Keyboard

Nov-Dec/79 A Publication of Hewlett-Packard Desktop Computer Division

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Keyboard
November-December 1979

Cover

Water flows from the foot of the aspens and conifers that make up the forests in the Colorado Rockies. A study of transpiration, the process by which trees absorb water and allow it to evaporate from their leaves, may be a key to developing ways in which subalpine forests can be preserved, and yet made to yield more of their water for use by man. The cover story begins on the facing page.

Photo credits

page 13 top — Hewlett-Packard San Diego Division

All other photographs and artwork in this issue by Hal Andersen, art director.

Water engine of the Rockies

by John Monahan
with Dr. Merrill R. Kaufmann

Above the irrigated valleys and below the tundra-covered peaks of the Rocky Mountains, the "engine" that is the subalpine forest consumes a fuel more precious than oil.

In this land of bear and backpacker, invaluable water is lost by trees through transpiration — the process of evaporation through foliage — water that is crucial to all forms of life.

As a necessary part of growth, the trees transpire a portion of rain and melted snow which might otherwise feed rivers and streams that sustain arid regions of the western United States. When, for instance, subalpine trees in Colorado or Wyoming transpire 2.54 mm (0.1 inches) of rainfall, more than 90,000 acre-feet of water are lost, thus failing to reach farmers, industry, and others downstream.

Studying water use

At the Fraser Experimental Forest, near Fraser, Colorado, scientists from the U.S. Forest Service Rocky Mountain Forest and Range Experiment Station are attempting to analyze the transpiration rates of four species of trees.

Knowing how much water these trees use, it might be possible to manage the subalpine zone in a way that conserves water and still provides satisfactory tree growth. Equipment such as the Hewlett-Packard 9825 Desktop Computer is part of this research.

The question of water usage among growing trees was unanswered for years for want of a reliable measurement device and a system for analyzing the resultant data.

In response, Dr. Merrill R. Kaufmann, principal plant physiologist at the Experiment Station, developed an instrument for measuring water vapor exchange, based on a smaller, handheld model he devised earlier.

The height of water tumbling through this "v" shape acts as a gage of water runoff at the experiment station near Fraser.

The new device determines the rate of transpiration, as well as the degree to which pores in the foliage, called stomata, control transpiration and photosynthesis. Until now, there were no implements which could provide continous readings throughout the growing season.

It took a year to machine parts and assemble electronic components, and a summer to prepare the test site.

Having solved the problem of data collection and measurement, finding a reliable system to control the gas exchange chambers and analyze their measurements was the next task.

The experiences of other scientists at the Experiment Station made it clear that a system centered around the 9825 Desktop Computer and other instruments would be suitable. The computer could control the operation of the transpiration measurement equipment, attached high in the branches of trees, and display real-time analyses inside the trailer that serves as the instrument station.

Setting up

And so, after a year of machining parts and assembling electronic components, and after a summer preparing the test site and building six-meter towers around each of the test trees — fir, spruce, pine, and aspen — the study was begun. Dr. Kaufmann also took a programming course about the 9825 at HP's Desktop Computer Division.

The test site looks west from the Fraser Experimental Forest toward Byer's Peak, a stately mountain wearing a mantle of pine and a crown of tundra beginning at 11,500 feet.

Two gas exchange chambers, or cuvettes, are located on each of the four towers. One chamber encloses a branch on the tree's sunny side, the other a shaded branch.

The chambers are about the size and shape of a mailbox, with tops made of transparent, one-mil Teflon film. Fans continuously force outside air past the enclosed foliage.

Inside are sensors for measuring light intensity, relative humidity, internal and external leaf temperature, and air temperature.

A bundle of electronic cables 30 meters long and containing 10 pairs of wires connects each of the chambers to the 9825-controlled data analysis system inside the trailer.

One branch from a conifer at the test site set up by Dr. Kaufmann is isolated inside this chamber, or cuvette. Periodic measurement cycles determine the amount of water the branch transpires.

But this sophisticated equipment is outclassed by the stomata, tiny valves in the leaves and needles that respond to light, temperature, differences in water vapor concentration from leaf to air, and plant water deficits.

Controlling tests

These first three factors are monitored automatically by an HP 3455 Voltmeter and 3495 Scanner, both controlled by the 9825. Data for each chamber is automatically acquired and analyzed.

This occurs at 20-minute intervals for the light readings, once each hour for temperature and humidity measurements. A real-time clock determines when data is collected.

Also on the hour, the computer's program activates a sequence of events at the chambers and inside the trailer. Through the relay actuators of the scanner, a fan inside a selected chamber starts whirring. After the air is stirred three seconds, a solenoid closes a pair of trap doors which seal the chamber.

In this closed environment, measurements of relative humidity, along with air and leaf temperature, are made each second for 15 seconds. Because the change in relative humidity during this period reflects how much water vapor is introduced into the chamber by the foliage, the actual transpiration rate and stomatal control over water vapor movement from leaf to air can be determined.

Control of the cuvettes, as well as data collection from instrumentation on the platforms, centers on the 9825A system located in a trailer overlooking the experimental forest.

Dr. Merrill R. Kaufmann is principal plant physiologist for the U.S.D.A. Forest Service's Rocky Mountain Forest and Range Experiment Station, Fort Collins, Colorado. He received his Ph.D. in forestry from Duke University in 1967, and spent 10 years at the University of California-Riverside before joining the Forest Service in 1977.

Dr. Merrill R. Kaufmann
Rocky Mountain Forest and Range Experiment Station
240 West Prospect
Fort Collins, Colorado 80526 U.S.A.

Handling data

When the 30-second collecting and measuring sequence for each branch is completed, the data is analyzed by the desktop computer. Its findings are printed by an HP 9871A Impact Printer. Observations are then easily made.

More than a week's worth of data is stored on magnetic tape. The only servicing that's needed is changing the computer's tape weekly and checking the printer's paper supply. The gas exchange chambers are inspected each day.

Electrical power outages in this remote area are not uncommon. However, the autostart feature of the expanded I/O ROM, coupled with the battery-powered real-time clock, ensure the program is reloaded and the time correct in the event of momentary power failures.

Snow comes to Byer's Peak in September, the growing season ends and this year's field research has come to a close.

But plans for 1980 call for reinstallation of the equipment in late April, in order to determine when the use of water by subalpine trees begins. The system will work through the summer and into fall, when again snow comes to the high country. By then the scientists hope to know how late in the season water consumption persists.

