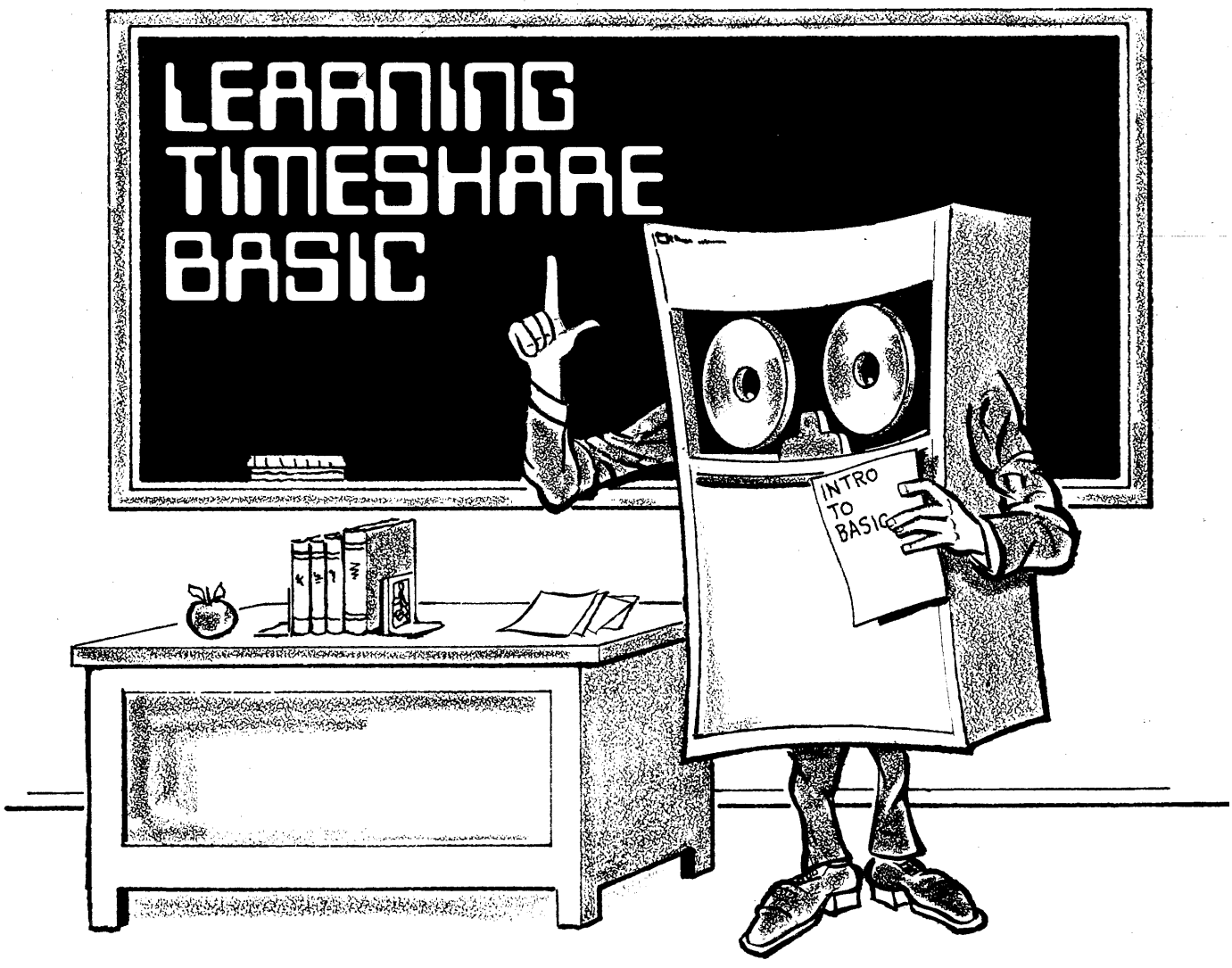
HEWLETT hp PACKARD



preface

BASIC means two things. First, BASIC means the Hewlett-Packard version of the BASIC programming language. And second, BASIC is the operating system provided with your computer (actually one of several timeshare systems offered by Hewlett-Packard.)

When you are using BASIC the programming language, you construct a set of instructions (called *statements*) which are to be performed by the computer when your program is run. Statements can be simple instructions such as PRINT "BLA-BLA" or involved instructions containing formulas and other intricacies. A program generally consists of many statements which tell the computer to perform a specific job. When a program is run, the statements are performed in the exact sequence that you number them.

When you are talking directly to the BASIC operating system, you issue *commands* which are carried out immediately by the computer. Commands tell the operating system to run a program, create a file, or some such task, and are generally brief, consisting of one or two words. Commands are never included in a program.

To master BASIC then, all you need to do is understand the various statements and commands available to you. This manual defines the most commonly used BASIC statements and commands, and will enable you to write simple, effective programs on any HP Timeshare BASIC system*. The complete set of BASIC statements and commands available to you is explained in the reference manual appropriate to your HP system. The latest HP Timeshare BASIC system is described in the *HP 2000/ACCESS BASIC Reference Manual (22687-90001)*.

**Learning Timeshare BASIC* is not applicable to the Hewlett-Packard Timeshared BASIC/2000, Level E, System.

content

the terminal

Before you get under way, be sure you have familiarized yourself with your terminal. The terminal is the device through which you talk to the computer and by which the computer speaks with you. Your terminal may be either a printing device (rather like a typewriter) or a video display screen device (sort of a cross between a typewriter and a television).

A printing device looks like this.

A video display screen terminal looks like this.

Either type of terminal may be used with BASIC, provided you know how to do the following operations:

1. How to turn the terminal on and off.
2. How to establish a connection between the terminal and the computer.
3. How to enter data and complete an entry.
4. How to correct mistakes, for example — erasing lines and backspacing over incorrect answers.
5. How to "break in" or override terminal activity in an emergency.
6. Where to find help when all else fails.

HOW TO TURN THE TERMINAL ON AND OFF

Look for the switch on the keyboard, on the side casing, at the back, or even on the bottom of the terminal. Turn it to ON, or if there are options, turn the switch to the position called LINE. Be sure to turn the switch to OFF when you have finished using the terminal.

HOW TO ESTABLISH A CONNECTION

Once the terminal is on, press the keys marked RETURN and LINE FEED. If the terminal is connected directly, the computer will say PLEASE LOG ON. If the terminal is not connected, the computer will say nothing at all and you must phone the computer to establish the connection. Procedures for calling a computer are included at the end of this manual.

HOW TO COMPLETE AN ENTRY

Every line of data you type must be terminated by the RETURN key. This key signals the computer that you have finished a line of input. The computer can respond by advancing the terminal to the next line permitting you to continue with your input, or even by asking you a new question. Remember — a line of input may consist of only one word or character. You must still press the RETURN key to indicate that you are finished. A line or character may be erased only before the RETURN key is pressed. So be sure you like what you typed before you press the RETURN key and send it to the computer.

HOW TO CORRECT MISTAKES

If you realize that you've made a typing error and have not yet finished typing the line of data containing the error, you can erase the line by holding down the CONTROL key and the X key at the same time. The line is thus effectively "erased" and the terminal advances to a clean line so that you may try typing the data again.

CONTROL/X prints a backward slash (\ ), erases the line, and advances the terminal.

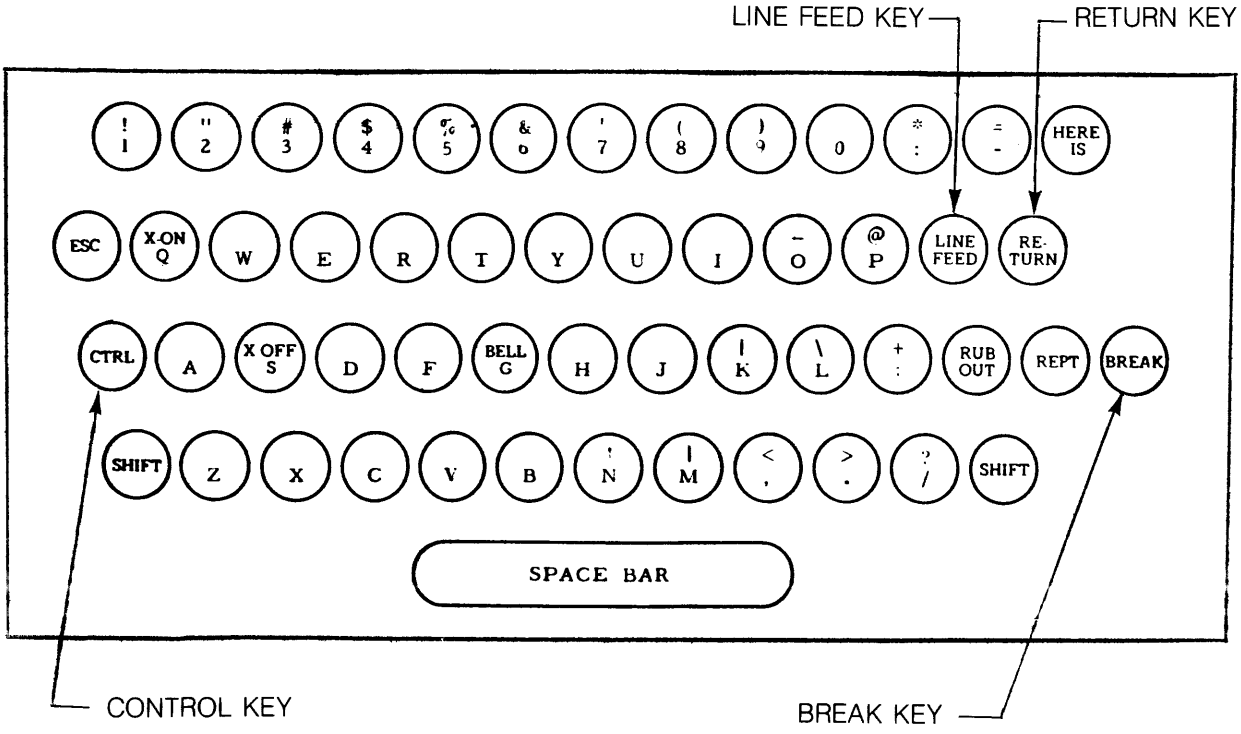
A single character in a line can also be erased. Press the CONTROL key and the H key simultaneously to erase the last character you typed. Then type the character you really wanted in the first place. If you erred for more than just one character, hold down the CONTROL key and press the H key together for each character to be erased.

THIS IS THE DATT← A Control/H prints an ← or _ and erases the last character.

Notice: The character keys used for correction may differ according to terminal. Experiment with your terminal until you are sure that you know how to erase lines and/or characters.

HOW TO BREAK IN

The BREAK key lets you stop a running (or runaway) program. This is an emergency button that lets you regain control of the terminal by interrupting a program when necessary. Of course, interrupting a program destroys what it was accomplishing, and you have to restart it by typing RUN. And it does start again, from the beginning.



logging on & off

Logging on is the way in which you establish contact with the BASIC operating system. All you have to do to log on is to type HELLO and then identify yourself by user name and password.

```
HELLO-B123,SECRET
```

That's an example of logging on.

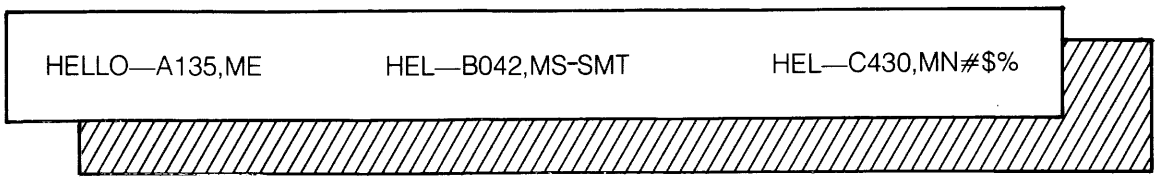
HELLO— is the command you must use. By the way it can be abbreviated to HEL— if you prefer.

B123 is my user name, you'll have to use the one assigned to you, of course. Just ask your system manager.

SECRET is my password. Once again you'll have to use your own. You can be really clever and invent a password that is entered with the CONTROL key so that it doesn't print, if you want to. Up to six digits or characters may be used in your password. The system manager is the only other person who needs to know what your password is.

The dash (—) after HELLO and the comma (,) between the user name and password must be typed just as shown in the example. If you forget to include them, you won't be logged on.

Now, using your user name and password, try logging on for yourself. Be sure that your terminal is turned on and connected to the computer. Then press the carriage return key and line feed key to position the carriage and introduce yourself to the computer.



Once you've introduced yourself, the computer begins to talk to you. First, it may print the day, date, and time. Then most likely, it will tell you its name and give you some free advice about getting news messages that have been stored for terminal users. The last time I logged on, the computer said

```
THURSDAY, MARCH 6, 1975    9:35 A.M.

H P  CUPERTINO  DATA  CENTER

GET/RUN $SCOOP FOR NEWS, ETC.

HAVE A NICE DAY
```

Having logged on, you may either create a program, or get an old one already stored in the computer and run it. To get a program you use the command GET— followed by the name of the program you want. To run the program type RUN. That's what the line GET/RUN $SCOOP FOR NEWS, ETC. means in the computer message above. I could have typed

```
GET-$SCOOP
RUN
```

for news, but I didn't. If your computer has a similar feature, it will tell you as soon as you log on.

Try it!

* * * * *

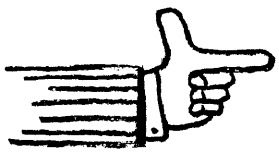
Whenever you'd like to terminate your working relationship with the operating system, just log off . . . by typing

```
BYE
```

That's all it takes.

You can log off any time (provided a program isn't running). Just enter the command BYE, turn off your terminal and go home. But remember, once you've logged off you cannot speak to the computer again without first logging on. And, logging off erases any programs you might have been using or creating. So you have to start from scratch when you log on again.

As far as starting from scratch and creating programs goes . . . read on



a program

A program is a set of statements. Each statement is an instruction to the computer. Each statement consists of one *line* of information and each is given its own *number*. Line numbers tell the computer the order in which the statements are to be performed. When you create a program, you enter statements (one at a time) at the terminal.

Okay, let's enter a program.

You're all logged on and ready to go — so type

```
SCR

10 PRINT 2+2
20 END

RUN
```

The computer then types

```
 4

DONE
```

That's it. You did it. You entered and executed your program. But to clarify exactly what happened, let's do it one more time, a bit more slowly.

SCR	The first thing you did was to type the characters SCR (followed by a carriage return). This command told the computer to scratch (delete) any existing programs or data in its memory to make room for the new program.
10 PRINT 2+2 20 END	Next, you typed your program, consisting of two lines. Line #10 said print the result of the arithmetic expression 2 plus 2. Line #20 said this is the end of the program.
RUN	Lastly, you executed or ran your program by typing the command RUN.
4	Performing the instructions in your program, the computer evaluated the arithmetic expression 2 plus 2 and printed the answer: 4.
DONE	The computer said it had finished with your program.

That is how the program was created and how it worked when it was executed.

Now, type RUN again.

```
RUN
4
DONE
```

The computer printed 4 again. That means that your program is still in memory within the computer and you can execute as often as you like.

Type SCR and then RUN and see what happens.

```
SCR
RUN
DONE
```

Nothing happens. Your program isn't in there any more. You scratched it by entering the SCR command. Now to execute your program, you must re-enter it. If you should log off, you'll also have to re-enter your program the next time you wish to use it. Logging off removes a program from memory as effectively as does typing SCR.

REVIEW TIME PS: The answers are in the back, if you need them.

1. The two commands we used were SCR and RUN. What did SCR tell the computer to do? How about RUN?

2. Our program consisted of the two lines
```
10 PRINT 2+2
20 END
```
What were the line numbers? Which line will always be executed first by the computer? What would happen if the line numbers were reversed? Or omitted? Try it and see.

3. 2+2 is an *arithmetic expression*. What did the computer do with it when the program was executed? What do you think would happen if the expression read 2−2? Try it and see.

4. See what happens if you put the arithmetic expression in quotes, such as

```
10 PRINT "2+2"
```

You have just converted the arithmetic expression to a string of characters. Strings enclosed in quotes are printed just as they are, as you just found out.

tricks of the trade

Now that you've written a program, there are other things that you can learn to do with it besides executing it. For instance, you can list the contents of a program, give it a name, or include notes among a program's statements.

