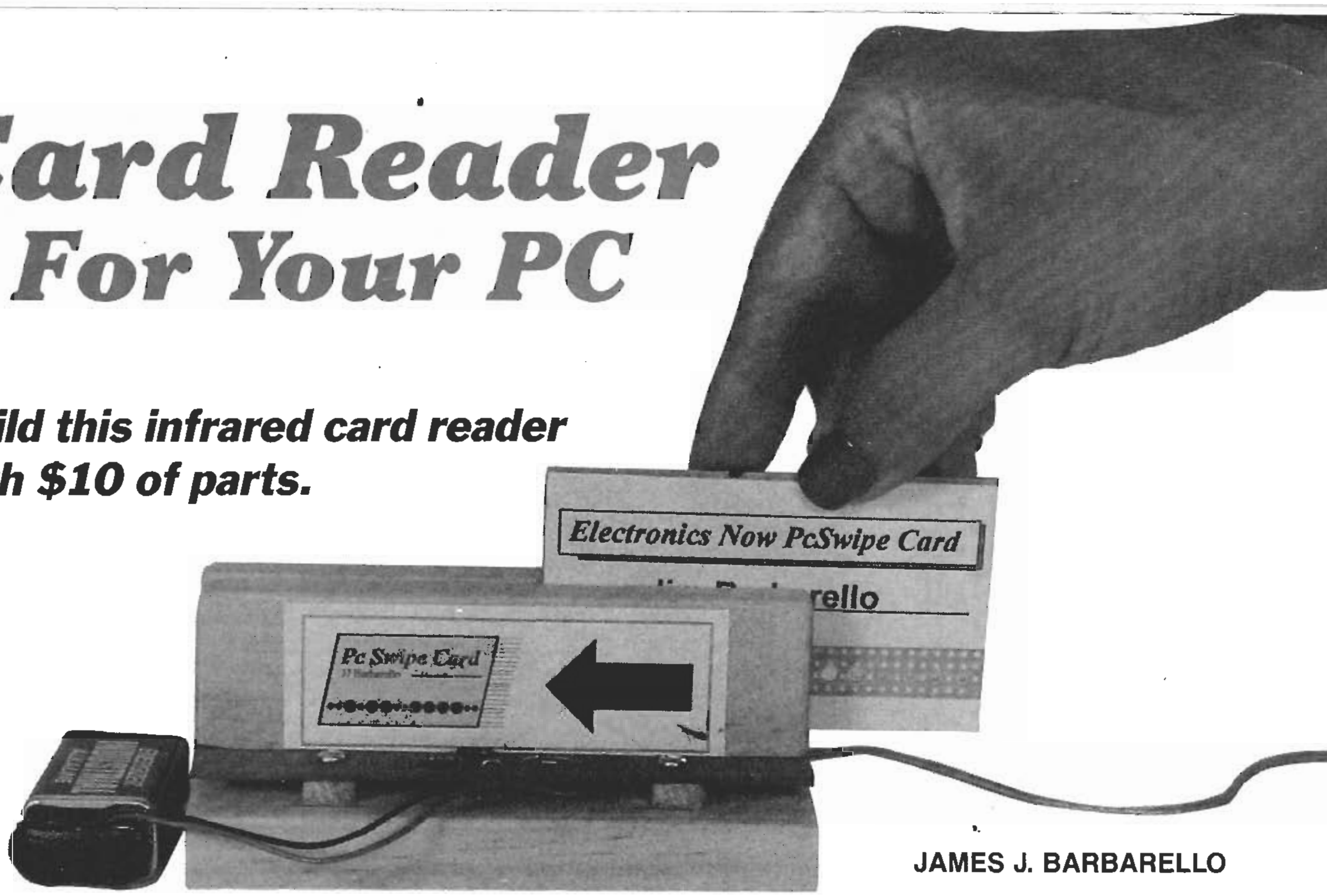
# Card Reader For Your PC

***Build this infrared card reader with $10 of parts.***





JAMES J. BARBARELLO


HERE'S A NEAT LITTLE PROJECT that consists of only five components and a simple Basic program. Construction time is about an hour, and parts cost less than $10, and are all readily available. What you end up with is a versatile gadget I call PcSwipe. PcSwipe is a PC-based infrared information reader that can be used in many ways.

