NIBBLE LIGHT PEN

by David Gauger, II

HARDWARE CONSTRUCTION PROJECT

Ten dollars worth of parts and a short machine language program are all you need to add this convenient accessory to your Apple. An Applesoft demonstration program illustrates techniques to use the light pen to select items from menus.

Most commercially available light pen systems use hardware to detect the location of the pen on the screen. This allows excellent resolution; some systems are even accurate to one pixel. They are also expensive, both in terms of hardware cost and system complexity, making them impractical for the average home user.

This article describes a light pen that is inexpensive and easy to construct. It uses just three active electronic parts that cost about $10 and an ordinary ballpoint pen case. Despite its simplicity, the Nibble Light Pen is reliable and provides an effective, direct way to interact with any of the Apple // family computers. Like the mouse, the Nibble Light Pen lets you bypass the keyboard. Just point the light pen at the screen — you don't even need to "click."

The Nibble Light Pen consists of two basic elements: the light-sensing hardware and a machine language driver routine. Briefly, when the light pen detects the light from an inverse block on the screen, it transmits a pulse to one of the lines in the Apple // game port. This signals the driver routine to determine the screen position and store the coordinates in memory. The calling program can then PEEK these memory locations to get the coordinates.

David Gauger, II, 1430 E. 55th Pl., Tulsa, OK, 74105. *Nibble Light Pen is compatible with DOS 3.3 and ProDOS.*

THE HARDWARE

Some hardware-based light pen systems use the time it takes for the monitor's electron beam to get from the bottom of the screen to the top to calculate the pen's location. Others interrupt the microprocessor when the pen detects the raster scan on a certain line. To keep the hardware simple, this system uses software to determine the pen's position. The hardware's sole function is to detect light.

The light pen is equipped with a photo-Darlington light detector. This is a sensitive but fairly inexpensive semiconductor that interfaces to the Apple // game port with the help of resistors.

There are three pushbutton inputs in the Apple // game port. Each one corresponds to a specific address in memory, and to a specific pin in the game port connector. When the light pen is pointed at a single inverse block, the photo-Darlington delivers about +1 volt to a pushbutton input. If the resulting voltage on that button's pin is +1 volt or more, a value of 128 (hex $80) or greater is stored in the corresponding memory address. To see if the button has been "pushed" by the light pen, the pen software reads the location (address) of the button. This is accomplished in BASIC by a PEEK statement.

Although it will work adequately without it, I added a variable resistor to the light pen. This component makes it possible to vary the light threshold level at which the photo-Darlington delivers +1 volt to the pushbutton input. It increases the circuit's sensitivity, allowing the pen to respond adequately to much dimmer light levels.

THE DRIVER

The software driver (**Listing 1**) is a short machine language routine that resides in memory page 3. It is designed to be used as a subroutine that can be CALLed by a machine language or BASIC program.

When CALLed, the driver searches every point on the screen for an inverse block. When it finds one, it determines whether this is the block at which the pen is pointing. If not, the routine looks for the next block. When the driver finds the correct block, it sounds a tone to notify the user, stores the vertical coordinate of the block in location 768 (hex $300), stores the horizontal coordinate in location 769 (hex $301), and returns to BASIC. From BASIC your main program can PEEK these two locations to find out where the pen is pointing.

CONSTRUCTION

To get the system up and running, con-

FIGURE 1: Schematic for Apple II, II Plus, and //e Light Pen (DIP Connector)
(Inset: Lead Diagram for Photo-Darlington [Sylvania ECG 3036 or Motorola MRD 360])