Whenever you type the command LIST the computer prints your program for you right at the terminal. Each line in the program is printed in its proper sequence so that you can see exactly how the statement will execute. Take your program from the previous pages and type it back into the computer if you've scratched it, then type

```
LIST
```

the program is printed immediately by the computer.

```
10  PRINT 2+2
20  END
```

The LIST command helps you check out your program periodically as you are creating it, find just where you need to insert lines, locate unnecessary lines, and of course, permits you to just admire the product of your genius. It also shows the program re-ordered by statement number even if you entered statements out of sequence.

* * * * *

At any time during entry of your program, you may give it a name, if you like. A program's name is totally up to your imagination. It can be from one to six characters in length and may be composed of any digits or letters that print at your terminal. (In other words, the CONTROL key cannot be used for names.) A program's name may not contain any characters other than digits or letters.

Let's give your program an appropriate name — how about ADAM! Okay, we must use the command NAME— (yes, the hyphen must be included.) Type

```
NAME-ADAM
```

Done. Your program now has a name and can be referenced by that name for later use. We'll explain more about referencing a program later on in the manual. But here's a clue, it has to do with the command GET— (if you really can't stand the suspense, rush ahead to the topic "your own library" — but that really is cheating.)

As we've already said, a program is composed of statements that are executed by the computer when the program is run. But did you know that you can insert remarks in your programs without disturbing the statements that the computer will execute? You can! These remarks may relate to what the program is intended to do or to how the program is handling a certain problem. Or your remarks may simply be reminders about features to be added to the program or something you want to develop further at a later time. Remarks say whatever you want them to say and have nothing to do with the performance of the program. They are, however, given line numbers and are listed with the rest of the program by the computer.

All you have to do to enter a remark is to preface it with the statement REMARK. To illustrate, let's add a note to your original program.

```
5  REMARK*THIS IS MY FIRST PROGRAM
```

Now let's list the program with its new name and pertinent comment.

```
LIST

ADAM


5  REMARK*THIS IS MY FIRST PROGRAM
10  PRINT 2+2
20  END
```

Now let's see if it still runs the way it used to.

```
RUN
 4

DONE
```

Yes indeed. The very same results.

Writing notes within a program is called documentation and is regarded by many as an important aspect of programming. A properly documented program can be understood and used by any BASIC programmer; while an undocumented program can result in hours of frustration. Be sure to sprinkle ample notes in your more complex programs, either during or after the act of creating them. It will help you to remember what the program is supposed to do. And it will make it possible for other programmers to use your program if they need to.

operators

BASIC uses symbols to define certain types of operations. BASIC also expects you to use these symbols (called *operators*) when defining operations. Each symbol corresponds to one operation such as an addition, subtraction, division, etc. Each symbol can be found on your terminal keyboard.

Sometimes a combination of keys are used to type a symbol. You may need to press the SHIFT key as well as the key containing the symbol, or you may need to enter two symbols together such as <=.

See if you can locate the following keys on your terminal. If there is no key containing a certain symbol, ask someone if there is another symbol you should use instead of the missing symbol.

SYMBOL	DEFINITION	EXAMPLE OF USE	
=	equality	X=Y	A1=B2
+	add	2+2	X+Z
–	subtract	3–1	A–T
/	divide	4/2	Y/Z
*	multiply	9*3	M*N
#	inequality	Z#W	
< >	inequality	Z< >W	11< >23
>	greater than	4>3	Y>Z
<	less than	3<5	Z<Y
>=	greater than or equal to	Y>=X	X>=6
<=	less than or equal to	X<=Y	9<=Z
**	exponentiate	4**3	X**9
↑	exponentiate	4↑3	X↑9

Using the program we invented on page 6, try substituting some of the above operators and see what happens when you run the program. We already used the plus sign (+), why not try subtracting, multiplying, dividing, or using an exponent?
Experiment!

input

Data which is received by a program and processed in some manner is called *input*. Input can be received from terminal entry, from within the program itself or even from disc or tape files. As far as your programs are concerned, BASIC provides three different means of specifying input:

- ✔ The LET statement
- ✔ The INPUT statement
- ✔ The READ statement

Think of these statements as "doors" through which information can pass into your program for processing. Place these doors in your program wherever you want a data to be admitted. We will explore each input statement in detail in a moment. But first, look at the following program.

```
40 PRINT "X+Y=" X+Y
50 END
```

Go ahead and enter it. We will use the program in the discussion of input and will add our input doors to it as we master them. We will also run the program from time to time, to see what changes our input doors have wrought.

✔ The LET Statement

The LET statement is a handy way of inserting a value, or of grouping data under a common name. When used to insert data into a program, the LET statement tells the computer to *let* a variable equal the value you want.

To illustrate, if I want X to be equal to 27 in our program, I type

```
20 LET X = 27
```

(20 LET 27 = X won't work The symbol must come first.)

> We use the letters A,B . . . X,Y,Z to represent items whose values can change. These are variables.

And if I want Y to equal 8, I type

```
30 LET Y = 8
```

From this point on the letter X will be 27 and the letter Y will be 8 as far as my program is concerned.

Let's list our program now and see what it says.

```
LIST
20  LET X=27
30  LET Y=8
40  PRINT "X+Y="X+Y
50  END
```

Now let's run it

```
RUN
X+Y=  35

DONE
```

The program added the value of our two variables and printed the result.

Using the LET statement to group data under a common name works as follows. We simplify the PRINT statement by merging X+Y into a single character . . . the letter Z. This is done with the statement

```
35 LET Z = X+Y
```

Next we change the PRINT statement to

```
40 PRINT "X+Y=" Z
```

Now to see exactly what we did to the program, let's list and run it.

```
LIST
20  LET X=27
30  LET Y=8
35  LET Z=X+Y
40  PRINT "X+Y="Z
50  END

RUN
X+Y=  35
DONE
```

It worked! So much for the LET statement. On to something a bit more challenging.

The INPUT Statement

The INPUT statement permits you to enter data for your program from your terminal while your program is executing. Here's how you use it. Insert an INPUT statement in your program at the point at which you want the program to stop and ask for data. When the program is run, it stops as soon as it encounters the INPUT statement and types a question mark (?) at your terminal. Data must then be typed at the terminal before the program can proceed. Each INPUT statement can ask for one or more variables.

Let's insert two INPUT statements in our program, one to ask for the value of X and one to ask for the value of Y.

```
20 INPUT X
30 INPUT Y
```

Now, list the program.

*Notice that the new lines numbered 20 and 30 replace the old LET statements we had created as lines 20 and 30 earlier.

Run it and let's see what happens.

```
RUN
?
```

The question mark asks for the value of the variable X and is printed by the computer in response to line 20 in the program. Give X the value of 106.

```
? 106
?
```

Another question mark ? Right, we had two INPUT statements, remember! The second question mark is seeking the value of the variable Y (see line 30 in the program.) Type 94.

```
? 94
X+Y= 200

DONE
```

The computer added X and Y and printed the answer . . . 200.

There is another way to specify variables with the INPUT statement. Multiple variables can be requested by a single INPUT statement. For example, we could have used the statement

```
20 INPUT X,Y
```

When more than one variable is requested by a line, a comma must separate the variables.

In this case it is necessary to erase the now unnecessary line 30 by simply entering the line number and leaving the line blank.

```
30

LIST
20  INPUT X,Y
35  LET Z=X+Y
40  PRINT "X+Y="Z
50  END

RUN
?
```

See. Line 30 is gone.

This question mark requests the values of both variables X and Y. Therefore, you must enter both

```
?106,94      (don't forget the comma)
X+Y= 200

DONE
```

Once again our answer is printed by the computer.

Consider what would have happened if you had given only one answer, say 106, and then pressed RETURN. Try it.

```
RUN

?106
??
??94
X+Y= 200

DONE
```

This tells you that more input must be entered. If the INPUT statement expects two values, you must enter two values..

There are error messages or signals that appear sometimes when you are using the INPUT statement:

TRANSMISSION ERROR. REENTER	This message means that the value was not correctly transmitted to the system. You must type it again.
BAD INPUT, RETYPE FROM ITEM xx	This message tells you that the value typed (xx) was not acceptable for the variable in question. Type in the correct value, and any which followed it.
EXTRA INPUT, WARNING ONLY	This message is merely a warning. It means that extra values were included in the line. The computer ignores the extra characters, there is no need to re-type.

To get the last two error messages you had to violate some pretty tricky INPUT statement expectations. You see, the INPUT statement determines the type of data to be entered, as well as merely requesting input.

Up to this point, we have been requesting numbers (numeric data). But you can request ASCII (character) data just as readily by adding a dollar sign to the variable. To illustrate

```
20 INPUT X$
```

Variables may be called any letter from A through Z or A$ through Z$. Variables from AO$ to ZO$ and A1$ to Z1$ may also be used.

Modify our program to request an alphabetic variable. Type and run

```
20 INPUT X$
35
40 PRINT X$

RUN
?X

X
DONE
```

A variable can be more than one character, just as it can be more than one digit. But you'll have to read "strings and things" up ahead to find out how.

* * REVIEW * *

This is a good time to stop and review what we've learned about input up to this point.

(1) Three statements can be used to specify input. Name them.

(2) Type the program
```
40 PRINT A
50 END
```

Using the LET statement, make the variable equal to 1000.

Now do it using an INPUT statement instead.

Okay, put the value A in the variable.
(Remember the $ if you use an INPUT statement.)

So far so good! Let's try multiple variables now.

(3) Take the program
```
10 LET A = 30
20 LET B = 40
30 LET C =  5

50 PRINT D
60 END
```

Enter line 40 so that when run the program prints the answer 75.

(4) Now let's change the program to
```
10 INPUT A
20 INPUT B
30 INPUT C
40 LET D = A+B+C
50 PRINT D
60 END
```

Change this program so that all three variables are requested by one INPUT statement. Don't forget to erase the two unnecessary lines that result.

So much for LET and INPUT.

(5) What's the input statement we've yet to discuss?

The READ Statement

The READ statement has a partner — the DATA statement. Together these two make it possible for you to place data in your program while you create it, and then to instruct the program to read the data at a specific point during execution.

Here's how these statements work. Insert these lines in our original input program.

```
10 READ X,Y
20 DATA 25,50
```

and list and run the program.

```
LIST
10  READ X,Y
20  DATA 25,50
40  PRINT "X+Y="X+Y
90  END
RUN
X+Y= 75
DONE
```

The data stored in a DATA statement is read from the left to the right by the READ statement. Thus in our program X was equal to 25 and Y to 50, and the total as pointed out by the computer was 75.

Now, let's add another kind of statement to our program.

```
50 GO TO 10
```

This statement tells the computer to *go to* line 10. Okay, list the program.

```
LIST
10  READ X,Y
20  DATA 25,50
40  PRINT "X+Y="X+Y
50  GOTO 10
90  END
```

Try to follow the flow of the program in your head. Pretend you are the computer:

1a) READ X — okay get the first number from the data statement 25
b) READ Y — get the next number from the data statement 50
2a) PRINT "X+Y=" ..... X+Y=
b) X+Y — add 25 and 50 75
3a) GO TO 10 — go back to line 10
4a) READ X — get the next number from the data statement .....

WAIT! *HOLD IT!*

There isn't any next number in the data statement, we already used them all. In such a situation, the computer would print

```
OUT OF DATA IN LINE 10
```

and stop the program.

Now what? Well it is possible to add more data by adding more DATA statements, or to increase the amount of data in our original DATA statement. Let's try a little of both . . . type

```
20
60 DATA 25,50,75,100
70 DATA 300,200
```

DATA statements can be placed anywhere within the program. I prefer to cluster them all together at the end of my program so that I can find them easily. And that's what I've done here by deleting line 20 and adding lines 60 and 70. (You may keep the data next to the READ statements if you wish, or group them all at the beginning. Since they can go anywhere, the choice is really up to you.)

Okay, let's list and run the program.

```
LIST

10  READ X,Y
40  PRINT "X+Y="X+Y
50  GOTO 10
60  DATA 25,50,75,100
70  DATA 300,200
90  END

RUN


X+Y= 75
X+Y= 175
X+Y= 500

OUT OF DATA  IN LINE  10
```

Check the answers by following the flow of the program as we did above. Satisfied? Good!

There is another solution to the OUT OF DATA . . . situation. We could use a RESTORE statement to tell the computer to re-use the same data over and over. Let's try it. Type

```
45 RESTORE
60 DATA 25,50
70
```

We've added a RESTORE statement to our program and, for convenience sake, gone back to our previously limited amount of data. Now, find the BREAK key on your terminal (you're going to need it!) Then list and run the program.

```
LIST
10  READ X,Y
40  PRINT "X+Y="X+Y
45  RESTORE
50  GOTO 10
60  DATA 25,50
90  END

RUN

X+Y= 75
X+Y= 75
X+Y= 75
X+Y= 75
X+Y= 75
X+Y= 75
```

The only way out is to hit the BREAK key . . . so press it, quick!

```
STOP
```

There. We got out of that.

Granted, in our simple program, RESTORE did nothing more than create a loop from which we had to extract ourselves rather awkwardly. But as we progress through the manual, the significance of this and other statements will become clear.

TEST YOUR KNOWLEDGE

(1) A READ statement works with a _____ statement.

(2) Data in a DATA statement is read from right to left. T or F

(3) How do you separate multiple variables in a READ or DATA statement?

a) by a dash
b) by a period
c) by a comma

Assignment:

Write a program that acquires data by using all three types of input statements.

a sense of direction

Remember our experience with the GO TO and RESTORE statements. We got into an endless loop from which the program could not escape. Well, it is possible to create useful loops in programs, and to insert controls within the program which terminate such loops at the appropriate moment. This is done with the IF . . . THEN statement.

Take the program

```
10 LET X = 5
20 PRINT X
30 LET X = X+5
40 GO TO 20
50 END
```

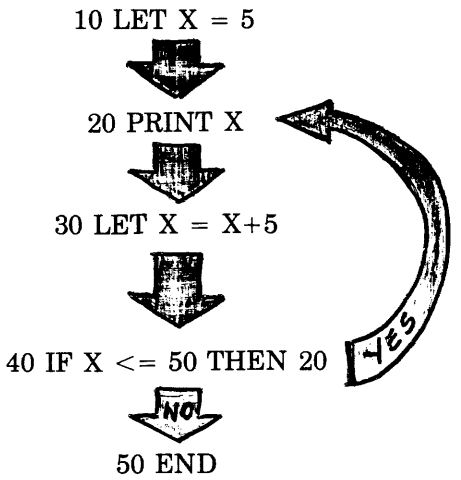
If we ran the program as it stands, we would once again find ourselves in an endless loop . . . in this case, counting by 5's ad infinitum.

But if we substitute an IF . . . THEN statement for the GO TO statement, we set an upper limit on that counting process and, in effect, tell the program when to stop counting.

Let's try counting to 50 by adding

```
40 IF X< = 50  THEN 20
```

Here's how the statement works. It tells the computer to make a decision based on the value of X, and where to go as a result of that decision. To illustrate:



If the value of X is 50 or less, go to line 20

If the value of X is *not* 50 or less go to next statement.

Okay, let's run our program and see if it functions as anticipated.

```
RUN
 5
 10
 15
 20
 25
 30
 35
 40
 45
 50

DONE
```

Sure enough.

