FORREST M. MIMS III

THE COMPUTER SCIENTIST

COMPUTER CONTROLLED LIGHT METER

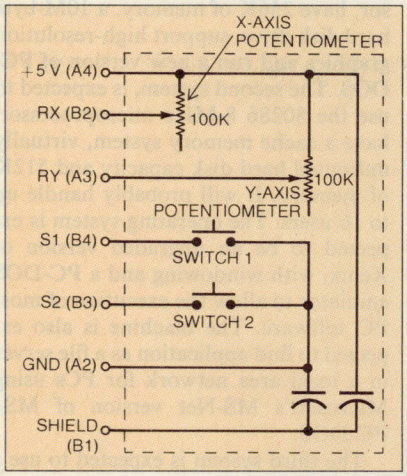


Fig. 1. PC*jr* joystick circuit.

PERSONAL computers equipped with analog joystick ports provide the potential for a host of applications never imagined by their designers. That's because the analog-to-digital conversion hardware and software required to support a pair of joysticks can be easily interfaced with a great many external sensing devices.

A good example is IBM's PC*jr*. This machine includes a pair of analog joystick ports ideal for use with sensors whose resistance varies with an applied stimulus such as temperature, light, heat or pressure. Up to four such sensors can be connected *directly* to the PC*jr*'s joystick ports with no buffering or interfacing hardware.

In this column we'll take a close look at PC*jr*'s joystick ports and use one of them to convert the Junior into both a simple light meter and a sophisticated, light-sensing storage oscilloscope. Even if you don't have access to a PC*jr*, read on. Some of the methods to be described apply equally well to the Apple IIe, Radio Shack's TRS-80 Color Computer and other machines with analog joystick ports.

Junior's Joysticks

Junior's *Attachable Joystick* (as IBM calls it) is a Made-in-Japan $40 special with both spring-return-to-center and free-floating modes. Two small plastic levers on the bottom of the joystick housing allow the spring-return feature to be selected independently for each axis. Thumbwheels on the top of the Joystick allow each axis to be mechanically centered when the spring-return feature is selected.

Lacking technical documentation, the quickest way to decipher a computer's joystick interface is simply to disassemble one of its joysticks. Junior's joysticks can be easily opened by removing a pair of self-tapping screws from the bottom side of the plastic enclosure.

Figure 1 shows the circuitry inside one of Junior's joysticks. The two potentiometers, one for the x-axis and the other for the y-axis, both have a linear taper resistance of 100,000 ohms. Two normally open, single-pole pushbutton "fire" switches are included.

Incidentally, be sure to note the *shield* connection (B1) in Fig. 1. The joystick cable is shielded to protect Junior's analog-to-digital (A/D) hardware from external noise. While experimenting with the joystick port, I soon learned the significance of the shield provision.

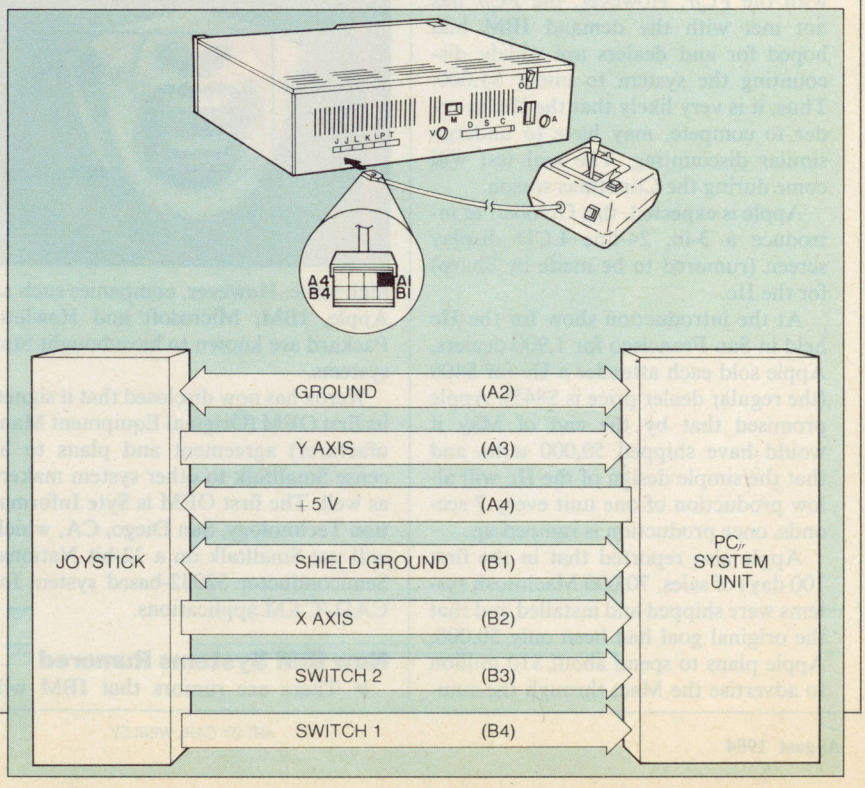
Figure 2 details the cable connections between a single joystick and one of Junior's system board joystick sockets. The most notable features are the unusual Berg-type connectors and, because of its shielding and its six wires, the rather thick (1/4-inch) and inflexible cable.

