Karl Brown

Build BERT, the Basic Educational Robot Trainer, Part 1

Even a child can program this talking robot, built from off-the-shelf components

Why is it, I asked myself, that very simple robots, even commercially available ones, require so much training to operate? This, I decided, was a problem, and I wanted to do something about it. What I eventually did was write a menu-driven, interactive control language intended to be simple enough for a ten-year-old to use. As it turned out, during field tests with children, I found that any child who could read could program a robot within one minute of hitting the keyboard.

This, I felt, was progress. The next step was to design a robot that my fellow computer club members could build and program. That was how BERT was born.

Simplifying the Project

My goal was to reduce the complexity of the project for those building a robot for the first time. Accordingly, all the hardware, the software, and the little bits in between have been designed and tested already. All the circuits have been designed around off-the-shelf components rather than expensive, hard-to-get technology. Most of the mechanical parts (switches, speakers) are inexpensive enough to purchase new (as opposed to scrounging through the junk box). Parts such as the printed circuit boards, gearbox, and ROM are available from Amarobot in Richmond, California. (A complete kit is available as well. See address at the end of this article.)

To build BERT (see photo 1), you will need only commonly used electronics tools: a fine-tip soldering iron, wire cutters, pliers, and so on. Photo 2 shows BERT's circuit boards assembled and cabled together. Figures 1 through 4 are complete schematics of BERT's circuitry, and figures 5 through 7 are assembly and parts location drawings. These drawings should provide all the information you need to assemble BERT's circuit boards. To program him, you will need a device capable of transmitting ASCII code at 300 baud, with 7 data bits, no parity, and 1 stop bit. In other words, almost any computer with a serial port, or a serial terminal itself, can be used.

While developing BERT and BERTL, his programming language, I was able to research and examine quite a few of the personal robots presently on the market. There was a wide range of on-board electronics, from a minimum of two driver chips to a maximum of a complete 68000-based system with a megabyte of RAM and two 500K-byte disk drives. The mechanics of all these robots were quite similar. The method of locomotion, almost without exception, was wheels powered by electric motors. Flashing LEDs for eyes, speech synthesizers, and obstacle sensors were found on nearly every "untethered" robot.

I determined that innovation was not really needed in the hardware. Rather, I felt that simplification was required in the

continued

Photo 1: *BERT in the Amarobot kit configuration.*

Karl Brown teaches electronics at Vancouver Community College. His hobbies include computer hardware design and juggling. He can be reached at Vancouver Community College, Electronics Department, 250 West Pender St., Vancouver, B.C., Canada V6B 1S9.

Photographs by Paul Avis

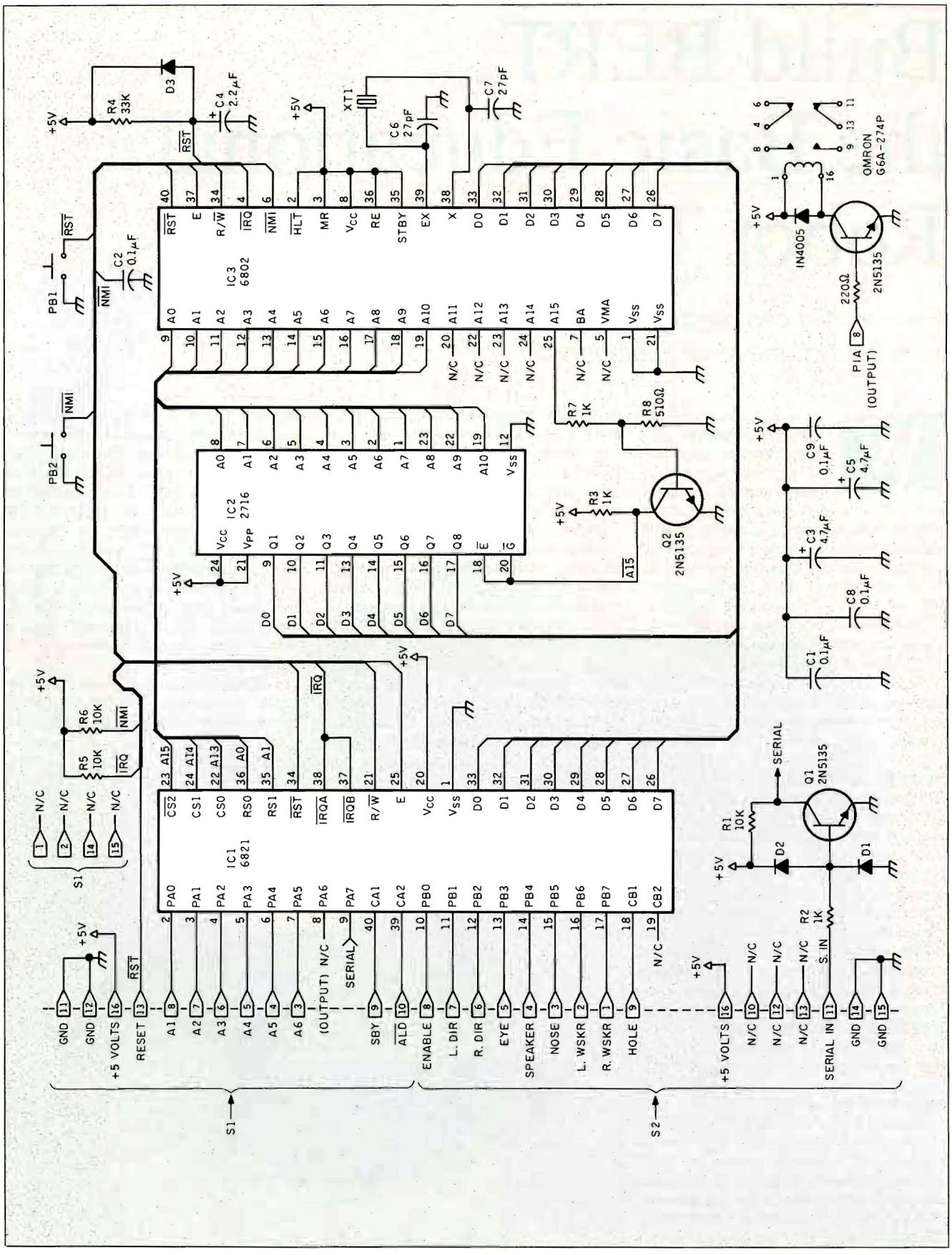


Figure 1: *Schematic diagram of BERT's on-board computer.*

software. It seemed to me that a different approach to robotics was needed. Instead of considering a robot as a robot per se, I started thinking of it as merely another peripheral for a computer. I felt that making a robot roam around should be no more difficult than making a printer print. In fact, I thought, I could actually treat the robot just like a printer.

Robot Printers

That is how I propose we control our robot—simply by sending it a string of ASCII text, which it will remember and then execute. The simplest command format I could think of looks like *Command, Parameter*, where *Command* is a single letter (say F for "forward") and *Parameter* is a hexadecimal number (00 to FF).

As an example, let's say we wanted our robot to traverse an area on the floor in the shape of a square. To accomplish this, he would have to execute the following maneuvers: move forward 30 centimeters, turn left 90 degrees, move forward 30 cm, turn left 90 degrees, move forward 30 cm, turn left 90 degrees, move forward 30 cm, and stop. Using the BERTL robot control language that I developed, that program would be F30, L15, F30, L15, F30, L15, F30, E.

The program could be written with any text editing program, EDLIN or WordStar (in nondocument mode) for example, and then sent to the robot via its serial port. After the last character (the E for "end") in the program has been transmitted, the robot will beep, then wait for his forward sensor (the "nose" sensor) to be activated. Upon activation of the nose sensor, he will do the little "square dance." After the entire program has been executed and the robot is right back where he started, he will sit there and wait for the nose sensor to be activated again. Should the sensor be activated, the square dance will be repeated. The above sequence will continue until either the test button is depressed (executing the self-test procedure) or the on-board computer is reset via the reset button or by cycling the power.

Gears and Microprocessors

The first question apt to spring up about a robot is "What can it do?" Well, quite independently, BERT can beep, blink, talk, move forward and backward, turn left and right, and avoid obstacles. Let's examine the hardware requirements necessary to enable our robot to perform these tasks.

Beep and blink: Beeping for attention can be handled by a speaker tied to a single bit of a parallel port. Simply toggle the bit every millisecond and the speaker beeps at 1 kilohertz. Blinking, another method of communication, is nothing more than an LED connected to another bit of our port. Toggling that particular bit will cause the LED to blink.

Speech: All we require for this task is two chips. A 28-pin speech synthesizer chip, an 8-pin amplifier chip, and a couple of capacitors and resistors compose the entire circuit.

