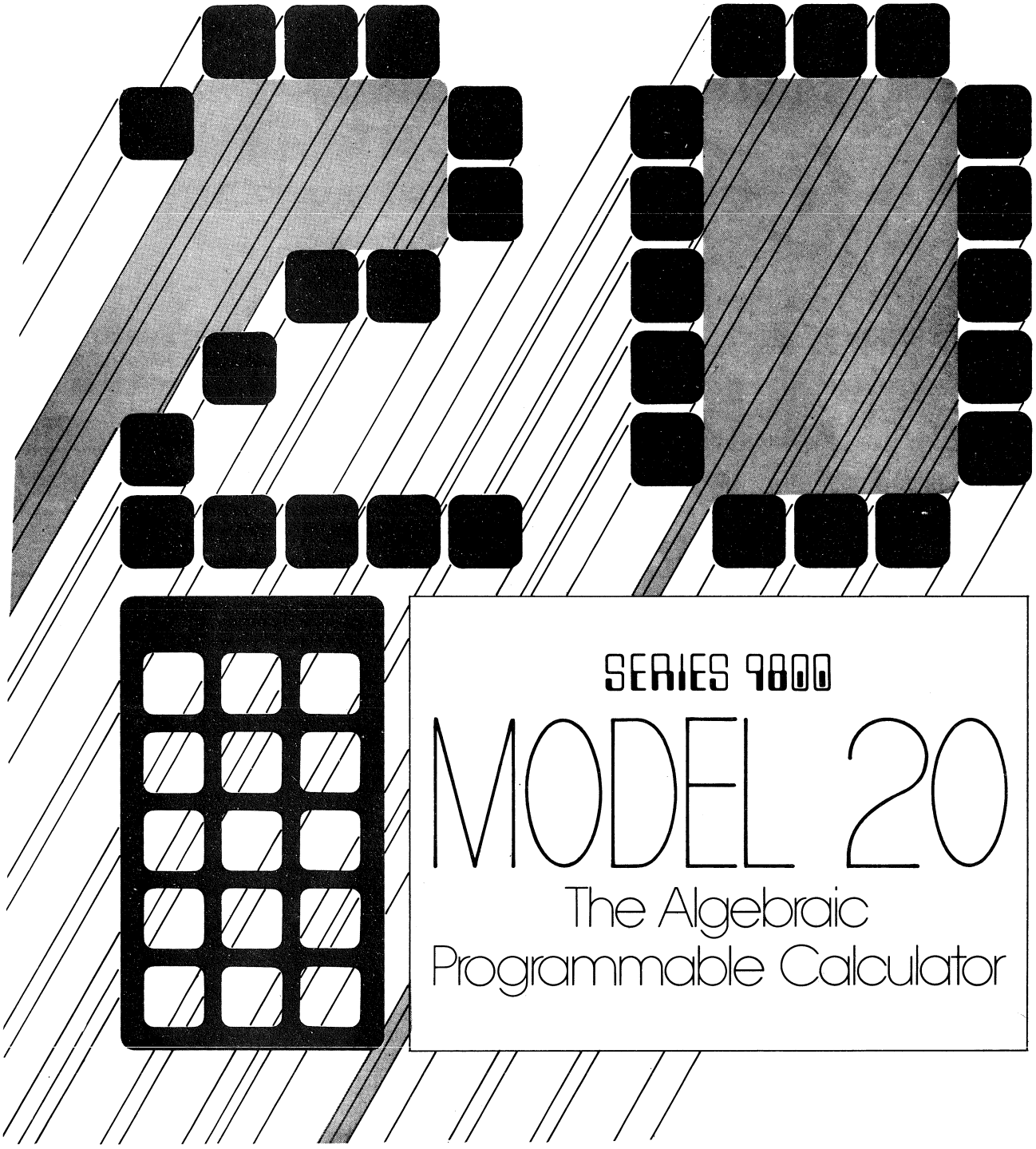
HEWLETT hp PACKARD

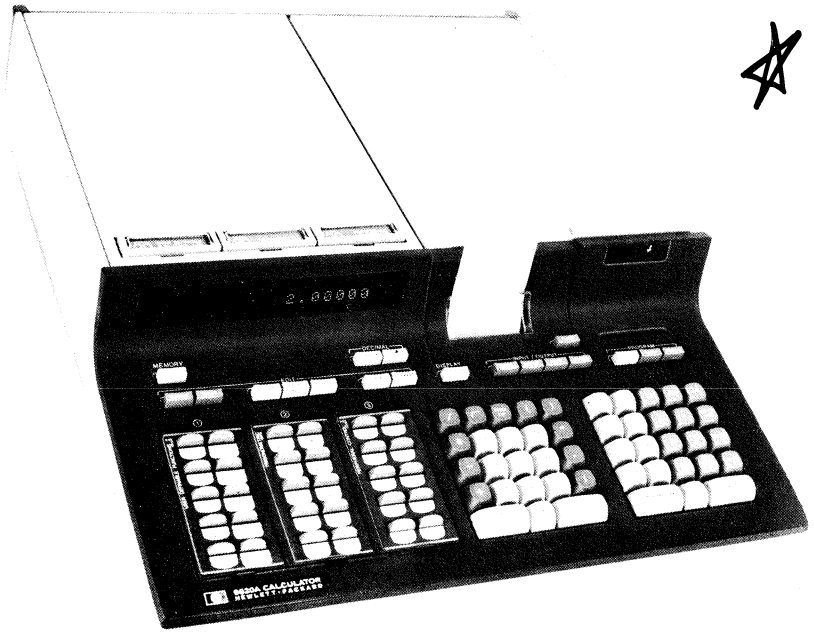


SERIES 9800

MODEL 20

The Algebraic Programmable Calculator

THE HP MODEL 9820A SYSTEM



* ALGEBRAIC LANGUAGE - EASY TO USE, NATURAL
* SIMPLE PROGRAMMING - EDITING, AND AUTOMATIC ERROR DETECTION.
* THREE SPECIAL FUNCTION BLOCKS - USER DEFINABLE, PERIPHERALS ETC.
* LARGE DISPLAY AND PRINTER - ALPHA-NUMERIC
* EXPANDABILITY - BOTH PERIPHERAL AND MEMORY
* CARD READER - BUILT IN MAG. CARD READER.

MEMORY

* BASIC MACHINE — CONTAINS 179 REGISTERS

* OPTIONS — OPT. 001 AN ADDITIONAL 256 REGISTERS.
— OPT. 002 TOTAL 1453 REG'S

* ROM'S — 11220A PERIPHERAL CONTROL I
11221A MATHEMATICS
11222A USER DEFINABLE
11223A CASSETTE/SPECIAL PROGRAMS
11224A PERIPHERAL CONTROL II

* THERE ARE FOUR TYPES OF MEMORY USED IN THE 9820A AND ARE AS FOLLOWS:-

- INTERNAL ROM — DEFINES THE KEYS AND CALCULATOR MODE OF OPERATION.
- INTERNAL RWM — MODEL 9820A USES THIS AREA FOR ITS OWN HOUSEKEEPING.
- PLUG IN ROM'S — FOR DEFINING LEFT HAND KEY BLOCKS
- USER'S RWM — MAIN AREA OF CALC. AND IS MADE UP OF 'R REGISTERS' PLUS 6 ALPHA REGISTERS.

