HEWLETT-PACKARD 9830A CALCULATOR
11289B ADVANCED PROGRAMMING II ROM

OPERATING MANUAL

OPERATING and PROGRAMMING MANUAL

ADVANCED PROGRAMMING II ROM 11289B

9830A CALCULATOR SHOWN WITH 9866A PRINTER

HEWLETT–PACKARD CALCULATOR PRODUCTS DIVISION

P.O. Box 301, Loveland, Colorado 80537, Tel. (303) 667-5000

(For World-wide Sales and Service Offices see rear of manual.)

PREFACE

With the ability of the 9880A/B Mass Memory to access and store large amounts of data, came the need for fast and easy routines to handle this data. The Advanced Programming II (APII) ROM supplies these routines.

Basic business applications like payroll and inventory control are speeded up when routines like SORT, SEARCH and TRANSFER are used. In addition, these routines make programming the 9830A and its peripherals much simpler. Other features of the APII ROM include error recovery (SERROR), numeric to string conversion (STRING), FLAG and BEEP. With SERROR and BEEP, programs can be designed so that anyone can operate the calculator successfully. There is no need to become familiar with error messages and no need to be at the calculator whenever a program is running. Flags make branching easier and conserve memory space, while the STRING statement enables the calculator and its peripherals to write monetary values in any currency format.

The APII ROM combines the useful business features of other HP 9830A ROMs with a number of convenient and time-saving routines to contribute to the programming capability of the 9830A for many commercial and business applications.

TABLE OF CONTENTS

TABLE OF CONTENTS

Chapter 1

GENERAL INFORMATION

EQUIPMENT SUPPLIED

One operating manual (P/N 09830-90019) is supplied with the Advanced Programming II ROM (Read-Only-Memory) P/N 11289B.

INSPECTION

Refer to Appendix A in the 9830A Calculator Operating and Programming Manual for the procedure used to verify the operation of ROMs.

INSTALLATION

The complete procedure to install a plug-in block is in the Operating and Programming Manual for the 9830A calculator. A few reminders follow.

- The block can be installed in any of the five external ROM slots. (See the NOTE below.)
- Switch the calculator off before installing or removing a block.
- The label on the block should be right-side-up and facing the ROM door when the block is correctly installed.
- Be sure that the block is properly mated to the connector at the back of the slot before switching the calculator on.

CHARACTERISTICS

The APII ROM requires five words of calculator Read-Write-Memory when installed in the calculator.

Keyboard execution is allowed on all APII ROM statements except the SERROR statement.

No new error notes are associated with the AP II ROM.

> **NOTE**
>
> The APII ROM contains three statements which are found in other 9830A ROMs (i.e., TRANSFER in API, BEEP in API and SERROR in DATA COMM 2). When using one of these statements from a particular ROM, ERROR 1 results if that ROM is not installed when the statement is executed.

SYNTAX

brackets [ ] - items enclosed in brackets are optional
coloring - items printed in color must appear as shown

MASS MEMORY

Many of the examples found in this manual use data files which are stored on tape cassette. To convert to data storage on the mass memory, a space on the mass memory must be reserved. Each record on the mass memory corresponds to a data file on the tape cassette. (NOTE: The mass memory ROM must be installed in the top ROM slot of the calculator.) By executing the statement below, prior to running a program, space will be reserved on the mass memory for the data used in the examples in this manual.*

```
OPEN "INVEN",20
```

The following changes must be made to all applicable program lines throughout this text.

Replace	`LOAD DATA I`	with	`READ#1,I;P,D$,M$,C`
Replace	`STORE DATA I`	with	`PRINT#1,I;P,D$,M$,C`

Also the FILES statement below should follow the DIM statement in all applicable programs.

(LINE) `25 FILES INVEN`

All program lines throughout the text that must be changed to accommodate the mass memory are preceded by "☆". (See the Appendices.)

For more information on the use of the mass memory, see the 9880A/B Mass Memory Operating Manual.

PRINTOUTS

A number of the printouts for the programs used in this manual are found in Appendix IV.

REQUIREMENTS

It is assumed that you are already familiar with BASIC programming and with the operating procedures for the HP 9830A calculator.

*When random (record by record) access is used with mass memory files, each record has a length of 256 words, while only 26 words are required for the data files in the examples in this manual.

Chapter 2

TRANSFER STATEMENT

DESCRIPTION

The TRANSFER statement places data, input as strings, into integer arrays for storage. TRANSFER also moves integer array data back to strings when needed. To use the TRANSFER statement, the string variables ROM must be installed in your calculator.

Using the TRANSFER statement, strings longer than 255 characters can be stored. In addition, more than 26 strings can be used and these may be stored together as a numeric array in one tape file. The TRANSFER statement allows strings to be retained in memory from program to program if an array is specified as integer-precision in a COM statement. TRANSFER also allows the SORT operation to be performed on string variables since they are stored as numerics.

The TRANSFER statement requires that an array be dimensioned in a DIM or COM statement as integer-precision. In the TRANSFER statement, the numeric array must be subscripted. The first subscript indicates the array row in which transfer is to begin; the second subscript indicates the column position at which transfer is to begin. For example, TRANSFER A$ TO A(2,1) means that transfer begins in row 2, column 1 of numeric array A.

NOTE

Both the API and the APII ROMs contain the TRANSFER statement. A program written using TRANSFER with *only* APII installed, will not run when *only* API is installed, or vice versa. (ERROR 1 results.) If API is exchanged for APII, or vice versa, all lines containing TRANSFER must be re-entered.

SYNTAX

To transfer a string to a numeric array:

TRANSFER string name [subscripts] TO integer array name (subscripts)

To transfer data from a numeric array to a string:

TRANSFER integer array name (subscripts) TO string name [subscripts]

The string name and the array name are letters from A to Z and the string name is followed by a dollar sign ($). The string name may optionally have subscripts following it. Without subscripts, the entire string is transferred; with subscripts, the specified substring is transferred.

String characters are stored into the array row-by-row, with two string characters in each array element. If a numeric array is dimensioned - AI(3,8) - and A$ is 20 characters, executing TRANSFER A$ TO A(1,1) would fill the array as shown.

Columns

Rows	1	2	3	4	5	6	7	8
1								
2								
3								

Two characters from the string are stored in each numeric array element, so that only ten elements of the array are filled. To pack another string, called B$, in the same numeric array, TRANSFER B$ TO A(2,3) would be executed.

Although this process would maximize the use of space in the numeric array, for referencing purposes it is often easier, when storing more than one string, to transfer one string per numeric array row. The program below illustrates this.

```
10 DIM AI[3,10],A$[20]
20 FOR I=1 TO 3
30 INPUT A$
40 TRANSFER A$ TO A[I,1]
50 NEXT I
```

The row length is dimensioned as exactly half the maximum string length in line 10 above. When a string of up to 20 characters is input, it fills the first row of the numeric array. (Remember, two characters are stored per integer array element.) Since the INPUT and TRANSFER loop is performed three times, three strings all with the same string name (A$), are saved, each in a separate row of the integer array. This allows more than 26 strings to be used in a single program.

To transfer the numeric array back to a string and output the three strings, the following loop can be used.

```
60 FOR J=1 TO 3
70 TRANSFER A[J,1] TO A$
80 PRINT A$
90 NEXT J
1000 END
```

EXAMPLES

In the following example the TRANSFER statement stores strings in the memory as an integer array. Then the numeric data can be stored on tape, either one row of the array in each tape cassette data file, or the whole array in one tape file.

To store more than 26 strings, a loop is used to input strings, all of which have the same string name. Each string is transferred to a row of the numeric array for storage after it is input.

The following program stores 90 strings, each 80 characters long. In this example the number of strings is limited by the largest array dimension (256) and the size of the calculator's memory.

```
10 DIM AI[90,40],A$[80]
20 FOR I=1 TO 90
30 INPUT A$[1,80]
40 TRANSFER A$ TO A[I,1]
50 NEXT I
```

Once transferred, the string can be recovered using a TRANSFER statement. The location in the array where the string is stored is included in the TRANSFER statement as is the string name used to output the string. To recover A$ from rows 1 through 90 and columns 1 through 40 of array A, add these lines:

```
60 FOR J=1 TO 90
70 TRANSFER A[J,1] TO A$
80 WRITE (15,90)A$
90 FORMAT B
100 NEXT J
1000 END
```

The string is transferred from the array back to a string using the same subscripts used when it was input. (To output 80 characters in one line, a WRITE statement is used. The WRITE statement references a "dummy" FORMAT statement in line 90, FORMAT B, indicating here that no format specifications are used in printing the strings.)

APPENDIX EXAMPLE

Appendix II contains a program which permits the storage of strings longer than 255 characters in an integer array.

NOTES

Chapter 3

SORT STATEMENT

DESCRIPTION

The SORT statement sequences any row or column of a numeric array in ascending order. Up to six rows or columns of a numeric array can be sorted at one time. More than six rows or columns of an array are sequenced using multiple SORT statements. Row sorts retain column integrity while column sorts retain row integrity.

In addition, by using the string variables ROM with the TRANSFER statement, alpha data, or strings, can be sorted. With TRANSFER and an array dimensioned as integer-precision, two characters of the string are stored in each array element. In this way, a sort on up to 12 characters of a string can be done at one time. Deeper string sorts require more than one SORT statement.

A primary sort is executed when only one row or column number is specified in the SORT statement. If two or more primary elements in the rows or columns of an array are identical, a secondary sort is required. The SORT statement then contains more than one row or column number. Primary sorts are much faster than secondary sorts because of the APII ROM's sorting techniques.* Row sorts are also faster than column sorts.

SYNTAX

To sort rows:

SORT array name, R, row number [, secondary row numbers]

To sort columns:

SORT array name, C, column number [, secondary column numbers]

The array name is a letter from A to Z representing the array to be sorted and R or C specifies either a row or column sort. If only one row or column number is specified, then row or column number indicates the number of the row or column on which the primary sequencing is performed. If more than one row or column number is specified, then a secondary sort is performed on all rows or columns specified. Up to six row or column numbers may be specified in one SORT statement. Row and column numbers are automatically rounded if a non-integer number is used, e.g., 1.4=1 but 1.8=2. Row or column numbers must be within the range of the array dimensioned, otherwise ERROR 42 is displayed during execution. Expressions can be used in place of primary and secondary row and column numbers.

*Primary sorts use shell sort techniques; secondary sorts use bubble sort techniques. A description of each is found in Appendix III.

SORTING NUMERIC ARRAYS

Example 1

The program below illustrates a simple numeric sort on column one of array A. When only *one* column (or row) is sorted, a primary sort is executed.

```
10 DIM A[3,5]
20 FOR I=1 TO 3
30 FOR J=1 TO 5
40 INPUT A[I,J]
50 NEXT J
60 NEXT I
70 SORT A,C,1
80 FOR I=1 TO 3
90 PRINT A[I,1],A[I,2],A[I,3],A[I,4],A[I,5]
100 NEXT I
1000 END
```

Then run the program and input:

Columns	(1)	(2)	(3)	(4)	(5)	
	2	3	4	5	6	(Row 1)
	2	2	2	2	2	(Row 2)
	2	3	2	3	2	(Row 3)

Line 70 sorts only column one. Since 2 occurs in column one of all three rows, a further or deeper sort is required to actually sequence the array in ascending order. By changing line 70 to:

```
70 SORT A,C,1,2,3,4,5
```

and pressing CONT 70 EXECUTE, the array is completely sorted and row integrity is maintained. All rows look exactly as they did when input, except for their position:

Row 1:	2	2	2	2	2	(Input as Row 2)
Row 2:	2	3	2	3	2	(Input as Row 3)
Row 3:	2	3	4	5	6	(Input as Row 1)

The SORT statement sequences only as many columns (or rows) as necessary to get correct sorted order and still retain row (or column) integrity. In the previous example, SORT is executed on column one and when two compared numbers are equal, column two is checked. If the two compared numbers of column two are equal then column three is checked and so on until the sort is completed. In the previous example only the first three columns are needed to complete the sort even though the instruction was to sort all five columns. In this way, maximum efficiency is achieved.

In general, row sorts work in the same way as do column sorts. Using the sorted data from the previous program:

Columns	(1)	(2)	(3)	(4)	(5)	
	2	2	2	2	2	(Row 1)
	2	3	2	3	2	(Row 2)
	2	3	4	5	6	(Row 3)

and changing line 70 to SORT A, R, 1, a primary sort is executed on row one only. Since all entries in row one are the same, the array remains unchanged. To completely sort the array, change line 70 to:

```
70 SORT A,R,1,2,3
```

and press CONT 70 EXECUTE. Since all entries in row one are the same, row two is checked for each comparison and then row three is checked when the row two elements are found to be equal. When the sort is complete the array is printed.

New columns	(1)	(2)	(3)	(4)	(5)	
	2	2	2	2	2	(Row 1)
	2	2	2	3	3	(Row 2)
	2	4	6	3	5	(Row 3)

Column integrity is maintained while the positions in the rows are rearranged in ascending order, e.g., column two is now fourth, column three is second, etc. Only as many rows as necessary are checked to sequence the array.

Column or row integrity must be maintained because the information in data files used in most business and commercial applications is usually related, either column-wise or row-wise. For example, John Jones' data file contains his name, address, employer and social security number. Sorting by one item, say social security number, retains row integrity so that his name, address and employer remain associated with his social security number.

Example 2

Appendix I provides a program and data for the following examples. Each of the twenty data files stored on tape contains an item part number, the item's description, manufacturer and cost.

Although storage by data file allows large amounts of information to be stored economically, access to this information can be slow and difficult. A directory to all the files on a tape can be created and stored at the beginning of each tape. The program below reads the information from each file and stores the part number, cost and file number of each item in a full-precision array of 20 rows and 3 columns. (This directory will be used in Chapter 4.)

```
10 COM P,D$[20],M$[12],C
20 DIM A[20,3]
30 FOR I=1 TO 20
40 LOAD  DATA I
50 A[I,1]=P
60 A[I,2]=C
70 A[I,3]=I
80 NEXT I
```

To sort array A by part number and then store it in file zero for use as a directory, add these lines and run the program.

```
90 SORT A,C,1
100 STORE  DATA 0,A
1000 END
```

For a printout of this directory, add these lines and run the program.

```
110 FOR J=1 TO 20
120 PRINT A[J,1];A[J,2];A[J,3]
130 NEXT J
```

Now it is not necessary to check each file for information about a certain part number. Instead, the directory can be checked to locate the specific file where the part number is stored.

