FAST BASIC IV ROM

OCTOBER 1979

INFOTEK SYSTEMS CORPORATION
1400 North Baxter Street
Anaheim, California 92806

TABLE OF CONTENTS

	PAGE
INTRODUCTION	1
CHAPTER 1: GENERAL INFORMATION	2
CHAPTER 2: $SEARCH STATEMENT	2
CHAPTER 3: HUNT STATEMENT	4
CHAPTER 4: SUM STATEMENT	6
CHAPTER 5: ASCALAR STATEMENT	8
CHAPTER 6: ESCALAR STATEMENT	9
CHAPTER 7: ESEND STATEMENT	10
CHAPTER 8: RLOAD STATEMENT	11
CHAPTER 9: FMATEL AND VFMATEL STATEMENTS	13
CHAPTER 10: FMATRC AND VFMATRC STATEMENTS	14
CHAPTER 11: ACS AND ASN FUNCTIONS	15
CHAPTER 12: MAX AND MIN STATEMENTS	16
CHAPTER 13: PAUSE STATEMENT	17
CHAPTER 14: ILIST COMMAND	17
CHAPTER 15: SXREF COMMAND	18
CHAPTER 16: USED COMMAND	19
APPENDIX I: SUMMARY OF FB-IV SYNTAX	20
APPENDIX II: INSTALLATION	21

INTRODUCTION

The Infotek Fast Basic IV ROM continues the expansion of the HP-9830 Basic Language, by adding 17 new statements, functions and commands to those already available through Hewlett-Packard or other Infotek ROMs. Application of this ROM will increase the speed and capability of the 9830 system. The array oriented capabilities are particulary powerful in conjunction with the giant arrays made possible by the Infotek 64K byte MX memory system.

Two new statements greatly increase the convenience and speed of searching arrays. The first of these is a $SEARCH statement which provides a universal means of rapidly searching an integer array for a specified character string without having to re-transfer the array. $SEARCH makes programs which involve searching string arrays easier to write, and much faster in execution. HUNT is a statement which can search a numeric array for a specified relation. Both $SEARCH and HUNT are flexible to the extent that the area of search may be specified.

Another three statements allow faster math operations in matrices. ASCALAR allows one matrix to operate on another, while ESCALAR allows a constant to operate on a matrix and SUM returns the sum of the elements of any part or all of a matrix. Each of these three statements perform these functions substantially faster than an equivalent standard basic program, and with only a single program line.

ESEND is provided for array manipulation. ESEND allows all or part of an array to be transferred to all or part of an array. RLOAD allows any part of an array which is stored on cassette or Infotek floppy disk to be brought into a specified array thereby greatly improving memory utilization and file structure. Both statements operate on an element by element basis with extensive control over the area of operation.

Given an array element number, FMATRC returns the row and column numbers of an array based on the current working dimensions. Similarly, FMATEL returns the element number, given the row and column numbers. The virtual array dimensioning capability of the Infotek Fast Basic III ROM which allows arrays to be greater than 256 on a side is further enhanced by the V (virtul dimension) option for both FMATEL and FMATRC.

MIN and MAX are provided in order to conveniently return the smallest or largest value in a list of expressions. To overcome the 30 second maximum delay time of the WAIT statement, a PAUSE is implemented to suspend program operation indefinitely unless terminated in the same manner as interrupting a wait. For increased conven-

ience and execution speed of programs using trigonometric functions, ACS (arccosine) and ASN (arcsine) are provided.

To assist in program development and debugging, three new commands are implemented. ILIST prints all lines which contain a specified instruction. SXREF prints an alphanumerically sorted cross reference list of all variables used in the program and the line numbers in which they appear, and USED returns the line number in which a specified variable is first used.

CHAPTER 1

----------GENERAL INFORMATION----------

EQUIPMENT SUPPLIED

One Fast Basic IV Instruction Manual (P/N 001-00019) is supplied with the Fast Basic IV ROM (Read Only Memory) PN 900-12611-19 (external) or 900-12436-19 (internal)

INSPECTION

Inspect the ROM for mechanical damage. Contact Infotek or your local Infotek representative if damage is found.

INSTALLATION

Installation of external (plug-in) ROMs is described in Appendix C of the Operating and Programming manual for the 9830 A/B calculators (Hewlett Packard P/N 09830-90001). The FB-IV ROM may be placed in any slot. The ROM will not slide into the slot unless the label is "right side up".