In ensuing years, data from this system and that gathered by portable handheld devices will help determine if runoff from subalpine watersheds can be increased by changing the species composition of the forests in this magnificent wilderness, the water engine of the Rockies.

Detecting defective ICs

by Brenda Hume
Hewlett-Packard Company
Desktop Computer Division

The advanced technology of integrated circuits (ICs) has made it increasingly important to monitor and control critical process parameters when manufacturing ICs. In order to do this, Hewlett-Packard has designed and built its own IC wafer parameter test station.

An HP System 45 controls all the instruments in the system. It also controls the application of stimuli to devices (such as transistors or capacitors) on the wafer being tested.

The apparatus used to test wafers is called a prober. Measurements are automatically input to the System 45 until several integrated circuits on the wafer have been tested.

Test startup

The test begins when an operator loads and aligns a wafer under a set of probes and logs the wafer into the System 45. All wafers are logged into a data base when received from the clean room. If the computer searches the central data base for a particular wafer and determines it has not been logged into the area, the computer tells the operator to take corrective action. If it has been logged, the computer tells the data base that the wafer is at this test operation.

Next, the System 45 will instruct the prober to search for the initial chip location on the wafer. The prober positions the wafer under probes that contact the device under test.

After the test on an initial chip is completed, the prober will move to the next chip and perform another test. The operator continues the test until a sample or complete run of wafers is finished. The System 45 then transfers the data into a central data base hosted by an HP 3000. Data is relayed to the 3000 via a communication system consisting of a System 35 with serial interfaces.

Parameter measurements

The test system performs various parameter measurements on transistors, capacitors and other static devices (devices not requiring timing stimulus or response). Types of measurements made for transistors include threshold voltage (V_T), junction leakage and I_D vs. V_D (drain current vs. drain voltage).

Capacitors are tested for fixed oxide charge, oxide thickness, flatband voltage and capacitance, doping and threshold voltage by performing capacitance vs. voltage measurements. Other parameters that could be measured on capacitors are oxide breakdown and transient lifetime.

The programmable prober offers two advantages: first, sample or random tests on a wafer can be performed. Second, a technician has control for wafer mapping. Wafer mapping is the process of illustrating parameters in relationship to their position on the wafer.

As the prober scans the wafer, data is collected in the System 45 and stored in a local data base. With graphics capability and interfacing to a 9872A plotter, the System 45 can draw a diagram of the wafer including measurements or parameter curves.

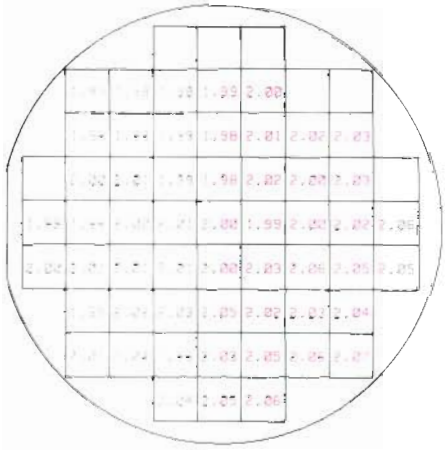


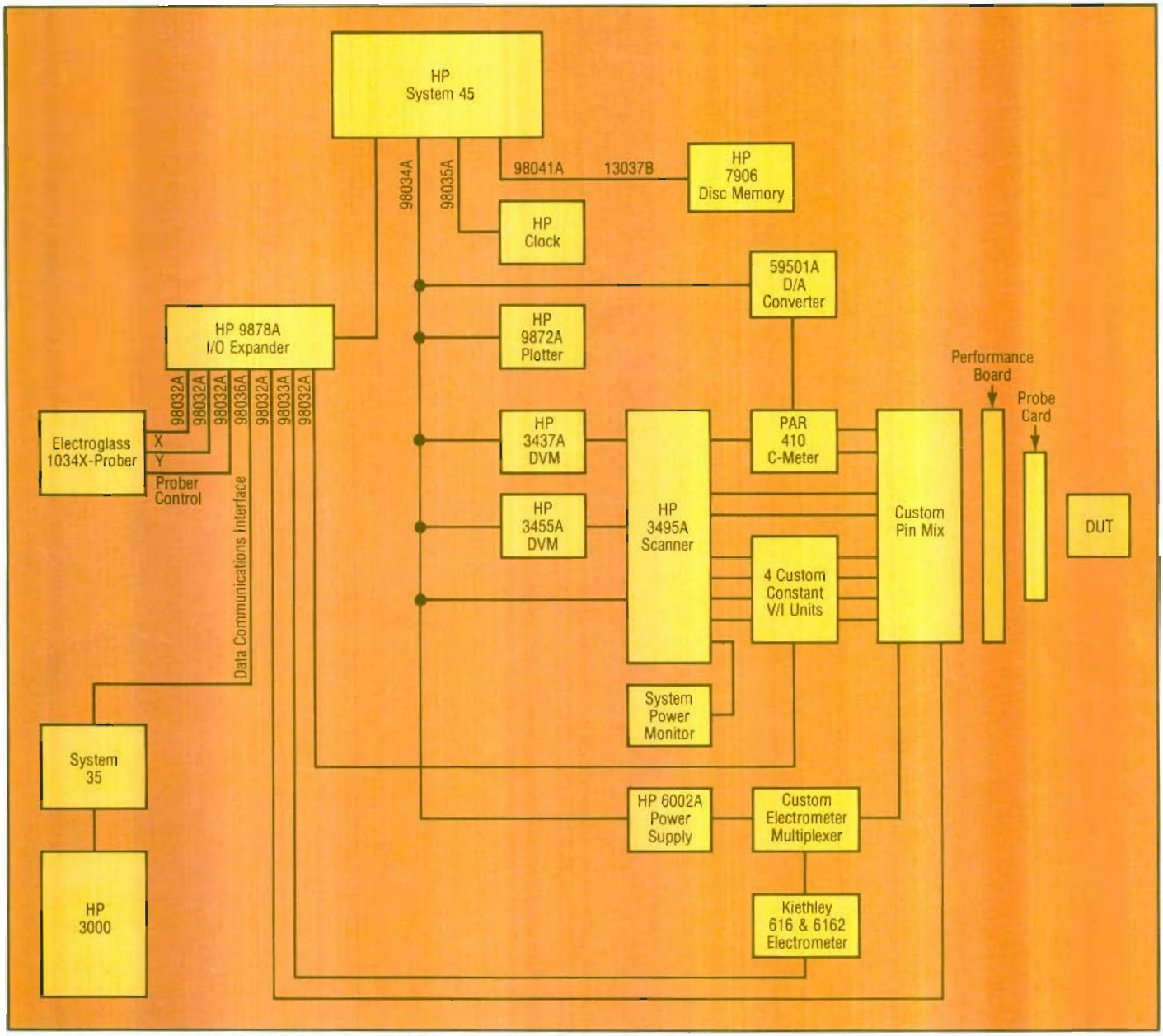
Figure 1

By drawing parameter contour plots, uniformity of doping or oxide thickness can be measured. Engineers can then pinpoint where problems exist in the production process.

