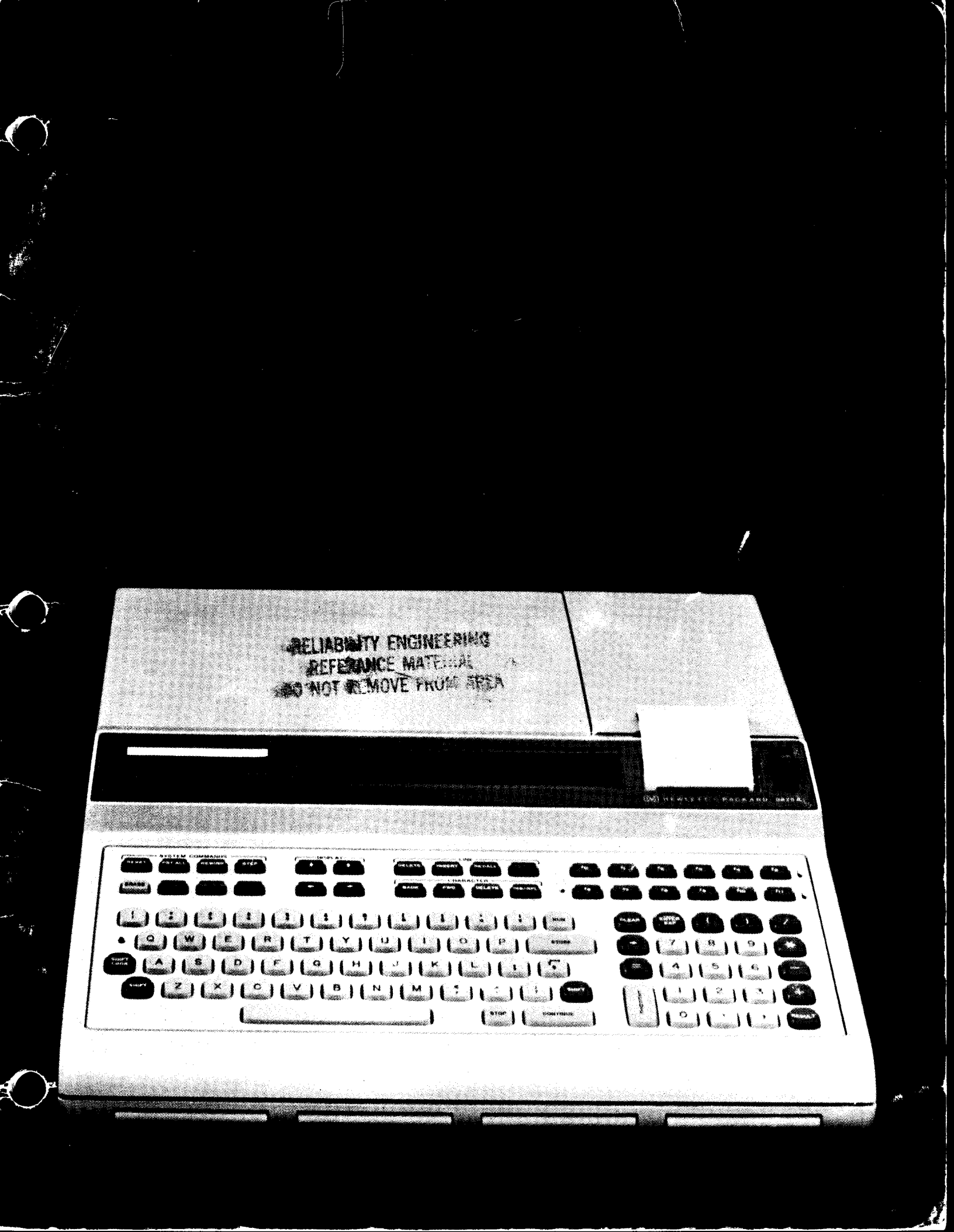

RELIABILITY ENGINEERING
REFERANCE MATERIAL
DO NOT REMOVE FROM AREA

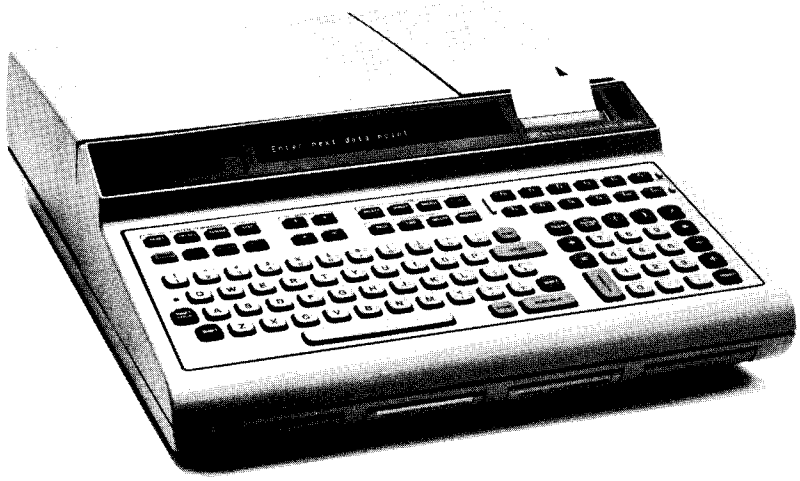
HEWLETT PACKARD

Warranty Statement

Hewlett-Packard products are warranted against defects in materials and workmanship. For Hewlett-Packard Calculator Products Division products sold in the U.S.A. and Canada, this warranty applies for ninety (90) days from date of delivery. Hewlett-Packard will repair or replace equipment which proves to be defective during the warranty period. This warranty includes labor, parts and surface travel costs, if any. Equipment returned to Hewlett-Packard for repair must be shipped freight prepaid. Repairs necessitated by misuse of the equipment, or by hardware, software, or interfacing not provided by Hewlett-Packard are not covered by this warranty.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. HEWLETT-PACKARD SHALL NOT BE LIABLE FOR CONSEQUENTIAL DAMAGES.

Advanced Programming



HP 9825A Calculator

Hewlett-Packard Calculator Products Division
P.O. Box 301, Loveland, Colorado 80537, Tel. (303) 667-5000
(For World-wide Sales and Service Offices see back of manual.)

Table of Contents

Chapter 1
General Information

Description

The Advanced Programming ROM (Read Only Memory), when in the HP 9825A Calculator, enables you to—

- Use for/next loops to repeat sections of a program.
- Pass parameters to subprograms including subroutines and functions.
- Store numbers in split and integer precision formats to conserve memory.
- Generate a list of the variables used in a program and the line numbers in which they occur using the cross reference statement.