Installation of an internal FB-IV ROM is described in Appendix II.

CHAPTER 2

----------$SEARCH STATEMENT----------

DESCRIPTION

The $SEARCH statement allows an integer array containing transferred character strings to be searched for a match to a specified string. The search is performed character by character. The element number which contains the first character of the match is

returned. Because string data is packed two characters per element, the first or second character within an element can be specified for starting the search and it is also returned as a variable to indicate which character of the element contains the start of the match string. In order to efficiently search a portion of an array, a character increment is provided in the $SEARCH statement. This allows an array to be searched row by row, begining with any desired column. The width of the search in bytes is equal to the match string length.

Use of the $SEARCH statement avoids the TRANSFER of string data to an explicit string. While neither the AP-I or AP-II Roms which contain the transfer statement are necessary for the use of $SEARCH, string data cannot be readily extracted from an integer array without the TRANSFER statement.

The STRING ROM is required for the $SEARCH statement.

SYNTAX:

$SEARCH <array name>,<starting element number>,<starting character>,<character increment>,<match string name [substring]>,<match element return variable>,<match character return variable>

The character specification is 0 for the first character and 1 for the second character packed in an integer array element. The character increment must be a positive integer precision number.

SYNTAX EXAMPLE

```
$SEARCH  A,2,0,20,A$(1,5),E,C
```

An integer precision array, A, is searched starting at the second element, first character for a match with A$[1,5]. The search increments by 20 characters. Thus, if the array had 10 columns, the string search would commence in the first byte of the second column of each row. The number of the element which contains the first character of the target string is returned in E. The character location in the element (E) is returned in C. If no match is found, E is set to 0.

USAGE EXAMPLE

The following program creates an integer array containing a string, then finds and recovers part of the string.

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

```
10 DIM AI[10],B$[20],C$[4]
20 B$="JOHNMARYJACKSAMPAT"
30 TRANSFER B$ TO A[2]
40 C$="MARY"
50 $SEARCH A,2,0,1,C$,E,B
60 TRANSFER A[E] TO B$
70 PRINT B$(B+1,LEN(E$)+B+1);"STARTS IN ELEMENT" E "BYTE" B
80 END

RUN

MARY STARTS IN ELEMENT 4 BYTE 0
```

After the TRANSFER statement in line 30, array A contains:

A		JO	HN	MA	RY	JA	CK	SA	MP	AT
	1	2	3	4	5	6	7	8	9	10

The $SEARCH statement in line 50 searches array A, starting at the first character of the second element and increment by 1 character until a match to the character string C$ (MARY) is found. The return variable E contains a 4 which corresponds to the element containing the characters MA, while the variable B contains a 0 which indicates that the match string starts at the first character in element 4.

The significance of the return byte order variable can be shown by changing line 40 of the $SEARCH example program as follows:

```
40 C$="PAT"
RUN
PAT  STARTS IN ELEMENT 9 BYTE 1
```

The TRANSFER statement in line 60 unpacks data from array A so that B$ contains MPAT. By knowing which byte of the first element contains the starting character, the M can be bypassed as is done in the PRINT statement in line 70, so that only PAT is printed.

CHAPTER 3

----------HUNT STATEMENT----------

DESCRIPTION

The HUNT statement rapidly compares elements of an array against

an expression. The four relational operators less than, greater than, equal to and not equal to are allowed for the comparison. Element increment and element count specifications are also provided in the instruction so that the operation may be limited to specific areas of an array such as columns, rows or diagonals. The user has complete control over the part of an array that is to be searched.

The return variable contains the element number that most nearly satisfies a relation. When LESS THAN is specified, the return variable contains the element number which is closest to, but still less then the match expression. The GREATER THAN relational operator behaves in a similar manner, returning the element number which contains a value closest to the number in the match expression which is still greater than the specified value. If more than one element satisfies the relationship, then the first such element number is returned. When EQUAL or NOT EQUAL is specified, the first element containing the match value is returned. When a specified relationship cannot be found, the return variable is set to zero.

The Infotek Fast Basic I ROM is required for operation of the HUNT statement. Error 1 results if the Fast Basic I ROM is not installed when HUNT is executed.

