BUILD THIS AUTOMATIC PARTS TRAY

Here's the software that makes our board-assembly system work, and an optical encoder to make sure it works perfectly.

JAMES J. BARBARELLO

Last time we built a PC-based stepper-motor controller, and an automated parts-dispensing system called the APT (Automated Parts Tray). This time, we'll present the format of the data file that specifies the parts to use in populating a PC board. Then we'll discuss the software (including a complete listing), and show how to use it to populate a sample PC board—the stepper-motor controller presented last time. As a bonus, we'll provide hardware and software details for adding an optical encoder that will allow your PC to sense when bin 1 of the APT is in place.

Data File Format. The APT data file is simply an ASCII text file that lists every part that is to be mounted on a board. The format of the file is shown in Listing 1. (Note: All listings contain line numbers for reference only. If you type a file in, do not enter the line numbers.) Line 1 of the data file provides a description of the board; the description is displayed at the bottom of the screen whenever the APT program is running. Line 2 provides the board's horizontal and vertical dimensions. All dimensions are in inches. Values in the data file should be separated by commas.

Line 3, which contains only a backslash ("\"), says to begin a new part. Following that is one line for every part hole. Each line contains three items: *x* and *y* coordinates, and a third value (polarity), which is normally 0, and should be set to 1 for the key point of a polarized component. For example, the cathode of a diode, pin 1 of an IC, or the positive end of an electrolytic capacitor could be used to help ensure correct part orientation. Each part may have as many as 20 holes. For parts with more than 20 holes (such as a 40-pin IC), simply identify the perimeter holes that define the location and orientation of the part.

After the line of the last hole for the current part comes a line containing only an asterisk ("*"). The following line contains a comment about the part, such as its name, its value, its placement, etc. The next line contains the bin number, and that completes the first part. The next line contains a backslash, which marks the beginning of the next part.

To see how this works in practice, let's examine a sample data file for the stepper-motor PC board, presented last time. Listing 2 shows an abbreviated version of the data file, and Fig. 1 shows the foil pattern. To help identify the *x* and *y* coordinates of each hole, we have overlaid a 0.1-inch × 0.1-inch grid on the pattern. The positive *x* direction may be opposite of what you expect. The reason is that the APT program displays things looking down on the component side of the board, but we have only the foil side to work with. You could work with a component side view, in which case *x* would increase in the positive direction from left to right.

The first set of hole coordinates (lines 4–12) is followed by a comment that indicates those are non-part holes. Typical of that variety would be mounting holes, or holes used for options that are not being installed. Here, the first two sets of coordinates are for the holes where J1 mounts. The last three are for the holes indicated (in the original article) as "Future Optical-Encoder Input," along with holes for ground and +5 volts. None of those holes will have a part inserted, so the bin number (following the asterisk) is zero. That tells the program to display the holes for reference purposes, but not to consider them as part holes, *i.e.*, not to rotate the parts tray.

As shown in lines 13–27, the first part to be installed is J1. Note that 11 holes are specified, and the last hole is shown as the polarity indicator. That is folllowed by the asterisk separator, and a comment line. Since J1 is relatively heavy, the comment indicates that it should be obtained from sup-

LISTING 1

```
1. Board Description (79 characters max)
2. Board Dimensions (x, y)
3. Backslash (\)
4. x,y,p Part1 Hole1(horizontal, vertical,
     polarized (1=yes))
5. x,y,p Part1 Hole2
   ...
n. x,y,p Part1 Hole n
n+1. * (Asterisk)
n+2. Part Description (79 characters max)
n+3. Bin Number (1--12, or 0 to skip)
n+4. Backslash (\)
n+5. x,y,p Part2 Hole1
   ...
```

LISTING 2

```
001. STEPPER CONTROLLER,
     V951003 - JJB
002. 2.85,2.8
003. \
004. .2,.5,0
005. .2,2.35,0
006. .7,.35,0
007. 2.6,1,0
008. 2.6,1.25,0
009. *
010. Non-Part Holes. Bin 0 means
     skip as a part to be placed.
011. 0
012. \
013. .6,.75,0
014. .5,.8,0
015. .6,.95,0
016. .6,1.05,0
017. .5,1.45,0
018. .6,1.6,0
019. .6,1.73,0
020. .6,1.84,0
021. .5,2,0
022. .6,1.95,0
023. .6,2.07,1
024. *
025. J1 - DB25 Conn, Pin 1 Red.
     GET FROM SUPPLEMENTARY
     BIN A.
026. 1
027. \
...
100. .9,.8,0
101. 1.3,.8,0
102. *
103. R7 - 2.2Kohm
104. 2
105. \
...
208. 2.6,.75,0
209. *
210. STEPPER MOTOR PHASE A
211. 1
212. \
...
228. 2.6,1.3,1
229. 2.65,1.6,0
230. *
231. POWER - RED IS +5V,
     WHITE IS GND
232. 1
232. \
```

