

DR. WANG'S PALO ALTO TINY BASIC

By Roger Rauskob

Tiny Basic was first proposed in Dr. Dobb's Journal. Li-Chen Wang's version of Palo Alto Tiny Basic originally appeared in Issue No. 5, May 1976 of Dr. Dobb's Journal. A complete listing was printed, but Dr. Wang did the assembly on an IBM computer and he defined different mnemonics. In order to assemble with an Intel Compatible Assembler a translation of most mnemonics had to be performed.

I had developed my own system, which consists of two small p.c. boards, one containing the 8080 CPU, 4k of 2708 type EROM and 1K of RAM. The other PCB contained the RS-232 Interface using the Intel 8251. So I wanted to change the I/O section.

If you want to change I/O, all routines are contained in the OUTC and CHKIO routines. My system uses the following configuration:

2708	EROMS	0000H	TO	07FFH
1k	OF RAM	1000H	TO	13FFH
8251	DATA PORT	0FAH		
8251	STATUS PORT			

Command Instruction	27H = 2 stop bits parity disabled. 8 bit characters. Baud Rate Factor of 04.
Mode Instruction	0CFH = No Hunt Mode. Not (RTS) forced to 0. Receive Enabled Data Terminal Ready Transmit Enabled.
Transmitter Ready Status Bit = Bit 0 (01H)	
Receiver Buffer Ready Status Bit = Bit 1 (02H)	

The program is contained in locations 0000H to 0768H.

In 1K of RAM 847 bytes are left over for program.

Tiny Basic does not offer much in terms of functions and general mathematical capabilities. But it is great to teach programming basics to children (and adults) and for games, since it has the RND function. It takes up little memory space and executes a lot faster than other basics.

Dr. Wang was very helpful and assisted me all the way. Some errors were eliminated. I appreciate his help and he deserves a lot of credit for his implementation of Tiny Basic.

See Microcomputer Software Depository Program Index for Copies of this program.

THE TINY BASIC LANGUAGE

Numbers

In Tiny Basic, all numbers are integers and must be less than or equal to 32767.

Variables

There are 26 variables denoted by letters A through Z. There is also a single array @(). The dimension of this array (i.e., the range of value of the index 1) is set automatically to make use of all the memory space that is left unused by the program. (i.e., 0 through SIZE/2, see SIZE function below.)

Functions

For the time being, there are only 3 functions:

- ABS(X) gives the absolute value of X.
- RND(X) gives a random number between 1 and X (inclusive).
- SIZE gives the number of bytes left unused by the program.

Arithmetic and Compare Operators

- / divide. Note that since we have integers only, $\frac{1}{3}=0$.
- * multiply.
- subtract.
- + add.
- > compare if greater than.
- < compare if less than.
- = compare if equal to. Note that to certain versions of Basic "LET A=B=0" means "set both A and B to 0". To this version of Tiny Basic, it means "set A to the result of comparing B with 0".
- # compare if not equal to.
- \geq compare if greater than or equal to.
- \leq compare if less than or equal to.
- $+, -, *,$ and $/$ operations result in a value of between -32767 and 32767. All compare operators result in a 1 if true and a 0 if not true.

Expressions

Expressions are formed with numbers, variables, and functions with arithmetic and compare operators between them. + and - signs can also be used at the beginning of an expression. The value of an expression is evaluated from left to right, except that * and / are always done first, and then + and -, and then compare operators. Parentheses can also be used to alter the order of evaluation.

Statements

A Tiny Basic statement consists of a statement

SOFTWARE SECTION

number between 1 and 32767 followed by one or more commands. Commands in the same statement are separated by a semi-colon ";". "GOTO", "STOP", and "RETURN" commands must be the last command in any given statement.

Program

A Tiny Basic program consists of one or more statements. When a direct command "RUN" is issued, the statement with the lowest statement number is executed first, then the one with the next lowest statement number, etc. However, the "GOTO", "GOSUB", "STOP", and "RETURN" commands can alter this normal sequence. Within the statement, execution of the commands is from left to right. The "IF" command can cause the execution of all the commands to its right in the same statement to be skipped over.

Commands

Tiny Basic commands are listed below with examples. Remember that commands can be concatenated with semi-colons. In order to store the statement, you must also have a statement number in front of the commands. The statement number and the concatenation are not shown in the examples.

REM or REMARK Command

REM anything goes
This line will be ignored by TBI.

LET Command

LET A=234-5*6, A=A/2, X=A-100,
@(X+9)=A-1

will set the variable A to the value of the expression 234-5*6 (i.e., 204), set the variable A (again) to the value of the expression A/2 (i.e., 102), set the variable X to the value of the expression A-100 (i.e., 2), and then set the variable @(11) to 101 (where 11 is the value of the expression X+9 and 101 is the value of the expression A-1).

LET U=A#B, V=(A>B)*X+(A<B)*Y

will set the variable U to either 1 or 0 depending on whether A is not equal to or is equal to B; and set the variable V to either X, Y or 0 depending on whether A is greater than, less than, or equal to B.

PRINT Command

PRINT

will cause a carriage-return (CR) and a line-feed (LF) on the output device.

PRINT A*3+1, "ABC 123 !@#", ' CBA '

will print the value of the expression A*3+1 (i.e., 307), the string of characters "ABC 123 !@#", and the string " CBA ", and then a CR-LF. Note that either single or double quotes can be used to quote strings, but pairs must be matched.

PRINT A*3+1, "ABC 123 !@#", ' CBA ',

MICROCOMPUTER DEVELOPMENT SOFTWARE

will produce the same output as before, except that there is no CR-LF after the last item is printed. This enables the program to continue printing on the same line with another "PRINT".

PRINT A, B, #3, C, D, E, #10, F, G

will print the values of A and B in 6 spaces, the values of C, D, and E in 3 spaces, and the values of F and G in 10 spaces. If there are not enough spaces specified for a given value to be printed, the value will be printed with enough spaces anyway.

PRINT 'ABC', ←, 'XXX'

will print the string "ABC", a CR without a LF, and then the string "XXX" (over the ABC) followed by a CR-LF.

INPUT Command

INPUT A, B

When this command is executed, Tiny Basic will print "A:" and wait to read in an expression from the input device. The variable A will be set to the value of this expression. Then "B:" is printed and variable B is set to the value of the next expression read from the input device. Note that not only numbers, but also expressions can be read as input.

INPUT 'WHAT IS THE WEIGHT'A, "AND SIZE"B

This is the same as the command above, except the prompt "A:" is replaced by "WHAT IS THE WEIGHT:", and the prompt "B:" is replaced by "AND SIZE:". Again, both single and double quotes can be used as long as they are matched.

INPUT A, 'STRING', ←, "ANOTHER STRING", B

The strings and the "←" have the same effect as in "PRINT".

IF Command

IF A<B LET X=3; PRINT 'THIS STRING'

will test the value of the expression A<B. If it is not zero (i.e., if it is true), the commands in the rest of this statement will be executed. If the value of the expression is zero (i.e., if it is not true), the rest of this statement will be skipped over and execution continues at next statement. Note that the word "THEN" is not used.

GOTO Command

GOTO 120

will cause the execution to jump to statement 120. Note that GOTO command cannot be followed by a semi-colon and other commands. It must be ended with a CR.

GOTO A*10+B

will cause the execution to jump to a different statement number as computed from the value of the expression.

GOSUB and RETURN Commands

GOSUB command is similar to GOTO command

SOFTWARE SECTION

except that: a) the current statement number and position within the statement is remembered; and b) a semi-colon and other commands can follow it in the same statement.

GOSUB 120

will cause the execution to jump to statement 120.

GOSUB A*10+B

will cause the execution to jump to different statements as computed from the value of the expression $A * 10 + B$.

RETURN

A RETURN command must be the last command in a statement and followed by a CR. When a RETURN command is encountered, it will cause the execution to jump back to the command following the most recent GOSUB command.

GOSUB can be nested. The depth of nesting is limited only by the stack space.

LIST

will print out all the statements in numerical order.

LIST 120

will print out all the statements in numerical order 120.

NEW

will delete all the statements.

Stopping the Execution

The execution of program or listing of program can be stopped by the Control-C key on the input device.

Abbreviation and blanks

You may use blanks freely, except that numbers, command key words, and function names can not have embedded blanks.

You can truncate all command key words and function names and follow each by a period. "P.", "PR.", "PRI.", and "PRIN." all stand for "PRINT." Also the word LET in LET command can be omitted. The "shortest" abbreviation for all the key words are as follows:

A.=ABS	F.=FOR	GOS.=GOSUB	G.=GOTO
IF=IF	IN.=INPUT	L.=LIST	N.=NEW
N.=NEXT	P.=PRINT	REM=REMARK	R.=RETURN
R.=RND	R.=RUN	S.=SIZE	S.=STEP
S.=STOP	TO=TO		

Implied=LET

Control of Output Device

The Control-O key on the input device can be used to turn the output device ON and OFF. This is useful when you want to read in a program punched on paper tape.

To produce such a paper tape, type "LIST" without CR. Turn on the paper tape punch and type a few Control-Shift-P's and then a CR. When listing is finished, type more Control-Shift-P's and turn off the punch.

To read back such a paper tape, type "NEW," CR, and Control-O, then turn on the paper tape reader.

MICROCOMPUTER DEVELOPMENT SOFTWARE

When the paper tape is finished, turn it off and type a Control-O again.

Control-Shift-P's and turn off the punch.

To read back such a paper tape, type "NEW," CR, and Control-?, then turn on the paper tape reader. When the paper tape is finished, turn it off and type a Control-O, then turn on the paper tape reader. When the paper tape is finished, turn it off and type a Control-O again.

Error Report

There are only three error conditions in TINY BASIC. The statement with the error is printed out with a question mark inserted at the point where the error is detected.

(1) **WHAT?** means it does not understand you. Example:

WHAT? 210 P?TINT "THIS" where PRINT is mistyped

WHAT? 260 LET A=B+3, C=(3+4), X=4

(2) **HOW?** means it understands you but does not know how to do it.

HOW? 310LET A=B*C?+2 where B*C is larger than 32767

HOW? 380 GOTO 412? where 412 dose not exist

(3) **SORRY** means it understands you and knows how to do it but there is not enough memory to do it.

Error Corrections

If you notice an error in typing before you hit the CR, you can delete the last character by the Rub-Out key or delete the entire line by the Alt-Mode key. Tiny basic will echo a back-slash for each Rub-Out. Echo for Alt-Mode consists of a LF, a CR, and an up-arrow.

To correct a statement, you can retype the statement number and the correct commands. Tiny Basic will replace the old statement with the new one.

To delete a statement, type the statement number and a CR only.

Verify the corrections by "LIST nnnn" and hit the Control-C key while the line is being printed.

FOR and NEXT Commands

FOR X=A+1 TO 3*B STEP C-1

The variable X is set to the value of the expression A+1. The values of the expressions (not the expressions themselves) 3*B and C-1 are remembered. The name of the variable X, the statement number and the position of this command within the statement are also remembered. Execution then continues the normal way until a NEXT command is encountered.

The STEP can be positive, negative or even zero. The word STEP and the expression following it can be omitted if the desired STEP is +1.

NEXT X

The name of the variable (X) is checked with that of the most recent FOR command. If they do not agree, that FOR is terminated and the next recent FOR is checked, etc. When a match is found, this variable will be set to its current value plus the value of the STEP expression saved by the FOR command. The updated

value is then compared with the value of the TO expression also saved by the FOR command. If this is within the limit, execution will jump back to the command following the FOR command. If this is outside the limit, execution continues following the NEXT command itself.

FOR can be nested. The depth of nesting is limited only by the stack space. If a new FOR command with the same control variable as that of an old FOR command is encountered, the old FOR will be terminated automatically.

STOP Command

STOP

This command stops the execution of the program and returns control to direct commands from the input device. It can appear many times in a program but must be the last command in any given statement, i.e., it cannot be followed by semi-colon and other commands.

Direct Commands

As defined earlier, a statement consists of a statement number followed by commands. If the statement number is missing, or if it is 0, the commands will be executed after you have typed the CR. All the commands described above can be used as direct commands. There are three more commands that can be used as direct command but not as part of a statement:

RUN

will start to execute the program starting at the lowest statement number.