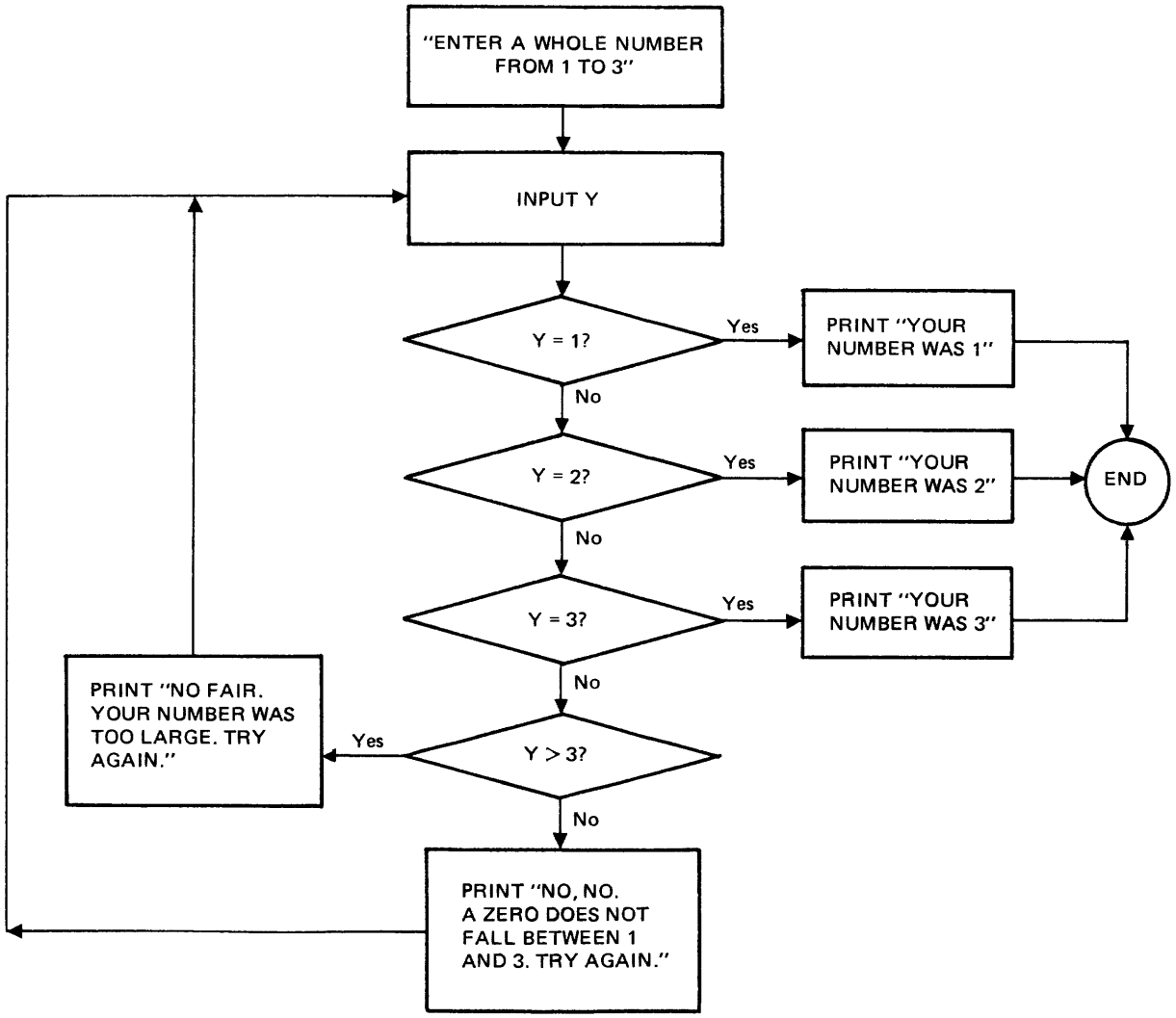
We can use other operators (page 11 remember) in our program if we wish. Try using the symbols for less than or greater than in the IF . . . NEXT statement and see if you can predict the results.

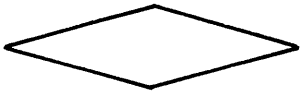
* * * * *

"What about using GO TO? Didn't we just obsolete it with the IF . . .THEN business?" Well you might ask. But let me hasten to assure you that such is not the case. The GO TO statement is still a valid and, indeed, useful statement. Let me show you how both GO TO and IF . . . THEN can be used together to create a program.

```
5  PRINT "ENTER A WHOLE NUMBER FORM 1 TO 3"
10 INPUT Y
15 IF Y = 1 THEN 45
20 IF Y = 2 THEN 55
25 IF Y = 3 THEN 65
30 IF Y > 3 THEN 75
35 PRINT "NO. A ZERO DOES NOT FALL BETWEEN 1 AND 3.  TRY AGAIN."
40 GO TO 10
45 PRINT "YOUR NUMBER WAS 1"
50 GO TO 90
55 PRINT "YOUR NUMBER WAS 2"
60 GO TO 90
65 PRINT "YOUR NUMBER WAS 3"
70 GO TO 90
75 PRINT "NO FAIR. YOUR NUMBER WAS TOO LARGE. TRY AGAIN."
80 GO TO 10
90 END
```

If you enter and run this program, you will discover that the one way to make the program come to a normal end is to do as it says and enter a 1, 2, or 3 when it tells you to. Entering any other whole number will send you back to the beginning of the program. The program's processing flow is cleverly depicted below.





is a decision box. It indicates that the computer must decide *yes* or *no* to a situation, and proceed accordingly.

is an operation box. It indicates that the computer must perform the task stated in the box and then proceed to the very next box (operation).

P.S. Flow charts are invaluable programming aids. They help you organize your thoughts before writing a program — and they help you figure out what a program's doing (if there's any doubt) after you've created it. Get into the habit of using flow charts. You'll be glad you did.

* * * * *

This seems a likely spot to introduce yet another new concept . . . the computed GO TO statement. A computed GO TO frees you from repetitive GO TO statements by letting the computer do some of the work of directing a program.

Look at the program on page 21 again. We are going to replace lines 15, 20, and 25 with one line — a computed GO TO statement. So instead of this

```
15 IF Y = 1 THEN 45
20 IF Y = 2 THEN 55
25 IF Y = 3 THEN 65
```

we'll have this

```
20 GO TO Y OF 45,55,65
```

which means the very same thing:

If Y is 1, go to the first number, in this case 45.
If Y is 2, go to the second number, 55.
If Y is 3, go to the third number, 65.
If Y is none of these, continue with the next statement.

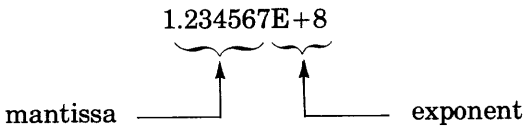
Don't forget to erase lines 15 and 25 when you insert the computed GO TO statement. A final hint, you'll see this type of statement again, at the end of the book. Don't let it take you by surprise.

scientific notation

Scientific notation is the way in which the computer expresses very large or very small numbers. In scientific notation a number is expressed by a *mantissa* and an *exponent.*

The mantissa presents the number as a decimal such as 1.234567. The computer can use up to seven digits from the mantissa.

The exponent defines the number of decimal places in the number. The exponent is separated from the mantissa by the letter E and the + or − sign of the exponent. If the exponent is positive (+) then the number is greater than 1.0. The exponent E+10 would be given to a very large number. If the exponent is negative (−) then the number is less than 1.0. The exponent E−6 would be given to a very small number.



* * EXAMPLES * *

The number 1,000,000,000,000 is written 1.000000E+12 in scientific notation. The exponent E+12 indicates that the decimal point is moved 12 positions to the right to produce the number.

The number .0000000672 is written 6.720000E−8 in the scientific notation. The exponent E−8 means that the decimal point is moved 8 positions to the left to make the number.

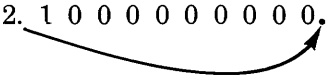
The number 987,654,321 is written 9.876543E+8 in scientific notation. The exponent indicates that there are 8 digits to the left of the decimal. Notice that the last two digits (21) were dropped by the computer. This means that zeros will be used for these digits. BASIC allows for only seven significant digits. So the closest it can come to the original number is 987,654,300.

Now let's practice converting numbers from scientific notation.

Conversion #1: 2.100000E+10

First write the number without the exponent: 2.100000

Now move the decimal to the right the number of places indicated by the exponent. 2. 1 0 0 0 0 0 0 0 0 0.



Re-write the number adding appropriate commas and that's all there is to it. 21,000,000,000

! ! ! Questions! Had the exponent been negative, in which direction would we have moved the decimal point? ? ? ? ? ?

Conversion #2: 9.100260E−2

First write the number without the exponent: 9.100260

Now move the decimal to the left the number of places indicated by the exponent. 0 0.0 9.1 0 0 2 6 0

Re-write the number without unnecessary zeros and that's all there is to it. .0910026

? ? ? Questions! Had the exponent been positive,in which direction would we have moved the decimal point? ? ? ? ? ? ? ? ? ? ? ? ?

By now no doubt you've go the general idea. If you'd like to test your skill, however, try the following

(1) Convert 3.094671E+1

(2) Then convert 3.094671E−1

Compare the answers.

Okay. Go the other way now. Convert numbers into scientific notation.

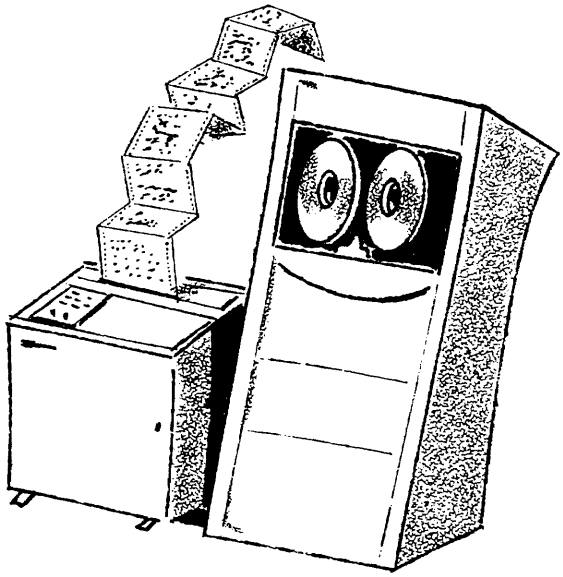
(3) Try the number 2,456,902,000,120,660,000

(4) Or .00000000000000000000000000001

Remember what the mantissa is, what the exponent does, and why the exponent is positive or negative. Those are the main points.

output

Output is what your program gives back to you at the end of or even in the midst of execution. Output is literally anything printed by the program; the ultimate result of your programming genius . . . or, stated more simply . . . output is "the answer" you went looking for in the first place.



The output statement with which we shall be concerned is the PRINT statement. The PRINT statement tells the computer to type a *value* at the terminal. A value may be a number or a word, a letter or a sentence, or the result of an expression. The PRINT statement also monitors the spacing of the values that it prints across the page at the terminal.

If a single value is involved, it is printed and the terminal automatically moves to the next line. Remember our first program?

```
10 PRINT 2+2
20 END
```

That's what it did. It simply printed the result of the expression 2+2 and advanced to the next line.

Well, it is also possible to use multiple values in a PRINT statement, provided they are separated by commas or semicolons.

```
10 PRINT 5,10,15,20,25
20 END
RUN
 5          10          15          20          25

DONE
```

This is the output

NOTICE that when a comma is used to separate the values in a PRINT statement, they are widely spaced across the page, beginning in columns 0, 15, 30, 45 and 60 unfailingly. Also note, 5 values on one line are the limit allowed with commas as separators.

Now then, try semicolons.

```
10 PRINT 5;10;15;20;25;30;35;40
20 END

RUN

 5       10      15      20      25      30      35      40

DONE
```

Two things changed. First, more values could be used (up to 12) and second the values were printed more concisely across the page.

When a value is enclosed in quotation marks, it is printed as is. Quotation marks are used primarily for alphabetic characters and special characters . . . not numbers. To illustrate

```
10 PRINT "COLUMN A", "COLUMN B","COLUMN C"
20 PRINT 120, 230, 820
30 PRINT 4.690,1,333.9
50 END

RUN

COLUMN A        COLUMN B        COLUMN C
 120             230             820
 4.69            1               333.9

DONE
```

"Ah ha! So that's how one adds titles and column headings." You're so right.

Now that you think you've got it, let me show you a few more fancy maneuvers. To print a blank line, just say PRINT. Let's add a blank line to our program.

```
15 PRINT
```

Or to control the placement of values across a line, use the control functions TAB, SPA, or LIN. TAB causes the carriage to *tab* over to the column specified. Change line 20 in our program to read

```
20 PRINT TAB (2);120;TAB(19);360;TAB(34);820
```

Note the placement of semicolons.
They must follow each TAB and value.

SPA causes the carriage to skip the number of columns specified. Change line 30 to

```
30 PRINT SPA(3),4.69;TAB(21);1;TAB(34);333.9
```

SPA uses a comma or a semicolon.

LIN generates a carriage return and, if specified, linefeeds. LIN(0) produces a single carriage return and zero linefeeds. While LIN(10) produces a single carriage return and 10 linefeeds. Add

```
35 PRINT LIN(2)
40 PRINT "THAT'S ALL"
```

Okay, list and run the program. Let's see exactly what we've done.

```
LIST
10  PRINT "COLUMN A","COLUMN B","COLUMN C"
15  PRINT
20  PRINT TAB(2);120;TAB(19);360;TAB(34);820
30  PRINT SPA(3),4.69;TAB(21);1;TAB(34);333.9
35  PRINT LIN(2)
40  PRINT "THAT'S ALL"
50  END

RUN

COLUMN A        COLUMN B        COLUMN C
  120               360             820
    4.69              1             333.9


THAT'S ALL

DONE
```

Did you anticipate the results? If you were fooled, go back and compare what you thought would happen with what really occurred. If you knew it all along, write a program that produces the following results (with the first decimal point in column 27).

```
      ASSETS      LIABILITIES

 40,000.00          5,000.00
    375.00         10,000.00
 15,000.00         33,000.00
    300.00          7,675.00
 ---------         ---------
 55,675.00         55,675.00
```

A SHORT NOTE

Writing a program by designing the output is *not* really working backward as you might initially have thought. Ninety-nine percent of all programming is accomplished just that way. You start with the output you want, evaluate the input you have, and devise a means of converting your input into output. The means you devise is, of course, the program.

There are several programming aids that are worth introducing here: coding sheets and formatting forms. Coding sheets enable you to write down all the things you need to tell the computer before you are actually sitting at the terminal. Formatting forms come into play even earlier. They are used to design the output (report usually) before you even begin coding.

To review then . . . the recommended sequence of events for creating a program is:

1. Design your output. Lay it all out on a formatting sheet, deciding on columns, headings and the like at this point.

2. Flow chart. Outline your plan of action, going from each bit of available input to each point of final output. Make your decisions here. Find the shortest and surest path for each item.

3. Code up your brilliant plan using the appropriate BASIC statements. This is the easy part.

4. Enter your program at the terminal. Then list and run it, and if necessary correct your coding or entry errors. That's all there is to to it.

By now you should have the program you set out to create. And it should execute just as you anticipated. There now, wasn't that easy?

other tricks

The FOR . . . NEXT connection begins with a FOR statement and ends with a NEXT statement. The FOR . . . NEXT connection creates controlled loops in which a group of statements can be intentionally repeated until a certain condition is reached. Relax, it's easier than it sounds.

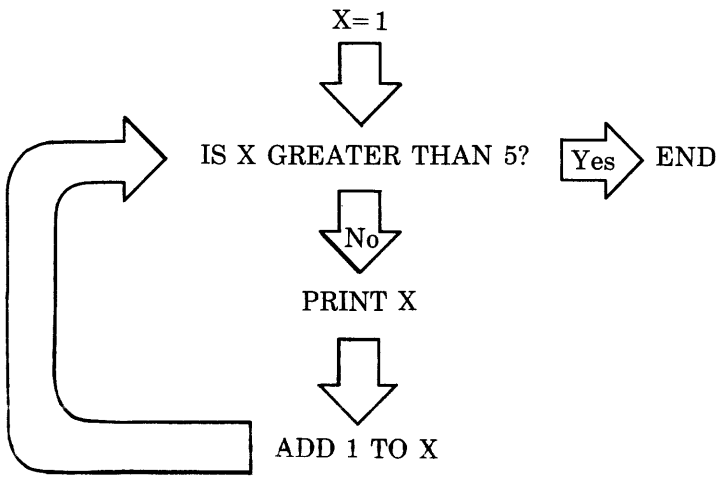
This is how it works.

```
10 FOR X = 1 TO 5
20 PRINT X
30 NEXT X
40 END
```

Line 10, the FOR statement, sets the initial value of X (in this case 1) and the ultimate value that X is to reach (in this case 5). Line 30, the NEXT statement, increases the value of X by 1 each time it is executed. The loop that is created by these statements is the continuous printing of X (line 20) until the value of X reaches 5.

```
RUN

 1
 2
 3
 4
 5

DONE
```

Or to depict the loop graphically



Got the idea? Here's a good spot to go back and review the section on IF . . . THEN statements. (Look under the title "a sense of direction".) You will notice that the loops involved in both cases are similar in result, but different in construction. There's no need to pick a favorite. Both IF . . . THEN and FOR . . . NEXT statements serve valid functions, and each is suited to a particular situation.