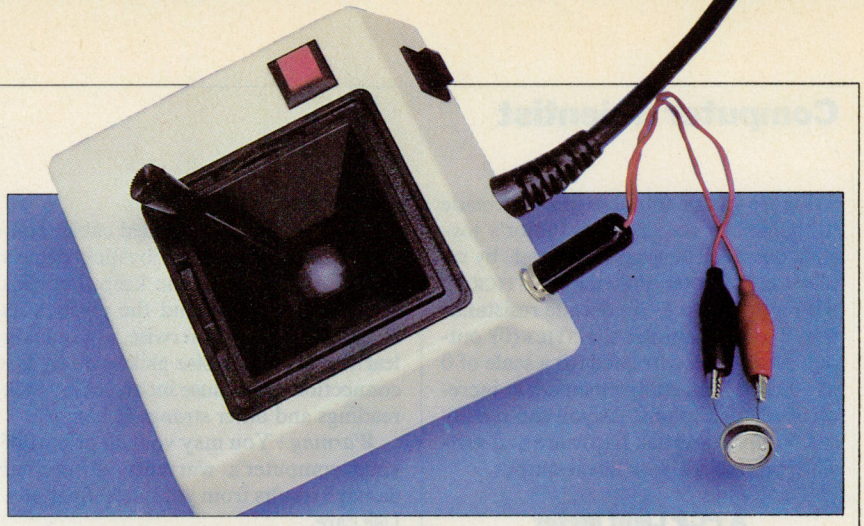
The Hardware-Software Connection

As I've noted before, Junior's version of BASIC is an expanded version of the BASIC supplied with Radio Shack's CoCo. Though both machines have analog joystick ports, however, their operation is quite different.

CoCo's 100K joystick potentiometers are connected across +5 volts and ground. Therefore, each potentiometer functions as a voltage divider that delivers to the joystick port a voltage ranging from about 0.25 to 4.75 V. This voltage is applied to a simple 6-bit firmware-driven A/D converter designed around

Fig. 2. Connections between the joystick and PC*jr*.





Adding a phone jack to joystick permits using a CdS sensor.

an analog comparator and a resistor network. For more details, see "Analog Sensors for Personal Computers" (COMPUTERS & ELECTRONICS, February 1984).