See Microcomputer Software Depository Program Index for copies of this program.

```

***** TINY BASIC FOR INTEL 8080 *****

TINY BASIC FOR INTEL 8080
VERSION 2.0
BY LI-CHEN WANG
MODIFIED AND TRANSLATED
TO INTEL MNEMONICS
BY ROGER RAUSCHOL
10 OCTOBER 1976
(C)OPYLEFT
ALL WRONGS RESERVED

***** ZERO PAGE SUBROUTINES ***
THE 8080 INSTRUCTION SET LETS YOU HAVE 8 ROUTINES IN LOW
MEMORY THAT MAY BE CALLED BY RST N. N BEING 0 THROUGH 7.
THIS IS A ONE BYTE INSTRUCTION AND HAS THE SAME POWER AS
THE THREE BYTE INSTRUCTION CALL LLH. TINY BASIC WILL
USE RST 0 AS START AND RST 1 THROUGH RST 7 FOR
THE SEVEN MOST FREQUENTLY USED SUBROUTINES.
TWO OTHER SUBROUTINES (CRLF AND TSTNUM) ARE ALSO IN THIS
SECTION. THEY CAN BE REACHED ONLY BY 3-BYTE CALLS.

DWB MACRO WHERE
DB WHERE SHR 8 +128
DB WHERE AND 0FFH
ENDM
ORG 0H

0000 START:
LXI SP,STACK :*** COLD START ***
MVI A,0FFH
JMP INIT

0008 E2 XTHL :*** TSTC OR RST 1 ***
0009 EF RST 5 :IGNORE BLANKS AND
000A BE CMP M :TEST CHARACTER
000B C36000 JMP TC1 :REST OF THIS IS AT TC1

000E 3E00 CRLF: MVI A,CR :*** CRLF ***
0010 F5 PUSH PSW :*** OUTC OR RST 2 ***
0011 300010 LDI OC5H :PRINT CHARACTER ONLY
0014 B7 DRA A :IF OC5H SWITCH IS ON
0015 C36006 JMP OC2 :REST OF THIS IS AT OC2

0018 CD7102 CALL EXPR2 :*** EXPR OR RST 3 ***
0019 E5 PUSH H :EVALUATE BN EXPRESSION
001C C32003 JMP EXPR1 :REST OF IT IS AT EXPR1

0020 7C MOV A,H :*** COMP OR RST 4 ***
0021 8A CMP D :COMPARE HL WITH DE
0022 C0 RNZ :RETURN CORRECT C AND
0023 7D MOV A,L :Z FLAGS
0024 BB CMP E :BUT OLD A IS LOST
0025 C9 RET
0026 414E DB 'BN' :*** IGNBLK/RST 5 ***
0028 1A SS1: LDAX D :IN TEXT, WHERE DE->
0029 FE20 CPI 20H :IGNORE BLANKS
002B C0 RNZ :IN TEXT, WHERE DE->
002C 13 INX D :AND RETURN THE FIRST
002D C32800 JMP SS1 :NON-BLANK CHAR. IN A

0030 F1 POP PSW :*** FINISH/RST 6 ***
0031 C0B304 CALL FIN :CHECK END OF COMMAND
0034 C3C604 JMP QMHRAT :PRINT "WHAT?" IF WRONG
0037 47 DE 'G' :TEXT?

0038 EF RST 5 :*** TSTV OR RST 7 ***
0039 D640 SUI 40H :TEST VARIABLES
003B D8 RC :C NOT A VARIABLE
003C C25800 JNZ TV1 :NOT "B" ARRAY
003F 13 INX D :IT IS THE "0" ARRAY
0040 CD1A04 CALL PRN :B SHOULD BE FOLLOWED
0043 29 DAD H :BY (EXPR) AS ITS INDEX?
0044 DH9F00 JC OHOW :IS INDEX TOO BIG?
0047 D5 PUSH D :WILL IT OVERWRITE
0048 EB XCHD :TEXT?
0049 CD5904 CALL SIZE :FIND SIZE OF FREE
004C E7 RST 4 :AND CHECK THAT
004D DHF404 JC RSORRY :IF SO, SAY "SORRY"
0050 216613 LXI H,VARBGN :IF NOT, GET ADDRESS
0053 CD7C04 CALL SUBCE :OF @EXPR) AND PUT IT
0056 D1 POP D :IN HL
0057 C9 RET :C FLAG IS CLEARED
0058 FE1B TV1: CPI 1BH :NOT @. IS IT A TO Z?
005A 3F CMC :IF NOT RETURN C FLAG
005B D8 RC :RC
005C 13 INX D :IF A THROUGH Z
005D 216613 LXI H,VARBGN :COMPUTE ADDRESS OF
0060 07 RLC :THAT VARIABLE
0061 85 ADD L :AND RETURN IT IN HL
0062 6F MOV L,A :WITH C FLAG CLEARED
0063 3E00 MVI A,B
0065 8C ADC H
0066 67 MOV H,A
0067 C9 RET

TSTC XCH HL,(SP) :*** TSTC OR RST 1 ***
IGNBLK THIS IS AT LOC. 8
CMP M AND THEN JMP HERE

0068 23 TC1: INX H :COMPARE THE BYTE THAT
0069 CA7300 JZ TC2 :FOLLOWS THE RST INST.
006C C5 PUSH B :WITH THE TEXT <DE>
006D 4E MOV C,M :IF NOT =, ADD THE 2ND
006E 6600 MVI B,0 :BYTE THAT FOLLOWS THE
0070 09 DAD B :RST TO THE OLD PC
0071 C1 POP B :I.E., DO A RELATIVE
0072 1B DCX D :JUMP IF NOT =
0073 12 INX D :IF =, SKIP THOSE BYTES
0074 23 TC2: INX H :AND CONTINUE
0075 E2 XTHL
0076 C9 RET

TSTNUM:
0077 210000 LXI H,0 :*** TSTNUM ***
0078 44 MOV B,H :TEST IF THE TEXT IS
007B EF RST 5 :A NUMBER
007C FE30 TN1: CPI 30H :IF NOT, RETURN 0 IN
007E D8 PUSH B :B AND HL
007F FE3A RC :IF NUMBERS, CONVERT
0081 D0 RNC :TO BINARY IN HL AND
0082 3EF0 MVI A,0FH :SET A TO # OF DIGITS
0084 A4 PNA H :IF H>255, THERE IS NO
0085 C29F00 JNZ OHON :ROOM FOR NEXT DIGIT
0088 04 INR B :B COUNTS # OF DIGITS
0089 C5 PUSH B :HL=10*HL+(NEW DIGIT)
008A 44 MOV B,H :WHERE 10* IS DONE BY
008B 4D MOV C,L :SHIFT AND ADD
008C 29 DAD H :WHERE 10* IS DONE BY
008D 29 DAD H :SHIFT AND ADD
008E 09 DAD B :DAD
008F 29 DAD H :DAD
0090 1A LDAX D :AND <DIGIT> IS FROM
0091 13 INX D :STRIPPING THE ASCII
0092 E60F ANI 0FH :CODE
0094 85 ADD L :ADD L
0095 6F MOV L,A :MOV L,A
0096 3E00 MVI A,0 :ADC H
0098 8C ADC H :MOV H,A
0099 67 MOV H,A :POP B
009A C1 PSH B :LDAX D :DO THIS DIGIT AFTER
009C F27C00 JP TN1 :DIGIT 5 SAYS OVERFLOW
009F D5 OHOW: PUSH D :*** ERROR: "HOW?" ***
00A0 11A600 AHOW: LXI D, HOW
00A3 C3C004 JMP ERROR

00A6 484F573F HOW: DB 'HOW?' :*** MAIN ***
00A8 00 DB CR
00B0 4F4B OK: DB 'OK'
00B0 00 DB CR
00B4 57484154 WHAT: DB 'WHAT?'
00B2 3F
00B3 00 DB CR
00B4 524F5252 SORRY: DB 'SORRY'
00B8 59 DB CR

***** MAIN *****
THIS IS THE MAIN LOOP THAT COLLECTS THE TINY BASIC PROGRAM
AND STORES IT IN THE MEMORY.

AT START, IT PRINTS OUT "(CR)OK(CR)", AND INITIALIZES THE
STACK AND SOME OTHER INTERNAL VARIABLES. THEN IT PROMPTS
">" AND READS A LINE. IF THE LINE STARTS WITH A NON-ZERO
NUMBER, THIS NUMBER IS THE LINE NUMBER. THE LINE NUMBER
(IN 16 BIT BINARY) AND THE REST OF THE LINE (INCLUDING CR)
IS STORED IN THE MEMORY. IF A LINE WITH THE SAME LINE
NUMBER IS ALREADY THERE, IT IS REPLACED BY THE NEW ONE. IF
THE REST OF THE LINE CONSISTS OF A CR ONLY, IT IS NOT STORED
AND ANY EXISTING LINE WITH THE SAME LINE NUMBER IS DELETED.

AFTER A LINE IS INSERTED, REPLACED, OR DELETED, THE PROGRAM
LOOPS BACK AND ASKS FOR ANOTHER LINE. THIS LOOP WILL BE
TERMINATED WHEN IT READS A LINE WITH ZERO OR NO LINE
NUMBER, AND CONTROL IS TRANSFERRED TO "DIRECT".