Pick the kind of statement best suited to the following applications. NOTE: the statement best suited is the one easiest to code.

ASSIGNMENT #1

Write a program that will count from 1 to the number you will supply in response to an INPUT statement.

ASSIGNMENT #2

Write a program that will compare two numbers and print the lowest of the two. One of the numbers to be compared must be supplied in response to an INPUT statement. The other number will, of course, be identified within the program.

FOR . . . NEXT with STEP

No, you do not always have to increment a value by 1 in a FOR . . . NEXT situation. You may increment by any number you wish . . . 4's . . . 13's . . . 92's . . . whatever. All you have to do is add the word STEP and the value of the increment to your FOR statement.

If we wanted to use our program to count by ten's instead of one's, we would say

```
10 FOR X = 10 TO 50 STEP 10
20 PRINT X
30 NEXT X
40 END

RUN

 10
 20
 30
 40
 50

DONE
```

That's all. Compare this program with the one on page 29, then try using STEP by yourself. Try counting by 4's, 13's, and 92's as promised above. I'm sure you'll get the idea.

more about operators

We've already discussed the symbols BASIC recognizes as operators. But you should know that there are other operators — in the form of words. You should also be aware of the fact that the computer processes operators in a certain order . . . the order of precedence.

First, let's define the new operators: MIN, MAX, AND, OR, and NOT.

Take the statement

```
100 LET C = A MIN B
```

What it actually says to the computer is "Let C equal the value of the minimum (or lowest) number either A or B." Thus if A equals 8 and B equals 6, C would be made equal to B (6) as it has the lowest value.

What do you think the next statement means?

```
100 LET C = A MAX B
```

You guessed it. It says to the computer "Let C equal the maximum (or highest) number either A or B. In this case, C would be equal to A, as its value of 8 is greater than B's value of 6.

The operators AND and OR form logical connections between two expressions. This is how they are used.

```
50 IF X<8 AND Y>5 THEN 100
```

This statement says "Go to line 100 if both conditions are true. The conditions of the statement are: if X is less than 8 and if Y is greater than 5." Thus, if both conditions are found by the computer to be true, statement 100 is executed next. But if only one, or if neither condition is found to be true, the computer executes the following statement next.

The following statement has one important difference from the one we just examined.

```
50 IF X<8 OR Y>5 THEN 100
```

In this case, the computer will go to line 100 if *either* condition is true: if X is less than 8 or if Y is greater than 5. Only if both conditions are false, does the computer proceed to the following statement.

So what have we proved? First that AND means "both" and second, the OR means "either", at least when one is speaking to a computer.

NOT is the negative member of the operator family, and as such works in mysterious ways. The form used for the operator is

```
25 IF NOT A THEN 35
```

meaning if A equals zero go to line 35, *but* if A is not equal to zero go instead to the following statement. We use the NOT operator to see if A is equal to zero. When A is zero the NOT operator is "true".

You need not comprehend precisely how NOT works its magic. Remember it is merely as a test for zero. If the variable does equal zero you go to the THEN statement. If the variable is not zero, you fall through to the next statement.

Yes, any variable may be used with NOT, we use A only for convenience. No, it is not an easy concept to understand.

Now to the matter of precedence and a couple of very important facts. Fact #1: More than one operator may be used in an expression. Fact #2: The order in which the operators are executed is determined by the precedence levels assigned to the operators:

PRECEDENCE LEVELS

highest	* * (or ↑)	(first performed)
	* /	
	+ −	
	MIN MAX	
	# = < > > < >= <=	
	AND	
lowest	OR	(performed last)

If two operators in an expression are of the same level, the order of execution is strictly left to right within the expression. To illustrate:

A+B−C or 5 + 6 − 7 is evaluated 5 + 6 − 7 = 11 − 7 = 4
A/B*C/D or 7/14*2/5 is evaluated 7 ÷ 14 × 2 ÷ 5 = .5 × 2 ÷ 5 = 1 ÷ 5 = .2
A MIN B MAX C MIN D is evaluated ((A MIN B) MAX C) MIN D

If you wish to alter the order of execution, you may do so by using parentheses to qualify the expression:

Let's say you want the sum of A and B subtracted from the sum of C and D. Then say
(C+D)−(A+B) Do not say A+B−C+D

Or suppose you wanted the sum of A and B multiplied by the result of D divided by C. Then say

(A+B)*(D/C) Do not say A+B*D/C

"But" you ask nervously. "What if the expression contains operators of differing precedence levels?"

The rule as applied here is two-fold. Operators are executed according to precedence levels *regardless* of their position in an expression. And, operators enclosed in parentheses are performed before those outside of the parentheses. To illustrate

C MIN A*B or 1 MIN 2*3 is evaluated $2 \times 3 = 6$ MIN $1 = 1$

while (C MIN A)*B or (1 MIN 2)*3 is evaluated 1 MIN $2 = 1 \times 3 = 3$

You need not learn the order of precedence by rote to do BASIC programming. You need only be aware of its existence, and refer to it as you generate arithmetic expressions.

Try a few expressions of your own, and see what difference parentheses make to the final outcome. Or, if you find it all hard to believe, program a few expressions in different ways, and let the computer prove it to you.

your own library

Your user number and password, aside from making it possible for you to log on, bestow upon you the convenience of your own little domain within the computer. This domain assigned exclusively to you is called your *library*.

You may store, modify, or remove programs at will within your library. In fact, there are three BASIC commands available to you which deal exclusively with accessing items in your library.

SAVE

Suppose you've written a program that you wish to use again the next time you log on. Well, you can store that program in your library by typing the command SAVE. The SAVE command requires nothing more than that the program be named and currently present in the computer.

Remember our first program, ADAM. The whole process of creating and storing the program in your library would go

```
SCR

10 PRINT 2+2
20 END

NAME-ADAM

SAVE
```

There. The program (or actually a copy of the program) is stored in your library now and will remain there until you delete it.

So, let's assume the time has come to use the program again. "How do we get it out of the library?" you ask. With the GET command. The GET command and the program's name, that is.

Try it. Type

```
SCR
```
to clear the area.

```
RUN
DONE
```
see nothing happens. Nothing's there — you just erased everything. But don't worry, the program's tucked safely away in your library.

```
GET-ADAM

RUN
```

```
 4
```
The program speaks. The GET command obtained a copy of the program from the library, and the RUN command executed it.

```
DONE
```

Yes, the hyphen must be used with the GET command. Don't omit it, or the computer will not know what to do.

PURGE

Once you have finished with a program forever, you can delete it from the library by typing the command PURGE— and the program's name.

Thus, to purge the program ADAM, we would type

PURGE—ADAM

The hyphen is, of course, required. And PUR— works as well as the whole word PURGE, if you prefer.

DELETE

To delete portions of a program rather than purging the whole thing,simply specify, the line number of the first statement to be deleted — and everything after that line will be erased.

"Ah," you say. "But, suppose I only wish to delete a middle section, or even just one line of the program . . . then what?" In such a case, you must furnish the first and last line numbers of the statements to be erased.

For example to delete lines 23 through 110 you'd type

DEL—23,110

In this instance where only one line is to be deleted, then that line number is both the first and last line number. To delete line 65, type

DEL—65,65

Got the idea?

CATALOG

After you've accumulated a whole sheaf of programs and files in your library, you may need to refresh your memory from time to time as to exactly what you do have stored. In this situation, the command CAT (short for catalog) can be used to obtain an alphabetic list of items in the library. To illustrate

CAT ← You type this and

all this happens automatically

NAME	LENGTH	RECORD	NAME	LENGTH	RECORD	NAME	LENGTH	RECORD
NANCY	2		PRNTR	66		TSTTRY	18	
WYYTC	24		VNAME	33				

As you no doubt noticed, the list of program names is accompanied by additional information about the items listed. To be specific, this additional information is a descriptive code*, and the length of the programs in blocks**. I will not define this code or the term *block* for you here. They are described in the reference manual. Unless you are at the stage of doing clever things with program access, the code will be omitted from your program catalog anyway. And, if you are using access or programming tricks, you are obviously using the reference manual as well.

So for our purposes, your program catalog will probably contain nothing more than the names of your programs:

```
CAT

NAME   LENGTH  RECORD    NAME   LENGTH  RECORD    NAME   LENGTH  RECORD

ADAM       19            B1        12            B2         6

BLOCK      14            CALC      31
```

files

From time to time you may find it convenient to save the results of one program for use as input to another program. This is done by means of BASIC files. You can think of computer files as simply invisible versions of the filing cabinets in your office. . . drawers containing information stored for later use. Only instead of drawers we use our library and instead of folders we use records.

To use BASIC files you must know how to use the following commands and statements:

- CREATE—
- PRINT
- READ
- PURGE—

The first thing you must do is to create a file. When you create a file, you give it a name and a length, and place it in your library for storage. Let's create a file two records long called FILEA. Type

```
CREATE-FILEA,2          Yes, the hyphen and comma are required!
```

Okay, now we have a file in our library called FILEA and capable of holding two records. It's empty at this point, but we'll soon fix that. Now, we are going to write two programs. . . one to write data into the file, and a second to read data from it. For the first program, type

```
10 FILES FILEA
20 LET X =9
30 LET Y = 11
40 PRINT #1; X+Y
50 PRINT #1; Y-X
60 END
```

The files statement identifies the file to receive the data.

The #1 added to the print statements tells the computer to print to the *first file* in the FILES statement. We defined only one file in the FILES statement, but we could have listed as many as 16.

Run the program

```
RUN

DONE
```

What? No output? Right, we wrote on the file not the terminal, remember?

I'll prove it to you with the second program. Type

```
10 FILES FILEA
20 READ #1; A,B

30 PRINT A+B
40 END
```

Now we've said print the sum of A and B. Where will it print? On the terminal, of course, we didn't specify #1 file now did we?

Let's see what happens.

```
RUN
22

DONE
```

What we said here is: read the data in the first file (#1), call the data items A and B.

Voila! That, essentially, is how you create and use files with your programs. To be sure, there's a lot more to the subject of files and when you feel you are ready or at least that you absolutely have to know more, rush to your reference manual for details.

Once you have finished with a file, you should remove it from your library. You do this by purging the file. Simply type the command PURGE, a hyphen, and the name of the file you want removed. To remove our file above, we would type

```
PURGE-FILEA
```

That's all there is to it.

strings and things

A string consists of up to 255 characters enclosed by quotation marks. Thus, a "string" in computerese is really a string of characters.

```
100 PRINT "THIS IS A STRING"     and so are
 20 PRINT "2+2="
 50 PRINT "X+Y="
 75 PRINT "YOUR NUMBER WAS 3"
```

Remember? We've been using strings in our PRINT statements all along . . . for headings and such. And we already know that any character (except quotation marks) can be used in a string. But there's more to strings than that.

BASIC contains special features which enable you to manipulate strings in a variety of useful ways. To understand these features, we must also understand string variables and substrings.

STRING VARIABLE

A string variable is used as a type of shorthand in BASIC programs. In other words, string variables are used to represent something else — usually a lengthy phrase or number. Remember on page 15, we talked about inserting a dollar sign next to a variable. Well, in so doing we created a string variable . . . A$,B$,C$, etc.

This is how a string variable works.

```
100 DIM A$(25)
110 A$ ="SAMPLE STRING"
```

Line 100 defined A$ as a variable capable of holding up to 25 characters.
Line 110 said that the contents of the variable are to be the characters SAMPLE STRING.

"But that's only 13 characters," you point out, astutely counting the blank space between SAMPLE and STRING as a character.

True! The 25 is the maximum limit of our variable — the minimum for any string is automatically zero. (Automatic means that we don't have to tell the computer. It already knows.) Any number of characters up to the maximum may actually be used.

The nice thing about our string is that it can be called A$ whenever we have to talk about it in our program. And A$ is a lot less trouble to write out than is "SAMPLE STRING".

TIMELY ASIDE: Any string of more than one character must be defined by a DIM statement. The format of the DIM (short for dimension) statement never varies. It is always DIM followed by the string's name and, in parentheses, the string's allowed maximum size.

"So, now that I've created one, what can I do with my string variable?"

I'm glad you asked, because there's a lot you can do with a string variable. You can have the program read or print it. You can enter it, change it, or even break it down into smaller pieces and use the parts of it.

Reading, printing, and entering a string variable are simple enough tasks. We've already done so several times in this manual.

```
125 READ A$
```
reads the string variable A$ from a DATA statement.

```
150 PRINT A$
```
prints the string variable A$ on the terminal.

```
175 INPUT A$
```
used to enter the variable A$ from the terminal.

Remember? "Sure, but how about changing a string variable? That's simple, too. Simply re-describe the string's content in a later statement . . . such as

```
160 A$ = "NEW SAMPLE STRING"
```

Take care not to exceed the maximum length specified for the string when you change its contents.Maximum length cannot be changed, only content can. Let's try a change. Type

```
100 DIM A$(25)
110 A$ = "SAMPLE STRING"
120 PRINT A$
130 A$ = "NEW SAMPLE STRING"
140 PRINT A$
150 END
```

Let's run it, and see if the variable does indeed get changed.

```
RUN

SAMPLE STRING
NEW SAMPLE STRING
DONE
```

It worked, of course. So finally, what about breaking a string into parts? For this we create *substrings* by specifying the first and last characters of the full string which are to be included in the substring. To illustrate

```
 50  DIM X$[10]
100  X$="ABCDEFG"
200  PRINT X$[2,6]
300  PRINT X$[3,4]
350  END
```

```
RUN
BCDEF
CD
DONE
```

Now, let's take it a little more slowly and see exactly how that went.

`50 DIM X$(10)`	First, we defined a string variable named X$ and said it's maximum size was 10 characters.
`100 X$ = "ABCDEFG"`	Then we defined the string's content to be the letters ABCDEFG.
`200 PRINT X$(2,6)`	Here's our first substring. We said print a substring of X$ consisting of characters two through six (BCDEF).
`300 PRINT X$(3,4)` `350 END`	Another substring. We said print another substring of X$, this time consisting of characters three and four (CD).
`RUN`	Then we ran the program and got
`BCDEF`	the first substring
`CD`	and the second substring as requested.
`DONE`	the computer said it was finished.

The parentheses and commas used in the above statements are required as used. If you omit them, the computer will not understand the statements.

A single character in the substring statement specifies that all subsequent characters in the string are to be included in the substring. This means that the statement:

```
250 PRINT X$(4)
```

would result in the output

```
DEFG
```

if added to the above program. I think you're getting the point here. We will use string variables and substrings in our final exercise. We will also use functions,

so read on . . . . . .

functions

You will find functions much easier to use than to read about. Just remember that there are two types of functions — those which are predefined by BASIC and those which you may define for yourself. And remember that functions always return a numeric result. They may be used as arithmetic expressions or as parts of expressions, as you prefer, anywhere a number is appropriate.

Predefined functions can be used in place of known values. They tell the computer to

- find the absolute value of the expression (X) . . . ABS(X)
- give the integer value of the expression (X) . . . INT(X)
- find the square root of the expression (X) . . . SQR(X)
- give the current length of the string A$ . . . LEN(A$)
- etc.

There are many, many other predefined functions available with BASIC. But we'll just dwell on these for the moment.

Say at some point in a program, I need to print the length of a string. If I know the present length of that string to be 10 characters, I say

```
174 PRINT 10
```

But if I don't know what the length of the string will be at this point in the program's execution, I use the function LEN, telling the program to figure out the length of the string and then print it. Assuming the string's name to be A$, the statement would read

```
175 PRINT LEN(A$)
```

Let's try it out and see if it works. Type

```
150 DIM A$(20)
160 A$ = "FRED"          We defined the contents of A$ to be FRED or four characters, so when
170 PRINT LEN(A$)        we say print the length we should get the answer 4
180 END

RUN

 4                       We did get an answer of 4. The function works.

DONE
```

Suppose I need to know the square root of a number. The following function will find and print it for me.

```
10 INPUT Z
20 PRINT SQR(Z)
30 END
```

Now, I'll run the program and give it a number to work with.

```
RUN
?12999.6
 114.016
DONE
```

You can substitute the functions ABS and INT in the above exercise and see how they work by giving them appropriate input to work on. In brief then, a function is used when a value is not known in advance. The functions instruct the computer to determine the value for you.