Modeling

During the engineering phase of the integrated circuit process design, engineers can use the System 45 to model certain devices. Once the actual device parameters have been satisfactorily compared with the models, the engineer has a basis for product design rules.

After design rules are determined and production integrated circuits are being manufactured, the test system is used to monitor key parameters for



This diagram illustrates how data flows through the Desktop Computer Division's in-house IC wafer test system.

production control and trend analysis. By analyzing the hard-copy illustration of test parameters, the engineer can define specific process problems.

For example, figure 1 shows a V_T difference. This is indicative of geographical problems on the wafer. The V_T readings on the upper left side are much lower than the V_T readings on the lower right side.

Software

The software for this system is written in BASIC and contains several I/O driver and measurement routines. Software writer Tracy Ireland explained several advantages of this. "The routines control the hardware, perform standard parameter measurements and control prober movement.

"A user, therefore, doesn't need an extensive knowledge of the hardware. Subroutines for the data base allow wafer data storage by run number and wafer number which permit wafer-to-wafer and run-to-run correlations. This provides maximum flexibility in trend analysis."

By using subroutine calls, data can be stored or retrieved without the user having any knowledge of mass storage. The data base software is written to perform different tests many times on a single wafer.

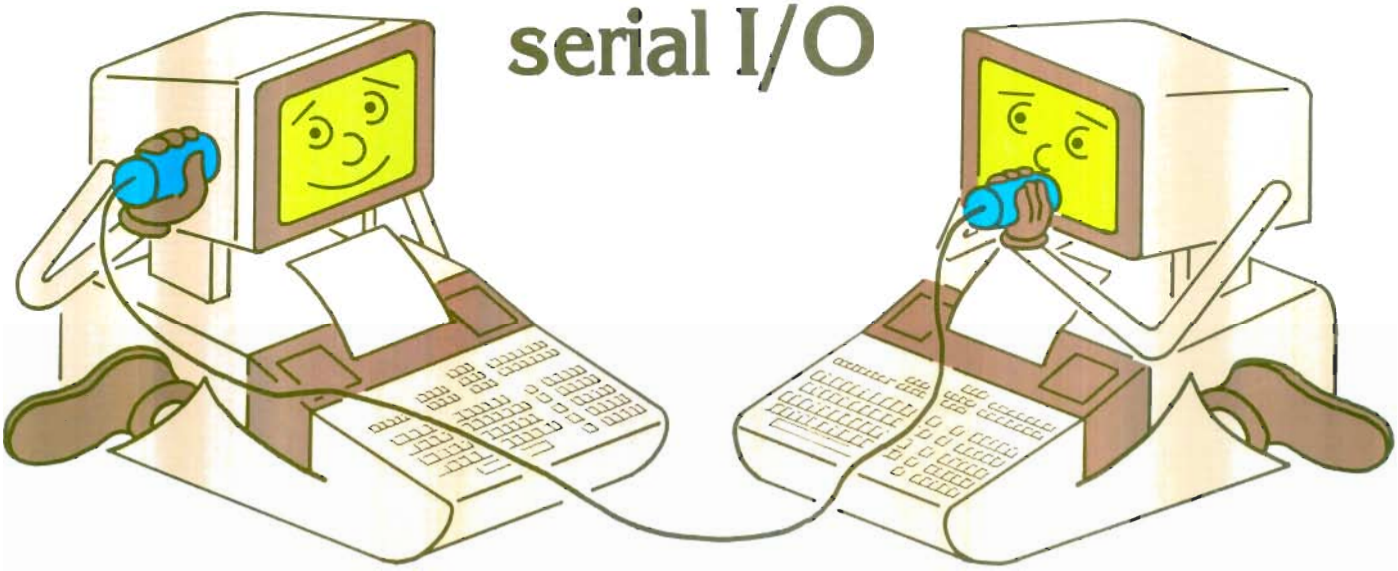
Self diagnostics

The major benefit of this system's software is the capability to diagnose problems within the system. For example, if the interface to the prober is not connected, the System 45 will inform the user of the problem. Because of these self diagnostics, the entire test system has experienced only 0.1% down time in its 1.5 years of operation.

The System 45 was selected for this test application for several reasons. The graphics capability provides an illustration of the integrated circuits on the wafer with test measurements. Fast I/O and computation overlap allow the user to perform calculations and subroutines simultaneously. Other features which are proving beneficial to this application include mass storage, self-documented HP BASIC, a hard-copy printer and interfacing.

System 45 provides an economical solution as a main controller for a test system requiring considerable computer power.

The unstandard interface: serial I/O



by Steve Leibson
Hewlett-Packard Company
Desktop Computer Division

The fact that the computer represents a relatively new technology may lead you to believe that I/O does also. The first electronic computers appeared in the 1940s, and serious work in computer data communications did not start until the next decade.

But when engineers did begin to connect computers to other devices, they used a technology that originated in the previous century — serial I/O.

Encoding for machines

The first electrical device used extensively for communications was the telegraph. Samuel Morse improved the telegraph mechanically, but more importantly, he devised the Morse Code. This was the first really practical encoding of the symbols humans use for communications into a machine-transmittable form.

Symbols are represented in the code by a series of dots and dashes, each character having its own unique representation. The dots and dashes may be considered the predecessors to the ones and zeros of the modern character codes we use in computer data communications.

Improvements of the telegraph led to printing telegraphs that required no human operator to decipher the codes. New codes and more advanced machines were devised, culminating in the teletypewriter.