SYNTAX

```
HUNT  <starting element number>,<element increment>,
      <array name>(<:>:=:#)<match expression>,<match
      element return variable>[,element count]
```

The starting element and increment numbers must be positive integers that are within the current working dimensions of the array.

SYNTAX EXAMPLE

```
HUNT  1,1,A<1E+99,E
```

This statement will return the element number in array A which contains a value closest to and less than 1E+99. Because the first starting point is the first element, the element increment is 1 and an element terminal count is not specified, the entire array will be searched for the highest value less than 1E+99.

USAGE EXAMPLE

The following program finds the largest value in an array.

```
10 DIM A[10]
```

```
20 FOR I=1 TO 10
30 A[I]=I
40 NEXT I
50 HUNT 1,2,A<1E+99,L5,4
60 END
```

Lines 20 through 40 place the numbers 1 through 10 in the corresponding elements of array A. The HUNT statement in line 50 starts at the first element, and increments by 2 elements looking for a value in array A which is less than 1E+99. The element number containing the value which is closest to the specified relationship is returned in L5. The HUNT is terminated after 4 elements are compared. The array element numbers and in this case also the values examined are:

1, 3, 5, 7

In this example, L5 returns a 7 which is the examined value in A closest to, but less than 1E+99. The ninth element is not checked as the HUNT was specified for a termination count of 4 comparisons.

```
HUNT  3,1,A=17,N
```

Operation begins in the third element of array A and compares each element for a value equal to 17. As an element terminal count is not specified, all elements of array A from the specified starting element are compared. Because array A does not contain a 17, the return variable, N, is set to zero.

CHAPTER 4

----------SUM STATEMENT----------

DESCRIPTION

The SUM statement permits simple and rapid summation of all or a specified portion of an array. Because the starting element number, element increment, and the number of elements to be summed are specified, the SUM statement can return the sum of an entire array, columns, rows, or any diagonal.

SYNTAX

SUM <array name (starting row, starting column)>,<element increment>,<sum return variable [,number of elements to be counted]>

The element increment and element count must be positive integers within the current working dimensions of the array.

SYNTAX EXAMPLE;

```
SUM   A(1,2),2,X,10
```

Begining with the second element, every other element is added, with the SUM returned in X because the increment is 2 and the termination count is 10, only the first 10 even numbered elements are summed.

USAGE EXAMPLE

The following program shows how easily the SUM statement can be used to find the SUM of an array.

```
10 DIM A(10,10)
20 MAT A=CON
30 SUM A(1,1),1,S
40 PRINT "THE SUM OF THE ARRAY IS"S
50 END

RUN
THE SUM OF THE ARRAY IS 100
```

Line 20 sets all elements to 1. The SUM statement in line 30 sums array A starting at row 1, column 1 with the element increment set to 1. As an element terminal count is not specified, all elements of the array are added. The return variable S contains the sum.

The following is a continuation of the above program and shows how the SUM statement can be used to quickly and easily find the sum of columns, rows, or a diagonal of the array.

```
 60 SUM A(1,2),10,C
 70 PRINT "THE SUM OF THE SECOND COLUMN IS " C
 80 SUM A(4,1),1,R,10
 90 PRINT "THE SUM OF THE FOURTH ROW IS" R
100 SUM A(1,10),9,D,10
110 PRINT "THE SUM OF THE REVERSE DIAGONAL IS" D
120 END

RUN
THE SUM OF THE SECOND COLUMN IS 10
THE SUM OF THE FOURTH ROW IS 10
THE SUM OF THE REVERSE DIAGONAL IS 10
```

The SUM statement in line 60 starts at row 1, column 2 of array A

and sums every tenth element. As each row contains ten elements, and element increment of ten will always fall in the same column. In this manner all of the elements in column 2 are summed and returned in the variable C. Line 80 contains a SUM statement which starts at row 4, column 1 and has an element increment of 1. Summation is terminated after 10 elements are added so that only the elements in the fourth row are summed.

The flexibility of the SUM statement is further shown in line 100 where the elements of the reverse diagonal (top right hand element to the bottom left hand element) are summed. The summation starts at row 1 column 10, with an element increment of 9. As each row has ten elements, the effect is to move from column 10 to column 9 while the row goes from 1 to 2. This sums the reverse diagonal. The terminal count of ten is necessary to prevent the last nine elements of the array from being included in the summation.