As previously stated, expressions can be used in place of row or column numbers in a SORT statement. By modifying the program on the previous page, a primary sort can be performed on the part number (input as P in the data files) *or* on the cost (input as C in the data files), depending on the user's needs. First delete lines 90 and 100 and add these lines:

```
82 DISP "SORT BY PART NO=1;BY COST=2";
84 INPUT X
86 SORT A,C,X
```

Depending on the value input by the user, a primary sort by part number or a primary sort by cost is executed. (See page IV-2 in Appendix IV for the printouts.)

A sort by costs shows that a number of costs are equal, like $5.00 for two of the items and $249.99 for three of the items. Since a match in column two exists, a further sort by column one, part numbers, can be executed by deleting lines 82 through 86 and adding:

```
90 SORT A,C,2,1
```

After the program is run, the two $5.00 items are still first but part number 7051 now follows part number 1086. The same is true where other costs in column two are identical. A secondary column sort provides a deeper sort and is useful only when two items in the primary column are identical. (See page IV-3 in Appendix IV for the printout.)

Secondary sorts are used in many business applications. For example, a sort can be executed first by department and then alphabetically by employee within each department, or first by manufacturer and then by part number for each manufacturer. Up to six rows or columns can be sorted using one SORT statement. For deeper sorts, more than one SORT statement (each with up to six row or column numbers) can be executed. Most business applications require this type of multi-level sequencing.

SORTING STRING VARIABLES

Example 3

Alpha data, or strings, such as manufacturer or employee names, can be sorted using TRANSFER with the SORT statement. Since two characters per array element are stored in an integer-precision array, up to 12 characters of a string may be sequenced using a single SORT statement. For deeper sorts, more than one SORT statement can be used.

In general, primary and secondary string sorts work in the same way as do numeric sorts, since once a string is transferred, it is stored (and sorted) as a numeric array. The characters are sequenced by the binary value of their ASCII codes (e.g., !, $, 1, 2, A, B, a, b, etc.).

Using the previous program on page 3-3, the descriptions (input as D$) and the manufacturers (input as M$) can be transferred to an integer-precision array for sorting. To do this, an integer-precision array must be dimensioned using a DIM statement and the strings must be transferred to (and from) the array.

The program below saves the part number (stored in each data file as P) and the cost (stored as C) in the full-precision array A (20,2).

```
10 COM P,D$[20],M$[12],C
20 DIM A[20,2],BI[20,17]
30 FOR I=1 TO 20
40 LOAD  DATA I
50 A[I,1]=P
60 A[I,2]=C
```

Then the descriptions (stored in each data file as D$) and manufacturers (stored as M$) are transferred into array B(20,17). The first column of array B will contain an entry row number (see line 70) which is used later to preserve the correspondence between the entries in arrays A and B.

```
70 B[I,1]=I
80 TRANSFER D$ TO B[I,2]
90 TRANSFER M$ TO B[I,12]
100 NEXT I
```

The SORT statement sequences array B alphabetically by description (D$) although the sort is actually performed on the numerics in the array, starting at column two where the character code values for D$ are saved.

```
110 SORT B,C,2
```

Once sequenced, the data is transferred from array B back to strings using these lines:

```
120 FOR J=1 TO 20
130 TRANSFER B[J,2] TO D$
140 TRANSFER B[J,12] TO M$
```

To output the 20 part numbers, descriptions, manufacturers and costs sorted by description, D$, add these lines:

```
150 PRINT A[B[J,1],1],D$,M$,A[B[J,1],2]
160 NEXT J
1000 END
```

The PRINT statement in line 150 uses the entry number from array B (indicating original position in array A) as one of the subscripts:

```
A(B(J,1),1)
```

This allows the integrity between the two arrays to remain, since the original row or entry number in array B is used to locate the part number and cost in the same row or entry number in array A.

Sorting array B by column two only, does not provide a thorough sort. The printout indicates that “DESK” and “DESKLAMP” are still not in correct alphabetical order. To execute a complete sequencing by description, change line 110 to:

```
110 SORT B,C,2,3,4,5,6,7
```

Press CONT 110 EXECUTE and when the sequencing is complete, "DESKLAMP" follows "DESK". Each column number specified in the SORT statement above provides for two additional characters in each string to be arranged in sequence.
(See page IV-4 in Appendix IV for the printout.)

To perform a major sort on manufacturers (stored in columns 12 through 14 in array B) and a minor sort by description (stored in columns 2 through 4 in array B), change line 110 to:

```
110 SORT B,C,12,13,14,2,3,4
```

and press CONT 110 EXECUTE. (See page IV-5 in Appendix IV for the printout.)

The sequenced array is arranged alphabetically by manufacturer and, if two manufacturers are identical, it is then arranged alphabetically by description. The sort goes no deeper than necessary to sequence the columns of the array listed in the SORT statement, thus saving time.

Primary and secondary row sorts work in the same way as do primary and secondary column sorts. Column integrity is preserved in row sorts. When string data is transferred to numeric arrays for sorting, column sorts must be used in most cases. (A row sort on a string will rearrange the letters in the string.)

As previously stated, a maximum of six rows or columns (up to 12 characters of a string) can be sequenced. More than one SORT statement is required to sequence strings longer than 12 characters. For example, to sort all 20 characters of the descriptions stored in columns 2 through 11 (two characters are saved per column in an integer array), line 110 is changed to:

```
110 SORT B,C,7,8,9,10,11
```

and line 115 is added:

```
115 SORT B,C,2,3,4,5,6
```

(See page IV-6 in Appendix IV for the printout.)

If only one row or column number is specified in one line of a series of SORT statements, and it is not in the first SORT statement, then the sequence that results may not be correct. For example, the following lines will sequence array B correctly:

```
110 SORT B,C,11
112 SORT B,C,6,7,8,9,10
115 SORT B,C,2,3,4,5
```

However, if line 110 is changed to SORT B, C, 7, 8, 9, 10, 11 and 112 is changed to SORT B, C, 6 the sequencing that results will not be correct. This is because a primary sort uses the shell sorting technique and the sorted order of the previously sorted columns is not maintained. (See Appendix III for a description of a shell sort.)

An array can also be sorted in descending order by any row or column. To do this, the appropriate row or column (or the entire array) is negated before and after the sort operation. In the previous program these lines can be added to negate both arrays:

```
105 MAT A=(-1)*A
106 MAT B=(-1)*B
116 MAT A=(-1)*A
117 MAT B=(-1)*B
```

(See page IV-7 in Appendix IV for the printout.)

In this example the matrix ROM is required to negate the array before and after the sort. To do this without the matrix ROM use a FOR...NEXT loop to negate the entire array.

Another solution that simply reverses the printout instead of inverting the array requires changing line 120 to FOR J=20 TO 1 STEP−1.

In sequencing strings, the calculator sorts the numeric array according to the binary code equivalent of each ASCII character that is input. The ASCII conversion table on the following pages indicates that the binary code values of the keyboard characters (e.g., a blank, an exclamation point, etc.), and the numbers and unshifted letters have lower binary codes than the shifted letters.*

For example, the following program illustrates the precedence of unshifted characters.

```
10 DIM A[5,5],A$[10]
20 FOR I=1 TO 5
30 INPUT A$
40 PRINT A$
50 TRANSFER A$ TO A[I,1]
60 NEXT I
70 SORT A,C,1,2,3,4,5
80 PRINT
90 FOR J=1 TO 5
100 TRANSFER A[J,1] TO A$
110 PRINT A$
120 NEXT J
1000 END
```

When running the program, input the following characters:

LOWERCASE (shift key pressed)
UPPERCASE (shift key not pressed)
%%%%%%%%%%
<<<<<<<<<<
5555555555

The binary codes from the ASCII table cause this output:

```
%%%%%%%%%%
5555555555
<<<<<<<<<<
UPPERCASE
LOWERCASE
```

*Contrary to normal typewriter operation, lower case alphabetics on the 9830A result when the shift key is pressed.

Table 3-1. I/O Characters and Equivalent ASCII (US ASCII) Forms

ASCII Char.	EQUIVALENT FORMS			KEY [1]
	Binary	Octal	Dec [2]	9830A
NULL	00000000	000	0	——
SOH	00000001	001	1	——
STX	00000010	002	2	——
ETX	00000011	003	3	——
EOT	00000100	004	4	——
ENQ	00000101	005	5	——
ACK	00000110	006	6	——
BELL	00000111	007	7	——
BS	00001000	010	8	——
HTAB	00001001	011	9	——
LF	00001010	012	10	——
VTAB	00001011	013	11	——
FF	00001100	014	12	——
CR	00001101	015	13	——
SO	00001110	016	14	——
SI	00001111	017	15	——
DLE	00010000	020	16	——
DC_1	00010001	021	17	——
DC_2	00010010	022	18	——
DC_3	00010011	023	19	——
DC_4	00010100	024	20	——
NAK	00010101	025	21	——
SYNC	00010110	026	22	——
ETB	00010111	027	23	——
CAN	00011000	030	24	——
EM	00011001	031	25	——
SUB	00011010	032	26	——
ESC	00011011	033	27	——
FS	00011100	034	28	——
GS	00011101	035	29	——
RS	00011110	036	30	——
US	00011111	037	31	——

ASCII Char.	EQUIVALENT FORMS			KEY [1]
	Binary	Octal	Dec [2]	9830A
space	00100000	040	32	Space Bar
!	00100001	041	33	SHIFT + (! 1)
"	00100010	042	34	——
#	00100011	043	35	SHIFT + (# 3)
$	00100100	044	36	SHIFT + ($ 4)
%	00100101	045	37	SHIFT + (% 5)
&	00100110	046	38	SHIFT + (& 6)
'	00100111	047	39	SHIFT + (' 7)
(	00101000	050	40	(
)	00101001	051	41	)
*	00101010	052	42	*
+	00101011	053	43	+
,	00101100	054	44	,
−	00101101	055	45	−
.	00101110	056	46	.
/ '	00101111	057	47	/
Ø	00110000	060	48	0
1	00110001	061	49	1
2	00110010	062	50	2
3	00110011	063	51	3
4	00110100	064	52	4
5	00110101	065	53	5
6	00110110	066	54	6
7	00110111	067	55	7
8	00111000	070	56	8
9	00111001	071	57	9
:	00111010	072	58	(* :)
;	00111011	073	59	(+ ;)
<	00111100	074	60	SHIFT + (< ,)
=	00111101	075	61	=
>	00111110	076	62	SHIFT + (> .)
?	00111111	077	63	SHIFT + (? /)

ASCII Char.	EQUIVALENT FORMS Binary	Octal	Dec [2]	KEY [1] 9830A
@	01000000	100	64	SHIFT RESULT
A	01000001	101	65	A
B	01000010	102	66	B
C	01000011	103	67	C
D	01000100	104	68	D
E	01000101	105	69	E
F	01000110	106	70	F
G	01000111	107	71	G
H	01001000	110	72	H
I	01001001	111	73	I
J	01001010	112	74	J
K	01001011	113	75	K
L	01001100	114	76	L
M	01001101	115	77	M
N	01001110	116	78	N
O	01001111	117	79	O
P	01010000	120	80	P
Q	01010001	121	81	Q
R	01010010	122	82	R
S	01010011	123	83	S
T	01010100	124	84	T
U	01010101	125	85	U
V	01010110	126	86	V
W	01010111	127	87	W
X	01011000	130	88	X
Y	01011001	131	89	Y
Z	01011010	132	90	Z
[	01011011	133	91	—
\	01011100	134	92	—
]	01011101	135	93	—
^	01011110	136	94	↑
_	01011111	137	95	—

ASCII Char.	EQUIVALENT FORMS Binary	Octal	Dec [2]	KEY [1] 9830A
`	01100000	140	96	—
a	01100001	141	97	SHIFT A
b	01100010	142	98	SHIFT B
c	01100011	143	99	SHIFT C
d	01100100	144	100	SHIFT D
e	01100101	145	101	SHIFT E
f	01100110	146	102	SHIFT F
g	01100111	147	103	SHIFT G
h	01101000	150	104	SHIFT H
i	01101001	151	105	SHIFT I
j	01101010	152	106	SHIFT J
k	01101011	153	107	SHIFT K
l	01101100	154	108	SHIFT L
m	01101101	155	109	SHIFT M
n	01101110	156	110	SHIFT N
o	01101111	157	111	SHIFT O
p	01110000	160	112	SHIFT P
q	01110001	161	113	SHIFT Q
r	01110010	162	114	SHIFT R
s	01110011	163	115	SHIFT S
t	01110100	164	116	SHIFT T
u	01110101	165	117	SHIFT U
v	01110110	166	118	SHIFT V
w	01110111	167	119	SHIFT W
x	01111000	170	120	SHIFT X
y	01111001	171	121	SHIFT Y
z	01111010	172	122	SHIFT Z
{	01111011	173	123	—
¦	01111100	174	124	—
}	01111101	175	125	—
~	01111110	176	126	—
DEL	01111111	177	127	—

11202A-C-50673 T-0001

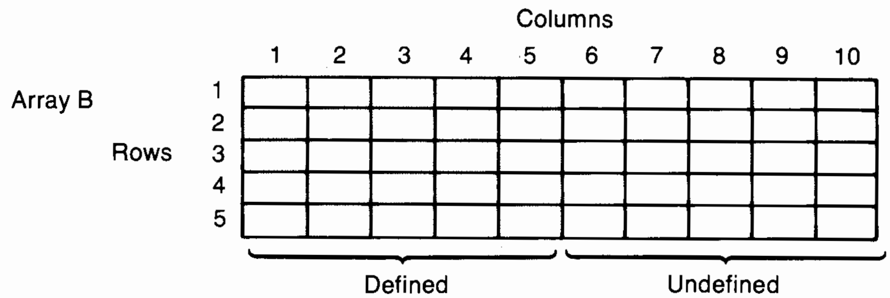
[1] When SHIFT is shown, the SHIFT key is held down while the following key is pressed. Where a key is not shown in the 9830A column, use a WRITE (FORMAT B) statement to output the decimal-equivalent number.

[2] Decimal numbers are used with 9830A WRITE (FORMAT B) statements.

PROGRAMMING HINTS

When partial data (or none at all) is transferred to one of the columns (or rows) of an array and a sort is attempted, ERROR 40 results. This is because the SORT statement cannot sequence undefined columns (or rows) in the array. Undefined columns in an array remain when the column dimension of the array is greater than is needed for the amount of data to be transferred to each row of the array.