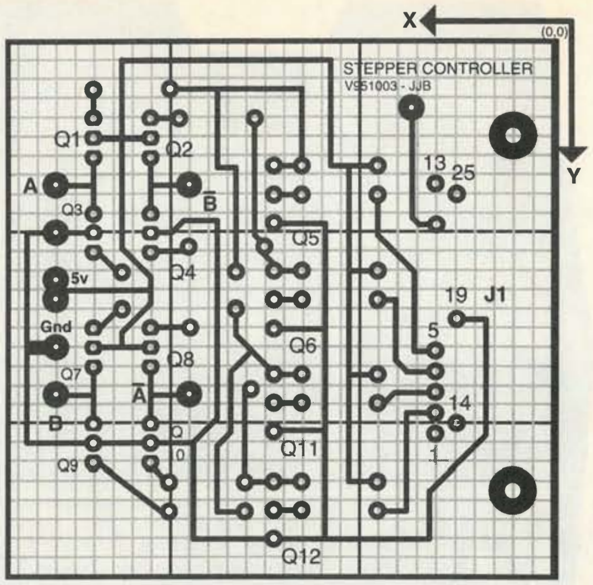


Fig. 1. Here's the foil-side view of the Stepper Motor Controller presented last time. The superimposed grid helps locate component positions accurately.