Forward, backward, left, and right: For these movements, the minimum requirement would be two motors attached to wheels, with some sort of feedback to tell the on-board computer how far the wheels have turned. The next require-

continued

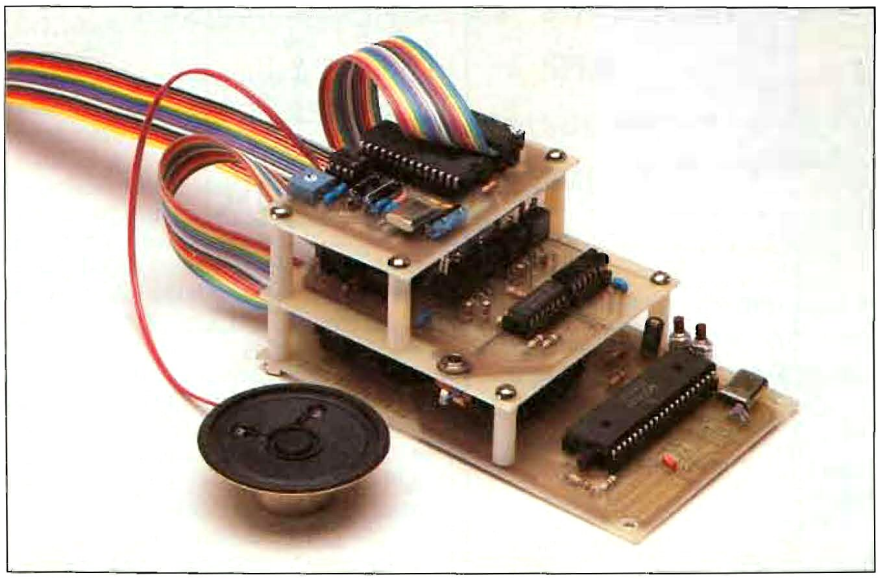


Photo 2: *BERT's circuit boards, from top to bottom: speech board, motor driver board, and the on-board computer, assembled and cabled together.*

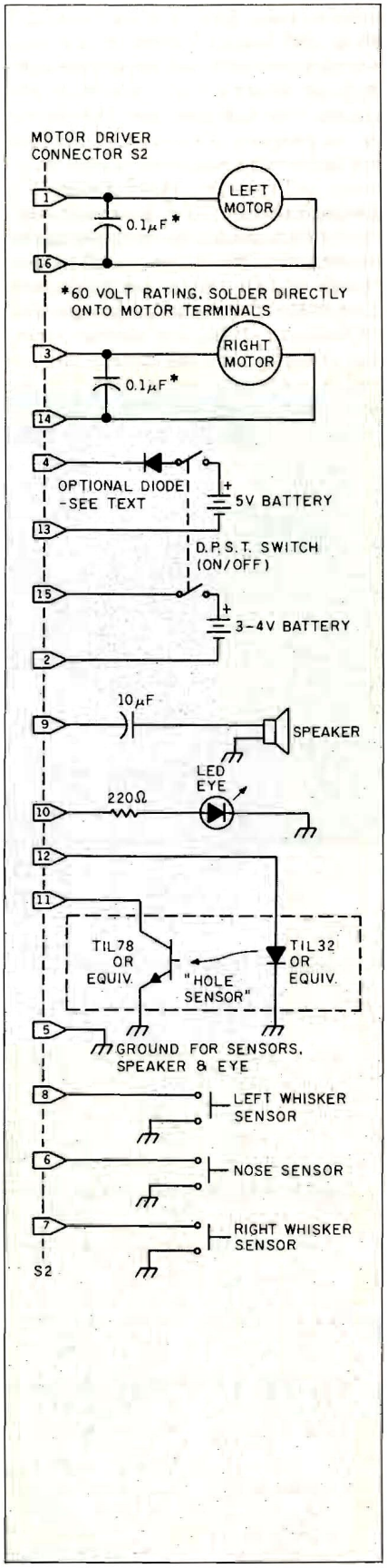


Figure 2: *Schematic diagram of the robot base.*

ment is some kind of motor control. I chose DC motors instead of the more common (in robotic circles) stepper type. Stepper motors rotate precisely the amount you tell them to. This ability allows designers to assume that they need not bother with positional feedback circuitry and software. This assumption is generally sound for disk drives and printers, where loads and environments can be maintained within design specifications. Hobby robots, however, are rarely allowed the luxury of a closely controlled environment. Hills, low batteries, rugs, and changing payloads cause a robot to stall or put widely varying strains on the drive train. If no feedback is obtained from the wheels, the controlling computer is completely oblivious to any positional inaccuracies that may be the result of the above load problems.

Using a DC motor, however, demands that some sort of feedback from the drive-train be used to control the motor's rotation. Thus, we have the benefits of feedback as well as the ultimate benefit for hobbyists: DC motors are cheaper.

Obstacle sensing: This can be tough or easy, depending upon how sophisticated we wish to be. The simplest (and least expensive) method would be to use a piece of wire and a switch. Not very glamorous, I admit, but certainly effective. A more elegant method would be to use an infrared proximity detector, a device that sends out a beam of infrared light and then looks for the reflection. When the reflection passes a certain threshold of brightness, the detector circuit outputs a signal, indicating an object nearby.

Both of these methods can be thought of as accomplishing the same task as a cat's whiskers (apparently, felines use their whiskers as feelers to determine clearances). That is to say, their output is a binary "go/no-go" signal. Therefore, I shall refer to them hereafter as the robots "whiskers."

Obstacle avoidance: Here we are asking our robot to demonstrate some form of intelligence. Data can be gained about the presence of an obstacle using the whiskers. To actually do something about the obstacle requires some form of deci-

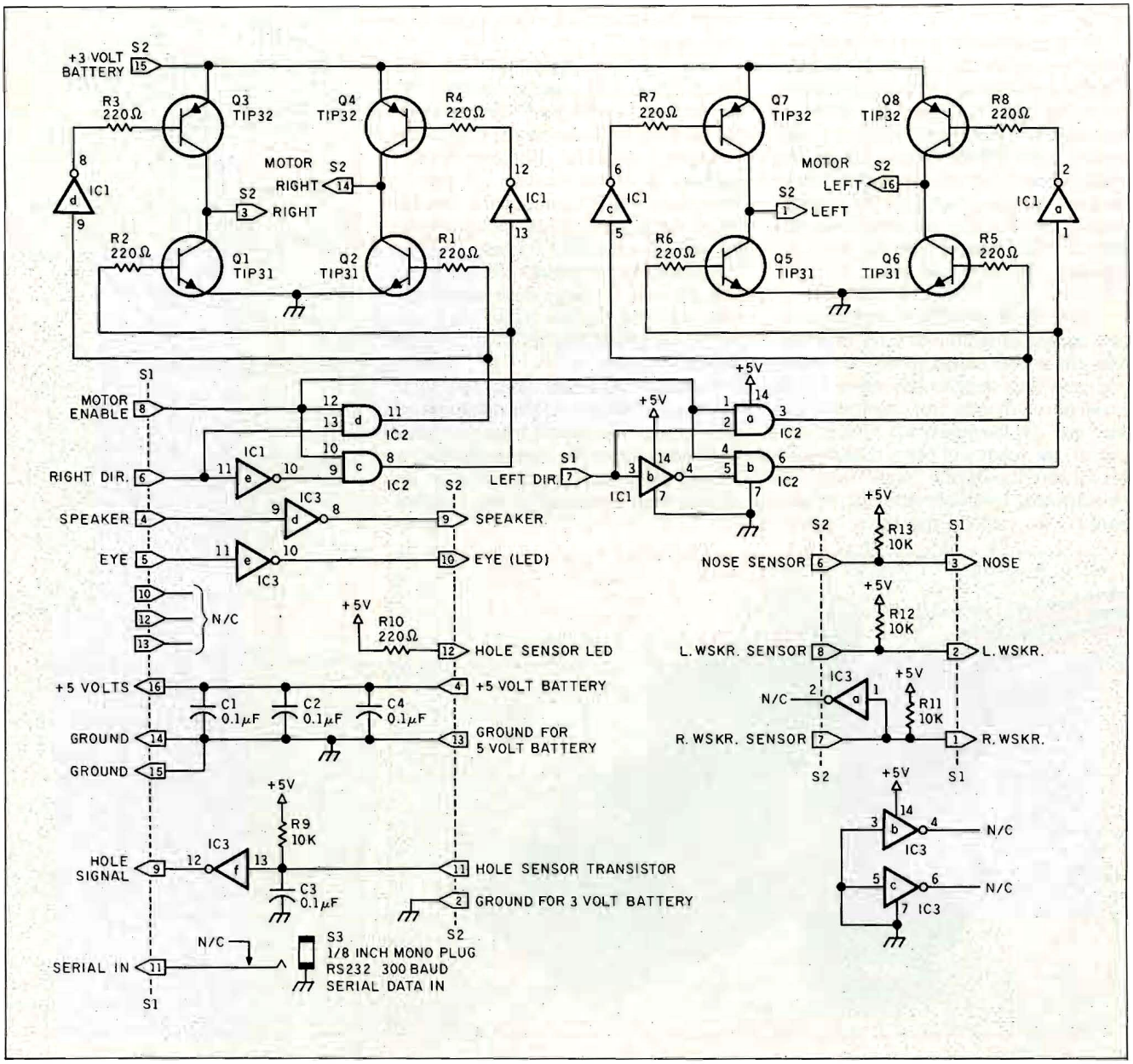


Figure 3: *Schematic diagram of BERT's motor control board.*

sion-making capability. An on-board computer (OBC) gives our robot this ability to make decisions based upon information gained by the robot's whiskers. BERT's OBC is a three-chip printed circuit board. The circuit design and microprocessor choice was influenced by two parameters: price and simplicity—in that order.