By the time of the teletypewriter, dots and dashes had become ones and zeros. Morse code was discarded in favor of codes that assigned the same number of bits to each character. This made it much easier for machines to decode. By the time electronic computers were invented, there already existed a wealth of technology for electronic data communications.

At first, teletypewriters served as I/O devices between humans and computers. The keyboard and printer of the teletypewriter provided a low-cost data entry and display mechanism. As technology progressed, terminals and faster printers replaced the teletypewriters, but retained serial I/O interfacing.

Transmitting over one wire

The basis for serial data communications is the transmission of information over a single wire. Interfacing techniques previously discussed in this series have relied on several parallel wires to carry information between devices. Each wire carries a single bit of a character composed of multiple bits.

When long distances are involved, the cost of running several wires in parallel becomes prohibitive. Serial interfacing is the solution.

Time sharing was born when computers became sufficiently powerful to handle several tasks simultaneously. Since computers were still very expensive, it was necessary to spread their cost over many users.

The problem then was how to connect users at several locations to the central computer facility. Stringing wires to each location was too expensive. Fortunately, such communication lines already existed. They belonged to the phone system.

Building a standard

Unfortunately, these connections were not necessarily wires. They could just as easily have been satellite or land-based microwave links, since these also made up the phone system. All were designed to carry voice signals, not computer data.

In addition, phone companies were extremely unhappy at the prospect of finding all kinds of strange signals in their networks. Because teletypewriters did not have standardized interfacing requirements the voltages involved could range anywhere from 6 to 140 volts. A standard was required.

The Electronic Industries Association (EIA) standard RS-232C resulted. This standard was specifically developed to do one thing. It defines the electrical characteristics for an interface between a piece of data terminal equipment (DTE) and a piece of data communications equipment (DCE). DTE is the terminal for the timeshare user, while DCE is a modulator-demodulator (modem) which encodes the computer data into voice-like signals that are permissible on the phone system.

Figure 1 is a picture of the wires associated with the RS-232C standard. Note that there is actually more than one wire involved. Pins 2 and 3 are the data-carrying wires, called transmitted and received data. Pin 7 is a signal ground serving as a signal current return path. These three wires are sufficient for communications between DCE and DTE.

What are all the other wires for, then? They serve as control wires between the DTE and DCE and are there merely for establishing and maintaining communications with the computer. Let's ignore them for now.

Examining data wires

Let us examine the data wires more closely. Information is sent out on the transmitted-data line while signals are received on the received-data line.

Note that this is not the same as the HP-IB interfacing standard discussed previously. On that interface, one set of data lines carried information in both directions. Technology had not progressed sufficiently for RS-232C to have bidirectional signal lines.

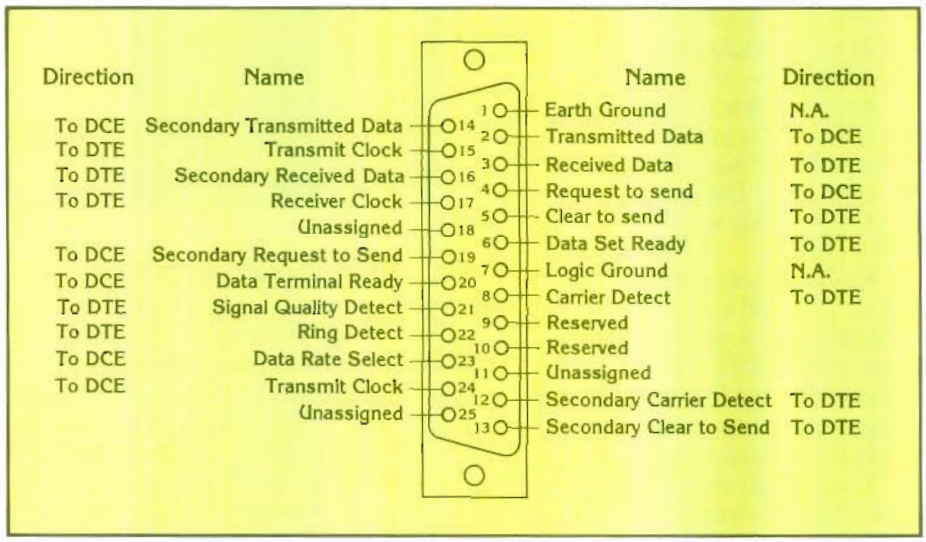


Figure 1

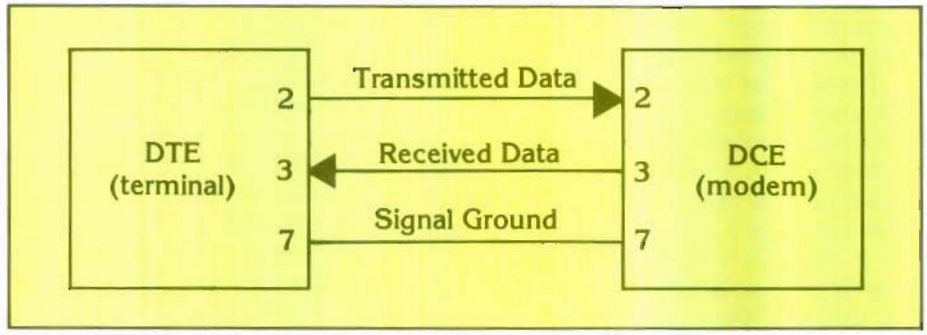


Figure 2

Who transmits on the transmitted data line? All of the names for the RS-232C signals are from the perspective of the DTE. So the DTE transmits on the transmitted data line and the DCE receives on it.

Similarly, the DTE receives data on the received data line and the DCE transmits on it. Figure 2 should clear up any confusion about this.

Now let's get confused again. We have a computer and a printer. We are going to connect them using their "standard" RS-232C cables.

But which of them is the DTE and which is the DCE? More specifically, which is going to transmit on pin 2 of the connector and which on pin 3? Neither the computer nor the printer is a terminal or a modem.

Manufacturers of these instruments may offer cables to allow their equipment to look like either DTE or DCE. More often however, the RS-232C connector is installed on the rear panel of the instrument and no choice is possible. In the case of two instruments of the same type, a crosswire cable may have to be assembled to get signals on the correct wires. Usually this task falls to the user.