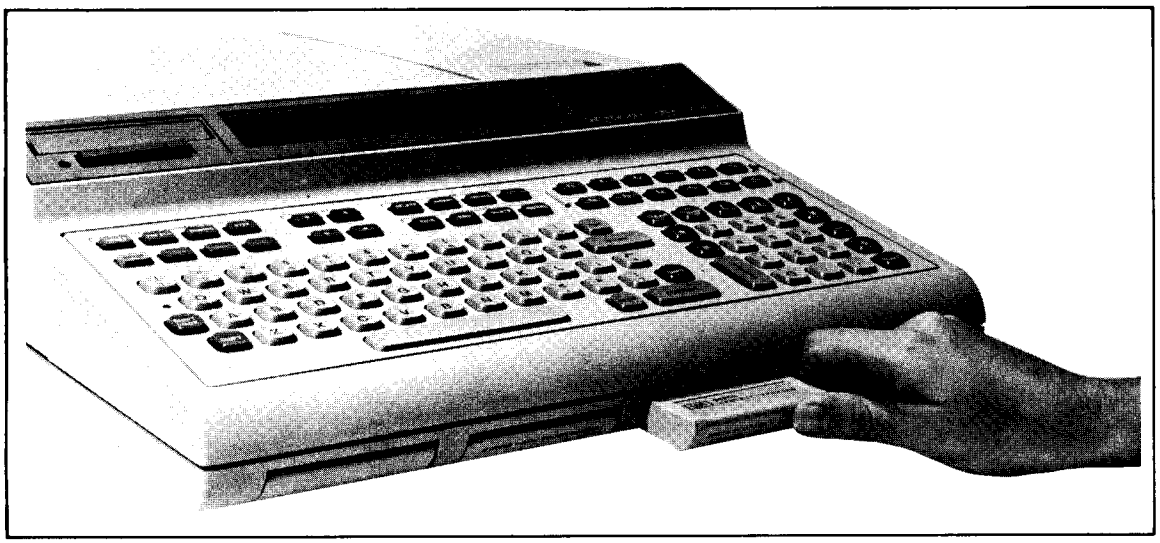
The Advanced Programming (AP) ROM uses four bytes of user RWM (Read Write Memory) when installed in the HP 9825A.

The AP ROM is packaged with another ROM in a single ROM card. The Advanced Programming Manual (P/N 09825-90021) is supplied with the AP ROM. This manual describes AP operations only.

Inspection and Installation

Refer to the HP 9825A System Test Booklet for the procedure to verify the operation of your ROM.

Your AP ROM can be plugged into any one of the four ROM slots located on the bottom front of the calculator, as shown below.



Installing the ROM

To install your ROM card, first turn off the calculator. With the label right side up, slide the ROM through the ROM slot door. Press it in until the front of the ROM card is even with the front of the calculator. Then turn your calculator on.

Syntax

The following conventions apply to the syntax for the statements and functions found in this manual.

dot matrix - All items in dot matrix are required, exactly as shown.

[ ] - All items in square brackets are optional, unless the brackets are in dot matrix.

See the Appendix for a list of the syntax of all AP statements and functions.

Error Messages

The AP ROM adds error messages A0 through A9 and special meanings to five mainframe errors of the calculator error message list. Explanations of these errors can be found on the inside back cover of this manual.

Requirements

Before using this manual you should be familiar with the calculator and the HPL programming language described in the HP 9825A Operating and Programming Manual.

Chapter 2
For/Next Loops

Description

The for and next statements enable you to repeat a group of statements within a program as many times as necessary.

for simple variable = initial value to final value [by step size value]
•
•
•
•
next same simple variable

The for and next statements, including the statements between them, form a loop within a program. The for statement defines the beginning of the loop and the number of times the loop is to be performed. The variable that follows the for and next statements can be any one of the simple variables A through Z.

The initial, final and step size values can be expressions. If the step size value is not specified, the default value is 1.

Here's an example of a for/next loop—

```
•
•
•
5: for I=1 to 5
•
•
•
•
10: next I
•
•
•
```

This for/next loop would be executed five times - when I = 1,2,3,4 and 5. Each time the next statement is executed, the value of I is incremented by one, the default step size value. When the value of I exceeds the final value (when I = 6)*, the loop is finished and the program continues at the statement following the next statement.

The advantages of using for/next looping instead of an if statement are shown in the following examples where the numbers 1 through 10000 are displayed in succession.

if statement

```
0: 1→I
1: dsp I
2: if (I+1→I)<=1
 0000:gto 1
3: beep
4: end
```

for/next loop

```
0: for I=1 to
 10000
1: dsp I
2: next I
3: beep
4: end
```

The program that uses the for/next loop is easier to key in, takes less calculator memory (40 bytes) and is executed faster (25 seconds). With the if statement, the program uses 48 bytes of memory and is executed in 32 seconds.

The initial value of the variable assigned in the for/next loop does not have to be 1. The following example totals the integers, 90 through 100, and prints the total (1045).

```
0: 0→A
1: for I=90 to
 100
2: I+A→A
3: next I
4: prt "Total=";
 A
5: end
```

```
Total=    1045.00
```

*This is an often overlooked aspect of for/next loops and is covered on page 7.

The next example illustrates that variables can be used in the for statement. The variables B and C are assigned values in the enter statement in line 1 and are used in the for statement in line 3.

```
0: 0→A
1: ent B,C
2: prt "B=",B,
 "C=",C
3: for I=B to C
4: I+A→A
5: next I
6: prt "Total=",
 A
7: end
```

```
B=          1.00
C=          3.00
Total=      6.00

B=          5.50
C=          8.50
Total=     28.00
```

If B = 1 and C = 3, the total of 1,2 and 3 (6) is printed. If B = 5.5 and C = 8.5, the total of 5.5, 6.5, 7.5 and 8.5 (28) is printed. In either case, the value of I is incremented by one after each loop. If the value of B is greater than the value of C, the loop is not executed and the program continues at the first statement following next I, in this example the print statement in line 6.

The following example illustrates an often overlooked aspect of for/next looping. After each loop is performed, the next statement increments the value of I by 1. Then the incremented value is compared with the final value. If the incremented value is not greater than the final value, the loop is repeated. When the incremented value is greater than the final value (when I = 11) the loop is no longer repeated and the statement following the next statement (spc) is executed.* Although the final loop is performed when I = 10, the last incremented value for I is 11 and the calculator retains this as the value of I.

```
0: for I=1 to 10
1: prt I
2: next I;spc
3: prt I
4: end
```

```
 1.00
 2.00
 3.00
 4.00
 5.00
 6.00
 7.00
 8.00
 9.00
10.00

11.00
```

*Statements following a next statement are not executed until the entire loop is completed. If a gto or gsb statement precedes a next statement on the same line, the gsb or gto isn't executed until the loop is completed.

The next program shows how the for/next loop can be used to assign values to arrays. In this example, the array variables A[1] through A[4] are assigned values.

```
0: dim A[4]
1: for I=1 to 4
2: I↑2→A[I]
3: prt I,A[I];
 spc
4: next I
5: end
```

```
 1.00
 1.00

 2.00
 4.00

 3.00
 9.00

 4.00
16.00
```

For/next loops can be nested or located inside one another up to a depth of 26 (one for each simple variable A through Z). However, one loop cannot overlap another. Before running the following programs, set the print all mode by pressing PRT ALL.