For example, assume you're a small-business person looking for a unique way to draw customers into your shop. You send a packet of advertising material to your customers. The packet contains a stiff, thin card that looks like a credit card, but it has a series of holes across the bottom. The packet also contains a note stating, "Here's your invitation to drop by Acme Electronics on Main Street. When you come in, bring your invitation card and run it through our computerized card reader. Who knows? You might be our grand prize winner!"

The customer's interest is piqued, so he makes a trip to your store. On entering, he sees a computer displaying the message, "Welcome! Try your card to see if you've won the grand prize." The customer runs his card through the reader and . . . you can complete the story.

**How it works**

PcSwipe works by sensing the presence and size of holes in a card passed by its infrared LED and phototransistor. The detection algorithm implemented in the Basic program is not affected by variations in speed as the card passes through the reader.

The card measures 2 × 3⅜ inches and contains a row of 16 holes spaced on 0.2-inch centers located one-half inch up from the bottom. Based on hole size, each hole can represent either a one or a zero. Small holes (1/16 inch) represent logic 0; large holes (⅛ inch) represent logic 1. Hole 1 functions as a "start" bit; the remaining 15 holes provide $2^{15}$ or 32,768 combinations.

The card can be fabricated easily from such materials as card stock, plastic, PC board stock, perforated board, or aluminum. The circuit can be powered from any 5–15-volt source, including a 9-volt battery. Further, you can use any PC to run the software.

In the demonstration program, PcSwipe simply displays a message on the screen. Of course, it could do more than just display messages. You could easily modify the program to look up names and other information in a file, print a receipt, or activate a relay. Your imagination is the only limit.

**Circuit and components**

Figure 1 shows the complete circuit. The LED is a high-output infrared emitter. It receives its power through current-limiting resistor R1. With a 9-volt power supply and a value of 220 ohms for R1, the diode will receive about 25 millamperes of current. (The diode has a forward voltage drop of about 1.2 volts). With a 5-volt supply, R1 should have a value of 150 ohms to keep diode current in the 25-milliampere range.

The LED energizes NPN phototransistor Q1, which is configured as a simple inverting amplifier. As more light shines on Q1, the output voltage at its collector decreases. With a value of 2.2 kilohms for R2, the circuit provides TTL-compatible logic levels.