We have just covered one source of incompatibility between pieces of RS-232C equipment: plug-to-plug. In

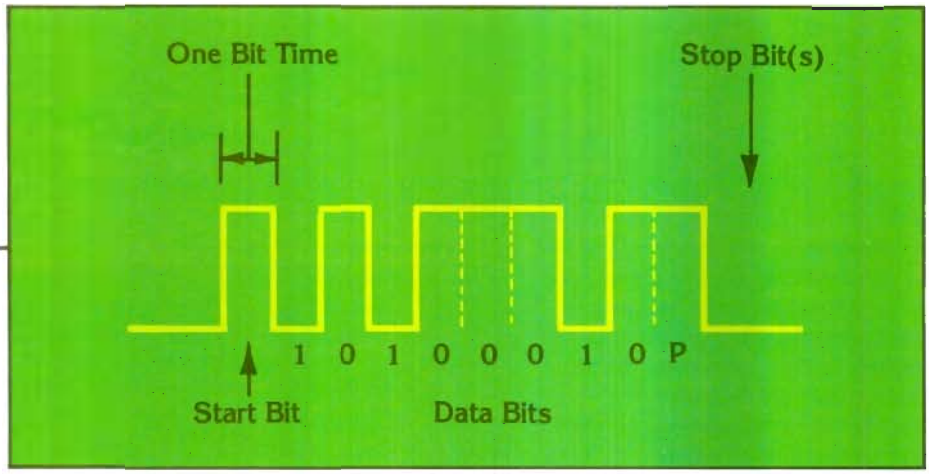


Figure 3

order to discuss other potential problems we must understand the data signal.

First, the data signal levels are not like those of the interfaces we have discussed previously. Those interfaces were based on TTL integrated circuits. That logic family is based on 0- and 5-volt signals.

RS-232C was a standard long before TTL and so uses different voltage levels. A positive voltage between 5 and 25 volts is used to represent a logic 0 while a negative voltage between −5 and −25 volts is used to represent a logic 1.

These levels are only for the data lines which use negative true logic. The control lines all use positive true logic and so a positive voltage represents a logic 1 and a negative voltage represents a logic 0.

Because the bits of a character are separated by time, a waveform is produced when transmitting a character. Such a waveform for transmitting the character "E" is shown in figure 3. The ASCII code for "E" is 1000101 in binary and is transmitted least significant bit first. The data line idles in the "1" state.

Waiting for the start bit

A start bit is always sent first to mark the beginning of the character. Then data bits are sent in order from least significant to most significant, each bit remaining on the line for a precisely-arranged length of time called a bit time.

The receiver, alerted to the incoming character by the start bit, times the incoming signal and samples the state of the data line as close to the center of the bit as it can. Naturally, both the transmitter and receiver must agree on the length of time each bit will be given on the line or the transmission will be garbled by samplings taken at the wrong times.

This bit time determines the maximum rate at which bits may be transmitted and thus defines the "bit rate" at which this particular serial interface is running. Standard bit rates are 50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2400, 3600, 4800 and 9600 bits per second.

Following the data bits in the transmission, there may be a parity bit, used for error detection. If a noise pulse should affect the data line at the wrong time, a bit in the transmission could be misread.

If the transmitter keeps track of the number of 1s in the character being transmitted, it could set the parity bit so that the total was either always even (even parity), or always odd (odd parity). The receiver can also keep track and use the parity bit to determine whether the transmission was received in error.

The last bits to be transmitted are the stop bits. These bits carry no information but allow the receiver time to prepare for the next character. There may be 1, 1.5 or 2 bits. Since this period is really just a resting time, fractional bits are allowed.

Picking up the pieces

Now, what might go wrong? There are several parameters on which the transmitter and the receiver must agree. The bit rate has already been mentioned. In addition, the parity (odd, even or none) and the number of stop bits both have to be the same in the transmitter and the receiver.

Character codes also have to be considered. Our example used the ASCII code that is the most commonly used character code today. ASCII is a seven-bit code. Other codes do exist however, and may include 5 bits (Baudot and Murray), 6 bits (IBM Correspondence Code) and 8 bits (EBCDIC). The RS-232C standard makes no mention of what character code should be used.

Now that we have our computer and printer connected so that they use the proper wires and they agree on bit rate, parity, number of stop bits and character codes, you may feel that the problem has been solved. — Not so. The printer uses the ASCII character set, has odd parity, and one stop bit. In addition, it has a switch for setting the bit rate up to 9600 baud.

Well, that's fine. We set the switch to its fastest setting so that our equipment doesn't waste a lot of time sending characters. We also set the computer to the same settings so that both the transmitter and the receiver agree on all of the parameters.

Finding the problem

We then write a program on the computer to send one line of print to the printer and it works! Finally, we list out the program on the printer so that we have a copy of our triumph. Unfortunately, several characters are lost in the transmission. We try again with equally dismal results. We run the program once more, and now that line prints perfectly. What is going wrong?

First, let's consider the data rate at which the computer is sending information. We are using ASCII (7 bits) plus a parity bit, a stop bit, and a

Surely RS-232C must have a handshake mechanism also — but no, this is not always true.

start bit for a total of 10 bits. We are transmitting these 10-bit pieces of information at 9600 bits per second, which translates into 960 characters per second.

The printer manual specifies that the printer can print 175 characters per second. We are sending information to the printer at more than five times the rate it can print them! The printer does have an internal buffer for 127 characters. After that, the transmitter must wait to allow the buffer to partially empty.

When we ran the program, fewer than 127 characters were sent to the printer and the printer's buffer could handle the data rate. Listing the program sent more than 127 characters to the printer and the buffer overflowed, causing some characters to be lost.

In previous articles, we discussed interfaces that had handshake mechanisms which prevented transmitters of information from going too fast for their receivers. Surely RS-232C must have a handshake mechanism also — but no, this is not always true.

Returning to figure 1, we will now consider the other RS-232C signals. We are looking in particular for two sets of handshake lines, one for the transmitted data line and the other for the received data line.

Aha! Pins 4 and 5, request-to-send and clear-to-send, look like prime contenders. And many printer manufacturers have fallen into this trap. According to the strict RS-232C definition, the DTE asserts the request-to-send line when it has some data to transmit. It then waits for the DCE to assert the clear-to-send line before transmitting.

That is one half of a traditional handshake. The problem is that the DCE is not allowed to drop the clear-to-send line until DTE has dropped the request-to-send line.