Correct Nesting

```
0: for I=1 to 3
1: for J=4 to 6
2: prt I,J;spc
3: next J
4: next I
5: end
```

```
 1.00
 4.00

 1.00
 5.00

 1.00
 6.00

 2.00
 4.00

 2.00
 5.00

 2.00
 6.00

 3.00
 4.00

 3.00
 5.00

 3.00
 6.00
```

Incorrect Nesting

```
 0: for I=1 to 3                    1.00
 1: for J=4 to 6                    4.00
 2: prt I,J;spc
 3: next I                          2.00
 4: next J                          4.00
 5: end
                                    3.00
                                    4.00

                          error 82 in 4
```

In the incorrect nesting example, the I loop is activated first and then the J loop is activated. The J loop is cancelled at the same time that `next I` is executed because it's an "inner loop". When the I loop is completed and `next J` is finally accessed, `error 82` is displayed. This is because the J loop was cancelled and was not reactivated after the last I loop.

For/next loops can be written in more than one line, as previously shown, or all in one line, like this–

```
0: for I=1 to 5;                    1.00
 prt I;next I;                      2.00
 prt "DONE"                         3.00
                                    4.00
                                    5.00
                          DONE
```

When line 0 is executed, the numbers 1 through 5 are printed as I is incremented by one. When the final value of I is reached, the last statement in the line is executed and `DONE` is printed.

If (STOP) is pressed while the program is running, the program halts when the current line is completely executed. If a for/next loop is completely contained in one line and (STOP) is pressed, the calculator will not stop until the loop is completed. Only (RESET) can stop the execution of the line containing the loop, before its normal termination. This can be avoided by putting the `for` and `next` statements on separate lines.

Each for statement can have only one associated next statement. When a for statement is executed, and there is already an active loop using the same simple variable, then the previous loop is cancelled and the new loop becomes active. In the following example, the first I loop (in line 0) is activated and then cancelled when the second I loop is activated in line 2. When line 4 is executed, control returns to the latest I loop (in line 2).

```
0: for I=1 to
 100
1: prt "I1=",I
2: for I=2 to 10
3: prt "I2=",I
4: next I
```

```
I1=          1.00
I2=          2.00
I2=          3.00
I2=          4.00
I2=          5.00
I2=          6.00
I2=          7.00
I2=          8.00
I2=          9.00
I2=         10.00
```

The optional step size value enables you to specify a step size other than 1, the default step size value. For example–

```
0: for I=0 to
 50 by 10
1: prt I
2: next I
```

```
 0.00
10.00
20.00
30.00
40.00
50.00
```

By adding the optional step size value to the for statement, the simple variable will be incremented by that value each time the next statement is executed. In the previous example, the loop is executed six times – when I = 0,10,20,30,40 and 50. As soon as the incremented value is greater than the final value, the loop is exited.

For/next loops can be decremented by using negative values for the optional step size value. For example–

```
0: for I=50 to
 0 by -10
1: prt I
2: next I
```

```
50.00
40.00
30.00
20.00
10.00
 0.00
```

The step size value does not have to be an integer; fractional numbers are allowed. For example–

```
for I=1 to 10 by .5
```

The initial value, the final value and the step size value can be variables or expressions. For example–

```
5: for I=A to B
 by (B-A)/100
•
•
•
10: next I
```

Once the for statement is executed, the initial, final and step size values can be changed without affecting the number of times the loop is repeated. In the following example, the variables A and B can be used within the loop for other purposes, but the loop itself is repeated only six times.

```
0: 1→A;6→B
1: for I=A to B
2: A-1→A
3: B+1→B
4: prt I,A,B;
 spc
5: next I
6: end
```

```
 1.00
 0.00
 7.00

 2.00
-1.00
 8.00

 3.00
-2.00
 9.00

 4.00
-3.00
10.00

 5.00
-4.00
11.00

 6.00
-5.00
12.00
```

Chapter 3
Subprograms

A subprogram is a programming routine that enables you to repeat an operation many times substituting different values each time the subprogram is called. There are two types of subprograms – subroutines* and functions.

Subroutines

A subroutine subprogram consists of one or more lines of programming which perform a specific task. A subroutine is accessed using a call (`cll`) statement followed by the name of the subroutine, enclosed in single quotes (apostrophes). As many parameters as needed can be used, within the limits of line length.

```
cll 'name' [(parameter 1[, parameter 2, ...])]
  •
  •
  •
"name":
  •
  •
  •
ret
```

The first statement in the subprogram is its name, written as a label (enclosed in quotation marks and followed by a colon). The last statement executed in a subprogram is always a return (`ret`) statement.

*Subroutine subprograms are similar to standard subroutines called by the gosub (`gsb`) statement within a mainframe program. To eliminate confusing the two, subroutine subprograms will be referred to as subroutine subprograms and standard subroutines will be referred to as mainframe subroutines in this manual.

Here's a program with a mainframe subroutine which prints the sum of two numbers—

```
0: 1→A;2→B
1: ssb "name";
 prt "DONE"
2: end
3: "name":
4: prt A+B
5: ret
```

And here's a program that uses a subroutine subprogram to do the same—

```
0: 1→A;2→B
1: cll 'name';
 prt "DONE"
2: end
3: "name":
4: prt A+B
5: ret
```

A look at both programs shows that the subroutines are identical, but the calling statements are different. A ssb statement, followed by the name of the subroutine enclosed in quotes, is used to access the mainframe subroutine, while a cll statement, followed by the name of the subprogram enclosed in apostrophes, is used to access the subroutine subprogram.

There's another difference between the two. The subroutine subprogram is executed immediately, but execution of the mainframe subroutine is delayed until all other statements in that line are executed, as shown by the following printouts.

Mainframe Subroutine

```
DONE
              3.00
```

Subroutine Subprogram

```
              3.00
DONE
```

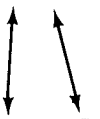
With the mainframe subroutine, DONE is printed before the routine is accessed and executed and program control returns to the line following the one containing the ssb statement.

The subroutine subprogram is accessed and executed immediately so the sum is printed first. Program control then returns to the statement following the call statement and DONE is printed.

In addition to the immediate execute feature, the call statement can pass parameters to the subroutine. In a subprogram, parameters are represented by p-numbers (parameter numbers). This enables you to call the subprogram repeatedly using different values for the parameters each time. Here's an example of this based on the previous two programs—

```
0: ent A,B
1: cll 'name'(A,
 B);prt "DONE"
2: end
3: "name":prt
 p1+p2
4: ret
```

```
1: cll 'name'(A,B)
•
•
•
3: "name":prt p1+p2
```



Passing Parameters

Before covering functions, here's some general information about parameters. A detailed explanation of parameters (p-numbers) is found on page 19.

Parameters that follow the call statement are always enclosed in parentheses and as many parameters as the length of the line allows can be used. These parameters can be constants, simple variables, expressions, r-variables or single elements of an array; entire arrays, strings*, string arrays* and text cannot be used as parameters. In the preceding example, p1 and p2 in line 3 correspond to parameters A and B.

Parameters can be passed back from subroutines to main programs by assigning a value to a p-number which corresponds to a variable. For example, lines 1 and 3 in the previous program can be changed to—

```
1: cll 'name'(A,
 B,C);prt C
•
•
•
3: "name":p1+
 p2+p3
```

Subprograms can be nested (called by another subprogram) as deeply as the calculator memory allows. Each call statement requires a minimum of 26 bytes of memory when executed. That memory is returned when `ret` is executed. If parameters are passed, additional memory is required.

*The String Variables ROM is required to use strings or string arrays in a program.