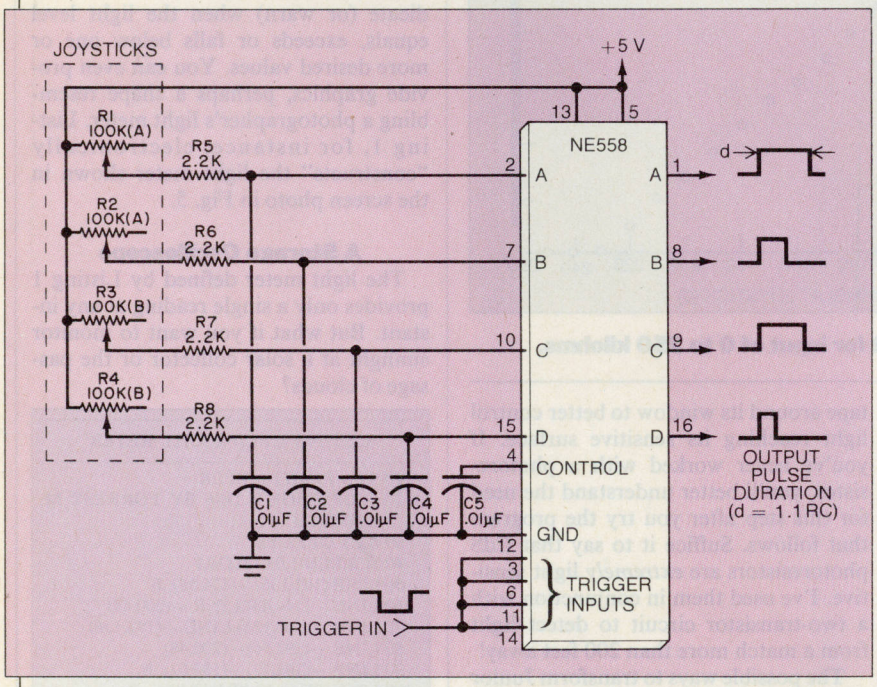
Though Junior's joysticks use the same 100K potentiometers as CoCo, their operating principle is entirely different. As you can see by referring back to Fig. 1, both potentiometers function merely as variable resistors and *not* as voltage dividers. In other words, while CoCo's joystick signal is a variable *voltage,* Junior's is a variable *resistance.*

Figure 3 shows the key ingredient of Junior's joystick hardware, a 558 quad timer IC. This chip includes four timer circuits, each of which is connected to a fixed capacitor *(C1-C4)* and a variable resistor (joystick potentiometers *R1-R4).* When triggered, each timer delivers an output pulse whose duration is approximately 1.1 × RC.

Junior's on-board ROM firmware includes a routine that measures the duration of each timer's output pulse. The resulting time is outputted as a number from about 3 to 255 when BASIC requests a joystick reading.

The lower joystick value would not necessarily extend to 0 if series resistors *R5-R8* were bypassed. These resistors are necessary to ensure reliable operation of the timers. Without the series resistors, at maximum handle excursions the resistance of the joystick potentiometers might be too low to allow the timers to function in the proper manner.

Fig. 3. Simplified input circuit for the PC*jr*.



Raw joystick values are returned by the STICK(*n*) function. Normally the retrieved values are assigned to a variable as in A=STICK(*n*). Here are the potentiometer assignments for each allowed value of *n*:

STICK(0) returns x coordinate for Joystick A.

STICK(1) returns y coordinate for Joystick A.

STICK(2) returns x coordinate for Joystick B.

STICK(3) returns y coordinate for Joystick B.

It's important to understand that STICK(0) retrieves the values for *all* four joysticks. Therefore, it's necessary to include in a program a STICK(0) function before attempting to read any of the remaining joystick values. STICK(1), STICK(2) and STICK(3) simply get the values retrieved by STICK(0). They do *not* sample the joysticks.

CoCo's joysticks always return a value of 0 to 63. Though Junior's joysticks provide better resolution, their numeric range isn't given by IBM. Instead, as Junior's BASIC reference manual observes, "The range of values for x and y depends on your particular joysticks."

As I mentioned above, Junior can return a *potential* maximum joystick range of from about 3 to 255. Maximum excursions of the IBM joystick I purchased output values of from 3 to 124. You can enter and run this routine to find the values for *your* joystick(s):

```
10 'STICKOUT
20 CLS
30 X=STICK(0)
40 Y=STICK(1)
50 LOCATE 15,15
60 PRINT "X="X;"Y="Y
70 GOTO 30
```

It's possible to take full advantage of Junior's entire range of joystick values when you replace the potentiometers with external sensors. This routine converts the 3-255 range of STICK(0) to a more convenient 0-100 range and prints the result:

```
10 'MAX STICK (0-100)
20 CLS
30 X=STICK(0)
40 X=(X/255)*100
50 LOCATE 15,15
60 PRINT "STICK(0):"
   INT(X)
70 GOTO30
```

Line 40 converts the retrieved value of STICK(0) (0-255) into a range of 0-100. However, the resistance of many sensors suitable for use with the PC*jr decreases* as the stimulus level *increases.* This line will reverse or give the complement of the previous result: 45 X=100−X.

