WANG LABORATORIES, INC.
ONE INDUSTRIAL AVENUE, LOWELL, MA 01851 • TEL: 617/459-5000, TWX 710-343-6769, TELEX 94-7421

2600 VP

WANG

Maintenance Manual

Single User, Non-Interrupt, Microprogrammed System

CPU consists of:
1. Bootstrap Rom (1K)
2. 3 Section ALU (Binary, Decimal, & Multiply) & System Control Logic.
3. Memory Control Logic
4. Control Memory (Up to 64 K X 24 Bit words)
5. Data Memory (Up to 64K X 8 Bit Words)
6. I/O Interface
7. Power Supply

Shugart SA900/901 Disk Drive- 3.1 Megabytes of Data w/ transfer rate of 250 KBits/sec.

2600VP-Data Memory Size - 16K Expandable to 64K in 16K increments.

RAM Size - Not switch selectable 64K 65,535 Bytes

Bootstrap Routing searches for end of Data Memory and stores final address in CPU File Register, it is then moved to another location in Data Memory and used as "End of Data Memory" Flag for CPU.

Basic 2 Language Reference Manual

Commands - instructions controlling operation but generally not programmable.

Statements - programmable instructions.

Functions & Operators - Used to construct numeric and alphanumeric expressions.

Expressions - Must be included in statements.

Legal Range of Program Line Number - 1 to 9999
Multiple statements may be put on same number when separated by colons.
Spaces stored in memory along with line occupying one byte.

Max Length of Program Line - Determined by free space in memory & capacity of CRT screen.
Max Line Length determined by CRT (16 X 24) - 1024 characters.
(80 X 24) - 1920 Characters.

If program line is too long for space still available in Memory, Error Message displayed but 256 Bytes stored temporarily in special buffer and may be edited to fit.

Max Line Length for Program Line to be stored on Disk is 253 Bytes. Any line entered into system over 253 will give ERROR A05 which is just a warning concerning disk and will still run.

Immediate Mode Statements - Not stored in main memory, but temporarily held in special buffer following execution and may be recalled for editing after it is executed.
Removed when another immediate mode statement executed or clear or M.I.
Special Buffer - 256 Bytes long. If immediate mode statement exceeds 256, excess not saved.

A numbered program line is stored in memory regardless of syntax.

Keywords are atomized in one byte "text atom". Takes up less memory and speeds up processing by less scanning of memory.

Resolution Phase - Entered just prior to execution, initiated by RUN. Reserves space in memory. For referenced variables and ensures all referenced line #'s exist. If error free, goes to Execution Phase otherwise back to Entry Phase.

SEVERAL WAYS TO INITATE EXECUTION:

Only RUN and LOAD RUN commands & Load Statements initiate Resolution.

2 Types of Data - Numeric and Alphanumeric Data. Each numeric value always occupies exactly 8 bytes.
2 Classes of Variables (Alpha & Numeric) - Each class can have 2 types of variables defined (scalar and array).
Alpha Scalar variable - from 1 to 124 bytes long (Default length is 16 bytes)

Array Variable - Collection of scalar variables identifies by common name . Each scalar variable referred to and an "element" of array and can be defined by Array Name and subscripts locating it. Subscripts are in (_)following Array Name.

Empty () must be used to reference entire array.

2 Dimensional Array - (Row, column) - Row (vertical) Column (Horizontal)

One Dimensional Array - Available memory only restriction on size.

Two Dimensional Array - max size 255 X 255 or 65,025 elements.

Ea. Element - From 1 to 124 bytes.

COM & DIM - variables defined by DIM are uncommon while COM defines common variables.

Variable Table - where variables in memory are stored begins at end of memory & builds as variables defined.

All common variables are put in variable table first and separated from noncommon by common variable pointer, CVP.

Common Variables - must be defined before any non common. Without parameters makes all variables non common.