Functions

A function subprogram consists of one or more lines of programming which perform a specific task. A function is accessed using the name of the function enclosed in single quotes (apostrophes) within an expression or statement in the program. As many parameters as needed can be used, within the limits of line length.

```
'name' [(parameter 1[, parameter 2, ...])]
   •
   •
   •
"name":
   •
   •
   •
ret parameter
```

The first statement in the function itself is its name, written as a label (enclosed in quotation marks and followed by a colon). The last statement executed in a function is always ret followed by a return parameter. The return parameter, like a parameter that follows call statements, can be a simple variable, a constant, an expression, an r-variable or an element of an array. In addition, a return parameter can be an array, a string*, a string array* or text.

Here's an example of a function based on the previous programs—

```
0: 1→A;2→B                                    3.00
1: prt 'name';                    DONE
 prt "DONE"
2: end
3: "name":
4: ret A+B
```

When the program is run, the function is accessed as line 1 is executed. The result of the function is automatically returned and substituted for the name of the function in the statement (prt 'name'). This causes the value of A + B to be printed.

Like a subroutine, a function is executed immediately and program control returns to the function ('name'). A function subprogram can be used in a program wherever an expression can be used.

*The String Variables ROM is required to use strings or string arrays in a program.

A parameter which follows a function call can be a simple variable, a constant, an r-variable, an expression or a single element of an array. (Entire arrays, strings*, string arrays* and text can't be parameters in a function call.) Parameters following a function call are always enclosed in parentheses and as many parameters as the length of the line allows can be used.

Here's an example of a function that uses parameters—

```
0: ent A,B
1: prt 'name'(A,
 B):prt "DONE"
2: end
3: "name":
4: ret p1+p2
```

If the return parameter is omitted from a function subprogram, `error 44` results; if a return parameter follows `ret` in a subroutine subprogram or a mainframe subroutine, it's ignored and no error is displayed.

Functions, like subroutines can be nested as deeply as the calculator memory allows. Each function call requires a minimum of 26 bytes of memory when executed. That memory is returned when `ret` is executed. If parameters are passed, additional memory is required.

*The String Variables ROM is required to use strings or string arrays in a program.

A function subprogram can be used within another subprogram or within an expression. When the function call is placed in the expression, the value returned by the function is used directly in the expression.

Here's an example of a function subprogram that computes the factorial of a number (lines 7 and 8) and uses it in the calculation in line 4 to find the number of combinations of N items taken R at a time.

```
0: fxd 0
1: ent "No. of
 items?",N
2: ent "No. take
 n at a time?",R
3: prt "Combinat
 ions of",N;"ite
 ms taken",R;
 "at a time="
4: prt '!'(N)/
 ('!'(R)*'!'(N-
 R));spc 3
5: end
6: "!":
7: 0→p2;1→p3
8: if p1#p2;p2+
 1→p2;p2p3→p3;
 jmp 0
9: ret p3
```

For 12 items taken 3 at a time the number of combinations is—

```
Combinations of
                 12
items taken       3
at a time=
               220
```

P-Numbers

A subprogram (subroutine or function) enables you to repeat an operation using different values each time the subprogram is called. This is accomplished by following the subprogram call with a list of parameters. When these parameters are passed to the subprogram, a parameter number or p-number is assigned to each parameter in the list. The p-numbers are assigned to the parameters consecutively, starting with p1. The subprogram operation is then performed using the values passed by the subprogram call.

In addition to passed parameters, there are local p-numbers. When allocated, a local p-number is initialized to zero. Local p-numbers are used in a subprogram as needed. Here's an example that uses passed parameters and a local p-number.

```
0: ent A,B
1: prt 'name'(A,
 B)
2: end
3: "name":p1p1+
 p2p2+p6;ret ⌜p6
```

When this program is run, p1 and p2 correspond to the passed parameters A and B, but p6 is a local p-number which, when allocated, is initialized to zero. When the subprogram operation is performed using p1 and p2, the result of the function (⌜p6) is returned and printed.

P-numbers are assigned to parameters consecutively, starting with p1. If you use a local p-number that doesn't follow the passed p-numbers in consecutive order, all p-numbers in between are automatically allocated as local p-numbers. When allocated, these p-numbers are initialized to zero. In the previous example, p3, p4 and p5 are initialized when p6 is allocated and require memory space, even though they are not used.

P0 is also a local p-number but it isn't initialized to zero. Instead, when the subprogram is called, p0 is initialized to the number of parameters passed to the subprogram.

Subprograms can be nested (called by another subprogram) as deeply as the calculator memory allows. In addition, a function subprogram can be used as the parameter for another subprogram (function or subroutine) like this—

```
•
•
20: cll 'SUB'('F
 UN'(A,B))
•
•
```

In the line above, A and B are parameters for the function 'FUN' and the result of the function is the parameter for the subroutine 'SUB'.

When subprograms having parameters are nested, each set of p-numbers is independent of the p-numbers in the next subprogram or level, even though the same p-numbers may be used in each. To illustrate independent p-numbers in nested subprograms, the following example converts a Fahrenheit temperature to Celsius and then outputs both temperatures. Notice that each subprogram uses p1 without affecting the value of the other.

```
0: fxd 0
1: "L":ent "Temp
 erature(F)?",T
2: cll 'Output'(
 T,'C'(T));gto
 "L"
3: "Output":
4: prt "Fahrenhe
 it=",p1,"Celsiu
 s=",p2
5: spc ;ret
6: "C":
7: p1-32*p2
8: ret 5p2/9
```

When the trace mode is established (`trc 0,8`) to monitor the activity of the running program, value assignments for each p-number used are not printed as they are for each simple variable. Instead, as in line 7 of the following traced printout, all p-numbers are referenced by `p=` without indicating the specific p-number.

```
0:
1:
T=              32
2:
6:
7:
p=               0
8:
2:
3:
4:
Fahrenheit=     32
Celsius=         0
5:

2:
1:
```

If a p-number is used as a parameter in a nested subprogram call, there may be some interaction between the p-numbers used in each subprogram. The following program uses nested subprogram calls with parameters to illustrate what happens to p-numbers, variables, expressions and constants in a parameter list when their values are changed in a subprogram.

```
0: fxd 0;2→A
1: cll 'Sub-1'(A
 ,5,1*A)
2: prt "Main",
 "A=",A;stp


3: "Sub-1":cll
 'Sub-2'(A,p1,
 p0,5,1*A)
4: 2p1→p1
5: 2p2→p2;2p3→p3
6: prt "Sub-1",
 p1,p2,p3;spc ;
 ret


7: "Sub-2":
8: 3p1→p1
9: 3p2→p2;3p3→p3
 ;3p4→p4;3p5→p5
10: prt "Sub-2",
 p1,p2,p3,p4,p5;
 spc ;ret
```

The main program (lines 0 through 2) contains the call for Sub-1 with three parameters – A, 5 and 1×A. Sub-1 (lines 3 through 6) calls Sub-2 which has five parameters – A, p1, p0, 5 and 1×A. Sub-2 (lines 7 through 10) triples the value of each parameter and then prints the values. Program control returns to line 4 (Sub-1) and the current value of each parameter is doubled and printed.

Here's a chart that shows the values of the parameters during program execution. The shaded chart below duplicates the chart at the top and shows values before Sub-2 is called.