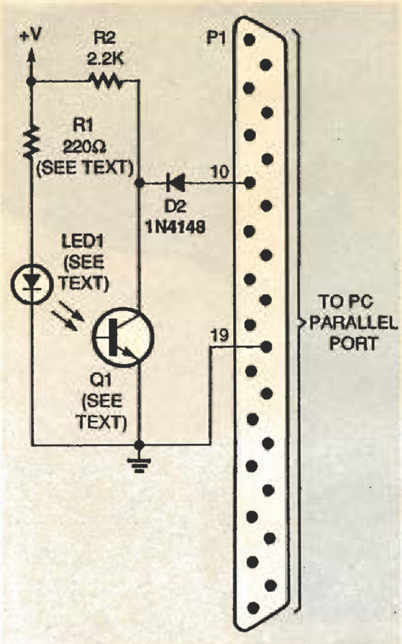



**FIG. 1—THE COMPLETE CIRCUIT consists of an LED, a phototransistor, a diode, and two resistors.**

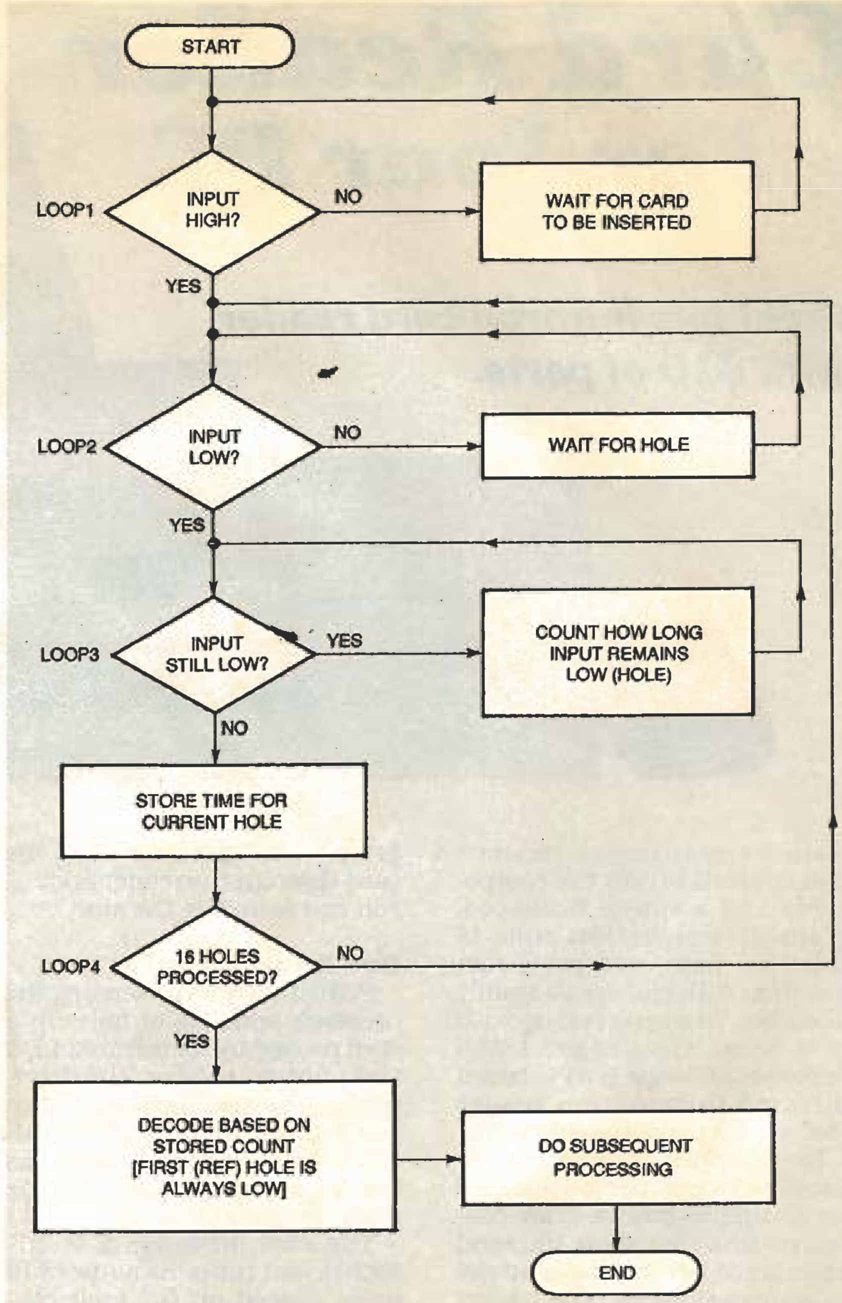



**FIG. 2—FLOWCHART OF THE BASIC SOFTWARE. Loop 1 waits for a card to be inserted. Loops 2 and 3 measure a hole, and Loop 4 ensures that all 16 holes are measured. Following loop 4, the software converts the decoded values into a binary number.**

The output of Q1 feeds one bit of a PC's parallel port. Diode D2 allows the use of power sources greater than 5 volts, thereby maintaining TTL level compatibility, even with higher supply voltages. If the voltage at the collector of Q1 ever exceeds 5 volts, D2 will block the voltage, thereby protecting the port. On the other hand, when Q1 goes logic low, D2 becomes forward-biased, so the low level can be sensed by the port.

The circuit can be powered by a wide range of supplies, including three series-connected AA cells for a total of 4.5 volts, a 5-volt power supply, a single 9-volt battery, an unregulated 9-volt source, a 100-milliampere power cube, or a variable power supply set that can supply between 4.5 and 15 volts.

A few notes about the emitter/detector pair: I used readily available Radio Shack parts, but others with similar characteristics can be used. Here's some information about the specified parts to help if you'd like to make substitutions.

The high-output infrared LED is a two-terminal device in a T-1¾ package. It has a minimum radiant power output of 16 milliwatts when driven with 100 milliamperes of current. At 20 milliamperes, its forward voltage is a maximum of 1.6 volts. The LED has a viewing angle, at half intensity, of 45°, meaning that it has a wide viewing angle. A flat on the package base indicates the cathode.

The infrared phototransistor is also packaged in a two-terminal, T-1¾ case. It is an NPN, silicon-transistor, with high speed and high photosensitivity. Rise and fall times are in the 5 to 10 microsecond range, and collector-emitter saturation voltage is between 0.3 and 0.5 volt. A flat on the package base indicates the collector.

