The X-10 system communicates commands to the control modules by taking advantage of the fact that higher frequencies can piggyback on top of lower frequencies. The power service in your home is low-frequency 60-Hz alternating current. The X-10 system injects high-frequency signals via the power outlet on the 60 Hz. The information encoded in those high-frequency signals then travels over your existing house wiring to the control modules, which decode and act on it.

Each module must normally be set to a unique address; the control module sends one command at a time to a unique address.

Modules vary in price and functionality, but their average cost is about $12. You can purchase compatible modules from Radio Shack, Sears, and Stanley hardware dealers.

The CP290 controller attaches directly to the serial port of your PC, and is sold with software that allows you to start controlling your home immediately. The CP290 typically sells for about $40; you can purchase one from Egghead Software or Radio Shack. I was fortunate enough to be able to purchase mine some time ago for $19 on a close-out sale.

The other major component of the Invisibot consists of a Covox Voice Master Key, which provides both speech recognition and synthesis. The Master Key plugs into a standard PC bus; it can record and play back speech or other sounds, and includes an excellent speech-recognition package.

A memory-resident software module allows you to store a set of commands, each of which can be sent to DOS when the board recognizes a given phrase. For example, when I speak the command, "Oh butler, brighten the den," the system sends the proper commands to DOS to: 1) Play back a prerecorded voice file containing the words, "Yes sir! Right away sir!" and 2) Run the X-10 control program with the proper parameters to increase brightness in the den to 100%.

Not being content to control things through X-10 modules, I then designed the universal remote control project described here. Ω