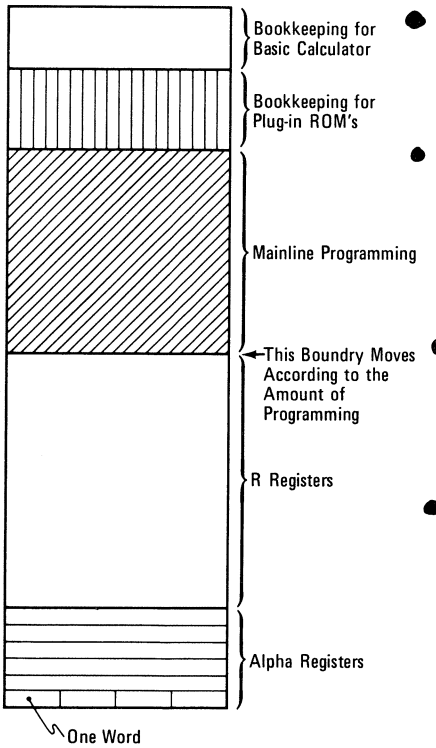
USER'S RWM MEMORY

* **CAPACITY** – THE SIZE OF USERS RWM (IN REG'S) DEPENDS ON ROMS USED - SOME ROMS TAKE UP A SMALL AMOUNT OF AVAILABLE MEMORY.

* **DATA STORAGE** - A REGISTER IS A UNIT OF MEMORY WHICH CAN CONTAIN UP TO 12 SIGNIFICANT DIGITS OF A NUMBER WITHIN THE FOLLOWING

RANGE:- $-9.999999999999 \times 10^{99}$ THRO' TO $9.999999999999 \times 10^{99}$

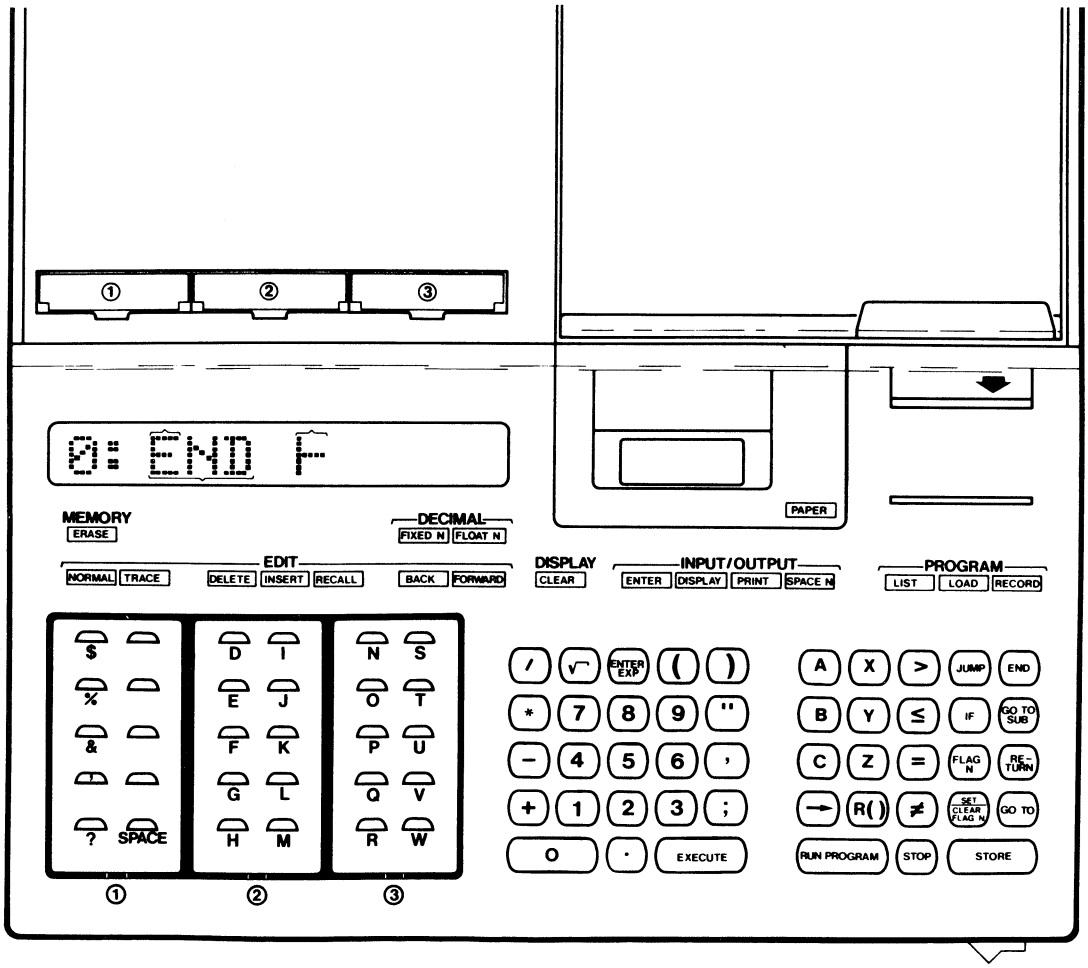
* **REGISTERS** – THE 9820A HAS SIX ALPHA REG'S NAMED A.B.C.X.Y.Z. WITH KEYS. THESE REGISTERS ARE ALWAYS AVAILABLE.



- THE REMAINING MEMORY CONSISTS OF 'R REGISTERS' 'R0'. R1. R2 ETC.
- PROGRAM STORAGE IS IN THE FORM OF LINES AND REGISTERS USED PER LINE DEPENDS ON LINE LENGTH.
- AS A GENERAL RULE WHEN PROGRAMMING EACH 8 KEY STROKES USES ONE REGISTER.
- THE MEMORY ACTUALLY COMPRISES WORDS AND EACH FOUR WORDS MAKE UP ONE REGISTER

THE MODEL 20 MEMORY

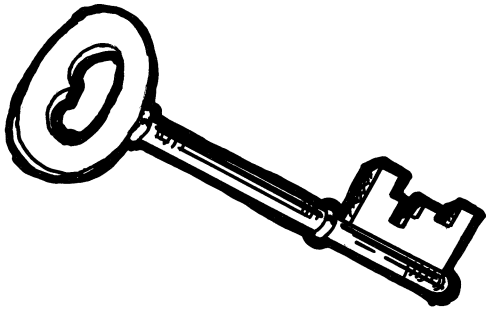
KEYBOARD AND KEYS



* RIGHT HAND KEY BLOCK - INCLUDE MEMORY ACCESS KEYS AND PROGRAMMING KEYS.

* CENTRE KEY BLOCK - NUMERIC AND DATA ENTRY KEYS + PUNCTUATION

* LEFT HAND KEY BLOCKS - ALPHA KEYS (MUST BE WITHIN " ") AND DEFINABLE KEYS WITH ROMS



THE SOLUTION OF THE PROBLEM LIES IN THE KEYS!