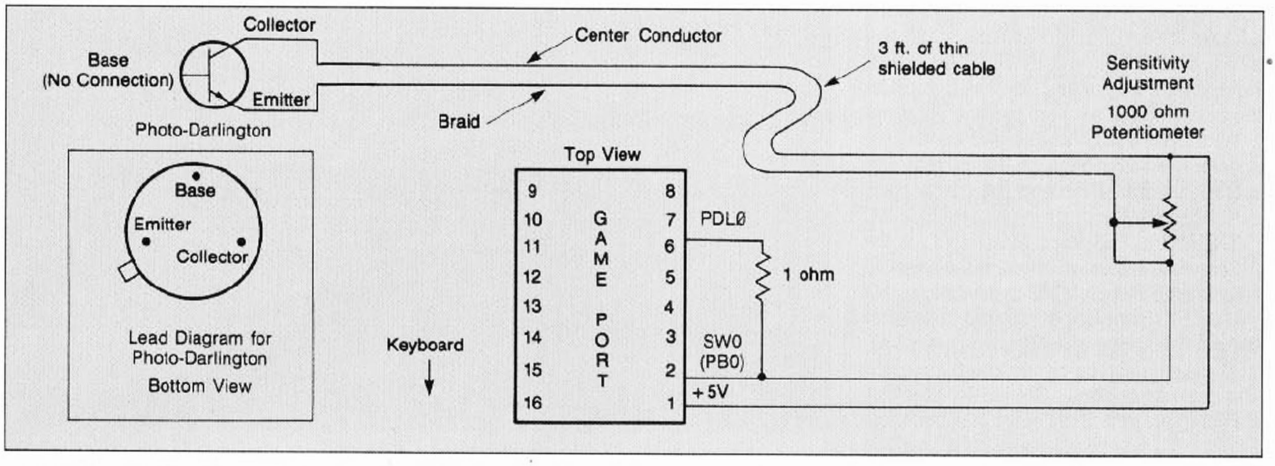
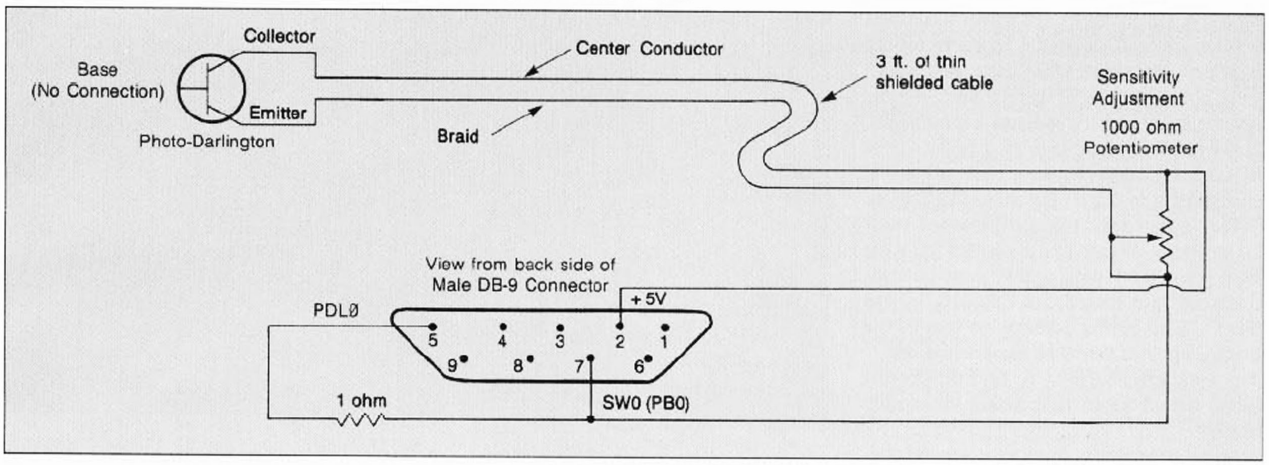


FIGURE 2: Schematic for Apple //c and //e Light Pen (DB Connector)



struct the light pen as shown in the diagram appropriate to your machine. If your computer uses a 16-pin DIP game socket (Apple II, II Plus or //e), then use the parts listed in **Table 1**, following the schematic diagram in **Figure 1**. Use **Figure 2** and the parts in **Table 2** if you have a //c or have equipped your machine with a DB-9 game socket. Next, type in the assembly listing (**Listing 1**), which is the same for all machines. If you do not have an assembler, consult the instructions in "A Welcome to New *Nibble* Readers" for help. Save the program to disk with the statement:

BSAVE PEN.DRIVER, A$302,L$8C

Any shielded cable may be used, but the more flexible it is, the easier the pen will be to manipulate. I used a disposable type of pen barrel to house the photo-Darlington, but anything that resembles a pen will do. Just be sure to feed the shielded cable down the center before you solder the light detector in place. Heat the leads of the photo-Darlington as little as possible, and mount it to provide close proximity to the screen (1/8 inch or less).