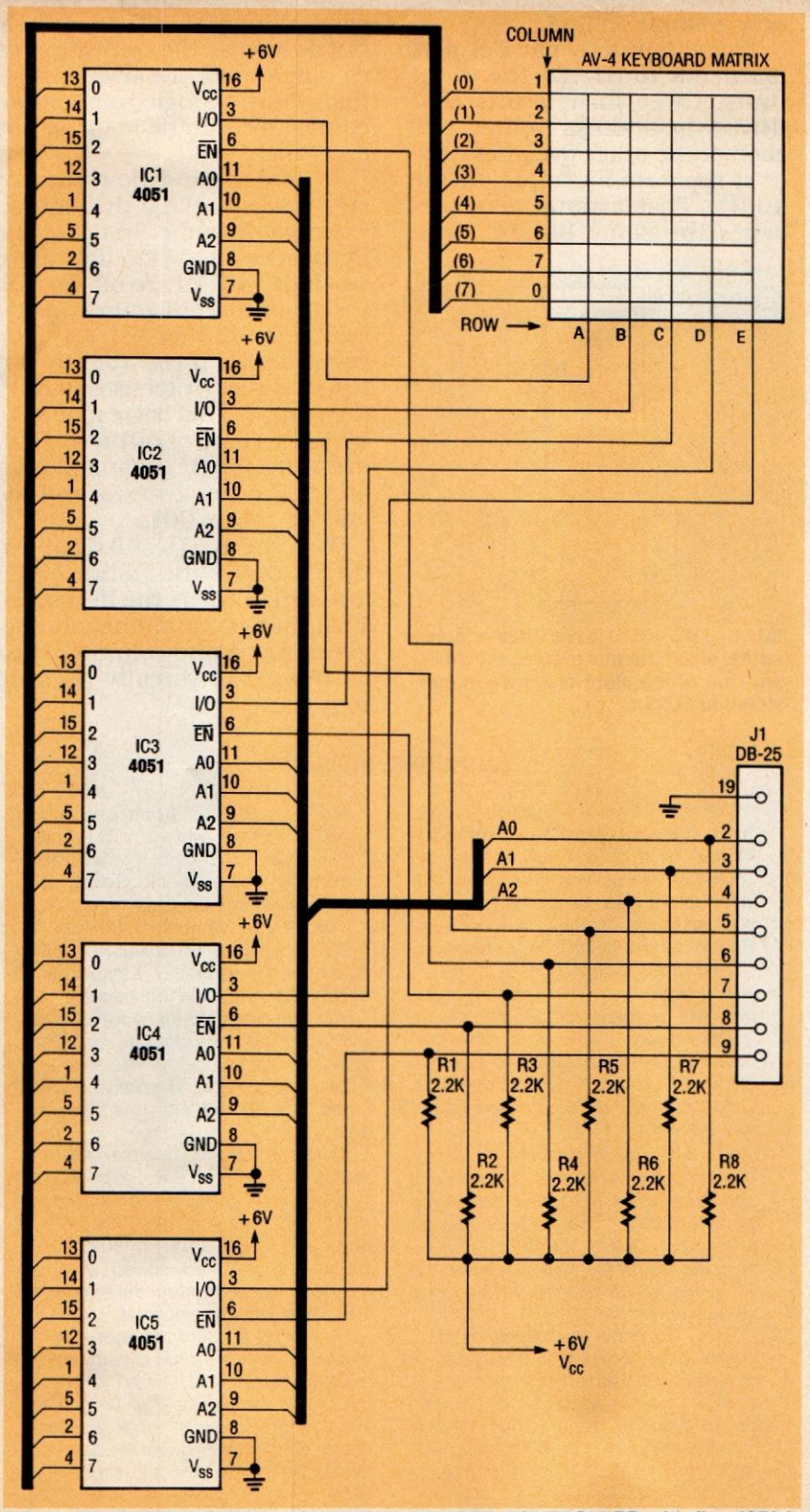


FIG. 2—INVISIBOT DRIVES A UNIVERSAL REMOTE CONTROLLER with five 4051 1-of-8 multiplexors.

Remember that until $\overline{EN}$ goes high again, the remote control will act as if the key remains pressed. Hence the software must reassert $\overline{EN}$.

Construction

Because of the simplicity of the circuit, the prototype was built using wire-wrap and point-to-point techniques. No PC board patterns are provided. The following are instructions for disassembling the remote control, mounting its PC board on a piece of perforated board, wiring the 4051's, making the remainder of the connections, and installing the device in a suitable enclosure.

First, remove the PC board from the AV-4 and clip off the metal battery contacts, leaving long power leads. Next, identify the row and column locations of the keyboard matrix. Then, using a hobby knife, carefully scrape a small patch of insulating paint from each trace of the matrix, leaving its shiny copper surface exposed.

Carefully solder connecting wires to the exposed traces of the AV-4 board. Cleaning them with a new rubber eraser or alcohol swab will help make a better solder connection. Tinning the wire and the traces before soldering will also help.

Mount the components on the perforated construction board, leaving enough space for the AV-4, as shown in Fig. 3. Mount five 16-pin DIP sockets for the 4051's, and a sixth to be used as a socket for the eight pull-up resistors. Then make all connections, using Fig. 2 as a wiring guide. Remember to connect pin 16 of each IC to V_{CC}, and pins 7 and 8 to ground.

Connect the power leads of the circuit and the AV-4 to the wall-mount transformer, *carefully observing polarity.* Bend the resistor leads to fit cleanly into the DIP socket, remove excess lead length, and insert the resistors. Now insert the 4051's,

PARTS LIST

R1–R8—2200 ohms, ¼-watt
IC1–IC5—4051 CMOS 1-of-8 decoder
J1—25-pin female DB-25 connector
6-volt DC, 300 mA, wall-mount transformer
Six 16-pin DIP sockets
Perforated construction board
Universal remote control (Memorex AV-4 or equivalent)
6-foot male-to-male DB-25 ribbon cable
Metal project case large enough to accommodate construction board and remote control
PC-board standoffs and mounting hardware

observing normal rules for handling electrostatic discharge-sensitive devices. Now we're ready to test the controller.

Important: Always apply power to the interface before attaching it to the PC, and disconnect it from the PC before turning off the controller. Because the interface draws very little power, you might want to leave it on all the time.

Use a nibbling tool (or a drill and a file) to make openings in the project enclosure for the DB-25 connector (J1), the power cable, and the AV-4's infrared LED's. Mount the board in the enclosure using four stand-offs, and the interface is complete.

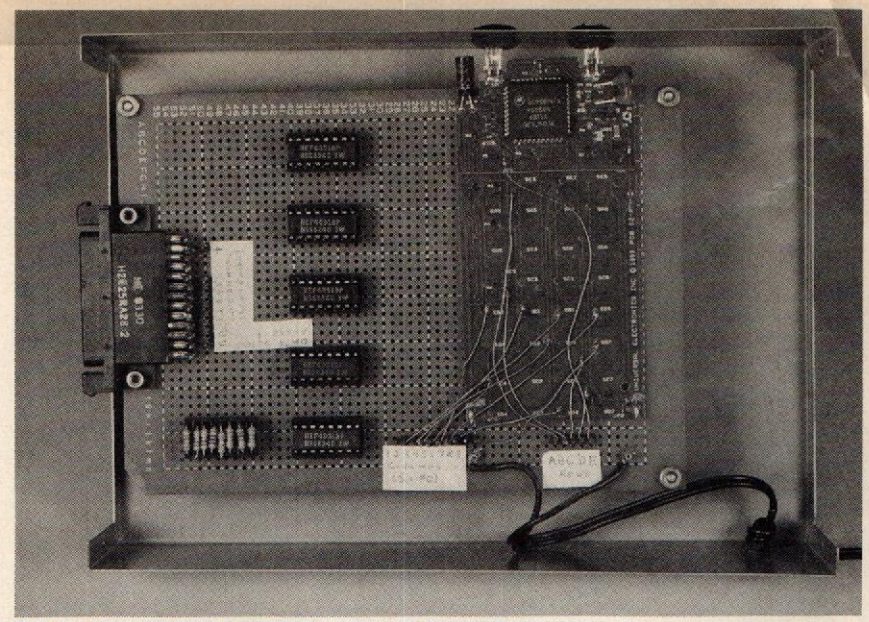


FIG. 3—MOUNT THE CONTROLLER AND CIRCUIT BOARD in the case as shown here.