DISPLAY

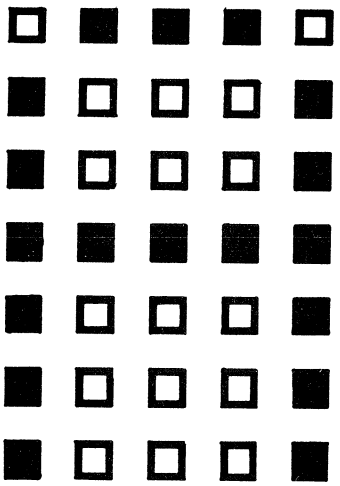
```
X→Y;PRT (A+B)/A;
```

1. Man-Machine (Model 20) Interface

User ⇔ IFA = 12;GTO5 ⇔ 9820A

User — Display — Calculator

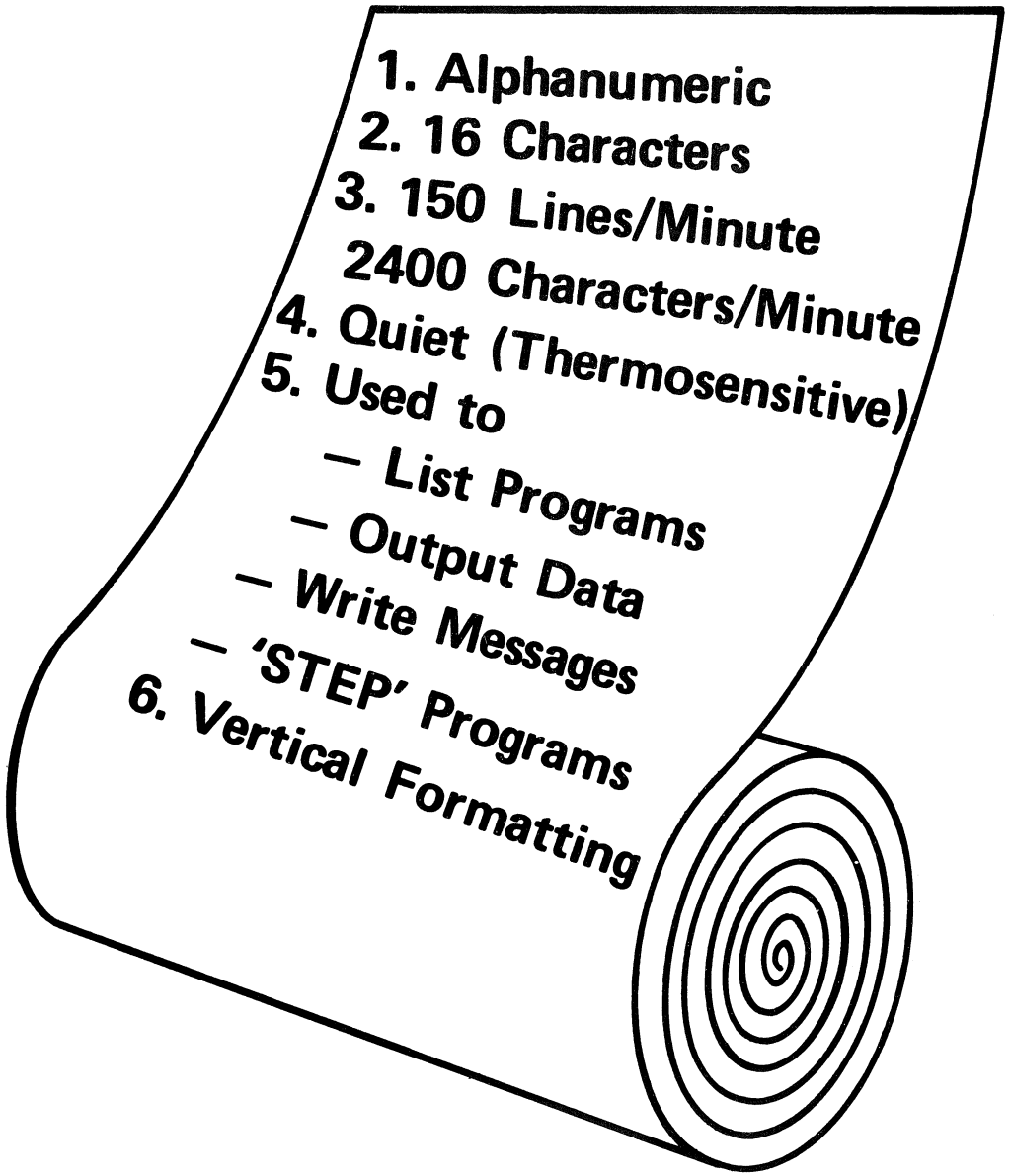
2. Alphanumeric
3. 16 Characters
4. 5 x 7 Matrix of LED's



* THE DISPLAY CAN BE USED TO :-

- INDICATE MODEL 20's LANGUAGE
- DISPLAY RESULTS OF CALCULATIONS
- DISPLAY MESSAGES TO THE OPERATOR
- DISPLAY DIAGNOSTIC INFORMATION IN AUTOMATIC ERROR DETECTION.

PRINTER



* FLEXIBLE - PROGRAMMABLE HARD COPY

PRODUCED BY THE STANDARD, BUILT IN 9820A PRINTER!

```
0:
FXD 3;ENT A,B,C,
X⊢
1:
DSP AA/B+C/XX→Y⊢
2:
JMP -2⊢
3:
END ⊢
```

THE LANGUAGE OF MODEL 9820A

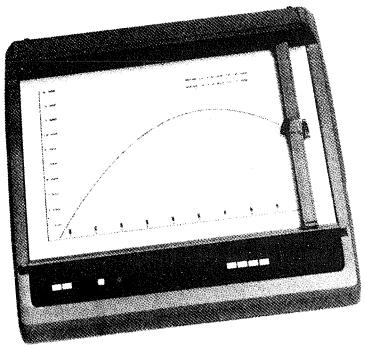
* **ALGEBRAIC** — LANGUAGE STRONGLY RESEMBLES CONVENTIONAL MATHEMATICS.
  - UNIT OF COMMUNICATION IS REFERRED TO AS A LINE
  - RULES FOR COMBINING ELEMENTS IS REFERRED TO AS SYNTAX.

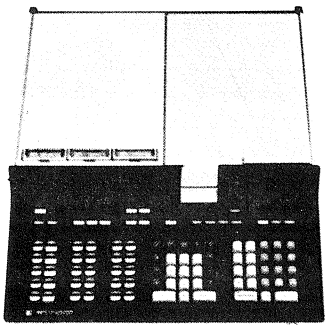
NOTE! ALTHOUGH THE 9820A IS SAID TO USE AN 'ALGEBRAIC LANGUAGE' IT CANNOT LITERALLY MANIPULATE UNKNOWN QUANTITIES IN THE ALGEBRAIC SENSE BUT MANIPULATES NUMBERS IN THE EVALUATION OF AN ALGEBRAIC EXPRESSION!

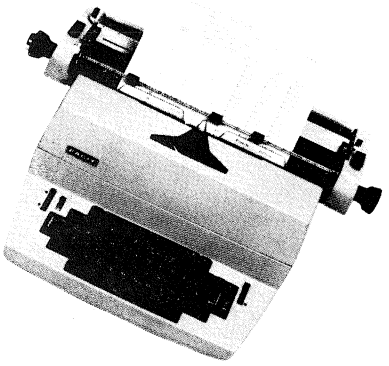
* **COMPUTATION** — EVALUATION OF AN EXPRESSION IS CARRIED OUT IN IT'S ENTIRETY AS A LINE AND NOT AT INDIVIDUAL STEPS.

General Points To Note

* Memory Erase – initializes total memory
* Clear – clears display does not affect memory
* Language – Algebraic
* Line – Fundamental unit of communication
* Statement – Smallest complete unit of language
* Display – used to inform activity etc.
* Line Number – each line must have a line number L.N^s. are consecutive.
* RWM – The RWM is volatile.
* Storage – The unit of Storage is a Register
* Register – can contain approx. 8 keystrokes of a program.
* Printer – 16 characters / line
* ROM's – left hand key blocks can be defined by ROM
* Keys – Full Alpha-Numeric capacity.