For example, in the program on page 3-7, if the array is dimensioned as AI(5,10) and A$ remains the same (ten characters) then half of array A is undefined and ERROR 40 will result. (Remember two characters per numeric array element are saved in each column.)



Undefined rows of an array are created when the number of entries or inputs is not identical to the number of rows dimensioned in the DIM or COM statement.

One way to eliminate ERROR 40, of course, is to make the number of columns in the integer-precision array half the dimensioned size of the strings to be input and the number of rows exactly the expected number of inputs.

Other solutions are possible when the matrix ROM is installed. By adding line 15 MAT A=ZER, the undefined areas of array A are defined (filled with zeroes) and ERROR 40 is not displayed.

Without the matrix ROM, these lines added to the previous program will accomplish the same:

```
11 FOR X=1 TO 5
12 FOR Y=1 TO 10
13 A[X,Y]=0
14 NEXT Y
15 NEXT X
```

Another solution using the matrix ROM, is to redimension the array before the strings are sorted using a REDIM statement. This makes the larger array size in the DIM statement available when needed.

When a SORT statement is executed on more rows or columns of the array than have been dimensioned in the DIM statement, ERROR 42 results. To avoid this, the sequencing numbers that follow SORT should not exceed the number of rows or columns in the array. For example, in the program on page 3-7, if line 70 were SORT A, C, X, then X must have a value between 1 and 5.

ERROR 74 may occur when data is stored on tape and the length of the strings stored is unknown. By defining the length of the string in the TRANSFER statement, no more than the allotted number of characters may be stored in a row of the array because all extra characters are truncated. Line 50 in the previous program can be modified to illustrate this:

```
50 TRANSFER A$[1,10] TO A[I,1]
```

ERROR 74 is eliminated.

Another reason for limiting the length of a string is to avoid the overflow of the string into the next row of the array. This is accomplished by modifying the input statement. For example, change line 30 of the previous program to read:

```
30 INPUT A$[1,10]
```

By specifying the length of the input string, no more than ten characters of the string (no matter how many characters are input) will be transferred into the array and ERROR 74 is avoided.

In general, a primary sort is faster than a secondary sort and a row sort is faster than a column sort. To sequence a large amount of data, a single SORT statement is slower than a primary row or column sort followed by the sorting of all secondary rows or columns. For example, if 256 rows of data were input for the previous program after the array had been correctly dimensioned — AI(256, 5), then this sort would be the fastest:

```
SORT A,C,1
SORT A,C,1,2,3
```

The sort shown above takes about five seconds while a single SORT statement, SORT A, C, 1, 2, 3, takes about a minute to complete. This is because of the sorting techniques used. Primary sequencing uses a shell sort; secondary sequencing uses a bubble sort. (See Appendix III for an explanation of shell and bubble sorts.)

APPENDIX EXAMPLE

Appendix II contains a program to sort arrays larger than 256 items.

NOTES

Chapter 4

SEARCH STATEMENT

DESCRIPTION

The SEARCH statement provides a fast and easy way to locate a specific element in any row or column of a numeric array. Using SEARCH, access to data (stored as numeric arrays) in long indexes, large tables or customer data files becomes easier. In addition to numerics, strings can be found using the SEARCH statement. The string variables ROM and the TRANSFER statement are required to search strings.

SYNTAX

To search a row:

SEARCH array name, R, row number, match variable, return variable

To search a column:

SEARCH array name, C, column number, match variable, return variable

The array name is a letter from A to Z representing the array to be searched, R or C specifies either a row or column search and row or column number indicates the number of the row or column to be searched. Match variable is the variable being searched for; the match variable is automatically converted to the precision of the array being searched: integer, split or full-precision. Return variable is a variable which equals the position or location in the column of the match variable if a row search is executed, or the position or location in the row of the match variable if a column search is executed. The return variable must be a simple variable, not an array variable. Singly dimensioned arrays are column arrays on the 9830A. Therefore a column search would be more useful on a singly dimensioned array.

SEARCHING NUMERIC ARRAYS

The following programs use the directory created in Chapter 2 (page 3-3). The directory is a full-precision array containing part numbers in column one, costs in column two and file numbers in column three. To get a printout of the directory, run the program below.

```
10 COM P,D$[20],M$[12],C
20 DIM A[20,3]
30 LOAD  DATA 0,A
40 FOR X=1 TO 20
50 PRINT A[X,1],A[X,2],A[X,3]
60 NEXT X
1000 END
```

Example 1

The following program loads the directory in tape file 0 and then searches the directory (array A) for the position in column one of part number 8586.

```
10 COM P,D$[20],M$[12],C
20 DIM A[20,3]
30 LOAD  DATA 0,A
40 SEARCH A,C,1,8586,X
50 PRINT X
1000 END
```

When the program is run, 16 is printed and displayed indicating that part number 8586 is in the sixteenth row of the directory.

The previous program found the position (*row number*) of a part number in the directory. By changing line 40 as shown below, the position (*column number*) of that part number in the sixteenth row can be found.

```
40 SEARCH A,R,16,8586,X
```

When the program is run, 1 is printed indicating that part number 8586 is found in column one of the sixteenth row of the directory. If a match number cannot be found in a row (or column) of an array, zero is returned.

Example 2

When a number occurs more than once in a numeric array, such as the $5.00 cost in the directory, there are two ways to locate the second occurrence of the number.

One method finds the first occurrence of the number, makes it negative and then repeats the search to locate the second occurrence of the number. This method can be used if negatives are not already found in the array being searched. To illustrate this, first modify line 40 to read:

```
40 SEARCH A,C,2,5,X
```

The SEARCH statement above causes column two (costs) of array A to be searched until $5 is located. When the first $5.00 price is located, 1 is printed and displayed indicating that the first item in the directory costs $5.00. The $5.00 in row one must be changed to −$5.00 so the second occurrence of $5.00 can be found. The $5.00 (located in row one, column two of array A) is negated by executing the statement below:

```
A(1,2)=-A(1,2)
```

Then press CONT 40 EXECUTE. The SEARCH starts over again from the beginning position of the column (or row) being searched and finds the only occurrence of $5.00 that is left. Always remember to make the negated value positive again when all searches are complete.

If the array being searched already contains negatives, a zero can be substituted for the first occurrence of a number so the second occurrence can be found. Again, be sure to change the numbers in the array back to their original values when the search is complete.

Another way to find the second occurrence of a number in an array is to first *sort* the array by the column to be searched. Then, if there is a second occurrence of a number, it follows the first occurrence. The first occurrence is found by executing a search on the appropriate column (or row). To illustrate this, add line 35 to the previous program:

```
35 SORT A,C,2
```

The sort is executed first, placing the two $5.00 costs in the first two rows of array A. Then a search on column two for the $5.00 cost shows that it occurs in row one of column one. Therefore the second occurrence is in row two of column one.

PROGRAMMING HINTS

A key column (or entry number column) can be used in an array to number the rows or columns when in their original position. This column then can be sorted to get the array back to its original order. To get the directory back to original order by file number, change line 35 to SORT A, C ,3 and change lines 40 and 50 below.

```
10 COM P,D$[20],M$[12],C
20 DIM A[20,3]
30 LOAD  DATA 0,A
35 SORT A,C,3
40 FOR X=1 TO 20
50 PRINT A[X,1],A[X,2],A[X,3]
60 NEXT X
1000 END
```

APPENDIX EXAMPLE

Appendix II contains a program which searches an array to locate strings.

NOTES

Chapter 5

STRING STATEMENT

DESCRIPTION

The STRING statement places a number in a predefined and formatted string, so that money values can be output in any currency format. To use the STRING statement, the string variables ROM must be installed in your calculator.

The STRING statement is specifically included in the APII ROM for writing monetary values where dollar signs, decimal points, commas and other currency indicators are to be inserted in specific positions of any currency amount.

First, a string, called an "image string", is defined by the user in the appropriate format with blanks where the monetary values are to be inserted. When a given number is input, the STRING statement causes the blank characters of the predefined image string to be filled with the specific numeric value input. Finally, the initial characters (such as $, £ , etc.) are abutted to the number and the remainder of the string (the left-most portion) is filled with blanks. This enables the user to write checks in any currency format. Since the 9830A has 12 digit accuracy, a maximum of 12 digits are retained in the currency representation when the STRING statement is used.

> **NOTE**
> Money values are framed by all characters which are specified to precede or follow, so that no blank spaces are available for altering the monetary value of a check.

SYNTAX

To convert a number to a formatted string:

STRING numeric, string name, number of decimal places

The numeric is the currency amount to be formatted, string name references a string that contains an image of the desired format and number of decimal places is the number of places in the image string that follow the assumed decimal point. (The decimal point may be represented by some character other than the period, such as the comma in German marks.)

> **NOTE**
> The number of decimal places specified in the STRING statement should *always* be identical to the number of decimal places shown in the image string.

EXAMPLE 1

The costs used in the previous program (see page 3-5 in Chapter 3) can be formatted in dollars and cents using the following program. (NOTE: Δ= a space)

```
10 COM P,D$[20],M$[12],C
20 DIM A[20,2],BI[20,17],I$[10]
30 FOR I=1 TO 20
40 LOAD  DATA I
50 I$="$ΔΔΔ.ΔΔ"
60 STRING C,I$,2
70 PRINT P,D$;TAB40;M$;TAB55;I$
80 NEXT I
1000 END
```

The printout contains the 20 dollar amounts each with dollar signs and decimal points and no leading blanks which could permit altering of the dollar amounts.

```
3948    DESK                    MATHERS      $149.95
2548    BOOKCASE                SMYTHE       $199.99
8655    BOOKCASE                SMYTHE        $50.00
3651    CHAIR WITH ARMS         JONES         $35.50
7711    CHAIR WITH ARMS         JONES         $74.99
9086    CABINET                 WINDSOR       $89.99
3667    BOOKCASE                SMYTHE       $120.50
5218    CHAIR WITHOUT ARMS      JONES         $19.95
4520    CHAIR WITH ARMS         HALSTEAD     $249.99
8531    DESKLAMP                MATHERS      $249.99
7051    WASTEBASKET             JONES          $5.00
4377    DESK                    MATHERS       $50.00
9886    CABINET                 WINDSOR      $120.50
1478    CABINET                 JONES        $599.99
9932    CABINET                 WINDSOR      $249.99
7128    CABINET                 WINDSOR       $89.99
4439    DESK                    HALSTEAD     $499.95
1086    ASHTRAY                 WINDSOR        $5.00
8586    ASHTRAY                 JONES         $25.00
3989    WASTEBASKET             HALSTEAD      $35.00
```

EXAMPLE 2

The following example illustrates the formatting of deutsche marks currency.

```
10 DIM A$[80],B$[80]
20 B$="*ΔΔΔ,ΔDM"
30 A$=B$
40 INPUT N
50 STRING N,A$,1
60 PRINT A$
70 GOTO 30
1000 END
```

The printout looks like this: `*123,4DM`

Here line 30 is used to recreate the image string each time the string is to be filled with a different numeric input.

When the program is run, a number is input causing the blank characters in B$ (line 20) to be filled with the specific currency amount input (N in line 40). Then the surrounding characters (⋆ and DM in line 20) are abutted to the currency amount and all blank spaces are moved to the left-most character positions.

EXAMPLE 3

The program below enables extra characters to be printed or displayed using the STRING statement. In the previous program, change line 20 to:

```
20 B$="*CASH ON HAND $ΔΔΔ,ΔΔΔ,ΔΔΔ.ΔΔ"
```

and line 50 to:

```
50 STRING N,A$[15],2
```

When the program is run, the number input is inserted in the image string in line 20 and spaces are moved to the left-most character positions. By specifying the substring A$(15) in line 50, the dollar sign and the correctly formatted currency amount follow the substring (⋆ CASH ON HAND) beginning at character position 15.

The printout looks like this: `*CASH ON HAND     $123,455.67`

EXAMPLE 4

Some European currencies require blanks in their monetary formats to separate hundreds, thousands, millions, etc. The program below illustrates how a character ("&" in line 20) is inserted to represent the blanks and is then found and removed before the monetary value is printed.

```
10 DIM A$[80],B$[80]
20 B$="*ΔΔΔ&ΔΔΔ&ΔΔΔ&ΔΔΔ.ΔΔ*"
30 A$=B$
40 INPUT N
50 STRING N,A$,2
60 I=POS(A$,"&")
70 IF I=0 THEN 100
80 A$[I,I]="Δ"
90 GOTO 60
100 PRINT A$
110 GOTO 30
1000 END
```

The printout looks like this: `*12 334 567.99*`

Notice that the asterisk is placed before and after the currency amount so that alteration of the amount is not possible.

PROGRAMMING HINTS

When the number of decimal places in the image string does not match the number of decimal places specified in the STRING statement, an incorrect result is obtained. To avoid this, always be sure the two are identical.

The STRING statement truncates the currency amount if the number of decimal places in the STRING statement is less than the number of decimal places of the input. To round the currency amount instead of truncating it, insert a rounding statement in the program. In Example 2, the rounding statement is added as line 45.

```
45 N=N+.05
```

To round a number with one decimal place, the appropriate decimal to add is .05; to round a number with two decimal places, add .005, and so on.

An error will occur (ERROR 72) if the number of digits preceding the decimal point of the numeric input is larger than the allotted space for it in the image string.

Chapter 6

SERROR STATEMENT

DESCRIPTION

The SERROR or set error statement provides the calculator with automatic recovery from most calculator errors. Error recovery removes the need for user intervention when an error occurs. The SERROR statement performs the same function as the continue command when the calculator stops because of an error condition.

Since error recovery is not logical for every possible error condition, error recovery routines should be selectively written for only certain specific anticipated errors. Error recovery is used for errors that the programmer will have no control over once the person who uses the program takes over. These errors are usually cassette errors (ERROR 58, 59, 60, 61, 62), I/O failures (ERROR 83) or mass memory errors (ERROR 99).

Error recovery routines should be incorporated in a program only when the program is near completion. Set error should not be used to debug a program because it makes detection of logic errors more difficult.

The SERROR statement is usually added to a program in the areas that could potentially cause errors, such as LOAD DATA, ENTER, READ# or PRINT#. When error recovery routines are finally added to the program, they should begin with WAIT 500 to prevent keyboard lockout due to infinite loops in the error recovery routine. Once the program is completely written and debugged, the WAIT statements can be removed.