### Software

To understand how the software senses information, refer to the flow chart, Fig. 2. At the beginning of the program (START), three assumptions are made: 1) The circuit is

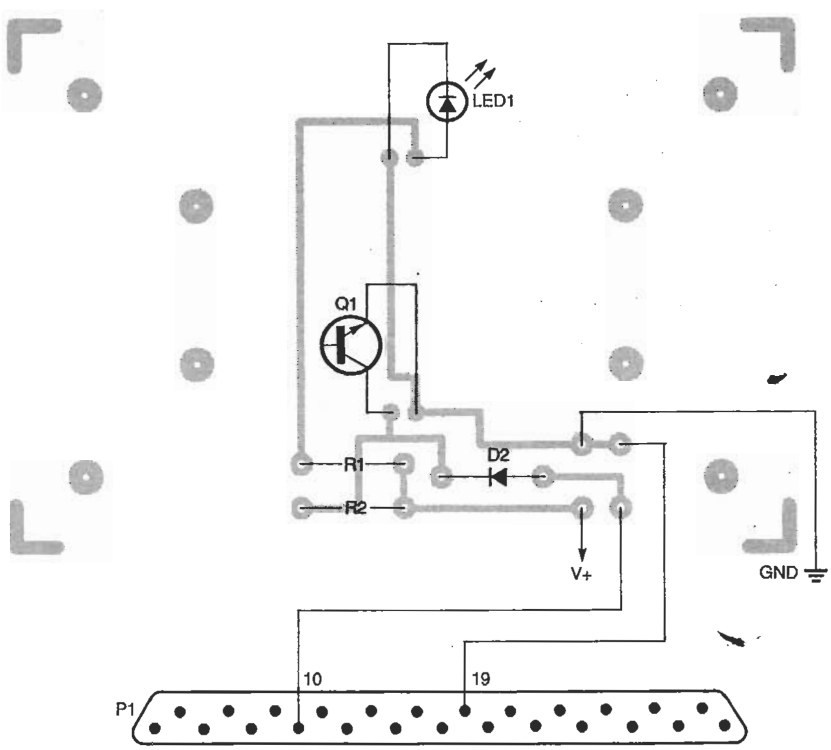



**FIG. 3—MOUNT ALL COMPONENTS as shown here.**

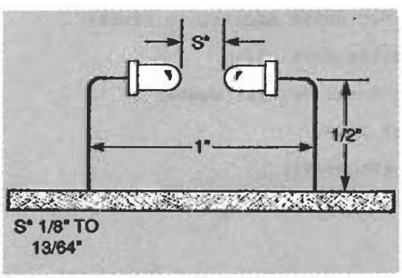



**FIG. 4—MOUNT THE LED AND transistor one inch apart, one-half inch above the bottom of the board, and separated by a gap of about one-eighth inch.**

powered up, 2) The LED is emitting infrared energy, and 3) The phototransistor is conducting, thus providing a logic low to the PC. The first loop then begins. It waits for the user to insert a card that will cause the transistor to conduct.

Once a card has been inserted, loop 2 looks for a logic low. When the low occurs, the program knows it's at the beginning of a hole. Loop 3 then begins counting to provide a relative measure of the diameter of the hole. Counting continues until the input goes high again, and then the count is saved.

Loop 4 continues the loop 2 and loop 3 process until all 16 holes have been counted. (Note

## PARTS LIST

**All resistors are ¼-watt, 5%, unless otherwise noted.**

LED1—High-output infrared LED SSY-IR53L (Radio Shack 276-143 or equivalent)
D1—not used
D2—1N4148 switching diode
P1—DB-25 male connector
Q1—Infrared phototransistor SY32PT NPN (Radio Shack 276-145 or equivalent)
R1—150 to 220 ohms (see text)
R2—2200 ohms
Miscellaneous: PC board and construction material (see text), power connector or 9-volt battery snap, swipe card material (punched perfboard or equivalent), mounting hardware, wire, solder, etc.

**Note: A disk containing several application programs (both source and executable code), as well as instructions on how to use and modify them is available for $12.00 from JJ Barbarello, 817 Tennent Road, Manalapan, NJ 07726. Specify SWIPE-S when ordering.**

that only 15 of the holes contain information. The first hole functions as a "start bit.") After processing all 16 holes, the counts are decoded to determine the binary number represented by the holes.