```

```

; TINY BASIC PROGRAM SAVE AREA STARTS AT THE MEMORY LOCATION
; LABELED "TXTBGN" AND ENDS AT "TXTEND". WE ALWAYS FILL THIS
; AREA STARTING AT "TXTBGN", THE UNFILLED PORTION IS POINTED
; BY THE CONTENT OF A MEMORY LOCATION LABELED "TXTUNF".
;
; THE MEMORY LOCATION "CURRNT" POINTS TO THE LINE NUMBER
; THAT IS CURRENTLY BEING INTERPRETED. WHILE WE ARE IN
; THIS LOOP OR WHILE WE ARE INTERPRETING A DIRECT COMMAND
; (SEE NEXT SECTION), "CURRNT" SHOULD POINT TO A 0.
;
; RSTART LODI SP,STACK
;
008A 310014    LXI SP,STACK
008D C0E000    ST1: CALL CRLF ; AND JUMP TO HERE
00C0 11AB00    LXI D,OK ; DE->STRING
00C3 97        SUB A ; A=0
00C4 CD6005    CALL PRSTG ; PRINT STRING UNTIL CR
;
00C7 21CE00    LXI H,ST+1 ;LITERAL 0
00CA 220110    SHLD CURRNT ;CURRNT->LINE # = 0
00CD 210000    ST2: LXI H,0
00D0 220910    SHLD LOPVAR
00D3 220310    SHLD STKGOS
00D6 3E3E      ST3: MVJ A,3EH ;PROMPT '>' AND
00D8 CDF040    CALL GETLN ;READ A LINE
00D8 D5        PUSH D ;DE->END OF LINE
00DC 119D13    LXI D,BUFFER ;DE->BEGINNING OF LINE
00DF CD7700    CALL TSTNUM ;TEST IF IT IS A NUMBER
00E2 EF        RST 5
00E3 7C        MOV A,H ;HL=VALUE OF THE # OR
00E4 B5        ORA L ;0 IF NO # WAS FOUND
00E5 C1        POP B ;BC->END OF LINE
00E6 CA3807    JZ RSTART
00E9 1B        DCK D ;BACKUP DE AND SAVE
00EA 7C        MOV A,H ;VALUE OF LINE # THERE
00EB 12        STAX D
00EC 1B        DCK D
00ED 7D        MOV A,L
00EE 12        STAX D
00EF C5        PUSH B ;BC,DE->BEGIN, END
00F0 D5        PUSH D
00F1 79        MOV A,C
00F2 92        SUB E
00F3 F5        PUSH PSW ;A=# OF BYTES IN LINE
00F4 CD3805    CALL FNDLN ;FIND THIS LINE IN SAVE
00F7 D5        PUSH D ;AREA, DE->SAVE AREA
00F8 C20B01    JNZ ST4 ;NZ NOT FOUND, INSERT
00FB D5        PUSH D ;Z FOUND, DELETE IT
00FC CD5405    CALL FNDNXT ;FIND NEXT LINE
;
; DE->NEXT LINE
;
00F7 C1        POP B ;BC->LINE TO BE DELETED
0100 2A1510    LHLD TXTUNF ;HL->UNFILLED SAVE AREA
0103 CDE505    CALL MVUP ;MOVE UP TO DELETE
0104 50        MOV H,B ;TXTUNF->UNFILLED AREA
0107 69        MOV L,C
0108 221510    SHLD TXTUNF ;UPDATE
0108 C1        ST4: POP B ;GET READY TO INSERT
010C 2A1510    LHLD TXTUNF ;BUT FIRST CHECK IF
010F F1        POP PSW ;THE LENGTH OF NEW LINE
0110 E5        PUSH H ;IS 3 (LINE # AND CR)
0111 FE03    CPJ 3 ;THEN DO NOT INSERT
0112 CRBA00    JZ RSTART ;MUST CLEAR THE STACK
0116 85        ADD L,A ;COMPUTE NEW TXTUNF
0117 6F        MOV L,A
0118 2E00    MVI A,0
011A 8C        ADC H
011B 67        MOV H,A ;HL->NEW UNFILLED AREA
011C 116613    LXI D,TXTEND ;CHECK TO SEE IF THERE
;
011F E7        RST 4 ;IS ENOUGH SPACE
0120 D2F304    JNC QSORRY ;SORRY, NO ROOM FOR IT
0123 221510    SHLD TXTUNF ;OK, UPDATE TXTUNF
0126 D1        POP D ;DE->OLD UNFILLED AREA
0127 CDE005    CALL MVDOWN
012A D1        POP D ;DE->BEGIN, HL->END
012B E1        POP H
012C CDE505    CALL MVUP ;MOVE NEW LINE TO SAVE
012F C3D600    JMP ST3 ;AREA
;
; *****
;
; WHAT FOLLOWS IS THE CODE TO EXECUTE DIRECT AND STATEMENT
; COMMANDS. CONTROL IS TRANSFERRED TO THESE POINTS VIA THE
; COMMAND TABLE LOOKUP CODE OF 'DIRECT' AND 'EXEC' IN LAST
; SECTION. AFTER THE COMMAND IS EXECUTED, CONTROL IS
; TRANSFERRED TO OTHER SECTIONS AS FOLLOWS:
;
; FOR 'LIST', 'NEW', AND 'STOP': GO BACK TO 'RSTART'.
; FOR 'RUN': GO EXECUTE THE FIRST STORED LINE IF ANY; ELSE
; GO BACK TO 'RSTART'.
; FOR 'GOTO' AND 'GOSUB': GO EXECUTE THE TARGET LINE.
; FOR 'RETURN' AND 'NEXT': GO BACK TO SAVED RETURN LINE.
; FOR ALL OTHERS: IF 'CURRNT' -> 0, GO TO 'RSTART'; ELSE
; GO EXECUTE NEXT COMMAND. (THIS IS DONE IN 'FINISH'.)
;
; *****
;
; *** NEW *** STOP *** RUN & FRIENDS *** & GOTO ***
;
; 'NEW(CR)' SETS 'TXTUNF' TO POINT TO 'TXTBGN'
;
; 'STOP(CR)' GOES BACK TO 'RSTART'
;
; 'RUN(CR)' FINDS THE FIRST STORED LINE, STORE ITS ADDRESS (IN
; 'CURRNT'), AND START EXECUTE IT. NOTE THAT ONLY THOSE
; COMMANDS IN TAB2 ARE LEGAL FOR STORED PROGRAM.
;
; THERE ARE 3 MORE ENTRIES IN 'RUN'.
; 'RUNNXL' FINDS NEXT LINE, STORES ITS ADDR. AND EXECUTES IT.
; 'RUNTSL' STORES THE ADDRESS OF THIS LINE AND EXECUTES IT.
; 'RUNNSML' CONTINUES THE EXECUTION ON SAME LINE.
;
; 'GOTO EXPRESSION(CR)' EVALUATES THE EXPRESSION, FIND THE TARGET
; LINE, AND JUMP TO 'RUNTSL' TO DO IT.
;
0122 CDC204    NEW: CALL ENDCHK ;*** NEW(CR) ***
0125 211710    LXI H,TXTBGN
0128 221510    SHLD TXTUNF
;
013B CDC204    STOP: CALL ENDCHK ;*** STOP(CR) ***
013E C2E000    JMP RSTART
;
0141 CDC204    RUN: CALL ENDCHK ;*** RUN(CR) ***
0144 111710    LXI D,TXTBGN ;FIRST SAVED LINE
;
; RUNNXL:
; LXI H,0 ;*** RUNNXL ***
; CALL FNDLR ;FIND WHATEVER LINE #
; JC RSTART ;C-PASSED TXTUNF, QUIT
;
; RUNTSL:
; XCHG ;*** RUNTSL ***
; SHLD CURRNT ;SET 'CURRNT'->LINE #
;
0150 EB        0151 220110    XCHG
0154 EB        0155 12        INX D ;BUMP PASS LINE #
0156 12        0157 12        INX D
;
0157 CD5406    CALL CHK10 ;*** RUNTSL ***
015A 21D006    LXI H,TB2-1 ;FIND COMMAND IN TAB2
015D C33B07    JMP EXEC ;AND EXECUTE IT
;
0160 DF        GOTO: RST 3 ;*** GOTO EXPRESSION ***
0161 D5        PUSH D ;SAVE FOR ERROR ROUTINE
0162 CDC204    CALL ENDCHK ;MUST FIND A CR
0165 C02805    CALL FNDLN ;FIND THE TARGET LINE
0168 C2A000    JNZ AHOW ;NO SUCH LINE #
016B F1        POP PSW ;CLEAR THE "PUSH DE"
016C C35001    JMP RUNTSL ;GO DO IT
;
; *****
;
; *** LIST *** & PRINT ***
;
; LIST HAS TWO FORMS.
; 'LIST(CR)' LISTS ALL SAVED LINES.
; 'LIST #(CR)' START LIST AT THIS LINE #.
; YOU CAN STOP THE LISTING BY CONTROL C KEY.
;
; PRINT COMMAND IS 'PRINT ...' OR 'PRINT ... (CR)'
; WHERE '...' IS A LIST OF EXPRESSIONS, FORMATS, BACK-
; ARROWS, AND STRINGS. THESE ITEMS ARE SEPARATED BY COMMAS.
;
; A FORMAT IS A FOUND SIGN FOLLOWED BY A NUMBER. IT CONTROLS
; THE NUMBER OF SPACES THE VALUE OF A EXPRESSION IS GOING TO
; BE PRINTED. IT STAYS EFFECTIVE FOR THE REST OF THE PRINT
; COMMAND UNLESS CHANGED BY ANOTHER FORMAT. IF NO FORMAT IS
; SPECIFIED, 6 POSITIONS WILL BE USED.
;
; A STRING IS QUOTED IN A PAIR OF SINGLE QUOTES OR A PAIR OF
; DOUBLE QUOTES.
;
; A BACK-ARROW MEANS GENERATE A (CR) WITHOUT (LF).
;
; A (CRLF) IS GENERATED AFTER THE ENTIRE LIST HAS BEEN
; PRINTED OR IF THE LIST IS A NULL LIST. HOWEVER IF THE LIST
; ENDED WITH A COMMA, NO (CRLF) IS GENERATED.
;
; LIST: CALL TSTNUM ;TEST IF THERE IS A #
;       CALL ENDCHK ;IF NO # WE GET A #
;       CALL FNDLN ;FIND THIS OR NEXT LINE
;       LS1: JC RSTART ;C-PASSED TXTUNF
;             CALL PRTLN ;PRINT THE LINE
;             CALL CHK10 ;STOP IF HIT CONTROL-C
;             CALL FNDLR ;FIND NEXT LINE
;             JMP LS1 ;AND LOOP BACK
;
016F CD7700    LIST: CALL TSTNUM ;TEST IF THERE IS A #
0172 CDC204    CALL ENDCHK ;IF NO # WE GET A #
0175 C03B05    CALL FNDLN ;FIND THIS OR NEXT LINE
0178 DDBA00    LS1: JC RSTART ;C-PASSED TXTUNF
017B CDD205    CALL PRTLN ;PRINT THE LINE
017E CDC406   ;STOP IF HIT CONTROL-C
0181 CD4005    CALL FNDLR ;FIND NEXT LINE
0184 C37801    JMP LS1 ;AND LOOP BACK
;
; PRINT: MVJ C,6 ;C = # OF SPACES
;       RST 1 ;IF NULL LIST & ","
;       DB 3BH
;       PR2-$-1
;       CALL CRLF ;GIVE CR-LF AND
;       JMP RUNTSL ;CONTINUE SAME LINE
;
0192 C01CF     PR2: RST 1 ;IF NULL LIST (CR)
0193 9D        DB CR
0194 9E        DB PR8-$-1
0195 C00E00    CALL CRLF ;ALSO GIVE CR-LF AND
0198 C34701    JMP RUNNXL ;GO TO NEXT LINE
019B CF        PR0: RST 1 ;ELSE IS IT FORMAT?
019C 23        DB '#'
019D 95        DB PR1-$-1
019E DF        RST 3 ;YES, EVALUATE EXPRESSION
019F 4D        MOV C,L ;AND SAVE IT IN C
01A0 C3A901    JMP PR2 ;LOOK FOR MORE TO PRINT
01A2 C6C005    PR1: CALL OTSTG ;IS IT A STRING?
01A6 C3B601    JMP PR2 ;IF NOT, MUST BE EXPRESSION
01A9 CF        PR3: RST 1 ;IF "", GO FIND NEXT
01AA 2C        DB '/'
01B0 96        DB PR6-$-1
01B1 C0B204    CALL FIN ;IN THE LIST
01B5 C39801    JMP PR0 ;LIST CONTINUES
01B6 CD0E00    PR6: CALL CRLF ;LIST ENDS
01B8 F7        RST 6
01B9 DF        PR8: RST 3 ;EVALUATE THE EXPRESSION
01B7 C5        PUSH B
01B9 CD9205    CALL PRTNUM ;PRINT THE VALUE
01BB C1        POP B
01BC C3A901    JMP PR3 ;MORE TO PRINT?
;
; *****
;
; *** GOSUB *** & RETURN ***
;
; 'GOSUB EXPRESSION' OR 'GOSUB EXPRESSION (CR)' IS LIKE THE 'GOTO'
; COMMAND, EXCEPT THAT THE CURRENT TEXT POINTER, STACK POINTER
; ETC. ARE SAVED SO THAT EXECUTION CAN BE CONTINUED AFTER THE
; SUBROUTINE 'RETURN'. IN ORDER THAT 'GOSUB' CAN BE NESTED
; (AND EVEN RECURSIVE), THE SAVE AREA MUST BE STACKED.
; THE STACK POINTER IS SAVED IN 'STKGOS'. THE OLD 'STKGOS' IS
; SAVED IN THE STACK. IF WE ARE IN THE MAIN ROUTINE, 'STKGOS' IS
; ZERO (THIS WAS DONE BY THE "MAIN" SECTION OF THE CODE),
; BUT WE STILL SAVE IT AS A FLAG FOR NO FURTHER 'RETURN'S.
;
; 'RETURN(CR)' UNDOES EVERYTHING THAT 'GOSUB' DID, AND THUS
; RETURN THE EXECUTION TO THE COMMAND AFTER THE MOST RECENT
; 'GOSUB'. IF 'STKGOS' IS ZERO, IT INDICATES THAT WE NEVER HAD A 'GOSUB' AND IS THUS AN ERROR.
;
; GOSUB: CALL PUSHA ;SAVE THE CURRENT "FOR"
;       RST 3 ;PARAMETERS
;       PUSH D ;AND TEXT POINTER
;       CALL FNDLN ;FIND THE TARGET LINE
;       01C1 2A0000    JNZ AHOW ;NOT THERE, SAY "HON?"
;       LHLD CURRNT ;FIND IT, SAVE OLD
;       01C1 2A0110    LHLD STKGOS ;OLD 'CURRNT' OLD 'STKGOS'
;       01C1 E5        PUSH H ;'CURRNT' OLD 'STKGOS'
;       01C1 E5        PUSH H
;       LXI H,0 ;AND LOAD NEW ONES
;       01D5 220910    SHLD LOPVAR
;       01D8 39        DAD SP
;       01D9 220310    SHLD STKGOS
;       01DC C35001    JMP RUNTSL ;THEN RUN THAT LINE
;
; RETURN:
;       CALL ENDCHK ;THERE MUST BE A CR
;       LHLD STKGOS ;OLD STACK POINTER
;       MOV A,H ;0 MEANS NOT EXIST
;       01E5 7C        ORA L
;       01E6 B5        JZ ONHAT ;SO, WE SAY: "WHAT?"
;       01E7 CRC604    SPHL ;ELSE, RESTORE IT
;       01E8 E1        POP H
;       01E9 220310    SHLD CURRNT ;AND THE OLD 'CURRENT'
;       01F0 220110    POP D ;OLD TEXT POINTER
;       01F1 D1        CALL POPA ;OLD "FOR" PARAMETERS
;       01F4 CDFD05    01F7 F7        RST 6 ;AND WE ARE BACK HOME
;
; *****
;
```

```

; *** FOR *** & NEXT ***
; 'FOR' HAS TWO FORMS:
; 'FOR VAR=EXP1 TO EXP2 STEP EXP3' AND 'FOR VAR=EXP1 TO EXP2'
; THE SECOND FORM MEANS THE SAME THING AS THE FIRST FORM WITH
; EXP3=1, (I.E., WITH A STEP OF +1.)
; TBI WILL FIND THE VARIABLE VAR, AND SET ITS VALUE TO THE
; CURRENT VALUE OF EXP1. IT ALSO EVALUATES EXP2 AND EXP3
; AND SAVE ALL THESE TOGETHER WITH THE TEXT POINTER ETC. IN
; THE 'FOR' SAVE AREA, WHICH CONSISTS OF 'LOPVAR', 'LOPINC',
; 'LOPLMT', 'LOPLN', AND 'LOPPT'. IF THERE IS ALREADY SOME-
; THING IN THE SAVE AREA (THIS IS INDICATED BY A NON-ZERO
; 'LOPVAR'), THEN THE OLD SAVE AREA IS SAVED IN THE STACK
; BEFORE THE NEW ONE OVERWRITES IT.
; TBI WILL THEN DIG IN THE STACK AND FIND OUT IF THIS SAME
; VARIABLE WAS USED IN ANOTHER CURRENTLY ACTIVE 'FOR' LOOP.
; IF THAT IS THE CASE, THEN THE OLD 'FOR' LOOP IS DEACTIVATED.
; (PURGED FROM THE STACK...)
;
; 'NEXT VAR' SERVES AS THE LOGICAL (NOT NECESSARILY PHYSICAL)
; END OF THE 'FOR' LOOP. THE CONTROL VARIABLE VAR IS CHECKED
; WITH THE 'LOPVAR'. IF THEY ARE NOT THE SAME, TBI DIGS IN
; THE STACK TO FIND THE RIGHT ONE AND PURGES ALL THOSE THAT
; DID NOT MATCH EITHER WAY. TBI THEN ADDS THE 'STEP' TO
; THAT VARIABLE AND CHECK THE RESULT WITH THE LIMIT. IF IT
; IS WITHIN THE LIMIT, CONTROL LOOPS BACK TO THE COMMAND
; FOLLOWING THE 'FOR'. IF OUTSIDE THE LIMIT, THE SAVE AREA
; IS PURGED AND EXECUTION CONTINUES.
01FB CD1906 FOR: CALL PUSHA ;SAVE THE OLD SAVE AREA
01FB CDA004 CALL SETVAL ;SET THE CONTROL VAR.
01FE 2B DCX H ;HL IS ITS ADDRESS
01FF 220910 SHLD LOPVAR ;SAVE THAT
0202 C11207 LXI H,TAB5-1 ;USE 'EXEC' TO LOOK
0205 C3B007 JMP EXEC ;FOR THE WORD 'TO'
0208 DF FR1: RST 3 ;EVALUATE THE LIMIT
0209 220010 SHLD LOPLMT ;SAVE THE LIMIT
020C C11907 LXI H,TAB6-1 ;USE 'EXEC' TO LOOK
020F C3B007 JMP EXEC ;FOR THE WORD 'STEP'
0212 DF FR2: RST 3 ;FOUND IT. GET STEP
0213 C11902 JLC FR4 ;NOT FOUND. SET TO 1
0216 C10100 FR2: LXI H,1H ;NOT FOUND. SET TO 1
0219 C20B10 FR4: SHLD LOPINC ;SAVE THAT TOO
021C C20B10 FR5: LHLD CURRNT ;SAVE CURRENT LINE #
021F C20B10 SHLD LOPLN ;AND TEXT POINTER
0222 EB XCHG ;AND TEXT POINTER
0223 221110 SHLD LOPPT ;FIND 'LOPVAR'
0226 010900 LXI B,0AH ;DIG INTO STACK TO
0229 0A0910 LHLD LOPVAR ;FIND 'LOPVAR'
0232 EB XCHG
;
0230 68 MOV H,B ;HL=0 NOW
0232 69 MOV L,B ;HERE IS THE STACK
0234 3E DB 3EH
;
0231 09 FR7: DAD B ;EACH LEVEL IS 10 DEEP
0233 7E MOV A,M ;GET THAT OLD 'LOPVAR'
0233 23 INX H
0234 B6 ORA M
0235 C45202 JZ FR8 ;0 SAYS NO MORE IN IT
0238 7E MOV A,M
0239 2B DCX H
023A B9 CMP D ;SAME AS THIS ONE?
023B C23102 JNZ FR7
023E 7E MOV A,M ;THE OTHER HALF?
023F B6 CMP E
0240 C23102 JNZ FR7
0242 EB XCHG ;YES, FOUND ONE
0244 210000 LXI H,0H ;TRY TO MOVE SP
0247 39 DAD SP
0248 44 MOV E,H
0249 4D MOV C,L
024A 210000 LXI H,0AH
024D 19 DAD D
024E CDE005 CALL MVDOWN ;AND PURGE 10 WORDS
0251 F9 SPHL ;IN THE STACK
0252 2A1110 FR8: LHLD LOPPT ;JOB DONE. RESTORE DE
0255 EE XCHG
0256 F7 RST 6 ;AND CONTINUE
;
0257 FF NEXT: RST 7 ;GET ADDRESS OF VAR.
0258 D4C604 JC 0WHAT ;NO VARIABLE, "WHAT?"
0259 220510 SHLD VARNXT ;YES, SAVE IT
025E D5 NX0: PUSH D ;SAVE TEXT POINTER
025F EB XCHG
0260 2A0910 LHLD LOPVAR ;GET VAR. IN 'FOR'
0263 7C MOV A,H
0264 B5 ORA L ;0 SAYS NEVER HAD ONE
0265 C4C704 JZ 0WHAT ;SO WE ASK: "WHAT?"
0268 E7 RST 4 ;ELSE WE CHECK THEM
0269 C47602 JZ NX3 ;OK, THEY AGREE
026C D1 PDP D ;NO, LET'S SEE
026D CDF005 CALL POPA ;PURGE CURRENT LOOP
0270 2B0510 LHLD VARNXT ;AND POP ONE LEVEL
0272 C5E002 JMP NX0 ;GO CHECK AGAIN
0276 5E NX3: MOV E,M ;COME HERE WHEN AGREED
0277 23 INX H
0278 56 MOV D,M ;DE=VALUE OF VAR.
0279 2B0B10 LHLD LOPINC ;OLD HL
027C E5 PUSH H
027D 7C MOV R,H
;
027E AA XRA D
027F 7A MOV A,D
0280 19 DAD D ;ADD ONE STEP
0281 F4B002 JM NX4
0284 AC XRA H
0285 F4A002 JM NX5
0288 EB NX4: XCHG
0289 2A0910 LHLD LOPVAR ;PUT IT BACK
028C 73 MOV M,E
028D 23 INX H
028E 72 MOV M,D
028F 2A0D10 LHLD LOPLMT ;HL=LIMIT
0292 F1 POP PSW ;OLD HL
0293 B7 ORA A
0294 F29802 JP NX1 ;STEP > 0
0297 EB XCHG
0298 C09804 NX1: CALL CKHLDE ;COMPARE WITH LIMIT
029B D1 POP D ;RESTORE TEXT POINTER
029C D4C002 JC NX2 ;OUTSIDE LIMIT
029F 2B0F10 LHLD LOPLN ;WITHIN LIMIT. GO
02A2 220110 SHLD CURRNT ;BACK TO THE SAVED
02A5 2A1110 LHLD LOPPT ;'CURRNT' AND TEXT
02A8 EB XCHG ;POINTER
02A9 F7 RST 6
02AA E1 NX5: POP H
02AB D1 POP D
02AC CDF005 NX2: CALL POPA ;PURGE THIS LOOP
02AF F7 RST 6