Memory configuration errors 1, 2 and check sum error 59 when attempting to LOAD or LINK a program are still non-recoverable. (ERROR 59, cassette check sum error, on a data file is recoverable when an error recovery routine is used. Syntax errors are also non-recoverable because error recovery is allowed only in the program mode.

A program cannot continue execution in subroutines, defined functions or FOR...NEXT loops unless the error is between ERROR 100 and ERROR 107. To discontinue the SERROR statement, STOP or END can be executed, the STOP key can be pressed or SERROR can be redefined.

> **NOTE**
>
> Both the DATA COMM 2 and the APII ROMS contain the SERROR statement. A program written using set error with *only* DATA COMM 2 installed will not run when *only* APII is installed, or vice versa. (ERROR 1 results.) If DATA COMM 2 is exchanged for APII, or vice versa, all lines containing the SERROR statement must be re-entered.

SYNTAX

To use error recovery:

SERROR variable, line number

The variable is the location where the error number will be stored and line number is the line where program execution is to continue after the normal error message is displayed. The SERROR statement takes four words of memory when the APII ROM is installed, so that the variable and line number information can be stored.

When an error occurs, a message is displayed in the form, "ERROR XXX IN LINE NNN". If a SERROR statement has been executed and an error occurs, the line number which was assigned for error recovery is accessed. The program continues execution at that line number and stores the error number using the specified variable.

Since the type of error is indicated by number, many possible error conditions can be checked at the point where execution continues. Each type of error can then be handled differently under program control.

EXAMPLE 1

The following program illustrates how data files may be checked for ERROR 59. If ERROR 59 occurs, line 110 is accessed and the error number is stored using the variable E.

```
10 COM P,D$[20],M$[12],C
20 SERROR E,110
30 I=0
40 F=1
50 LOAD  DATA F
60 PRINT D$,P
70 F=F+1
80 IF F <= 20 THEN 50
90 DISP "DONE"
100 END
110 WAIT 1000
120 IF E#59 THEN 190
130 I=I+1
140 IF I<3 THEN 50
150 PRINT "**UNABLE TO READ FILE";F;"SKIPPING TO FILE";F+1
160 I=0
170 F=F+1
180 GOTO 50
190 PRINT "**ERROR CONDITION NOT PROGRAMMED; ERROR=";E
200 END
```

Lines 130 and 140 allow the file to be reread three times before the error recovery routine is accessed. Line 110 contains a WAIT statement so that if an error occurs anywhere else in the program, the STOP key, when pressed, will stop the program from executing an infinite loop. For example, if line 120 were mistakenly typed IF A # 59 THEN 190, and A did not contain the error code, line 110 would be repetitively accessed.
(See page IV-16 in Appendix IV for the printout.)

PROGRAMMING HINTS

The SERROR statement is executed whenever "ER" is in the first two positions of the 9830A display *and* the calculator comes to a STOP or END. Therefore, if any word with "ER" in the first two characters of the display is printed or displayed and STOP or END is executed, then the SERROR statement is executed. To avoid this, insert blanks (or other characters) in columns one and two of any output to be displayed.

Line 190 of the previous program contains asterisks as the first two characters of the PRINT statement so that the error recovery routine is not executed when the calculator stops with the "ER" message in the display.

Chapter 7

FLAG FUNCTIONS

DESCRIPTION

The FLAG functions allow sixteen separate flags, numbered 0 through 15, to be set, cleared and tested. Flags can be used for testing or branching in a program. Flags use less memory and execute faster than testing variables as flags. The FLAG function,used to test a flag, returns 0 when a flag is cleared and 1 when a flag is set.

All flags are set by the user. Any command that initializes the memory also clears all flags. The LOAD and LINK commands do not affect flags. The 16 flag functions take one word of calculator memory when the APII ROM is installed in the calculator since each flag requires a single bit.

SYNTAX

To set a flag:

SFLAG flag number

To clear a flag:

CFLAG flag number

To test a flag:

FLAG flag number

The flag number is a number between 0 and 15. When testing whether a flag is set or cleared, either 1 is returned if the flag is set or 0 is returned if the flag is cleared. Testing a flag does not change the condition of the flag.

> **NOTE**
> All flags remain set or cleared from one program to the next, therefore it is necessary to execute a CFLAG statement to assure that a previously used flag is not set when a new program is run. The RUN command does not clear flags.

EXAMPLE 1

These lines are taken from the Flag Example in Appendix II. They illustrate the setting and clearing of flags and the use of IF NOT FLAG. Notice that the flags are cleared in lines 30 and 40 each time the program is run. Otherwise the flags would remain set continously after their first use.

Lines 80 and 120 set the flags. Flag 1 is set in line 80 if the user wants the VALUE OF EACH ITEM printed. Flag 2 is set in line 120 if the user wants the SUBTOTALS BY MANUFACTURER printed.

The second half of the program generates the printout according to the flags that were set or not set in the beginning of the program. Lines 240 and 330 enable the calculator to jump to the appropriate section of the program where print instructions for the previously set conditions (flags) appear.

```
10 COM P,D$[20],M$[12],C
20 DIM A[20,7],A$[10],X$[12]
30 CFLAG 1
40 CFLAG 2
50 DISP "PRINT VALUE OF EACH ITEM (Y/N)";
60 INPUT A$
70 IF A$#"Y" THEN 90
80 SFLAG 1
90 DISP "PRINT SUBTOTALS BY MFG (Y/N)";
100 INPUT A$
110 IF A$#"Y" THEN 130
120 SFLAG 2
  •
  •
  •
  •
240 IF  NOT FLAG2 THEN 320
  •
  •
  •
330 IF  NOT FLAG1 THEN 350
  •
  •
  •
410 END
```

For example, when line 240 is accessed, if flag 2 was set, then the subtotals are printed. If flag 2 was not set, then line 320 is accessed and the routine to print subtotals is skipped. When line 330 is accessed, if flag 1 was set, then the item values are printed. If flag 1 was not set, then line 350 is accessed and the routine to print item values is skipped.

PROGRAMMING HINTS

When debugging a program, a flag can be set to indicate whether a line has been accessed or not. For example, in the lines below, if line 60 is accessed, then 1 is printed when line 1000 is accessed. If 0 is printed, line 60 was never accessed.

```
60 TRANSFER A$ TO A[I,1]
70 SFLAG 16
•
•
•
1000 PRINT  "FLAG16=";FLAG16
```

This technique eliminates the need for lengthy TRACE routines in some cases.

APPENDIX EXAMPLE

A complete listing of the program in Example 1 is found in Appendix II.

Chapter 8

BEEP STATEMENT

DESCRIPTION

The BEEP statement is used to create an audible signal which can be used in a number of ways. BEEP can signal that a particular calculation has been completed or that a certain program sequence has been accessed. Beep can also be used to audibly indicate that the calculator is ready for data input so that the user does not have to remain at the calculator. BEEP can be used to make a constant beeping noise to signal program completion or other calculator conditions. The BEEP statement can be executed in either the calculator (non-program) mode or the program mode.

> **NOTE**
> Both the API and the APII ROMs contain the BEEP statement. A program written using beep with *only* API installed will not run when *only* APII is installed, or vice versa. (ERROR 1 results.) If API is exchanged for APII, or vice versa, all lines containing BEEP must be re-entered.

SYNTAX

To produce the beeping sound:

BEEP

EXAMPLE 1

The following example shows how beep audibly signals the user when the calculator is ready for data input.

```
10 DISP "INPUT DATA";
20 BEEP
30 INPUT N
1000 END
```

EXAMPLE 2

In the following example, beep indicates to the user that the final value of X has been calculated after the FOR...NEXT loop in lines 20 through 40.

```
10 X=2
20 FOR I=1 TO 100
30 X=X*2
40 NEXT I
50 BEEP
1000 END
```

EXAMPLE 3

When the example below is completed, beeping continues until the STOP key is pressed. The loop, lines 997 through 999 contains a WAIT statement.

```
10 X=3
•
•
•
•
•
997 BEEP
998 WAIT 120
999 GOTO 997
1000 END
```

> **NOTE**
> WAIT 120 (a delay of 120 milliseconds between beeps) allows the timing cycle of one beep to be completed before the beep is repeated.

APPENDIX I

The following program creates 20 data files which store part number, description, manufacturer and item cost of 20 pieces of office furniture. (The string variables ROM must be installed in your calculator to use this program.)

Before keying in this program, mark a blank tape with one 100 word file (file zero) and twenty 50 word files. (For more information on the use of the MARK command, refer to the 9830A Operating and Programming Manual.) Your own data may be used if the proper DIM and COM specifications are given, however the data provided below is used in most of the examples in this manual. (Be sure to input all data in unshifted (upper case) mode. See page 3-7 for an explanation of this.)

```
 10 COM P,D$[20],M$[12],C
☆20 FOR I=1 TO 20
 30 DISP "PART NO.";
 40 INPUT P
 50 DISP "DESCRIPTION";
 60 INPUT D$
 70 DISP "MANUFACTURER";
 80 INPUT M$
 90 DISP "COST";
 100 INPUT C
☆110 STORE  DATA I
 120 NEXT I
 130 END
```

DATA FILES

ITEM	PART NO.	DESCRIPTION	MANUFACTURER	COST
1	3948	desk	Mathers	149.95
2	2548	bookcase	Smythe	199.99
3	8655	bookcase	Smythe	50.00
4	3051	chair with arms	Jones	35.50
5	7711	chair with arms	Jones	74.99
6	9086	cabinet	Windsor	89.99
7	3667	bookcase	Smythe	120.50
8	5218	chair without arms	Jones	19.95
9	4520	chair with arms	Halstead	249.99
10	8531	desklamp	Mathers	249.99
11	7051	wastebasket	Jones	5.00
12	4377	desk	Mathers	50.00
13	9886	cabinet	Windsor	120.50
14	1478	cabinet	Jones	599.99
15	9932	cabinet	Windsor	249.99
16	7128	cabinet	Windsor	89.99
17	4439	desk	Halstead	499.95
18	1086	ashtray	Windsor	5.00
19	8586	ashtray	Jones	25.00
20	3989	wastebasket	Halstead	35.00

☆Add line 15 (FILES INVEN) and change the other starred line for use with the mass memory. See page 1-2 for instructions.

APPENDIX II

TRANSFER EXAMPLE

The program below permits storage of strings longer than 255 characters using more than one INPUT statement. Each input of 80 characters is stored in the array starting where the last substring stopped. Twenty strings, each 300 characters long, are input and stored in the integer-precision array, AI(20, 150).

```
10 DIM AI[20,150],A$[255],B$[80],C$[45]
20 FOR I=1 TO 20
30 DISP "ENTER 80 CHARACTER STRING;0=END";
40 INPUT A$[1,80]
50 IF A$[1,1]="0" THEN 170
60 DISP "ENTER 80 MORE CHARACTERS";
70 INPUT A$[81,160]
80 DISP "ENTER 80 MORE CHARACTERS";
90 INPUT A$[161,240]
100 DISP "ENTER 15 MORE CHARACTERS";
110 INPUT A$[241,255]
120 DISP "ENTER 45 MORE CHARACTERS";
130 INPUT B$[1,45]
140 TRANSFER A$ TO A[I,1]
150 TRANSFER B$ TO A[I,128]
160 NEXT I
```

To output these strings, the strings are transferred from the numeric array back to strings using the program lines below.

```
170 FOR J=1 TO I-1
180 TRANSFER A[J,1] TO A$
190 TRANSFER A[J,128] TO B$
200 FORMAT B
210 WRITE (15,200)A$[1,80]
220 WRITE (15,200)A$[81,160]
230 WRITE (15,200)A$[161,240]
240 WRITE (15,200)A$[241,255];
250 WRITE (15,200)B$
260 NEXT J
270 END
```

SORT EXAMPLE

The SORT statement is used to order sequences that contain more than 256 data items. The following program sorts two integer-precision arrays (each with 200 items) separately by description and then outputs the arrays in "merged" order. The merge routine retains the sorted order of each array and prints out a single array (or list of 400 data files) in correct sorted order.

Line 50 transfers D$ from the first 200 files to array A for sorting (line 130), while line 100 transfers D$ from the next 200 files for sorting (line 140). The strings are compared in lines 160 to 180 and the correctly sorted data files are printed. (This example requires that a larger space (OPEN "INVEN" 400) be reserved on the mass memory.)

```
   10 COM P,D$[20],M$[12],C
   20 DIM AI[200,11],BI[200,11],A$[20],B$[20]
 ☆ 30 FOR I=1 TO 200
 ☆ 40 LOAD  DATA I
   50 TRANSFER D$ TO A[I,1]
   60 A[I,11]=I
   70 NEXT I
   80 FOR J=1 TO 200
 ☆ 90 LOAD  DATA J+200
  100 TRANSFER D$ TO B[J-200,1]
  110 B[J-200,11]=J
  120 NEXT J
  130 SORT A,C,1,2,3,4,5
  140 SORT B,C,1,2,3,4,5
  150 A0=B0=1
  160 TRANSFER A[A0,1] TO A$
  170 TRANSFER B[B0,1] TO B$
  180 IF A$>B$ THEN 280
☆ 190 LOAD  DATA A[A0,11]
  200 PRINT M$,D$,TAB50;P;TAB60;C
  210 A0=A0+1
  220 IF A0 <= 200 THEN 260
  230 IF B0>200 THEN 370
  240 A$="↑↑↑↑↑"
  250 GOTO 180
  260 TRANSFER A[A0,1] TO A$
  270 GOTO 180
☆ 280 LOAD  DATA B[B0,11]
  290 PRINT M$,D$,TAB50;P;TAB60;C
  300 B0=B0+1
  310 IF B0 <= 200 THEN 350
  320 IF A0>200 THEN 370
  330 B$="↑↑↑↑↑"
  340 GOTO 180
  350 TRANSFER B[B0,1] TO B$
  360 GOTO 180
  370 END
```

☆Add line 25 and change all other starred lines for use with the mass memory. See page 1-2 for instructions.

SEARCH EXAMPLE

The SEARCH statement is designed primarily to locate numbers in a numeric array (i.e., file numbers or customer numbers). SEARCH can also be used to locate strings stored in integer-precision numeric arrays.

The following program is based on the example found in Appendix I and Chapter 2. Lines 30 through 100 load the 20 data files stored on tape into the calculator's memory. Array A, a full precision array, contains part numbers and prices for the 20 items; array B, an integer array, contains descriptions and manufacturers of the items. Line 70 sets up an entry number which is the original row (and tape file) number and which maintains correspondence between arrays A and B.