CHAPTER 5

----------ASCALAR STATEMENT----------

DESCRIPTION

The ASCALAR statement allows one array to operate element by element on another array and place the results of the operation in a third array. All three arrays may be the same or may be dissimilar in name, precision, number of elements, dimensions and number of dimensions.

Operation continues until the number of elements in one of the three arrays is exceeded.

SYNTAX EXAMPLE;

```
ASCALAR  <first array name>(+:-:*:/)<second array name>,
         <result array name>
```

USAGE EXAMPLE;

```
ASCALAR  A+B,C
```

Corresponding element of arrays A and B are added and the results are placed in the corresponding elements of array C.

The following program shows how one array can be operated upon by another.

```
10 DIM KI[6,6],LS[50],M[3,10]
20 MAT K=IDN
30 MAT L=CON
40 ASCALAR K+L,M
50 MAT PRINT M;
60 END

2  1  1  1  1  1  1  2  1  1
1  1  1  1  2  1  1  1  1  1
1  2  1  1  1  1  1  1  2  1
```

Arrays K and L are set to all ones. ASCALAR adds them and places the results in array M. Note that the arrays are dissimilar in all respects.

CHAPTER 6

----------ESCALAR STATEMENT----------

DESCRIPTION

The ESCALAR statement allows an array to be operated upon by an expression on an element by element basis, and the results placed in the same or a second array. The two arrays may be dissimilar in precision, number of elements and dimensions. Operation continues until the number of elements in one of the two arrays is exceeded.

SYNTAX EXAMPLE

ESCALAR <source array name>(+:-:*:/)<expression>,<destination array name>

USAGE EXAMPLE

```
ESCALAR  A*2,B
```

Each element in array A is multiplied by 2 and the product is placed in the corresponding element in array B.

1. In the following program, line 20 sets array A to all ones while line 30 causes each element of A to be multiplied by PI and the product stored in the corresponding element of array B.

```
10 DIM AI(3,2),B(1,5)
20 MAT A=CON
```

```
30 ESCALAR A*PI,B
40 FIXED 3
50 MAT PRINT B;
60 END

RUN

3.142   3.142    3.142    3.142    3.142
```

CHAPTER 7

----------ESEND STATEMENT----------

DESCRIPTION

The ESEND statement is used to move specific elements from one array to another. Separate increment specifications are provided for the source and destination arrays thereby allowing the transfer of any part of one array to any part of another array. For example the diagonal elements of one array could be moved to a row, column or diagonal of another array. Unlike send, transfer always begins with the first element.

The source and destination arrays need not be of the same precision nor have the same dimensions or number of elements. The ESEND statement continues until the next transfer would exceed the bounds of either the source or destination array.

SYNTAX EXAMPLE

ESEND <source array name (starting row, starting column)>,<source element increment>TO<destination array name (starting row, starting column)>,<destination element increment>

The starting point array subscripts must be within the current working dimensions of the array.

USAGE EXAMPLE

```
ESEND  A(1,1),2 TO B(2,2),1
```

Data is moved from array A to array B beginning with the first element of A which is placed in B(2,2). The source increment is 2 and the destination increment is 1. Every other element of array A is moved to fill consecutive elements of array B.

The following program uses ESEND to move the diagonal of array A to the reverse diagonal of array B.

```
10 DIM A(5,5),BI(5,5)
20 MAT A=CON
30 MAT B=ZER
40 ESEND A(1,1),6 TO B(1,5),4
50 MAT PRINT B;
60 END

RUN

 0  0  0  0  1
 0  0  0  1  0
 0  0  1  0  0

 0  1  0  0  0
 1  0  0  0  0
```

Lines 20 and 30 set up arrays A and B. The ESEND statement in line 40 moves every sixth element of array A starting at row 1 column 1 to every fourth element in array B starting at row 1 column 5. ESEND moves the left to right diagonal of array A to the right to left diagonal of array B.

CHAPTER 8

----------RLOAD STATEMENT----------

DESCRIPTION

The RLOAD statement is used to transfer all or part of an array stored either on the 9830 cassette or an Infotek floppy disk system into a specified array. The starting location and increment are specified, thereby allowing any part of the stored array to be loaded and placed into a smaller array. For example a column of a 100X100 stored array can be moved to a 100 element vector without having to load the entire 10,000 element array from the magnetic media.