Any photo-Darlington may be used, provided it has adequate sensitivity for this application. You may have difficulty finding one though, since photo-Darlingtons are used almost exclusively in industrial applications, and electronics stores like Radio Shack do not carry them. An industrial electronics supply house should do the trick.

ADJUSTING THE LIGHT PEN'S SENSITIVITY

When the pen is constructed and the software has been loaded, use the following procedure to adjust the light pen's sensitivity. Set the brightness and contrast of the monitor to your liking. With the driver in memory, type in and run this short test program:

```
10 HOME
20 VTAB 10: HTAB 12
30 INVERSE
40 PRINT " ": NORMAL: REM 1 SPACE
50 CALL 770 : REM LIGHT PEN DRIVER
60 PRINT PEEK(768)
70 PRINT PEEK(769)
```

The Apple responds by placing an inverse space on the screen. Note that this square of light is probably blinking or flickering. If it is not, try turning the variable resistor to one end or the other of its range. If you still cannot get it to flicker, there is probably something wrong with the light pen or the driver software.

Turn the sensitivity adjustment to the end of its range so that the light blinks on and off very rapidly. The sensitivity is now at its lowest level. Turn the adjustment up until the light begins to flicker, then back it off slightly. The sensitivity is now at its highest, and the pen will respond to lower light levels from the screen. Touch the pen to the inverse space. The Apple should beep and then print a 9 above an 11. These are the coordinates of the block the driver found.

Set the sensitivity so that the light pen system locates the correct block every time. There should be a range of settings that make this possible. If the setting is too high or too low, the driver will sometimes return with the wrong block. The light pen system is now finished and ready to be used.

THE SOFTWARE

Because of the disjointed arrangement of text screen memory in the Apple, it is difficult to keep track of screen positions with simple row and column pointers. Fortunately, there is a built-in ROM routine called BASCALC that calculates the starting address of a screen row, given the number of

the row. Just place the row number in the Accumulator, execute BASCALC, and the address shows up in the addresses $28 and $29 (called BASL and BASH). An offset, representing the column position, completes the calculation of a screen memory location.

With BASCALC, it is possible to use two counters to keep track of the current text screen position. I use one counter to track the line number given to BASCALC (VCOUNT). The other, HCOUNT, is the offset value, which, in effect, is the horizontal counter. These are the two locations that you PEEK from BASIC to find out where the pen is pointing when the driver is called.

The Apple screen can accommodate 960 characters at one time. Obviously, checking for the light pen at all 960 character locations would be unnecessarily slow. A more efficient method would be to scan the screen for a pre-selected character. When scanning for a specific character, the driver would test perhaps 10 or 20 character locations for the light pen, instead of 960. Since the pen detects light, I chose to have it scan for the inverse space, which radiates a lot of light. However, it is possible to modify the driver to scan for any character that gives off enough light to trigger the photo-Darlington.

We've narrowed the scan down to 10 or 20 locations. Which one is pointed to by the light pen? In a typical application, the routine scans the screen and finds 10 inverse blocks. The pen is pointing to one of the blocks, but 9 of the 10 blocks are "wrong." The most efficient way to find the correct block would be to first detect the wrong blocks.

I used the process of elimination to identify the correct block. As long as the light pen points at an inverse square, it transmits +1 volt to the pushbutton input (PB0). To test a block, the routine replaces the inverse space with a regular, dark space, and it checks the voltage at the PB0. If the PB0 still holds +1 volt, the pen is not pointing at the darkened test block.

The "block off" test alone is not sufficient to positively identify the correct block. For instance, suppose the pen is pointing at the ground when the driver is called. In this case, since the pen detects no light, it does not transmit voltage to the PB0. However, when the driver tests the first block, it assumes that the light pen is pointing at the darkened test block. The solution is to add a second test: turn the test block back on by storing an inverse space there again, and check the PB0 to see if the light pen again transmits +1 volt.