LISTING 3

```
1 REM** STEPPER.BAS (c) 1995 JJ Barbarello,
      Manalapan, NJ (908) 536-5499
2 REM** V951013
3 DEF SEG = 64
4 DEFINT A: DIM a(4), h(500, 1), p(20, 1), comment$(500), bin(500)
5 abinold = 1: ainit = 12: OUT add, ainit
6 a(1) = 5: a(2) = 3: a(3) = 10: a(4) = 12
7 add = 888
8 OPEN "stepper.fil" FOR INPUT AS #1
9 LINE INPUT #1, file$: CLOSE 1
10 OPEN file$ FOR INPUT AS #1
11 LINE INPUT #1, title$
12 LINE INPUT #1, boardsize$
13 h(0, 0) = VAL(boardsize$)
14 comma = INSTR(boardsize$, ",")
15 h(0, 1) = VAL(MID$(boardsize$, comma + 1, 20))
16 REM******* READ SIZE OF PCB AND LOCATION OF HOLES
17 ctr1 = 1
18 WHILE NOT EOF(1)
19 LINE INPUT #1, a$: B$ = LEFT$(a$, 1)
20 SELECT CASE B$
21 CASE "0" TO "9", "."
22 h(ctr1, 0) = VAL(a$)
23 comma1 = INSTR(a$, ",")
24 h(ctr1, 1) = VAL(MID$(a$, comma1 + 1, 20))
25 ctr1 = ctr1 + 1
26 CASE IS = "*"
27 LINE INPUT #1, a$
28 comment$ = a$
29 LINE INPUT #1, a$
30 bin = VAL(a$)
31 CASE IS = "\"
32 CASE ELSE
33 END SELECT
34 WEND
35 REM** CLOSE & RE-OPEN FILE, DUMMY READ INITIAL DATA
36 CLOSE 1
37 OPEN file$ FOR INPUT AS #1
38 LINE INPUT #1, title$
39 LINE INPUT #1, boardsize$
40 LINE INPUT #1, backslash$
41 REM******* CALCULATE SCALE FACTORS FOR PCB DISPLAY
42 IF (h(0, 1) + 2) > h(0, 0) THEN
43 ystep = 28 / h(0, 1)
44 xstep = ystep * 7 / 5
45 adjust1:
46 IF (xstep * h(0, 0) * 10 > 540) OR (ystep * h(0, 1) * 10 > 280) THEN
47 ystep = ystep - .1: xstep = ystep * 7 / 5: GOTO adjust1
48 END IF
49 ELSE
50 xstep = 56 / h(0, 0)
51 ystep = xstep * .68
52 adjust2:
53 IF (xstep * h(0, 0) * 10 > 540) OR (ystep * h(0, 1) * 10 > 280) THEN
54 xstep = xstep - .1: ystep = xstep * .68: GOTO adjust2
55 END IF
56 END IF
57 REM******* START-UP SCREEN
58 SCREEN 9: CLS
59 LINE (3, 3)-(630, 335), 7, B: LINE (5, 5)-(627, 333), 7, B
60 LINE (230, 30)-(386, 65), 15, B
61 LOCATE 4, 35: COLOR 15, 9: PRINT "APT SYSTEM"
62 PAINT (235, 35), 9, 15: LOCATE 6, 18: COLOR 7, 1
63 PRINT "(c) 1995, JJ Barbarello, Manalapan, NJ 07726"
64 LOCATE 7, 33: PRINT "(908) 536-5499": COLOR 15, 1
65 LOCATE 13, 20: PRINT "Position Tray at Bin 1,
      then press Enter...";
66 OUT add, ainit
67 a$ = INPUT$(1)
68 REM******* WORK SCREEN
69 CLS
70 LINE (1, 1)-(630, 293), 3, BF
71 LINE (1, 1)-(630, 293), 7, B
72 LINE (1, 322)-(630, 334), 3, BF
73 LINE (1, 322)-(630, 334), 7, B
74 titleoffset = (80 - LEN(title$)) / 2
75 LOCATE 25, titleoffset: COLOR 8, 0: PRINT title$; : COLOR 15, 0
76 REM******* CALCULATE WORKING VARIABLES & DISPLAY PCB
77 x = h(0, 0) * 10 * xstep: y = h(0, 1) * 10 * ystep
78 xoffset = (620 - x) / 2: yoffset = (292 - y) / 2
79 LINE (xoffset - 4, yoffset - 2)-(xoffset + x + 4, yoffset + y + 8), 8, BF
80 FOR i = xoffset TO xstep * h(0, 0) * 10 + xoffset STEP xstep
81 FOR j = yoffset TO ystep * h(0, 1) * 10 + 2 + yoffset STEP ystep
82 PSET (i, j), 12
83 NEXT j
84 NEXT i
85 FOR i = 1 TO ctr1 - 1
86 CIRCLE (h(i, 0) * 10 * xstep + xoffset, h(i, 1) * 10 * ystep + yoffset), 1, 14
87 NEXT i
88 REM******* READ PART LOCATION DATA FROM FILE AND DISPLAY
```

plementary bin A. However, the bin identifier is cited as 1. There's still no need to move the tray.

Bin 1 contains the 100-ohm resistors. When we need a new value (R7, 2.2K, lines 100–105), note that the bin number increases to "2." Parts identification continues until it's time to attach the stepper-motor wires (lines 208–212). Here, only a single hole is identified, and the tray stays "parked" at bin 1. The file ends with the power connections (lines 228–232).

The APT Software. Now let's see how the software puts the data file to use. The program, STEPPER.BAS, is shown in Listing 3. (Line numbers are included for reference only. They must not be included in the run-time file). Lines 8 and 9 open a one-line file called STEPPER.FIL. That file contains the name of the text file that contains the PC-board data. You can create the file using any text editor or from a DOS command line as follows:

```
C:APT>copy con stepper.fil
      <Enter>
datafile.txt <Enter>
<Ctrl> <Z> <Enter>
C:APT>
```

In the above, replace "datafile.text" with the name of the file with the PC-board data that you created. The term <Enter> means press the Enter key. <Ctrl> <Z> means press and hold the CTRL key while you press the Z key.

The program then opens the PC-board data file and reads all the component location information (lines 10–40). The program uses that information to calculate the scale at which the PC board will be displayed (42–56). The next step is to display basic program information (57–75). Lines 77–87 then display the PC board. Note that the program runs in screen mode 9 (see line 58), which is a screen resolution of 640 × 350. Nearly all EGA, VGA, and SVGA adapters support that mode, but earlier adapters do not, so don't plan on using that old 8088-based PC you have lying around for this project unless it has an appropriate video adapter.

Next, the program (lines 89–125) displays the holes associated with each part, calling subroutine HWCONTROL (from 113) to position the APT tray at the designated bin.