The search routine is found in lines 110 through 320 with the output in lines 210 through 240. The item to be searched for is input as X$ and then transferred to array C so numeric values of the characters can be compared. The search is executed and the first match of the leading two characters, C(1,1) is returned as Z. This item from array B is then transferred back to a string (Y$) and is compared with the input (X$). If they are equal, then the part number, description, manufacturer and price of the item being searched for is printed. This first occurrence is then negated and all other occurrences of the item are located in the same way. Then "SEARCH COMPLETE" is printed.

If X$ does not equal Y$, then that part of array B is negated in line 190 and the search continues. When no further searches are required, a search variable of zero is input and "END OF SEARCH" is printed. (See page IV-11 and IV-12 in Appendix IV for the printout.)

```
  10 COM P,D$[20];M$[12],C
☆ 20 DIM A[20,2],BI[20,17],CI[1,10],X$[20],Y$[20]
  30 FOR I=1 TO 20
☆ 40 LOAD  DATA I
  50 TRANSFER D$ TO B[I,1]
  60 TRANSFER M$ TO B[I,11]
  70 B[I,17]=I
  80 A[I,1]=P
  90 A[I,2]=C
  100 NEXT I
```

☆Add line 25 and change all other starred lines for use with the mass memory. See page 1-2 for instructions.

```
110 DISP "SEARCH FOR; TO END PRESS 0";
120 INPUT X$[1,20]
130 IF X$[1,1]="0" THEN 320
140 TRANSFER X$ TO C[1,1]
150 SEARCH B,C,1,C[1,1],Z
160 IF Z=0 THEN 260
170 TRANSFER B[Z,1] TO Y$
180 IF Y$=X$ THEN 210
190 B[Z,1]=-B[Z,1]
200 GOTO 150
210 TRANSFER B[Z,1] TO D$
220 TRANSFER B[Z,11] TO M$
230 PRINT A[B[Z,17],1],D$,M$,A[B[Z,17],2]
240 B[Z,1]=-B[Z,1]
250 GOTO 150
260 FOR I=1 TO 20
270 B[I,1]=ABSB[I,1]
280 NEXT I
290 PRINT "SEARCH COMPLETE"
300 PRINT
310 GOTO 110
320 PRINT "END OF SEARCH"
330 END
```

FLAG EXAMPLE

The following program uses flags to output data according to the user's specifications. (Data used here is found in Appendix I.) By setting or not setting flag 1 and flag 2 (in lines 80 to 120), the user can choose which of four possible outputs he wants.
(See page IV-13 through IV-15 in Appendix IV for the printouts.)

Four possible outputs:

1. If both flags are set, ITEM VALUES and INVENTORY SUBTOTALS are printed.
2. If only flag 1 is set, ITEM VALUES are printed.
3. If only flag 2 is set, INVENTORY SUBTOTALS are printed.
4. If neither flag is set, then only TOTAL INVENTORY VALUE is printed.

TOTAL INVENTORY VALUE is printed with all four outputs.

```
  10 COM P,D$[20],M$[12],C
☆ 20 DIM A[20,7],A$[10],X$[12]
  30 CFLAG 1
  40 CFLAG 2
  50 DISP "PRINT VALUE OF EACH ITEM (Y/N)";
  60 INPUT A$
  70 IF A$#"Y" THEN 90
  80 SFLAG 1
  90 DISP "PRINT SUBTOTALS BY MFG (Y/N)";
  100 INPUT A$
  110 IF A$#"Y" THEN 130
  120 SFLAG 2
  130 FOR I=1 TO 20
☆ 140 LOAD  DATA I
  150 TRANSFER M$ TO A[I,1]
  160 A[I,7]=I
  170 NEXT I
  180 SORT A,C,1,2,3,4,5,6
☆ 190 LOAD  DATA A[1,7]
  200 X$=M$
  210 C0=C1=0
  220 FOR I=1 TO 20
☆ 230 LOAD  DATA A[I,7]
  240 IF  NOT FLAG2 THEN 320
  250 IF M$=X$ THEN 310
  260 PRINT TAB38;X$;TAB50;"***SUBTOTAL***";C1
  270 PRINT
  280 C1=C
  290 X$=M$
  300 GOTO 320
  310 C1=C1+C
  320 C0=C0+C
  330 IF  NOT FLAG1 THEN 350
  340 PRINT TAB5;P;TAB15;D$;TAB35;M$;TAB50;C
  350 NEXT I
  360 IF  NOT FLAG2 THEN 390
  370 PRINT TAB38;X$;TAB50;"***SUBTOTAL***";C1
  380 PRINT
  390 PRINT TAB35"*******TOTAL INVENTORY VALUE=";C0
  400 PRINT
  410 END
```

☆Add line 25 and change all other starred lines for use with the mass memory. See page 1-2 for instructions.

APPENDIX III

PRIMARY (SHELL) SORT

When only one row or column number appears in a SORT statement, the 9830A performs a primary or shell sort. A shell sort sequences the items in a list by comparing two items which are a certain number of positions away from each other and then exchanging them if they are not already in correct sorted order. For example, a primary sort is performed on the column of 5 numbers below.

The number of items (n) in the list is divided by two and the resulting integer becomes the distance (d) between the numbers to be compared. To begin this example d=2, so the first number in the list is compared with the number in the d+1 position. When all comparisons are made, the distance (d) is recalculated and the comparisons are continued until d=0.

9
5
2
5
4

d=5/2=2.5=2

Pass 1

9 is compared with 2 and an exchange is made.

2
5
9
5
4

9 is compared with 4 and an exchange is made.

2
5
4
5
9

5 is compared with 5 and no exchange is made because they are equal.

2
5
4
5
9

Pass 1 is now complete since 5 cannot be compared with the non-existent 6th element in the list. The distance is recalculated by dividing it by two and the resulting integer becomes the new distance.

2
5
4
5
9
?

d=2/2=1

Pass 2

2 is compared with 5 and no exchange is made.	2 5 4 5 9
5 is compared with 4 and an exchange is made.	2 4 5 5 9
5 is compared with 5 and no exchange is made because they are equal.	2 4 5 5 9
5 is compared with 9 and no exchange is made.	2 4 5 5 9

Pass 2 is complete and no further sorting is required because d=½=.5=0. The shell sorting technique causes items that are very much out of sequence to be sorted quickly. With each pass the value of d is recalculated until it equals zero. The sort is then complete. The shell sort usually requires less passes over the list of items to achieve sorted order and is generally faster than the bubble sort.

SECONDARY (BUBBLE) SORT

When more than one row or column number appears in a SORT statement, the 9830A performs a secondary or bubble sort. A bubble sort sequences the items in a table by interchanging adjacent pairs of items in a table until the table is completely sorted. For example, a secondary sort is performed on the two columns of numbers below.

9	8
5	6
2	1
5	3
4	5

Pass 1

9 is compared with 5 and an exchange is made.

5	6
9	8
2	1
5	3
4	5

9 is compared with 2 and an exchange is made.

5	6
2	1
9	8
5	3
4	5

9 is compared with 5 and an exchange is made.

5	6
2	1
5	3
9	8
4	5

9 is compared with 4 and an exchange is made.

5	6
2	1
5	3
4	5
9	8

Pass 1 is complete and the pair of numbers in the lowest or bottom position of the table are now in correct sorted order.

Pass 2

5 is compared with 2 and an exchange is made.

2	1
5	6
5	3
4	5
9	8

5 is compared with 5 and because they are equal the second column is checked. 6 is compared with 3 and 56 and 53 are exchanged.

2	1
5	3
5	6
4	5
9	8

5 is compared with 4 and an exchange is made.

2	1
5	3
4	5
5	6
9	8

5 is compared with 9 and no exchange is made.

2	1
5	3
4	5
5	6
9	8

Pass 2 is complete because all comparisons between unsorted items have been made. The numbers in the two lowest positions are now in correct sorted order.

Pass 3

2 is compared with 5 and no exchange is made.

2	1
5	3
4	5
5	6
9	8

5 is compared with 4 and an exchange is made.

2	1
4	5
5	3
5	6
9	8

Pass 3 is complete since all comparisons between unsorted items have been made. Now the numbers in the bottom three positions are correctly sorted.

Pass 4

2 is compared with 4 and no exchange is made.

2	1
4	5
5	3
5	6
9	8

Pass 4 is complete since all unsorted items have been compared leaving the items in the bottom 4 positions of the list in sorted order. The bubble sort is now complete in 4 passes.

APPENDIX IV

Listings and printouts for the programs in this manual that use the 20 data files from Appendix I follow.

The program below prints the information stored in the 20 data files.

```
☆10 COM P,D$[20],M$[12],C
☆20 FOR I=1 TO 20
☆30 LOAD  DATA I
 40 PRINT P,D$,TAB45;M$;TAB55;C
 50 NEXT I
 60 END
```

```
3948        DESK                         MATHERS     149.95
2548        BOOKCASE                     SMYTHE      199.99
8655        BOOKCASE                     SMYTHE      50
3051        CHAIR WITH ARMS              JONES       35.5
7711        CHAIR WITH ARMS              JONES       74.99
9086        CABINET                      WINDSOR     89.99
3667        BOOKCASE                     SMYTHE      120.5
5213        CHAIR WITHOUT ARMS           JONES       19.95
4520        CHAIR WITH ARMS              HALSTEAD    249.99
8531        DESKLAMP                     MATHERS     249.99
7051        WASTEBASKET                  JONES       5
4377        DESK                         MATHERS     50
9886        CABINET                      WINDSOR     120.5
1478        CABINET                      JONES       599.99
9932        CABINET                      WINDSOR     249.99
7128        CABINET                      WINDSOR     89.99
4439        DESK                         HALSTEAD    499.95
1086        ASHTRAY                      WINDSOR     5
8586        ASHTRAY                      JONES       25
3989        WASTEBASKET                  HALSTEAD    35
```

☆Add line 15 (FILES INVEN) and change the other starred line for use with the mass memory. See page 1-2 for instructions.

The listing below prints a directory containing part numbers, costs and file numbers,which uses a variable to perform a sort either by part number or cost.

```
   10 COM P,D$[20],M$[12],C
   20 DIM A[20,3]
☆ 30 FOR I=1 TO 20
☆ 40 LOAD  DATA I
   50 A[I,1]=P
   60 A[I,2]=C
   70 A[I,3]=I
   80 NEXT I
   82 DISP "SORT BY PART NO=1;BY COST=2";
   84 INPUT X
   86 SORT A,C,X
   110 FOR J=1 TO 20
   120 PRINT A[J,1];A[J,2];A[J,3]
   130 NEXT J
   1000 END
```

Sort by part number:

1086	5	18
1478	599.99	14
2548	199.99	2
3051	35.5	4
3667	120.5	7
3948	149.95	1
3989	35	20
4377	50	12
4439	499.95	17
4520	249.99	9
5218	19.95	8
7051	5	11
7128	89.99	16
7711	74.99	5
8531	249.99	10
8586	25	19
8655	50	3
9086	89.99	6
9896	120.5	13
9932	249.99	15

Sort by cost:

1086	5	18
7051	5	11
5218	19.95	8
8586	25	19
3989	35	20
3051	35.5	4
4377	50	12
8655	50	3
7711	74.99	5
7128	89.99	16
9086	89.99	6
9886	120.5	13
3667	120.5	7
3948	149.95	1
2548	199.99	2
8531	249.99	10
4520	249.99	9
9932	249.99	15
4439	499.95	17
1478	599.99	14

☆Add line 25 and change the other starred line for use with the mass memory. See page 1-2 for instructions.

The following program performs a major sort on cost and then a minor sort (if two costs are equal) on part numbers. The sorted array is then printed.

```
  10 COM P,D$[20],M$[12],C
  20 DIM A[20,3]
☆ 30 FOR I=1 TO 20
☆ 40 LOAD  DATA I
  50 A[I,1]=P
  60 A[I,2]=C
  70 A[I,3]=I
  80 NEXT I
  90 SORT A,C,2,1
  110 FOR J=1 TO 20
  120 PRINT A[J,1],A[J,2],A[J,3]
  130 NEXT J
  1000 END
```

1086	5	18
7051	5	11
5218	19.95	8
8586	25	19
3989	35	20
3051	35.5	4
4377	50	12
8655	50	3
7711	74.99	5
7128	89.99	16
9086	89.99	6
3667	120.5	7
9886	120.5	13
3948	149.95	1
2548	199.99	2
4520	249.99	9
8531	249.99	10
9932	249.99	15
4439	499.95	17
1478	599.99	14

☆Add line 25 and change the other starred line for use with the mass memory. See page 1-2 for instructions.

The program below sorts two arrays by 12 characters of D$, the description. The sorted arrays are then printed.

```
 10 COM P,D$[20],M$[12],C
 20 DIM A[20,2],B[20,17]
☆
 30 FOR I=1 TO 20
☆40 LOAD  DATA I
 50 A[I,1]=P
 60 A[I,2]=C
 70 B[I,1]=I
 80 TRANSFER D$ TO B[I,2]
 90 TRANSFER M$ TO B[I,12]
 100 NEXT I
 110 SORT B,C,2,3,4,5,6,7
 120 FOR J=1 TO 20
 130 TRANSFER B[J,2] TO D$
 140 TRANSFER B[J,12] TO M$
 150 PRINT A[B[J,1],1],D$,M$,A[B[J,1],2]
 160 NEXT J
 1000 END
```

1086	ASHTRAY	WINDSOR	5
8586	ASHTRAY	JONES	25
2548	BOOKCASE	SMYTHE	199.99
8655	BOOKCASE	SMYTHE	50
3667	BOOKCASE	SMYTHE	120.5
9086	CABINET	WINDSOR	89.99
9886	CABINET	WINDSOR	120.5
1478	CABINET	JONES	599.99
9932	CABINET	WINDSOR	249.99
7128	CABINET	WINDSOR	89.99
3051	CHAIR WITH ARMS	JONES	35.5
7711	CHAIR WITH ARMS	JONES	74.99
4520	CHAIR WITH ARMS	HALSTEAD	249.99
5218	CHAIR WITHOUT ARMS	JONES	19.95
3948	DESK	MATHERS	149.95
4377	DESK	MATHERS	50
4439	DESK	HALSTEAD	499.95
8531	DESKLAMP	MATHERS	249.99
7051	WASTEBASKET	JONES	5
3989	WASTEBASKET	HALSTEAD	35

☆Add line 25 and change the other starred line for use with the mass memory. See page 1-2 for instructions.