The decoding scheme is simple but powerful, as it ignores changes in velocity as the card moves through the slot. The ratio of the large-hole diameters to the small-hole diameters is 2:1. Allowing for some variation in vertical alignment and size, a 1.5:1 factor in the software determines whether the current hole is larger, smaller, or the same as the previous hole. The first hole functions as a refer-

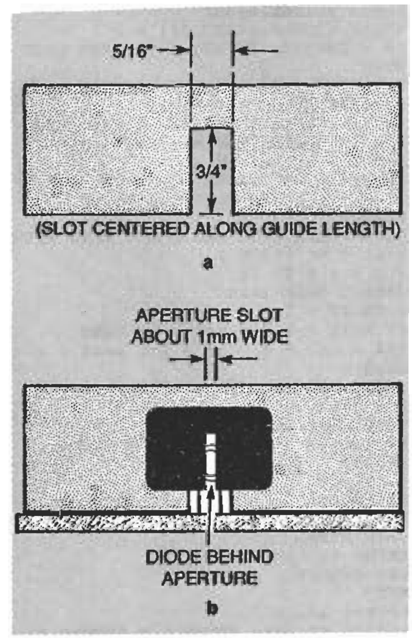



**FIG. 5—CUT A SLOT IN EACH GUIDE as shown in (a). Cut a 1 mm slit in a write-protect tab, and center the tab over the guide slot, as shown in (b).**

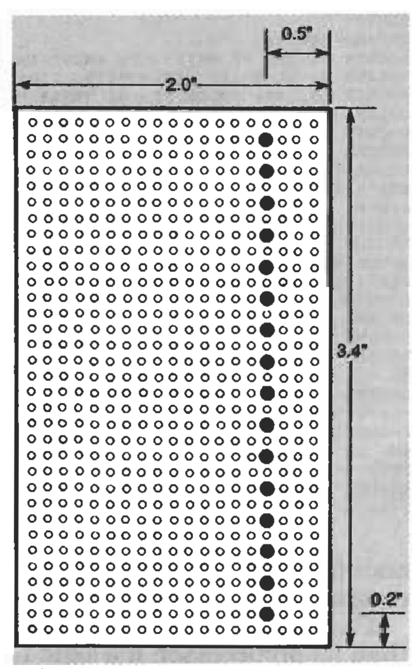



**FIG. 6—TO MAKE A SWIPE CARD, cut a piece of perfboard as shown here.**

## LISTING 1—MAIN PROGRAM

```
REM*****************************************************
REM** SWIPE.BAS V950121 (c) 1995, JJ Barbarello *****
REM** NOTICE: This is a non-compilable version *****
REM*****************************************************
CLEAR : CLS : DEFINT A-X: DEFSTR Y-Z: DIM x(16)
DEF SEG = 64: ON ERROR GOTO errortrap
OPEN "R", 1, "BITPORT.DAT": FIELD 1, 4 AS a$
IF LOF(1) = 0 THEN
a1 = PEEK(8) + 256 * PEEK(9) + 1
ELSE
GET 1, 1: a1 = VAL(a$) + 1
END IF
CLOSE 1
REM************** MAIN PROGRAM LOOP
start1:
GOSUB screenlayout
WHILE (INP(a1) AND 64) = 0
a$ = INKEY$: IF a$ <> "" THEN GOTO readytoend
WEND
x = 0: j = 0: start! = TIMER
readholes:
WHILE (INP(a1) AND 64) = 64: WEND
x = 0: WHILE (INP(a1) AND 64) = 0: x = x + 1: WEND
j = j + 1: x(j) = x
IF x = 0 OR (TIMER - start!) > 2 THEN ERROR 6
IF j < 16 THEN GOTO readholes
done1:
VIEW PRINT 3 TO 24: CLS : VIEW PRINT: BEEP
stat = 0: ttl = 0
FOR i = 2 TO 16
SELECT CASE stat
CASE IS = 0
IF x(i) > 1.5 * x(i - 1) THEN
ttl = ttl + 2 ^ (i - 2): stat = 1
ELSE
stat = 0
END IF
CASE IS = 1
IF x(i) < .667 * x(i - 1) THEN
stat = 0
ELSE
ttl = ttl + 2 ^ (i - 2): stat = 1
END IF
CASE ELSE
ERROR 6
END SELECT
NEXT
LOCATE 14, 3
LOCATE 10, 35: PRINT "ID SENSED:"; ttl
GOSUB screenlayout
GOTO start1
readytoend:
IF a$ = CHR$(27) THEN CLS : LOCATE 18, 1, 1: END
BEEP: GOTO readholes
REM**
REM** SCREEN LAYOUT
REM**
screenlayout:
LOCATE 1, 34, 0: PRINT "Pc SWIPE CARD";
LOCATE 2, 1: PRINT STRING$(79, 220)
LOCATE 18, 35: COLOR 23, 0: PRINT "Waiting........"; :
COLOR 7, 0
LOCATE 21, 33: PRINT "(Press ESC to end)"
RETURN
REM**
REM** ERROR TRAP
REM**
errortrap:
IF ERR = 6 THEN
SOUND 500, 1
CLS : LOCATE 1, 34: PRINT "Pc SWIPE CARD";
LOCATE 2, 1: PRINT STRING$(79, 220): COLOR 0, 7
LOCATE 9, 25: PRINT SPACE$(34)
LOCATE 10, 25: PRINT " Error In Reading Swipe Card. "
LOCATE 11, 25: PRINT " Wait For The Beep and Try Again.
"
LOCATE 12, 25: PRINT SPACE$(34): COLOR 7, 0
start! = TIMER
WHILE (TIMER - start!) < 1: WEND: CLS
END IF
BEEP
RESUME start1
```