HWCONTROL. The purpose of HWCONTROL is to find the shortest route between the current and the next bin positions, and then move the tray to the new position.

HWCONTROL actually does three

```
89 hole = 1: partno = 1
90 WHILE NOT EOF(1)
91 LINE INPUT #1, a$: B$ = LEFT$(a$, 1)
92 SELECT CASE B$
93 CASE "0" TO "9", "."
94 x = VAL(a$)
95 comma1 = INSTR(a$, ",")
96 comma2 = INSTR(comma1 + 1, a$, ",")
97 y = VAL(MID$(a$, comma1 + 1, comma2 - comma1 + 1))
98 positive = VAL(MID$(a$, comma2 + 1, LEN(a$) - comma2))
99 x = x * 10 * xstep + xoffset: y = y * 10 * ystep + yoffset
100 p(hole, 0) = x: p(hole, 1) = y: hole = hole + 1
101 IF positive = 1 THEN highlight = 12 ELSE highlight = 15
102 CIRCLE (x, y), 3, highlight
103 PAINT (x, y), highlight, highlight
104 CASE IS = "*"
105 LINE INPUT #1, a$
106 comment$ = a$
107 LINE INPUT #1, a$
108 bin = VAL(a$)
109 LOCATE 22, 1: PRINT SPACE$(79);
110 LOCATE 23, 1: PRINT SPACE$(79);
111 LOCATE 22, 1: PRINT comment$
112 LOCATE 23, 1: COLOR 7, 0: PRINT "Part"; partno; "in BIN:"; bin:
      COLOR 15, 0
113 IF bin > 0 THEN GOSUB hwcontrol: a$ = INPUT$(1)
114 FOR i = 1 TO hole - 1
115 CIRCLE (p(i, 0), p(i, 1)), 3, 10
116 PAINT (p(i, 0), p(i, 1)), 8, 10
117 CIRCLE (p(i, 0), p(i, 1)), 3, 8
118 CIRCLE (p(i, 0), p(i, 1)), 1, 14
119 NEXT i
120 hole = 1
121 CASE IS = "\"
122 IF bin > 0 THEN partno = partno + 1
123 CASE ELSE
124 END SELECT
125 WEND
126 SOUND 600, 1: SOUND 875, 2: SOUND 800, 2
127 bin = 1: GOSUB hwcontrol
128 SCREEN 0: CLS : LOCATE 10, 38: PRINT "Done"
129 LOCATE 18, 1
130 END
131 '**************************************
132 REM** SUBROUTINE HWCONTROL
133 hwcontrol:
134 abin = bin
135 IF abin < 1 OR abin > 12 THEN ERROR 5
136 abin2 = abin
137 IF abin2 < abinold THEN abin2 = abin2 + 12
138 asteps = abin2 - abinold
139 SELECT CASE asteps
140 CASE 1 TO 6
141 lo = 1: hi = 4: steps = 1
142 CASE IS > 6
143 lo = 4: hi = 1: steps = -1: asteps = 12 - asteps
144 CASE IS < 0
145 lo = 4: hi = 1: steps = -1: asteps = -asteps
146 CASE ELSE
147 SOUND 675, 1: SOUND 500, 1
148 RETURN
149 END SELECT
150 OUT add, ainit
151 binold = abinold
152 FOR i = 1 TO asteps
153 FOR j = lo TO hi STEP steps
154 OUT add, a(j)
155 IF i > 1 AND i < asteps THEN delay = .075 ELSE delay = .125
156 START! = TIMER
157 WHILE (TIMER - START!) < delay: WEND
158 NEXT j
159 binold = binold + steps
160 SELECT CASE binold
161 CASE IS = 0
162 binold = 12
163 CASE IS = 13
164 binold = 1
165 CASE ELSE
166 END SELECT
167 NEXT i
168 IF steps = -1 THEN
169 OUT add, a(lo)
170 ainit = a(lo)
171 ELSE
172 ainit = a(hi)
173 END IF
174 START! = TIMER
175 WHILE (TIMER - START!) < 1: WEND
176 OUT add, 0
177 abinold = abin
178 SOUND 500, 1: SOUND 675, 1
179 RETURN
```