COM CLEAR - statement used to move CVP up or down. If used with common variable, CVP move down to before making that non common variable. If used with non common variable CVP moves up to just before named variable.

Internal Stacks - Temporary storage of certain control info; always used in 2 stacks, operator & value.

RETURN-CLEAR - dummy return statement which clears subroutine info from stacks w/out retuning to statement after GOSUB.

RETURN CLEAR ALL - Clears entire contents of stacks.

END - Terminates program, clears stacks, gives free space. In immediate mode - stacks not flushed.

Work Buffer - minimum of 192 Bytes. Between Program Test Area and Variable Table, not part of free space. Up to 128 Bytes may be used by value stack.

If Value Stack uses part of work buffer, SPACE would give - Free Space. To recall immediate Mode Line of Statement which when entered gave error just key:

EDIT - RECALL

When entering multiple values in response to an INPUT statement. If an erroneous value is entered all data following and including, is rejected while previous data is accepted

SELECT - 5 Purposes

1. Device address for I/O Statements.
2. Output Parameters for communicating w/output devices.
3. Desired Math Mode for Arith. Operations.
4. Rounding or truncation of numeric results.
5. Desired Sys. response to specific math errors.

2600 DEVICE TABLE

LIST DT:

CI	INPUT	PLOT	TAPE
0taa	0taa	0taa	0taa

CO	PRINT	LIST
0taaee00	0taaee00	0taaee00

#0	#1
ptaasssscccceeee	ptaasssscccceeee
#2 ↓ #14	#3 ↓ #15

t = Device Type
aa = Unit Device Address
ee = Line Length
p = Platter Parameter

ssss = Starting Sector Address
cccc = Current Sector Address
eeee = Ending Sector Address

DEVICE TABLE AFTER MASTER INITIALIZATION

Keyboard	Keyboard	PLOT	TAPE
0/0/01	0/0/01	0/4/13	0/1/0A

CRT	CRT	CRT
0/0/05/40/00	0/0/05/40/00	0/0/05/40/00

HEX 40 = 64

DISK

0/3/10/0000/0000/0000	0/0/00/0000/0000/0000
0/0/00/0000/0000/0000	0/0/00/0000/0000/0000

NUMERIC OPERATIONS:

Numeric Values - stored in special floating point format
13 Digits, sign, & signed 2 digit interger exponent
Legal Range - -10 (100) less VALUE less -10 (-99) less =value less 10 (+100). Greater than 13 digits gives error.
Leading zeros ignored.
Numeric Scalar Variables - -A - Z - A1 -Z9 286 Possibilities
Numeric Array Variable - Subscript in (,)
Space must be reserved for numeric array variable w/DIM or COM
No zero subsript in Arrays.
One Dimensional Array 1 to 65,535 max
Two Dimensional Array 255 X 255 max.

Order of Evaluation Values in () done first always

1. Exponentiation left to right
2. X & / Left to Right
3. + & - Left to Right

Trigonometric Function Evaluated in Radians, unless degrees or GRAD selected SELECT D or G

CHAPTER 5

Alphanumeric Variables - Distinguished from numeric variables by preceding $. May not be used for standard decimal arithmetic operations. Two types - Alpha Scalar and Alpha Array Variables (Subsripted) Alpha Scalar can store one character string from 1 to 124 characters. Legal to have an alpha scalar & an alpha array variable with same name in program. Exp A$ & A$(5). End of variable normally last non-blank character unless variable is all blanks in which case represented by 1.

"Current Value" - character string stored in alpha variable.

"Current Length" - length of string up to 1st trailing blank.

LEN Instruct. gives current length of variable.

EXP 10 A$ = "ABCD____" 20 Print LEN (A$) RUN 4

Literal String - character string in " ".
Length of Alpha Variable 1 to 124 char. Put aft $ or () Exp A$1 A$124.
Hexadecimal Literal String - consists of 1 or more hexdec codes.
Specified in HEX Function EXP. A$ = HEX 313233
PRINT A$ RUN 123

Any hex code may be used except FA thru FF, used for system use.