```

```

; **** REM *** IF *** INPUT *** & LET (& DEFLT) ***
;
; 'REM' CAN BE FOLLOWED BY ANYTHING AND IS IGNORED BY TBI.
; TBI TREATS IT LIKE AN 'IF' WITH A FALSE CONDITION.
;
; 'IF' IS FOLLOWED BY AN EXPR. AS A CONDITION AND ONE OR MORE
; COMMANDS (INCLUDING OTHER 'IF'S) SEPARATED BY SEMI-COLONS.
; NOTE THAT THE WORD 'THEN' IS NOT USED. TBI EVALUATES THE
; EXPR. IF IT IS NON-ZERO, EXECUTION CONTINUES. IF THE
; EXPR. IS ZERO, THE COMMANDS THAT FOLLOW ARE IGNORED AND
; EXECUTION CONTINUES AT THE NEXT LINE.
;
; 'INPUT' COMMAND IS LIKE THE 'PRINT' COMMAND, AND IS FOLLOWED
; BY A LIST OF ITEMS. IF THE ITEM IS A STRING IN SINGLE OR
; DOUBLE QUOTES, OR IS A BACK-ARROW, IT HAS THE SAME EFFECT AS
; IN 'PRINT'. IF AN ITEM IS A VARIABLE, THIS VARIABLE NAME IS
; PRINTED OUT FOLLOWED BY A COLON. THEN TBI WAITS FOR AN
; EXPR. TO BE TYPED IN. THE VARIABLE IS THEN SET TO THE
; VALUE OF THIS EXPR. IF THE VARIABLE IS PRECEDED BY A STRING
;
; (AGAIN IN SINGLE OR DOUBLE QUOTES), THE STRING WILL BE
; PRINTED FOLLOWED BY A COLON. TBI THEN WAITS FOR INPUT EXPR.
; AND SET THE VARIABLE TO THE VALUE OF THE EXPR.
;
; IF THE INPUT EXPR. IS INVALID, TBI WILL PRINT "WHAT?", "
; "HOW?" OR "SORRY" AND REPRINT THE PROMPT AND REDO THE INPUT.
; THE EXECUTION WILL NOT TERMINATE UNLESS YOU TYPE CONTROL-C.
; THIS IS HANDLED IN 'INPERR'.
;
; 'LET' IS FOLLOWED BY A LIST OF ITEMS SEPARATED BY COMMAS.
; EACH ITEM CONSISTS OF A VARIABLE, AN EQUAL SIGN, AND AN EXPR.
; TBI EVALUATES THE EXPR. AND SET THE VARIABLE TO THAT VALUE.
; TBI WILL ALSO HANDLE 'LET' COMMAND WITHOUT THE WORD 'LET'.
; THIS IS DONE BY 'DEFLT'.
;
; REM: LXI H,0H ;*** REM ***
; DB 3EH
;
; 02B0 210000 IFF: RST 3 ;*** IF ***
; 02B1 3E MOV A,H ;IS THE EXPR.=0?
;
; 02B4 DF 02B5 7C 02B6 B5 02B7 C25701 02B8 C05605 02B9 D25001 02B0 C3B009
; JNZ RUNSML ;NO, CONTINUE
; CALL FNDSPK ;YES, SKIP REST OF LINE
; JNC RUNTSL
; JMP RSTART
;
; INPERR: LHLD STKINP ;*** INPERR ***
; SPHL ;RESTORE OLD SP
; POP H ;AND OLD 'CURRNT'
; SHLD CURRNT
; POP D ;AND OLD TEXT POINTER
; POP D ;REDO INPUT
;
; INPUT: PUSH D ;SAVE IN CASE OF ERROR
; IP1: CALL QTSTG ;IS NEXT ITEM A STRING?
; 02C0 C06005 02C1 C3DB02 02C2 FF 02C3 7E 02C4 0A 02C5 DA1503 02C6 C3B002
; CALL QTSTG ;NO
; JMP IP2
; ORA L
; RST 7 ;YES, BUT FOLLOWED BY A
; JC IP4 ;VARIABLE? NO
; 02C7 015033 02C8 220110 02C9 D1 02C0 D1 02C1 D1
; JMP IP3 ;YES, INPUT VARIABLE
; 02C2 D5 02C3 FF 02C4 1A 02C5 0A 02C6 0B 02C7 01C002 02C8 220110
; PUSH D ;SAVE FOR 'PRTSTG'
; RST 7 ;MUST BE VARIABLE NOW
; JC 0WHAT ;"WHAT?" IT IS NOT?
; LDAX D ;GET READY FOR 'PRTSTG'
; 02C9 4F 02C1 97 02C2 12 02C3 D1 02C4 01
; MOV C,A
; SUB A
; STAX D
; POP D
;
; 02C5 C06005 CALL PRTSTG ;PRINT STRING AS PROMPT
;
; 02C6 79 02C7 1B 02C8 12 02C9 D5 02C0 210110 02C1 E5 02C2 220110
; MOV A,C ;RESTORE TEXT
; DCX D
; STRX D
; PUSH D ;SAVE TEXT POINTER
; LHLD CURRNT ;ALSO SAVE 'CURRNT'
; PUSH H
; 02C3 210000 02C4 220110 02C5 21C002 02C6 1A 02C7 01 02C8 0A
; RST 3 ;SAVE TEXT POINTER
; LXI H,IP1 ;A NEGATIVE NUMBER
; SHLD CURRNT ;AS A FLAG
; PUSH H
; 02C9 0B 02C0 0B 02C1 0B 02C2 0B 02C3 0B 02C4 0B 02C5 0B 02C6 0B
; SHLD CURRNT ;PRINT THIS TOO
; CALL GETLN ;AND GET A LINE
; LXI D,BUFFER ;POINTS TO BUFFER
; RST 3 ;EVALUATE INPUT
; NOP ;CAN BE 'CALL ENDCHK'
; 02C7 0013 02C8 00 02C9 00 02C0 00 02C1 00 02C2 00 02C3 00 02C4 00
; NOP
; NOP
; NOP
; POP D ;OK, GET OLD HL
; XCHG
; 02C5 119013 02C6 0B 02C7 0B 02C8 0B 02C9 0B 02C0 0B 02C1 0B 02C2 0B
; HVI A,3AH ;PRINT THIS TOO
; CALL GETLN ;AND GET A LINE
; LXI D,BUFFER ;POINTS TO BUFFER
; RST 3 ;EVALUATE INPUT
; NOP ;CAN BE 'CALL ENDCCHK'
; 02C3 0014 02C4 00 02C5 00 02C6 00 02C7 00 02C8 00 02C9 00 02C0 00
; POP D ;AND OLD TEXT POINTER
; 02C1 01 02C2 01 02C3 01 02C4 01 02C5 01 02C6 01 02C7 01 02C8 01
; PUSH D ;IS NEXT CH. ?,??
; 02C9 0B 02C0 0B 02C1 0B 02C2 0B 02C3 0B 02C4 0B 02C5 0B 02C6 0B
; DB IPS-$-1 ;YES, MORE ITEMS.
; 02C7 F7 02C8 0B 02C9 0B 02C0 0B 02C1 0B 02C2 0B 02C3 0B 02C4 0B
; IP5: PUSH D ;PURGE JUNK IN STACK
; 02C5 F7 DEFLT: LH1D 1A 02C6 FE00 02C7 01 02C8 0B 02C9 0B 02C0 0B
; CALL SETVAL ;*** LET ***
; RST 1 ;SET VALUE TO VAR.
; DB ??
; JZ LT1 ;ELSE IT IS 'LET'
;
; LET: CALL SETVAL ;*** LET ***
; RST 1 ;SET VALUE TO VAR.
; DB ??
; JZ LT1 ;ELSE IT IS 'LET'
;
; *** EXPR ***
;
; 'EXPR' EVALUATES ARITHMETICAL OR LOGICAL EXPRESSIONS.
; <EXPR> ::= <EXPR2><REL_OP><EXPR>
; WHERE <REL_OP> IS ONE OF THE OPERATORS IN TAB8 AND THE
; RESULT OF THESE OPERATIONS IS 1 IF TRUE AND 0 IF FALSE.
; <EXPR> ::= <C>+<EXPR2>+<C> OR -<EXPR2>+<C>-<EXPR3>+<C> ...
; WHERE <C> ARE OPTIONAL AND <...> ARE OPTIONAL REPEATS.
; <EXPR> ::= <EXPR4><C><C> OR <C><EXPR4><C><C> ...
; <EXPR4> ::= <VARIABLE>

```

```

    <FUNCTION>
    <EXPR2>
    <EXPR2> IS RECURSIVE SO THAT VARIABLE <Y> CAN HAVE AN <EXPR>
    AS INDEX. FUNCTIONS CAN HAVE AN <EXPR> AS ARGUMENTS, AND
    <EXPR4> CAN BE AN <EXPR> IN PARENTHES.
    <EXPR> CALL EXPR2   THIS IS AT LOC. 18
    <PUSH HL>      SAVE <EXPR2> VALUE
    0220 212107 EXPR1: LXI H,TAB8-1 ;LOOKUP REL OP.
    0220 C3B007 JNP EXEC ;GO DO IT
    0223 CD5C03 XP11: CALL XP18 ;REL OP."="
    0226 D8 RC ;NO, RETURN HL=0
    0227 6F MOV L,A ;YES, RETURN HL=1
    0228 C9 RET
    0229 CD5C03 XP12: CALL XP18 ;REL OP. "#"
    022C C8 R2 ;FALSE, RETURN HL=0
    023D 6F MOV L,A ;TRUE, RETURN HL=1
    023E C9 RET
    023F CD5C03 XP13: CALL XP18 ;REL OP. ">"
    0242 C8 RZ ;FALSE
    0243 D8 RC ;ALSO FALSE, HL=0
    0244 6F MOV L,A ;TRUE, HL=1
    0245 C9 RET
    0246 CD5C03 XP14: CALL XP18 ;REL OP. "<"
    0249 6F MOV L,A ;REL, TRUE, RETURN
    024A C8 RZ
    024B D8 RC
    024C 6C MOV L,H ;ELSE SET HL=0
    024D C9 RET
    024E CD5C03 XP15: CALL XP18 ;REL OP. "="
    0251 C0 RNZ ;FALSE, RETRUN HL=0
    0252 6F MOV L,A ;ELSE SET HL=1
    0253 C9 RET
    0254 CD5C03 XP16: CALL XP18 ;REL OP. "<="
    0257 D0 RNC ;FALSE, RETURN HL=0
    0258 6F MOV L,A ;ELSE SET HL=1
    0259 C9 RET
    025A E1 XP17: POP H ;NOT REL OP.
    025B C9 RET
    025C 79 XP18: MOV A,C ;SUBROUTINE FOR ALL
    025D E1 POP H ;REL OP. S
    025E C1 POP B
    025F E5 PUSH H ;REVERSE TOP OF STACK
    0260 C5 PUSH B
    0261 4F MOV C,A
    0262 CD7103 CALL EXPR2 ;GET 2ND <EXPR2>
    0265 EB XCHG ;VALUE IN DE NOW
    0266 E3 XTHL ;1ST <EXPR2> IN HL
    0267 CD9804 CALL CKHLDE ;COMPARE 1ST WITH 2ND
    0268 D1 POP D ;RESTORE TEXT POINTER
    026B 210000 LXI H,0H ;SET HL=0, A=1
    026E 3E01 HVI A,1
    0270 C9 RET
    0271 C9 EXPR2: RET L ;NEGATIVE SIGN?
    0272 20 DS 2 ;YES, FAKE 0-
    0273 00 DB XP21-$-1
    0274 C3B002 LXI H,0H ;YES, FAKE 0-
    0275 C3B002 JNP XP26 ;TREAT LIKE SUBTRACT
    0276 C9 XP21 RST 1 ;POSITIVE SIGN? IGNORE
    0278 2B DS 2 ;NO
    0279 00 DB XP22-$-1
    027D CD4501 XP22: CALL EXPR2 ;1ST <EXPR2>
    0280 C9 XP23: RST 1 ;ADD?
    0281 1B DS 2 ;NO
    0282 15 DB XP25-$-1
    0283 E5 PUSH H ;YES, SAVE VALUE
    0284 CD4502 CALL EXPR2 ;GET 2ND <EXPR2>
    0287 EB XCHG ;2ND IN DE
    0288 E3 XTHL ;1ST IN HL
    0289 7C MOV A,H ;COMPARE SIGN
    028A AA XRA D
    028B 7A MOV A,D
    028C 19 DAD D
    028D D1 POP D ;RESTORE TEXT POINTER
    028E FA8002 JM XP23 ;1ST 2ND SIGN DIFFER
    0291 AC HRA H ;1ST 2ND SIGN EQUAL
    0292 F28000 JP XP25 ;SO IS RESULT
    0295 C3F900 JNP OHOW ;ELSE WE HAVE OVERFLOW
    0298 C9 XP25: RST 1 ;SUBTRACT?
    0299 2D DS 2 ;NO
    029A 86 DB XP42-$-1
    029B E5 XP26: PUSH H ;YES, SAVE 1ST <EXPR2>
    029C CD4502 CALL EXPR2 ;GET 2ND <EXPR2>
    029F CD8604 CALL CHGSGN ;NEGATE
    0302 C3B703 JNP XP24 ;AND ADD THEM
    0295 CD8604 EXPR3: CALL EXPR4 ;GET 1ST <EXPR4>
    0298 C9 XP31: RST 1 ;MULTIPLY?
    0299 2D DB XP34-$-1
    029A 45 PUSH H ;YES, SAVE 1ST
    029C CD8604 CALL EXPR4 ;HAD GET 2ND <EXPR4>
    029F 0600 MVI B,0H ;CLEAR B FOR SIGN
    0301 C08704 CALL CHGSGN ;CHECK SIGN
    0294 E3 XTHL ;1ST IN HL
    0295 C3B704 TCHL ;CHGSGN ;CHECK SIGN OF 1ST
    0298 EB MVI H,0 ;1ST
    0299 E1 XTHL ;IS HL > 255 ?
    029B 71 MOV A,H ;NO
    029B E1 ORA A ;IS HL > 255 ?
    029C 71 JC XP22 ;NO
    029D 7A MOV A,D ;YES, HOW ABOUT DE
    029E 82 DAD D
    0301 EB XCHG ;PUT SMALLER IN HL
    0302 C2H000 JNZ RHOW ;ALSO >, WILL OVERFLOW
    0305 70 XP22: MOV A,L ;THIS IS DUNE
    0306 210000 LXI H,0H ;CLEAR RESULT
    0309 B7 ORA A ;ADD AND COUNT
    030A C3F703 JC XP25 ;OVERFLOW
    030D 19 XP22: DAD D
    030E 000000 JC RHOW ;OVERFLOW
    0301 20 DCR R
    0302 C2D003 JNZ XP23 ;FINISHED
    0305 C2F703 JNP XP25 ;FINISHED
    0308 C9 XP24: RST 1 ;DIVIDE?
    0309 2F DS 2 ;NO
    030A 45 DB XP42-$-1
    030B 45 PUSH H ;YES, SAVE 1ST <EXPR4>
    030C CD8604 CALL EXPR4 ;AND GET 2ND ONE
    030F 0600 MVI B,0H ;CLEAR B FOR SIGN
    0301 C08304 CALL CHGSGN ;CHECK SIGN OF 2ND
    0304 E3 XTHL ;GET 1ST IN HL
    0305 CD8604 CALL CHGSGN ;CHECK SIGN OF 1ST
    0308 EB XCHG ;1ST
    0309 E3 XTHL ;IS HL > 255 ?
    030A 7A MOV A,D ;DIVIDE BY 0?
    030C B2 ORA E ;SAY "HOW?"
    030F C5 PUSH B ;ELSE SAVE SIGN
    0301 C3D604 CALL DIVIDE ;USE SUBROUTINE
    0304 60 MOV H,B ;RESULT IN HL NOW
    0305 69 MOV L,C
    0306 C1 POP B ;GET SIGN BACK
    0307 00 DB XP25: POP D ;AND TEXT POINTER
    0308 7C MOV A,H ;HL MUST BE +
    0309 B7 ORA A
    030A F9F00 JM OHOW ;ELSE IT IS OVERFLOW
    030D 78 MOV A,B
    030E B7 ORA A
    030F FC8604 CM CHGSGN ;CHANGE SIGN IF NEEDED
    0302 C3B802 JNP XP31 ;LOOK FOR MORE TERMS
    0405 210107 EXPR4: LXI H,TAB4-1 ;FIND FUNCTION IN TAB4
    0408 C3B807 JNP EXEC ;AND GO DO IT
    040B FF XP40: RST 7 ;NO, NOT A FUNCTION
    040C D41404 JC XP41 ;NOR A VARIABLE
    040F 7E MOV A,M ;VALUE IN HL
    0410 23 INX H
    0411 66 MOV H,M
    0412 6F MOV L,A
    0413 C9 RET
    0414 C07700 XP41: CALL TSTNUM ;OR IS IT A NUMBER
    0417 78 MOV A,B ;# OF DIGIT
    0418 B7 ORA A
    0419 C8 RNZ ;OK
    041A CF PARN: RST 1 ;NO DIGIT, MUST BE
    041B 05 DB XP42-$-1
    041D DF RST 2 ;"<EXPR>""
    041E CF RST 1
    041F 29 DB XP42-$-1
    0420 01 DB XP42-$-1
    0421 C9 XP42: RET
    0422 C3C604 XP43: JNP OHWAT ;ELSE SAY: "WHAT?""
    0425 C01A04 RND: CALL PARN ;*** RND(<EXPR>) ***
    0428 7C MOV A,H ;<EXPR> MUST BE +
    0429 B7 ORA A
    042A F9F00 JM OHOW
    042D B5 ORA L ;AND NON-ZERO
    042E 05 PUSH D ;SAVE BOTH
    042F H PUSH H
    0432 291210 LHLD RANPNT ;GET MEMORY AS RANDOM
    0436 116907 LXI D,LSTROM ;NUMBER
    0439 E7 RST 4
    0440 D44004 JC RR1 ;WRAP AROUND IF LAST
    0442 D10000 LXI H,START
    0444 5E RR1: MOV A,H
    0445 23 INX H
    0446 56 MOV D,M
    0443 224210 SHLD RANPNT
    0446 E1 POP H
    0447 EB XCHG
    0448 C5 PUSH B
    0449 C0E604 CALL DIVIDE ;RND(N)=MOD(M,N)+1
    044C C1 POP B
    044D D1 POP D
    044E 23 INX H
    044F C9 RET
    0450 C01A04 RBS: CALL PARN ;*** ABS(<EXPR>) ***
    0452 1B DCX D
    0451 C08204 CALL CHGSGN ;CHECK SIGN
    0457 12 INX D
    0458 C9 RET
    0459 241510 SIZE: LHLD TXTUNF ;*** SIZE ***
    0460 D5 PUSH D ;GET THE NUMBER OF FREE
    0460 EB XCHG
    0458 216813 LXI H,VREGN ;BYTES BETWEEN 'TXTUNF' AND 'VREGN'
    0461 C07004 CALL SUBDE ;POP D
    0465 C9 RET
    ****
    *** DIVIDE *** SUBDE *** CHKSGN *** CHGSGN *** & CKHLDE ***
    DIVIDE DIVIDES HL BY DE, RESULT IN BC, REMAINDER IN HL
    SUBDE SUBTRACTS DE FROM HL
    CHKSGN CHECKS SIGN OF HL. IF +, NO CHANGE. IF -, CHANGE SIGN AND FLIP SIGN OF B
    CHGSGN CHANGES SIGN OF HL AND B UNCONDITIONALLY.
    CKHLDE CHECKS SIGN OF HL AND DE. IF DIFFERENT, HL AND DE ARE INTERCHANGED. IF SAME SIGN, NOT INTERCHANGED. EITHER CHSE, HL DE ARE THEN COMPARED TO SET THE FLAGS.
    DIVIDE:
    0466 E5 PUSH H ;*** DIVIDE ***
    0467 6C MOV L,H ;DIVIDE H BY DE
    0468 3E00 MVI H,0
    0469 C0194 CALL DV1
    046D 41 MOV B,C ;SAVE RESULT IN B
    046E 7D MOV R,L ;REMAINDER+L>/DE
    046F E1 POP H
    0470 6C MOV H,R
    0471 6EFF DV1: MVI C,6FF ;RESULT IN C
    0472 7C INR C ;DUMB ROUTINE
    0474 C07004 CALL SUBDE ;DIVIDE BY SUBTRACT
    0477 D27204 JNC DV2 ;AND COUNT
    0478 19 DAD D
    0478 C9 RET
    SUBDE: MOV A,L ;*** SUBDE ***
    SUB E ;SUBTRACT DE FROM
    CHKSGN:
    0482 7C MOV A,H ;*** CHKSGN ***
    0484 B7 ORA A ;CHECK SIGN OF HL
    0485 F0 RP ;IF -, CHANGE SIGN
    CHGSGN:
    0486 7C MOV A,H ;*** CHGSGN ***
    0487 F5 PUSH PSW
    0488 2F CMA ;CHANGE SIGN OF HL
    0489 67 MOV A,H
    048A 7D MOV A,L
    048B 2F CMA
    048C 6F MOV L,A
    048D 23 INX H
    048E F1 POP PSW
    048F AC XRA H
    0490 F29F00 JP OHOW
    0493 78 MOV A,B ;AND ALSO FLIP B
    0494 E690 XRI 80H
    0495 47 MOV B,A
    0497 C9 RET