Now that you know how a function is used, you can go to the reference manual and read about all of the predefined functions available to you. Just in case you don't find the very one you need, though, you'd better read on and learn how to define your own functions.

All you need to define a new function is a DEF (short for define) statement. In the DEF statement you *name* the function FNx(y) substituting any letter (A-Z) for the x and any variable for the y. Then you *equate* the function to the formula you want the computer to perform.

It will become clear, if we define one right away. So let's say

```
100 DEF FNZ(X) = X*X
```

The function just defined squares a number, that is — it multiplies a number by itself. That is what the formula X*X means.

We called the function FNZ and have used the X in our formula to hold the place of the variable. That is what FNZ(X) means.

Let's try it out. Type

```
50 INPUT A
100 DEF FNZ(X)=X*X
150 PRINT FNZ(A)          See. Here we supply the name of the real variable . . . A.
200 END
```

Now run it.

```
RUN
 ?
```

Enter the number 12.

```
?12
144

DONE
```

There's our answer . . . 144 the square of 12. Our function works as promised.

Just for fun, let's compare our squaring function with the predefined SQR function, using the square root function to check the accuracy of our squaring function.

Add the following statements to our program.

```
160 INPUT B
170 PRINT SQR(B)
```

Now run it again.

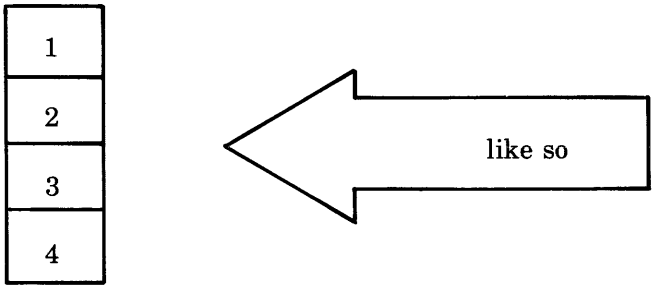
```
RUN

?12                Remember this part? We supply the 12 and it squares it to 144.
 144

?144               Now we supply the 144 and see what the square root is.
 12

DONE
```

You should experiment with defining functions for a while. Define the kind that you will be needing in your own programs. Be as creative and involved as you wish with the part to the right of the equal sign. I'm sure you will soon agree that functions are much easier to use than to explain.

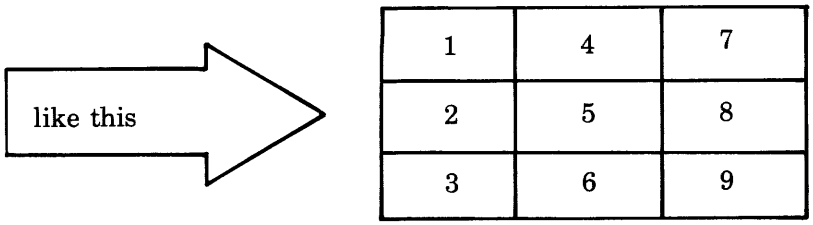
an array is

a group of data identified by a common name. The data in an array is divided into units called elements.

An array may consist of one dimension which provides a *list of elements*



Or an array may be two dimensional, providing a *table of elements*

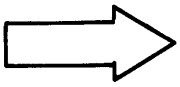


Each array must be given a one character name (A through Z). "Ah ha! A single character is also used to identify a varible." That's correct, an array is really just a fancy variable. And just like string variables a DIM (dimension) statement must be used to define an array.

Type

```
10 DIM A(4,3)
```

We just created an array named A having the dimensions 4 by 3. That's four rows by three columns.

Rows go across  Columns go down 

Now, let's get some data into that poor little naked array. Type

```
20 MAT READ A
30 DATA 1,2,3,4,5,6,7,8,9,10,11,12
40 MAT PRINT A
90 END
```

I'll bet you've figured out what's going to happen now. Right?

The MAT READ statement tells the computer to take the data from the DATA statement and place it in the array. (Surely you remember how a READ/DATA arrangement works.) And yes, the data is inserted into the array in a predictable sequence . . . row by row.

The PRINT statement will print out our array for us. So let's run the program and see how the array looks.

```
RUN
 1            2            3
 4            5            6
 7            8            9
 10           11           12
```

Compare the DATA statement and the sequence of data in the array. That is the order in which data will always be stored in an array, and that's important to know. "Why?" Well, because often we want to refer to a particular element in the array. And an element is identified by its position.

Let's say we want to print only the element containing the number 8. How would we tell the computer to do this? With a PRINT statement and with the element's dimensions (row and column). The 8 is in row 3 column 2, so we would say

```
40 PRINT A(3,2)
```

Okay, replace the first PRINT statement with the one above and run the program.

```
RUN
 8
DONE
```

Note: Use MAT PRINT to print the whole array, and use PRINT to print an element.

So what have we learned about arrays? We've learned some new statements — MAT READ and MAT PRINT. Or so it seemed. But think about it. These are actually the old familiar READ and PRINT statements with MAT added on in front. The MAT tells the computer that we are referring to an array. MAT stands for *matrix* which is another word for tabular array. (You might also find arrays called *subscripted variables* in some books. But an array by any other name, acts the same.)

You are encouraged to experiment with arrays and to read more about them in your reference manual.

ASSIGNMENT

What do you think would happen if we placed MAT at the front of an INPUT statement? Right, we would be able to input the array from our terminal.

Write a program to do just that.

subroutines

Once your programs begin to grow beyond the simple 5 or 10 line versions we've been using, you will find it invaluable to be able to divide them into handy parcels. This is the point at which the terms *main program* and *subroutine* appear in your vocabulary.

A main program, or more precisely the main part of a program, usually acts much like a traffic cop — providing directions to the various subroutines. Each *subroutine* is devoted to a special task such as preparing the heading, or is used to perform calculations or even store data.

Along with the concept of subroutines and main programs come two new BASIC statements: GOSUB and RETURN. These statements enable you to direct the computer to specific parts of the program and then back to the place from which it was directed. In other words, these statements are used to direct the computer from the main program to subroutines and back.

Let's see exactly how they are used in a real program. We are going to create a bill of sale which totals as many purchases as we wish to provide. To do this we will use a main program, three subroutines, and a data base. Watch closely, now. I'll ask questions at the end.

```
 5  REMARK*MAIN PROGRAM
 10 DIM B$(10)
 15 LET T=0
 20 PRINT "NUMBER OF TRANSACTIONS"
 25 INPUT N
 30 GOSUB 150
 35 GOSUB 50
 40 GOSUB 100
 45 END

 50 REMARK * REPORT BODY SUBROUTINE
 55 FOR X = 1 TO N
 60 READ A,B$,C
 65 LET D = A*C
 70 LET T = T+D
 75 PRINT A,B$,C,D
 80 NEXT X
 85 RETURN

100 REMARK * SUMMARY SUBROUTINE
110 PRINT "TOTAL CHARGES";T
120 RETURN

150 REMARK * HEADING SUBROUTINE
155 PRINT "QTY","DESC","UNIT PRICE","CHARGE"
160 PRINT
165 RETURN
```

```
200 REMARK * DATA BASE
210 DATA 6,"BOLTS",.15
220 DATA 15,"NUTS",.08
230 DATA 30,"SCREWS",.07
240 DATA 45,"NAILS",.03
999END
```

Now that you've entered it, run it and most of your questions about the program will be instantly answered . . . like "What does it do?" "How does it work?" etc.

```
RUN


NUMBER OF TRANSACTIONS
?4
QTY             DESC            UNIT PRICE      CHARGE

 6              BOLTS             .15             .9
 15             NUTS              .08             1.2
 30             SCREWS            .07             2.1
 45             NAILS             .03             1.35
TOTAL CHARGES 5.55

DONE
```

* * * * *

EXAM TIME, FOLKS!

Chart the flow of action in the subroutine program. You really need chart only the GOSUB and RETURN statements for our purposes here. But if you didn't understand some of the internals of the program, you'd better leaf back through the manual until you find the explanations you missed.

Replace the existing data base in the program with data of your own choosing. Remember you must identify the number of transactions at the beginning of the program. We used four originally; you may use as many as you wish.

express yourself

As our final exercise, you get a chance to express the feelings which no doubt are welling up in you now that you have finished the manual and are ready to pick up your BASIC programmer's pencil.

The last program affords you just that opportunity. In addition, it contains many of the statements and routines you've just finished learning about . . . strings, functions, operators, string variables, input statements, output statements, and subroutines. See how many of them you can spot as you enter the program for execution at your terminal. Try charting the flow of action in the program after you've run, paying particular attention to bringing the program to a normal end.

And of course, don't forget to use the program to express your feelings at this awesome moment. As you will soon discover, all complimentary remarks are graciously accepted. Anything else will illicit an appropriate computer reply.

THE PROGRAM

```
  5  DIM A$[72],B$[72]
 10  PRINT "*** COMMENT ***"
 20  PRINT "AT LAST, A CHANCE TO EXPRESS YOUR FEELINGS."
 30  PRINT "YOU MAY SELECT THE VERB AND OBJECT FOR YOUR"
 40  PRINT "STATEMENT. SIMPLY TYPE THE NUMBER OF THE WORD"
 50  PRINT "OR PHRASE YOU WISH FROM EACH LIST."
100  PRINT "PICK AN OBJECT:"
105  PRINT "1=SLEEPING,2=HP BASIC,3=PROGRAMMING,4=THIS MANUAL"
125  GOSUB 500
130  IF A=0 THEN 125
135  A2=A
136  PRINT "PICK A VERB:"
137  PRINT "1=HATE,2=DON'T UNDERSTAND,3=PREFER,4=LOVE"
140  GOSUB 500
145  IF A=0 THEN 125
150  A1=A
170  A$="I "
175  GOTO A1 OF 190,200,210
180  B$="LOVE "
185  GOTO 215
190  B$="HATE "
195  GOTO 215
200  B$="DON'T UNDERSTAND "
205  GOTO 215
210  B$="PREFER "
215  GOSUB 600
220  GOTO A2 OF 235,245,255
225  B$="THIS MANUAL."
230  GOTO 260
```

```
235  B$="SLEEPING."
240  GOTO 260
245  B$="HP BASIC."
250  GOTO 260
255  B$="PROGRAMMING."
260  GOSUB 600
265  PRINT A$
270  IF A2=1 THEN 290
275  IF A1>2 THEN 320
280  PRINT "WHERE HAVE I FAILED?"
285  GOTO 295
290  PRINT "WHAT DOES THAT HAVE TO DO WITH BASIC?"
295  PRINT "CARE TO TRY AGAIN";
300  INPUT A$
305  PRINT
310  IF A$[1,3]="YES" THEN 100
315  END
320  PRINT "THAT'S THE SPIRIT!"
325  END
500  INPUT A
505  IF A=INT(A) AND A >= 1 AND A <= 4 THEN 530
510  IF A=9 THEN 999
515  PRINT "SELECT ONE OF THE FOUR CHOICES."
520  A=0
530  RETURN
600  A$[LEN(A$)+1,LEN(A$)+LEN(B$)]=B$
605  RETURN
999  END
```

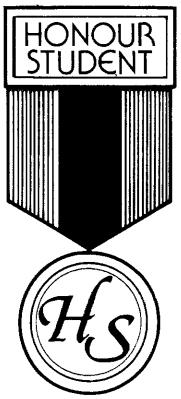
Once entered, the program is used to construct a simple three word sentence. The first word is always "I". The last two words are selected, by you, from a list provided by the program.

Okay, run it.

Here are some of the expressions selected by previous students.

```
RUN

*** COMMENT ***
AT LAST, A CHANCE TO EXPRESS YOUR FEELINGS.
YOU MAY SELECT THE VERB AND OBJECT FOR YOUR
STATEMENT. SIMPLY TYPE THE NUMBER OF THE WORD
OR PHRASE YOU WISH FROM EACH LIST.
```

```
PICK AN OBJECT:
1=SLEEPING,2=HP BASIC,3=PROGRAMMING,4=THIS MANUAL
?2
PICK A VERB:
1=HATE,2=DON'T UNDERSTAND,3=PREFER,4=LOVE
?2
I DON'T UNDERSTAND HP BASIC.
WHERE HAVE I FAILED?
CARE TO TRY AGAIN?  YES


PICK AN OBJECT:
1=SLEEPING,2=HP BASIC,3=PROGRAMMING,4=THIS MANUAL
?3
PICK A VERB:
1=HATE,2=DON'T UNDERSTAND,3=PREFER,4=LOVE
?1
I HATE PROGRAMMING.
WHERE HAVE I FAILED?
CARE TO TRY AGAIN?  YES


PICK AN OBJECT:
1=SLEEPING,2=HP BASIC,3=PROGRAMMING,4=THIS MANUAL
?1
PICK A VERB:
1=HATE,2=DON'T UNDERSTAND,3=PREFER,4=LOVE
?4
I LOVE SLEEPING.
WHAT DOES THAT HAVE TO DO WITH BASIC?
CARE TO TRY AGAIN?YES


PICK AN OBJECT:
1=SLEEPING,2=HP BASIC,3=PROGRAMMING,4=THIS MANUAL
?4
PICK A VERB:
1=HATE,2=DON'T UNDERSTAND,3=PREFER,4=LOVE
?3
I PREFER THIS MANUAL.
THAT'S THE SPIRIT!


DONE
```



the answers

page 7

1 The SCR command clears the computer in anticipation of new data.
The RUN command executes a program.

2 The line numbers are 10 and 20. The lowest line number is always executed first. If the line numbers are reversed, the END command would execute first. A program without line numbers will not be accepted by the computer.

3 The computer evaluated the arithmetic expression when the program was executed. The expression 2−2 would yield a result of 0.