These two tests usually locate the correct block, but they're not foolproof. To increase accuracy, I added another block off test. With adequate brightness and the sensitivity level set correctly, the reliability of these three tests approaches 100%.

When it does find the correct block, the driver replaces the space with an inverse block (the last test is a block off test) and sounds a two-pitch bell different from the Apple's bell. The horizontal and vertical counters already indicate the correct coordinates, so the driver returns to the calling program with an RTS.

If the driver does not find the correct block during the first scan down the screen, it returns to the top and scans again. The routine will return to the calling program only if it can find the correct block.

TABLE 1: Parts List for the Apple II, II Plus and //e DIP Connector

Item	Quantity	Source	Cost
Photo-Darlington	1	Industrial electronics supplier Sylvania #ECG 3036 or RCA #MRD 360	$4.75
1 kilohm potentiometer	1	Radio Shack (#271-227)	$0.59
1 ohm resistor	1	Industrial electronics supplier (Radio Shack does not stock)	$0.90
16-pin DIP connector	1	Radio Shack (#276-1980)	$1.69
Shielded cable	3 ft.	Radio Shack (#278-1277)	$2.39
Disposable pen (BIC Biro works well)	1		

TABLE 2: Parts List for the Apple //c and //e DB-9 Game Ports

Item	Quantity	Source	Cost
Photo-Darlington	1	Industrial electronics supplier Sylvania #ECG 3036 or RCA #MRD 360	$4.75
1 kilohm potentiometer	1	Radio Shack (#271-227)	$0.59
1 ohm resistor	1	Industrial electronics supplier (Radio Shack does not stock)	$0.90
Male DB-9 connector	1	Radio Shack (#276-1537)	$1.99
Shielded cable	3 ft.	Radio Shack (#278-1277)	$2.39
Disposable pen (BIC Biro works well)	1		

INSTALLING THE SYSTEM

To incorporate the Nibble Light Pen system into your own program, first place an inverse block or blocks on the screen. In BASIC, the code might look like this:

```
10 VTAB 10: HTAB 12: INVERSE:
   PRINT " ": NORMAL
```

Next, CALL 770 (hex $302) which is the driver, and touch the light pen to any inverse block on the screen. When the software has found the spot you're pointing at (it only takes a fraction of a second), you will hear a beep. Locations 768 and 769 now contain the vertical and horizontal coordinates of the inverse block you indicated.

It is important to note that the horizontal and vertical counters start counting at zero.

LISTING 1: PEN.DRIVER

```
1    ********************
2    *     PEN.DRIVER     *
3    * BY DAVID GAUGER II *
4    * COPYRIGHT (C) 1986 *
5    * BY MICROSPARC, INC *
6    * CONCORD, MA  01742 *
7    ********************
8    *
9    * MERLIN ASSEMBLER
10
```

The vertical counter ranges from 0-23, and the horizontal, from 0-39. If the driver returns with a vertical value of 16 and a horizontal value of 20, this is the same spot on the screen as one defined by VTAB 17 and HTAB 21.

The pen's resolution is partly determined by the photo-detector. I have used photo-Darlingtons that can resolve one block of light in two. This means that an inverse block cannot be placed immediately adjacent on any side. The photo-Darlington can adequately resolve light blocks immediately diagonal to one another fairly well. Should you want to place a block on every line or column, the blocks must be placed diagonally in zigzag fashion in order for the pen to discern them.

APPLICATIONS

Your imagination is the only limit to your applications of the Nibble Light Pen. For example, a typical instruction screen might have the prompt at the bottom: "Press <RETURN> to continue." Instead of using an INPUT or GET statement to control program flow at this point, place one inverse space on the screen labeled "Touch pen here to continue." Then call the light pen driver. The driver will not return to the calling program until it finds the block at which the pen is pointing. The result is effective program control without using the keyboard.

Another obvious use is in menu selection. In any computer magazine you'll find plenty of advice on menu input, how to organize menu screens, and the error trapping that inevitably accompanies keyboard input. Using a light pen with a menu eliminates many of these problems.