Take a long drink

The situation is similar to taking a drink from a garden hose with a friend controlling the spigot. It's easy to start the flow, but you had better be prepared to take either a long drink or a short shower.

The DTE and DCE signals were intended as a handshake between the terminal and the modem to allow the terminal to request control of the communications link from the modem. It also makes it possible for the modem to tell the terminal when control has been acquired.

Some manufacturers have ignored this strict definition and used the clear-to-send line as a handshake line anyway. Others avoid the definition conflict by using the data-terminal-ready or data-set-ready line (depending on whether they are emulating a terminal or modem). None of these lines was intended for the purpose of handshaking characters, however, and use by one instrument does not guarantee recognition by the other.

Let us consider the possibilities of using the clear-to-send line. If the printer drops the clear-to-send signal in the middle of a character, what should the computer do?

If it stops immediately in the middle of a character, the character is certain to be garbled. If it waits till the end of the current character to stop, it may overrun the receiver's buffer. Because this possibility is not covered in the standard, the results are not predictable without carefully reading manuals for both instruments.

Finally, consider the device that started this discussion, the teletypewriter. The RS-232C standard at least defines signal levels and a pinout on a connector. There are no standards for teletypewriters. The serial transmission concepts are the same with start and stop bits, data bits and parity bits, but the signal interface is called current loop.

Instead of positive and negative voltage levels to represent logic 0 and 1 levels, current loop uses the absence or presence of current. Presence may be either 20 or 60 milliamps depending on the teletypewriter model. There is no standard connector or pinout for teletypewriters.

Despite all of these problems, designers of serial interfaces for computers strive to include current loop capabilities in their designs. Where RS-232C is limited to 50 feet for a direct connection, current loop can be run much farther. In addition, the teletypewriter interface has been around for many years and several instruments still useful to connect to computers use it. Teletypewriters remain a cost-effective solution as a combination printer and terminal.

Handle with care

As you can see, serial interfacing needs to be approached more carefully than other forms of hardware interfacing techniques. There are several factors that are far from being standardized.

As technology progresses, serial interfaces become more adept at covering an ever-widening range of hardware. Unfortunately, this still doesn't guarantee an efficient interface in every application, because serial I/O remains the unstandard interface.

Programming Tips

MAT SORT & MAT SEARCH System 35A/35B/45B

This programming tip tells how to use the MAT SORT and MAT SEARCH statements on mixed upper and lower case strings and produce results as if all the characters were in the same case.

When doing comparisons on character strings where case is not a concern, the normal procedure is to use the UPC$ function, e.g., UPC$(A$)<UPC$(B$). However, when using the MAT SORT or MAT SEARCH statements defined by the System 35 or System 45 Advanced Programming ROM, this technique is not available. One could use a FOR/NEXT loop to go through and UPC$ all the strings in the array to be sorted or searched, but that would sacrifice the original form of the data.

Let's say, for example, that you have typed in data for a list of names that included ADAMS, WOTTEN, MACMAHON, ZHUKOV, and GREENOUGH. A short time later, another person adds names that include Thomas, Galerius, Cowley, Seward, MacKaye and Machiavelli. You typed names entirely in upper case letters, while the second person added names with initial upper case letters.

This is not a problem with a short list, as you can change them with little difficulty. However, if there are hundreds of names added to the list, it can be a problem if you try to sort them alphabetically. With a standard sort of the names above, they would print out:

```
ADAMS
Cowley
GREENOUGH
Galerius
MACMAHON
MacKaye
Machiavelli
Seward
Thomas
WOTTEN
ZHUKOV
```

The reason for this is that upper case letters appear before lower case in the ASCII character set, and receive priority. Even if both of you type with initial upper case letters, the names MacKaye and MacMahon would appear before Machiavelli in a simple sort, because of the upper case letters in the names.

The solution to this problem is to use the LEXICAL ORDER IS statement with an appropriately modified lexical order table, one in which the upper and lower case letters have the same sequence numbers. The first program given below will create such a table from the 'ASCII' table on the cassette that comes with the Advanced Programming ROM. The second program shows how to set up the LEXICAL ORDER from that table. The third program, with output, shows the results of doing a MAT SORT with this LEXICAL ORDER.

Create table

```
10  OPTION BASE 1
20  INTEGER Table(354)
30  ASSIGN #1 TO "ASCII"
40  MAT READ #1;Table
50  FOR I=0 TO 25
60  Table(100+I)=Table(68+I)
70  NEXT I
80  CREATE "UPC$LT",8
90  ASSIGN #2 TO "UPC$LT"
100 MAT PRINT #2;Table
110 END
```

You should note that with languages other than English, the integer table (Create table, line 20) would have to be larger. Designing the program to operate correctly would require referring to the Advanced Programming ROM manual.

Set up LEXICAL ORDER

```
10  CALL Table_set_up
20  !
30  ! USER PROGRAM
40  !
50  END
60  SUB Table_set_up
70  OPTION BASE 1
80  INTEGER Table(354)
90  ASSIGN #1 TO "UPC$LT"
100 MAT READ #1;Table
110 LEXICAL ORDER IS Table(*)
120 SUBEND
```

MAT SORT with LEXICAL ORDER

Here we'll insert our quasi data base to demonstrate that the MAT SORT with this LEXICAL ORDER will disregard the case of the letters and print out the names in alphabetical order, preserving their original form.

```
10  DIM String$(10)
20  DATA ADAMS,WOTTEN,
        ZHUKOV,GREENOUGH,
                MACMAHON
30  DATA Thomas,Galerius,
     Cowley,Seward,MacKaye,
              Machiavelli
40  MAT READ String$
50  MAT SORT String$
60  MAT PRINT String$
70  END

ADAMS
Cowley
Galerius
GREENOUGH
Machiavelli
MacKaye
MACMAHON
Seward
Thomas
WOTTEN
Zhukov
```

by Stephen M. Taylor
Hewlett-Packard Company,
Desktop Computer Division

Reading data again

The programming tip, "READ/DATA capability in HPL," which appeared on page 13 of the September/October issue of *Keyboard*, contained several misprints in a program listing. The corrected version of that program is reprinted here, along with a line-by-line explanation supplied by the author of the tip, Joseph Pepin.

```
 0: dim D$[100]
 1: buf "data",D$,3
 2: red 'DATA',A,B,C
 3: prt A,B,C
 4: red 'DATA',D,E,F
 5: prt D,E,F
 6: stp
 7: "DATA":list #'dat',r0,r1
 8: r0+1→r0
 9: if (pos(D$,"  ")=0)*buf "data",1,r
10: ""→D$[1,r1+7]
11: ret "data,"&char(r1+48)
12: "dat":ret "data.1"
13: "data 1,1,2,2,3,3":
14: "data 5,6,7":
15: end
```

Line 0: Dimensions a string that is going to hold a line of the program listing.