How to Begin

* **Initializing** ——— Before using the calc. A wise precaution is to press memory erase. This does the same as turn off as follows :-

  - Clears all user RWM.
  - Clears all flags.
  - Establishes Float 9.
  - Specifies 'Normal' mode.
  - Checks which Roms installed.
  - Performs initialization for any special Roms e.g. Deg. etc.

* **Writing Lines** ——— If in doubt before writing a new line press clear.

  - Enter lines by pressing the keys required.
  - Each line is syntax checked and occurrence of notes in the display indicate an error.
  - Only the last 13 - 16 characters are visible in display at any one time.

Okay Let's Go!

WRITING A LINE

* TRY THE FOLLOWING EXAMPLE :-

TAKE THE LINE :

```
PRT 3,4,√(3*3+4*4)
```

AND BEGIN TO KEY IT INTO THE CALC.

PRESS: [CLEAR]

PRESS: [PRINT] [3] [,] [4] [,] [√] [(] [3] [*] [3] [+] [4] [*], but don't press [4] [)] yet.

* THE DISPLAY NOW SHOWS :-

```
PRT 3,4,√(3*3+4*
```

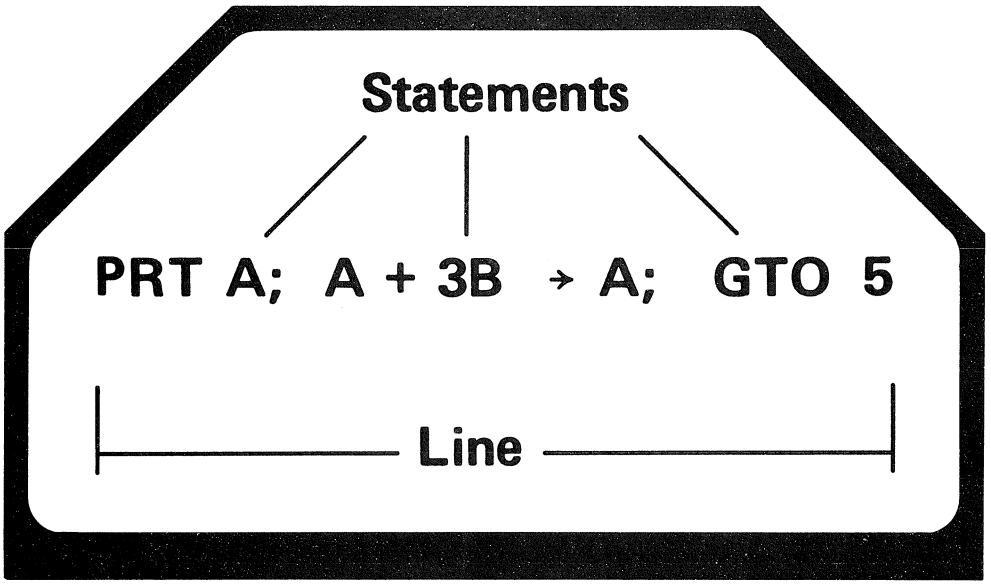
THE DISPLAY IS NOW FULL - WHEN KEYING IN THE NEXT 4 AND) THE CONTENTS OF THE DISPLAY WILL SHIFT LEFT :-

```
3,4,√(3*3+4*4)
```

IN THE FINAL DISPLAY NOTE THAT THERE WAS A SHIFT LEFT OF FOUR CHARACTERS, DUE TO THE FACT THAT THE LEFT MOST ITEM WAS A PRTb. (A MNEMONIC IS NEVER SPLIT IN THE DISPLAY, IT IS EITHER THERE OR NOT THERE).

TRY KEYING IN A FEW OF YOUR OWN !

LINES



Maximum length of a line: between 35 and 68 keystrokes

Note

There is no one-to-one relationship between registers and keystrokes, statements, or lines

* **MULTI-STATEMENT LINES** – MORE THAN ONE STATEMENT IS ALLOWABLE IN A LINE BUT SEMI-COLON'S MUST BE USED TO SEPARATE THEM.

  e.g. `10→A:20→B:30→C`

* **MAXIMUM LINE LENGTH** – NOT FIXED BUT DEPENDS ON WHICH KEYS PRESSED. 'NOTE 9' WILL INDICATE THAT A LINE IS TOO LONG – LINE MUST BE SHORT-ENED BEFORE IT IS ACCEPTED. (KEYSTROKES PER LINE 35-68)

* **LINE MODIFICATION** – CORRECTIONS, INSERTIONS ETC. CAN BE EFFECTED BY USING THE KEYS RELEVANT TO THE DESIRED OPERATION AS FOLLOWS :-

  'BACK' – POSITIONS THE MODIFICATION POINT.
  'INSERT' – OPENS SPACE IN LINE
  'DELETE' – DELETES CHARACTERS/FUNCTIONS
  'FORWARD' – BRINGS BACK THE REMAINDER OF THE LINE

NOTE! USING 'FORWARD' THE RIGHT-MOST PART OF THE LINE MUST BE IN THE DISPLAY BEFORE STORING OR EXEC.

TRY MODIFYING ONE OF YOUR LINES!

SYNTAX ERRORS

* NOTES - DURING LINE WRITING ENTRIES ARE CHECKED FOR LANGUAGE ERRORS A WRONG COMBINATION FOR EXAMPLE * / TOGETHER WILL PRODUCE AN ERROR, (SEE 'NOTES') AND A NOTE WILL BE DISPLAYED.

INDICATION	MEANING
NOTE 01	Most syntax errors. Improper key while in Enter Mode.
NOTE 02	An instruction is followed by a parameter of the wrong type, illogical value, or, a parameter is missing. If flag 14 is not set: Square root of a quantity whose value is negative.
NOTE 03	Extra (or missing).
NOTE 04	Extra) or missing (.
NOTE 05	Parameter out of range when specifying an R register, i.e., too large, or negative.
NOTE 06	Attempt to store a number into something other than a register.
NOTE 07	A RET has not been preceded by a matching GSB.
NOTE 08	Improper branching statement.
NOTE 09	Attempt to use a line that is too long. Subroutines nested too deeply.
NOTE 10	A computation has resulted in an intermediate or final numerical result that is outside the range of the Calculator. NOTE 10 will occur only if flag 14 is not set.
NOTE 11	Pressing a key of a left-hand keyblock while an associated plug-in ROM is not installed, unless the key is used within a quote field. Attempt to execute an ENTER statement from the keyboard.
NOTE 12	Attempt to store a line when there is insufficient memory to accommodate it. No GTO or GSB preceding LOD when loading information under control of an existing program.
NOTE 13	Attempt to record on a protected side of a magnetic card.
NOTE 14	Magnetic card reader operation is not completed; press EXECUTE and insert the next side.
NOTE 15	A program is being loaded from a magnetic card while the three left hand keyblocks are not defined in the same way as when the card was recorded.
NOTE 16	Attempted printer operation while the printer is out of paper.

STORING PROGRAMS

* **PROGRAMS** - ARE STORED IN CALCULATOR MEMORY IN POSITIONS CONTROLLED BY THE PROGRAM LINE COUNTER.
  - THE STATEMENT "GTO (N)" WILL SET THE LINE COUNTER TO THE REQUIRED LOCATION.
  - THE LINE MUST EXIST BEFORE USING THE GTO OR IT MUST BE THE NEXT HIGHEST LINE NO. IN MEMORY.