```

SOFTWARE SECTION

MICROCOMPUTER DEVELOPMENT SOFTWARE

```

0198 7C CKHLD: MOV A, H
0199 AA XRA D ; SAME SIGN?
019A F2E04 JP CK1 ; YES, COMPARE
019D EB XCHG ; NO, XCH AND COMP
019E E7 CK1 RST 4
019F C9 RET

; **** SETVAL **** FIN ENDCHK *** & ERROR (& FRIENDS) ***
; "SETVAL" EXPECTS A VARIABLE, FOLLOWED BY AN EQUAL SIGN AND
; THEN AN EXPR. IT EVALUATES THE EXPR. AND SET THE VARIABLE
; TO THAT VALUE.
; "FIN" CHECKS THE END OF A COMMAND. IF IT ENDED WITH ";",
; EXECUTION CONTINUES. IF IT ENDED WITH A CR, IT FINDS THE
; NEXT LINE AND CONTINUE FROM THERE.
; "ENDCHK" CHECKS IF A COMMAND IS ENDED WITH CR. THIS IS
; REQUIRED IN CERTAIN COMMANDS. (GOTO, RETURN, AND STOP ETC.)
; "ERROR" PRINTS THE STRING POINTED BY DE (AND ENDS WITH CR).
; IT THEN PRINTS THE LINE POINTED BY 'CURRENT' WITH A "?"
; INSERTED AT WHERE THE OLD TEXT POINTER SHOULD BE ON TOP
; OF THE STACK) POINTS TO . EXECUTION OF TB IS STOPPED
; AND TBL IS RESTARTED. HOWEVER, IF 'CURRENT' => ZERO
; INDICATING A DIRECT COMMAND, THE DIRECT COMMAND IS NOT
; PRINTED. AND IF 'CURRENT' => NEGATIVE # (INDICATING 'INPUT'
; COMMAND), THE INPUT LINE IS NOT PRINTED AND EXECUTION IS
; NOT TERMINATED BUT CONTINUED AT 'INPERR'.
; RELATED TO 'ERROR' ARE THE FOLLOWING:
; "WHAT" SAVES TEXT POINTER IN STACK AND GET MESSAGE "WHAT?"
; "WHMR" JUST GET MESSAGE "WHAT?" AND JUMP TO 'ERROR'.
; "OSORRY" AND "ASORRY" DO SAME KIND OF THING.
; "OHOW" AND "AHOW" IN THE ZERO PAGE SECTION ALSO DO THIS

SETVAL
01A0 FF RST 7 ;*** SETVAL ***
01A1 DAC604 JC OHWAT ;"WHAT?" NO VARIABLE
01A1 E5 PUSH H ;SAVE ADDRESS OF VAR.
01A5 CF RST 1 ;PASS "=" SIGN
01A6 2D DB ?=?
01A7 89 DB SV1-#-1
01A8 DF RST 2 ;EVALUATE EXPR
01A9 4H MOV B, H ;VALUE IN BC NOW
01A9 4D MOV C, L
01A9 E1 POP H ;GET ADDRESS
01A9 74 MOV M, C ;SAVE VALUE
01AD 22 INX H
01AE 79 MOV M, B
01AF C9 RET

01B0 C3C604 SV1 JMP OHWAT ;NO "=" SIGN

01B1 CF FIN: RST 1 ;*** FIN ***
01B1 3B DB 3BH
01B5 04 DB F1L-#-1
01B6 F1 POP PSW ;";", PURGE RET ADDR
01B7 C5701 JMP RUNML ;CONTINUE SAME LINE
01B8 CF FI1: RST 1 ;NOT ";", IS IT CR?
01B8 00 DB CR
01B8 04 DB F12-#-1
01B8 F1 POP PSW ;YES, PURGE RET ADDR
01B8 C24701 JMP RUNNL ;RUN NEXT LINE
01C1 C9 FI2: RET ;ELSE RETURN TO CALLER

; ENDCHK:
01C2 EF RST 5 ;*** ENDCHK ***
01C2 FE00 CPI CR ;END WITH CR?
01C5 C8 RZ ;OK, ELSE SAY: "WHAT?" 

01C6 D5 OHWAT: PUSH D ;*** OHWAT ***
01C7 11AE00 WHRAT: LXI D, WHRT ;*** WHRAT ***
01CA 97 ERROR: SUB A ;*** ERROR ***
01CB CD6005 CALL PRTSTG ;PRINT "WHAT?", "HOW?"
01CE D1 POP D ;OR "SORRY"
01CF 1A LDAX D ;SAVE THE CHARACTER
01D0 F5 PUSH PSW ;AT WHERE OLD DE ->
01D1 97 SUB A ;AND PUT A 0 THERE
01D2 12 STAX D
01D2 3A0110 LHLD CURRENT ;GET CURRENT LINE #
01D6 E5 PUSH H
01D7 7E MOV A, M ;CHECK THE VALUE
01D8 22 INX H
01D9 B6 ORR M
01DA D1 POP D
01DB CAB000 JZ RSTART ;IF ZERO, JUST RESTART
01DE 7E MOV A, M ;IF NEGATIVE,
01DF B7 ORA A
01E0 FAC302 JM INPERR ;REDO INPUT
01E3 CDD005 CALL PRTLN ;ELSE PRINT THE LINE
01E6 1B DCX D ;UPTO WHERE THE 0 IS
01E7 F1 POP PSW ;RESTORE THE CHARACTER
01E8 12 STAX D
01E9 2E2F MWI A, 3FH ;PRINT A "?"
01EB D7 RST 2
01EC 97 SUB A ;AND THE REST OF THE
01ED CD6005 CALL PRTSTG
01F0 C3B000 JMP RSTART

01F2 D5 OSORRY: PUSH D ;*** OSORRY ***
01F3 A5 ASORRY: D. D-SORRY ;*** ASORRY ***
01F4 11B400 LXI D, SORRY ;*** ASORRY ***
01F7 C3C004 JMP ERROR

; **** GETLN *** FNDLN (& FRIENDS) ***
; "GETLN" READS A INPUT LINE INTO 'BUFFER'. IT FIRST PROMPT
; THE CHARTRER IN A (GIVEN BY THE CALLER), THEN IT FILLS THE
; THE BUFFER AND ECCHO'S IT IGNORES LF'S AND NULL'S, BUT STILL
; ECCHO'S THEM BACK. RUB-OUT IS USED TO CAUSE IT TO DELETE
; THE LAST CHARACTER (IF THERE IS ONE). AND ALT-MOD IS USED TO
; CAUSE IT TO DELETE THE WHOLE LINE AND START IT ALL OVER.
; CR TELLS THE END OF A LINE, AND CAUSE "GETLN" TO RETURN.
; "FNDLN" FINDS A LINE WITH A GIVEN LINE # (IN HL) IN THE
; TEXT SAVE AREA. DE IS USED AS THE TEXT POINTER. IF THE
; LINE IS FOUND, DE WILL POINT TO THE BEGINNING OF THAT LINE
; (I.E., THE LOW BYTE OF THE LINE #), AND FLAGS ARE NC & Z.
; IF THAT LINE IS NOT THERE AND A LINE WITH A HIGHER LINE #
; IS FOUND, DE POINTS TO THERE AND FLAGS ARE NC & NZ. IF
; WE REACHED THE END OF TEXT SAVE ARE AND CANNOT FIND THE
; LINE, FLAGS ARE C & NZ.
; "FNDLN" WILL INITIALIZE DE TO THE BEGINNING OF THE TEXT SAVE
; AREA TO START THE SEARCH. SOME OTHER ENTRIES OF THIS
; ROUTINE WILL NOT INITIALIZE DE AND DO THE SEARCH.
; "FNDNLP" WILL START WITH DE AND SEARCH FOR THE LINE #.
; "FNDNXT" WILL BUMP DE BY 2, FIND A CR AND THEN START SEARCH.
; "FNDSKP" USE DE TO FIND A CR, AND THEN START SEARCH.

; **** GETLN ***
01F8 D7 GETLN: RST 2 ;*** GETLN ***
01F8 119D43 LXI D, BUFFER ;PROMPT AND INIT.
01F8 CD9406 GL1: CALL CHKIO ;CHECK KEYBOARD
01F8 CAFE04 JZ GL1 ;NO INPUT, WAIT
01F8 FE7F CPI 7FH ;DELETE LAST CHARACTER?
01F8 C23005 JZ GL2 ;YES
01F8 D7 RST 2 ;INPUT, ECHO BACK
01F8 FE0A CPI 0FH ;IGNORE LF
01F8 B7 ORA A ;IGNORE NULL
01F8 CAFE04 JZ GL1 ;DELETE THE WHOLE LINE?
01F8 FE7D CPI 7DH ;YES
01F8 CA2005 JZ GL4 ;ELSE, SAVE INPUT
01F8 12 STAX D ;AND BUMP POINTER
01F8 13 INX D ;WAS IT CR?
01F8 FE0D CPI 0DH ;YES, END OF LINE
01F8 C8 MOV A, E ;ELSE, MORE FREE ROOM?
01F8 FE0D CPI BUFEND AND OFFH
01F8 C2F004 JNZ GL1 ;YES, GET NEXT INPUT
01F8 23 DB GL3: MOV A, E ;DELETE LAST CHARACTER
01F8 FE90 CPI BUFFER AND OFFH
01F8 CA2005 JZ GL4 ;NO, REDO WHOLE LINE
01F8 1B DCX D ;YES, BACKUP POINTER
01F8 3E5C MWI A, SCH ;AND ECHO A BACK-SLASH
01F8 D7 RST 2
01F8 C3FE04 JMP GL1 ;GO GET NEXT INPUT
01F8 C09E00 GL4: CALL CRLF ;REDO ENTIRE LINE
01F8 3E5E MWI A, 0SEH ;CR, LF AND UP-ARROW
01F8 C3FA04 JMP GETLN

; **** FNDLN ***
01F8 7C FNDLN: MOV A, H ;*** FNDLN ***
01F8 E7 ORA A ;CHECK SIGN OF HL
01F8 FA9F00 JM OHOW ;IT CANNOT BE -
01F8 111710 LXI D, TXTBNQ ;INIT. TEXT POINTER

; **** FNDLNF ***
01F8 E5 FNDLNf: FL1: PUSH H ;SAVE LINE #
01F8 2A1510 LHLD TXTNF ;CHECK IF WE PASSED END
01F8 2B DCX H
01F8 E7 RST 4 ;GET LINE # BACK
01F8 E1 POP H
01F8 D8 RC ;CNZ PASSED END
01F8 1A LEAD D ;WE DID NOT, GET BYTE 1
01F8 95 SUB L ;IS THIS THE LINE?
01F8 47 MOV B, A ;COMPARE LOW ORDER
01F8 13 INX D
01F8 1A LEAD D ;GET BYTE 2
01F8 9C SEB H ;COMPARE HIGH ORDER
01F8 00 JC FL2 ;AND, NOT THERE YET
01F8 1B DCX D ;ELSE WE EITHER FOUND
01F8 B0 ORA B ;IT, OR IT IS NOT THERE
01F8 C9 RET ;INC, Z:FOUND; NC, NZ:NOT

; **** FNDNXT ***
01F8 13 INX D ;FIND NEXT LINE
01F8 12 FL2: INX D ;JUST PASSED BYTE 1 & 2

; **** FNDSKP ***
01F8 1A LEAD D ;*** FNDSKP ***
01F8 FE0D CPI CR ;TRY TO FIND CR
01F8 00 JNZ FL2 ;KEEP LOOKING
01F8 13 INX D ;FOUND CR, SKIP OVER
01F8 C24005 JMP FL1 ;CHECK IF END OF TEXT

; **** PRTSTG *** OTSTG *** PRTNUM *** & PRTLN ***
; "PRTSTG" PRINTS A STRING POINTED BY DE. IT STOPS PRINTING
; AND RETURNS TO CALLER WHEN EITHER A CR IS PRINTED OR WHEN
; THE NEXT BYTE IS THE SAME AS WHAT WAS IN A (GIVEN BY THE
; CALLER). OLD A IS STORED IN B; OLD B IS LOST.
; "OTSTG" LOOKS FOR A BACK-ARROW, SINGLE QUOTE, OR DOUBLE
; QUOTE. IF NONE OF THESE, RETURN TO CALLER. IF BACK-ARROW,
; OUTPUT A CR WITHOUT A LF. IF SINGLE OR DOUBLE QUOTE, PRINT
; THE STRING IN THE QUOTE AND DEMANDS A MATCHING UNQUOTE.
; AFTER THE PRINTING THE NEXT 3 BYTES OF THE CALLER IS SKIPPED
; OVER (USUALLY A JUMP INSTRUCTION).
; "PRTNUM" PRINTS THE NUMBER IN HL. LEADING BLANKS ARE ADDED
; IF NEEDED TO FILL THE NUMBER OF SPACES TO THE NUMBER IN C.
; HOWEVER, IF THE NUMBER OF DIGITS IS LARGER THAN THE # IN
; C, ALL DIGITS ARE PRINTED ANYWAY. NEGATIVE SIGN IS ALSO
; PRINTED AND COUNTED IN. POSITIVE SIGN IS NOT.
; "PRTLN" PRINTS A SAVED TEXT LINE WITH LINE # AND ALL.

; PRTSTG
01F8 47 PRTSTG: RST 1 ;*** PRTSTG ***
01F8 1A LEAD D ;GET A CHARACTER
01F8 12 INX D ;BUMP POINTER
01F8 B8 CMP B ;SHMP AS OLD D?
01F8 00 R2 ;YES, RETURN
01F8 07 PS 2 ;ELSE PRINT IT
01F8 FE00 CPI CR ;WHS IT A CR?
01F8 00 JNZ PS1 ;NO, NEXT
01F8 C24105 JM PS1 ;WHS CR, RUN NEXT LINE
01F8 C9 RET ;YES, RETURN

; OTSTG
01F8 CF OTSTG: RST 1 ;*** OTSTG ***
01F8 22 DB ;_
01F8 0F DB OT2-#-1
01F8 2E22 MWI A, 2FH ;IT IS A "
01F8 C06005 OT1: CALL PRTSTG ;PRINT UNTIL ANOTHER
01F8 FE00 CPI CR ;WHS LAST ONE A CR?
01F8 E1 POP H ;RETURN ADDRESS
01F8 C4701 JZ RUNNL ;WHS CR, RUN NEXT LINE
01F8 33 OT2: INX H ;SKIP 3 BYTES ON RETURN
01F8 22 INX H
01F8 22 INX H
01F8 E9 PCLH ;RETURN
01F8 0F OT3: RST 1 ;IS IT A ?
01F8 27 DB ;_
01F8 05 DB OT4-#-1
01F8 2E27 MWI A, 27H ;YES, DO SAME
01F8 C27105 JM OT1 ;AS IN "
01F8 5F OT4: RST 1 ;IS IT BACK-ARROW?
01F8 00 DB SFH
01F8 00 DB OT5-#-1
01F8 2E8D MWI A, 0SDH ;YES, CR WITHOUT LF
01F8 D7 PRT 2 ;DO IT TWICE TO GIVE
01F8 07 PRT 2 ;TINY ENOUGH TIME
01F8 E1 POP H ;RETURN ADDRESS
01F8 C27A05 JNP OT2
01F8 C9 QTS RET ;NONE OF ABOVE

; PRTNUM
01F8 0000 MWI B, 0 ;*** PRTNUM ***
01F8 C02001 CALL CHKSGN ;CHECK SIGN
01F8 F29D05 JP PN1 ;NO SIGN
01F8 062D MWI B, "-";B-SIGN
01F8 00 DCR C ;TAKES SPACE
01F8 D5 PUSH D ;SAVE
01F8 110A00 LXI D, 0AH ;DECIMAL