Power Supplies

How do we power our robot? If we were to use an AC power supply, the attendant power cord would severely restrict the mobility of the little beast. Batteries are the answer, but do we use rechargeable or disposable batteries? If rechargeable, shall we choose Gel-Cell or nickel-cadmium? How many hours of operation will we get before recharging is necessary? What type of battery charger should be used?

For the sake of simplicity and cost, I recommend that commonly available nickel-cadmium batteries and charger be used. BERT requires two power supplies (battery packs). This is necessary for two reasons. First, the OBC requires +5 volts (plus or minus 0.4 V). Most small DC motors require between 1.5 and 4 V. Second, small DC motors are electrically noisy. Connect a motor to a battery, then look across the battery's terminals with an oscilloscope. You will find spikes with an amplitude in the hundreds of volts. Obviously, a separate battery will be needed to power the on-board computer.

BERT's Specifications

BERT's brain consists of a 1-MHz Motorola 6802 microprocessor with 128 bytes of on-chip RAM, a 2K-byte 2716 ROM, and a 6821 PIA (peripheral interface adapter) used as two 10-bit parallel I/O ports.

BERT consumes 1 watt of DC power, and his motors consume up to 500 milliamperes. We download programs to BERT via an RS-232C, 300-baud serial interface. BERT uses LEDs for his eyes, and three switches for his nose and left and right sensors. He beeps from one small 8-ohm speaker and speaks from another, under control of an SP0256-AL2 speech synthesizer with a 64-word vocabulary.

BERT can test himself with a built-in self-test subroutine and can execute 15 different subroutine branch conditions. You can interface external devices to BERT through a 1-bit I/O port in the PIA. This can be used to initiate switch closure for activating external devices.

If you have ever tried to write machine language, feedback-driven motor control routines, you will be glad to know that our OBC's software is already written in 6800 machine code. It is supplied in the kit as a preprogrammed EPROM.

Putting the Pieces Together

In the following discussion, the schematic diagrams of the OBC (figure 1), robot base (figure 2), motor control board (figure 3), and speech board (figure 4) should be consulted for reference. (See also photo 2.)

Assembly/parts location drawings of these boards (figures 5, 6, 7, and 8) show the location of all components, and the parts lists give a complete description of part identification and values. These should be followed closely when you assemble and solder the printed circuit boards. In part 2 of this article, we'll interconnect the printed circuit boards and test them.

The Robot Base

The design of BERT's base is fairly straightforward. The main requirements are that the driving wheels be somewhere near the base's center of gravity, and each driving wheel must be powered by

continued

BERT can run for 4 to 6 hours without a recharge.

its own DC motor. You can build your own base from scratch, buying some type of motorized toy and modifying it, or purchase the preassembled base from Amarobot. I strongly suggest the latter.

BERT uses an optical "switch" to sense how many rotations the wheels have made. Looking at the schematic diagram of the robot base (see figure 2), you can see an infrared LED and its companion phototransistor labeled "hole sensor." These constitute the optical switch. If we were to place the LED on one side of a solid wheel and the transistor on the other, then drill a hole through the wheel, when the hole came around to the LED's position, light from the LED would fall upon the transistor. Since light from the LED causes the transistor to turn on, it would output a logic-low signal.

In the above scenario, with one hole drilled through the wheel, each revolution would equal one "unit." If you choose to build your own base or modify an existing vehicle, you must arrange the infrared LED and phototransistor in such a position as to give the OBC some indication of how many units it has gone.

It is not absolutely necessary to drill the holes in the wheel itself. Since we are using small DC motors (which are typically high-speed, low-torque devices), there will probably be some sort of gear train involved. You could drill the holes in any convenient gear of the train. But I suggest that the gear you choose be near the end of the train (i.e., nearer to the wheel than the motor); otherwise the gear backlash could affect the accuracy of positional feedback.

The number of holes used will depend on the circumference of the driving wheels. A good point to aim for would be about one hole per centimeter of wheel circumference. For example, if your wheel were 9.6 cm in diameter, giving a circumference of 30 cm, then you would need 30 holes in the wheel. If, for the same size wheels, you were to choose to drill the holes in a gear instead, you would have to base the number of holes on the gear ratio relative to the wheels' circumference.

If you feel that building gearboxes, calculating gear ratios, drilling holes, and fabricating a bracket for the hole sensor is too much like work, you can purchase a gearbox with two motors, gears, and sensors completely assembled from Amarobot. Due to the limited drive current of the motor driver board, I recommend that the weight of the bare motorized base (less batteries, speakers, sensors, and electronics) be less than 4 kilograms. This will allow for approximately 2 kilograms of batteries and still leave us some payload capability (enough, say, for a small cat).

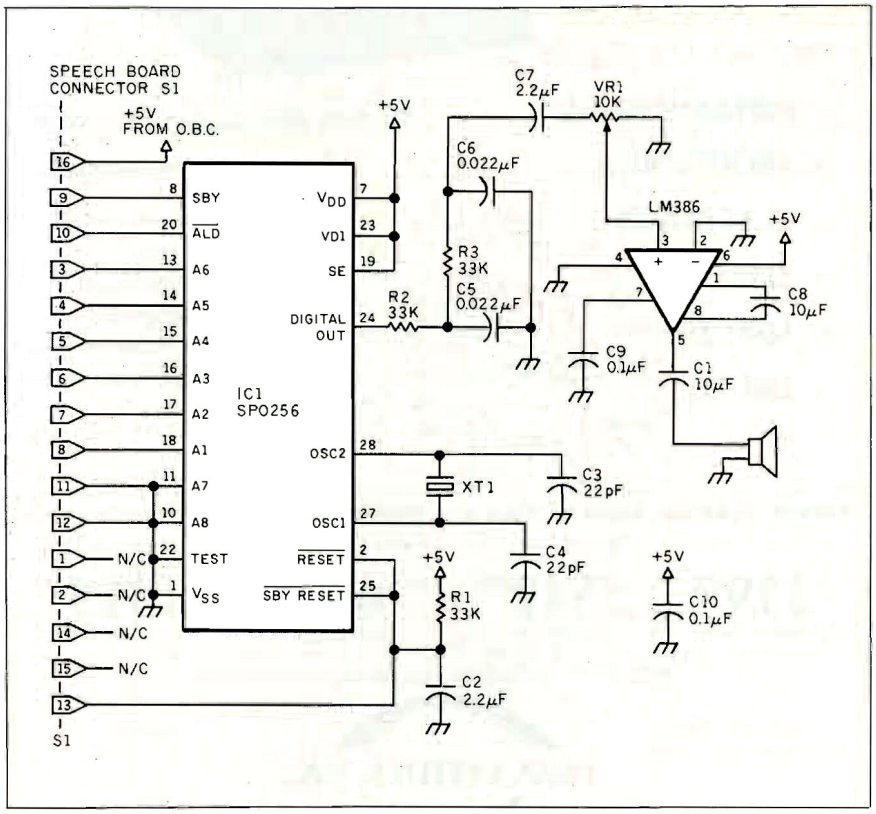


Figure 4: *Schematic diagram of BERT's speech board.*

Base Population

The permanent inhabitants of the platform are shown in the robot base schematic (figure 2). These devices mount directly onto the base and connect to the electronics via one DIP connector (motor driver connector S2).

The sensors (nose, left, and right) can be microswitches attached to a bumper of sorts. My favorite "sensor" is an old keyboard switch with a springy loop of wire glued onto it as a bumper. This combination is inexpensive and serviceable.