The stored array and the destination array need not have the same dimensions or number of elements. The RLOAD statement continues until the next transfer would exceed the bounds of either the stored array or the destination array. Both the stored and destination arrays must be of the same precision.

SYNTAX EXAMPLE

```
RLOAD  [# device select code,]<file number>,<destina-
       tion array name>,<stored array starting element>,
       number>,<stored array element increment>
```

The starting element and increment numbers must be positive integers within the dimensions of the stored array.

USAGE EXAMPLE

```
RLOAD  1,A,1,3
```

File 1 on the internal cassette, starting with the first element, and continuing with every third element is loaded from the tape into array A.

In the following program, a data array is generated and stored on the internal cassette. After being stored, part of the data are recovered and placed into a smaller array using the RLOAD statement. Before running this program, place a rewound and unprotected "scratch" tape in the internal cassette.

```
 10 DIM A(10,10),B(1,10).
 20 FOR I=1 TO 10
 30 FOR J=1 TO 10
 40 A(I,J)=(I-1)*10+J
 50 NEXT J
 60 NEXT I
 70 MAT PRINT A;
 80 PRINT
 90 MARK 1,400
100 STORE DATA 0,A
110 RLOAD 0,B,61,1
120 MAT PRINT B;
130 END
```

Lines 10 thru 80 loads array A with the numbers 1 thru 100 and prints the array. Lines 90 and 100 mark the tape and store array A in file 0. Line 110 moves array A elements, starting with 61 and incrementing by 1 to array B. Element 61 in array A is the first column of row 6. Beacuse the increment is 1 and array B has 10 elements only, the sixth row of array A will fill array B. Line 120 prints array B.

```
RUN

 61   62   63   64   65   66   67   68   69   70
```

CHAPTER 9

----------FMATEL AND VFMATEL STATEMENTS----------

DESCRIPTION

The FMATEL statement converts row and column values for a specified array to the corresponding array element number based on the current working dimensions of that array. The VFMATEL statement similarly converts virturally dimensioned array row and column values. Virtual dimensioning circumvents the 9830 intrinsic limitation of 256 for array dimensions.

Virtually dimensioned arrays are described in the Infotek Fast Basic III ROM Instruction manual. The VFMATEL option requires the Infotek Fast Basic III ROM and the H-P Matrix ROM.

SYNTAX EXAMPLE

```
[V]FMATEL  <array name>,<row number>,<column number>,<element
           number>
```

USAGE EXAMPLE

```
FMATEL     A,1,2,X
```

The element number in array A for row 1, column 2 is returned in X. The row and column numbers for both FMATEL AND VFMATEL must be positive integers within the current working dimensions of the array.

```
 10 DIM A(30,30)
  .
  .
  .
100 MAT A=CON
110 REDIM A(17,23)
120 A(11,17)=0
130 FMATEL A,11,17,E
140 PRINT"ARRAY A ELEMENT"E"IS ZERO"
150 END
```

A 900 element 30X30 array is set to all ones and redimensioned to a 391 element 17X23 array. Line 120 sets A(11,17) to a zero and line 130 returns the element number based on the current working dimensions.

The following program example demonstrates the convenience of VFMATEL.

```
10 DIM AI(250,20)
20 MAT A=ZER
30 VDIM A(2500,2)
40 ARRAY A(2500,2),X,-1
50 VSEARCH A,C,2,-1,X
60 VFMATEL A,X,2,E
70 PRINT"A MINUS ONE IS IN ELEMENT"E
80 END

RUN

A MINUS ONE IS IN ELEMENT 5000
```

Lines 10 and 20 initialize an array and set all elements to zero. Line 30 gives this 5000 element array VIRTUAL dimensions of 2500 rows and 2 columns. Line 40 updates row 2500, column 2 to A minus one and line 50 performs a search of column 2 for an element containing A minus one. The operations of lines 30, 40 and 50 are described in detail in the Infotek FAST BASIC III Manual. Line 50 returns the row number in column 2 that contains A minus one in X. Line 60 converts row X, column 2 to an element number based on the virtual dimensions of 2500 rows and 2 columns.