The program that follows performs a major sort on the first six characters of M$ (manufacturers) and then (if two manufacturers are identical) a minor sort on D$ (descriptions). The printout is shown below.

```
 10 COM P,D$[20],M$[12],C
☆20 DIM A[20,2],B[20,17]
 30 FOR I=1 TO 20
☆40 LOAD  DATA I
 50 A[I,1]=P
 60 A[I,2]=C
 70 B[I,1]=I
 80 TRANSFER D$ TO B[I,2]
 90 TRANSFER M$ TO B[I,12]
 100 NEXT I
 110 SORT B,C,12,13,14,2,3,4
 120 FOR J=1 TO 20
 130 TRANSFER B[J,2] TO D$
 140 TRANSFER B[J,12] TO M$
 150 PRINT A[B[J,1],1],D$,M$,A[B[J,1],2]
 160 NEXT J
 1000 END
```

4520	CHAIR WITH ARMS	HALSTEAD	249.99
4439	DESK	HALSTEAD	499.95
3989	WASTEBASKET	HALSTEAD	35
8586	ASHTRAY	JONES	25
1478	CABINET	JONES	599.99
3051	CHAIR WITH ARMS	JONES	35.5
7711	CHAIR WITH ARMS	JONES	74.99
5218	CHAIR WITHOUT ARMS	JONES	19.95
7051	WASTEBASKET	JONES	5
3948	DESK	MATHERS	149.95
4377	DESK	MATHERS	50
8531	DESKLAMP	MATHERS	249.99
2548	BOOKCASE	SMYTHE	199.99
8655	BOOKCASE	SMYTHE	50
3667	BOOKCASE	SMYTHE	120.5
1086	ASHTRAY	WINDSOR	5
9086	CABINET	WINDSOR	89.99
9886	CABINET	WINDSOR	120.5
9932	CABINET	WINDSOR	249.99
7128	CABINET	WINDSOR	89.99

☆Add line 25 and change the other starred line for use with the mass memory. See page 1-2 for instructions.

This program uses more than one SORT statement to sequence strings longer than 12 characters. The arrays are then printed.

```
  10 COM P,D$[20],M$[12],C
 ☆20 DIM A[20,2],B[20,17]
  30 FOR I=1 TO 20
 ☆40 LOAD DATA I
  50 A[I,1]=P
  60 A[I,2]=C
  70 B[I,1]=I
  80 TRANSFER D$ TO B[I,2]
  90 TRANSFER M$ TO B[I,12]
  100 NEXT I
  110 SORT B,C,7,8,9,10,11
  115 SORT B,C,2,3,4,5,6
  120 FOR J=1 TO 20
  130 TRANSFER B[J,2] TO D$
  140 TRANSFER B[J,12] TO M$
  150 PRINT A[B[J,1],1],D$,M$,A[B[J,1],2]
  160 NEXT J
  1000 END
```

```
1086     ASHTRAY                  WINDSOR     5
8586     ASHTRAY                  JONES       25
2548     BOOKCASE                 SMYTHE      199.99
8655     BOOKCASE                 SMYTHE      50
3667     BOOKCASE                 SMYTHE      120.5
9086     CABINET                  WINDSOR     89.99
9886     CABINET                  WINDSOR     120.5
1478     CABINET                  JONES       599.99
9932     CABINET                  WINDSOR     249.99
7128     CABINET                  WINDSOR     89.99
3051     CHAIR WITH ARMS          JONES       35.5
7711     CHAIR WITH ARMS          JONES       74.99
4520     CHAIR WITH ARMS          HALSTEAD    249.99
5218     CHAIR WITHOUT ARMS       JONES       19.95
3948     DESK                     MATHERS     149.95
4377     DESK                     MATHERS     50
4439     DESK                     HALSTEAD    499.95
8531     DESKLAMP                 MATHERS     249.99
7051     WASTEBASKET              JONES       5
3989     WASTEBASKET              HALSTEAD    35
```

☆Add line 25 and change the other starred line for use with the mass memory. See page 1-2 for instructions.

The following program sequences the arrays in descending order by description and then prints them out.

```
  10 COM P,D$[20],M$[12],C
☆20 DIM A[20,2],B[20,17]
  30 FOR I=1 TO 20
☆40 LOAD  DATA I
  50 A[I,1]=P
  60 A[I,2]=C
  70 B[I,1]=I
  80 TRANSFER D$ TO B[I,2]
  90 TRANSFER M$ TO B[I,12]
  100 NEXT I
  105 MAT A=(-1)*A
  106 MAT B=(-1)*B
  110 SORT B,C,11
  112 SORT B,C,6,7,8,9,10
  115 SORT B,C,2,3,4,5
  116 MAT A=(-1)*A
  117 MAT B=(-1)*B
  120 FOR J=1 TO 20
  130 TRANSFER B[J,2] TO D$
  140 TRANSFER B[J,12] TO M$
  150 PRINT A[B[J,1],1],D$,M$,A[B[J,1],2]
  160 NEXT J
  1000 END
```

```
7051    WASTEBASKET            JONES      5
3989    WASTEBASKET            HALSTEAD   35
8531    DESKLAMP               MATHERS    249.99
3948    DESK                   MATHERS    149.95
4377    DESK                   MATHERS    50
4439    DESK                   HALSTEAD   499.95
5218    CHAIR WITHOUT ARMS     JONES      19.95
3051    CHAIR WITH ARMS        JONES      35.5
7711    CHAIR WITH ARMS        JONES      74.99
4520    CHAIR WITH ARMS        HALSTEAD   249.99
9086    CABINET                WINDSOR    89.99
9886    CABINET                WINDSOR    120.5
1478    CABINET                JONES      599.99
9932    CABINET                WINDSOR    249.99
7128    CABINET                WINDSOR    89.99
2548    BOOKCASE               SMYTHE     199.99
8655    BOOKCASE               SMYTHE     50
3667    BOOKCASE               SMYTHE     120.5
1086    ASHTRAY                WINDSOR    5
8586    ASHTRAY                JONES      25
```

☆Add line 25 and change the other starred line for use with the mass memory. See page 1-2 for instructions.

This program formats the dollar amounts from the data files using the STRING statement. The printout follows.

```
  10 COM P,D$[20],M$[12],C
☆20 DIM A[20,2],B[20,17],I$[10]
  30 FOR I=1 TO 20
☆40 LOAD  DATA I
  50 I$="$ΔΔΔ.ΔΔ"
  60 STRING C,I$,2
  70 PRINT P,D$;TAB40;M$;TAB55;I$
  80 NEXT I
  1000 END
```

```
3948     DESK                     MATHERS    $149.95
2548     BOOKCASE                 SMYTHE     $199.99
8655     BOOKCASE                 SMYTHE      $50.00
3051     CHAIR WITH ARMS          JONES       $35.50
7711     CHAIR WITH ARMS          JONES       $74.99
9086     CABINET                  WINDSOR     $89.99
3667     BOOKCASE                 SMYTHE     $120.50
5218     CHAIR WITHOUT ARMS       JONES       $19.95
4520     CHAIR WITH ARMS          HALSTEAD   $249.99
8531     DESKLAMP                 MATHERS    $249.99
7051     WASTEBASKET              JONES        $5.00
4377     DESK                     MATHERS     $50.00
9886     CABINET                  WINDSOR    $120.50
1478     CABINET                  JONES      $599.99
9932     CABINET                  WINDSOR    $249.99
7128     CABINET                  WINDSOR     $89.99
4439     DESK                     HALSTEAD   $499.95
1086     ASHTRAY                  WINDSOR      $5.00
8586     ASHTRAY                  JONES       $25.00
3989     WASTEBASKET              HALSTEAD    $35.00
```

☆Add line 25 and change the other starred line for use with the mass memory. See page 1-2 for instructions.

The program below is based on the Sort Example in Appendix II for the 20 data files. It uses the STRING statement to get a printout which has formatted dollar amounts. (See lines 20, 191, 192, 200, 281, 282 and 290.)

```
  10 COM P,D$[20],M$[12],C
☆ 20 DIM A[10,11],B[10,11],A$[20],B$[20],I$[10]
  30 FOR I=1 TO 10
☆ 40 LOAD  DATA I
  50 TRANSFER D$ TO A[I,1]
  60 A[I,11]=I
  70 NEXT I
  80 FOR J=11 TO 20
☆ 90 LOAD  DATA J
  100 TRANSFER D$ TO B[J-10,1]
  110 B[J-10,11]=J
  120 NEXT J
  130 SORT A,C,1,2,3,4,5
  140 SORT B,C,1,2,3,4,5
  150 A0=B0=1
  160 TRANSFER A[A0,1] TO A$
  170 TRANSFER B[B0,1] TO B$
  180 IF A$>B$ THEN 280
☆ 190 LOAD  DATA A[A0,11]
  191 I$="$ΔΔΔ.ΔΔ"
  192 STRING C,I$,2
  200 PRINT M$,D$,TAB50;P;TAB60;I$
  210 A0=A0+1
  220 IF A0 <= 10 THEN 260
  230 IF B0>10 THEN 370
  240 A$="↑↑↑↑↑"
  250 GOTO 180
  260 TRANSFER A[A0,1] TO A$
  270 GOTO 180
☆ 280 LOAD  DATA B[B0,11]
  281 I$="$ΔΔΔ.ΔΔ"
  282 STRING C,I$,2
  290 PRINT M$,D$,TAB50;P;TAB60;I$
  300 B0=B0+1
  310 IF B0 <= 10 THEN 350
  320 IF A0>10 THEN 370
  330 B$="↑↑↑↑↑"
  340 GOTO 180
  350 TRANSFER B[B0,1] TO B$
  360 GOTO 180
  370 END
```

☆Add line 25 and change all other starred lines for use with the mass memory. See page 1-2 for instructions.

WINDSOR	ASHTRAY	1086	$5.00
JONES	ASHTRAY	8586	$25.00
SMYTHE	BOOKCASE	2548	$199.99
SMYTHE	BOOKCASE	8655	$50.00
SMYTHE	BOOKCASE	3667	$120.50
WINDSOR	CABINET	9086	$89.99
WINDSOR	CABINET	9886	$120.50
JONES	CABINET	1478	$599.99
WINDSOR	CABINET	9932	$249.99
WINDSOR	CABINET	7128	$89.99
JONES	CHAIR WITH ARMS	3051	$35.50
JONES	CHAIR WITH ARMS	7711	$74.99
JONES	CHAIR WITHOUT ARMS	5218	$19.95
HALSTEAD	CHAIR WITH ARMS	4520	$249.99
MATHERS	DESK	3948	$149.95
MATHERS	DESK	4377	$50.00
HALSTEAD	DESK	4439	$499.95
MATHERS	DESKLAMP	8531	$249.99
JONES	WASTEBASKET	7051	$5.00
HALSTEAD	WASTEBASKET	3989	$35.00

The following program is a modification of the Search Example in Appendix II which locates strings. It uses the STRING statement to get a printout containing formatted dollar amounts. (See lines 20, 221, 222 and 230.)

```
  10 COM P,D$[20],M$[12],C
  20 DIM A[20,2],B[20,17],C[1,10],X$[20],Y$[20],I$[10]
☆ 30 FOR I=1 TO 20
☆ 40 LOAD  DATA I
  50 TRANSFER D$ TO B[I,1]
  60 TRANSFER M$ TO B[I,11]
  70 B[I,17]=I
  80 A[I,1]=P
  90 A[I,2]=C
  100 NEXT I
  110 DISP "SEARCH FOR; TO END PRESS 0";
  120 INPUT X$[1,20]
  130 IF X$[1,1]="0" THEN 320
  140 TRANSFER X$ TO C[1,1]
  150 SEARCH B,C,1,C[1,1],Z
  160 IF Z=0 THEN 260
  170 TRANSFER B[Z,1] TO Y$
  180 IF Y$=X$ THEN 210
  190 B[Z,1]=-B[Z,1]
  200 GOTO 150
  210 TRANSFER B[Z,1] TO D$
  220 TRANSFER B[Z,11] TO M$
  221 I$="$ΔΔΔ.ΔΔ"
  222 STRING A[B[Z,17],2],I$,2
  230 PRINT A[B[Z,17],1],D$,M$,I$
  240 B[Z,1]=-B[Z,1]
  250 GOTO 150
  260 FOR I=1 TO 20
  270 B[I,1]=ABSB[I,1]
  280 NEXT I
  290 PRINT "SEARCH COMPLETE"
  300 PRINT
  310 GOTO 110
  320 PRINT "END OF SEARCH"
  330 END
```

☆Add line 25 and change all other starred lines for use with the mass memory. See page 1-2 for instructions.

```
 3051          CHAIR WITH ARMS       JONES       $35.50
 7711          CHAIR WITH ARMS       JONES       $74.99
 4520          CHAIR WITH ARMS       HALSTEAD   $249.99
SEARCH COMPLETE

 5218          CHAIR WITHOUT ARMS    JONES       $19.95
SEARCH COMPLETE

 3948          DESK                  MATHERS    $149.95
 4377          DESK                  MATHERS     $50.00
 4439          DESK                  HALSTEAD   $499.95
SEARCH COMPLETE

 8531          DESKLAMP              MATHERS    $249.99
SEARCH COMPLETE

 7051          WASTEBASKET           JONES        $5.00
 3989          WASTEBASKET           HALSTEAD    $35.00
SEARCH COMPLETE

 2548          BOOKCASE              SMYTHE     $199.99
 8655          BOOKCASE              SMYTHE      $50.00
 3667          BOOKCASE              SMYTHE     $120.50
SEARCH COMPLETE

 1086          ASHTRAY               WINDSOR      $5.00
 8586          ASHTRAY               JONES       $25.00
SEARCH COMPLETE

 9086          CABINET               WINDSOR     $89.99
 9886          CABINET               WINDSOR    $120.50
 1478          CABINET               JONES      $599.99
 9932          CABINET               WINDSOR    $249.99
 7128          CABINET               WINDSOR     $89.99
SEARCH COMPLETE

END OF SEARCH
```