Figure 4 is a calibration graph I made to gauge the accuracy of Junior's joystick output. The graph was made by replacing the potentiometer for STICK(0) with a precision (1%) decade resistance box. I then recorded the STICK(0) output, which was corrected for a scale of 0 to 100 and then complemented, at increments of 10 kilohms. As you can readily see, Junior's joystick hardware and software produces a very linear output.

A PC*jr* Light Meter

Among the simplest sensors that can be connected directly to PC*jr*'s joystick ports is the cadmium sulfide (CdS) photoresistor. One I've used with excellent results is Radio Shack's catalog number 276-116 CdS Photocell. The resistance of this cell is about 100 ohms in bright light and several million ohms in total darkness.

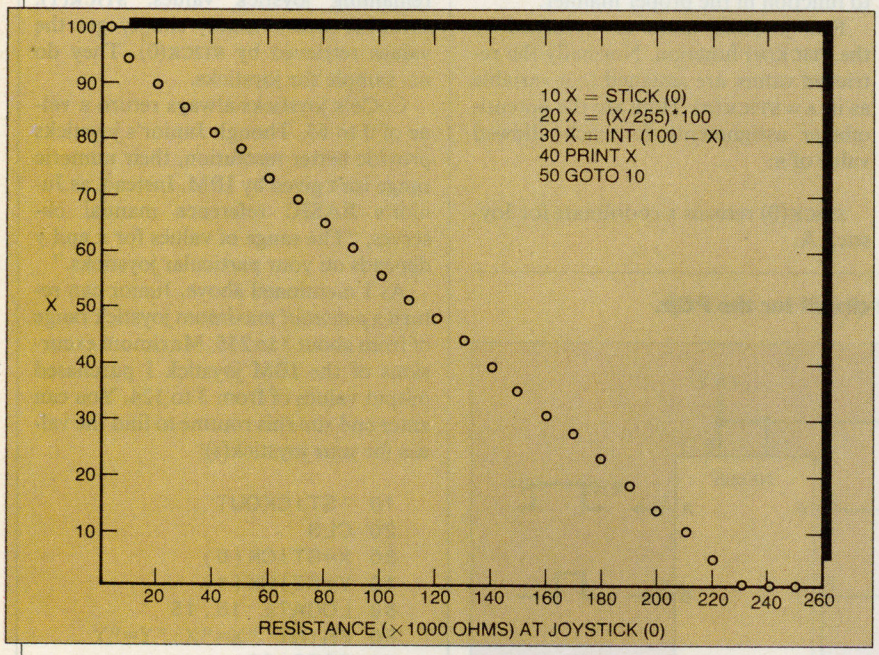


Fig. 4. Corrected joystick output for input of 0 to 250 kilohms.

The simplest way to connect a CdS cell to Junior is to open a joystick housing and remove the wires connected to the x-axis potentiometer (the one opposite the cable entry point). Carefully remove half an inch of insulation from each lead and wrap one of the exposed wires around each of the photoresistor's leads.

If you intend to do lots of experimenting with your joystick ports, you'll be better off installing a miniature phone jack for each potentiomter. Use the kind with a built-in switch connection. There's plenty of room, and your joystick will function normally until you plug a sensor into the jack.

Unless you use a shielded cable (connect the shield to the joystick cable's shield), it's important to keep the leads between the sensor and the joystick as short as possible. Otherwise, as I quickly learned, external noise picked up by the connections will cause incorrect joystick readings and other strange behavior.

Warning : You may void all or part of your computer's warranty if internal damage results from your modifications. Use care.

As for the photoresistor, it's a good idea to wrap a cylinder of black electrical tape around its window to better control light reaching its sensitive surface. If you've never worked with a photoresistor, you'll better understand the need for this step after you try the program that follows. Suffice it to say that CdS photoresistors are *extremely* light sensitive. I've used them in conjunction with a two-transistor circuit to detect light from a match more than 200 feet away!