4 The computer would respond with 2+2. Any characters or numbers enclosed by quotation marks are considered part of a string and are printed as entered.

page 16

1 The LET statement. The INPUT statement. The READ statement.

2
```
30 LET A = 1000
30 INPUT A
40 PRINT "A"      or  30 INPUT A$
                      40 PRINT A$
```

3
```
LET D = A+B+C
```

4
```
10 INPUT A,B,C
20
30
```

5 The READ statement.

page 19

1 DATA
2 F
3 C

page 24

1 30.94671

2 .3094671

3 2.456902E+18

4 1.000000E-30

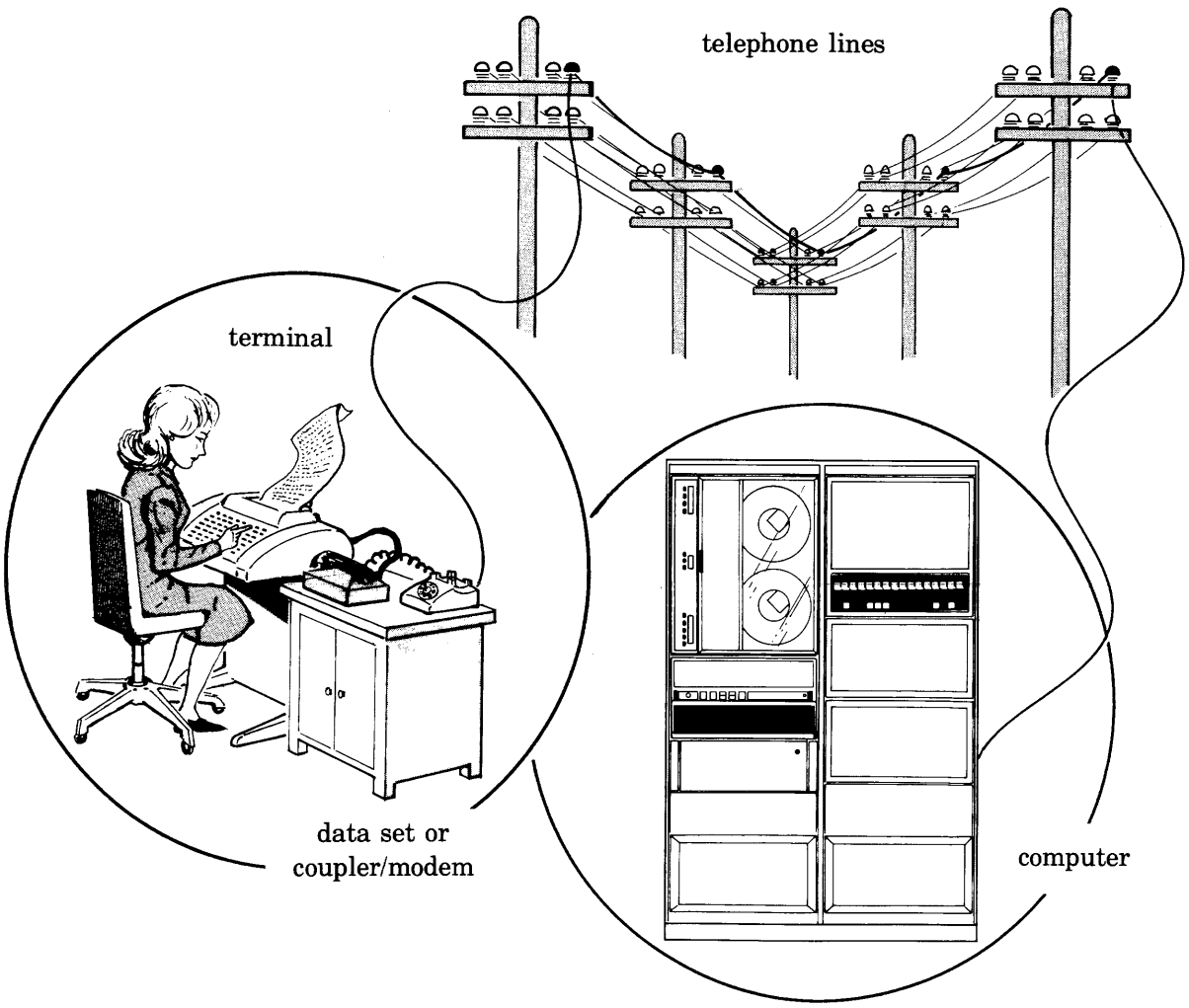
Page 27

Assignment solution

```
LIST

10  PRINT "                              ASSETS     LIABILITIES"
15  PRINT
20  PRINT SPA(20);"40,000.00";TAB(37)"5,000.00"
25  PRINT SPA(23);"375.00";TAB(36);"10,000.00"
30  PRINT SPA(20);"15,000.00";TAB(36);"33,000.00"
35  PRINT SPA(23);"300.00";TAB(37);"7,675.00"
40  PRINT SPA(20);"---------";TAB(36);"---------"
45  PRINT SPA(20)"55,675.00";TAB(36);"55,675.00"
99  END

RUN

                         ASSETS      LIABILITIES

                     40,000.00         5,000.00
                        375.00        10,000.00
                     15,000.00        33,000.00
                        300.00         7,675.00
                     ---------        ---------
                     55,675.00        55,675.00

DONE
```

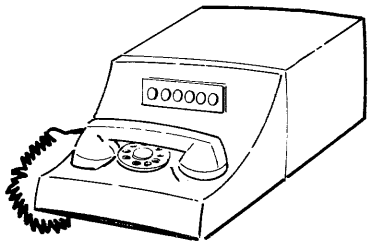
calling the computer

Terminals which are not connected directly with the computer are called remote terminals. Regular telephone lines are used to connect remote terminals with the computer.

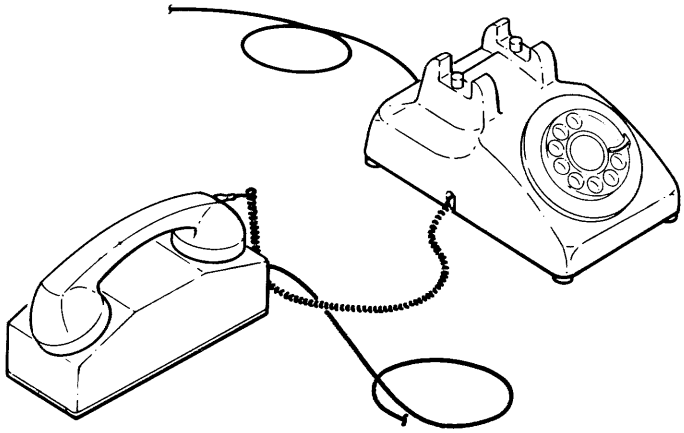


To establish the connection for your terminal, you must call the computer using the telephone number furnished by the system manager. This means that you must know how to operate your telephone connecting device as well as your terminal.

If your equipment is all in one unit, like this you have a *data set.*



If your equipment is in two pieces, like this you have a regular telephone and a *coupler/modem.*



HOW TO USE A DATA SET

1 Turn the terminal ON and set the switch to the ON-LINE position.

2 Press the TALK button on the data set.

3 Remove the receiver and dial the computer's telephone number.

4 When you hear a high pitched tone in the receiver, press the DATA button on the data set until it lights, and place the receiver in its cradle.

5 Log on, you're ready to go.

HOW TO USE A COUPLER/MODEM

1 Turn the terminal ON and set the switch to the ON-LINE position.

2 Set the coupler's power switch to ON.

3 Set the coupler's line switch (if there is one) to ON-LINE.

4 If your coupler has a duplex switch, set it to FULL.

5 Remove the telephone's receiver and dial the computer's telephone number.

6 When you hear a high-pitched tone, place the receiver into the coupler receptacle.

7 Log on, you're ready to go.

HEWLETT hp PACKARD

SALES & SERVICE OFFICES

AFRICA, ASIA, AUSTRALIA

AMERICAN SAMOA
Calculators Only
Oceanic Systems Inc.
P.O. Box 777
Pago Pago Bayfront Road
Pago Pago 96799
Tel: 633-5513
Cable: OCEANIC-Pago Pago

ANGOLA
Telectra
Empresa Técnica de Equipamentos Eléctricos, S.A.R.L.
R. Barbosa Rodrigues, 42-1°DT.°
Caixa Postal, 6487
Luanda
Tel: 35515/6
Cable: TELECTRA Luanda

AUSTRALIA
Hewlett-Packard Australia Pty. Ltd.
31-41 Joseph Street
Blackburn, Victoria 3130
P.O. Box 36
Doncaster East, Victoria 3109
Tel: 89-6351
Telex: 31-024
Cable: HEWPARD Melbourne

Hewlett-Packard Australia Pty. Ltd.
31 Bridge Street
Pymble
New South Wales, 2073
Tel: 449-6566
Telex: 21561
Cable: HEWPARD Sydney

Hewlett-Packard Australia Pty. Ltd.
153 Greenhill Road
Parkside, 5063, S.A.
Tel: 27-2591
Telex: 82536 ADEL
Cable: HEWPARD ADELAIDE

Hewlett-Packard Australia Pty. Ltd.
141 Stirling Highway
Nedlands, W.A. 6009
Tel: 86-5455
Telex: 93859 PERTH
Cable: HEWPARD PERTH

Hewlett-Packard Australia Pty. Ltd.
121 Wollongong Street
Fyshwick, A.C.T. 2609
Tel: 95-3733
Telex: 62650 Canberra
Cable: HEWPARD CANBERRA

Hewlett Packard Australia Pty. Ltd.
5th Floor
Teachers Union Building
495-499 Boundary Street
Spring Hill, 4000 Queensland
Tel: 29-1544
Telex: 42133 BRISBANE

GUAM
Medical/Pocket Calculators Only
Guam Medical Suply, Inc.
Jay Ease Building, Room 210
P.O. Box 8383
Tamuning 96911
Tel: 646-4513
Cable: EARMED Guam

HONG KONG
Schmidt & Co.(Hong Kong) Ltd.
P.O. Box 297
Connaught Centre
39th Floor
Connaught Road, Central
Hong Kong
Tel: H-255291-5
Telex: 74766 SCHMC HX
Cable: SCHMIDTCO Hong Kong

INDIA
Blue Star Ltd.
Kasturi Buildings
Jamshedji Tata Rd.
Bombay 400 020
Tel: 29 50 21
Telex: 2156
Cable: BLUEFROST

Blue Star Ltd.
Sahas
414/2 Vir Savarkar Marg
Prabhadevi
Bombay 400 025
Tel: 45 78 87
Telex: 4093
Cable: FROSTBLUE

Blue Star Ltd.
Band Box House
Prabhadevi
Bombay 400 025
Tel: 45 73 01
Telex: 3751
Cable: BLUESTAR

Blue Star Ltd.
14/40 Civil Lines
Kanpur 208 001
Tel: 6 88 82
Telex: 292
Cable: BLUESTAR

Blue Star Ltd.
7 Hare Street
P.O. Box 506
Calcutta 700 001
Tel: 23-0131
Telex: 7655
Cable: BLUESTAR

Blue Star Ltd.
Blue Star House,
34 Mahatma Gandhi Rd.
Lajpatnagar
New Delhi 110 024
Tel: 62 32 76
Telex: 2463
Cable: BLUESTAR

Blue Star Ltd.
Blue Star House
11/11A Magarath Road
Bangalore 560 025
Tel: 55668
Telex: 430
Cable: BLUESTAR

Blue Star Ltd.
Meeakshi Mandiran
xxx/1678 Mahatma Gandhi Rd.
Cochin 682 016 Kerala
Tel: 32069, 32161, 32282
Telex: 046-514
Cable: BLUESTAR

Blue Star Ltd.
1-1-117/1
Sarojini Devi Road
Secunderabad 500 003
Tel: 70126, 70127
Cable: BLUEFROST
Telex: 459

Blue Star Ltd.
23/24 Second Line Beach
Madras 600 001
Tel: 23954
Telex: 379
Cable: BLUESTAR

Blue Star Ltd.
Nathraj Mansions
2nd Floor Bistupur
Jamshedpur 831 001
Tel: 7383
Cable: BLUESTAR
Telex: 240

INDONESIA
BERCA Indonesia P.T.
P.O. Box 496
1st Floor JL. Cikini Raya 61
Jakarta
Tel: 56038, 40369, 49886
Telex: 42895
Cable: BERCACON

BERCA Indonesia P.T.
63 JL. Raya Gubeng
Surabaya
Tel: 44309

ISRAEL
Electronics & Engineering Div. of Motorola Israel Ltd.
16, Kremenetski Street
P.O. Box 25016
Tel-Aviv
Tel: 03-389 73
Telex: 33569
Cable: BASTEL Tel-Aviv

JAPAN
Yokogawa-Hewlett-Packard Ltd.
Ohashi Building
1-59-1 Yoyogi
Shibuya-ku, Tokyo
Tel: 03-370-2281/92
Telex: 232-2024YHP
Cable: YHPMARKET TOK 23-724

Yokogawa-Hewlett-Packard Ltd.
Nissei Ibaraki Building
2-8 Kasuga 2-chrome, Ibaraki-shi
Osaka,567
Tel: (0726) 23-1641
Telex: 5332-385 YHP OSAKA

Yokogawa-Hewlett-Packard Ltd.
Nakamo Building
24 Kami Sasajima-cho
Nakamura-ku, **Nagoya** , 450
Tel: (052) 571-5171

Yokogawa-Hewlett-Packard Ltd.
Tanigawa Building
2-24-1 Tsuruya-choo
Kanagawa-ku
Yokohama, 221
Tel: 045-312-1252
Telex: 382-3204 YHP YOK

Yokogawa-Hewlett-Packard Ltd.
Mito Mitsui Building
105, 1-chrome, San-no-maru
Mito, Ibaragi 310
Tel:: 0292-25-7470

Yokogawa-Hewlett-Packard Ltd.
Inoue Building
1348-3, Asahi-cho, 1-chome
Atsugi, Kanagawa 243
Tel: 0462-24-0452

KENYA
Technical Engineering Services (E.A.)Ltd.,
P.O. Box 18311
Nairobi
Tel: 557726/556762
Cable: PROTON

Medical Only
International Aeradio(E.A.)Ltd.,
P.O. Box 19012
Nairobi Airport
Nairobi
Tel: 336055/56
Telex: 22201/22301
Cable: INTAERID Nairobi

KOREA
American Trading Company Korea
C.P.O. Box 1103
Dae Kyung Bldg., 8th Floor
107 Sejong-Ro,
Chongro-Ku, **Seoul**
Tel: (4 lines) 73-8924-7
Telex: K-28338
Cable: AMTRACO Seoul

MALAYSIA
Teknik Mutu Sdn. Bhd.
2 Lorong 13/6A
Section 13
Petaling Jaya,**Selangor**
Tel: Kuala Lumpur-54994 or 54916
Telex: MA 37605

Protel Engineering
P.O. Box 1917
Lot 259, Satok Road
Kuching, **Sarawak**
Tel: 20262
Cable: PROTEL ENG

MOZAMBIQUE
A.N. Goncalves, Lta.
162, 1° Apt. 14 Av. D. Luis
Caixa Postal 107
Lourenco Marques
Tel: 27091, 27114
Telex: 6-203 Negon Mo
Cable: NEGON

NEW ZEALAND
Hewlett-Packard (N.Z.) Ltd.
4-12 Cruickshank Street
Kilbirnie, Wellington 3
Mailing Address: Hewlett-Packard (N.Z.) Ltd.
P.O. Box 9443
Courtney Place
Wellington
Tel: 877-199
Telex: NZ 3839
Cable: HEWPACK Wellington

Hewlett-Packard (N.Z.) Ltd.
Pakuranga Professional Centre
267 Pakuranga Highway
Box 51092
Pakuranga
Tel: 569-651
Telex: NZ 3839
Cable: HEWPACK,Auckland

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
Scientific Division
79 Carlton Gore Rd., Newmarket
P.O. Box 1234
Auckland
Tel: 75-289
Telex: 2958 MEDISUP
Cable: DENTAL Auckland

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
P.O. Box 1994
147-161 Tory St.
Wellington
Tel: 850-799
Telex: 3858
Cable: DENTAL, Wellington

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
P.O. Box 309
239 Stanmore Road
Christchurch
Tel: 892-019
Cable: DENTAL, Christchurch

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
303 Great King Street
P.O. Box 233
Dunedin
Tel: 88-817
Cable: DENTAL, Dunedin

NIGERIA
The Electronics Instrumentations Ltd.
N6B/770 Oyo Road
Oluseun House
P.M.B. 5402
Ibadan
Tel: 61577
Telex: 31231 TEIL Nigeria
Cable: THETEIL Ibadan

The Electronics Instrumentations Ltd.
144 Agege Motor Road, Mushin
P.O. Box 6645
Lagos
Cable: THETEIL Lagos

PAKISTAN
Mushko & Company, Ltd.
Oosman Chambers
Abdullah Haroon Road
Karachi-3
Tel: 511027, 512927
Telex: KR894
Cable: COOPERATOR Karachi

Mushko & Company, Ltd.
38B, Satellite Town
Rawalpindi
Tel: 41924
Cable: FEMUS Rawalpindi

PHILIPPINES
The Online Advanced Systems Corporation
Filcapital Bldg.
11th Floor, Ayala Ave.
Makati, **Rizal**
Tel: 86-40-81, ext. 223,263
Telex: 3274 ONLINE

RHODESIA
Field Technical Sales
45 Kelvin Road North
P.O. Box 3458
Salisbury
Tel: 705231 (5 lines)
Telex: RH 4122

SINGAPORE
Hewlett-Packard Singapore (Pte.) Ltd.
Blk. 2, 6th Floor, Jalan Bukit Merah
Redhill Industrial Estate
Alexandra P.O. Box 58,
Singapore 3
Tel: 633022
Telex: HPSG RS 21486
Cable: HEWPACK, Singapore

SOUTH AFRICA
Hewlett-Packard South Africa (Pty.), Ltd.
Private Bag Wendywood
Sandton, Transvaal 2144
Hewlett-Packard House
Daphne Street, Wendywood,
Sandton, Transvaal 2144
Tel: 802-104016
Telex: SA43-4782JH
Cable: HEWPACK JOHANNESBURG

Hewlett-Packard South Africa (Pty.), Ltd.
P.O. Box 120
Howard Place, Cape Province, 7450
Pine Park Center, Forest Drive,
Pinelands, Cape Province, 7405
Tel: 53-7955 thu 9
Telex: 57-0006