The following program is based on the Flag Example in Appendix II. It uses the STRING statement to print out the formatted dollar amounts. (See lines 20, 251, 252, 260, 331, 332, 340, 361, 362, 370, 381, 382, 390). The four possible printouts follow.

```
 10 COM P,D$[20],M$[12],C
 20 DIM A[20,7],A$[10],X$[12],I$[40]
☆30 CFLAG 1
 40 CFLAG 2
 50 DISP "PRINT VALUE OF EACH ITEM (Y/N)";
 60 INPUT A$
 70 IF A$#"Y" THEN 90
 80 SFLAG 1
 90 DISP "PRINT SUBTOTALS BY MFG (Y/N)";
 100 INPUT A$
 110 IF A$#"Y" THEN 130
 120 SFLAG 2
 130 FOR I=1 TO 20
☆140 LOAD  DATA I
 150 TRANSFER M$ TO A[I,1]
 160 A[I,7]=I
 170 NEXT I
 180 SORT A,C,1,2,3,4,5,6
☆190 LOAD  DATA A[1,7]
 200 X$=M$
 210 C0=C1=0
 220 FOR I=1 TO 20
☆230 LOAD  DATA A[I,7]
 240 IF  NOT FLAG2 THEN 320
 250 IF M$=X$ THEN 310
 251 I$="***SUBTOTAL***$ΔΔΔΔ.ΔΔ"
 252 STRING C1,I$[15],2
 260 PRINT TAB38;X$;TAB50;I$
 270 PRINT
 280 C1=C
 290 X$=M$
 300 GOTO 320
 310 C1=C1+C
 320 C0=C0+C
 330 IF  NOT FLAG1 THEN 350
 331 I$="$ΔΔΔ.ΔΔ"
 332 STRING C,I$,2
 340 PRINT TAB5;P;TAB15;D$;TAB35;M$;TAB50;I$
 350 NEXT I
 360 IF  NOT FLAG2 THEN 381
 361 I$="***SUBTOTAL***$ΔΔΔΔ .ΔΔ"
 362 STRING C1,I$[15],2
 370 PRINT TAB38;X$;TAB50;I$
 380 PRINT
 381 I$="*****TOTAL INVENTORY VALUE=$ΔΔΔΔΔ.ΔΔ"
 382 STRING C0,I$[28],2
 390 PRINT TAB35;I$
 400 PRINT
 410 END
```

☆Add line 25 and change all other starred lines for use with the mass memory. See page 1-2 for instructions.

Printout of ITEM VALUES and SUBTOTALS.

```
4520  CHAIR WITH ARMS     HALSTEAD    $249.99
4439  DESK                HALSTEAD    $499.95
3989  WASTEBASKET         HALSTEAD     $35.00
                             HALSTEAD  ***SUBTOTAL*** $784.94

3051  CHAIR WITH ARMS     JONES        $35.50
7711  CHAIR WITH ARMS     JONES        $74.99
5218  CHAIR WITHOUT ARMS  JONES        $19.95
7051  WASTEBASKET         JONES         $5.00
1478  CABINET             JONES       $599.99
8586  ASHTRAY             JONES        $25.00
                             JONES     ***SUBTOTAL*** $760.43

3948  DESK                MATHERS     $149.95
8531  DESKLAMP            MATHERS     $249.99
4377  DESK                MATHERS      $50.00
                             MATHERS   ***SUBTOTAL*** $449.94

2548  BOOKCASE            SMYTHE      $199.99
8655  BOOKCASE            SMYTHE       $50.00
3667  BOOKCASE            SMYTHE      $120.50
                             SMYTHE    ***SUBTOTAL*** $370.49

9086  CABINET             WINDSOR      $89.99
9886  CABINET             WINDSOR     $120.50
9932  CABINET             WINDSOR     $249.99
7128  CABINET             WINDSOR      $89.99
1086  ASHTRAY             WINDSOR       $5.00
                             WINDSOR   ***SUBTOTAL*** $555.47

                      *****TOTAL INVENTORY VALUE= $2921.27
```

Printout of ITEM VALUES only.

```
4520     CHAIR WITH ARMS      HALSTEAD       $249.99
4439     DESK                 HALSTEAD       $499.95
3989     WASTEBASKET          HALSTEAD        $35.00
3051     CHAIR WITH ARMS      JONES           $35.50
7711     CHAIR WITH ARMS      JONES           $74.99
5218     CHAIR WITHOUT ARMS   JONES           $19.95
7051     WASTEBASKET          JONES            $5.00
1478     CABINET              JONES          $599.99
8586     ASHTRAY              JONES           $25.00
3948     DESK                 MATHERS        $149.95
8531     DESKLAMP             MATHERS        $249.99
4377     DESK                 MATHERS         $50.00
2548     BOOKCASE             SMYTHE         $199.99
8655     BOOKCASE             SMYTHE          $50.00
3667     BOOKCASE             SMYTHE         $120.50
9086     CABINET              WINDSOR         $89.99
9886     CABINET              WINDSOR        $120.50
9932     CABINET              WINDSOR        $249.99
7128     CABINET              WINDSOR         $89.99
1086     ASHTRAY              WINDSOR          $5.00
                    *****TOTAL INVENTORY VALUE= $2921.27
```

Printout of SUBTOTALS only.

```
HALSTEAD     ***SUBTOTAL*** $784.94
JONES        ***SUBTOTAL*** $760.43
MATHERS      ***SUBTOTAL*** $449.94
SMYTHE       ***SUBTOTAL*** $370.49
WINDSOR      ***SUBTOTAL*** $555.47
*****TOTAL INVENTORY VALUE= $2921.27
```

Printout of TOTAL INVENTORY VALUE only. (All four printouts include this.)

```
*****TOTAL INVENTORY VALUE= $2921.27
```

The program that follows checks the tape files for ERROR 59 (tape cassette check sum error) using the error recovery routine.

```
10 COM P,D$[20],M$[12],C
20 SERROR E,110
30 I=0
40 F=1
50 LOAD  DATA F
60 PRINT D$,P
70 F=F+1
80 IF F <= 20 THEN 50
90 PRINT "DONE"
100 END
110 WAIT 1000
120 IF E#59 THEN 190
130 I=I+1
140 IF I<3 THEN 50
150 PRINT "**UNABLE TO READ FILE";F;"SKIPPING TO FILE";F+1
160 I=0
170 F=F+1
180 GOTO 50
190 PRINT "**ERROR CONDITION NOT PROGRAMMED;ERROR=";E
200 END
```

```
DESK                3948
BOOKCASE            2548
BOOKCASE            8655
CHAIR WITH ARMS 3051
CHAIR WITH ARMS 7711
CABINET             9086
BOOKCASE            3667
CHAIR WITHOUT ARMS              5218
CHAIR WITH ARMS 4520
DESKLAMP            8531
**UNABLE TO READ FILE 11   SKIPPING TO FILE 12
DESK                4377
CABINET             9886
CABINET             1478
CABINET             9932
CABINET             7128
DESK                4439
ASHTRAY             1086
ASHTRAY             8586
WASTEBASKET         3989
DONE
```

HEWLETT hp PACKARD

SALES & SERVICE OFFICES

UNITED STATES

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

Medical Only
228 W. Valley Ave.,
Room 302
Birmingham 35209
Tel: (205) 879-2081/2

ARIZONA
2336 E. Magnolia St.
Phoenix 85034
Tel: (602) 244-1361
TWX: 910-951-1331

2424 East Aragon Rd.
Tucson 85706
Tel: (602) 889-4661

***ARKANSAS**
Medical Service Only
Little Rock 72205
Tel: (501) 664-8773

CALIFORNIA
1430 East Orangethorpe Ave.
Fullerton 92631
Tel: (714) 870-1000
TWX: 910-592-1288

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
TWX: 910-367-2092

9606 Aero Drive
P.O. Box 23333
San Diego 92123
Tel: (714) 279-3200
TWX: 910-335-2000

Calculators Only
601 California St.
San Francisco 94108
Tel: (415) 989-8470

COLORADO
5600 South Ulster Parkway
Englewood 80110
Tel: (303) 771-3455
TWX: 910-935-0705

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.D. Box 13910
6177 Lake Ellenor Dr.
Orlando 32809
Tel: (305) 859-2900
TWX: 810-850-0113

21 East Wright St.
Suite 1
Pensacola 32501
Tel: (904) 434-3081

GEORGIA
P.O. Box 28234
450 Interstate North
Atlanta 30328
Tel: (404) 434-4000
TWX: 810-766-4890

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

LOUISIANA
P.O. Box 840
3239 Williams Boulevard
Kenner 70062
Tel: (504) 721-6201
TWX: 810-955-5524

KENTUCKY
Medical/Calculator Only
8003 Troutwood Court
Louisville 40291
Tel: (502) 426-4341

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

4 Choke Cherry Road
Rockville 20850
Tel: (301) 948-6370
TWX: 710-828-9685
710-828-0487

P.O. Box 1648
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 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
11902 Elm Street
Suite 4C
Omaha 68144
Tel: (402) 333-6017

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

NEW MEXICO
P.D. 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

Calculators Only
1251 Avenue of the Americas
Floor 32 - Suite 3296
New York City 10020
Tel: (212) 265-5575

New York City
Manhattan, Bronx
Contact Paramus, NJ Dffice
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: 510-221-2168

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
Night: 243-7305
TWX: 810-423-9431

330 Progress Rd.
Dayton 45449
Tel: (513) 859-8202
TWX: 810-459-1925

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

OKLAHOMA
P.D. Box 32008
Oklahoma City 73132
Tel: (405) 721-0200
TWX: 910-830-6862

OREGON
17890 SW Boones Ferry Road
Tualatin 97062
Tel: (503) 620-3350
TWX: 910-467-8714

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
2890 South Main Street
Salt Lake City 84115
Tel: (801) 487-0715
TWX: 910-925-5681

VIRGINIA
Medical Only
P.D. 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 SE
Bellevue 98004
Tel: (206) 454-3971
TWX: 910-443-2446

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

WISCONSIN
9431 W. Beloit Road
Suite 117
Milwaukee 53227
Tel: (414) 541-0550

FOR U.S. AREAS NOT LISTED:
Contact the regional office nearest you: Atlanta, Georgia... North Hollywood, California... Rockville, (4 Choke Cherry Rd.) Maryland...Skokie, Illinois. Their complete addresses are listed above.

***Service Only**

CANADA

ALBERTA
Hewlett-Packard (Canada) Ltd.
11748 Kingsway Ave.
Edmonton TSG OX5
Tel: (403) 452-3670
TWX: 610-831-2431

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

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

MANITOBA
Hewlett-Packard (Canada) Ltd.
513 Century St.
St. James
Winnipeg R3H OL8
Tel: (204) 786-7581
TWX: 610-671-3531

NOVA SCOTIA
Hewlett-Packard (Canada) Ltd.
800 Windmill Road
Dartmouth B3C 1L1
Tel: (902) 469-7820

ONTARIO
Hewlett-Packard (Canada) Ltd.
1785 Woodward Or.
Ottawa K2C OP9
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

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

CENTRAL AND SOUTH AMERICA

ARGENTINA
Hewleff-Packard Argentina
S.A.C.e.I
Lavalle 1171-3° Piso
Buenos Aires
Tel: 35-0436, 35-0627, 35-0341
Telex: 012-1009
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, 1.152-Bela Vista
01307-**São Paulo**-SP
Tel: 288-71-11, 287-81-20, 287-61-93
Telex: 309151/2/3
Cable: HEWPACK São Paulo

Hewlett-Packard Do Brasil
I.E.C. Ltda.
Praca Dom Feliciano, 78-8° andar (Sala 806/8)
9000-**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
2000-**Rio de Janeiro**-GB
Tel: 257-80-94-DDD (021)
Telex: 2100 79 HEWPACK
Cable: HEWPACK Rio de Janeiro

CHILE
Calcagni y Metcalfe Ltda.
Calle Lira 81, Oficina 5
Casilla 2118
Santiago, 1
Tel: 398613
Cable: CALMET

COLOMBIA
Instrumentación
Henrik A. Langebaek & Kier S.A.
Carrera 7 No. 48-59
Apartado Aéreo 6287
Bogota, 1 D.E.
Tel: 45-78-06, 45-55-46
Cable: AARIS Bogota
Telex: 44400INSTCO

COSTA RICA
Cientifica Costarricense S.A.
Apartado 10159
San José
Tel: 21-86-13
Cable: GALGUR San José

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

NICARAGUA
Roberto Terán G.
Apartado Postal 689
Edificio Terán
Managua
Tel: 3451, 3452
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.
Division: Aparatos y Equipos Medicos
Division: Aparatos y Equipos Scientificos y de Investigacion
P.O. Box 676
Chile, 482, Edificio Victoria
Asunción
Tel: 4-5069, 4-6272
Cable: RAMEL

PERU
Compañia Electro Médica S.A.
Ave. Enrique Canaval 312
San Isidro
Casilla 1030
Lima
Tel: 22-3900
Cable: ELMED Lima

PUERTO RICO
San Juan Electronics, Inc.
P.D. Box 5167
Ponce de León 154
Pda. 3-PTA de Tierra
San Juan 00906
Tel: (809) 725-3342, 722-3342
Cable: SATRONICS San Juan
Telex: SATRON 3450 332

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
Edificio Segre
Tercera Transversal
Los Ruices Norte
Caracas 107
Tel: 35-00-11
Telex: 21146 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) 33 66 06 to 09
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: (01) 81 66 40
Cable: HEWPACK AS
Telex: 186 40 hp as

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

FINLAND
Hewlett-Packard Oy
Nahkahousuntie 5
P.O. Box 6
SF-00211 **Helsinki** 21
Tel: 6923031
Cable: HEWPACKOY Helsinki
Telex: 12-15363

FRANCE
Hewlett-Packard France
Quartier de Courtaboeuf
Boite Postale No. 6
F-91401 **Orsay**
Tel: (1) 907 78 25
Cable: HEWPACK Orsay
Telex: 60048

Hewlett-Packard France
Agence Régional
Chemin des Mouilles
Boite Postale No. 12
F-69130 **Ecully**
Tel: (78) 33 81 25,
Cable: HEWPACK Ecully
Telex: 31 617

Hewlett-Packard France
Agence Régionale
Zone Aéronautique
Avenue Clément Ader
F-31770 **Colomiers**
Tel: (61) 78 11 55
Telex: 51957

Hewlett-Packard France
Agence Régionale
Centre d'aviation générale
F-13721 **Aéroport de Marignane**
Tel: (91) 89 12 36
TWX: 41770 F