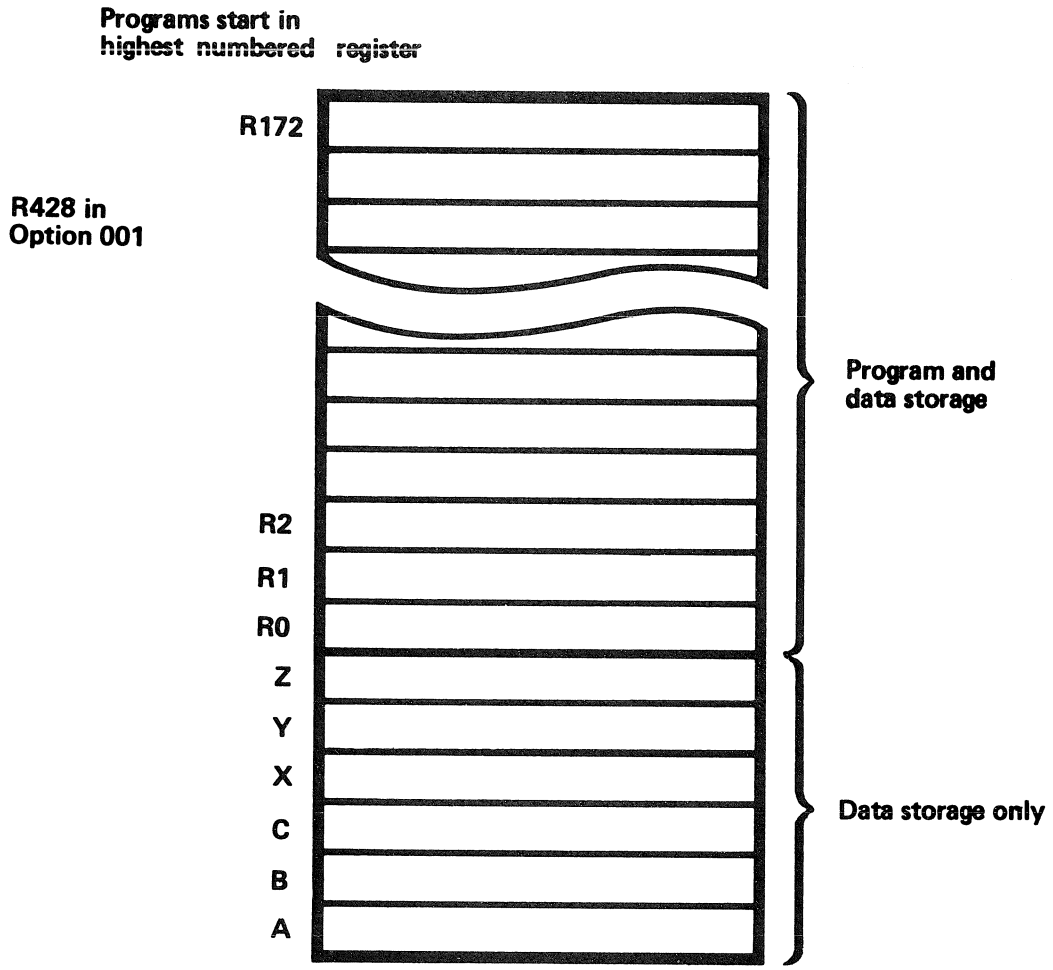
* **IN STORING LINES** THEY MUST BE ENTERED IN SEQUENCE.

  NOTE! WHEN END IS STORED ALL LINES WITH HIGHER NUMBERS ARE ERASED AND MEMORY RETURNS TO 'R' REGISTERS.

* **PROGRAM STACKING** – THE CALCULATOR MEMORY CAN BE LOADED WITH SEVERAL PROGRAMS AT THE SAME TIME – VERY USEFUL WHEN COMBINED WITH SYMBOLIC OR RELATIVE ADDRESSING.

* **LISTING PROGRAMS** - A PRINTOUT OF PROGRAM MEMORY CAN BE EFFECTED BY EXECUTING " END LIST"
  THE LAST ITEM LISTED IS THE REMAINING 'R REGISTERS'

The Memory Map



1 Register can contain

- **one data number of up to 12 digits**
- **or approximately 8 keystrokes**

EDITING PROGRAMS

* **RECALLING A LINE** - TO BRING A STORED LINE BACK TO DISPLAY FOR EDITING, THE FOLLOWING SHOULD BE EXECUTED :-

GTO (n) RECALL

WHERE n IS THE LINE Nº.

- MODIFICATION CAN NOW BE MADE AS BEFORE. THE FORWARD KEY IS THEN USED TO BRING BACK THE RIGHT MOST PART OF THE LINE.
- IF THE 'F' IS NOT THEN IN DISPLAY THE LINE MUST BE RESTORED BY PRESSING 'STORE'.
- AUTOMATIC HOUSEKEEPING IS THEN CARRIED OUT BY CALC. IF THE MODIFIED LINE IS LONGER OR SHORTER THAN THE ORIGINAL.

'Tell' the Calculator to DO it, that is to perform the operation by pressing the key 'Execute'.

Execute

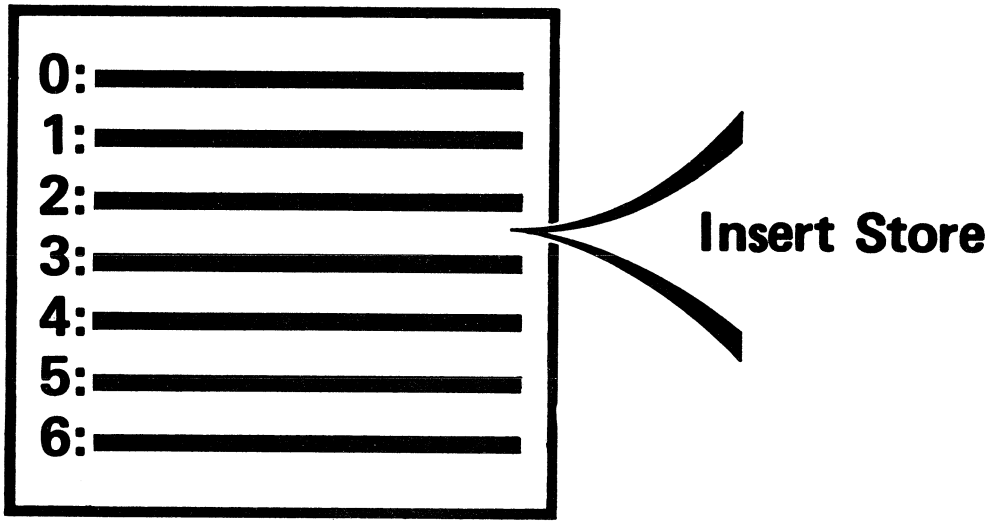
* Inserting and Deleting Lines

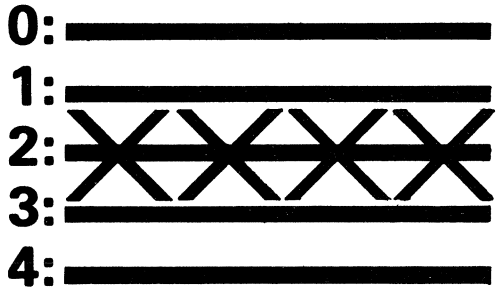
— To insert a line. Set the program line counter to the required No. - Write the new line and then execute "INSERT STORE"

- To delete a line quite simply bring the line into display and execute 'DELETE'

Note! Adjustment of R registers and renumbering of lines is carried out automatically by the calculator.

- Remember to check references to line Nos in program after inserting or deleting.