Sub-1

Passed Parameters	Initial Values	Corresponding p-numbers
A	2	p1
5	5	p2
1×A	2	p3

Sub-2

Passed Parameters	Initial Values	Corresponding p-numbers	Values after line 8	Values after line 9
A	2	p1	6	18*
p1	2	p2	6*	18
p0	3	p3	3	9
5	5	p4	5	15
1×A	2	p5	2	6

*Since A and p1 (in Sub-1) and p1 and p2 (in Sub-2) are all different names for the same value, when p1 (in Sub-2) is tripled in line 8, A and p1 (in Sub-1) and p2 (in Sub-2) are also tripled. The same is true in line 9 when p2 is tripled.

```
Sub-2
 18
 18
  9
 15
  6
```

Sub-1 (Results before calling Sub-2)

Results after return from Sub-2

Passed Parameters	Initial Values	Corresponding p-numbers	Values after Sub-2 execution	Values after line 4	Values after line 5
A	2	p1	18	36	36
5	5	p2	5	5	10
1×A	2	p3	2	2	4

```
Sub-1
 36
 10
  4
```

When program control returns to the main program, the final value of A is printed.

```
Main
A=                    36
```

Although p-numbers can be used only within subprograms, they can be accessed in the live keyboard mode or by stopping execution during a subprogram. A stop statement can be used in a subprogram to stop execution of the subprogram. The current value of any of the p-numbers in the subprogram can be displayed or changed, but new p-numbers can't be created.

Chapter 4
Split and Integer Precision Storage

With the AP and String Variables ROM installed in your HP 9825A, you can compactly store values in split and integer precision formats using string variables. In stored form, the values cannot be used directly in calculations, although they can easily be converted back to numeric values for that purpose. This enables you to store large amounts of data using half (split precision) or one fourth (integer precision) as much memory as full precision storage requires.

Split Precision Storage

Using split precision format, full precision numbers (twelve digit mantissa with sign and exponent) are rounded to six digits and stored in string variables. Only values with exponents in the range of ±63 can be stored using split precision format.

The full to split (f t s) function stores a value in split precision format by encoding the value into four characters* (or bytes) which can then be stored in a previously dimensioned string variable. The location within the string variable (first and last characters) where the encoded value is to be stored should always be specified to eliminate truncation of the rest of the string. The value to be stored must be enclosed in parentheses.

f t s (expression)

*The first character contains the exponent and sign. Each of the three remaining characters contain two BCD (Binary Coded Decimal) digits.

To unpack the value, the split to full (stf) function is used. The string variable must also be enclosed in parentheses.

stf (string variable)

Here's a program that uses the fts function to store a list of ten random numbers. (The rnd function in line 4 generates the random numbers.) The numbers are packed into a string array consisting of ten strings, each four characters long.*

```
0: fxd 11
1: dim A$[10,4]
2: prt "STORING"
3: for I=1 to 10
4: rnd(1)→A;prt
 A
5: fts (A)→A$[I]
6: next I
7: spc
```

The rest of the program unpacks the stored values using the stf function and then prints the numbers. The values being recovered are six digit numbers because they were rounded before they were stored using the fts function.

```
8: prt "RECOVERI
 NG"
9: for J=1 to 10
10: stf(A$[J])→A
 ;prt A
11: next J
12: end
```

Now press (RUN) to start the program and compare these printouts with yours. (Press (RESET) before running any of the example programs in this chapter to get printouts identical to those shown.)

```
STORING
   0.67821900935
   0.38218685905
   0.41914846259
   0.50385703791
   0.74376887685
   0.50962543530
   0.59499108444
   0.38750201234
   0.88919237916
   0.81079087248
```

```
RECOVERING
   0.67821900000
   0.38218700000
   0.41914800000
   0.50385700000
   0.74376900000
   0.50962500000
   0.59499100000
   0.38750200000
   0.88919200000
   0.81079100000
```

*Normally the first and last characters of the string variable being used for storage (i.e., A$[I,1,4]) must be specified, otherwise the remainder of the string may be truncated after the last character stored. However, in this program it's not needed, since each string is only four characters long.

All values are rounded to six digits before they are stored. If you attempt to store a number with an exponent outside the range of −63 to +63 (and flag 14 is clear), error A8 is displayed and flag 15 is set (to 1)*. To avoid this error, you can set flag 14 before the fts function is executed. This causes a default value to be substituted and stored. If the exponent is less than −63, the underflow default value is 0; if the exponent is greater than +63, the overflow default value is ±9.99999e63. Flag 15 is set regardless of whether flag 14 is set or not.

To illustrate what happens when the exponent is less than −63 (underflow), execute these statements−

```
erase a
dim A$[4]:fts(1.2345678e-65)→A$
```

And the display shows−

```
error A8
```

Then set flag 14, and the underflow value is automatically substituted. Key in and execute these statements−

```
sfs14
flt9:fts(1.2345678e-65)→A$
stf(A$)
```

Which substitutes, stores and displays−

```
0.000000000e 00
```

To illustrate what happens when the exponent is greater than +63 (overflow), execute the following statements−

```
erase a
dim A$[4]:fts(1.2345678e65)→A$
```

And the display shows−

```
error A8
```

*Remember that flag 15 is set when ***any*** math error occurs.

By setting flag 14 first, the overflow default value is substituted. Key in and execute these statements–

```
sf 14
flt9;fts(1.2345678e65)→A$
stf(A$)
```

Which substitutes, stores and displays–

```
 9.99999000e 63
```

The next example uses split precision format to store four full precision numbers in each simple string in a string array. As many numbers as the size of the memory and the size of the string array allow, can be stored in split precision format. This means that you can use a string array just like a chart or a table to store data (part numbers, temperatures, etc.) for easy reference. This program also uses the rnd function in line 4, to generate the values to be stored.

```
0: fxd 11
1: dim A$[4,16]
2: for I=1 to 4
3: for J=1 to 4
4: rnd(1)→A;prt
 A
5: fts (A)→A$[I,
 4(J-1)+1,4J]
6: next J;spc
7: next I
8: spc 3
```

Notice that in line 5 three expressions are used to position the value in the appropriate string – the string used for storage (I), the beginning character of the string where the value is to be stored (4(J-1)+1) and the end character where the value is to be stored (4J).

To recall the numbers from split precision format, add these lines to the program and run it.

```
9: for K=1 to 4
10: for L=1 to 4
11: stf(A$[K,
 4(L-1)+1,4L])→A
 ;prt A
12: next L;spc
13: next K
14: end
```

And the printout looks like this–

```
0.67821900935                0.67821900000
0.38218685905                0.38218700000
0.41914846259                0.41914800000
0.50385703791                0.50385700000

0.74376887685                0.74376900000
0.50962543530                0.50962500000
0.59499108444                0.59499100000
0.38750201234                0.38750200000

0.88919237916                0.88919200000
0.81079087248                0.81079100000
0.87512375514                0.87512400000
0.97907806819                0.97907800000

0.40465534756                0.40465500000
0.31514729971                0.31514700000
0.03887905206                0.03887910000
0.69728278019                0.69728300000
```

Some applications require that data be stored in a linear array. By storing data in a single string instead of string arrays, numbers can be stored even more compactly by saving the bytes of memory that would have been allocated for the setting up (overhead) of a string array.