I have written a simple demonstration program to illustrate these two applications. Simply type in **Listing 2**, and save it on a disk that already contains the light pen driver (PEN.DRIVER, **Listing 1**) with the command:

SAVE LIGHT.PEN.DEMO

This demonstration program uses the light pen in the most simple and elementary ways; it's just meant to get you started. Other applications could include graphics, data input, screen layout and games. It may also be an ideal input device for people with certain handicaps, and it is flexible enough to be incorporated into just about any program or language.

MODIFICATIONS

Here are a few ideas for changes you can make. The routine can be changed to scan the screen for characters other than the inverse block. It could also be adapted to scan for a range of characters, such as the entire inverse character set or just inverse numbers. I have found the system sensitive enough to detect regular video characters,

```
                        11
                        12  * THIS PROGRAM EXPECTS A PHOTO-DARLINGTON TO BE CONNECTED
                        13  * FROM PIN 1 TO PIN 2 OF THE GAME PORT.
                        14  * INPUT IS READ FROM PB0 (PIN 2): ADDRESS $C061 (-16287)
                        15  * ADDITIONALLY, A 1 OHM RESISTOR IS NEEDED FROM PIN 2
                        16  * TO PIN 6 OF THE GAME PORT.
                        17
                        18
                        19          ORG   $302
                        20  PEN     EQU   $C061
                        21  VCOUNT  EQU   $300
                        22  HCOUNT  EQU   $301
                        23  BASCALC EQU   $FBC1
                        24  BASL    EQU   $28
                        25  WAIT    EQU   $FCA8
                        26  SPEAKER EQU   $C030
                        27  BELL    EQU   $FBDD
                        28
0302: A0 00             29  SETUP   LDY   #$00
0304: 8C 00 03          30          STY   VCOUNT
0307: A9 00             31          LDA   #$00
0309: 20 C1 FB          32          JSR   BASCALC     ;CALC ADDRESS OF FIRST LINE
                        33
                        34
030C: B1 28             35  MAINLOOP LDA  (BASL),Y   ;GET CHARACTER
030E: C9 20             36          CMP   #$20        ;IS IT A BLOCK?
0310: F0 08             37          BEQ   PENTEST     ;YES -- TEST PEN
0312: C8                38  NEXTSPOT INY             ;NO
0313: C0 28             39          CPY   #$28        ;END OF LINE?
0315: F0 2F             40          BEQ   NEWLINE     ;YES
0317: 4C 0C 03          41          JMP   MAINLOOP    ;NO - DO IT ALL AGAIN
                        42
                        43
031A: A9 A0             44  PENTEST LDA  #$A0
031C: 91 28             45  TEST1   STA   (BASL),Y   ;TURN OFF BLOCK
031E: 20 58 03          46          JSR   DETECT      ;ANY LIGHT?
0321: 90 07             47          BCC   TEST2       ;NO - TEST PASSES
0323: A9 20             48          LDA   #$20        ;YES - TEST FAILS
0325: 91 28             49          STA   (BASL),Y    ;WHERE YOU GOT IT
0327: 4C 12 03          50          JMP   NEXTSPOT    ;AND TRY NEXT SPOT
032A: A9 20             51  TEST2   LDA   #$20