```

SOFTWARE SECTION

MICROCOMPUTER DEVELOPMENT SOFTWARE

```

05A1 05 PUSH D ;SAVE AS A FLAG
05A2 00 DCR C ;C SPACES
05A2 C5 PUSH B ;SAVE SIGN & SPACE
05A4 CD6604 PN2: CALL DIVIDE ;DIVIDE HL BY 10
05A7 78 MOV A,B ;RESULT 0?
05A8 B1 ORA C
05A9 CAB405 JZ PN3 ;YES, WE GOT ALL
05AC E2 XTHL ;NO, SAVE REMAINDER
05AD 2D DCR L ;AND COUNT SPACE
05AE E5 PUSH H ;HL IS OLD BC
05AF 60 MOV H,B ;MOVE RESULT TO BC
05B0 69 MOV L,C
05B1 C0A405 JMP PN2 ;AND DIVIDE BY 10
05B4 C1 PN3: POP B ;WE GOT ALL DIGITS IN
05B5 00 PN4: DCR C ;THE STACK
05B6 79 MOV R,C ;LOOK AT SPACE COUNT
05B7 B7 ORA A
05B8 FAC105 JM FN5 ;NO LEADING BLANKS
05B8 3E20 MVI R,20H ;LEADING BLANKS
05B0 D7 RST 2
05B8 C3E505 JMP PN4 ;MORE?
05C1 78 PN5: MOV R,B ;PRINT SIGN
05C2 B7 ORA A
05C2 C11000 CNZ 10H ;LAST REMAINDER IN E
05C6 5D MOV E,L ;CHECK DIGIT IN E
05C7 7B PN6: MOV R,E ;CHECK DIGIT IN E
05C8 FEEA CPI 0FH ;10 IS FLAG FOR NO MORE
05CA D1 POP D
05CB C8 RZ ;IF SO, RETURN
05CC C630 ADI 30H ;ELSE COVERT TO ASCII
05CE D7 RST 2 ;AND PRINT THE DIGIT
05CF C3C705 JMP PN6 ;GO BACK FOR MORE

05D2 1A FRTLN: LDAX D ;*** PRTLN ***
05D2 6F MOV L,A ;LOW ORDER LINE #
05D4 12 INX D
05D5 1A LDAX D ;HIGH ORDER
05D6 67 MOV H,A
05D7 13 INX D
05D8 0E04 MVI C,4H ;PRINT 4 DIGIT LINE #
05D9 CD9205 CALL PRTNUM ;FOLLOWED BY A BLANK
05D0 3E20 MVI R,20H ;AND THEN THE TEXT
05D9 D7 RST 2
05E0 97 SUB R,C ;AND THEN THE TEXT
05E1 CD6005 CALL PRTSTG
05E4 C9 RET

; **** MVUP *** MVDOWN *** POPA *** & PUSHR ***
; MVUP MOVES A BLOCK UP FROM WHERE DE-> TO WHERE BC-> UNTIL
; DE = HL
; MVDOWN MOVES A BLOCK DOWN FROM WHERE DE-> TO WHERE HL->
; UNTIL DE = BC
; POPA RESTORES THE FOR LOOP VARIABLE SAVE AREA FROM THE
; STACK
; PUSHR STACKS THE FOR LOOP VARIABLE SAVE AREA INTO THE
; STACK

05E5 E7 MVUP: RST 4 ;*** MVUP ***
05E6 C8 RZ ;DE = HL, RETURN
05E7 1A LDAX D ;GET ONE BYTE
05E8 02 STAX B ;MOVE IT
05E9 12 INX D ;INCREASE BOTH POINTERS
05EA 02 INX B
05EB C3E505 JMP MVUP ;UNTIL DONE

MVDOWN:
05E6 78 MOV R,B ;*** MVDOWN ***
05E7 92 SUB D ;TEST IF DE = BC
05F0 C0F605 JNZ MD1 ;NO, GO MOVE
05F1 79 MOV R,C ;MAYBE, OTHER BYTE?
05F4 92 SUB E
05F5 C8 RZ ;YES, RETURN
05F6 1B MD1: DCX D ;ELSE MOVE A BYTE
05F7 3B DCX H ;BUT FIRST DECREASE
05F8 1A LDAX D ;BOTH POINTERS AND
05F9 77 MOV M,A ;THEN DO IT
05FA C3E605 JMP MVDOWN ;LOOP BACK

05F0 C1 POPA: POP B ;BC = RETURN ADDR
05F1 E1 POP H ;RESTORE LOPVAR, BUT
05F2 20H SHLD LOPVAR ;#0 MEANS NO MORE
05F3 7C MOV R,H
05F4 1A JZ PP1 ;YEP, GO RETURN
05F5 E1 POP H ;NOPE, RESTORE OTHERS
05F6 08H SHLD LOPINC
05F7 E1 POP H
05F8 20H SHLD LOPLMT
05F9 E1 POP H
0600 20H SHLD LOPLN
0601 20H SHLD LOPPT
0602 E1 POP H
0603 20H SHLD LOPR
0604 20H RET

0619 C10E12 PUSHA: LXI H,STKLMT ;*** PUSHA ***
0610 CD8601 CALL CHGSSGN

061F C1 POP B ;BC=RETURN ADDRESS
0620 39 DAD SP ;IS STACK NEAR THE TOP?
0621 02F204 JNC OSORRY ;YES, SORRY FOR THAT.
0624 JA0910 LHLD LOPVRR ;ELSE SAVE LOOP VRR'S
0627 7C MOV A,H ;BUT IF LOPVRR IS 0
0628 B5 ORA L ;THAT WILL BE ALL
0629 C4CF05 JZ PU1
0630 2A1110 LHLD LOPPT ;ELSE, MORE TO SAVE
0631 E5 PUSH H
0632 2A0F10 LHLD LOPLN
0633 E5 PUSH H
0634 2A0D10 LHLD LOPLMT
0635 E5 PUSH H
0636 2A0B10 LHLD LOPINC
0637 E5 PUSH H
0638 2A0910 LHLD LOPR
0639 E5 PU1: PUSH H
0640 C5 PUSH B ;BC = RETURN ADDR
0641 C9 RET

; THESE ARE THE ONLY I/O ROUTINES IN TBI.
; OUTC IS CONTROLLED BY A SOFTWARE SWITCH 'OCSW'. IF OCSW=0
; OUTC WILL JUST RETURN TO THE CALLER. IF OCSW IS NOT 0,
; IT WILL OUTPUT THE BYTE IN A. IF THAT IS A CR, A LF IS ALSO
; SEND OUT. ONLY THE FLAGS MAY BE CHANGED AT RETURN. ALL REGS
; ARE RESTORED.

; 'CHKIO' CHECKS THE INPUT. IF NO INPUT, IT WILL RETURN TO
; THE CALLER WITH THE Z FLAG SET. IF THERE IS INPUT, Z FLAG
; IS CLEARED AND THE INPUT BYTE IS IN A. HOWEVER, IF THE
; INPUT IS A CONTROL-O, THE 'OCSW' SWITCH IS COMPLEMENTED, AND
; Z FLAG IS RETURNED. IF A CONTROL-C IS READ, 'CHKIO' WILL
; RESTART TBI AND DO NOT RETURN TO THE CALLER.

; OUTC PUSH AF THIS IS AT LOC. 10
; LD A,OCSW CHECK SOFTWARE SWITCH
; IOR A

0642 320010 INIT: STA OCSW
0643 3ECF MVI R,0CFH
0644 03FB OUT 0FBH
0645 3E27 MVI R,27H
0646 03FB OUT 0FBH
0647 1619 MVI D,19H
PATL0P:
064F C0DE00 CALL CRLF
0652 15 DCR D
0653 C2F005 JNZ PATL0P
0655 97 SUB R
0657 11R205 LXI D,MSG1
0658 C06005 CALL PRTSTG
0659 210000 LXI H,START
0660 221210 SHLD RANPNT
0662 211710 LXI H,TXTBGN
0666 221510 SHLD TXTUNF
0669 C2B400 JMP RSTART
0670 C27106 OC2: JNZ OC3 ;IT IS ON
0671 DFBF POP PSW ;IT IS OFF
0672 C9 RET ;RESTORE AF AND RETURN
0673 6E01 OC3: IN 0FBH ;COME HERE TO DO OUTPUT
0675 C47106 HNI 1H ;STATUS BIT
0678 F1 JZ OC3 ;NOT READY, WAIT
0679 D2FA POP PSW ;READY, GET OLD A BACK
0680 FE0D OUT 0FAH ;AND SEND IT OUT
0681 C1 CR CPI CR ;WAS IT CR?
0682 C0 RNZ ;NO, FINISHED
0683 2E0A MVI R,LF ;SEND LF
0684 D7 RST 2 ;THIS IS RECURSIVE
0685 2E0D MVI R,CR ;GET CR BACK IN R
0686 C9 RET
0687 E002 CHKIO: IN 0FBH ;*** CHKIO ***
0688 00 NOP ;STATUS BIT FLIPPED?
0689 00 ANI 2H ;MASK STATUS BIT
0690 00 R2 ;NOT READY, RETURN "Z"
0691 DFBF IN 0FAH ;READY, READ DATA
0692 E67F ANI 7FH ;MASK BIT 7 OFF
0693 FE0F CPI 0FH ;IS IT CONTROL-O?
0694 C29D05 JNZ C11 ;NO, MORE CHECKING
0695 3A0010 LDA OCSW ;CONTROL-O FLIPS OCSW
0696 2E CMA ;ON TO OFF, OFF TO ON
0697 320010 STA OCSW
0698 C3B406 JMP CHKIO ;GET ANOTHER INPUT
0699 FE03 CII: CPI 3H ;IS IT CONTROL-C?
06A0 C2B400 RNZ ;NO, RETURN "Nz"
06A2 54494E59 MSG1: DB 'TINY'
06A7 20 JMP 4215249 DB 'BASIC'
06A0 42 DB CR

; **** TABLES *** DIRECT *** & EXEC ***
; THIS SECTION OF THE CODE TESTS A STRING AGAINST A TABLE.
; WHEN A MATCH IS FOUND, CONTROL IS TRANSFERRED TO THE SECTION
; OF CODE ACCORDING TO THE TABLE.
; AT 'EXEC', DE SHOULD POINT TO THE STRING AND HL SHOULD POINT
; TO THE TABLE-1. AT 'DIRECT', DE SHOULD POINT TO THE STRINGS.
; HL WILL BE SET UP TO POINT TO TAB1-1, WHICH IS THE TABLE (CII)
; FOR ALL DIRECT AND STATEMENT COMMANDS.
; R1 WILL IN THE STRING WILL TERMINATE THE TEST AND THE PARTIAL
; MATCH WILL BE CONSIDERED AS A MATCH. E.G., 'PR%', 'PR1%', 'PR11%' WILL ALL MATCH 'PRINT'.
; THE TABLE CONSISTS OF ANY NUMBER OF ITEMS. EACH ITEM
; IS A STRING OF CHARACTERS WITH BIT 7 SET TO 0 AND
; A JUMP ADDRESS STORED HI-LOW WITH BIT 7 OF THE HIGH
; BYTE SET TO 1.
; END OF TABLE IS AN ITEM WITH A JUMP ADDRESS ONLY. IF THE
; STRING DOES NOT MATCH ANY OF THE OTHER ITEMS, IT WILL
; MATCH THIS NULL ITEM AS DEFAULT.

TAB1: .DIRECT COMMANDS
06A6 4C195254 DB 'LIST'
+ DWA LIST
06B2 81 + DB (0016FH SHR 8) +128
06B3 6F + DB 0016FH AND 0FFH
06B4 5255ME DB 'RUN'
06B7 81 + DB (00141H SHR 8) +128
06B8 41 + DB 00141H AND 0FFH
06B9 4E4557 DB 'NEW'
+ DWA NEW
06B0 81 + DB (00132H SHR 8) +128
06B2 32 + DB 00132H AND 0FFH
TAB2: .DIRECT/STATEMENT
06B6 4E455854 DB 'NEXT'
+ DWA NEXT
06C2 82 + DB (00257H SHR 8) +128
06C3 57 + DB 00257H AND 0FFH
06C4 4C4554 DB 'LET'
+ DWA LET
06C7 82 + DB (00323H SHR 8) +128
06C8 22 + DB 00323H AND 0FFH
06C9 4946 DB 'IF'
+ DWA IFF
06CB 82 + DB (002B4H SHR 8) +128
06CC B4 + DB 002B4H AND 0FFH
06CD 474F544F DB 'GOTO'
+ DWA GOTO
06D1 91 + DB (00160H SHR 8) +128
06D2 60 + DB 00160H AND 0FFH
06D8 81 + DB (001BFH SHR 8) +128
06D9 BF + DB 001BFH AND 0FFH