Line 1: Establishes a buffer named "data", which is the string already dimensioned in line 0, and is Type 3, byte-oriented fast read/write buffer.

Line 2: Reads three items of data into A, B, and C. Calling the function 'DATA' will return the buffer name "data", after the function has placed the data into the buffer.

Line 3: Prints the data just read.

Lines 4 and 5: Similar to lines 2 and 3, shows that the 'DATA' function automatically positions a pointer to the next set of data.

Line 7: The 'DATA' function: Uses r0 as a line pointer, and lists a single line of program into the buffer, whose name is returned by the 'dat' function.

Line 8: Advances the line pointer. Originally at 0, it will point to the next line the next time the 'DATA' function is called. Setting r0 to 0 will mimic the BASIC RESTORE statement.

Line 9: Checks to see if the program line just read in was a dummy data statement. If not, it empties the buffer and tries again.

Line 10: The program line just read in was a dummy data statement. Blank out the statement number and the "data". The red statement reads the remainder of the data statement as if from an external device.

Line 11: Returns the name of the buffer to the red statement. An optional format number may be enclosed within parentheses after a 'DATA' call. Otherwise the standard format is used.

Line 12: The 'dat' function: Returns the name of the buffer to the list# statement on line 7. The .1 ensures that the listed line contains no extra line feeds or the check sum.

Lines 13 and 14: These are dummy data statements containing the data in a long label.

by Joseph Pepin
Western Electric
Engineering Research Center
P.O. Box 900
Princeton, New Jersey 08540 U.S.A.

Dynamic file allocation System 35A/45A/45B

In designing a system, there are times when the absolutes inherent in the language of the computer conflict with the changing reality of our problem. One such case is when we really don't know within, say, 10%, the amount of data storage space that will ultimately be needed for our application. Once a file is created with the CREATE statement, its size cannot be changed.

Let's say we're working with a mailing list for a new newsletter. If the newsletter is very successful, we might eventually need to keep track of 1000 names and addresses. If that happens, we would be happy to invest in storage space for the names, of course, but then it is also possible that the mailing list might not grow very soon. We don't want to permanently allocate space which may never be needed, nor do we want to permanently restrict the file size at the start of our system design.

In any application, we can choose a somewhat arbitrary space increment. Let's say for our mailing list we decide to increase the file size by 50 records at a time. When the first space of 50 records is full and we try to add the 51st entry, the program will automatically set up a second space of 50 records. Further, since our file is named "MAIL", we will identify the individual subsets of "MAIL" as "MAIL1", "MAIL2", "MAIL3", etc.

The following program lines show the technique required to numerically assign the names for five files.

```
FOR Subset=1 TO 5
CREATE "MAIL"&VAL$ (Subset),50
NEXT Subset
```

The result of the above program would be "MAIL1", "MAIL2", "MAIL3", "MAIL4" and "MAIL5" created on the mass storage tape or disk. Later, we'll see how these files can be allocated as needed.

In a system, we would want to be able to check and see if the file already existed before creating it. The following program lines show the use of the ASSIGN statement return variable for checking to see if the file already exists. If the value of the return

variable is 1, the file does not exist.

```
ASSIGN #1 TO "MAIL"&
          VAL$ (Subset),Checkword
IF Checkword=1 THEN Nofile
```

When we print into a file, if there is no more space, we want to allocate more space. But if we are reading from the file, we need to have some way to determine when we reach the end of a file whether there are more file subsets or not. The following program lines determine whether a new file subset should be created depending upon a preset flag called Mode$.

```
IF Mode$="READ" THEN Finished
IF Mode$="PRINT" THEN CREATE
    "MAIL"&VAL$(Subset),50
```

Whether reading from the file or printing in the file, when an end of file condition is detected, we want to return to the portion of the program where a new file is allocated. The following program line guarantees return to the Allocate subroutine whenever an end condition is detected.

```
ON END #1 GOSUB Allocate
```

In the following program, we allocate the first file subset at the beginning of the print routine and again at the beginning of the read routine. Part of the procedure for file allocation is to establish a return to file allocation when the end of file is detected.

This program allows keyboard entry of any number of data items, and then prints these entries from the file when data entry is complete. Of course, in actual applications, there will be other steps such as updating the file and sorting the records which are not shown here.

```
10    DIM Name$[30],Street$[30],City$[30],Zip$[10],Mode$[5]
20 Entry:   Subset=0
30    Mode$="PRINT"
40    GOSUB Allocate
50    LINPUT "Enter name.  To exit, enter END.",Name$
60    IF Name$="END" THEN Exit
70    LINPUT "Enter street address",Street$
80    LINPUT "Enter city and state",City$
90    LINPUT "Enter zip code",Zip$
100   PRINT #1;Name$,Street$,City$,Zip$
110   GOTO 50
120 Exit:    PRINT #1;"END"
130   INPUT "ENTRY COMPLETE.  DO YOU WANT A PRINTOUT?",A$
140   IF A$[1,1]<>"Y" THEN Finished
150 Printout:  Subset=0
160   Mode$="READ"
170   GOSUB Allocate
180   READ #1;Name$,Street$,City$,Zip$
190   PRINT USING Form;Name$,Street$,City$,Zip$,
200 Form: IMAGE K,/,K,/,K,/,K,2/
210   GOTO 180
220 Finished:   ASSIGN * TO #1
230   DISP "PROGRAM COMPLETE"
240   END
250 Allocate:  Subset=Subset+1
260 Assign:  ASSIGN #1 TO "MAIL"&VAL$(Subset),Checkword
270   IF Checkword=1 THEN Nofile
280   ON END #1 GOSUB Allocate
290   RETURN
300 Nofile:   IF Mode$="READ" THEN Finished
310   IF Mode$="PRINT" THEN CREATE "MAIL"&VAL$(Subset),3
320   GOTO Assign
```

by Donna Kimble,
System 45 Instructor,
Hewlett-Packard Company,
Desktop Computer Division

Added features

Discover a new plot

A graphics plotter with automatic paper advance is coming off the assembly line from the San Diego Division of Hewlett-Packard. It is a new version of the 9872A Plotter called the 9872S.