CHAPTER 10

----------FMATRC AND VFMATRC STATEMENTS----------

DESCRIPTION

The FMATRC statement converts the element number of a specified array into the corresponding row and column values based on the current working dimensions of that array. The VFMATRC statement simiarly converts the element number of a virtually dimensioned array into the corresponding virtual row and column values. Virtual dimensioned arrays are described in the Infotek Fast Basic III ROM Instruction Manual.

SYNTAX

```
[V]FMATRC <array name>,<element number>,<row return variable>,
          <column return variable>
```

USAGE EXAMPLE

```
FMATRC A,1,R,C
```

Both R and C are set to 1 because A(1,1) corresponds to the first element of an array. The element number must be a positive integer within the current working dimensions of the array.

The following program shows how conveniently the row and column numbers corresponding to a given element number can be determined.

```
 10 DIM A(1,250)
  .
  .
  .
100 A(1,190)=1
110 REDIM A(5,50)
120 FMATRC A,190,R,C
130 PRINT"ELEMENT 190 IS NOW AT ROW"R"COLUMN"C
140 END
```

Line 100 sets element 190 of A 1X250 array to a 1. Line 110 redimensions the array to 5X50. Line 120 converts the row and column numbers for the 190th element based on the new working dimensions of 5X50. This would be a far more cumbersome operation without the FMATRC statement.

CHAPTER 11

----------ACS and ASN FUNCTIONS----------

DESCRIPTION

The ACS and ASN functions provide more convenient means of dealing with arcsine and arccosine. The ACS function returns the arccosine, while the ASN function returns the arcsine. Both operate in the current trigonometic mode of the 9830.

SYNTAX EXAMPLES

```
ACS (argument)

ASN (argument)
```

The argument need not be enclosed by parentheses although it is a good practice. Error 4 results if the argument is outside the range of -1 to +1.

USAGE EXAMPLES

```
PRINT-ALL
DEG
ASN(.5)
30
ACS(.5)
60
```

CHAPTER 12

----------MAX AND MIN STATEMENTS----------

DESCRIPTION

The MAX statement returns the maximum value of an expression or variable in a list of expressions or variables. The MIN statement similarly returns the minimum value.

SYNTAX EXAMPLE

```
MAX <return variable>,<expression[,expression]>

MIN <return variable>,<expression[,expression]>
```

The MAX and MIN statements will operate on as many expressions, variables or constants as may be placed within 80 characters.

USAGE EXAMPLE

```
10 A=1
20 B=2
30 MAX L,6,A,9,B,B*2+A,B+A
40 PRINT"THE MAX VALUE OF LINE 30 IS"L
50 END

RUN

THE MAX VALUE OF LINE 30 IS 9
```

The maximum value of the list of constants, variables and expressions appearing in line 30 is returned in L. By adding lines 50 and 60 to the previous program as follows:

```
50 MIN S,6,A,9,B,B*2+A,B+A
60 PRINT"AND THE MINIMUM IS"S
```

```
     CONTINUE 50

     AND THE MINIMUM IS 1
```

We get the minimum value of the list.

CHAPTER 13

----------PAUSE STATEMENT----------

DESCRIPTION

The PAUSE statement allows a program to suspend operation indefinitely until an operator presses any key on the calculator except SHIFT and REWIND (which have no effect). If STOP is pressed program execution is terminated. The PAUSE statement functions similarly to a WAIT statement except that the maximum time available from a WAIT statement is about 30 seconds. The PAUSE statement is not allowed in keyboard mode.

SYNTAX EXAMPLE

```
     PAUSE
```

EXAMPLE

The following program shows how the PAUSE statement will hold a display until the operator can take action.

```
10 DISP "THE CALCULATOR AWAITS"
20 PAUSE
30 DISP "WELCOME BACK"
40 END
```

CHAPTER 14

----------ILIST COMMAND----------

DESCRIPTION

The ILIST command causes all program lines which contain a specified instruction to be listed in line number order. The listing may be produced on any select code. If a select code is not specified, the command defaults to select code 15.

The ILIST command is permitted in the keyboard mode only.

SYNTAX EXAMPLE

. ILIST [#select code,]<statement or function name>

USAGE EXAMPLE

A typical application would be to find all RETURN statements in a program. For demonstration purposes load any existing program.