TABLE 1—MOTION CONTROL SEQUENCE

Step	A(5)	B*(4)	B(3)	A*(2)	Decimal
1	1	0	1	0	10
2	0	0	1	1	3
3	0	1	0	1	5
4	1	1	0	0	12

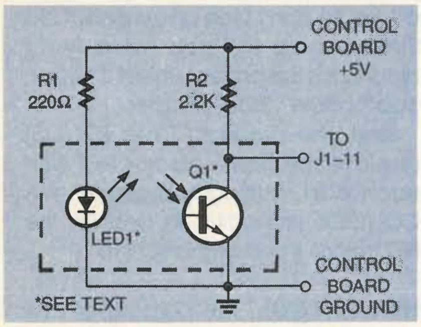


Fig. 2. The optical encoder consists of little more than an IR emitter and detector.

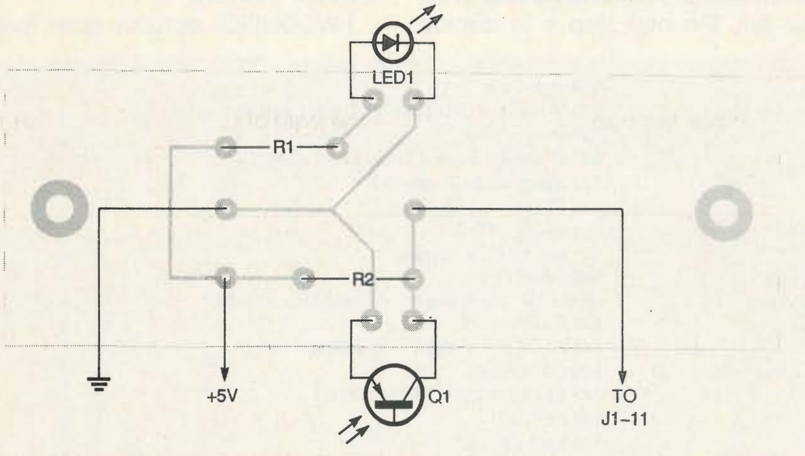


Fig. 3. Mount all components for the optical encoder board as shown here. Note that though shown off board, LED1 and Q1 are mounted as discussed in the text.

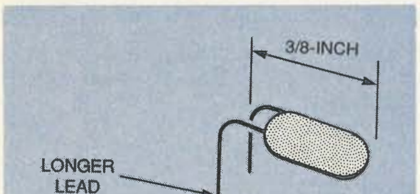


Fig. 4. Bend the leads of LED1 and Q1 to keep the bodies of those devices about a half-inch above the PC board.

things: It decides whether to move the tray, decides how to move the tray, and actually moves the tray. Table 1 shows the motion-control sequence. By performing steps 1–4 sequentially, the tray rotates clockwise (CW). By performing the sequence backward (4–1), the tray rotates counterclockwise (CCW).

Line 134 saves the bin number to a temporary variable (abin). Line 135 ensures that a valid bin number is being processed. Line 136 creates another copy (abin2) of the current bin number. Then, if the value of the new bin is less than the value of the last bin (*i.e.*, the current tray position), line 137 adds 12 to abin2. To understand why, let's look at an example.

If we are at bin 6 (abinold = 6) and we want to go to bin 4 (abin2 = 4) we want to go backward two positions. So, adding 12 to abin2 gives 16. The number of steps is thus 16 − 6 = 10.

Now look at the SELECT CASE routine (139–149). If the number of steps we need to take is between one and six, we simply set up to rotate clockwise (CW) the specified number of steps. If we need to go more than six steps, we set up to go counterclockwise (CCW). Likewise, if the number of steps is less than zero, we also set up to go CCW.

In our example, line 143 does three things. First, it sets variable lo to 4 and hi to 1. Second, it sets the stepping value to −1, so the For-Next loop will count backward. The result of the first two steps is CCW rotation. Third, it subtracts the current astep value (which is 10) from 12 to arrive at 2. As a result, the tray will move two steps CCW, from bin 6 to bin 4.

To continue the example, assume we now want to move to bin 7. Line 137 does not come into play, since 7 is greater than 4. Line 138 calculates the difference as 3, which is handled in lines 140–141. The result is CW rotation, so the tray moves CW three positions to bin 7.

It's also worth pointing out that if the number of steps needed to change bins is zero, the CASE ELSE statement takes effect, so the routine returns without performing any action. That's important to remember when building the data file, since sequencing parts with identical values speeds up the assembly process.