As for the batteries, I recommend nickel-cadmium because of their cell voltage. A fully charged cell is approximately 1.25 V. If you use four of them in series, you have $4 * 1.25 = 5$ V—just perfect for the OBC.

(If, however, you decide to use disposable batteries with a voltage of approximately 1.5 V per cell, you would have to deal with 6 V. To get this down to a manageable voltage, a diode of .7 V in series with the batteries would lower the voltage to approximately 5.3, within the OBC's range. The placement of this diode is shown in figure 2 and labeled "optional diode.")

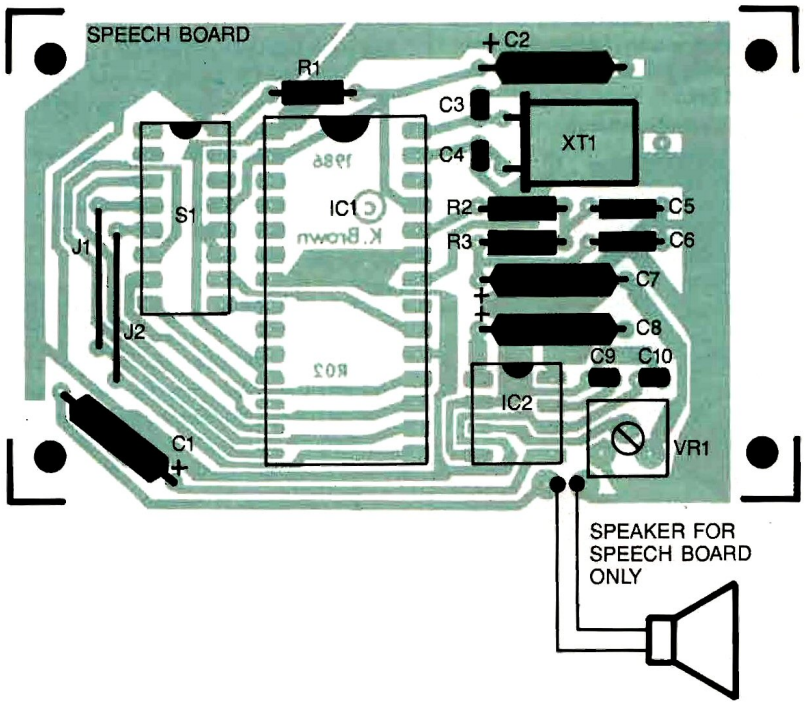
The OBC uses up about 200 milliamperes. The motors use a widely varying amount of current depending upon many conditions. I recommend using four "D" cells (4 amperes per hour per cell) for the OBC and three "C" cells (1.8 amperes per hour per cell) for the motors. This combination gives a fair power per weight ratio with approximately 4 to 6 hours of operation per charge.

One last thing about batteries. They can be dangerous. For safety's sake, install the on/off switch in an accessible area. Unfortunately, small batteries, unlike small power supplies, are capable of lots of current for a short period. I have helped many people build their own robot and have seen more than one small wiring accident turn into a full-blown fire. Even though none are shown in the diagram, you might actually consider using a fuse or two in the higher-current areas.

The Motor Control Board

When the OBC wishes to turn a motor on, it makes a pin on the PIA go high. When high, this pin is at approximately +4 V. This signal is used to drive two

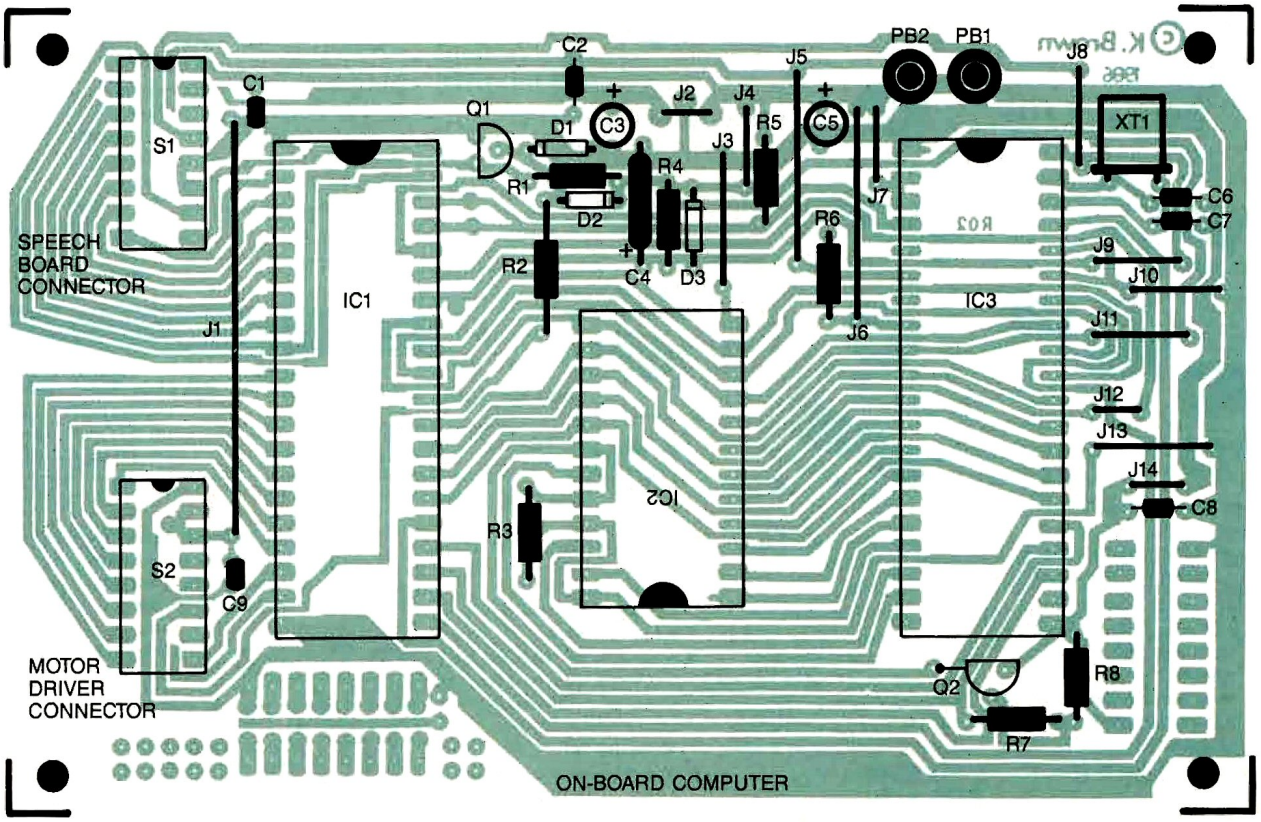
continued

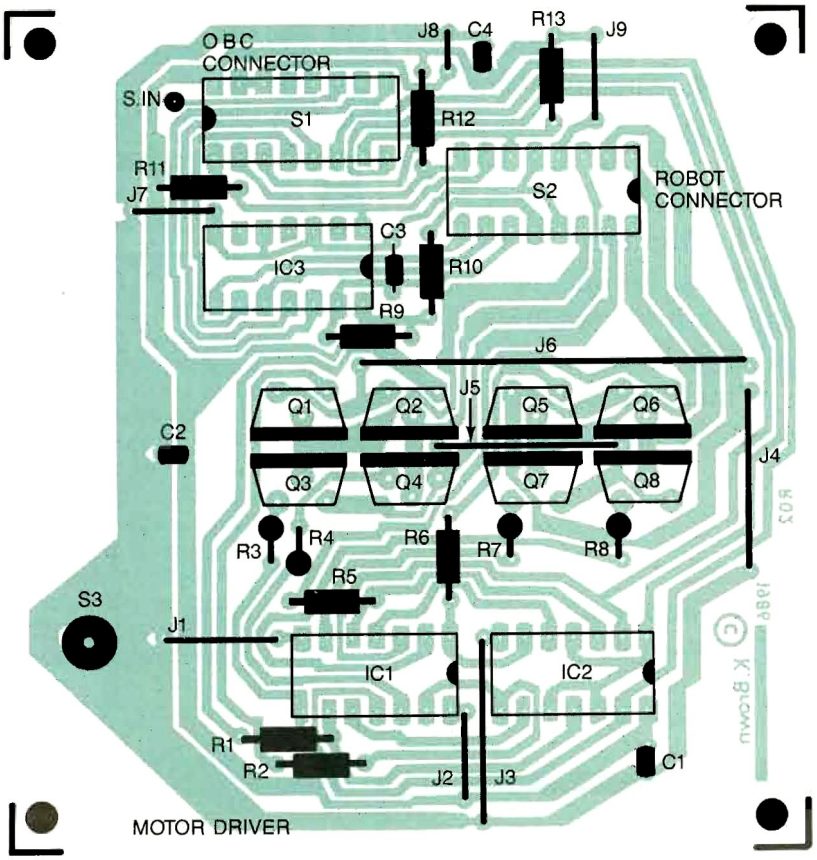


Speech Board Parts List

R1, R2, R3	33K ¼W
VR1, potentiometer, 1 turn 10K	
IC1, voice synthesizer, SP0256A-AL2 (General Instrument)	
IC2, audio amplifier, LM386 (National Semiconductor)	
C1, C8	10 μF, 12 V
C2, C7	2.2 μF, 12 V
C3, C4	22 pF, 12 V
C5, C6	.022 μF, 12 V
C9, C10	0.1 μF, 12 V
X1, crystal, any value from 3.12 MHz to 3.579545 MHz.	
S1, socket, 16-pin DIP	
Socket, 28-pin DIP (synthesizer chip)	
Socket, 8-pin DIP (audio amplifier)	
Small speaker, 8-ohm impedance	