The possible ways to transform Junior into a light meter are limited only by your imagination. For starters, try this:

```
10 'PCjr LIGHT METER
20 CLS
30 X=STICK(0)
40 X=(X/255)*100
50 X=INT(100-X)
60 LOCATE 2,2
70 PRINT "LIGHT
   LEVEL:",X
80 GOTO 30
```

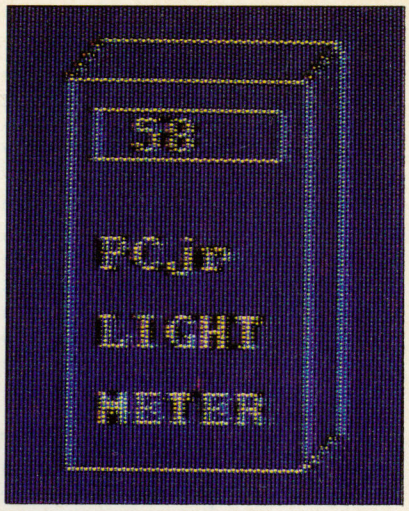


Fig. 5. Simulated light meter.

This routine simply provides on a scale of 0 to 100 the level of light striking the photoresistor. Of course you can add many frills to the program. For example, you can add a correction factor that compensates for the inherent nonlinearity of photoresistive devices. You can also add various beeps and tones that indicate (or warn) when the light level equals, exceeds or falls below one or more desired values. You can even provide graphics, perhaps a shape resembling a photographer's light meter. Listing 1, for instance, electronically "constructs" the light meter shown in the screen photo in Fig. 5.

A Storage Oscilloscope

The light meter defined by Listing 1 provides only a single reading at any instant. But what if you want to monitor sunlight at a solar collector or the passage of clouds?

LISTING 1. PC*jr* LIGHT METER

```
10 'PCjr LIGHT METER
20 'COPYRIGHT 1984 BY FORREST M.
   MIMS III
30 KEY OFF:CLS
40 SCREEN 1,0:COLOR1
50 LINE (192,136)-(136,56),,B
60 LINE -(144,48):LINE -(200,48)
70 LINE -(200,128):LINE -(192,136)
80 LINE -(192,56)-(200,48)
90 LINE -(142,72)-(186,62),,B
100 LOCATE 12,19:PRINT "PCjr"
110 LOCATE 14,19:PRINT "LIGHT"
120 LOCATE 16,19:PRINT "METER"
130 X=STICK(0)
140 X=(X/255)*100
150 X=INT(100-X)
160 LOCATE 9,19:PRINT X
170 GOTO 130
```

There are various ways to have Junior keep a record of readings for later retrieval and study. The simplest is to save on disk a list of readings under an appro-

(Continued on page 76)

Computer Scientist

(Continued from page 22)

priate file name. The program in Listing 2 does this and more as you will see from the details below.

Though Listing 2 is designed to run on the PC*jr*, with modification it will work on many other machines with analog joystick ports. The most compatible machines are those such as Radio Shack's Color Computer that use Microsoft BASIC.