```

SOFTWARE SECTION

MICROCOMPUTER DEVELOPMENT SOFTWARE

Merry Christmas
Happy New Year

```

060A 52455455 DB 'RETURN'
060E 524E           DB RETURN
060F 81           + DB (001DFH SHR 8) +128
060E1 DF           DB 001DFH AND 0FFH

060E2 52454D0      DB 'REM'
060E3 82           + DB (002B0H SHR 8) +128
060E6 B0           + DB 002B0H AND 0FFH

060E7 464F52       DB 'FOR'
060E8 81           + DB (001F8H SHR 8) +128
060E9 F8           + DB 001F8H AND 0FFH

060E C 494E5055    DB 'INPUT'
060F 54           DB INPUT

060F1 82           + DB (002CDH SHR 8) +128
060F2 CD           DB 002CDH AND 0FFH

060F3 5052494E    DB 'PRINT'
060F7 54           DB PRINT

060F8 81           + DB (00187H SHR 8) +128
060F9 87           + DB 00187H AND 0FFH

060FA 52544F50    DB 'STOP'
060F9 81           + DB (0012BH SHR 8) +128
060FF 3B           + DB 0012BH AND 0FFH

060F0 83           + DB DEFLT
060F1 1D           + DB (0031DH SHR 8) +128
060F2 25           + DB 0031DH AND 0FFH

TAB4:          ; FUNCTIONS
060F2 524E44       DB 'RND'
060F3 84           + DB RND
060F6 25           + DB (00425H SHR 8) +128
060F7 25           + DB 00425H AND 0FFH

060F7 414253       DB 'ABS'
060F8 84           + DB ABS
060F9 50           + DB (00450H SHR 8) +128
060F0 50           + DB 00450H AND 0FFH

060F0 524595A5      DB 'SIZE'
060F1 84           + DB SIZE
060F2 59           + DB (00459H SHR 8) +128
060F3 59           + DB 00459H AND 0FFH

060F4 84           + DB XP40
060F5 0B           + DB (0040BH SHR 8) +128
060F6 0B           + DB 0040BH AND 0FFH

TAB5:          ; "TO" IN "FOR"
060F6 544F           DB 'TO'
060F7 84           + DB FR1
060F8 82           + DB (00208H SHR 8) +128
060F9 05           + DB 00208H AND 0FFH

060F9 0NWHAT
060F10 84          + DB (004C6H SHR 8) +128
060F11 C6           + DB 004C6H AND 0FFH

TAB6:          ; "STEP" IN "FOR"
060F11 52544550      DB 'STEP'
060F12 84           + DB FR2
060F13 82           + DB (00212H SHR 8) +128
060F14 12           + DB 00212H AND 0FFH

060F14 FR3
060F15 82           + DB (00216H SHR 8) +128
060F16 16           + DB 00216H AND 0FFH

TAB8:          ; RELATION OPERATORS
060F16 3E3D           DB '>='
060F17 84           + DB XP11
060F18 82           + DB (00333H SHR 8) +128
060F19 33           + DB 00333H AND 0FFH

060F20 23           + DB '#'
060F21 84           + DB XP12
060F22 82           + DB (00339H SHR 8) +128
060F23 39           + DB 00339H AND 0FFH

060F24 3E           + DB '>'
060F25 83           + DB XP13
060F26 82           + DB (0033FH SHR 8) +128
060F27 2F           + DB 0033FH AND 0FFH

060F28 3D           + DB '='
060F29 83           + DB XP15
060F30 82           + DB (0034EH SHR 8) +128
060F31 4E           + DB 0034EH AND 0FFH

060F32 303D           DB '<='
060F33 83           + DB XP14
060F34 82           + DB (00246H SHR 8) +128
060F35 46           + DB 00246H AND 0FFH

060F36 20            DB '<'
060F37 82           + DB XP16
060F38 82           + DB (00254H SHR 8) +128
060F39 54           + DB 00254H AND 0FFH

060F40 82           + DB XP17
060F41 82           + DB (0025AH SHR 8) +128
060F42 5A           + DB 0025AH AND 0FFH

;
; DIRECT:
060F43 21AD06       LXI H,TAB1-1 ;*** DIRECT ***
EXEC:          ;*** EXEC ***
EX0:   RST 5           ; IGNORE LEADING BLANKS
EX0:   PUSH D           ; SAVE POINTER
EX0:   T0D 1A           ; IF FOUND ' ' IN STRING
EX1:   LDIX D           ; BEFORE ANY MISMATCH
EX1:   INC D             ; WE DECLARE A MATCH
EX1:   CPI 2EH           ; IF MATCH, TEST NEXT
EX1:   JZ EX3
EX1:   INC H             ; HL->TABLE
EX1:   CMP M             ; IF MATCH, TEST NEXT
EX1:   JZ EX1
EX1:   INC H             ; INC NO. FIND JUMP ADDR.
EX1:   JNC EX2

060F43 39 3E7F       MVI A,07FH ; ELSE, SEE IF BIT 7
060F44 31 3E7F       DCX D           ; OF TABLE IS SET, WHICH
060F45 BE             CMP M             ; IS THE JUMP ADDR. (<H>)
060F46 DA6107         JC EX5           ; <1> YES, MATCHED
060F47 20             INX H             ; INC NO. FIND JUMP ADDR.
060F48 35 3E7F       JNC EX2           ; NO, GO TO IT

060F49 23             INX H             ; BUMP TO NEXT TAB. ITEM
060F50 54             POP D             ; RESTORE STRING POINTER
060F51 00             JMP EX0           ; TEST AGAINST NEXT ITEM

060F52 54 3E7F       EMDI H,07FH ; PARTIAL MATCH, FIND
060F53 32             EMDS H,07FH ; JUMP ADDR. WHICH IS
060F54 6E             CMP M             ; FLAGGED BY BIT 7
060F55 D05C07         INC EX4           ; LOAD HL WITH THE JUMP
060F56 7E             MOV R,A           ; ADDRESS FROM THE TABLE
060F57 23             INC H             ; LOAD HL WITH THE JUMP
060F58 5E             MOV L,M           ; ADDRESS FROM THE TABLE
060F59 E67F           ANI 7FH           ; MASK OFF BIT 7
060F60 57             MOV H,R           ; LOAD HL WITH THE JUMP
060F61 F1             POP PSW           ; CLEAN UP THE GARBAGE
060F62 E9             PCHL             ; AND WE GO ON IT

LSTROM:        ; LISTING OUTPUT
060F63 0000           ORG 1000H ; HERE ON MUST BE RAM
060F64 0001           DS 1             ; SWITCH FOR OUTPUT
060F65 1001           DS 2             ; POINTS TO CURRENT LINE
060F66 1002           STKDS 2           ; SAVES SP IN 'GOSUB'
060F67 1005           VRHNT 2           ; TEMP STORAGE
060F68 1007           STKINP 2           ; SAVES SP IN 'INPUT'
060F69 1009           LOPVAR 2           ; FOR LOOP SAVE AREA
060F70 100E           LOPINC 2           ; INCREMENT
060F71 100D           LOPLMT 2           ; LIMIT
060F72 100F           LOPLN 2            ; LINE NUMBER
060F73 1011           LOPPT 2            ; TEXT POINTER
060F74 1013           RPNPNT 2           ; RANDOM NUMBER POINTER
060F75 1015           TXTJNF 2           ; -UNFILLED TEXT AREA
060F76 1017           TXTBGN 2           ; TEXT SAVE AREA BEGINS
060F77 1026           ORG 1366H ; TEXT SAVE AREA ENDS
060F78 1026           TXTEND 2           ; VARIABLE @()
060F79 1026           VRREGN 2           ; INPUT BUFFER
060F80 1026           BUFFER 2           ; BUFFER ENDS
060F81 1026           BUFFEND 2          ; TOP LIMIT FOR STACK
060F82 1026           STKLMT 2           ; STACK STARTS HERE
060F83 1026           STACK: DS 0
060F84 0000           CR EOU 00H
060F85 0000           LF EOU 00H
060F86 0000           END

060F87 0450           ABS 00A0           ASORR 04F4           RWHRT 04C7
060F88 13D0           BUFEN 139D          CHSGG 0486           CKHIO 0684
060F89 0483           C11 069D          CK1 049E           CKHLD 0498
060F90 000D           CRLF 000E           CURRN 1001           DEFLT 031D
060F91 0738           DIVID 0466          DV1 0471           DV2 0473
060F92 0F78           ENDCH 04C2           ERROR 04CA           EX0 0738
060F93 073D           EX2 0756          EX3 0759           EX4 075C
060F94 0761           EXEC 0738          EXPR1 032D          EXPKR2 0371
060F95 0345           EXPR3 0405          FI1 048R           F12 04C1
060F96 0482           FIN 0482           FL1 0540           FL2 0555
060F97 0540           FNOLP 0540          FNDSK 0556          FOR 01F8
060F98 0554           FNNDN 0554          FR8 0252           GETLN 04FA
060F99 021C           FR5 0231           FR2 0212           FR3 0216
060F9A 04FE           GL1 0523           GL2 0523           GL4 0530
060F9B 0160           GOTO 0160          H0W 0046           IFF 0284
060F9C 0000           INPER 02C2          INPUT 02CD          IP1 02CD
060F9D 02E8           IP2 02B8          IP3 031C          LET 0323
060F9E 000A           LF 000A           LIST 016F          LOPIN 100B
060F9F 100F           LOPLN 1011          LOPPT 1011          LOPVA 1009
060F9G 0769           LSTRO 0769          LT1 0220           MD1 056F
060F9H 055E           MVDOH 055E          NEW 0132           NEXT 0257
060F9I 025E           NX0 0298          NX1 0298           NX2 02AC
060F9J 025E           NX3 0288          NX5 0288           NX6 0276
060F9K 1000           OC5 0008          OK 0008           PARN 041R
060F9L 0599           PN1 0599           PN2 0584           PN3 0584
060F9M 0501           PNS 0501           PNE 05C7           POPR 05FD
060F9N 0198           PR0 0198           PR1 01A2           PR2 0192
060F9O 0182           PR6 0182           PR8 01B6           PRINT 0187
060F9P 0592           PRTNU 0592          PRTST 0560           PS1 0561
060F9Q 0619           PUSH 0619          PUSHP 0619          PSY 0585
060F9R 057H           OT2 057H           OT3 057E           QT4 0586
060F9S 056C           OTSTG 056C          QWHT 04C6           RR1 0448
060F9T 0268           REM 0268           RETUR 01D6          RND 0425
060F9U 0141           RUN 0141           RUNX 0147           RUNSM 0157
060F9V 0495           SETVR 0495          S12 0455           S0RY 0084
060F9W 0000           S11 0000           S12 0000           SSA 0028
060F9X 0000           S12 0000           S13 0006           S14 0198
060F9Y 1400           STCKL 1400          START 0600          STKQG 1002
060F9Z 067E           STOP 067E          SUBD 067C           SV4 0480
060F9A 067E           THB1 067E          THB2 067E          TRB4 0714
060F9B 071A           THB5 071A          THB8 0672          TRB5 0673
060F9C 067E           THM1 067E          THM2 0672          TRC2 0714
060F9D 067C           TM 067C           TM 067C           TMC 0673
060F9E 0677           TXTEN 1366          TXTNU 0677          TXTBG 1817
060F9F 1366           UHAT 009E          XPI1 0333           XPI2 0239
060F9G 009E           XPI2 0333          XPI3 0345           XPI4 0254
060F9H 025C           XPI3 0345          XPI4 0254           XPI5 0257
060F9I 025C           XPI4 0254          XPI5 0254           XPI6 0259
060F9J 025C           XPI5 0254          XPI6 0254           XPI7 0257
060F9K 025C           XPI6 0254          XPI7 0254           XPI8 0259
060F9L 025C           XPI7 0254          XPI8 0254           XPI9 0259
060F9M 025C           XPI8 0254          XPI9 0254           XPI10 0259
060F9N 025C           XPI9 0254          XPI10 0254          XPI11 0259
060F9O 025C           XPI10 0254          XPI11 0254          XPI12 0259
060F9P 025C           XPI11 0254          XPI12 0254           XPI13 0259
060F9Q 025C           XPI12 0254          XPI13 0254           XPI14 0259
060F9R 025C           XPI13 0254          XPI14 0254           XPI15 0259
060F9S 025C           XPI14 0254          XPI15 0254           XPI16 0259
060F9T 025C           XPI15 0254          XPI16 0254           XPI17 0259
060F9U 025C           XPI16 0254          XPI17 0254           XPI18 0259
060F9V 025C           XPI17 0254          XPI18 0254           XPI19 0259
060F9W 025C           XPI18 0254          XPI19 0254           XPI20 0259
060F9X 025C           XPI19 0254          XPI20 0254           XPI21 0259
060F9Y 025C           XPI20 0254          XPI21 0254           XPI22 0259
060F9Z 025C           XPI21 0254          XPI22 0254           XPI23 0259
060F9A 025C           XPI22 0254          XPI23 0254           XPI24 0259
060F9B 025C           XPI23 0254          XPI24 0254           XPI25 0259
060F9C 025C           XPI24 0254          XPI25 0254           XPI26 0259
060F9D 025C           XPI25 0254          XPI26 0254           XPI27 0259
060F9E 025C           XPI26 0254          XPI27 0254           XPI28 0259
060F9F 025C           XPI27 0254          XPI28 0254           XPI29 0259
060F9G 025C           XPI28 0254          XPI29 0254           XPI30 0259