Figure 5: *Speech board assembly and parts location drawing, and parts list.*





Motor Driver Board Parts List

R1 through R8,R10	220 ohm ¼W
R9, R11, R12, R13	10K ¼W
IC1	74LS04
IC2	74LS08
IC3	74LS14
C1 through C4	0.1 μF, 12 V
Q1, Q2, Q5, Q6, transistor, TIP31	
Q3, Q4, Q7, Q8, transistor, TIP32	
S1, S2	16-pin DIP socket
S3	⅛-inch phone jack
Cable, ribbon 16-pin DIP	
Socket, 14-pin DIP (for IC1, IC2, and IC3)	

Figure 7: *Motor driver board assembly and parts location drawing, and parts list.*

On-board Computer Parts List

R1, R5, R6	10K ¼W
R2, R3, R7	1K ¼W
R4	33K ¼W
R8	510 ohm ¼W
IC1	6821 PIA
IC2	2716 EPROM
IC3	6802 microprocessor
C1, C2, C8, C9	0.1 μF, 12 V
C3, C5 (radial leads)	4.7 μF, 12 V
C4 (axial leads)	2.2 μF, 12 V
C6, C7	27 pF, 12 V
Q1, Q2	transistor, 2N5135
D1, D2, D3	diode, 1N4148
XT1	crystal, 4 MHz
S1, S2	socket, 16-pin DIP
PB1, PB2	switch, SPST, momentary, push-button
Socket 40-pin DIP	(microprocessor)
Socket 40-pin DIP	(PIA)
Socket 24-pin DIP	(EPROM)

Figure 6: *On-board computer assembly and parts location drawing, and parts list.*

Circuit Board Assembly Instructions

1. Solder in all capacitors.
2. Solder in all jumpers.
3. Solder in all sockets. The OBC connector socket is installed backward (pin 1 facing backward relative to the other sockets).
4. The ⅛-inch phone jack, which is the robot's serial data connector, is connected to ground via its screw collar. Connect its TIP connector to the solder pad labeled "S.IN" near connector S1.
5. Solder in all resistors.
6. Solder in all transistors.

Caution! The metal heat-sink tabs on the transistors are connected to the collector lead. Do not let the transistors touch each other or any other circuit component.

7. Install all integrated circuits, observing precautions to protect chips from static electricity.

pairs of current amplifier transistors. The DC motor may require 50 milliamperes to start turning with no load. If the motor is loaded down, we might be talking about currents of half an amp or so.

Four transistors are used to control each motor. These are arranged in a common "H" or "bridge" configuration. The transistors used in this circuit are rated far in excess of the current loads we will be putting on them. This allows us to simplify the circuit, cut down on weight, and save money by doing away with a heat-sink.

Referring to the motor control board schematic (figure 3), you can see that we are using TTL logic to provide base current for the transistors. This really cuts down on the parts count, which makes the circuit very easy to build and troubleshoot. Admittedly, this design does push the drive capability of the gates a bit. However, hundreds of people have used this circuit and I've yet to hear of a failed gate.

Since BERT is such a small robot, he really needs some method of attracting attention. He can either flash an LED or beep a speaker. While the PIA has enough current capability to do both, it would be unwise to risk using its outputs

continued

Since BERT is so small, he needs some method of attracting attention. He can either flash an LED or beep a speaker.

while there are gates to spare on IC3 (the 74LS14). Utilizing some of these unused gates affords some protection for our expensive PIA.

The base-mounted speaker must have one end tied to ground, with its free end tied to a 10-microfarad capacitor. The free end of the capacitor must be tied to pin 9 of the motor driver board's connector S2. The LED used for the robot's eye must have its cathode tied to ground, with the anode connected to a 220-ohm resistor. The resistor's free end must then go to pin 10 of the motor driver board's connector S2 (see figure 3).

Sensors

BERT's three sensors—nose, left whisker, and right whisker— can be simple push-button switches. When the switch is depressed by striking an object, it should make a pin on the PIA go low. In other words, under normal conditions, the PIA's sensor-connected pins should be high. This is accomplished by tying one end of three 10K resistors (one for each of the sensors) to +5 V, and the other end to its sensor's output. When the switch is closed, it will pull one end of its 10K resistor to ground. These three resistors (R11, R12, and R13) are located on the motor control board.

Next month, in part 2, we'll test each of BERT's circuit boards and base wiring and get down to programming his onboard computer. ■

The following parts are available from Amarobot, 2913 Ohio St., Richmond, CA 94804, (415) 451-6780.

Complete set of printed circuit boards, drilled, with component overlay and solder mask $24.85

BERTL EPROM $17.95

Gearbox $9.95

Wheels..... $2.95

Base, dome, gearbox, two bumpers with switches, and wheels $49.95

California residents add 6.05% sales tax.

Karl Brown

Build BERT, the Basic Educational Robot Trainer, Part 2

This month we show how to troubleshoot and program your robot

Last month I introduced you to BERT, a simple, programmable robot made from inexpensive parts (see photo 1). I described how his circuits worked and how to build his circuit boards. Now I'll discuss how to troubleshoot and program your assembled BERT.

After the four main components—base, motor driver board, on-board computer, and speech board—have been built and connected (see photo 2 in part 1), you should have a working BERT. Well, he *should* work. To see if he is fully functional, put him on the floor and press the self-test button. He should perform the routine shown in table 1. If he doesn't perform as expected, you must determine the problem by first isolating it to one of the main components, then referring to that component's schematic and its troubleshooting section.

Motor Check

Testing of the motor driver circuitry is accomplished with some wires and a resistor. These will temporarily substitute for the on-board computer (OBC). First, secure the board to the robot base. Make sure all electrical connections to the motors, both batteries, speaker, sensors, and LED are secure. After this is done, we are ready to begin testing.

Caution! Do not stuff any wire into the connector socket while performing these tests. Doing so will cause the socket's

continued

Karl Brown teaches electronics at Vancouver Community College. His hobbies include computer hardware design and juggling. He can be reached at Vancouver Community College, Electronics Department, 250 West Pender St., Vancouver, British Columbia, Canada V6B 1S9.