LISTING 2. PC*jr* STORAGE LIGHT METER AND OSCILLOSCOPE

```
10 'PCjr DATA STORAGE LIGHT METER AND OSCILLOSCOPE
20 'COPYRIGHT 1984 BY FORREST M. MIMS III
30 CLS:SCREEN 1,0:COLOR 1:KEY OFF
40 PRINT " PCjr DATA STORAGE LIGHT METER"
50 LOCATE 10,1:PRINT "PRESS 1 TO SELECT STORAGE MODE."
60 PRINT:PRINT "PRESS 2 TO RETRIEVE DATA."
70 PRINT:PRINT "PRESS Q TO QUIT."
80 V$=INKEY$:IF V$="1" THEN 110
90 IF V$="Q" OR V$="q" THEN CLS:END
100 IF V$="2" THEN 440 ELSE 80
110 LOCATE 16,1:PRINT "CONNECT Cds PHOTOCELL TO STICK(0)."
120 LOCATE 18,1:PRINT "LIGHT METER READINGS WILL BE"
130 PRINT "SAVED ON DISK. SELECT A FILE NAME"
140 INPUT "AND ENTER IT NOW: ", LITE$:PRINT
150 IF LITE$="" THEN 120
160 INPUT "ENTER NUMBER OF SAMPLES: ", N
170 INPUT "ENTER SECONDS BETWEEN SAMPLES: ",S
180 OPEN LITE$ FOR OUTPUT AS #1
190 PRINT:PRINT "PRESS ANY KEY TO BEGIN."
200 IF INKEY$="" THEN 200
210 T$=TIME$:D$=DATE$
220 WRITE #1, D$,T$,S:'SAVE DATE, TIME & SAMPLE INTERVAL
230 CLS:PRINT
240 PRINT "SAMPLE LIGHT LEVEL (0 TO 100)"
250 C=1:Z=1:PRINT
260 ON TIMER(S) GOSUB 290:'INITIALIZE TIMER
270 TIMER ON
280 GOTO 280
290 'LIGHT MEASUREMENT SUBROUTINE
300 Y=STICK(0): 'RETRIEVE PHOTOCELL VALUE
310 Y=INT((Y/255)*100):Y=100-Y
320 PRINT:PRINT "  ";C;
    "  ";Y
330 WRITE #1,Y:BEEP: 'SAVE SAMPLE ON DISK
340 C=C+1:IF C=N+1 THEN 360
350 RETURN
360 CLOSE#1:PRINT
370 BEEP:FOR I=1 TO 200: NEXT I:BEEP
380 PRINT "MEASUREMENT CYCLE COMPLETE."
390 PRINT:PRINT "PRESS D TO RETRIEVE DATA.":PRINT
400 PRINT "PRESS R TO RERUN PROGRAM."
410 O$=INKEY$:IF O$="R" OR O$="r." THEN 30
420 IF O$="D" OR O$="d" THEN 440 ELSE 410
430 CLS:Z=1
440 'RETRIEVE DATA FROM DISK
450 PRINT:PRINT
460 ON ERROR GOTO 960: 'CATCH FILE ERROR
470 INPUT "ENTER FILE NAME: ", LITE$:CLS
480 IF LITE$="" THEN 470
490 OPEN LITE$ FOR INPUT AS # 1
500 INPUT #1,D$,T$,S:CLS
510 PRINT:PRINT "PRESS 1 TO VIEW DATA LIST.":PRINT
520 PRINT "PRESS 2 TO PLOT FIRST 13 DATA SAMPLES"
530 PRINT" ON SCOPE GRATICULE."
540 LOCATE 12,1:PRINT "AFTER DATA LIST OR TRACE HAS BEEN"
550 PRINT "DISPLAYED, PRESS R TO RERUN PROGRAM."
560 A$=INKEY$:IF A$="1" THEN 580
570 IF A$="2" THEN 690 ELSE 560
580 'PRINT DATA LIST ON MONITOR
590 CLS:PRINT "TEST:"; LITE$;"  ";D$;
    "  ";T$
600 PRINT "SAMPLE INTERVAL:"; S; "SECONDS" :PRINT
610 PRINT "SAMPLE LIGHT LEVEL (0 TO 100)" :PRINT
620 INPUT #1,Y: 'RETRIEVE DATA FROM DISK
630 PRINT:PRINT "  ";Z;"  ";Y
640 Z=Z+1
650 IF EOF (1) THEN 670
660 GOTO 620
670 BEEP:CLOSE#1
680 K$=INKEY$:IF K$="R" OR K$="r" THEN 30 ELSE 680
690 'PLOT DATA ON GRATICULE
700 GOSUB 780
710 FOR X=26 TO 314 STEP 24
720 INPUT #1,Y:Y=(Y*.01)*160:Y=160-Y
730 LINE -(X,Y): 'DRAW TRACE SEGMENT
740 IF EOF(1) THEN 760
750 NEXT X
760 CLOSE #1:BEEP
770 K$=INKEY$:IF K$="R" OR K$="r" THEN 30 ELSE 770
780 'GRATICULE SUBROUTINE
790 CLS:J=0
800 LOCATE 25,2:PRINT "SECS/DIV:"; S;" "; T$;"  ";D$
810 LINE (314,8)-(26,168),,B
820 FOR C=26 TO 314 STEP 24
830 LINE (C,8)-(C,168),,,&HCCCC
840 NEXT C
850 FOR R=8 TO 168 STEP 16
860 LINE (26,R)-(314,R),,,&HCCCC
870 NEXT R
880 LOCATE 23,4
890 PRINT "0 1 2 3 4 5 6 7 8 9 10 11 12"
900 FOR L=19 TO 3 STEP-2
910 LOCATE L,1
920 J=J+1:PRINT J
930 NEXT L
940 PSET (26,150): 'MOVE CURSOR TO LEFT GRATICULE BORDER
950 RETURN
960 IF ERR=53 THEN PRINT "NO SUCH FILE. TRY AGAIN.":PRINT:RESUME 470
```