The following example stores numbers in a simple string using the rnd function to generate the values to be stored.

```
0: fxd 9
1: dim A$[80]
2: for I=1 to
 77 by 4
3: rnd(1)→A;prt
 A
4: fts (A)→A$[I,
 I+3]
5: next I
6: spc
```

To recover the numbers, add these lines and run the program.

```
7: for J=1 to
 77 by 4
8: stf(A$[J,J+
 3])→A;prt A
9: next J
10: end
```

To get these printouts–

```
0.678219009
0.382186859
0.419148463
0.503857038
0.743768877
0.509625435
0.594991084
0.387502012
0.889192379
0.810790872
0.875123755
0.979078068
0.404655348
0.315147300
0.038879052
0.697282780
0.414818142
0.862057758
0.990574867
0.073462872
```

```
0.678219000
0.382187000
0.419148000
0.503857000
0.743769000
0.509625000
0.594991000
0.387502000
0.889192000
0.810791000
0.875124000
0.979078000
0.404655000
0.315147000
0.038879100
0.697283000
0.414818000
0.862058000
0.990575000
0.073462900
```

Integer Precision Storage

Using integer precision format, numbers in the range −32768 to +32767 can be stored as integers in string variables even more compactly than split precision format.

The full to integer (`fti`) function rounds a value to an integer and stores it in integer precision format by encoding the value into two characters (or bytes) which can then be stored in a previously dimensioned string variable. The location within the string variable (first and last characters) where the encoded value is to be stored should always be specified to eliminate truncation of the rest of the string. The value to be stored must be enclosed in parentheses.

`fti` (expression)

To recover or unpack the value, the integer to full (`itf`) function is used. The string variable must also be enclosed in parentheses.

`itf` (string variable)

The following program uses the `fti` function to store a list of ten random numbers. (The `rnd` function in line 4 generates the random numbers.) The numbers are packed into a string array consisting of ten strings, each two characters long.*

```
0: fxd 2
1: dim A$[10,2]
2: prt "STORING"
3: for I=1 to 10
4: 250rnd(1)→A;
 prt A
5: fti (A)→A$[I]
6: next I
7: spc
```

The rest of the program unpacks the stored values using the `itf` function and then prints the numbers. The values being recovered are integers within the range previously stated because they were rounded before they were stored.

```
8: prt "RECOVERI
 NG"
9: for J=1 to 10
10: itf(A$[J])→A
 ;prt A
11: next J
12: end
```

*Normally the first and last characters of the string variable being used for storage (i.e., A$[I,1,2]) must be specified, otherwise the remainder of the string may be truncated after the last character stored. However, in the following program it's not needed since each string is only two characters long.

Now press (RUN) to start the program and compare the listings.

```
STORING                    RECOVERING
         169.55                    170.00
          95.55                     96.00
         104.79                    105.00
         125.96                    126.00
         185.94                    186.00
         127.41                    127.00
         148.75                    149.00
          96.88                     97.00
         222.30                    222.00
         202.70                    203.00
```

If you attempt to store a number outside the range −32768 to +32767 using integer precision format (and flag 14 is clear) `error A8` is displayed and flag 15 is set.*

To avoid `error A8`, you can set flag 14 before the `fti` function is executed. This causes an overflow default value (−32768 or +32767) to be substituted. Flag 15 is set regardless of whether flag 14 is set or not.

To illustrate overflow, execute these statements−

```
erase a
dim A$[2]⏎fti(-54321)→A$
```

And the display shows−

```
error A8
```

By setting flag 14 first, the overflow default value is substituted without displaying an error. Key in and execute these statements−

```
sf 14
fxd 0⏎fti(-54321)→A$
itf(A$)
```

And the default value is automatically substituted, stored and displayed−

```
-32768
```

*Remember that flag 15 is set when ***any*** math error occurs.

If the value to be packed is between −.5 and .5, then it is rounded to zero as shown here−

```
fxd 11:fti(.12345)→A$:itf(A$)
```

```
0.00000000000
```

Here's an example that uses integer precision format to store eight values in each simple string of a string array. As many numbers as the size of the memory and the size of the string array allow, can be stored in integer precision format. This means that you can use a string array to store data in a table or chart for easy reference. This program also uses the rnd function to generate the values to be stored.

```
0: fxd 2
1: dim A$[4,8]
2: for I=1 to 4
3: for J=1 to 4
4: 250rnd(1)→A:
 prt A
5: fti (A)→A$[I,
 2(J-1)+1,2J]
6: next J:spc
7: next I
8: spc 3
```

Notice that in line 5, three expressions are used to position the value in the appropriate string - the string being used for storage (I), the beginning character where the value is to be stored (2(J-1)+1) and the last character where the value is to be stored (2J).

To recall the numbers from integer precision format, add these lines to the program and run it.

```
9: for K=1 to 4
10: for L=1 to 4
11: itf(A$[K,
 2(L-1)+1,2L])→A
 :prt A
12: next L:spc
13: next K
14: end
```

And the printout looks like this—

```
169.55                    170.00
 95.55                     96.00
104.79                    105.00
125.96                    126.00

185.94                    186.00
127.41                    127.00
148.75                    149.00
 96.88                     97.00

222.30                    222.00
202.70                    203.00
218.78                    219.00
244.77                    245.00

101.16                    101.00
 78.79                     79.00
  9.72                     10.00
174.32                    174.00
```

Some applications require data storage in a string or linear array. By storing data in a single string instead of string arrays, numbers can be stored even more compactly by saving the memory that would have been allocated for the setting up (overhead) of a string array.

The following example stores numbers in a single string using the rnd function to generate the values to be stored.

```
0: fxd 2
1: dim A$[40]
2: for I=1 to
 39 by 2
3: 250rnd(1)→A;
 prt A
4: fti (A)→A$[I,
 I+1]
5: next I
6: spc
```

To recover the numbers, add these lines and run the program.

```
7: for J=1 to
 39 by 2
8: itf(A$[J,J+
 1])→A;prt A
9: next J
10: end
```

And the printout shows–

```
169.55          170.00
 95.55           96.00
104.79          105.00
125.96          126.00
185.94          186.00
127.41          127.00
148.75          149.00
 96.88           97.00
222.30          222.00
202.70          203.00
218.78          219.00
244.77          245.00
101.16          101.00
 78.79           79.00
  9.72           10.00
174.32          174.00
103.70          104.00
215.51          216.00
247.64          248.00
 18.37           18.00
```

Summary

Full precision numbers (twelve digit mantissa plus exponent and sign) can be compactly stored in strings or in string arrays using one of two possible storage formats. Split precision format packs data in half the memory space that full precision storage requires and integer precision format packs data in one fourth the memory space that full precision storage requires.

Storing a number using full precision format requires eight bytes of memory. Using split precision format, only four bytes of memory are required to store a number. This is accomplished by limiting the range and precision of the numbers that can be stored. Using split precision format, the number is rounded to six digits before storage. In addition, the exponent must be in the range −63 to +63. If it's not in that range, then flag 15 is set (to 1) and `error 88` is displayed (if `flg 14` is clear). To avoid `error 88`, you can set flag 14 before executing the `fts` function, causing a default value to be substituted and stored. For an overflow error, the default value is ±9.99999e63; if it's an underflow error the default value is 0.