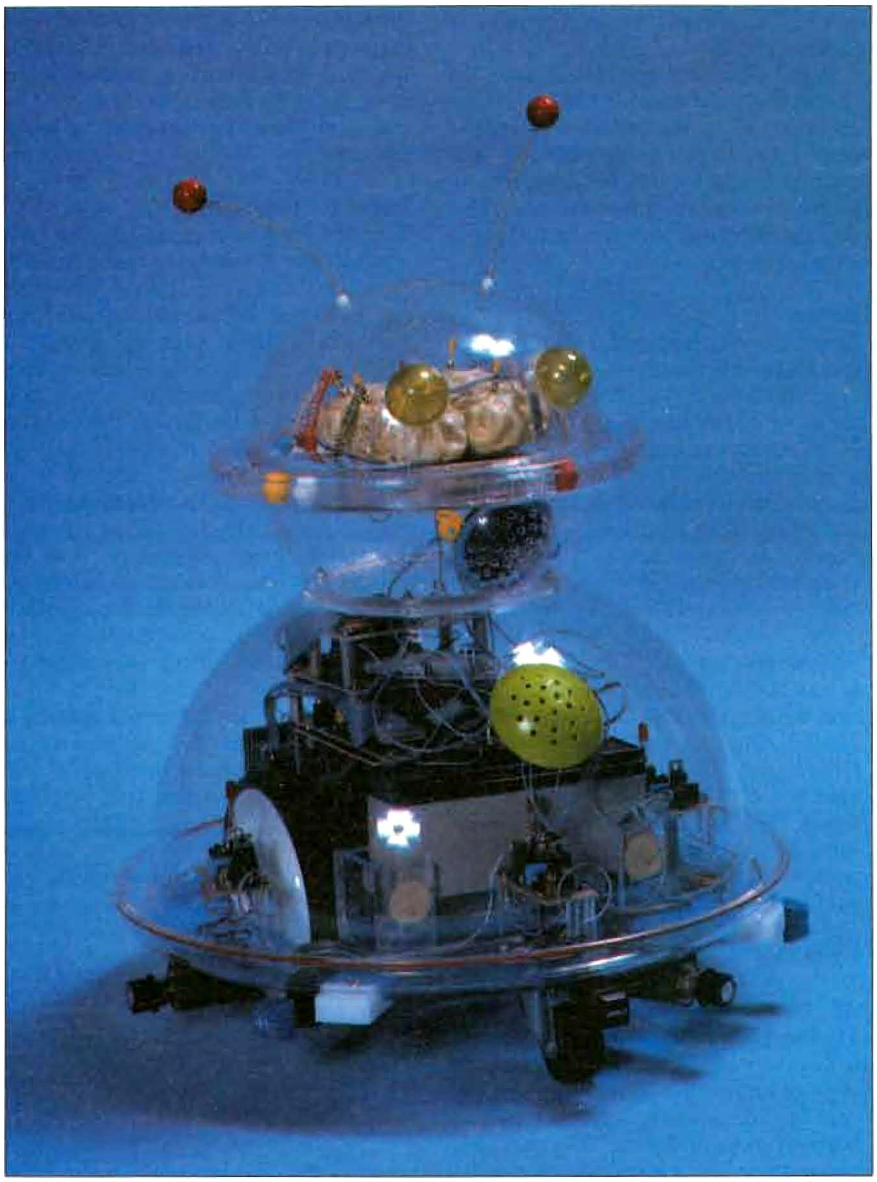


Photo 1: *BERT has "skin" constructed of acrylic salad bowls.*

connector pin spacing to enlarge, making the connector's pins unable to correctly grip the DIP connector.

To check for motor operation, you will need three pieces of wire (approximately 10 centimeters long) and a 510-ohm resistor. Solder one end of the resistor to +5 volts. Then, strip 1 millimeter of insulation off each end of each wire. Now solder one end of each wire to the following points on the solder side of the motor driver board: wire 1 to IC2, pin 12; wire 2 to IC1, pin 11; and wire 3 to IC1, pin 3.

With the above connections made and power from both batteries being fed to the board, temporarily connect the free ends of wires 2 and 3 to ground. Then, connect the free end of wire 1 to the free end of the 510-ohm resistor. Both motors should spin in the reverse direction. If either one turns forward instead of backward, just reverse the connections to that motor's terminals.

When this test is completed, leave the connection between wire 1 and the resistor intact. Now, connect the free ends of wires 2 and 3 to the resistor as well. Both motors should spin in the direction required to move the robot forward. This concludes the motor driver circuitry test. Remove all three wires.

Beeper Test

To test the beeper speaker, simply solder one of the aforementioned wires to the free end of the 510-ohm resistor. Solder the other end of that wire to IC3, pin 9. Now, solder one end of another piece of wire to ground. With your ear right up against the beeper speaker, brush the free end of that wire against IC3, pin 9. You should hear some noise. When you've completed your testing of this circuit, remove the resistor as well as all wires.

Sensor Testing

Sensor testing requires a voltmeter. Connect the negative end of the voltmeter to ground. Connect the other end to the OBC socket (S1), pin 7. You should measure +5 V. Depress the right whisker sensor. You should measure 0.0 V. Perform the same test for the nose and left whisker sensors on pins 6 and 8, respectively, of the same connector (S1).

LED Test

Testing the LED (eye) circuitry is simple. All you do is temporarily ground IC3, pin 11. The LED should illuminate. The circuits involving the sensors, speaker, and eye are very simple. If any of these circuits fail to check out, first check the connections to the batteries, then their voltage level. After that, merely start at the beginning of the circuit (where the test wires are attached) and follow the levels through the circuit until the break is found.

The motor driver circuits are a bit more complex, but not much. There are actually two circuits, one for each motor. Remember to troubleshoot only one at a time.

Again, check the power connections first. Next, check the inputs to the gates. Find out what logic levels are on the inputs; then, using the truth table for AND gates, figure out what levels should be on the outputs of the gates. With this information, and following the outputs to the bases of all four transistors, write down on the schematic what level should be on the base of each transistor.

At this point you will need a voltmeter. Attach its negative lead to ground. Check the TIP31's emitters. They must measure 0.0 V. Each base that should be high will measure approximately 0.7 V. Each base that should be low will measure less than 0.3 V. If they do not, trace the connections from the base back to its driving gate and check the value of its series resistor.

Some Helpful Hints

When BERT is powered up, the OBC is in a reset state. A power-on reset lasts only $\frac{1}{10}$ second. If the reset button is pressed, the reset state will last as long as the button is held down, plus $\frac{1}{10}$ second. In either case, as long as the reset state lasts, the motor driver board will cause both motors to turn in the forward direction. In fact, if the motor driver board does not receive signals from the OBC (as is the case during the OBC's reset state), it will always cause both motors to turn in the forward direction. Therefore, if at this stage BERT goes forward and keeps going forward, it is likely that the problem lies with the OBC or the connections between the motor driver board and the OBC.

The beep produced by the speaker is the result of the 6802 microprocessor causing the peripheral interface adapter to toggle one of its outputs (pin 14). If the beep is not heard, use a logic probe or oscilloscope to check the PIA's pin for a ¼-second train of pulses. If the train is there, follow the signal along its path to the speaker to find where the circuit is broken. If the pulses are not found at the PIA's pin, there is either a problem on the motor driver board that is holding the pin at a constant level (short to ground or +5 V) or you have a problem with the OBC and must troubleshoot it.

Table 1: *BERT's self-test routine.*

Move forward,
Turn eyes on,
Speak the word "cookie,"
Turn left,
Turn right,
Left to center,
Beep once,
Move backward (to original position),
Beep once,
Turn eyes off,
Perform system initialization (reset).

Feedback Problems

When the test button is pressed, pin 6 of the 6802 should go low and the OBC should enter its nonmaskable-interrupt routine. The first thing this routine does is output signals to the motor driver board that cause the motors to spin in a forward direction. These signals are output from the PIA's pins 10, 11, and 12.

Immediately after the test button is pressed, these pins should go high and remain in that state until the robot has gone a distance equal to 50 (decimal) holes from the hole sensor. The actual distance traveled will depend on the size of the wheels. Should these pins not respond as above, there is either a problem on the motor driver board that is holding the pins at a constant level (short to ground or +5 V) or you have a problem with the OBC and must troubleshoot it.

If the signals at these pins do transition, but only briefly, here is the most probable reason: BERT uses feedback from the drivetrain to determine how far he has gone. If he does not have this feedback, he has no way of knowing the distance he has traveled. This feedback comes from the hole-sensor circuit. If BERT fails to stop, look at the output of this circuit.

BERT counts these pulses at PIA pin 18. This input pin is a negative-going, edge-sensitive input. In the self-test mode, BERT will turn the motors on in the forward direction and leave them on until he has counted 50 negative edges (or holes) at PIA pin 18. Should the output of the hole-sensor circuit be faulty, or not arrive at the PIA, BERT will simply keep going.

Noise

There is a very subtle problem which can arise with BERT that relates to the hole-sensor circuit described above. Small DC motors are electrically noisy. This noise manifests itself as large (up to 100 V in amplitude) and narrow (100 nanoseconds to 20 microseconds) voltage transients, or spikes.

These spikes may be large enough to propagate through not only the power leads (from battery to boards), but also

continued

Programming in BERTL

BERTL is BERT's control language. It is an interpretive language that does syntax checking at program entry time. Table A lists the BERTL commands and their functions.

Each of the commands (except E) requires a parameter. If you command BERT to go forward, for example, he needs to know how far. The entire command would be Fnn, where nn can be any hexadecimal number from 01 to FF (FF hexadecimal equals 255 decimal).

BERTL commands can be downloaded from any serial device with the following characteristics: 300 bps, 7 data bits, 2 stop bits, and no parity. One way to enter BERTL programs into your robot is to write a short program that receives characters from your computer's keyboard and then sends them via the serial port to the robot.

The only drawback to this method is that it is impossible to store the program for future use or to change the robot's program without reentering it entirely. I suggest that the program be edited on a computer using a word processor or text editor and then sent to the robot. This allows you to conveniently edit and store many different BERTL programs.