Listing 2 is menu driven so you can select any of its principle operating modes. When the program is entered and has been run, the first menu allows you to begin one or more light meter readings (line 50), retrieve from disk a previous series of readings (line 60) or exit the program (line 70).

If you select the light meter option, the program asks you to enter a file name for the test session (110-140), the number of samples you wish to make (160) and the interval (in seconds) between samples (170). After the program opens a disk communications file (180), the entire test session, including the date, time, sample interval and the light meter reading, is automatically saved on disk (see 210-220, etc.).

If you select the program's data retrieval option, the program firsts asks for the file name (470) and then whether you wish to see a list of the stored data samples (510) or a graph of the first thirteen samples superimposed on a simulated oscilloscope graticule (520). In any case, you can return to the main menu to rerun the program by pressing R (400-410).

Let's review a typical measurement session. Say we've asked Junior to make five light readings separated by intervals of sixty seconds. Line 240 prints a test heading on the monitor and lines 260-280 actuate Junior's internal timer.

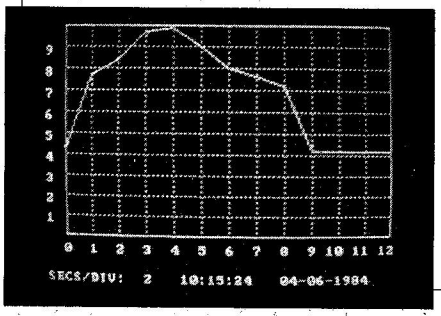
Junior's built-in timer can be set to transfer program execution to the light measurement subroutine after any interval of from 1 to 86,400 seconds (1 second to 24 hours). This means you don't have to waste time inventing calibrated delay loops to provide accurate sample intervals.

Back to our sample run, when the first sixty seconds have elapsed, program control is transferred to the light measurement subroutine at lines 290-350.

Line 300 retrieves the photoresistor reading (Y=0 to 255). Line 310 provides a correction factor that converts the raw reading to a scale of from 0 to 100 [Y=INT((Y/255)*100)].

The photoresistor's resistance is *inversely* proportional to the incident light (i.e. its resistance *decreases* as the light level *increases*). Therefore, line 310 also

Sweep of light across sensor.



complements the reading (Y = 100 − Y) so it will increase when the light level increases. However, line 310 does *not* supply a correction factor to compensate for the photoresistor's nonlinearities.

After the corrected and complemented (but not calibrated) reading is printed on the screen (320) and saved on disk (330), Junior emits a beep and program control returns to the timer routine. After the five samples have been measured, Junior beeps twice. The data tabulated on the screen might resemble this:

SAMPLE	LIGHT LEVEL (0 TO 100)
1	17
2	21
3	37
4	43
5	52

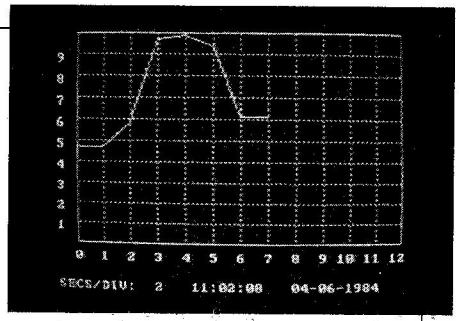
Pressing D tells Junior to retrieve the data from disk. You can then request a tabulated list or a plot of the first thirteen samples superimposed on an oscilloscope graticule, both annotated with file name, date, time and sample interval.