The following program illustrates how the drnd function internally rounds the value to be packed to six digits before storage in split precision format.

```
0: dim A$[4]
1: for I=1 to 10
2: rnd(1)→A
3: fts (A)→A$
4: if drnd(A,
 6)#stf(A$);prt
 A,"Different";
 stp
5: next I
6: prt "ALL OK";
 spc 2
7: end
```

```
ALL OK
```

Using integer precision format, only two bytes of memory are required to store a number. Integers in the range −32768 to +32767 can be stored using integer precision format. If you attempt to store a number that's outside of this range using integer precision format, flag 15 is set and error 48 is displayed (if flag 14 is clear). To avoid error 48, you can set flag 14 before executing the fti function, causing an overflow default value (−32768 or +32767) to be substituted and stored. If the value to be packed is between −.5 and .5, then it is rounded to zero.

This program shows how the prnd function internally rounds the value to be packed to the nearest integer value before storage in integer precision format.

```
0: dim A$[2]
1: for I=1 to 10
2: 32767rnd(1)→A
3: fti (A)→A$
4: if prnd(A,
 0)#itf(A$);prt
 A,"Different";
 stp
5: next I
6: prt "ALL OK";
 spc 2
7: end
```

```
ALL OK
```

When storing numbers in a string variable using the fts or fti functions, the locations where storage begins and ends within the string variable must be specified; otherwise the string may be truncated after the last character stored.

Chapter 5
Cross Reference Statement

Description

The cross reference (`xref`) statement prints each variable used in your program followed by the line numbers in which it appears.

```
xref
```

For programs with many variables, the `xref` statement aids in keeping track of these variables and their locations in your program. The `xref` statement can be executed from the keyboard, in the live keyboard mode or within a program. The variables used in the program – simple, numeric array, string and r-variables – are printed, in that order. Within each type, the variables are arranged alphabetically.

When `xref` is executed, it searches the program once for each of the 79 possible variables (26 simple, 26 numeric array, 26 string and r-variables*). The `xref` statement does not list references to p-numbers (see Chapter 3) or variables used in Matrix ROM statements (see the Matrix ROM Programming Manual).

*All r-variables are considered as one for this statement and they appear together at the end of the cross reference listing.

The following program finds prime numbers and their logarithms using simple, numeric array, string and r-variables.

```
0: dim P$[1000→r
 0],L[r0/2→r1],
 D$[16]
1: 0→I;2→X
2: "Loop":
3: if not 'Prime
 '(X);goto "Not
 Prime"
4: I+1→I
5: fti (X)→P$[2(
 I-1)+1]
6: log(X)→L[I]
7: fxd 0;"Prime"
 &str(I)&"="&str
 (X)→D$;fxd 4;
 dsp D$;",Log=",
 L[I]
8: if I<r1;goto
 "Not Prime"
9: dsp "Done";
 end
10: "Not Prime":
11: X+1→X;goto
 "Loop"
12: "Prime":
13: rp1→p2
14: for J=1 to I
15: if (itf(P$[2
 (J-1)+1])→p3)>p
 2;ret 1
16: if Xmodp3=0;
 ret 0
17: next J;ret 1
```

*The String ROM is required to run this program, since it uses the string (str) function and the concatenation (&) operator.

By executing the `xref` statement, these variables are listed–

```
I     1     4     4
5     6     7     7
8     14

J     14    15    17

X     1     3     5
6     7     11    11
16

L[*]0     6     7

D$    0     7     7

P$    0     5     15

r     0     0     0
8
```

Appendix

A P Syntax

Syntax Conventions

The following conventions apply to the syntax for the statements and functions found below.

dot matrix - All items in dot matrix are required, exactly as shown.

[ ] - All items in square brackets are optional, unless the brackets are in dot matrix.

For/Next Loops

for simple variable = initial value to final value [by step size value]
•
•
•
next same simple variable

Subprograms

Subroutines

cll "name" [(parameter $_1$[; parameter$_2$; ...])]
•
•
•
"name" :
•
•
•
ret

Functions

"name" [(parameter $_1$[; parameter $_2$; ...])]
•
•
•
"name" :
•
•
•
ret parameter

Split and Integer Precision Storage Functions

Split Precision Storage

```
fts [expression ]
stf [string variable ]
```

Integer Precision Storage

```
fti [expression ]
itf [string variable ]
```

Cross Reference Statement

```
xref
```

HEWLETT hp PACKARD

SALES & SERVICE OFFICES

UNITED STATES

ALABAMA
8290 Whitesburg Dr., S.E.
P.O. Box 4207
Huntsville 35802
Tel: (205) 881-4591
TWX: 810-726-2204

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
TWX: 510-955-4099

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

P.O. Box 13910
6177 Lake Ellenor Dr.
Orlando 32809
Tel: (305) 859-2900
TWX: 810-850-0311

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

GEORGIA
P.O. Box 105005
Atlanta 30348
Tel: (404) 434-4000
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

***St. Joseph**
Tel. (217) 469-2133

INDIANA
7301 North Shadeland Ave.
Indianapolis 46250
Tel: (317) 842-1000
TWX: 810-260-1796

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

***KANSAS**
Derby
Tel: (316) 267-3655

KENTUCKY
Medical/Calculator 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
TWX: 810-955-5524

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

New York City
Manhattan, Bronx
Contact Paramus, NJ Office
Tel: (201) 265-5000
Brooklyn, Queens, Richmond
Contact Woodbury, NY Office
Tel: (516) 921-0300

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) 455-2486
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
TWX: 510-926-1516

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
TWX: 910-830-6862

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

PENNSYLVANIA
111 Zeta Drive
Pittsburgh 15238
Tel: (412) 782-0400
Night: 782-0401
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
***Memphis**
Medical Service only
Tel: (901) 274-7472

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

TEXAS
P.O. Box 1270
201 E. Arapaho Rd.
Richardson 75080
Tel: (214) 231-6101
TWX: 910-867-4723

P.O. Box 27409
6300 Westpark Drive
Suite 100
Houston 77027
Tel: (713) 781-6000
TWX: 910-881-2645

205 Billy Mitchell Road
San Antonio 78226
Tel: (512) 434-8241
TWX: 910-871-1170

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
2914 Hungary Springs Road
Richmond 23228
Tel: (804) 285-3431
TWX: 710-956-0157

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; Skokie, Illinois. Their complete addresses are listed above.