Concatenation Operator - combines strings w/out intervening char.

Exp. A$= "WANG" B$ = "LABS" C$ = A$ & B$ PRINT C$ WANG LABS

May only be used in 1 statement LET, Illegal w/other statements

Alphanumeric Operators & Operands

ADD	DSC	Alpha Var A$, B1$, C$(2), D$()
ADDC	OR	Alphanum. Literal String "JOHN"
AND	SUB	Hex Literal String Hex (OB)
BOOLH	SUBC	String Funct STR (A$,1,J)
DAC	XOR	All Funct. (ALL (FF)
		Bin Funct. (BIN I + 1)

& - Cannot be used in conjunction w/ other operators.

2200 PROGRAMMING

Powering On - make sure all power on sw off, cable up, power on peripherals, power on.
Master Initialization - when main power turned on mem. clred 2.
READY will come up on CRT.
*If READY does not appear immediately, leave power on 15 sec and then power off and on again. Then READY should come up.

CRT - 16 (64 Char) Lines capability. If more than 16 lines enter next line goes to bottom & all move up.
: - Colon must appear to enter INFO
: _ Has mark next to : denotes where input char. pos (CRT Cursor)

RESET KEY - clrs. CRT and prints READY & :_
stops any processing
unlocks keyboard (will not alter memory in any way).
Literal String - any char within set of " " followed by ; will be printed out exactly as in " " following print command.

EXECUTE KEY - Line checked for grammatical correctness. If line wrong, error message generated. If line correct, line executed and cleared from memory. If line # in front of line 2200 waits for next instruction.

Special Function Keys: 16 keys at top of keyboard.
Keys used to customize calculator for user needs.

Order of Execution: 1. Exponentiation (↑) a. Multiplication (*), Division (/), Subtraction (-), Addition (+). When parenthesis are used, the expression within is done first. 2 Math symbols cannot be next to each other. Use (), i.e. 5X-3 is wrong, 5X(-3) is right. You can use as many () as you like and it will not affect problem.

KEYBOARD FUNCTION KEYS

SIN (EXPRESSION) = FIND SIN OF EXPRESSION
COS (") = COSINE
TAN (") = TANGENT
ARCSIN (") = ARCSIN
ARC COS (") = ARCCOSINE
ARCTAN (") = ARCTANGENT
π = Assigns the value (3.14159265359)
LOG (") = Find natural Logarithm of expression.
EXP (") = Find value of e raised to value of expression
SQR (") = Find Square Root of Expression
E - Exponent of 10.
Floating Pt. Notation - can include up to 13 digits, dec pt., sign, & a 2 digit pos of neg exponent.
Keyword Print will display on only 9 digits w/remaining digits kept internally.
Largest EXP is E99, smallest is E-99, no dec. or fraction allowed.

ONE LINE PROGRAMMING

Statement in a statement line must be separated by :
When looking for unknown X = 5 + 19, unknown must be left of =.

Scalar Variable - variable that can represent only 1 number at a time. The value may be ± 10 -99 to ±10 99.
A scalar variable may be represented only by 1 letter or 1 letter followed by one digit.

Undefined variables automatically = 0

ALL POSSIBLE COMMANDS may be keyed in or typed in.

CLEAR - clears 2200 memory. CLEAR P - clears only program, not variable.
CLEAR V - clears just variables. CLEAR N - clears non-common variables.

FORMAT -
Zoned Format - 4 (16 space) zones horizontally/line. All variables, numeric values, & expressions including single PRINT statement w/commas separating elements.
On Multiple Statement lines the comma must come before the colon":".

Packed Format - less than 4 values/ line - just use semi-colon instead of commas.

TAB Format - when TAB (#) is typed in, begin cursor position at column indicated within (). First column is 0 & last is 63. If numeric, will leave space for + sign.
-Contents of () in TAB command can be any algebraic expression, however only integer portion of evaluation recognized.
-If # in () is less than 0 or greater than 63, begin with 1st column of next line.
-If printing position is past requested TAB loc, command ignores.