Programming considerations

We don't have enough space to present a complete listing of the entire program. Moreover, if you don't use an AV-4 and a Sony TV, the commands won't work for you anyway. However, we present enough information so that you can test the unit and modify the command structure to suit your needs. In addition, the code will be posted on the *Electronics Now BBS* (516-293-2283, 1200/2400, 8N1) as a file called NVISIBOT.BAS. All code was developed in Microsoft QuickBASIC.

The interface uses only the data lines of the port, and ignores the control lines, so the software does not use the familiar LPRINT command. Instead, it uses the OUT instruction, which sends a byte of data directly to the specified port.

Listing 1 shows a set of constants that makes it easy to specify different rows and columns in the software. The first group (Chip1–Chip5) provides the values to enable each 4051; the second group shows the values to enable each row of the matrix; and the third group shows the values to enable each column. Enabling a particular position in the matrix is simply a matter of executing an OUT statement. (*Editor's Note: Determining the address of your LPT port can be tricky; more on that below. For now, assume use of LPT1 at an address of 0378 hex, or 888 decimal.*) To enable position A1 in the matrix, execute the command:

LISTING 1—INVISIBOT CONSTANTS

```
REM ********************************
REM * INVISIBOT INTERFACE CONTROL *
REM ********************************
REM
REM lpt 1 port = 378h = 888 decimal
REM data lines: A = 1, B = 2, C = 4
REM chip inhibit lines: chip 1 = 8,
REM chip 2 = 16, chip 3 = 32,
REM chip 4 = 64, chip 5 = 128

REM ****** chip ENABLES table *****
chip1 = 255 - 8 - 7
chip2 = 255 - 16 - 7
chip3 = 255 - 32 - 7
chip4 = 255 - 64 - 7
chip5 = 255 - 128 - 7

REM ****** data channel table *****
data0 = 1
data1 = 2
data2 = 3
data3 = 4
data4 = 5
data5 = 6
data6 = 7
data7 = 0

REM ****** chip correspondence ****
REM | chip# | URC Row |
REM     1         A
REM     2         B
REM     3         C
REM     4         D
REM     5         E

rowa = chip1
rowb = chip2
rowc = chip3
rowd = chip4
rowe = chip5

REM ****** data correspondence ****
REM |  ABC  | URC Column |
REM    000        1
REM    001        2
REM    010        3
REM    011        4
REM    100        5
REM    101        6
REM    110        7
REM    111        0
```

LISTING 2—STALL ROUTINE

```
REM ********************************
REM *        STALL ROUTINE         *
REM *                              *
REM * make a URC keypress long     *
REM * enough to ensure command     *
REM * gets sent, then clear the    *
REM * keypress and allow circuit   *
REM * to settle                    *
REM ********************************
stall:
REM hold the button down
REM for a time ...
FOR i = 1 TO 1500
NEXT i

REM let all buttons up and wait
REM for a time ...
OUT 888, 255
REM FOR i = 1 TO 5
REM NEXT i

RETURN
REM ********************************
```

OUT 888, ROWA + COL1

Next execute a short delay, and then disable all rows and columns. Listing 2 shows a subroutine that delays processing after "pressing" a key, followed by a "release" of all keys.

Listing 3 shows a demonstration routine that selects the TV mode of the AV-4, turns on the TV, ramps up the volume, changes the channel, and turns the TV off again. Unless you use an AV-4 and a Sony TV, these commands probably won't work for you.

Listing 4 shows a diagnostic routine that enables each row and column of the matrix, requesting that you press Enter after each position. The routine will be useful for verifying that

LISTING 3—EXERCISE FUNCTIONS

```
REM ****************************
REM the following code exercises
REM some basic command functions

REM select the tv
REM as the device to control
PRINT "sending tv: a5"
OUT 888, rowa + col5
GOSUB stall
INPUT x

REM turn on the tv
PRINT "sending POWER: e6"
OUT 888, rowe + col6
GOSUB stall
INPUT x

REM crank up the volume
FOR k = 1 TO 20
PRINT "sending VOLUME UP: b6"
OUT 888, rowb + col6
GOSUB stall
INPUT x
NEXT k

REM change the tv channel
PRINT "sending CHANNEL DOWN: c6"
OUT 888, rowe + col6
GOSUB stall
INPUT x

REM turn off the tv
PRINT "sending POWER: e6"
OUT 888, rowe + col6
GOSUB stall

END
```

the circuit is wired properly. You should see a change in the state of each line as it is tested.