032C: 91 28             52          STA   (BASL),Y    ;PUT BLOCK BACK
032E: 20 58 03          53          JSR   DETECT      ;ANY LIGHT?
0331: B0 03             54          BCS   TEST3       ;YES - TEST PASSES
0333: 4C 12 03          55          JMP   NEXTSPOT    ;NO - TRY NEXT SPOT
0336: A9 A0             56  TEST3   LDA   #$A0
0338: 91 28             57          STA   (BASL),Y    ;TURN BLOCK BACK OFF
033A: 20 58 03          58          JSR   DETECT      ;ANY LIGHT?
033D: 90 43             59          BCC   EXIT        ;NO - TEST PASSES - BLOCK FOUND
033F: A9 20             60          LDA   #$20
0341: 91 28             61          STA   (BASL),Y    ;SO PUT BLOCK BACK
0343: 4C 12 03          62          JMP   NEXTSPOT    ;AND TRY AGAIN
                        63
0346: EE 00 03          64  NEWLINE INC   VCOUNT      ;INCREMENT VERT COUNTER
0349: AD 00 03          65          LDA   VCOUNT
034C: C9 18             66          CMP   #$18        ;IS IT MORE THAN THE 24TH LINE?
034E: F0 B2             67          BEQ   SETUP       ;YES - TIME TO START AT THE TOP
0350: 20 C1 FB          68          JSR   BASCALC     ;NO - FIGURE NEW BASE ADDRESS
0353: A0 00             69          LDY   #$00        ;ZERO HORIZONTAL COUNTER
0355: 4C 0C 03          70          JMP   MAINLOOP    ;AND DO IT ALL AGAIN
                        71
0358: 8C 01 03          72  DETECT  STY   HCOUNT      ;SAVE HORIZONTAL COUNTER
035B: A2 08             73          LDX   #$08        ;SET UP COUNTER
035D: A0 00             74          LDY   #$00        ;LOAD Y COUNTER
035F: AD 61 C0          75  GETPEN  LDA   PEN         ;SEE IF PEN SAW LIGHT
0362: 30 0B             76          BMI   YESLITE
0364: 88                77          DEY
0365: D0 F8             78          BNE   GETPEN      ;THIS LOOP TESTS THE PEN
0367: CA                79          DEX               ;MANY TIMES
0368: D0 F5             80          BNE   GETPEN
036A: 18                81          CLC               ;CLEAR CARRY: NO LIGHT
036B: AC 01 03          82          LDY   HCOUNT      ;RESTORE HORIZONTAL COUNTER
036E: 60                83          RTS
036F: 38                84  YESLITE  SEC               ;SET CARRY: YES LIGHT
0370: AC 01 03          85          LDY   HCOUNT      ;RESTORE HORIZONTAL COUNTER
0373: 60                86          RTS
                        87
0374: A0 C0             88  BELL2   LDY   #$C0        ;LENGTH OF BELL2
0376: A9 08             89  LOOP    LDA   #$08        ;PITCH OF BELL2
0378: 20 A8 FC          90          JSR   WAIT
037B: AD 30 C0          91          LDA   SPEAKER     ;CLICK SPEAKER
037E: 88                92          DEY               ;DECREMENT COUNTER
037F: D0 F5             93          BNE   LOOP        ;DO IT ALL AGAIN
0381: 60                94          RTS
                        95
0382: 18                96  EXIT    CLC               ;FOR SAFETY
0383: A9 20             97          LDA   #$20
0385: 91 28             98          STA   (BASL),Y    ;WHERE YOU GOT IT
0387: 20 DD FB          99          JSR   BELL        ;REGULAR BELL
038A: 20 74 03         100          JSR   BELL2       ;DIFFERENT BELL FOR FUN
038D: 60               101          RTS

--End assembly--

140 bytes

Errors: 0
```