Automatic paper advance makes it possible for the plotter to operate unattended to output a number of graphs, or provide multiple copies of one high quality, multicolor graph. It uses the same multicolor mechanism featured in the 9872A.

A paper advance mechanism is integrated into the plotter to provide paper advance on program command or front panel command. Plots can be made on either half-size or full-size pages, then cut into sheets and stacked — all automatically.

Paper for the plotter comes in 200-foot rolls, and is moved through the plotter by a sprocket drive. A roll of standard plotting paper is 11 in. wide, and can be used to produce either 11 x 17 in. plots (140 to a roll), or 8½ x 11 in. plots (280 to a roll). Paper is also available to produce plots in 210 x 297 mm or 297 x 420 mm sheets for metric users.

Some of the applications for which the automatic paper advance is especially useful include automated test systems, where the plotter can produce individual plots for a series of measurements; multi-user computer graphic ouput stations, where significant labor savings can be had by using a system which runs unattended; and multiple-copy production of a single graphic for presentation use, where again labor is saved. Optionally, until April 1, 1980, HP will convert existing 9872A Plotters to include chart advance on request.

System 35 Enhancements

AP ROM

A new Advanced Programming ROM provides for array sorting, array searching, flexible character sorting and CAT to A$. It comes with a single module in a ROM drawer, an Advanced Programming Manual and a lexical tables tape cartridge. The cartridge contains collating tables for ASCII and French, German, Spanish and Swedish/Finnish languages.

Mass storage ROM

The new Mass Storage ROM for the 9835A/B is now available. While the earlier 98331A supported only the internal tape cassette and the 9885M/S floppy disc, the new 98331B ROM supports not only those devices, but the 7905M/S, the 7906M/S/H, the 7920M/S and the 7925M/S. The 98331B comes with the Mass Storage Manual and a ROM drawer with two ROM modules in it. If you need to add a hard disc to your present 9835A/B system, you need the 98331B.

Keyboard

System 35 Desktop Computers shipped since late summer have included a true typewriter-like keyboard, with raised keys. The new keyboard is similar in appearance to that of the System 45, and uses the same quality key caps.

Much more important, the keyboard has a projected reliability improvement factor of 3 to 5 over the keyboard which it replaces on the System 35. All of this will minimize downtime and repair costs of the computers.

Local language options are available to make the machine English (standard), French (option 810), Spanish (820), German (830) or Swedish/Finnish (850).

Owners of System 35s produced before introduction of the new keyboard can order it for retrofitting. The upgrade kit for the System 35A is part number 98328F. The kit for the System 35B is part 98329F.

Update

Share your knowledge

Since publication of this magazine began in 1969, scores of desktop computer users have shared their knowledge and experience by submitting articles about their applications to *Keyboard*. You have had the benefit of their help in understanding the potential of computers, and have learned about desktop computer capabilities that you likely had never imagined. We would like you to return that favor by sharing your knowledge with fellow readers.

The purpose of *Keyboard* is to serve users of HP desktop computers. One of the ways we do this is by publishing informative articles about how users around the world are applying their computers. We try to make these articles as interesting as possible, while conveying to you the technical information you need to understand the worth of various applications. Each article is printed with a mailing address for the author, so that you can contact him or her for further information. If you find that the applications of others are interesting to you, then perhaps you can understand why others may find your own application beneficial, interesting, and perhaps even exciting.

Keyboard continually receives letters and phone calls from readers who want to submit application articles for publication in the magazine, but don't know how to get started. If you are interested in submitting an article on your application to *Keyboard*, feel free to call or write us at the address listed below on this page. We can help define the scope of your article, give you tips on how to write to the *Keyboard* audience, and even help decide what kinds of photographs to use in illustrating the article.

Further information on writing for *Keyboard* is available in an article which appeared on page 21 of the July/August issue. If you do not have a copy of this issue, we will be happy to send it to you.

Programming Tips book

Do you enjoy reading the Programming Tips section of *Keyboard*? Do you continually find that you would like even more Tips? Do you have a file full of torn-up back issues of *Keyboard* that you search through to get programming ideas?

Well, get ready to dispose of those well-worn files. We are completing a book in which we have edited and compiled the 9800 series Programming Tips published in past issues of *Keyboard*. Between its covers you will find Programming Tips for the 9810, 9815, 9820, 9821, 9825, 9830, 9831 and System 35 and 45 Desktop Computers.

The book contains Programming Tips from *Keyboard*'s inception in 1969 up to the present, and includes Tips published in this issue. What is more, the Programming Tips book is available to users of 9800 series desktop computers at no charge — it's *Keyboard*'s 10th Anniversary present to you.

All the Tips have been organized by mainframe and indexed for convenient use. In addition, the book includes a section of miscellaneous Tips which apply to several different mainframes, or involve general programming concepts.

Programming Tips books will be available February 1, 1980. If you would like to place your order now, write to the address below for your free copy. And be sure to include your mailing address, or attach a mailing label from your latest issue of *Keyboard*. Please write to:

Hewlett-Packard/*Keyboard*
3404 East Harmony Road
Fort Collins, Colorado 80525 U.S.A.
ATTENTION: Chris Stumbough

Keyboard

Hewlett-Packard, 3404 E. Harmony Road
Fort Collins, Colorado U.S.A. 80525
Phone: 303/226-3800

Publications Manager - Ed Bride
Editor - Bill Sharp
Feature Writer - John Monahan
Editorial Assistant - Chris Stumbough
Art Director - Hal Andersen
Typography - Paula Dennee

For further information on HP products or applications, please contact your local Hewlett-Packard Sales and Service Office or write to Keyboard

CHANGE OF ADDRESS: To change your address or delete your name from our mailing list please send us your old address. Send changes to Hewlett-Packard *Keyboard*, 3404 E. Harmony Road, Fort Collins, Colorado 80525, U.S.A. For Europe, send changes to Hewlett-Packard N.V., Post Box 529, 1180 AM Amstelveen, Holland

Nov 1, 1979