All that's left now is to perform the actual motion. Line 150 outputs a decimal 12, which is always the start of the movement sequence. Then the For-Next loop (153–158) cycles through the four steps. Lines 155–157 provide a variable delay that allows the motor time to spin up the first time through the loop. Subsequently, a shorter delay suffices, thus speeding movement.

Lines 174–175 provide a one-second delay, after which line 176 sends a value of zero. The delay keeps the stepper energized to ensure that it

PARTS LIST FOR THE OPTICAL ENCODER

Q1—Infrared detector, 940nm, T1 case (Jameco 112176 or equivalent)
LED1—Infrared LED diode, T1 case (Jameco 112150 or equivalent)
R1—220 ohms, ¼-watt, 5% resistor
R2—2200 ohms, ¼-watt, 5% resistor
Poster board, PC board, wire, mounting hardware, etc.
NOTE: The following are available from James J. Barbarello, 817 Tennent Road, Manalapan, NJ 07726. PC Board: APT-PC ($12.00). Software including source and executable code for all software in the series, sample data files, and an enhanced version of the APT software (including the optical encoder application): APT-S ($12.00). All prices include shipping. International orders add $5.00 (U.S. funds only). NJ residents must add sales tax. The author will accept phone calls 6–8:00 PM EST, Monday through Friday ONLY (Tel: 908-536-5499). The author will gladly answer written questions if accompanied by a self-addressed stamped envelope.

brakes properly. The zero then de-energizes the stepper. We could leave it energized, but if no further motion is needed in the near future, continuing to apply current will simply cause unnecessary heating of the coil. The last act of the subroutine is to record the current bin position (abin) as the previous bin position (abinold), in preparation for the next time through the loop.

LISTING 4

```
1 REM** STEPFDBK.BAS
2 REM** V951014
3 CLS : DEF SEG = 64: add = 888
4 DIM a(4): a(1) = 5: a(2) = 3: a(3) = 10:
     a(4) = 12: ainit = 12
5 LOCATE 1, 25: PRINT "ALIGN
     OPTICAL ENCODER DISK"
6 LOCATE 3, 1: PRINT "Searching...";
7 startloop:
8 FOR j = 1 TO 4
9 OUT add, a(j)
10 IF (INP(add + 1) AND 128) / 128 = 1
     THEN GOTO foundbin1
11 start! = TIMER
12 WHILE (TIMER - start!) < .075: WEND
13 NEXT j
14 GOTO startloop
15 foundbin1:
16 OUT add, ainit
17 LOCATE 3, 1
18 PRINT "Rotate Disk So Aperture is
     Directly Under IR Diode."
19 PRINT "When Done, press Enter to
     Turn Off Stepper Motor...";
21 a$ = INPUT$(1)
22 OUT add, 0
```

Running the Software. To run the software, you need three things: STEPPER.BAS (an ASCII file containing the QBasic program), STEPPER.FIL (a file that contains the name of the data file), and a data file (*e.g.*, STEPPER.DAT). You can create the files yourself, as shown in Listings 1, 2, and 3, or you can obtain them directly from the author, as discussed in the Parts List.

Next, connect the APT hardware to a power supply and parallel port. Run QBasic, load STEPPER.BAS, and run it. The program asks you to move the tray to bin position 1. Remove the APT's cover, and position the tray so bin 1 is accessible through the cutout. Reinstall the cover and press <Enter>.

The program then displays your PC board, highlighting each part in turn, and rotating the bin as necessary to provide access to the specified part. After the last part, the program de-energizes the stepper motor.

Now you're ready to create a data file for your own PC board. Go to it!

Optical Encoding. The current APT system is *open-loop:* We must assume that the tray moved to where we sent it, because the system provides no feedback to verify whether it did. A *closed-loop* system would provide such feedback. We're going to provide that feedback using optical encoding. Optical encoding is simply a way to sense rotational position using light.

The circuit is quite simple, as shown in Fig. 2. The collector of Q1 is normally high. When infrared (IR) energy hits the detector, it begins to conduct, so its collector goes low. By monitoring the state of Q1's collector, we can, under software control, say whether light is passing through or being blocked by a device. The device in this case happens to be a disc with a slit in it. The disc mounts concentric with the motor shaft, and the IR detector/emitter pair mount in such a way that the slit in the disc will pass between them as the motor rotates.