To determine which LPT port to use, run the program shown in Listing 5. It will display the decimal and hexadecimal addresses of those parallel ports that are installed on your machine.

Adapting for other devices

When writing your own control program, keep the following in mind: Because the interface does not include a backup battery, the control program should send the proper setup commands for your particular device when you first start the program. Because the PC can "press" the remote control's buttons much faster than a human being, you could even program it to send keystrokes on-the-fly, each time you send a command. Doing so would allow you to overcome the four-device limit typical of inexpensive remotes, allowing you unlimited control over TV's, VCR's, CD players, video-disc players, and other IR-controlled devices.

The circuit described here is not specific to the Memorex AV-4; it could be attached to almost any device with a 5×8 or smaller keyboard matrix.

With only slight modifications, it could control much larger layouts. One way to increase the capacity of the interface while still using only eight data lines would be to add more 4051's. With additional decoding, the five control lines could control $2^5 = 32$ different 4051's. That's a 32×8 matrix, or a whopping 256 keys! Ω

LISTING 4—SONY TEST COMMANDS

```
REM the next piece of code
REM initializes the URC for
REM a SONY TV (000)
REM the "INPUT x" statements
REM aren't necessary, they're
REM just here so you can
REM observe each command.
REM The AV-4 has a surface-
REM mounted LED that blinks
REM once as each command is
REM received, and multiple
REM times when a valid device
REM code has been
REM successfully programmed.
REM ************************
PRINT "sending A: e5"
OUT 888, rowe + col5
GOSUB stall

INPUT x

PRINT "sending V: e3"
OUT 888, rowe + col3
GOSUB stall

INPUT x

PRINT "sending 4: e1"
OUT 888, rowe + col1
GOSUB stall

INPUT x

PRINT "sending #0: b2"
OUT 888, rowb + col2
GOSUB stall

INPUT x

PRINT "sending #0: b2"
OUT 888, rowb + col2
GOSUB stall

INPUT x

PRINT "sending #0: b2"
OUT 888, rowb + col2
GOSUB stall

INPUT x
REM ************ end URC init
```

LISTING 5—LPT ADDRESSING

```
REM display addresses of lpt ports
REM jkh 2/21/93
DEF SEG = &H40
lpt1 = &H8: lpt2 = &HA: lpt3 = &HC
a = PEEK(lpt1) + 256 * PEEK(lpt1 + 1)
b = PEEK(lpt2) + 256 * PEEK(lpt2 + 1)
c = PEEK(lpt3) + 256 * PEEK(lpt3 + 1)
PRINT "Decimal", "Hex"
PRINT "LPT1:"; a, HEX$(a)
IF b > 0 THEN PRINT "LPT2:"; b, HEX$(b)
IF c > 0 THEN PRINT "LPT3:"; c, HEX$(c)
```

AUTOCOUPLER

continued from page 67

does, LED1 will light up. Connect an audio source to the audio leads and try sending some audio down the line. Your calling party should hear it loud and clear. Next, connect an amplifier and speaker, to the audio port and listen to the calling party. Shortly after the caller hangs up, the coupler should automatically drop the line. With your amplifier still connected, pick up an extension phone; you should hear the dial tone through your off-line coupler. That confirms the audio pass-through from the MPC-2 section that makes the coupler caller-ID compatible.

The remote connections can be tested with a VOM. When the coupler first picks up, pin 5 of the DB-25 connector will go low for a moment. Depending on which way you set jumper JU2, the contacts of relay RY2 (available at J3, the DB-25 connector, pins 1–4) will close either momentarily or will latch as the unit picks up. A ring signal will produce a high output at pin 9 of J3.

You should be able to force the coupler to pick up by momentarily grounding pin 13 of J3 via switch S1. Be sure you don't ground pin 13 directly (R16 must be in place) or you might damage IC5. You can force the coupler to hang up by grounding pin 8 of J3 via S1. Test the external inhibit by connecting pins 11 and 12 of J3 together. Then remove JU1 and have someone call the coupler; it should not answer. Next, connect pin 10 of J3 to ground; the coupler should answer.

Conclusion

The couplers can send or receive audio signals and they have been used as outgoing message centers, listen lines, and remote-control interfaces, just to mention a few. You'll find that it is an excellent and versatile telephone-line interface suitable for automatically connecting your telephone line to a variety of projects. Ω