If you request a tabulation of the stored data, lines 580-670 in Listing 2 retrieve the data and print it on the screen. If you request that the data be plotted on a superimposed graticule, lines 690-760 are run.

Incidentally, line 460 catches the error should you attempt to retrieve a non-existent data file. You can then re-enter a correct file name. This error-trapping feature works for only one cycle, however. If you enter a *second* non-existent file name Junior exits its error-trapping mode and flashes a "Bad file name" error notice. You then have to rerun the program.

The program's graph option is complicated by the need to reconvert the retrieved data to the scale of the graph. To provide as large a graticule as possible in the space above the caption area, I selected a vertical scale of 0 to 160. Line 720 converts the retrieved data (0 to 100) to a scale of 0 to 160. The data is then complemented to match Junior's inverted graphic coordinate system.

The graticule itself is drawn by the subroutine at line 780. Line 820 draws the graticule outline. The grid is drawn by lines 830-880. Notice the ",,,&HCCCC" appended to lines 840 and 870. This causes the grid to be drawn with dashed rather than solid lines. Omit ",,,&HCCCC" if you prefer a solid grid.



Eight light level readings.

Going Further

This column is merely a preliminary look at what can be accomplished by adding external sensors to Junior's joystick ports. Many other sensor options are also possible. In future colums we'll examine additional sensors and address the tricky question of sensor calibration.

In the meantime, remember that the basic principles described in this month's column apply to machines other than the PC*jr*. For additional information on applying these principles, see "Analog Sensors for Personal Computers" and "Use Your TRS-80 Color Computer as a Storage Oscilloscope" (COMPUTERS & ELECTRONICS, February 1984). ◇

DO-IT-YOURSELF COMPUTER-SIMULATED INSTRUMENTS

DURING the past year I've developed several programs that transform computers into various kinds of electronic test equipment and measuring instruments. Eric Mims, my fifteen-year old son, has joined me in this endeavor. His latest program, which won a couple of science fair awards, transforms Radio Shack's CoCo into a digital thermometer that stores temperature samples at any specified interval. Later, the program retrieves the stored data and produces a bargraph of a series of measurements.

Though the programs Eric and I have developed provide capabilities as sophisticated as a storage oscilloscope, I didn't fully recognize the enormous significance of do-it-yourself computer-simulated test instruments until running, for the first time, Listing 1, which is, by comparison, a *very* simple program.

The ethereal image of the simulated light meter on the monitor raised my level of computer consciousness to a new high. Sure, no great skill was required to arrive at the program in Listing 1. But that's just the point. Unless you've spent the better part of a day assembling with a soldering iron and assorted hand tools a traditional light meter, it's difficult to fully appreciate the power provided by that relatively simple program.

Imagine, the computer-simulated light meter in Fig. 5 is a product of a quick sketch on a graphics worksheet, a few minutes of orchestrated keystrokes and little else! The meter can be easily modified to add extra features. For example, a correction factor that compensates for sensor nonlinearities can be easily inserted. Tired of its labels or color? Both can be easily changed. Bored with a conventional 0 to 100 readout? Change it, with a few extra keystrokes, to 100 to 0. Or −100 to +100. Or *any* scale you can imagine.

When it's not in use, the light meter wastes no space gathering dust on a shelf. Instead, it can be electronically vaporized by pressing a few keys. When it's time to make another light measurement, the meter can be reconstructed in seconds from the instructions previously saved on a few inches of magnetic tape or a tiny piece of floppy disk.

You can even make custom light meters and give them to your friends. Or, as I've done here, you can share them with literally thousands of fellow computer users in the pages of a magazine.

Yes, my computer-simulated gadgets require a power cord and they don't fit in a shirt pocket . . . yet. With *today's* technology, however, one can easily envision a pocketable computer about the size of Radio Shack's PC-3 equipped with a couple of analog-to-digital input ports and a liquid-crystal screen having a few thousand resolution points.

A machine like this would be a universal measuring instrument. Depending upon the resident program and the external sensor, it could function as a full-featured, programmable thermometer, light meter, pressure gauge, pulse-rate monitor and many other applications for which sensors are available. ◇