Parts alignment is somewhat critical, so we recommend the use of a PC board; a foil pattern accompanies this article. The parts-placement guide for that board appears in Fig. 3. The IR LED has a bluish lens, and the cathode is identified by a longer lead. You can verify that configuration by connecting the longer lead to +5-volts DC through a 220-ohm resistor, and connecting the shorter lead to ground. There should be about 1.2 volts across the diode. The detector has a clear lens, and the emitter is identified by a longer lead. Each device comes in a T1 package, which is just the right size for this application. A T1½ package would be too large.

Bend the IR devices as shown in Fig. 4, and install them on the PC board. Keep the component bodies about ½-inch above the plane of the PC board. Then install the two resistors and three wires for +5V, ground, and the output. The output wire connects to pin 11 of J1 on the original APT circuit.

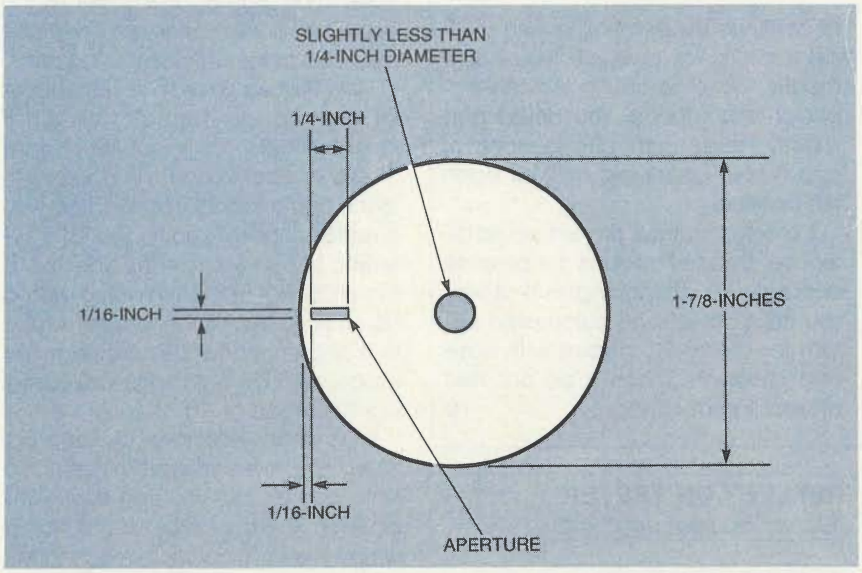


Fig. 5. The encoder disc must press-fit on the motor shaft, so be careful creating the center hole.

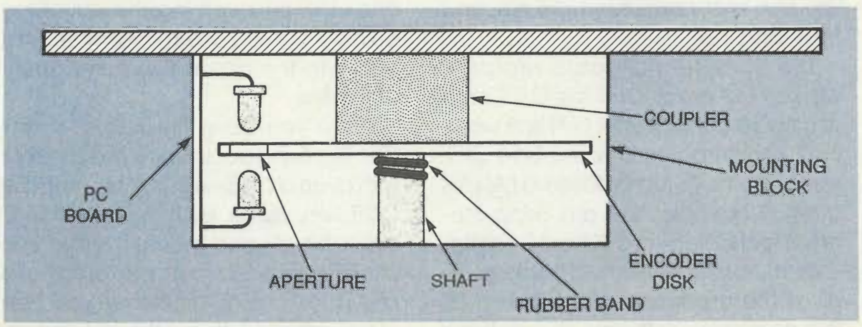


Fig. 6. Position the PC board so that the IR devices line up with the disc aperture between them.

You can make an encoder disc from poster paper or other stiff, thin material, as shown in Fig. 5. The diameter of the center hole should be slightly smaller than the shaft of the stepper motor. To mount the disc, remove the coupler on the stepper motor and loop a small rubber band around the shaft several times.