## LISTING 2—TEST PROGRAM

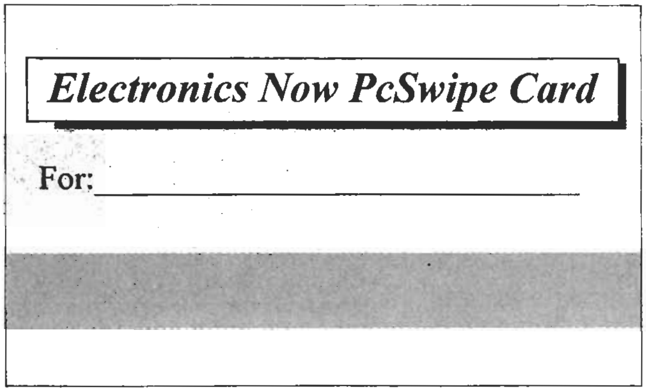
```
REM***************************
REM** SWIPETST.BAS 1/20/95 *
REM***************************
CLEAR : CLS: DEFINT A-X: DEF SEG = 64
a1 = PEEK(8) + 256 * PEEK(9) + 1
LOCATE 1, 34, 0: PRINT "PcSWIPE TEST"
LOCATE 2, 1: PRINT STRING$(79, 220)
LOCATE 4, 31: PRINT "(Press ESC To End)"
previous = (INP(a1) AND 64) / 64
LOCATE 10, 39
IF previous = 1 THEN PRINT "HI" ELSE PRINT "LO"
loop01:
a = (INP(a1) AND 64) / 64
a$ = INKEY$: IF a$ <> "" THEN GOTO endit
LOCATE 10, 39
IF a = 1 AND previous = 0 THEN
SOUND 600, 1
PRINT "HI"
previous = 1
ELSEIF a = 0 AND previous = 1 THEN
SOUND 100, 1
PRINT "LO"
previous = 0
END IF
GOTO loop01
endit:
END
```

## LISTING 3—DECIMAL TO BINARY CONVERSION

```
REM****************************
REM** SWIPENOS.BAS 1/20/95 **
REM****************************
CLEAR : CLS : DIM n$(14)
LOCATE 1, 23: PRINT "PC SWIPE DECIMAL TO BINARY
CONVERSION"
LOCATE 2, 1: PRINT STRING$(79, 220)
loop1:
LOCATE 6, 23: INPUT "Enter Decimal Number (0 to
32767).."; n
IF n < 0 OR n > 32767 THEN
BEEP
LOCATE 6, 20: PRINT SPACE$(50)
GOTO loop1
END IF
number = n
FOR i = 14 TO 0 STEP -1
bin = 2 ^ i
IF bin <= n THEN n = n - bin: n$(i) = CHR$(79) ELSE
n$(i) = CHR$(248)
NEXT
LOCATE 10, 1
LOCATE 10, 23: PRINT CHR$(218); STRING$(33, 196);
CHR$(191)
FOR i = 11 TO 15
LOCATE i, 23
PRINT CHR$(179); SPACE$(33); CHR$(179)
NEXT i
LOCATE 16, 23: PRINT CHR$(192); STRING$(33, 196);
CHR$(217)
LOCATE 13, 25: PRINT "Ref"; : LOCATE 14, 25: PRINT
CHR$(179);
LOCATE 15, 25: PRINT CHR$(248); " ";
FOR i = 0 TO 14
PRINT n$(i); " ";
NEXT i
LOCATE 12, 35: PRINT USING "ID: #####"; number
LOCATE 20, 23: PRINT "Press a key to try again, ESC to
end...";
LOCATE 6, 23: PRINT SPACE$(50)
a$ = INPUT$(1)
IF ASC(a$) = 27 THEN END
LOCATE 20, 23: PRINT SPACE$(50)
GOTO loop1
```

ence for all subsequent possible comparisons.