Delete

RUNNING A PROGRAM

* 'END RUN PROGRAM' - THIS STATEMENT STARTS THE PROGRAM FROM LINE ZERO.

  - IF PROGRAM IS TO BE STARTED FROM A LINE OTHER THAN 0, (MORE THAN ONE PROGRAM IN MEMORY) THEN THE LINE COUNTER MUST BE SET e.g.

    END EXECUTE GTO (n) RUN PROGRAM.

* ERRORS - NOT ALL ERRORS ARE DETECTED WHEN A LINE IS WRITTEN. IF AN ERROR IS DETECTED DURING RUNNING A PROGRAM, THE PROGRAM WILL STOP AND DISPLAY:-

  "NOTE n IN m"

  WHERE n IS THE NOTE N°.
  " m " " LINE N°.

* NORMAL / TRACE - THESE TWO MODES ALLOW HARD COPY OF RESULTS AND OPERATIONS, ESPECIALLY USEFUL IN i.e. DEBUGGING.

  - BOTH 'TRACE' AND 'NORMAL' ARE PROGRAMMABLE.

NUMERICAL COMPUTATIONS

* **REAL CONSTANTS** CAN BE USED IN BOTH FIXED AND FLOATING FORMAT, USING THE 'ENTER EXP' KEY PUTS AN 'E' INTO DISPLAY. THIS CAN BE FOLLOWED BY A MAX. OF 2 DIGITS.

* **REAL VARIABLES** - ARE REGISTERS CONTAINING A REAL CONSTANT.
  - NAMES OF REGISTERS (VARIABLES) MAY BE USED IN MATHEMATICAL EXPRESSIONS.
  - REAL VARIABLES AVAILABLE :- A.B.C. X.Y.Z. AND THE AVAILABLE R( ) REGISTERS.

* **DESIGNATION OF 'R' REGISTERS**
  - THE R( ) KEY IS USED, FOLLOWED BY A NON-NEGATIVE VALUE.
  - VALUE CAN BE A REAL CONSTANT, A VARIABLE OR AN EXPRESSION TO BE COMPUTED:-

VALID STATEMENTS: R40 , R[30+7] , R(70/2) R(R5+10), RR2 , RRR3.

NOTE! IF THE VALUE FOLLOWING R( ) IS NOT AN INTEGER, THE FRACTION IS IGNORED.

REPRESENTATION OF NUMBERS

Fixed Point	Floating Point
123.45	1.2345×10^{2} or 1.2345 E 02
0.0012345	1.2345×10^{-3} or 1.2345 E-03
1.2345	1.2345×10^{0} or 1.2345 E 00
—	1.2345×10^{95} or 1.2345 E 95

Floating Point = Scientific = E

Notation

Example:

3. 1415926

Fixed 4 Execute → 3.1416

Float 2 Execute → 3.14 E00

MATHEMATICAL SYMBOLS

* **SYMBOLS** - AS FOLLOWS :-

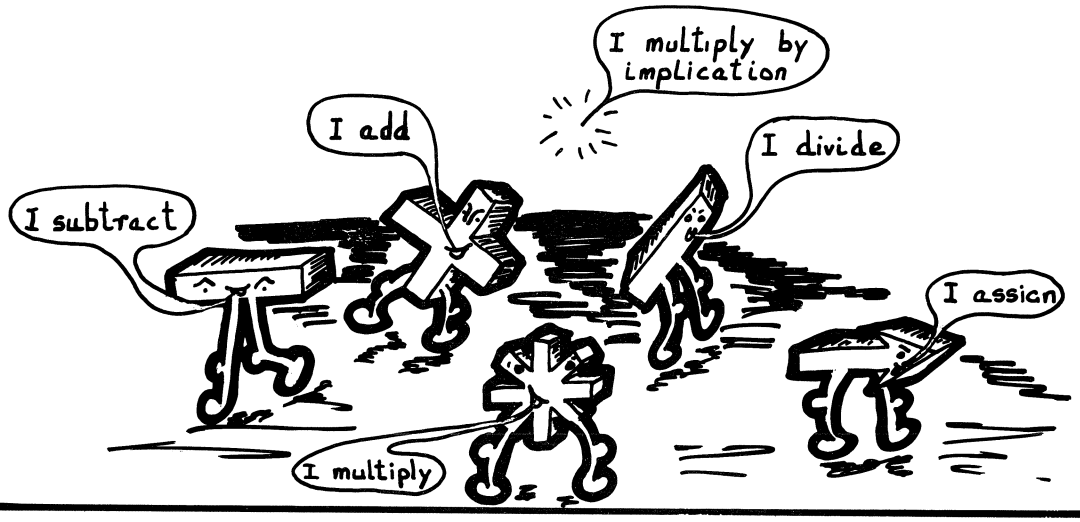
+	ADDITION
−	SUBTRACTION
/	DIVISION
*	MULTIPLICATION

* **IMPLIED MULTIPLICATION** - THE 9820A ALLOWS IMPLIED MULTIPLICATION BY SIMPLY GROUPING THE VARIABLES i.e.

$$AB = A*B, \quad 5(B+C) = 5*(B+C)$$

* **ASSIGNMENT** - THE SYMBOL '→' GIVES THE VARIABLE TO THE RIGHT OF THE SYMBOL THE VALUE SPECIFIED ON THE LEFT i.e.

10 → A - DEFINES 'A' EQUAL TO 10 (OR REGISTER 'A' CONTAINS 10)



IMPLIED Z-STORAGE

A result of any arithmetic expression may be stored in any register, e.g.

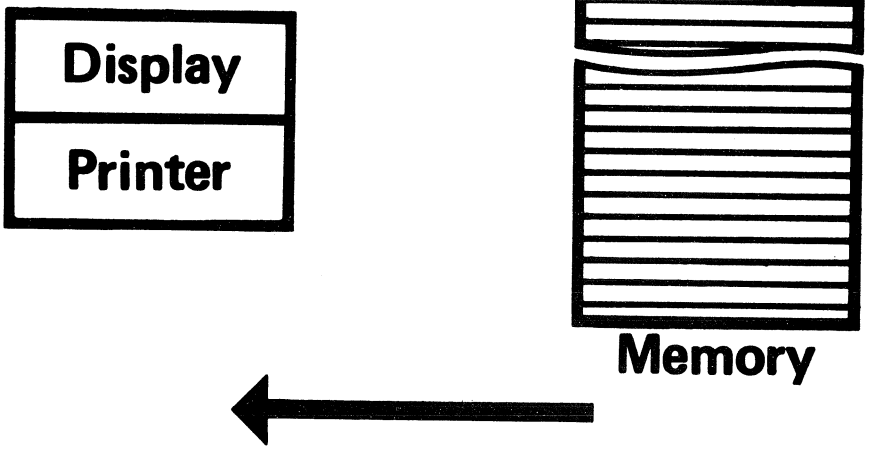
```
(A + B) * (X - C) →Y
```

However, if the user does not specify where the result of an arithmetic expression is to be stored, this number will be automatically stored in the Z-register, e.g.

```
(A + B) * (X - C)        Execute
is equivalent to
(A + B) *(X - C) →Z      Execute
or 14.2                  Execute
is equivalent to
14.2 → Z                 Execute
```

DATA STORAGE AND RECALL

12	→	A	Execute
6.3	→	X	Execute
19.5	→	R( ) 12	Execute



R ( ) 12	Execute
Display A	Execute
Print R( ) 12	Execute
Print X	Execute

EXTRA MATHEMATICAL FUNCTIONS

* **SQUARE ROOT** - THIS IS THE ONLY SUPPLIED FUNCTION IN THE BASIC MODEL 9820A. ITS HAS THE SYMBOL '√'. FURTHER MATHEMATICAL FUNCTIONS ARE SUPPLIED BY THE MATHS ROM AS FOLLOWS:-

FUNCTION OR OPERATION	MNEMONIC OR SYMBOL	NAME
ln x	LN	natural logarithm
e^x	EXP	exponential function
log x	LOG	common logarithm
$\log^{-1} x$	TN↑	antilog (base 10)
a^b	↑†	exponentiation
sin θ	SIN	sine
cos θ	COS	cosine
tan θ	TAN	tangent
$\sin^{-1} x$	ASN	inverse sine, or arcsine
$\cos^{-1} x$	ACS	inverse cosine, or arccosine
$\tan^{-1} x$	ATN	inverse tangent, or arctangent
$\lvert x \rvert$	ABS	absolute value
integer of x	INT	integer
π	π	pi
selection of degrees, radians, or grads	TBL	Table

† ↑ by itself is not permitted. The simplest form is ⟨quantity⟩ ↑ ⟨quantity⟩; a^b is represented by A↑B.