The Commands in Detail

Here is a description of the BERTL commands:

• Bnn (Backward): This command causes BERT to reverse both wheels. They will continue in reverse until the number of pulses from the feedback circuit equals the parameter. In BERT's case, a value of 32 (hexadecimal) equals approximately 32 centimeters.

• Ccn (Call subroutine): This command causes the conditional execution of a subroutine, depending upon the condition of the on-board sensors. The parameter determines both the condition upon which the subroutine is called and which particular subroutine (out of a possible four) is to be called.

The first digit in the parameter is the condition. If it is 0, the call is unconditional and will be done regardless of the status of the on-board sensors. If it is a hexadecimal number other than 0, the condition of the on-board sensors will be tested to see if they match c. The conditions specified by c are shown in table B. (Note the logical conditions AND and OR.)

The second digit in the parameter specifies which subroutine will be called. The subroutines are 0, 1, 2, and 3 (see the command N). For the command B12 C00, the robot would go backward 12 cm, then call subroutine 0 regardless of the state of the on-board sensors. For the command F09 C51, the robot will first go forward 9 cm and then, if the nose button is low and both whiskers high, call subroutine 1.

• E (End): This command has no parameter. Upon receiving it, the robot "honks" to indicate that it has received the downloaded program. It waits for you to activate its nose sensor, after which it executes the program. It then waits for its nose sensor to be activated again. This procedure can be repeated indefinitely.

• Fnn (Forward): Turn both motors on in the forward direction. Otherwise, this

continued

Table A: *The BERTL command set.*

Command	Function
B =	Backward
C =	Call (subroutine, conditional or unconditional)
E =	End program
F =	Forward
G =	Goback (to main routine, unconditional)
H =	Honk
I =	Eye
L =	Left
N =	Name (of subroutine)
O =	Output
P =	Pause
R =	Right
S =	Shoot
T =	Talk (allophone or word)

Table B: *Conditions for sensor parameter* c. *The symbol "&" represents the logical AND, and "+" is the logical OR condition.*

Value of c	Logical condition of sensors
0	Don't care (disregard state of sensors)
1	Left low & Nose low & Right high
2	Left low & Nose high & Right low
3	Left low & Nose high & Right high
4	Left high & Nose low & Right low
5	Left high & Nose low & Right high
6	Left high & Nose high & Right low
7	Left high & Nose high & Right high
8	Left low & Nose low & Right low
9	Left low + Nose low + Right low
A	Left low + Nose low + Right high
B	Left low + Nose high + Right high
C	Left high + Nose low + Right low
D	Left high + Nose low + Right high
E	Left high + Nose high + Right low
F	Left high + Nose high + Right high

Table C: *Allophones used by the SPO256 chip on the speech board.*

00	Delay	10 ms	10	mm	milk	20	aw	out	30	wh	when
01	Delay	30 ms	11	tt1	part	21	dd2	do	31	yy1	yes
02	Delay	50 ms	12	dh1	they	22	gg3	wig	32	ch	church
03	Delay	100 ms	13	iy	see	23	vv	vest	33	er1	letter
04	Delay	200 ms	14	ey	beige	24	gg1	guest	34	er2	bird
05	oy	boy	15	dd1	could	25	sh	ship	35	ow	sew
06	i	sky	16	uw1	to	26	zh	azure	36	dh2	they
07	eh	end	17	ao	aught	27	rr2	brain	37	ss	vest
08	kk3	comb	18	aa	hot	28	ff	food	38	nn2	no
09	pp	pal	19	yy2	yolk	29	kk2	sky	39	hh2	hoe
0A	jh	dodge	1A	ae	hat	2A	kk1	can't	3A	or	store
0B	nn1	thin	1B	hh1	he	2B	zz	zoo	3B	ar	alarm
0C	ih	sit	1C	bb1	rib	2C	ng	anchor	3C	yr	clear
0D	tt2	two	1D	th	thin	2D	ll	lake	3D	gg2	got
0E	rr1	rural	1E	uh	book	2E	ww	wool	3E	el	saddle
0F	ax	succeed	1F	uw2	food	2F	xr	repair	3F	bb2	business

command is identical to B.

• G00 (Goback): Go back to main routine. This command must be placed at the end of each subroutine. Its parameter is always 00.

• Hnn (Honk): Cause the robot's speaker to honk (beep) *nn* times, where *nn* is a hexadecimal number from 00 to FF. Note that all other robot activity will cease during the honk. The honk duration is 0.52 second.

• Inn (Eye): Turn the robot's lights on or off. If the parameter is 00, the lights will turn off. Any other value turns the lights on.

• Lnn (Left): This command causes the robot to pivot on its axis (the imaginary point between the drive wheels) in a counterclockwise (when viewed from above) direction. This is done by moving the left wheel in a forward direction while moving the right wheel in reverse. This motion continues until the feedback circuit outputs enough pulses to equal the parameter.

• N00, N01, N02, N03 (Name of subroutine): BERTL allows four subroutines. Each may be called unconditionally or only when the on-board sensors match a specific condition (see the command C). The format of a subroutine is as follows: name, commands, goback. For the subroutine N00 F60 H02 B60 G00, the robot will go forward 60 cm, honk twice, go backward 60 cm, and then return to the main program.

• Onn (Output): If the parameter is 00, the output of the 1-bit OBC port will switch to a logic low. If the parameter is any other value, the output level of the port will be a logic high. The port will support two TTL logic loads and therefore may need buffering if it is used to switch higher-current loads.

• Pnn (Pause): This command causes the robot to pause (do nothing) for the decimal equivalent of the number of seconds specified in hexadecimal in the parameter. For example, the command P3C would cause the robot to pause for 60 seconds.

• Rnn (Right): Turn right (or, to be more specific, cause the robot to pivot in a clockwise direction). Otherwise, the command is identical to Lnn.

• Snn (Shoot): This is a light and sound effect. It causes the LED to light up and the speaker to produce a "phaser" sound. Otherwise, it is the same as Hnn.

• Tnn (Talk): This command is used to control the optional speech board. If the speech board is not used on the robot, the OBC's CA2 output must be connected to the CA1 line. If this is not done, the robot will "hang" whenever this command is used.

BERT's speech chip is the General Instrument SPO256-AL2. This chip has specific sounds (allophones) stored in it. When strung together, these sounds simulate human speech. The parameter for this command is any hexadecimal number from 00 to 7B.

There are a total of 63 allophones and 63 words available to the BERTL programmer. Note that the speech chip continues to make the last portion of the last allophone it was commanded to emit. You can clear its throat by using 00 (silence) as the last sound emitted.

The command string T09 T2D T13 T2B T00 uses allophones to make the robot say the word "please." This can also be accomplished with the command T67, a command that uses BERT's preprogrammed 63-word vocabulary. Each word is represented by a number in the range of 40 to 7B hexadecimal. (See table C for the allophones and table D for BERT's vocabulary.)

Syntax Checking

When BERT receives commands via his serial port, he checks each one for syntax and validity. If bad syntax or an invalid command is detected, BERT will honk and light up his eye. If this happens, reset the robot and check your program for errors (a favorite error is typing o instead of 0).

Table D: *BERT's preprogrammed vocabulary. Each word is automatically terminated by the allophone "PA5" (04).*

40	ZERO	4F	CAN	5E	LIKE	6D	START
41	ONE	50	CHECK	5F	LITTLE	6E	STOP
42	TWO	51	COMPUTER	60	ME	6F	SORRY
43	THREE	52	COOKIE	61	MEMORY	70	SWITCH
44	FOUR	53	CORRECT	62	MOMMY	71	TALK
45	FIVE	54	DATE	63	NO	72	TEA
46	SIX	55	DO	64	NOT	73	THANK
47	SEVEN	56	EDUCATIONAL	65	OFF	74	TRAINER
48	EIGHT	57	EMERGENCY	66	ON	75	YES
49	NINE	58	ENTER	67	PLEASE	76	YOU (U)
4A	ALARM	59	ERROR	68	READY	77	ALERT
4B	ARE	5A	ESCAPE	69	ROBOT	78	LEFT
4C	BASIC	5B	EYE (I)	6A	ROBOTS	79	RIGHT
4D	BE	5C	HELLO	6B	SEE (C)	7A	WANT
4E	BERT	5D	IS	6C	SPEAK	7B	SILENCE

The BERTL language is also tolerant of the following characters: space, carriage return, and linefeed. While an error beep will not result from the preceding ASCII characters, nothing else will happen either. However, these characters must not come between the hexadecimal digits. For example, F44 L22H01 B 17 E would be acceptable, whereas, F44 L22H01 B1 7 E would not. (In this example, the robot would go forward, turn left, honk once, and go backward.)