FOR TO - starts loop & decides # of time Loop Repeated (For X = 1 to 25).

NEXT - increments counter & serves as outer boundary of loop, (NEXT X).

CRT Plotting - For X = 0 to 10: Print TAB(X); "+":NEXT X
Use Print TAB together w/ X varying

PROGRAMMING

Statement Line number - inserted by numeric keyboard 1 - 9999. Keying statement will automatically generate # in increments of 10.

RUN - will reinitialize all non-common program variables. RUN can start at beginning or by entering STMT # aft it, at that stmt. Keying in STMT # & a statement followed by EXECUTE will enter line in memory.

LIST - will list program on CRT.

LIST S - will display 15 lines at a time, hitting EXEC will display next 15. If a certain group of STMT lines want to be read, enter beginning & end line # - 1 line, just its line #.

SELECT P n - pause statements to allow time to scan program where n = 1/6 sec pause aft every line w/ 0-9 values.

STOP - could be put in program w/ line # to halt at predeteremined place. When comes to this instruction, will halt & put STOP on CRT. If may be used w/ literal string " " where anything in quotes will be printed. W/ program halted by STOP, user can:

1. Use 2200 as calculator & execute stmt line w/out # (immediate mode).
2. Print variables for inspections.
3. Redefine variable used in program to see its affect.
4. Cont. Prog exec from different proram line.
5. Cont. at next instruct w/CONTINUE instructed.

END - Halts program & displays total amount of unused free space, memory remaining. STOP & END can be used to separate different prog in memory. 2200 requires 700 bytes work area (Byte - comparable to program step) in 4K machine. 4096 Bytes with memory clear. END would show free space = 3398

Conditional Branch - For Next

REM - uses memory space. non-executable instruction. use to insert explanatory comments in program. Will not be seen in program when executed only when listed.

GOTO Line # - unconditional Branch.

DATA - used to store #'s alphanumerics in program. Values would follow Data Stmt & be separated by commas (used w/Read Stmt).

READ - used w/ DATA STMT to assign variable names to each value would be followed by variables names separated by comma. If fewer items are read than were stored, a data pointer will keep track & you will next start off w/next value.

RESTORE - used with READ & DATA STMT. Allows data pointer to go to lst value or if followed by a # to the value at that #'s position (If try to restore to a non-existant data value will err and terminate program execution).

IF/THEN - conditional branch. If compare is on THEN Line # - 6 possible comparisons: =,>,<,>=,<= ,<>. If true goes to line 3 of then. If false, go to next instruction. When using DATA statement program will halt when run out of values in Data Statement.

INPUT x, y - stops program with ? on CRT which allows user to enter corresponding Amount of data values separated by commas. A "Literal String" may be used w/Input which would be followed by ?. Literal String has max length of 192 keystrokes.

ARRAY - set of #'s arranged in table, such that ea # uniquely identified by position. ARRAY is called R(,); first # = row, second # = column. 286 array names possible (A-Z, A0-Z9). 64 char max length.

DIM - Dimension statement - Before array used in mem, must be set aside for entire array. Using this STMT. MAX. Dimension are 255 Rows and 255 Columns. DIM (Array Name) (Rows, Col) for more than one Array w/single DIM separate w/ comma. DIM Statement must appear before any variables in program numeric value of subscript may not be zero.

NESTED LOOPS - use 2 or mor FOR/NEXT Loops having 1 FOR/NEXT Loops within other. This will cause inner loop to go thru an entire dimension for every time outer loop goes through once.