END OF LISTING 1

although the period and comma present some difficulty.

You may find that from time to time the driver makes a mistake and returns with the coordinates of a block to which you are not pointing. In writing the driver I sacrificed a bit of accuracy in favor of speed. The portion of code that actually detects light from the pen, the DETECT subroutine, is the culprit. Specifically, **lines 74 and 75** are counters that form loops to check the pen for light 2,048 times, each time the subroutine is called. Obviously, this takes a bit of time even at machine language speed, but the more times you check the pen for light, the more accurate your results will be. If you require more accuracy and are willing to forego execution speed, I suggest that you LDX with 0A in **line 74**. If you're assembling the driver at hex $302, this means that you store 0A at location hex $35C.

Resolution could perhaps be improved by changing the photo-Darlington, arranging a tube to narrow the area of light it responds to, or substituting another photo-detector such as a photo-diode or light-dependent resistor. Another idea is to use the low resolution graphics mode to scan for the pen. Theoretically, this should double the pen's resolution. Also, there is no reason why another pushbutton input could not be used. Using PB2 would allow you to use the paddles or a joystick simultaneously with the light pen (this is not feasible on the //c).

LISTING 2: LIGHT.PEN.DEMO

```
1   REM ************************
2   REM * LIGHT PEN DEMO       *
3   REM * BY DAVID GAUGER II   *
4   REM * COPYRIGHT (C) 1986   *
5   REM * BY MICROSPARC, INC   *
6   REM * CONCORD, MA  01742   *
7   REM ************************
100  REM  *** INITIALIZE ***
110  GOSUB 190
120  PRINT  CHR$ (4);"BLOAD PEN.DRIVER"
130 PEN = 770: REM    LOCATION OF  MACHINE LA
    NGUAGE DRIVER
140 V = 768:H = 769: REM          VERTICAL (CO
    LUMN) AND         HORIZONTAL (ROW) COORDI
    NATE  LOCATIONS
150  GOTO 210
160  REM  *** SUBROUTINES ***
170  INVERSE : PRINT " ";: NORMAL : RETURN : REM
      SUBROUTINE  TO PRINT AN INVERSE SPACE
      (BLOCK) ON THE SCREEN
180  PRINT " ";: RETURN : REM      SUBROUTINE
    TO PRINT A SPACE
190  TEXT : HOME : RETURN
200  REM  *** MAIN PROGRAM ***
210  HOME : HTAB 5: PRINT "NIBBLE LIGHT PEN D
    EMONSTRATION": PRINT "** COPYRIGHT 1986
    BY MICROSPARC, INC.**"
220  VTAB 8
230  PRINT "THIS PROGRAM IS MEANT TO DEMONSTR
    ATE"
240  PRINT "ONE POSSIBLE USE FOR THE NIBBLE L
    IGHT"
250  PRINT "PEN IN YOUR OWN PROGRAMS."
260  PRINT : PRINT
270  PRINT
280  PRINT "PLEASE NOTE THAT THE PROGRAM IS N
    OW"
290  PRINT "WAITING FOR YOU TO TOUCH THE INVE
    RSE"
300  PRINT "SPACE WITH THE LIGHT PEN."
310  VTAB 23: HTAB 5
320  PRINT "TOUCH PEN HERE TO CONTINUE ==>";
330  GOSUB 170: REM  PRINT          INVERSE SPA
    CE
340  CALL PEN: REM     PROGRAM       CONTROL PA
    SSED TO LIGHT      PEN DRIVER WHICH RETUR
    NS     ONLY WHEN IT FINDS PEN
350  REM  *** MENU SCREEN ***
360  GOSUB 190: HTAB 5: PRINT "NIBBLE LIGHT P
    EN DEMONSTRATION"
370  VTAB 8: HTAB 12: GOSUB 170: GOSUB 180
380  PRINT "CATALOG DISK"
390  VTAB 11: HTAB 12: GOSUB 170: GOSUB 180
400  PRINT "RING BELL"
410  VTAB 14: HTAB 12: GOSUB 170: GOSUB 180
420  PRINT "LIST PROGRAM"
430  VTAB 17: HTAB 12: GOSUB 170: GOSUB 180
440  PRINT "END"
450  CALL PEN
460  REM  * LIGHT PEN DECODING *
470  IF  PEEK (V) = 7 THEN  PRINT  CHR$ (4);"
    CATALOG": FOR X = 1 TO 2000: NEXT X: GOTO
    360
480  REM  NOTE THAT THE DRIVER   RETURNS (V)
    ONE LESS THAN      THE VTAB VALUE OF THE
    SAME      SPOT
490  REM   NOTE ALSO THAT WE DONT HAVE TO DEC
    ODE THE HORIZ.  VALUE IN THIS CASE BECAU
    SE   THE VERTICAL VALUE ALONE    IS ENOUG
    H
500  IF  PEEK (V) = 10 THEN  PRINT  CHR$ (7) +
     CHR$ (7) +  CHR$ (7) +  CHR$ (7) +  CHR$
    (7);: GOTO 360: REM  5 BELLS
510  IF  PEEK (V) = 13 THEN  HOME : LIST : HOME
    : GOTO 360
520  REM     IF (V)=16 THEN THE     PROGRAM WIL
    L FALL THROUGH     TO THIS POINT
530  END
```

END OF LISTING 2