If a hole is at least 50% larger than its predecessor, it a logic 1. If it is less than 66% of the size of its predecessor, it is a logic 0. If a hole is neither larger or smaller than its predecessor, it must be the same. Velocity from hole to hole (a distance of about 0.2 inch), remains essentially the same, so speed variations during the swipe will not affect sensing accuracy. The main program appears in Listing 1. All software listings needed for this project can be downloaded from the Gernsback BBS



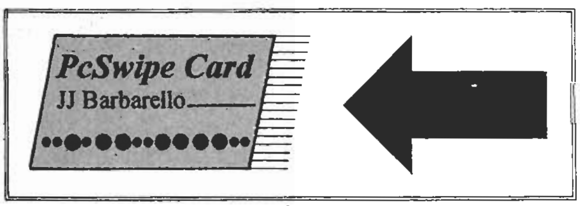
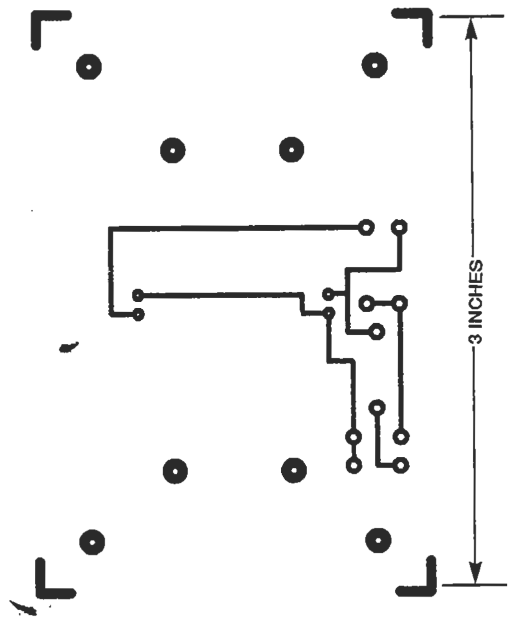




FIG. 7—LABEL THE SWIPE CARD (a). As an option you can add a directional label (b) to the base unit.




PcSwipe foil pattern.

(516-293-2283, V.32, V.42bis) as a text file called PCSWIPE.TXT.

**Construction**

The components can be mounted on a copper-clad perforated board or a PC board. A foil pattern for a suitable board is included in this article; the components should be placed as shown in Fig. 3.

Figure 4 is a side view of the board. Note that Q1 and LED1 are mounted opposite each other, one inch apart. Bend the leads of each device at a 90° angle, allowing a gap of about ⅛-inch between the two devices. The devices should be horizontal, with an imaginary centerline running through both at a point ½-inch above the board. Tack-solder each device to the board, check their alignment, then add solder to position them securely. Next mount R1, R2, and D2. Make sure you position all semiconductors with their pins in the correct holes.

Note that there are two sets of four mounting holes shown in the foil pattern and in Fig. 3. The four outer holes are for mounting the PC board to a base. The four inner holes are for the card guides. All eight holes can be drilled with a ⅛-inch diameter drill for a No. 4-40 machine screw or a No. 4 sheet-metal screw.

If you're anxious to try out your PcSwipe now, attach a battery or power supply to the board and the DB-25 connector to the LPT1 port of your PC. Run QBasic, and enter the test program shown in Listing 2. You should see the word LO in the center of the screen. Place a piece of paper between Q1 and LED1. The screen should now display HI, and you should hear a high-pitched beep. If you remove the paper, the word will return to LO, and you'll hear a low-pitched beep.

If you don't get the initial LO, check the power source, connections, and component placement. Correct any errors and try again. When you know the circuit is working, start the mechanical assembly.