Example:

```
10 DIM X(5,3)        Set up ARRAY X, 15 elements, 5 rows & 3 col.
20 FOR B = 1 to 5    B = 1 thru 5 for each loop.
30 FOR C = 1 to 3    C = 1 thru 3 for each loop.
40 X(B,C) = 1        Set Element at (B,C) 1(lst loop),1, = 1
50 NEXT C            Increase C and goto 40 until C = 3 Then goto 60
                     until B = 5
60 NEXT B            Increase B and goto 30 where C= 1 again C = 3
                     then to 70.
70 END               Finished.
```

STRING VARIABLE - variables whose value are literal strings. Denoted by a letter or letter & # followed by $. 286 possibilities. Until assigned value assumed to consist of one space. Unless specified in DIM statement Max of 16 Alphanumeric spaces/ w/DIM 64 characters possible.

GOSUB - cannot be last instruction. Followed by statement # which begins a subroutine. Last instruction must be a RETURN in subroutine may have REM after but not an instruction. RETURN will bring you to next instruction after GOSUB that initiated subroutine.

String Arrays - can be dimensioned, max. subscript 255. Max char. ea. string array element can assume 16, unless size set to another value. 1 to 64 char. in length.

STR - allows ability to manipulate string variables. STR (string position of # of chars. want varaiable as lst character. W Name, to work with, to work W/ rest of string if no # inserted).

LEN - to determine length of alphanumeric string excluding blanks because it will represent # of characters in string variable.

Program Chaining - when program too large for memory stored on ext. device & loaded 1 segment at a time.

COM - used to keep specified variables intact when loading in now segment or subroutine which will clear all memory otherwise. May represent any type of variable, separate variables with ,s.

LOAD - program stops everything cleared except common variables, recorded programs loaded into memory and executed. If followed with starting & ending STMT #'s only this section of memory cleared along w/uncommon variables.

PRINTUSING & IMAGE - always used together. Image literal strings need no " ". Contains line # & % sign w/ either exact image of what is to be printing or symbolic (#) representation of how should be printed out. Printusing always followed by STMT # of image STMT when using sym. ref., put , after image line # then whatever is to be written as in PRINT STMT. (, in PU STMT, If only one sym rep skip to next line); Put on same line. Only # can represent a digit, a period a decimal point. Images which contain extra # symbols to right of decimal point than data in PU STMT are represented w/ zeros.

- If contain extra # sym to left of dec. pt. then leading blanks are left.

- If fewer # sym. to left of dec. pt., then data in PU STMT #; are printed instead of number indicating image not large enough.

- ,s may be used in image statement to help read numbers.

- If + used in front of image, correct sign + or - is placed in front of first significant digit.

- If - used in front, only - sign print, nothing will appear before positive numbers.

- If no sign is used & - appears number will be shifted to right one place.

- When use $ will appear immediately in front of number of if - in from of -.

- Four↑ (↑↑↑↑) used in image stmt to represent exponent field.

Exponent value in output adj. to align dec pt. in value w/ decimal pt. in image.

HEXIDECIMAL FUNCTION

All char & commands outputted to peripheral are represented by Z digit. Hex code --256 codes--00-FF.

HEX ( ) - put two digit code in () or can put more than 1 code just side by side without funct, just ().

TRACE - when used will printout changing values of variables & transfer to STMT # when doing branches.

RENUMBER a,b,c - [a - line number to begin renumbering from.] b - new line number for first line to be renumbered. c - increment between lines. No numbers then renumber from beginning w/10 in increments of 10.

DEFFN (Function number) (Dummy Variable) - function name (36), 0-9 & A-Z.

- Dummy variable can be any numeric variable, solely place holder, doesn't effect variable of same name elsewhere in program.

- to reference function FN (Function Name) (Variable).

- A Function can refer to an already defined function.

- A Function cannot refer to self nor can two function refer back to each other.

DEFFN' 0-255 Literal String Variable - # 0 -31 put literal string or variable on special function keys 0-15,lower case; 16-31, upper case.

- Also can be used for subroutines, called marked subroutines. marked subroutines
```
40 DEFFN ' 100
60 RETURN
```

- To access you would have a GOSUB '100 referring to integer used in DEFFN STMT.