LEARN THESE FUNCTIONS!

HIERARCHY

How is an expression like the following executed?

$$((AA - 2.5\ X)\ 7C - \sqrt{(-4 + 6.2B)})$$

1. **An expression is executed from left to right.**
2. **Hierarchy**

First **Functions (SIN A, LN B)**
Exponentiation (A↑B)
Unary Minus (– B)
Implied Multiply (AB)
Explicit Multiplication, Division (A*X, A/B)
Addition, Subtraction (A + B, A – B)
Last **Relational Operators (>, ≤, =, ≠)**

The use of parentheses enables the order of execution to be changed! In $\sqrt{(4 + 5)}$ the '+' is executed before the '√'.

EXAMPLES OF MATHEMATICAL COMBINATION

- 3.14 → A : STORE 3.14 IN REG. A. PREVIOUS CONTENTS LOST.
- X ↑ Y : X RAISED TO THE POWER OF Y
- 1/[ALN X] : THE RECIPROCAL OF aLn x

* MATHEMATICAL NOTATION

- $-(a+b)$: -(A+B) OR -A-B
- a^{x+2} : A↑(X+2)
- $(ca)^x$: [CA]↑X
- $(a^{x+2})c$: A↑(X+2)*C OR CA↑[X+2]
- $\frac{x}{a+\frac{y}{3+z}}$: X/[A+Y/[3+Z]]
- e^{x+1} : EXP[X+1]
- $a=b^2+c^2$: B↑2 + C↑2 → A

FOLLOW THE RULES AND ALL WILL BE WELL. WATCH OUT FOR ORDER OF EXECUTION!

MODEL 9820A

LANGUAGE

* **SYNTAX** — THIS IS THE EXACT LITERAL OR FUNCTIONAL DESCRIPTION OF THE LANGUAGE USED IN THE MACHINE. DURING OPERATION ALL STATEMENTS ETC. ARE CHECKED FOR THEIR SYNTAX.

* **FIXED/FLOAT**

SYNTAX : FXD [⟨value⟩]
FLT [⟨value⟩]

- THE VALUE SHOULD BE A SINGLE DIGIT BETWEEN 0 AND 9.
- WHEN FIRST TURNED ON THE 20 IS IN FLT 9 MODE
- WHEN CHANGING BETWEEN FXD/FLT THE MACHINE REMEMBERS THE LAST PARAMETER USED - IF THE MODE IS CHANGED BACK WITHOUT USING A NEW PARAMETER, THE ORIGINAL ONE IS USED.
- NOTE 2 WILL OCCUR IF THE PARAMETER IS LESS THAN 0 OR GREATER THAN 9
- A NUMBER. TOO LARGE FOR THE CURRENT FXD SPECIFICATION WILL BE DISPLAYED IN FLOATING FORM. THE REVERSE IS NOT TRUE.

NOTE! THESE OPERATIONS ONLY AFFECT VISUAL OUPUT. NOT MACHINE ACCUR'Y!

* DISPLAY

SYNTAX : DSP [⟨list⟩]

- CAUSES EACH PARAMETER IN THE LIST TO BE DISPLAYED FOR APPROX .175 SECS.
- EACH ADDITIONAL 'DSP' AFTER A PARAMETER DISPLAYS THAT PARAMETER AN ADDITIONAL .175 SECS.
- IF A LITERAL IS LONGER THAN 16 CHARACTERS, ONLY THE LEFT-MOST 16 WILL BE VISIBLE.

EXAMPLES :

```
FXD 2;1→A;2→B⊢
DSP A,B⊢
```

```
      1.00
      2.00
```

```
DSP "A=",A⊢
```

```
A=
      1.00
```

```
FLT 9;FXD 8;DSP 99⊢
```

```
   99.000000000
```

```
DSP 100⊢
```

```
1.000000000E 02
```

```
DSP "100"⊢
```

```
100
```

```
DSP "ABCDEFGHIJK
LMNOPQRSTUVWXYZ"⊢
```

```
ABCDEFGHIJKLMNOP
```

```
DSP "bbbbTHISbISbAN";
DSP ;DSP ;DSP ;DSP "bb
EXAMPLEbOFbA";
DSP ;DSP ;DSP ;DSP "LONG
bLITERALb!"⊢
```

```
   THIS IS AN
   EXAMPLE OF A
   LONG LITERAL !
```

"TRY THEM"

(IN THESE CASES THE '⊢' MEANS EXECUTE)

* PRINT/SPACE

SYNTAX : PRT ⟨list⟩

SYNTAX : SPC [⟨value⟩]

'PRT'

- EACH PARAMETER IS PRINTED ON ITS OWN ROW.
- IF A LITERAL IS LONGER THAN 16 CHARACTERS. ONLY THE LEFT MOST 16 WILL BE PRINTED.
- EXTRA 'PRT' STATEMENTS BEHAVE IN A SIMILAR WAY TO THE EXTRA 'DSP'

'SPC'

- SPACE STATEMENTS ARE USED TO CONTROL PAPER ADVANCE.
- VALUE SHOULD BE BETWEEN 0 AND 15.
- 'SPC' STATEMENTS ARE WRITTEN WITH THE 'SPACE N' KEY - NOT WITH 'SPACE'

EXAMPLES

```
FLT 5;FXD 2⊢
10→A;20→B;2.6E10→C⊢
PRT "A=",A,"B=",B,"C=",C⊢
```

```
A=
              10.00
B=
              20.00
C=
      2.60000E 10
```

```
PRT "A+B=",A+B⊢
```

```
A+B=
              30.00
```

```
PRT "*";SPC 10;PRT "*"⊢
```

```
*




*
```

```
2→X;4→Y;PRT "*";
SPC X+Y;PRT "*"⊢
```

```
*




*
```

* ASSIGNMENT

SYNTAX : ⟨value⟩ → ⟨real, variable⟩

- THE STATEMENT ASSIGNS A NUMERIC VALUE TO A VARIABLE (PUTS NUMBERS INTO REGISTERS.)
- MORE THAN ONE VARIABLE CAN BE INCLUDED IN A LIST i.e.
  25 → A → B → C etc.
- VARIABLES AND COMBINATIONS OF VARIABLES CAN BE USED TO DEFINE OTHER VARIABLES.
- A PARAMETER WITHOUT AN ASSIGNMENT IS TAKEN TO BE AN IMPLIED STORE INTO Z