I recommend that you make judicious use of subroutines to reduce your program's memory requirements. If something has to be done two or more times, use a subroutine. But bear in mind that no subroutine can call another subroutine. This is impossible due to the severely limited stack space of the OBC.

Should you decide to use a word processing program to enter, edit, and transmit a BERTL program, be careful. A lot of word processors embed nonprinting characters (ASCII codes above 127 hexadecimal) in the text for their own purposes. If you are using WordStar, for example, always create and edit BERTL programs using the nondocument command. When "printing" your program to the robot, suppress page formatting.

The program shown in listing A will cause BERT to beep and say "hello." Following that, he will jog forward, stop and check his sensors, and then take appropriate action if any one of the sensors is activated (low). If no sensor is activated (by bumping into some object), BERT will again go forward, stop, and check his sensors. If no sensors are activated throughout his journey, BERT will jog forward a total of four times, then finally stop and wait for either his nose or reset button to be pressed.

Listing A: *A sample BERTL program. Note: Do not try to enter the comments into BERT.*

```
H01 T5C     ; HONK ONCE & SAY "HELLO".
F0F         ; GO FORWARD 15 UNITS, THEN
C30         ; IF SENSOR CONDITION "3" IS MET,
            ; CALL SUBROUTINE NAME0.
C51         ; IF SENSOR CONDITION "5" IS MET,
            ; CALL SUBROUTINE NAME "1".
C62         ; IF SENSOR CONDITION "6" IS MET,
            ; CALL SUBROUTINE NAME "2".
F0F C30 C51 C62         ; DO THE SAME THING AGAIN
F0F C30 C51 C62         ; AND AGAIN
F0F C30 C51 C62         ; AND AGAIN
N00         ; SUBROUTINE NAME "0" STARTS HERE.
T79         ; SAY "RIGHT", THEN
R09 G00     ; TURN RIGHT AND GOBACK TO MAIN
            ; PROGRAM.
N01         ; SUBROUTINE NAME "1" STARTS HERE.
T4A         ; SAY "ALARM", THEN
B10 R06 G00       ; BACKUP 16 UNITS, TURN RIGHT AND
            ; GOBACK TO MAIN PROGRAM.
02          ; SUBROUTINE NAME "2" STARTS HERE.
T78         ; SAY "LEFT", THEN
L06 G00       ; TURN LEFT AND GOBACK TO MAIN
            ; PROGRAM.
E           ; END OF PROGRAM.
```

The appropriate action to be taken if a sensor is low depends upon which sensor is low. If the nose sensor is activated (by bumping into a wall, for example), then the subroutine N01 will be called. This subroutine will cause BERT to say "sorry," then back up, turn to the right, and return to the main program.

through the air, as radio frequency energy. This RF interference can be picked up by any antenna in the vicinity, even the piece of wire connecting the hole sensor to the PIA. The PIA interprets these spikes as hole transitions and soon counts enough of them to satisfy the command parameter. BERT thinks he has traveled the required distance when, in reality, he has moved a short distance, or not at all.

If this noise causes problems for your robot, you might consider using shielded cable from the hole-sensor circuit to the connector on the motor driver board. If the problem persists, try increasing the size of filter capacitor C3 on the motor driver board.

If the LED does not illuminate at this point, check for pin 13 of the PIA going low. If it does not, your problem is likely the OBC. If it does go low and the LED refuses to illuminate, follow the signal from the PIA to the LED to find the circuit break.

If BERT has operated well up to this point, but has failed to turn, it is not likely that the problem is with the OBC. The guilty party is probably the motor driver board. To prove this, check the PIA's pin 12 at the precise time in the self-test when BERT is supposed to be turning left. Pin 12 should be low. Follow this signal through to the motor driver board. If it arrives there, you have isolated the trouble to the motor driver board.

If BERT fails to turn right, the PIA pin to look for is pin 11. It should be low. To cause the motors to spin in a reverse direction, PIA pins 11 and 12 should be low, while pin 10 is high. If this is the case, but the motors do not reverse, follow these signals to the motor driver board. If they arrive there, the problem is in the board. If BERT successfully completes all the above movements but fails at one of the self-test points, you probably have an intermittent connection. Check your wiring and soldering.

Photo 2: *The results of a BERT construction class at Vancouver Community College.*

Communicating with BERT

After a reset (either power-on or manual), BERT enters his "get commands" state. In this state he sits quietly, receiving and storing ASCII data arriving at his S.IN (serial input) connector. He will stay in this state until one of three events occurs:

1. An E ("End") instruction arrives.
2. His memory fills up with the received data.
3. Improper data is received.

Events 1 and 2 are normal. When they occur, BERT will beep, then enter his "nose?" state. While in this state, BERT constantly monitors his nose button. The only way out of this state is to punch his nose (press the sensor). He will then enter his "interpret" state.

In this state BERT acts like a robot, toddling around executing whatever commands you gave him during his "get commands" state. When he finishes doing all the things you told him to do, he reverts back to his "nose?" state and awaits his chance to do it all over again.

Event 3, however, is an abnormal event, and BERT does not like it. Improper data is defined as either "any data which represents a nonexistent command" or "any non-ASCII data."

Upon receipt of either of these types of garbage, BERT will make a rude noise and glare at you (make a beep and turn on his eye). To put him back in a good mood, press his reset button, then examine your code for the bad instruction, correct the error, and prepare to download the corrected code.

Speech Board

Attempting the self-test without the speech board could get you into trouble as well. Since the OBC waits for the SBY signal from the speech board, it could wait forever (doing nothing else but waiting). To allow BERT to do the self-test without the speech board, short pins 39 and 40 of the PIA together.

If the speech board said "ready" when BERT was first powered up, but did not say "cookie" at this point, the most likely problem is an intermittent connection between the OBC and the speech board. Check all wiring.

The speech board produces sound in response to numbers sent by the OBC. These numbers are loaded into the speech board via a low-going pulse output at the PIA's pin 39. Using a logic probe or oscilloscope, check for this pulse at the PIA, then follow it to see if it actually gets to pin 20 of the SPO256 speech chip.

To tell the OBC that it is ready to receive another number, the SPO256 outputs a low-going pulse on its pin 8. If this pulse is not output by the SPO256, check the power on the speech board (it should be +5 V). If power is okay, check all solder connections and look for bent IC pins. If this pulse is output by the SPO256, you must make sure that it gets to the PIA's pin 40.

If all these handshaking signals come from and go to where they are supposed to, odds are that the SPO256 is working and the problem is in the LM386 audio-amplifier circuit.

Customizing

If you wish to modify BERT, please do. Any robot is only a beginning. Having built it yourself, you are the person best qualified to handle any modifications. My own personal BERT is an awful mess of add-ons and kludges. Perhaps someday I'll clean him up and make those tem-

porary modifications more permanent—maybe just after I add a little more RAM, or perhaps after I modify the nose sensor. . .

Finally, should you start to feel limited by the BERTL language, or should you even wish to write your own robot control language, you may be interested in obtaining an optional ROM for BERT. After you've installed this ROM and added an RS-232C driver (to allow the OBC to talk back to the host computer), BERT will be transformed into a 6800 machine language development device. This will open the door to unlimited control over the hardware. It will allow you to enter, edit, and debug machine language code in the OBC. This ROM, complete with instructions and commented source code listing, is called KRMIN and is available through Amarobot.

[Editor's note: *Since this article was written, the author has added design enhancements to BERT, such as increased motor drive capability, more RAM, and an additional output port. The BERTL language has been revised to accommodate instantaneous response to on-board sensors (allowing BERT to play a version of robot laser tag) as well as increased vocabulary. The circuit boards and the EPROM required for these modifications, in addition to the ones described here, are available from Amarobot. See the address at the end of this article.*]

Pulling the Plug

Well, that's about all there is to it. If you follow all the steps in both parts of this article, you should have your very own robot—probably not the most elaborate robot ever featured as a construction article, but the most complete.

Everything, from hardware to software, is defined, debugged, and obtainable right now. Many configurations of BERTs are roaming around classrooms and basements (see photo 2). The first BERT users group has started here in Vancouver. You can correspond with them by writing the Vancouver Robot Club, c/o Seaport Pacific Services Ltd., Suite 611, 470 Granville St., Vancouver, British Columbia, Canada V6C 1V5.

If you wish to receive a copy of their latest newsletter, send your address and $1 (don't send a stamped envelope unless the postage is Canadian). I wish you good luck with your BERT, and I hope that building and programming him provides you with the opportunity to enter the fascinating domain of robotics and artificial intelligence. ■

All parts for BERT are available from Amarobot, 2913 Ohio St., Richmond, CA 94804, (415) 451-6780.