- Variables, are put in () separated by commas.

- To enter special function key subroutines w/ variable, insert values for variables before keying.

- To enter via program use GOSUB' (integer or function)(variables values) A-Z & A0-Z0.

Arrays - Variables w/ 1 or 2 # subscript in (,). 1st subscript is row across second is column down. Before using an array area must be set aside for it by DIM.

ON (Variable) GOTO, GOSUB Line #, Line # - will GOTO or GOSUB to line # represented by variable, if variable = 2 go 2nd line #.

- If variable higher than # of line # go next instruction.

PACK (Image) Alpha variable array From numeric variable array

UNPACK (Image) Alpha variable To Numeric Variable

- When considering how many values can be packed/alphanumeric variable, first determine the number of bytes:

RULES FOR PACKING

1. Every 2 digits in image require 1 byte of mem. # represents 1 digit
2. + or - takes 1/2 byte (byte will contain sign of both # & exponent for images).
3. If no sign absolute value of # store 2 assumed +.
4. Dec. pt. not packed, specified in image when unpacking.
5. Packed numeric value always occupies whole # of bytes.
6. If image format 1 value edited as fitted pt. # extending w/ zeros any fraction & inserting leading zeros for insignificant integer digits, according to image specifications.
7. If image format 2, value edited as floating #, no leading zeros, exponent occupies 1 byte.

FORMAT 1 - Fixed Point ##.## FORMAT 2 - Exponential #.##↑↑↑↑

- When Packing able to store two #'s in 1 character space because each character space has eight bits and a # can be put in both upper and lower halves, so numbers will take up 1/2 normal space required when packed.

POS (alpha variable "00-9" A-FF) - will search designated alpha variable for first char. which meets comparison to otherside of equation & represent the number in position left to right of that character.

NUM (alpha variable) - scan alpha numeric variable from beginning or when used w/ STR() from any position to count consecutive valid numeric char until find non-numeric char or fails to conform to basic # format.
- Valid numeric char. include 0-9, spaces, = -, ., & E.

- Use to determine length of numeric portion of alpha-numeric variable.

BIT & BYTE MANIPULATION

Every character or (keyword) in alphanumeric variable requires 1 byte of memory. Every character is represented by 8 bits. 1 byte = 8 bits. Each digit in Binary # represents an increasing power of 2.

	7	6	5	4	3	2	1	0
Power	2	2	2	2	2	2	2	2
Value of Pos	128	64	32	16	8	4	2	1
Binary Digit	0	0	1	0	0	1	0	1 = 32 + 4 + 1 = 37

1 Byte - 8 bits - represent 2 binary numbers - Hexadecimal Code.

Each Key represents unique Hex Code - 1 byte of memory.

Bits 1-4 - 2nd digit of Hex Code Bits 5-8 - 1st digit of Hex Code.

Key	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
A-41	0	1	0	0	0	0	0	1
$-24	0	0	1	0	0	1	0	0

ADD C (alpha variable to be worked on, alpha variable; XX- (0-9 or A-F).

C If included then carrier between bytes XX or 2nd alpha variable - will be added to 1st alpha. Variable 2 types of Addition: 1. Immediate Addition - when (XX) 2 Hex digits are used where (XX) is add to each char of alpha variable. 2. String to String Addition - when 2nd term in () is alpha variable then value of 2nd string added to first starting from right addition done in binary.

AND or XOR (Alpha variable, alpha variable : XX- (0-9 or A-F).

AND - compares specified XX or 2nd alpha variable to 1st alpha variable setting a bit to "1". If both bits "1" otherwise "0".
OR - compares bit for bit, if both off "0" otherwise "1".
XOR - exclusive or - if both bits same "0" if different "1".
INIT XX- (hex Code) Alpha variable, Alpha Variable
"Character" Alpha Array, Alpha Array
Alpha variable
- ea. character in variable or array is set = to char in () & if it is alpha variable to first char, left most of that alpha variable.
- If () is char must be in " ".