Hewlett-Packard South Africa (Pty.), Ltd.
P.O. Box 37099
Overport, Durban 4067
641 Ridge Road, Durban
Durban, 4001
Tel: 88-7478,88-1080,88-2520
Telex: 6-7954
Cable: HEWPACK

TAIWAN
Hewlett-Packard Far East Ltd.,
Taiwan Branch
39 Chung Shiao West Road
Sec. 1, 7th Floor
Taipei
Tel: 389160,1,2,3
Telex: 21824 HEWPACK
Cable: HEWPACK TAIPEI

Hewlett-Packard Far East Ltd.
Taiwan Branch
68-2, Chung Cheng 3rd. Road
Kaohsiung
Tel: (07) 242318-Kaohsiung

Analytical Only
San Kwang Instruments Co., Ltd.,
No. 20, yung Sui Road
Taipei, 100
Tel: 3715171-4 (4 lines)
Telex: 22894 SANKWANG
Cable: SANKWANG TAIPEI

TANZANIA
Medical Only
International Aeradio (E.A.), Ltd.
P.O. Box 861
Daressalaam
Tel: 21251 Ext. 265
Telex: 41030

THAILAND
UNIMESA Co., Ltd.
Elcom Research Building
Bangjak Sukumvit Ave.
Bangkok
Tel: 932387, 930338
Cable: UNIMESA Bangkok

UGANDA
Medical Only
International Aeradio(E.A.), Ltd.,
P.O. Box 2577
Kampala
Tel: 54388
Cable: INTAERIO Kampala

ZAMBIA
R.J. Tilbury (Zambia) Ltd.
P.O. Box 2792
Lusaka
Tel: 73793
Cable: ARJAYTEE, Lusaka

OTHER AREAS NOT LISTED, CONTACT:
Hewlett-Packard Intercontinental
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 493-1501
TWX: 910-373-1267
Cable: HEWPACK Palo Alto
Telex: 034-8300, 034-8493

CANADA

ALBERTA
Hewlett-Packard (Canada) Ltd.
11748 Kingsway Ave.
Edmonton T5G 0X5
Tel: (403) 452-3670
TWX: 610-831-2431 EDTH

Hewlett-Packard (Canada) Ltd.
915-42 Avenue S.E. Suite 102
Calgary T2G 1Z1
Tel: (403) 287-1672
TWX: 610-821-6141

BRITISH COLUMBIA
Hewlett-Packard (Canada) Ltd.
837 E. Cordova Street
Vancouver V6A 3R2
Tel: (604) 254-0531
TWX: 610-922-5059 VCR

MANITOBA
Hewlett-Packard (Canada) Ltd.
513 Century St.
St. James
Winnipeg R3H 0L8
Tel: (204) 786-7581
TWX: 610-671-3531

NOVA SCOTIA
Hewlett-Packard (Canada) Ltd.
800 Windmill Road
P.O. Box 9331
Dartmouth B2Y 3Z6
Tel: (902) 469-7820
TWX: 610-271-4482 HFX

ONTARIO
Hewlett-Packard (Canada) Ltd.
1785 Woodward Dr.
Ottawa K2C 0P9
Tel: (613) 225-6530
TWX: 610-562-8968

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
Mississauga L4V 1M8
Tel: (416) 678-9430
TWX: 610-492-4246

QUEBEC
Hewlett-Packard (Canada) Ltd.
275 Hymus Blvd.
Pointe Claire H9R 1G7
Tel: (514) 697-4232
TWX: 610-422-3022
TLX: 05-821521 HPCL

Hewlett-Packard (Canada) Ltd.
2376 Galvani Street
Ste-Foy G1N 4G4
Tel: (418) 688-8710
TWX: 610-571-5525

FOR CANADIAN AREAS NOT LISTED:
Contact Hewlett-Packard (Canada) Ltd. in Mississauga.

CENTRAL AND SOUTH AMERICA

ARGENTINA
Hewlett-Packard Argentina S.A.
Av. Leandro N. Alem 822 - 12°
1001**Buenos Aires**
Tel: 32-4461/62/63/64
Telex: Public Booth

BOLIVIA
Stambuk & Mark (Bolivia) Ltda.
Av. Mariscal, Santa Cruz 1342
La Paz
Tel: 40626, 53163, 52421
Telex: 3560014
Cable: BUKMAR

BRAZIL
Hewlett-Packard do Brasil I.E.C. Ltda.
Rua Frei Caneca, 1140/52 Bela Vista
01307-**São Paulo**-SP
Tel: 288-71-11, 287-81-20, 287-61-93
Telex: 391-1132-3602 HPBR-BR
Cable: HEWPACK São Paulo

Hewlett-Packard do Brasil I.E.C. Ltda.
Rua Padre Chagas, 32
90000-**Pórto Alegre**-RS
Tel: (0512) 22-2998, 22-5621
Cable: HEWPACk pohto Alegre

Hewlett-Packard do Brasil I.E.C. Ltda.
Rua Siqueira Campos, 53, 4° andar-Copacabana
20000-**Rio de Janeiro**-GB
Tel: 257-80-94-DDD (021)
Telex: 391-212-1905 HEWP-BR
Cable: HEWPACK Rio de Janeiro

CHILE
Calcagni y Metcalfe Ltda.
Alameda 580-Of. 807
Casilla 2118
Santiago, 1
Tel: 398613
Telex: 3520001 CALMET
Cable: CALMET Santiago

Medical Only
General Machinery Co., Ltda.
Paraguay 494
Casilla 13910
Santiago
Tel: 31123, 31124
Cable: GEMCO Santiago

COLOMBIA
Instrumentación
Henrik A. Langebaek & Kier S.A.
Carrera 7 No. 48-75
Apartado Aéreo 6287
Bogotá, I.D.E.
Tel: 69-88-77
Cable: AARIS Bogotá
Telex: 044-400

COSTA RICA
Cientifica Costarricense S.A.
Calle Central, Avenidas 1 y 3
Apartado 10159
San José
Tel: 21-86-13
Cable: GALGUR San José

ECUADOR
Medical Only
A.F. Viscaino Compañia Ltda.
Av. Rio Amazonas No. 239
P.O. Box 2925
Quito
Tel: 242-150,247-033/034
Cable: Astor Quito

Calculators Only
Computadoras y Equipos Electrónicos
P.O. Box 2695
990 Toledo (y Cordero)
Quito
Tel: 525-982
Telex: 02-2113 Sagita Ed
Cable: Sagita-Quito

EL SALVADOR
Instrumentacion y Procesamiento Electronico de el Salvador
Bulevar de los Heroes II-48
San Salvador
Tel: 252787

GUATEMALA
IPESA
Avenida La Reforma 3-48, Zona 9
Guatemala City
Tel: 63627, 64786
Telex: 4192 Teletro Gu

MEXICO
Hewlett-Packard Mexicana, S.A. de C.V.
Torres Adalid No. 21, 11° Piso
Col. del Valle
Mexico 12, D.F.
Tel: (905) 543-42-32
Telex: 017-74-507

Hewlett-Packard Mexicana, S.A. de C.V.
Ave. Constitución No. 2184
Monterrey, N.L.
Tel: 48-71-32, 48-71-84
Telex: 038-843

NICARAGUA
Roberto Terán G.
Apartado Postal 689
Edificio Terán
Managua
Tel: 25114, 23412,23454
Cable: ROTERAN Managua

PANAMA
Electrónico Balboa, S.A.
P.O. Box 4929
Calle Samuel Lewis
Cuidad de Panama
Tel: 64-2700
Telex: 3431103 Curunda, Canal Zone
Cable: ELECTRON Panama

PARAGUAY
Z.J. Melamed S.R.L.
División: Aparatos y Equipos Médicos
División: Aparatos y Equipos Cientificos y de Investigación
P.O. Box 676
Chile-482, Edificio Victoria
Asunción
Tel: 4-5069, 4-6272
Cable: RAMEL

PERU
Compañía Electro Médica S.A.
Los Flamencos 145
San Isidro Casilla 1030
Lima 1
Tel: 41-3703
Cable: ELMED Lima

PUERTO RICO
Hewlett-Packard Inter-Americas
Puerto Rico Branch Office
P.O. Box 2908I
65th Inf. Station
San Juan 00929
Calle 272, Urb. Country Club
Carolina 00639
Tel: (809) 762-7355/7455/7655
Telex: HPIC-PR 3450514

URUGUAY
Pablo Ferrando S.A.
Comercial e Industrial
Avenida Italia 2877
Casilla de Correo 370
Montevideo
Tel: 40-3102
Cable: RADIUM Montevideo

VENEZUELA
Hewlett-Packard de Venezuela C.A.
Apartado 50933, Caracas 105
Edificio Segre
Tercera Transversal
Los Ruices Norte
Caracas 107
Tel: 35-01-07, 35-00-84, 35-00-65, 35-00-31
Telex: 25146 HEWPACK
Cable: HEWPACK Caracas

FOR AREAS NOT LISTED, CONTACT:
Hewlett-Packard
Inter-Americas
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 493-1501
TWX: 910-373-1260
Cable: HEWPACK Palo Alto
Telex: 034-8300, 034-8493

EUROPE, NORTH AFRICA AND MIDDLE EAST

AUSTRIA
Hewlett-Packard Ges.m.b.H.
Handelskai 52
P.O. box 7
A-1205 **Vienna**
Tel: (0222) 35 16 21 to 27
cable: HEWPAK Vienna
Telex: 75923 hewpak a

BELGIUM
Hewlett-Packard Benelux
S.A./N.V.
Avenue de Col-Vert, 1,
(Groenkraaglaan)
B-1170 **Brussels**
Tel: (02) 672 22 40
Cable: PALOBEN Brussels
Telex: 23 494 paloben bru

CYPRUS
Kypronics
19, Gregorios & Xenopoulos Rd.
P.O. Box 1152
CY-**Nicosia**
Tel: 45628/29
Cable: KYPRONICS PANDEHIS
Telex: 3018

CZECHOSLOVAKIA
Vyvojova a Provozni Zakladna
Vyzkumnych Ustava v Bechovicich
CSSR-25097
Bechovice u Prahy
Tel: 89 93 41
Telex: 121333

DDR
Entwicklungslabor der TU Dresden
Forschungsinstitut Meinsberg
DDR-7305
Waldheim/Meinsberg
Tel: 37 667
Telex: 518741

DENMARK
Hewlett-packard A/S
Datavej 52
DK-3460 **Birkerød**
Tel: (02) 81 66 40
Cable: HEWPACK AS
Telex: 166 40 hpas

Hewlett-Packard A/S
Navervej 1
DK-8600 **Silkeborg**
Tel: (06) 82 71 66
Telex: 166 40 hpas
Cable: HEWPACK AS

FINLAND
Hewlett-Packard OY
Nahkahousuntie 5
P.O. Box 6
SF-00211 **Helsinki** 21
Tel: 6923031
Cable: HEWPACKOY Helsinki
Telex: 12-1563

FRANCE
Hewlett-Packard France
Quartier de Courtaboeuf
Boite Postale No. 6
F-91401 **Orsay** Cédex
Tel: (1) 907 78 25
Cable: HEWPACK Orsay
Telex: 600048

Hewlett-Packard France
"Le Saquin"
Chemin des Mouilles
Boite Postale No. 12
F-69130 **Ecully**
Tel: (78) 33 81 25,
Cable: HEWPACK Eculy
Telex: 310617

Hewlett-Packard France
Agence Régionale
Péricentre de la Cépière
Chemin de la Cépière, 20
F-31300 **Toulouse-Le Mirail**
Tel:(61) 40 11 12
Cable: HEWPACK 51957
Telex: 510957

Hewlett-Packard France
Agence Régionale
Aéroport principal de
Marseille-Marignane
F-13721**Marignane**
Tel: (91) 89 12 36
Cable: HEWPACK MARGN
Telex: 410770

Hewlett-Packard France
Agence Régionale
63, Avenue de Rochester
Boîte Postale
F-35014 **Rennes** Cédex
Tel: (99) 36 33 21
Cable: HEWPACK 74912
Telex: 740912

Hewlett-Packard France
Agence Régionale
74, Allée de la Robertsau
F-67000 **Strasbourg**
Tel: (88) 35 23 20/21
Telex: 890141
Cable: HEWPACK STRBG

Hewlett-Packard France
Agence Régionale
Centre Vauban
201, rue Colbert
Entrée A2
F-59000 **Lille**
Tel: (20) 51 44 14
Telex: 820744

GERMAN FEDERAL REPUBLIC
Hewlett-Packard GmbH
Vertriebszentrale Frankfurt
Bernerstrasse 117
Postfach 560 140
D-6000 **Frankfurt** 56
Tel: (0611) 50 04-1
Cable: HEWPACKSA Frankfurt
Telex: 04 13249 hpffmd

Hewlett-Packard GmbH
Technisches Buero Böblingen
Herrenbergerstrasse 110
D-7030 **Böblingen**, Württemberg
Tel: (07031) 667-1
Cable: HEPAK Böblingen
Telex: 07265739 bbn

Hewlett-Packard GmbH
Technisches Buero Düsseldorf
Emanuel-Leutze-Str.1 (Seestern)
D-4000 **Dusseldorf**
Tel: (0211) 59 71-1
Telex: 085/86 533 hpdd d

Hewlett-Packard GmbH
Technisches Buero Hamburg
Wendenstrasse 23
D-2000 **Hamburg** 1
Tel: (040) 24 13 93
Cable: HEWPACKSA Hamburg
Telex: 21 63 032 hphh d

Hewlett-Packard GmbH
Technisches Buero Hannover
Mettendorter Strasse 3
D-3000 **Hannover-Kleefeld**
Tel: (0511) 55 60 46
Telex: 092 3259

Hewlett-Packard GmbH
Technisches Buero Nuremberg
Neumeyer Str. 90
D-8500 **Nuremberg**
Tel: (0911) 56 30 83/85
Telex: 0623 860

Hewlett-Packard GmbH
Technisches Buero München
Unterhachinger Strasse 28
ISAR Center
D-8012 **Ottobrunn**
Tel: (089) 601 30 61/7
Cable: HEWPACKSA München
Telex: 0524985

Hewlett-Packard GmbH
Technisches Buero Berlin
Keith Strasse 2-4
D-1000 **Berlin** 30
Tel: (030) 24 90 86
Telex: 18 3405 hpbln d

GREECE
Kostas Karayannis
18, Ermou Street
GR-**Athens** 126
Tel: 3237731
Cable: RAKAR Athens
Telex: 21 59 62 rkar gr

Analytical Only
"INTECO"
G. Papathanassiou & Co.
Marni 17
GR - **Athens** 103
Tel: 522 1915
Cable: INTEKNIKA Athens
Telex: 21 5329 INTE GR

Medical Only
Technomed Hellas Ltd.
52,Skoufa Street
GR - **Athens** 135
Tel: 626 972,663 930,614 959
Cable: ETALAK Athens
Telex: 21-4693 ETAL GR

HUNGARY
MTA
Müszerügyi és Méréstechnikai
Szolgalata
Lenin Krt. 67
1391 **Budapest** VI
Tel: 42 03 38
Telex: 22 51 14

ICELAND
Medical Only
Elding Trading Company Inc.
Hafnarhvoli - Tryggvatotu .
IS-**Reykjavik**
Tel: 1 58 20
Cable: ELDING Reykjavik

IRAN
Hewlett-Packard Iran Ltd.
Mir-Emad Avenue
14th Street No.13
P.O. Box 41/2419
IR-**Tehran**
Tel: 85 10 82/86
Telex: 21 25 74 Khrm ir

IRELAND
Hewlett-Packard Ltd.
King Street Lane
GB-**Winnersh,**Wokingham
Berks, RG11 5AR
Tel: (0734) 78 47 74
Telex: 847178/848179

ITALY
Hewlett-Packard Italiana S.p.A.
Casella postale 3645
I-20100 **Milano**
Tel: (2) 6251 (10 lines)
Cable: HEWPACKIT Milano
Telex: 32046

Hewlett-Packard Italiana S.p.A.
Via Pietro Maroncelli 40
(ang. Via Visentin)
I-35100 **Padova**
Tel: (49) 66 48 88
Telex: 41612 Hewpacki

Medical only
Hewlett-Packard Italiana S.p.A.
Via d'Aghiardi, 7
I-56100 **Pisa**
Tel: (050) 2 32 04
Telex: 32046 via Milano