EXAMPLES

```
FXD 2;25→A→B→C↵
                    25.00

PRT A,B,C↵

25.00
25.00
25.00
```

```
FXD 2;0→A→B↵
PRT A+1→A,B+10→B;SPC 1↵
```

Press EXECUTE several times.

```
 1.00
10.00

 2.00
20.00
```

```
FXD 2;END↵    0↵    PRT Z↵
                    0.00

25↵    PRT Z↵
                   25.00

100→A↵    A↵    PRT Z↵
                  100.00
```

The statement:

```
R1R2→R3R4→X
```

is equivalent to the series of statements:

```
R1R2→R3
R3R4→X
```

If the statement were stored while the program line counter was at two, the display would be:

```
2:(R1R2→R3)R4→X↵
```

write the line:

```
10→B;A;25→C
```

then press STORE.

```
0:10→B;A→Z;25→C↵
```

* INPUT OF DATA

SYNTAX : ENT [⟨literal⟩ ;] ⟨real variable⟩
[[; ⟨literal⟩] ; ⟨real variable⟩

- ENTER STATEMENTS WILL CAUSE THE PROGRAM TO STOP AND WAIT FOR A VALUE FROM THE KEYBOAD TO ASSIGN A VARIABLE
- THE VALUE IS KEYED IN AND IT IS TERMINATED BY RUN PROGRAM.

NOTE! IF A LINE CONTAINS BOTH ENTER AND GOTO, THE ENTER MUST COME BEFORE THE GTO OR THE GTO WILL BE IGNORED.

- AT AN ENTER STATEMENT THE DISPLAY WILL SHOW THE NEXT VARIABLE OR A LITERAL AT THE LEFT OF THE VAR.
- WHILE WAITING FOR A RUN PROGRAM ANY CALCULATIONS CAN BE DONE TO CALCULATE A PARTICULAR VALUE.

EXAMPLES

Press MEMORY ERASE, or, END EXECUTE STORE EXECUTE. Then store the following program:

```
0:
FXD 2⊢
1:
ENT A,B,C⊢
2:
PRT "A=",A,"B=",
B,"C=",C;SPC 8⊢
3:
GTO 1⊢
4:
END ⊢
```

PRESS: [END] [RUN PROGRAM]

```
A
```

PRESS: [1] [0] [RUN PROGRAM]

```
B
```

PRESS: [2] [0] [RUN PROGRAM]

```
C
```

PRESS: [1] [0] [+] [2] [0] [RUN PROGRAM]

```
A=
                10.00
B=
                20.00
C=
                30.00
```

```
A
```

* ENTER (CONT.)

- YOU MAY NOT USE AN ENTER STATEMENT:-
  1. AS PART OF A LINE TO BE EXECUTED FROM KEYBOARD.
  2. AS PART OF A SUB-ROUTINE INITIATED BY A GSB FROM THE KEYBOARD.

WITH UDF ROM:

3. AS PART OF A SUB PROGRAM STARTED BY F ✱ EXECUTE.
4. AS PART OF AN IMMEDIATE EXECUTE SUB PROGRAM WHEN CALCULATOR IS IN 'ENTER MODE!

EXAMPLES

Initialize the program memory. Then store and run the following program:

```
0:
FXD 0⊢
1:
ENT AA→B,XX→Y⊢
2:
PRT A,B,X,Y⊢
3:
GTO 0⊢
4:
END ⊢
```

PRESS: (END) (RUN PROGRAM)

```
AA→B
```

PRESS: (1) (0) (RUN PROGRAM)

```
XX→Y
```

PRESS: (2) (0) (RUN PROGRAM)

```
0:
FXD 2⊢
1:
ENT "WHICH R REG
ISTER",R1,"WHAT
VALUE",RR1⊢
2:
PRT RR1,"IS STOR
ED IN REG",R1;
SPC 8⊢
3:
GTO 0⊢
4:
END ⊢
```

Flag 13 is a useful indicator that the last datum has been entered. Consider the following program for summing a series of numbers:

```
0:
FXD 2;0→B⊢
1:
ENT "NEXT NUMBER
",A⊢
2:
IF FLG 13=0;PRT
A;A+B→B;GTO 1⊢
3:
PRT "TOTAL",B;
SPC 8;END ⊢
```

* ABSOLUTE GO TO

SYNTAX : GTO ⟨unsigned integer constant⟩

- ENABLES BRANCHING FROM ONE LINE TO ANOTHER
- THE VALUE MAY NOT BE A VARIABLE OR HAVE A DECIMAL POINT.

* RELATIVE GO TO

SYNTAX : GTO + ⟨integer constant⟩
GTO − ⟨integer constant⟩

- ENABLES BRANCHING TO A LINE RELATIVE TO THE LINE CONTAINING THE GTO STATEMENT.

* LABELLED GO TO

SYNTAX : GTO ⟨literal⟩

- ENABLES BRANCHING TO A LINE HAVING THE SAME LABEL AT THE BEGINNING.
- THE SEARCH FOR THE LABEL STARTS AT LINE ZERO.

EXAMPLES

ABSOLUTE

```
0:
PRT "0";GTO 4⊢
1:
PRT "1";GTO 3⊢
2:
PRT "2";GTO 1⊢
3:
PRT "3";GTO 0⊢
4:
PRT "4";GTO 2⊢
5:
END ⊢
```

LABELLED

```
0:
PRT "0";GTO "→"⊢
1:
"YYCC";PRT "1";
GTO 0⊢
2:
"XXBB";PRT "2";
GTO "YYYCC"⊢
3:
"XXAA";PRT "3";
GTO "XXXBB"⊢
4:
"AA";PRT "4";
GTO "XXAA"⊢
5:
"A";PRT "5";GTO
"AA"⊢
6:
"→";PRT "6";GTO
"A"⊢
```

RELATIVE

```
0:
PRT "0";GTO +4⊢
1:
PRT "1";GTO +2⊢
2:
PRT "2";GTO -1⊢
3:
PRT "3";GTO -3⊢
4:
PRT "4";GTO -2⊢
5:
END ⊢
```

* GO TO SUB AND RETURN

SYNTAX : GSB ⟨unsigned integer constant⟩
GSB + ⟨integer constant⟩
GSB − ⟨integer constant⟩
GSB ⟨literal⟩

- ENABLES BRANCHING TO BE MADE THE SAME AS GTO - HOWEVER IN THE SUBROUTINE THE END LINE CONTAINS:

..... : RET ⊢

WHEN THE 'RET' IS SEEN BY THE CALC. THE PROGRAM CONTINUES AT THE NEXT LINE AFTER THE 'GSB' STATEMENT!

- IN THE BASIC CALCULATOR GSB's MAY BE NESTED TO A MAX 31 LEVELS. - IF UDF ROM IS IN PLACE THE MAX DEPTH IS APPROX 62.

EXAMPLE

```
0:
PRT "IN MAIN PRO
GRAM"⊢
1:
GSB "SUB1"⊢
2:
PRT "IN MAIN PRO
GRAM","AGAIN"⊢
3:
STP ⊢
4:
"SUB1";PRT "ONE
LEVEL DEEP"⊢
5:
GSB "SUB2"⊢
6:
RET ⊢
7:
"SUB2";PRT "TWO
LEVELS DEEP"⊢
8:
RET ⊢
9:
END ⊢
```

KNOW THE DIFFERENCE BETWEEN GTO AND GSB!

RESULT:

```
IN MAIN PROGRAM
ONE LEVEL DEEP
TWO LEVELS DEEP
IN MAIN PROGRAM
AGAIN
```