ROTATE (Alpha Variable, 1-7) - bits in ea char rotate right to left w/ end around from 1 to 7 positions.

BOOL 0-F (Alpha variable,(XX-00-FF) Alpha variable) - takes 1st alpha variable & performs 1 of 16 functions, 0-F, w/ 2nd term. To identify code want to use 1010 are compared where you find answer to these term w/comparison you want to make & answer will represent 0-F.

BIN (Alpha Variable) = 0-255 - first char in alpha variable set equal to char w/bit configuration form 0-255.

VAL (Alpha variable/ literal string) - the binary value of the first char of the variable or string is converted to decimal equivalent.

HEXPRINT Alpha Variable , Alpha variable
" " Array " " Array , ... - prints out hex code value of ea char of alpha variable or array. If comma between variable next variable on next line, if semi-colon printed without space between on same line.

CONVERT Alpha variable To Numeric Variable - converts valid 2200 number in alpha variable to its numeric value & assigns that numeric value to the numeric variable in STMT.

CONVERT Numeric Variable To Alpha Variable, (Image)
Image has 2 formats, 1. Fixed Pt. (##.##)
2. Floating Pt. (#.##)
- Value of numeric variable put alpha variable in image shown.

Keyin Alpha variable, line number, line number - looks for 1 character inputted from either a 2215 or 2222 Keyboard, Paper Tape Reader, Punched Card Reader, etc. & puts it at 1st position of alpha variable then goes to first line #. If special function key pushed goes to 2nd line #.
- If char not available at time go to next instruction.

LET - optional word used to help read program when assigning variables, Expl.
LET = 5.

SYNTAX ERROR - when required format of 2200 stmt violated pressing sequence of keys not regognized as accepted combo. Recognized & Noted as soon as Execute key hit.

LOAD #n/XXXX, "Name", Line #1, Line #2 -

#n - File # to which device is assigned (1-6)
XXXX - Device Address
"Name" - name of program (1-8 characters) to be searched for & loaded.
Line #1 - first line to which program will be saved loading.
Line #2 - mem. saved/cleared to this line.

SAVE #n/XXX, P "Name", Line #1, Line #2 -

P - sets protection bit on program. File to be saved.
INPUT - when inputting for an alpha variable if don't use " " commas & carriage returns terminate & leading spaces ignored.
When comparing strings. - short strings are filled out w/ trailing spaces to compare w/ longer strings.

GOSUB line # & Return - upon finishing subroutine at line # there should be a RETURN STMT at end of subroutine which will bring program back to next STMT after GOSUB whether in same STMT line as GOSUB or the next STMT used.

**Cannot overlap FOR/NEXT loops & subroutines.

PRINTUSING Line # of image, print element [, ;] -
Digits to right of decimal under formatted for in image are lost digits to left of decimal point under formatter for cause #(image not big enough) to be printed.

If image overformats leading zeros inserted.

Literals in IMAGE stmt do not need quotation marks.

4↑ symbols ↑↑↑↑ represent image of exponent field.

**Exponent value adj to align w/decimal point in image.

W/only 1 image & several pieces of data to PU commas put ea. piece of data on new line, semi-colon output continues on same line

If no sign in IMAGE & neg number. in PU STMT that number shifted 1 place to the right & - sign put in.

If $ in Image immediately precedes pos. number, immediately precedes "-" sign which precedes #0.

symbol used to represent each character of strings also.

COM - must be 1st executable prog. line in program. Common variables must be defined before any non-common variables.

STMT Line - may have up to 192 char keystrokes including EXECUTE.

TRACE - turns trace on. TRACE OFF - turns trace off.

*2200 - all characters or commands outputted to peripherial represented by 2 digit Hex code.

256 Hex Codes 00-FF Print Hex (0909) same as PRINT HEX (09); Hex(09).