Hewlett-Packard Italiana S.p.A.
Via G. Armellini 10
I-00143 **Roma**
Tel: (06) 54 69 6l
Telex: 61514
Cable: HEWPACKIT Roma

Hewlett-Packard Italiana S.p.A.
Via San Quintino, 46
I-10121 **Torino**
Tel: (011) 52 82 64/54 84 68
Telex: 32046 via Milano

Medical/Calculators Only
Hewlett-Packard Italiana S.p.A.
Via Principe Nicola 43 G/C
I-95126 **Catania**
Tel:(095) 37 05 04

Hewlett-Packard Italiana S.p.A.
Via Amerigo Vespucci, 9
I-80142 **Napoli**
Tel: (081) 33 77 11

Hewlett-Packard Italiana S.p.A.
Via E. Masi, 9/B
I-40137 **Bologna**
Tel: (051) 30 78 87

KUWAIT
Al-Khaldiya Trading &
Contracting Co.
P.O. Box 830
Kuwait
Tel: 42 49 10
Cable: VISCOUNT

LUXEMBURG
Hewlett-Packard Benelux
S.A./N.V.
Avenue du Col-Vert, 1,
(Groenkraaglaan)
B-1170 **Brussels**
Tel: (02) 672 22 40
Cable: PALOBEN Brussels
Telex: 23 494

MOROCCO
Gerep
190, Blvd. Brahim Roudani
Casablanca
Tel: 25-16-76/25-90-99
Cable: Gerep-Casa
Telex: 23739

NETHERLANDS
Hewlett-Packard Benelux N.V.
Van Heuven Goedhartlaan 121
P.O. Box 667
NL- **Amstelveen** 1134
Tel: (020) 47 20 21
Cable: PALOBEN Amsterdam
Telex: 13 216 hepa nl

NORWAY
Hewlett-Packard Norge A/S
Nesveien 13
Box 149
N-1344 **Haslum**
Tel: (02) 53 83 60
Telex: 16621 hpnas n

POLAND
Biuro Informacji Technicznej
Hewlett-Packard
U1 Stawki 2 6P
00-950**Warsaw**
Tel: 39 67 43
Telex: 81 24 53 hepa pl

UNIPAN
Zaklad Doswiadczalny
Budowy Aparatury Naukowej
U1. Krajowej Rady
Narodowej 51/55
00-800 **Warsaw**
Tel: 20 62 21
Telex: 81 46 48

Zaklady Naprawcze Sprzetu
Medycznego
Plac Komuny Paryskiej 6
90-007 **Lodz**
Tel: 334-41, 337-83

PORTUGAL
Telectra-Empresa Técnica de
Equipamentos Eléctricos S.a.r.l.
Rua Rodrigo da Fonseca 103
P.O. Box 2531
P-**Lisbon** 1
Tel: (19) 68 60 72
Cable: TELECTRA Lisbon
Telex: 12598

Medical only
Mundinter
Intercambio Mundial de Comércio
S.a.r.l.
Av.A.A.de Aguiar 138
P.O. Box 2761
P - **Lisbon**
Tel: (19) 53 21 31/7
Cable: INTERCAMBIO Lisbon

RUMANIA
Hewlett-Packard Reprezentanta
BD.N. Balcescu 16
Bucharest
Tel: 158023/138885
Telex: 10440

I.I.R.U.C.
Intreprinderea Pentru
Intretinerea
Si Repararea Utilajelor de Calcul
B-dul prof. Dimitrie Pompei 6
Bucharest-Sectorul 2
Tel: 12 64 30
Telex: 01183716

SAUDI ARABIA
Modern Electronic Establishment
King Abdul Aziz str.(Head office)
P.O. Box 1228
Jeddah
Tel: 31173-332201
Cable: ELECTRA

P.O. Box 2728 (Service center)
Riyadh
Tel: 62596-66232
Cable: RAOUFCO

SPAIN
Hewlett-Packard Española, S.A.
Jerez No. 3
E-**Madrid** 16
Tel:(1) 458 26 00 (10 lines)
Telex: 23515 hpe

Hewlett-Packard Española, S.A.
Milanesado 21-23
E-**Barcelona** 17
Tel: (3) 203 6200 (5 lines)
Telex: 52603 hpbe e

Hewlett-Packard Española, S.A.
Av Ramón y Cajal, 1-9°
(Edificio Sevilla I)
E-**Seville** 5
Tel: 64 44 54/58

Hewlett-Packard Española S.A.
Edificio Albia II 7° B
E-**Bilbao**-1
Tel: 23 83 06/23 82 06

Calculators Only
Hewlett-Packard Española S.A.
Gran Via Fernando El Católico, 67
E-**Valencia**-8
Tel: 326 67 28/326 85 55

SWEDEN
Hewlett-Packard Sverige AB
Enighetsvägen 3
Fack
S-161 20 **Bromma** 20
Tel: (08) 730 05 50
Cable: MEASUREMENTS
Stockholm
Telex: 10721

Hewlett-Packard Sverige AB
Frötallsgatan 30
S-421 32 **Västra Frölunda**
Tel: (031) 49 09 50
Telex: 10721 Via Bromma Office

SWITZERLAND
Hewlett-Packard (Schweiz) AG
Zürcherstrasse 20
P.O. Box 307
CH-8952 **Schlieren-Zurich**
Tel: (01) 730 52 40
Cable: HPAG CH
Telex: 53933 hpag ch

Hewlett-Packard (Schweiz) AG
9, Chemin Louis-Pictet
CH-1214 **Vernier-Geneva**
Tel: (022) 41 49 50
Cable: HEWPACKAG Geneva
Telex: 27 333 hpag ch

SYRIA
Medical/calculator only
Sawah & Co.
Place Azmé
B.P. 2308
SYR-**Damascus**
Tel: 16367, 19697, 14268
Cable: SAWAH, Damascus

TURKEY
Telekom Engineering Bureau
P.O. Box 437
Beyoglu
TR-**Istanbul**
Tel: 49 40 40
Cable: TELEMATION Istanbul
Telex: 23609

Medical only
E.M.A.
Muhendislik Kollektif Sirketi
Adakale Sokak 41/6
TR-**Ankara**
Tel: 175622

Analytical only
Yilmaz Ozyurek
Milli Mudafaa Cad No. 16/6
Kizilay
TR-**Ankara**
Tel: 25 03 09
Telex: 42576 Ozek tr

UNITED KINGDOM
Hewlett-Packard Ltd.
King Street Lane
GB-**Winnersh,** Wokingham
Berks, RG11 5AR
Tel: (0734) 78 47 74
Cable: Hewpie London
Telex:847178/9

Hewlett-Packard Ltd.
"The Graftons"
Stamford New Road
GB-**Altrincham**
Cheshire WA14 1DQ
Tel: (061) 9289021
Cable: Hewpie Manchester
Telex: 668068

Hewlett-Packard Ltd.
Lygon Court
Dudley Road
GB-**Halesowen**, Worcs
Tel: (021) 550 7053
Telex: (021) 550 7273

Hewlett-Packard Ltd.
Wedge House
799, London Road
GB-**Thornton Heath**
Surrey CR4 6XL
Tel: (01) 6840105
Telex: 946825

Hewlett-Packard Ltd.
c/o Makro
South Service Wholesale Centre
Wear Industrial Estate
Washington
GB-**New Town**, County Durham
Tel: Washington 464001 ext. 57/58

USSR
Hewlett-Packard Representative
Office USSR
Pokrovsky Boulevard 4/17, KV 12
Moscow101000
Tel: 294-2024
Telex: 7825 hewpak su

YUGOSLAVIA
Iskra-standard/Hewlett-Packard
Miklosiceva 38/VII
61000 **Ljubljana**
Tel: 31 58 79/32 16 74
Telex: 31300

MEDITERRANEAN AND MIDDLE EAST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard S.A.
Mediterranean and Middle
East Operations
35, Kolokotroni Street
Platia Kefallariou
GR-Kifissia-**Athens**, Greece
Tel: 8080337/359/429

SOCIALIST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard Ges.m.b.H
P.O. Box 7
A-1205 **Vienna**, Austria
Tel: (0222) 35 16 21 to 27

FOR OTHER AREAS NOT LISTED CONTACT
Hewlett-Packard S.A.
7, rue du Bois-du-Lan
P.O. Box
CH-1217 Meyrin 2 - **Geneva**
Switzerland
Tel: (022) 41 54 00

UNITED STATES

ALABAMA
8290 Whitesburg Dr., S.E.
P.O. Box 4207
Huntsville 35802
Tel: (205) 881-4591

Medical Only
228 W. Valley Ave.,
Room 220
Birmingham 35209
Tel: (205) 942-2081

ARIZONA
2336 E. Magnolia St.
Phoenix 85034
Tel: (602) 244-1361

2424 East Aragon Rd.
Tucson 85706
Tel: (602) 294-3148

***ARKANSAS**
Medical Service Only
P.O. Box 5646
Brady Station
Little Rock 72205
Tel:)501) 664-8773

CALIFORNIA
1430 East Orangethorpe Ave.
Fullerton 92631
Tel: (714) 870-1000

3939 Lankershim Boulevard
North Hollywood 91604
Tel: (213) 877-1282
TWX: 910-499-2170

6305 Arizona Place
Los Angeles 90045
Tel: (213) 649-2511
TWX: 910-328-6147

***Los Angeles**
Tel: (213) 776-7500

3003 Scott Boulevard
Santa Clara 95050
Tel: (408) 249-7000
TWX: 910-338-0518

***Ridgecrest**
Tel: (714) 446-6165

2220 Watt Ave.
Sacramento 95825
Tel: (916) 482-1463

9606 Aero Drive
P.O. Box 23333
San Diego 92123
Tel: (714) 279-3200

COLORADO
5600 South Ulster Parkway
Englewood 80110
Tel: (303) 771-3455

CONNECTICUT
12 Lunar Drive
New Haven 06525
Tel: (203) 389-6551
TWX: 710-465-2029

FLORIDA
P.O. Box 24210
2806 W. Oakland Park Blvd.
Ft. Lauderdale 33307
Tel: (305) 731-2020

***Jacksonville**
Medical Service only
Tel: (904) 725-6333

P.O. Box 13910
6177 Lake Ellenor Dr.
Orlando 32809
Tel: (305) 859-2900

P.O. Box 12826
Pensacola 32575
Tel: (904) 434-3081

GEORGIA
P.O. Box 105005
Atlanta 30348
Tel: (404) 955-1500
TWX:810-766-4890

Medical Service Only
***Augusta** 30903
Tel: (404) 736-0592

HAWAII
2875 So. King Street
Honolulu 96814
Tel: (808) 955-4455

ILLINOIS
5500 Howard Street
Skokie 60076
Tel: (312) 677-0400
TWX: 910-223-3613

Effective Nov.1, 1976
35201 Tollview Dr.
Rolling Meadows 60008
Tel: (312) 255-9800
TWX: 910-687-2260

INDIANA
7301 North Shadeland Ave.
Indianapolis46250
Tel: (317)842-1000
TWX: 810-260-1796

IOWA
1902 Broadway
Iowa City 52240
Tel: (319) 338-9466
Night: (319) 338-9467

KENTUCKY
Medical Only
Atkinson Square
3901 Atkinson Dr.,
Suite 207
Louisville 40218
Tel: (502) 456-1573

LOUISIANA
P.O. Box 840
3239 Williams Boulevard
Kenner 70062
Tel: (504) 721-6201

MARYLAND
6707 Whitestone Road
Baltimore 21207
Tel: (301) 944-5400
TWX: 710-862-9157

2 Choke Cherry Road
Rockville 20850
Tel: (301) 948-6370
TWX: 710-828-9684

MASSACHUSETTS
32 Hartwell Ave.
Lexington 02173
Tel: (617) 861-8960
TWX: 710-326-6904

MICHIGAN
23855 Research Drive
Farmington Hills 48024
Tel: (313) 476-6400
TWX: 810-242-2900

MINNESOTA
2400 N. Prior Ave.
Roseville 55113
Tel: (612) 636-0700
TWX: 910-563-3734

MISSISSIPPI
***Jackson**
Medical Service only
Tel: (601) 982-9363

MISSOURI
11131 Colorado Ave.
Kansas City 64137
Tel: (816) 763-8000
TWX: 910-771-2087

148 Weldon Parkway
Maryland Heights 63043.
Tel: (314) 567-1455
TWX: 910-764-0830

NEBRASKA
Medical Only
7171 Mercy Road
Suite 110
Omaha 68106
Tel: (402) 392-0948

NEW JERSEY
W. 120 Century Rd
Paramus 07652
Tel: (201) 265-5000
TWX: 710-990-4951

NEW MEXICO
P.O. Box 11634
Station E
11300 Lomas Blvd., N.E.
Albuquerque 87123
Tel: (505) 292-1330
TWX: 910-989-1185

156 Wyatt Drive
Las Cruces 88001
Tel: (505) 526-2485
TWX: 910-983-0550

NEW YORK
6 Automation Lane
Computer Park
Albany 12205
Tel: (518) 458-1550
TWX: 710-441-8270

201 South Avenue
Poughkeepsie 12601
Tel: (914) 454-7330
TWX: 510-248-0012

39 Saginaw Drive
Rochester 14623
Tel: (716) 473-9500
TWX: 510-253-5981

5858 East Molloy Road
Syracuse 13211
Tel: (315) 454-2486
TWX: 710-541-0482

1 Crossways Park West
Woodbury 11797
Tel: (516) 921-0300
TWX: 710-990-4951

NORTH CAROLINA
P.O. Box 5188
1923 North Main Street
High Point 27262
Tel: (919) 885-8101

OHIO
16500 Sprague Road
Cleveland 44130
Tel: (216) 243-7300
TWX: 810-423-9431

330 Progress Rd
Dayton 45449
Tel: (513) 859-8202
TWX: 810-474-2818

1041 Kingsmill Parkway
Columbus 43229
Tel: (614) 436-1041

OKLAHOMA
P.O. Box 32008
Oklahoma City 73132
Tel: (405) 721-0200

OREGON
17890 SW Lower Boones
Ferry Road
Tualatin 97062
Tel: (503) 620-3350

PENNSYLVANIA
111 Zeta Drive
Pittsburgh 15238
Tel: (412) 782-0400
TWX: 710-795-3124

1021 8th Avenue
King of Prussia Industrial Park
King of Prussia 19406
Tel: (215) 265-7000
TWX: 510-660-2670

SOUTH CAROLINA
6941-0 N. Trenholm Road
Columbia 29260
Tel: (803) 782-6493

TENNESSEE
***Knoxville**
Medical Services only
Tel: (615) 523-5022

***Nashville**
Medical Service only
Tel: (615) 244-5448

TEXAS
P.O. Box 1270
201 E. Arapaho Rd.
Richardson 75080
Tel: (214) 231-6101

P.O. Box 27409
6300 Westpark Drive
Suite 100
Houston 77027
Tel: (713) 781-6000

205 Billy Mitchell Road
San Antonio 78226
Tel: (512) 434-8241

UTAH
2160 South 3270 West Street
Salt Lake City 84119
Tel: (801) 487-0715

VIRGINIA
Medical Only
P.O. Box 12778
No. 7 Koger Exec. Center
Suite 212
Norfolk 23502
Tel:(804) 497-1026/7

P.O.Box 9854
2014 Hungary Springs Road
Richmond 23228
Tel: (804) 285-3431

WASHINGTON
Bellefield Office Pk.
1203-114th Ave. S.E.
Bellevue 98004
Tel: (206) 454-3971
TWX: 910-443-2446

***WEST VIRGINIA**
Medical/Analytical Only
Charleston
Tel: (304) 345-1640

WISCONSIN
9004 West Lincoln Ave.
West Allis 53227
Tel: (414) 541-0550

FOR U.S. AREAS NOT LISTED:
Contact the regional office
nearest you: Atlanta, Georgia...
North Hollywood, California...
Rockville, Maryland...Rolling Meadows,
Illinois.Their complete
addresses are listed above.

***Service Only**

PART NO. 22687-90009
Printed in U.S.A. 5/76

HEWLETT hp PACKARD

Sales and service from 172 offices in 65 countries.
5303 Stevens Creek Blvd., Santa Clara, California 95050