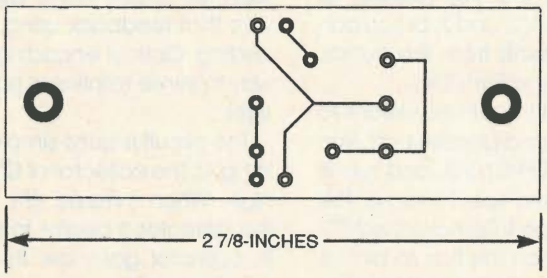
Referring to Fig. 6, push the encoder disc onto the shaft. The disc should have a firm press-fit, without bending or distorting. Reinstall the coupler and adjust the disc and the rubber band so that the disc rests directly against the bottom of the coupler and is held tight by the rubber band. Position the encoder PC board against the short ends of the blocks holding the stepper motor (shown last time), and adjust the positions of the IR emitter and detector so that they are close to the encoder disc and centered along the aperture in the disc. Secure the encoder PC board to the short ends of the blocks.

To test the circuit and align the disc, use the program shown in Listing 4. Connect the APT to your power supply and parallel port, but leave the tray off for now. Run the program and note that the encoder disc rotates. The disc should stop with the aperture directly between, or at least near, the IR devices. If it's directly under, you're done. Otherwise, rotate the disc slightly, as instructed by the program. In either case, press <Enter> to end the program. Move the shaft (not the disc) so the aperture is not under the diode and run the program again.Now, the disc should stop with its aperture directly under the diode. After completing the alignment, reinstall the tray.

Let's take a quick look at the alignment program. Lines 7–16 form a loop that continually looks for the aperture. Within that loop, lines 8–13 constitute our old stepper-motor movement routine. Line 10 senses when pin 11 of J1 goes low (input = 1), indicating that IR energy from the diode is passing through the aperture and pulling the detector's output low. Since the aperture could be detected during any of the four motion steps, line 16 repositions the stepper motor at the start of the sequence (ainit). The program then lets you align the aperture beneath the diode. Last, line 22 de-energizes the motor.

What's Next? There are several directions you could take if you wanted to continue experimenting with stepper motors. For example, you could modify the program to include encoder-disc sensing. You could also modify the encoder disc to incorporate twelve apertures, one for each bin position.

A more ambitious project would be to use stepper motors to provide accurate X-Y positioning. In that way you could create an automated system for drilling PC boards with ease and precision. That will be our next project; look for it soon! Ω



Since parts alignment is fairly critical, use this PC board when building the optical encoder.

REFLECTION TESTER

(Continued from page 62)

ment. You have positively found an impedance mismatch in the cable. You also know exactly where it is, and how long it is.

The TV-antenna-cable problem worked out great. Can the C.R.T. help you figure out why your LAN isn't working? Let's imagine a really bad scenario. You have just installed a LAN for a small business. You are using Ethernet technology in a 10Base2 configuration. You have finished hooking up all of the hardware and installed all the necessary software, but nothing works. What do you do?

After you sit and stare for a while thinking how nice it would be camping in the middle of a stand of eastern white pine, a thought crosses your mind. Could the problem be in the cabling? You take out your C.R.T. and put it in-line with one of the workstations. The waveform on the scope looks like the display in Fig. 11. The measurement of the initial step up accounts for the 25 foot length of cable between the C.R.T. and the workstation, but at the end there is a strange step down.

A step down in the display, as we just learned in the TV antenna cable example before, indicates an impedance mismatch. But where is that step down coming from? The workstation is properly terminated with a 50-ohm terminator. But then you notice that you have wired the system with RG-59/U instead of RG-58/U. The cable is not properly matched to the Ethernet cards and terminators. The Ethernet is interpreting that mismatch as data collisions. Obviously, new cable will have to be ordered, but hopefully the cable has not yet been run under the floor tiles or through the ceiling joists.

Now that we've seen several cases of impedance mismatches, what does a healthy cable look like? Figure 12 shows a cable that is in good condition and is properly terminated with a resistor that is equal to the characteristic impedance of the line. It is a length of RG-58/U terminated with a 50-ohm resistor. If the proper cable was chosen for the LAN problem, the scope display would have looked like Fig. 12 instead of Fig. 11.

If the world were an ideal place, Fig. 12 would show a straight line after the initial ringing, but you can see that it doesn't. There is still a slight hump where the terminator connects to the cable, but it is very small and there are no multiple reflections. Also notice one other curious thing: the waveform is slowly curving down over time. That is due to the capacitive component of the line.

There you have it—a poor man's TDR. Suprising accuracy is practicable with even a crude setup. If you find the C.R.T. very useful, lay it out on printed-circuit board and mount it either in a box or in your oscilloscope. If you have any questions or problems, you can reach the author via e-mail at 75104.3104@compuserve.com. Ω