***Service Only**

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 931
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 1L9
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.C.e.I.
Lavalle 1171-3° Piso
Buenos Aires
Tel: 35-0436, 35-0627, 35-0341
Telex: Public Booth No. 9
Cable: HEWPACK ARG

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.
Praca Dom Feliciano, 78-8° andar (Sala 806/8)
90000-**Pôrto Alegre**-RS
Tel: 25-84-70-DDD (0512)
Cable: HEWPACK Pôrto 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: 350001 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 Compañía Ltda.
Av. Rio Amazonas No. 239
P.O. Box 2925
Quito
Tel: 527-088,527-804
Cable: ASTOR Quito

Calculators Only
Computadoras y Equipos Electrónicos
P.O. Box 2695
Avda. 12 De Octubre No. 2207
Quito
Tel: 233869, 236783
Telex: 02-2113 Sagita Ed
Cable: Sagitra-Quito

EL SALVADOR
IPESA
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 Científicos 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: 413485
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: 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: 35-00-07, 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

AUSTRIA
Hewlett-Packard Ges.m.b.H
Handelska 52/3
P.O. Box 7
A-1205 **Vienna**
Tel. (0222) 35 16 21 to 32
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

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**
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
Telex: 510957F

Hewlett-Packard France
Agence Régionale
Aéroport principal de
Marseille-Marignane
F-13721 **Marignane**
Tel. (91) 89 12 36
Cable: HEWPACK MARGN
Telex: 410774F

Hewlett-Packard France
Agence Régionale
63, Avenue de Rochester
F-35000 **Rennes**
Tel. (99) 36 33 21
Cable: HEWPACK 74912
Telex: 740912F

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

Medical/Calculator Only
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 130
D-7030 **Böblingen**, Württemberg
Tel. (07031) 667-1
Cable: HEPAK Böblingen
Telex: 07265739 bbn

Hewlett-Packard GmbH
Technisches Buero Dusseldorf
Emanuel-Leutze-Str. 1 (Seestern)
D-4000 **Dusseldorf**
Tel. (0211) 59711
Telex: 85/86 533 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
Mellendorfer 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
Telex: 52 49 85
Cable: HEWPACKSA München
Telex: 0524985

(West Berlin)
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. 521 915
Cable: INTEKNIKA
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

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 19
IR-**Tehran**
Tel. 85 10 82/86

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

ITALY
Hewlett-Packard Italiana S.p.A.
Casella postale 3645
I-20124 **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: 32046 via Milano

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

Hewlett-Packard S.p.A.
Via G. Armellini 10
I-00143 **Roma**-Eur
Tel. (06) 54 69 61
Telex: 61514
Cable: HEWPACKIT Roma

Hewlett-Packard Italiana S.p.A.
Via San Quintino, 46
I-10121 **Turino**
Tel. 53 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 05

Hewlett-Packard Italiana S.p.A.
Via Amerigo Vespucci, 9
I-80142 **Napoli**
Tel. (81) 33 77 11

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

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

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
Ul. Stawki 2 6P
00-950 **Warsaw**
Tel: 39 67 43
Telex: 81 24 53 hepa pl

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

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 Technical Office
BD. N. Balcesci 16
Bucharest
Tel. 1580 23/138885
Telex: 10440

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**
Tel: 23 83 06/23 82 06

Calculators Only
Hewlett-Packard Española S.A.
Gran Via Fernando El Catolico, 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
Frotalsgatan 30
S-421 32 **Vastra Frolunda**
Tel: (031) 49 09 50
Telex: 10721 Via Bromma Office

SWITZERLAND
Hewlett-Packard (Schweiz) AG
Zurcherstrasse 20
P.O. Box 307
CH-8952 Schlieren **Zurich**
Tel. (01) 98 18 21
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

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

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 1DJ
Tel: (061) 928 9021
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.
4th Floor
Wedge House
799, London Road
GB-**Thornton Heath** CR4 6XL
Surrey
Tel: (01) 684 0103/0105
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

Hewlett-Packard Ltd.'s registered
address for V.A.T. purposes
only:
70, Finsbury Pavement
GB-**London**, EC2A1SX
Registered No. 690597

USSR
Hewlett-Packard Representative
Office USSR
Pokrovsky Boulevard 4/17, Suite 12
Moscow 101000
Tel: 294 2024
Telex: 7825 hewpak SU

YUGOSLAVIA
Iskra-standard/Hewlett-Packard
Office
Miklosiceva 38/VII
61000 **Ljubljana**
Tel: 315-879/321-674
Telex: 31583 YU HEWPAK

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

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

GUAM
Medical/Pocket Calculators Only
Guam Medical Supply, 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.
Laipatnagar
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

IRAN
Hewlett-Packard Iran Ltd.
Mir-Emad Ave.
14th Street, No. 19
IR-**Tehran**
Tel: 85 1082/86
Telex: 21 25 74

ISRAEL
Electronics & Engineering Div.
of Motorola Israel Ltd.
16 Kremenetski Street
P.O. Box 25016
Tel-Aviv
Tel: 38973
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.
Nisse Ibaraki Building
2-8 Kasuga 2-chome, 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.
Tangawa 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

LEBANON
Constantin E. Macridis
Clemenceau Street 34
P.O. Box 7213
RL-**Beirut**
Tel: 36 63 97/8
Telex: 21114 Leb
Cable: ELECTRONUCLEAR Beirut

MALAYSIA
Teknik Mutu Sdn. Bhd.
2 Lorong 13/6A
Section 13
Petaling Jaya, **Selangor**
Tel: 773455(5 lines)

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
N68/770 Oyo Road
Oluseun House
P.M.B. 5402
Ibadan
Tel: 61577
Telex: 31231 TEIL Nigeria
Cable: THETEIL Ibadan

The Electronics Instrumenta
tions 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.
388, Satellite Town
Rawalpindi
Tel: 41924
Cable: FEMUS Rawalpindi

PHILIPPINES
The Online Advanced Systems
Corporation
6th Floor, Yujuico Building
560 Quintin Paredes Street
Binondo, **Manila**
Tel: 40-05-41, 40-05-31
Telex: 3327 GENBANK

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 Taiwan
38, Po-Ai Lane, San Min Chu,
Kaohsiung
Tel: (07) 242318

Analytical Only
San Kwang Instruments Co., Ltd.,
No. 20, yung Sui Road
Taipei, 100
Tel: 3713171-4
Telex: 22894 SANKWANG
Cable: SANKWANG TAIPEI

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

THAILAND
UNIMESA Co., Ltd.
Elcom Research Building
Bangjak Sukumvit Ave
Bangkok
Tel: 932387, 930038
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

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**
Telex: 21-6588
Cable: HEWPACKSA Athens

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

Subject Index

Error Messages

error A0 Relational operator in for statement not allowed. No closing apostrophe in subprogram name.

error A1 For statement has no matching next statement.

error A2 A next statement encountered without a previous for statement.

error A3 Non-numeric parameter passed as a p-number.

error A4 No return parameter for a function subprogram.

error A5 Improper p-number reference since no functions or subroutines are running.

error A6 Attempt to allocate local p-numbers from the keyboard.

error A7 Wrong number of parameters in fts, stf, fti or itf function. Parameter for stf or itf function must be a string (not a numeric). Parameter for stf or itf function contains too few characters.

error A8 Overflow or underflow in fts function or overflow in fti function.

error A9 String Variables ROM missing for stf or itf functions.

These mainframe errors have additional meanings when the AP ROM is installed.

error 09 Attempt to execute a next statement from keyboard while for/next loop with same variable is executed in program or from program while for/next loop with same variable is executed from keyboard. Attempt to call a function or subroutine from keyboard.

error 26 Negative p-number reference.

error 32 Non-numeric value in for statement. Non-numeric parameter in fts or fti function.

error 38 Memory overflow during function or subroutine call.

error 48 Memory overflow while using for statement or while allocating local p-numbers.