Hewlett-Packard France
Agence Régionale
63, Avenue de Rochester
F-35000 **Rennes**
Tel: 74912 F
Telex: 74 912 F

Hewlett-Packard France
Agence Régionale
74, Allée de la Robertsau
F-67000 **Strasbourg**
Tel: (88) 35 23 20/21
Telex: 89141
Cable: HEWPACK STRBG

Medical/Calculator Only
Hewlett-Packard France
Agence Régionale
Centre Vauban
201, rue Colbert
Entrée Az
F-59000 **Lille**
Tel: (20) 51 44 14

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: 41 32 49 fra

Hewlett-Packard GmbH
Technisches Buero Böblingen
Herrenbergerstrasse 110
D-7030 **Böblingen**, Württemberg
Tel: (07031) 66 72 87
Cable: HEPAK Böblingen
Telex: 72 65 739 bbn

Hewlett-Packard GmbH
Technisches Buero Düsseldorf
Vogelsanger Weg 38
0-4000 **Düsseldorf**
Tel: (0211) 63 80 31/5
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
0-3000 **Hannover-Kleefeld**
Tel: (0511) 55 60 46
Telex: 092 3259

Hewlett-Packard GmbH
Technisches Buero Nuremberg
Hersbrückerstrasse 42
D-8500 **Nuremberg**
Tel: (0911) 57 10 66
Telex: 623 860

Hewlett-Packard GmbH
Technisches Buero Müchen
Unterhachinger Strasse 28
ISAR Center
0-8012 **Ottobrunn**
Tel: (089) 601 30 61/7
Telex: 52 49 85
Cable: HEWPACKSA München

(West Berlin)
Hewlett-Packard GmbH
Technisches Buero Berlin
Keith Strasse 2-4
D-1000 **Berlin** 30
Tel: (030) 24 90 86
Telex: 18 34 05 hpbln d

GREECE
Kostas Karayannis
18, Ermou Street
GR-**Athens** 126
Tel: 3230-303 Sales/SVC
3230-305 Adm. Order Proc.
Cable: RAKAR Athens
Telex: 21 59 62 rkar gr

Hewlett-Packard S.A.
Mediterranean & Middle East Operations
35 Kolokotroni Street
Platia Kefallariou
Gr-Kifissia-**Athens**
Tel: 8080337, 8080359, 8080429, 8018693
Telex: 21 6588
Cable: HEWPACKSA Athens

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
Cable: ETALAK Athens
Telex: 21-4893 ETAL GR

IRELAND
Hewlett-Packard Ltd.
King Street Lane
Winnersh, Wokingham
GB-**Berkshire** RG11 5AR
Tel: Wokingham 784774
Telex: 847178/848179

Hewlett-Packard Ltd.
"The Graftons"
Stamford New Road
GB-**Altrincham**, Cheshire
Tel: (061) 928-9021
Telex: 668068

ITALY
Hewlett-Packard Italiana S.p.A.
Via Amerigo Vespucci 2
I-20124 **Milan**
Tel: (2) 6251 (10 lines)
Cable: HEWPACKIT Milan
Telex: 32046

Hewlett-Packard Italiana S.p.A.
Via Pietro Maroncelli 40
(ang. Via Visentin)
I-35100 **Padova**
Tel: 66 40 62/66 31 88
Telex: 32046 via Milan

Medical Only
Hewlett-Packard Italiana S.p.A.
Via Medaglie d'Oro, 2
I-56100 **Pisa**
Tel: (050) 2 32 04
Telex: 32046 via Milan

Hewlett-Packard S.p.A.
Via G. Armellini 10
I-00143 **Rome**-Eur
Tel: (6) 5912544/5
Telex: 61514
Cable: HEWPACKIT Rome

Hewlett-Packard Italiana S.p.A.
Via San Quintino, 46
I-10121 **Turin**
Tel: 53 82 64/54 84 68
Telex: 32046 via Milan

Medical/Calculators Only
Hewlett-Packard Italiana S.p.A.
Via Principe Nicola 43 G/C
I-95126 **Catania**
Tel: (095) 370505

LUXEMBURG
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

NETHERLANDS
Hewlett-Packard Benelux N.V.
Weerdestein 117
P.O. Box 7825
NL-**Amsterdam**, 1011
Tel: (020) 5411522
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
Analytical/Medical Only
Hewlett-Packard
Warsaw Technical Ottice
Szpitalna 1
00-120 **Warsaw**
Tel: 268031
Telex: 812453

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
Sarl Avenida Antonio Augusto de Aguiar 138
P.O. Box 2761
P - **Lisbon**
Tel: (19) 53 21 31/7
Cable: INTERCAMBIO Lisbon

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 Ramon y Cajal, 1
Edificio Sevilla I, planta 9°
E-**Seville**
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
Alvaro Bazen, 12
(Edificio Luz)
E - **Valencia** - 10
Tel: 60 42 00

SWEDEN
Hewlett-Packard Sverige AB
Enighetsvägen 1-3
Fack
S-161 20 **Bromma** 20
Tel: (08) 730 05 50
Cable: MEASUREMENTS Stockholm
Telex: 10721

Hewlett-Packard Sverige AB
Hagakersgatan 9C
S-431 41 **Mölndal**
Tel: (031) 27 68 00/01
Telex: Via Bromma

SWITZERLAND
Hllett-Packard (Schweiz) AG
Zürcherstrasse 20
P.O. Box 64
CH-8952 Schlieren **Zurich**
Tel: (01) 98 18 21
Cable: HPAG CH
Telex: 53933 hpag

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

TURKEY
Telekom Engineering Bureau
Saglik Sok No. 15/1
Ayaspasa-Beyoglu
P.O. Box 437 Beyoglu
TR-**Istanbul**
Tel: 49 40 40
Cable: TELEMATION Istanbul

UNITED KINGDOM
Hewlett-Packard Ltd.
King Street Lane
Winnersh, Wokingham
GB-**Berkshire** RG11 5AR
Tel: Wokingham 784774
Telex: 847178/848179

Hewlett-Packard Ltd.
"The Graftons"
Stamford New Road
GB-**Altrincham**, Cheshire
Tel: (061) 928-9021
Telex: 668068

Hewlett-Packard Ltd.
c/o Makro
South Service Wholesale Centre
Amber Way
Halesowen Industrial Estate
GB-**Halesowen**, Worcs
Tel: Birmingham 7860

Hewlett-Packard Ltd.
4th Floor
Wedge House
799, London Road
GB-**Thornton Heath** CR4 6XL,
Surrey
Tel: (01) 684 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
London, EC2A1SX
Registered No. 690597

USSR
Hewlett-Packard USSR
c/o Commercial Office
American Embassy (Box M)
A-1091 **Vienna**, Austria
Tel: 221-79-71
Telex: 7825 hewpak SU

YUGOSLAVIA
Iskra-Standard/Hewlett-Packard
Topniska 58/3
61000 **Ljubljana**
Tel: 315-879/321-674
Telex: 31300

SOCIALIST COUNTRIES PLEASE CONTACT:
Hewlett-Packard S.A.
7, rue du Bois-du-Lan
P.O. Box 349
CH-1217 Meyrin 1 **Geneva**
Switzerland
Tel: (022) 41 54 00
Cable: HEWPACKSA Geneva
Telex: 2 24 86

AFRICA, ASIA, AUSTRALIA

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
Tel: 89-6351, 89-6306
Telex: 31-024
Cable: HEWPARO Melbourne

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

Hewlett-Packard Australia Pty. Ltd.
97 Churchill Road
Prospect 5082
South Australia
Tel: 44 8151
Cable: HEWPARO Adelaide

Hewlett-Packard Australia Pty. Ltd.
141 Stirling Highway
Claremont, W.A. 6010
Tel: 86-5455
Telex: 93859
Cable: HEWPARD

Hewlett-Packard Australia Pty. Ltd.
121 Wollongong Street
Fyshwick, A.C.T., 2609
Tel: 95 3733

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

CEYLON
United Electricals Ltd.
P.O. Box 681
60, Park St.
Colombo 2
Tel: 26696
Cable: HOTPOINT Colombo

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

HONG KONG
Schmidt & Co.(Hong Kong) Ltd.
P.O. Box 297
Connalight Centre
39th Floor
Connaught Road, Central
Hong Kong
Tel: 240168, 232735
Telex: HX4766
Cable: SCHMIDTCO Hong Kong

INDIA
Blue Slar Ltd.
Kasturi Buildings
Jamshedji Tata Rd.
Bombay 400 020
Tel: 29 50 21
Telex: 3751
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
Kampur 208 001
Tel: 6 88 82
Cable: BLUESTAR

Blue Star Ltd.
7 Hare Street
P.O. Box 506
Calcutta 700 001
Tel: 23-0131
Telex: 655
Cable: BLUESTAR

Blue Star Ltd.
Blue Star House,
34 Ring Road
Lajpat Nagar
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

Blue Star Ltd.
1-1-117/1
Sarojini Oevi Road
Secunderabad 500 003
Tel: 7 63 91, 7 73 93
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: 38 04
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: 2895 Jakarta

IRAN
Multi Corp International Ltd.
Avenue Soraya 130
P.O. Box 1212
IR-**Teheran**
Tel: 83 10 35-39
Cable: MULTICORP Tehran
Telex: 2893 mci In

ISRAEL
Electronics & Engineering
Oiv. of Motorola Israel Ltd.
17 Aminadav Street
Tel-Aviv
Tel: 36941 (3 lines)
Cable: BASTEL Tel-Aviv
Telex: 33569

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.
Nisei Ibaragi Bldg.
2-2-8 Kasuga
Ibaragi-Shi
Osaka
Tel: (0726) 23-1641
Telex: 5332-385 YHP DSAKA

Yokogawa-Hewlett-Packard Ltd.
Nakamo Building
No. 24 Kamisasazima-cho
Nakamura-ku, **Nagoya** City
Tel: (052) 571-5171

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

Yokogawa-Hewlett-Packard Ltd.
Mito Mitsui Building
1-4-73 San-no-maru
Mito, 310
Tel: 0292-25-7470

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

KENYA
Technical Engineering Services
P.O. Box 18311
Nairobi, Kenya
Tel: 57726
Cable: PROTON

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

KUWAIT
Al-Khaldiya Trading & Contracting Co.
Al Soor Street
Michaan Bldg. No. 4
Kuwait
Tel: 42 99 10
Cable: VISCOUNT

LEBANON
Constantin E. Macridis
Clemenceau Street 34
P.O. Box 7213
RL-**Beirut**
Tel: 220846
Telex: 21114 Leb
Cable: ELECTRONUCLEAR Beirut

MALAYSIA
MECOMB Malaysia Ltd.
2 Lorong 13/6A
Section 13
Petaling Jaya, **Selangor**
Cable: MECOMB Kuala Lumpur

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.
94-96 Dixon Street
P.O. Box 9443
Courtenay Place,
Wellington
Tel: 59-559
Telex: 3898
Cable: HEWPACK Wellington

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

Analytical/Medical Only
Oental & Medical Supply Co. Ltd.
Scientific Division
79 Carlton Gore Road
Newmarket
P.O. Box 1234
Auckland
Tel: 75-289
Cable: DENTAL Auckland

NIGERIA
The Electronics Instrumentatiofns Ltd.
N6B/770 Oyo Road
Oluseun House
P.M.B. 5402
Ibadan
Tel: 22325
Cable: THETEIL Ibadan

The Electronics Instrumentations Ltd. (TEIL)
16th Floor Cocoa House
P.M.B. 5402
Ibadan
Tel: 22325
Cable: THETEIL Ibadan

PAKISTAN
Mushko & Company, Ltd.
Oosman Chambers
Abdullah Haroon Road
Karachi 3
Tel: 511027, 512927
Cable: CODPERATOR Karachi

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

PHILIPPINES
Electromex, Inc.
6th Floor, Amalgamated Development Corp. Bldg.
Ayala Avenue, Makati, Rizal
C.C.P.O. Box 1028
Makati, Rizal
Tel: 86-18-87, 87-76-77,
Cable: ELEMEX Manila

SINGAPORE
Mechanical & Combustion-Engineering Company Pte., Ltd.
10/12, Jalan Kilang
Red Hill Industrial Estate
Singapore, 3
Tel: 647151 (7 lines)
Cable: MECOMB Singapore

Hewlett-Packard Singapore (Pte.) Ltd.
Blk. 2, 6th FLOOR, Jalan Bukit Merah
Redhill Industrial Estate
Alexandra P.O. Box 87,
Singapore 3
Tel: 633022
Telex: HPSG RS 21486
Cable: HEWPACK, Singapore

SOUTH AFRICA
Hewlett-Packard South Africa (Pty.), Ltd.
Hewlett-Packard House
Oaphne Street, Wendywood
Sandton, Transvaal 2001
Tel: 802-1040
Telex: SA43-4782JH
Cable: HEWPACK

Hewlett-Packard South Africa (Pty.), Ltd.
Breecastle House
Bree Street
Cape Town
Tel: 2-6941/2/3
Cable: HEWPACK Cape Town
Telex: 0006 CT

Hewlett-Packard South Africa (Pty.), Ltd.
641 Ridge Road, Durban
P.O. Box 37099
Overport, Durban, 4067
Tel: 88-6102
Telex: 6-7954

TAIWAN
Hewlett-Packard Taiwan
39 Chung Shiao West Road
Sec. 1 Overseas Insurance
Corp. Bldg. 7th Floor
Taipei
Tel: 389160.1,2,
Telex: TP824 HEWPACK
Cable: HEWPACK Taipei

Hewlett-Packard Taiwan
38, Po-Ai Lane, San Min Chu,
Kaohsiung
Tel: 297319

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

UGANDA
Uganda Tele-Electric Co., Ltd.
P.O. Box 4449
Kampala
Tel: 57279
Cable: COMCO Kampala

VIETNAM
Peninsular Trading Inc.
P.O. Box H-3
216 Hien-Vuong
Saigon
Tel: 20-805, 93398
Cable: PENTRA, SAIGON 242

ZAMBIA
R.J. Tilbury (Zambia) Ltd.
P.O. Box 2792
Lusaka
Zambia, Central Africa
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
Export Trade Company
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 493-1501
TWX: 910-373-1267
Cable: HEWPACK Palo Alto
Telex: 034-8300, 034-8493

11289B ADVANCED PROGRAMMING II OPERATING MANUAL

PART NO. 09830-90019
MICROFICHE NO. 09830-99019

PRINTED IN U.S.A.
MAY 2, 1975