**Mechanical assembly**

The swipe guides are two pieces of pine (or other wood) measuring 1 inch high by 3 inches long by ½ inch wide. As shown in Fig. 3, bevel the front inside edges of both guides to ease card insertion. Also, make a slot in each guide as shown in Figure 5-*a*.

Install one guide over LED1 and secure it from the bottom with appropriate hardware (No. 4 sheet-metal screws work well). Temporarily bend Q1 toward the PC board so that you can access the front of the diode. Obtain a piece of adhesive-backed opaque paper or similar material; a standard floppy-disk write-protect tab works fine. Cut a 1-mm slit in the tab as shown in Figure 5-*b*; this provides the LED aperture mentioned earlier. Center the tab in front of LED1, bend Q1 back into place, and install the remaining guide. Last, mount the PC board on a suitable base. I used a small pine block measuring about 2½ × 4½ inches.

**Making a swipe card**

The swipe card needs a row of holes located ½-inch up from the bottom. Space the holes 0.2-inch apart. The card material should be relatively rigid and no thicker than ⅛-inch. Form the holes cleanly.

The easiest way to make the card is to cut it from pre-punched perforated board or use it as a guide for drilling holes in other material; the holes on perforated board are spaced 0.1-inch apart. The board is usually 1⁄16-inch thick, and can be machined easily. Because the holes must be spaced 0.2 inch apart, block every other hole on the board with some substance opaque to infrared light. Household tub and tile caulking works fine.

Using Fig. 6 as a guide, cut a piece of perforated board to 3.4 inches wide (34 holes) by 2 inches high (20 holes). Make sure one row of holes is exactly ½-inch above the bottom edge of the card. Next, spread a small amount of caulking over the surface of the board, filling in all the holes. Remove excess caulking from both sides and let the board dry before continuing.

Starting from the left edge, and in the fifth row up, drill every other hole with a 1/16-inch drill bit, for a total of 16 holes. That provides the logic 0 holes; now drill logic 1 holes (⅛-inch) where desired. Remember that the leftmost hole is the reference; the next hole is $2^0$, the next hole is $2^1$, and so on. The program shown in Listing 3 allows you to enter a decimal number 0 to 32767, and it presents a visual indication of the holes to drill.

When all holes have been drilled, lightly sand both faces of the card to remove burrs and caulking residue. Finish the card by attaching an attractive label like that shown in Fig 7-*a*. Cut out and discard the shaded area before adhering the two remaining pieces of the label, one above and the other below the row of holes.

**Enhancements**

If you'd like to customize PcSwipe for a particular application, here are a few ideas you can try:

- Increase the guide length to 5 inches or more. The greater length will decrease the possibility that the card will rock in the guides, and it will minimize the possibility of erroneous readings.
- Add a power switch to extend battery life.
- Add a label to the outside of the guide to indicate the preferred direction for swiping. PcSwipe is bi-directional, in that the card can be swiped from either end, as long as the reference hole enters first. The label will help ensure that users insert their cards properly, at the beveled ends of the guides. A label can also help to minimize wear on both the guides and the card. A sample label suitable for mounting on the card reader is shown in Fig. 7-*b*.
- Using a separate bit of the parallel port, you can control a device such as a light or door lock. For example, the following QBasic code will energize bit 2 of the parallel port whenever ID 1946 is read:

```
If TTL = 1946 Then Out AD1-1, 1 Else Out
        AD1-1, 0
```

To use a bit from the parallel port as an output, build a buffer circuit with a transistor and a relay. Don't drive an external device directly.

That floppy-only 8088 PC sitting unused in your basement could serve well in this application. Compile the program so it can run efficiently, then add it to the AUTOEXEC.BAT file. Voila—an instant home-automation controller.

Here are a few rules of thumb if you'd like to vary the design of the cards:

- Absolute hole diameter is not critical.
- The diameter difference between the 0 and 1 holes should be in a ratio of at least 2:1 for the card to work properly.
- Accurate horizontal spacing of holes is not critical, as long as there is sufficient space to differentiate between a hole and a non-hole.
- Slits or other geometric shapes can be cut instead of holes, as long as they meet the 2:1 width ratio.
- The number of holes read by the software can be modified easily.

The software also offers many opportunities for enhancement. The disk offered by the author demonstrates many software enhancements. Here's an idea to get you started. Add a disk file that contains information related to the ID on each card. Each record in the file could contain name, address, and other useful data. The ID could serve as an index to the appropriate record, so that when a particular ID is read, the card holder's information can be displayed or printed. Ω