ILIST RETURN

or, find all conditional branches;

ILIST IF

Should an ILIST be desired on a printer, select code 2, then the above instructions should be:

ILIST #2, RETURN

ILIST #2, IF

CHAPTER 15

----------SXREF COMMAND----------

DESCRIPTION

The SXREF command provides an alphanumerically sorted list of variables and all line numbers in which each appears. If no device is specified, the command defaults to select code 15.

The SXREF command is permitted in keyboard mode only.

The Infotek Fast Basic II ROM is required for the SXREF statement.

SYNTAX EXAMPLE

SXREF [#select code]

USAGE EXAMPLE

TYPE IN THE FOLLOWING PROGRAM

100 X=C=A=0

```
120 J=4
130 A(B,C)=A+D

SXREF

A()   130
A     100    130
B     130
C     100    130
D     130
J     120
X     100
```

Note: Caution should be exercised in using the SXREF command as it will re-initialize all of the variables.

CHAPTER 16

----------USED COMMAND----------

DESCRIPTION

The USED command provides a very convenient means of determing if a simple variable already exists in a program. USED will display the first line number in which the variable is used. Unlike a cross reference listing, USED does not reinitialize the program and is considerably faster as a printer is not involved.

The USED command is permitted in keyboard mode only.

SYNTAX EXAMPLE

```
USED <simple variable>
```

EXAMPLE

Enter the following program into the 9830:

```
10 X9=10
20 A1=4
30 Y=1.75
40 END
USED X9
10
USED Y
30
```

```
USED B
0
```

Entering USED X would return (line) 10 in the display. USED Y would return a 30, and USED B would display 0 as the simple variable B is not used in the program.

APPENDIX I

----------SUMMARY OF FB-IV SYNTAX----------

STATEMENT, COMMAND, OR FUNCTION	CHAP.
$SEARCH <array name>,<starting element number>,<starting character>,<element increment>,<match string name [substring]>,<match element return variable>,<match character return variable>	2
ACS <(argument)>	11
ASN <(argument)>	11
ASCALAR <first array name>(+,-,*,/)<second array name>,<destination array name>	5
ESCALAR <source array name>(+,-,*,/)<expression>, destination array name>	6
ESEND <source array name(starting row, starting column)>,<source element increment>TO<destination array name(starting row,starting column)> destination element increment	7
FMATEL <array name>,<row number>,<column number>,<element return variable>	9
FMATRC <array name>,<element number>,<row return variable>,<column return variable>	10
HUNT <starting element number>,<element increment>,<array name>(: :=:#)<match expression>,<match element return variable[,element terminal count]>	3

ILIST [#SELECT CODE,]<statement or function name> 14

MAX <return variable>,<expression[,expression] 12

MIN <return variable>,<expression,[,expression] 12

PAUSE 13

RLOAD [#input device select code,]<stored data file number>,<destination array name>,<stored array starting element number>,<stored array element increment> 8

SUM <array name(first row, first column)>,<element increment>,<summation return variable>[,number of elements to be counted] 4

SXREF [#select code] 15

APPENDIX II

----------INSTALLATION----------

Installing the internal ROM:

1. Place input power switch in the OFF position.

2. Remove the input power cord from the wall outlet and the power jack at the rear of the HP 9830A/B.

3. Lift the thermal printer from the computer (if so equipped) and place to one side.

4. Remove the six screws from the top cover of the computer.

5. Slide the top cover back about two-thirds of the way by using the plastic handles at the back of the cover.

6. Remove the single screw that retains the two crossed aluminum hold-down brackets. Note the location of the brackets and how they are attached. Remove the brackets.

7. The first three card positions behind the front panel along the left side of the computer are reserved for internal optional ROM's. Locate any ROM slot from among the three specified. The slot will have a black card guide on the left side and a red guide on the right side.

8. Position the circuit card over the card guide with the component side of the card facing toward the rear of the computer. Confirm that the guide and handle colors match.

9. Carefully lower the circuit card down the guides and into the connector well until contact is made with the connector. Be certain that the edges of the card are within the edges of the connector well.

10. Apply even pressure with the thumbs to the top of the handles to seat the circuit card in the connector. The card is fully seated when the top is approximately even with the cards in from or in back.

11. Replace the two aluminum hold-down brackets and secure with one screw.

12. Slide the cover forward and secure with the six screws.

13. Replace the thermal printer.

14. Verify that the input switch is in the OFF position.

15. Connect the power cord to the computer and the wall outlet. This completes the installation.