060F9H 025C           XPI29 0254          XPI30 0254           XPI31 0259
060F9I 025C           XPI30 0254          XPI31 0254           XPI32 0259
060F9J 025C           XPI31 0254          XPI32 0254           XPI33 0259
060F9K 025C           XPI32 0254          XPI33 0254           XPI34 0259
060F9L 025C           XPI33 0254          XPI34 0254           XPI35 0259
060F9M 025C           XPI34 0254          XPI35 0254           XPI36 0259
060F9N 025C           XPI35 0254          XPI36 0254           XPI37 0259
060F9O 025C           XPI36 0254          XPI37 0254           XPI38 0259
060F9P 025C           XPI37 0254          XPI38 0254           XPI39 0259
060F9Q 025C           XPI38 0254          XPI39 0254           XPI40 0259
060F9R 025C           XPI39 0254          XPI40 0254           XPI41 0259
060F9S 025C           XPI40 0254          XPI41 0254           XPI42 0259
060F9T 025C           XPI41 0254          XPI42 0254           XPI43 0259
060F9U 025C           XPI42 0254          XPI43 0254           XPI44 0259
060F9V 025C           XPI43 0254          XPI44 0254           XPI45 0259
060F9W 025C           XPI44 0254          XPI45 0254           XPI46 0259
060F9X 025C           XPI45 0254          XPI46 0254           XPI47 0259
060F9Y 025C           XPI46 0254          XPI47 0254           XPI48 0259
060F9Z 025C           XPI47 0254          XPI48 0254           XPI49 0259
060F9A 025C           XPI48 0254          XPI49 0254           XPI50 0259
060F9B 025C           XPI49 0254          XPI50 0254           XPI51 0259
060F9C 025C           XPI50 0254          XPI51 0254           XPI52 0259
060F9D 025C           XPI51 0254          XPI52 0254           XPI53 0259
060F9E 025C           XPI52 0254          XPI53 0254           XPI54 0259
060F9F 025C           XPI53 0254          XPI54 0254           XPI55 0259
060F9G 025C           XPI54 0254          XPI55 0254           XPI56 0259
060F9H 025C           XPI55 0254          XPI56 0254           XPI57 0259
060F9I 025C           XPI56 0254          XPI57 0254           XPI58 0259
060F9J 025C           XPI57 0254          XPI58 0254           XPI59 0259
060F9K 025C           XPI58 0254          XPI59 0254           XPI60 0259
060F9L 025C           XPI59 0254          XPI60 0254           XPI61 0259
060F9M 025C           XPI60 0254          XPI61 0254           XPI62 0259
060F9N 025C           XPI61 0254          XPI62 0254           XPI63 0259
060F9O 025C           XPI62 0254          XPI63 0254           XPI64 0259
060F9P 025C           XPI63 0254          XPI64 0254           XPI65 0259
060F9Q 025C           XPI64 0254          XPI65 0254           XPI66 0259
060F9R 025C           XPI65 0254          XPI66 0254           XPI67 0259
060F9S 025C           XPI66 0254          XPI67 0254           XPI68 0259
060F9T 025C           XPI67 0254          XPI68 0254           XPI69 0259
060F9U 025C           XPI68 0254          XPI69 0254           XPI70 0259
060F9V 025C           XPI69 0254          XPI70 0254           XPI71 0259
060F9W 025C           XPI70 0254          XPI71 0254           XPI72 0259
060F9X 025C           XPI71 0254          XPI72 0254           XPI73 0259
060F9Y 025C           XPI72 0254          XPI73 0254           XPI74 0259
060F9Z 025C           XPI73 0254          XPI74 0254           XPI75 0259
060F9A 025C           XPI74 0254          XPI75 0254           XPI76 0259
060F9B 025C           XPI75 0254          XPI76 0254           XPI77 0259
060F9C 025C           XPI76 0254          XPI77 0254           XPI78 0259
060F9D 025C           XPI77 0254          XPI78 0254           XPI79 0259
060F9E 025C           XPI78 0254          XPI79 0254           XPI80 0259
060F9F 025C           XPI79 0254          XPI80 0254           XPI81 0259
060F9G 025C           XPI80 0254          XPI81 0254           XPI82 0259
060F9H 025C           XPI81 0254          XPI82 0254           XPI83 0259
060F9I 025C           XPI82 0254          XPI83 0254           XPI84 0259
060F9J 025C           XPI83 0254          XPI84 0254           XPI85 0259
060F9K 025C           XPI84 0254          XPI85 0254           XPI86 0259
060F9L 025C           XPI85 0254          XPI86 0254           XPI87 0259
060F9M 025C           XPI86 0254          XPI87 0254           XPI88 0259
060F9N 025C           XPI87 0254          XPI88 0254           XPI89 0259
060F9O 025C           XPI88 0254          XPI89 0254           XPI90 0259
060F9P 025C           XPI89 0254          XPI90 0254           XPI91 0259
060F9Q 025C           XPI90 0254          XPI91 0254           XPI92 0259
060F9R 025C           XPI91 0254          XPI92 0254           XPI93 0259
060F9S 025C           XPI92 0254          XPI93 0254           XPI94 0259
060F9T 025C           XPI93 0254          XPI94 0254           XPI95 0259
060F9U 025C           XPI94 0254          XPI95 0254           XPI96 0259
060F9V 025C           XPI95 0254          XPI96 0254           XPI97 0259
060F9W 025C           XPI96 0254          XPI97 0254           XPI98 0259
060F9X 025C           XPI97 0254          XPI98 0254           XPI99 0259
060F9Y 025C           XPI98 0254          XPI99 0254           XPI100 0259
060F9Z 025C           XPI99 0254          XPI100 0254          XPI101 0259
060F9A 025C           XPI100 0254         XPI101 0254          XPI102 0259
060F9B 025C           XPI101 0254         XPI102 0254          XPI103 0259
060F9C 025C           XPI102 0254         XPI103 0254          XPI104 0259
060F9D 025C           XPI103 0254         XPI104 0254          XPI105 0259
060F9E 025C           XPI104 0254         XPI105 0254          XPI106 0259
060F9F 025C           XPI105 0254         XPI106 0254          XPI107 0259
060F9G 025C           XPI106 0254         XPI107 0254          XPI108 0259
060F9H 025C           XPI107 0254         XPI108 0254          XPI109 0259
060F9I 025C           XPI108 0254         XPI109 0254          XPI110 0259
060F9J 025C           XPI109 0254         XPI110 0254          XPI111 0259
060F9K 025C           XPI110 0254         XPI111 0254          XPI112 0259
060F9L 025C           XPI111 0254         XPI112 0254          XPI113 0259
060F9M 025C           XPI112 0254         XPI113 0254          XPI114 0259
060F9N 025C           XPI113 0254         XPI114 0254          XPI115 0259
060F9O 025C           XPI114 0254         XPI115 0254          XPI116 0259
060F9P 025C           XPI115 0254         XPI116 0254          XPI117 0259
060F9Q 025C           XPI116 0254         XPI117 0254          XPI118 0259
060F9R 025C           XPI117 0254         XPI118 0254          XPI119 0259
060F9S 025C           XPI118 0254         XPI119 0254          XPI120 0259
060F9T 025C           XPI119 0254         XPI120 0254          XPI121 0259
060F9U 025C           XPI120 0254         XPI121 0254          XPI122 0259
060F9V 025C           XPI121 0254         XPI122 0254          XPI123 0259
060F9W 025C           XPI122 0254         XPI123 0254          XPI124 0259
060F9X 025C           XPI123 0254         XPI124 0254          XPI125 0259
060F9Y 025C           XPI124 0254         XPI125 0254          XPI126 0259
060F9Z 025C           XPI125 0254         XPI126 0254          XPI127 0259
060F9A 025C           XPI126 0254         XPI127 0254          XPI128 0259
060F9B 025C           XPI127 0254         XPI128 0254          XPI129 0259
060F9C 025C           XPI128 0254         XPI129 0254          XPI130 0259
060F9D 025C           XPI129 0254         XPI130 0254          XPI131 0259
060F9E 025C           XPI130 0254         XPI131 0254          XPI132 0259
060F9F 025C           XPI131 0254         XPI132 0254          XPI133 0259
060F9G 025C           XPI132 0254         XPI133 0254          XPI134 0259
060F9H 025C           XPI133 0254         XPI134 0254          XPI135 0259
060F9I 025C           XPI134 0254         XPI135 0254          XPI136 0259
060F9J 025C           XPI135 0254         XPI136 0254          XPI137 0259
060F9K 025C           XPI136 0254         XPI137 0254          XPI138 0259
060F9L 025C           XPI137 0254         XPI138 0254          XPI139 0259
060F9M 025C           XPI138 0254         XPI139 0254          XPI140 0259
060F9N 025C           XPI139 0254         XPI140 0254          XPI141 0259
060F9O 025C           XPI140 0254         XPI141 0254          XPI142 0259
060F9P 025C           XPI141 0254         XPI142 0254          XPI143 0259
060F9Q 025C           XPI142 0254         XPI143 0254          XPI144 0259
060F9R 025C           XPI143 0254         XPI144 0254          XPI145 0259
060F9S 025C           XPI144 0254         XPI145 0254          XPI146 0259
060F9T 025C           XPI145 0254         XPI146 0254          XPI147 0259
060F9U 025C           XPI146 0254         XPI147 0254          XPI148 0259
060F9V 025C           XPI147 0254         XPI148 0254          XPI149 0259
060F9W 025C           XPI148 0254         XPI149 0254          XPI150 0259
060F9X 025C           XPI149 0254         XPI150 0254          XPI151 0259
060F9Y 025C           XPI150 0254         XPI151 0254          XPI152 0259
060F9Z 025C           XPI151 0254         XPI152 0254          XPI153 0259
060F9A 025C           XPI152 0254         XPI153 0254          XPI154 0259
060F9B 025C           XPI153 0254         XPI154 0254          XPI155 0259
060F9C 025C           XPI154 0254         XPI155 0254          XPI156 0259
060F9D 025C           XPI155 0254         XPI156 0254          XPI157 0259
060F9E 025C           XPI156 0254         XPI157 0254          XPI158 0259
060F9F 025C           XPI157 0254         XPI158 0254          XPI159 0259
060F9G 025C           XPI158 0254         XPI159 0254          XPI160 0259
060F9H 025C           XPI159 0254         XPI160 0254          XPI161 0259
060F9I 025C           XPI160 0254         XPI161 0254          XPI162 0259
060F9J 025C           XPI161 0254         XPI162 0254          XPI163 0259
060F9K 025C           XPI162 0254         XPI163 0254          XPI164 0259
060F9L 025C           XPI163 0254         XPI164 0254          XPI165 0259
060F9M 025C           XPI164 0254         XPI165 0254          XPI166 0259
060F9N 025C           XPI165 0254         XPI166 0254          XPI167 0259
060F9O 025C           XPI166 0254         XPI167 0254          XPI168 0259
060F9P 025C           XPI167 0254         XPI168 0254          XPI169 0259
060F9Q 025C           XPI168 0254         XPI169 0254          XPI170 0259
060F9R 025C           XPI169 0254         XPI170 0254          XPI171 0259
060F9S 025C           XPI170 0254         XPI171 0254          XPI172 0259
060F9T 025C           XPI171 0254         XPI172 0254          XPI173 0259
060F9U 025C           XPI172 0254         XPI173 0254          XPI174 0259
060F9V 025C           XPI173 0254         XPI174 0254          XPI175 0259
060F9W 025C           XPI174 0254         XPI175 0254          XPI176 0259
060F9X 025C           XPI175 0254         XPI176 0254          XPI177 0259
060F9Y 025C           XPI176 0254         XPI177 0254          XPI178 0259
060F9Z 025C           XPI177 0254         XPI178 0254          XPI179 0259
060F9A 025C           XPI178 0254         XPI179 0254          XPI180 0259
060F9B 025C           XPI179 0254         XPI180 0254          XPI181 0259
060F9C 025C           XPI180 0254         XPI181 0254          X
```

Merry Christmas

Happy New Year