DR. WANG'S PALO ALTO TINY BASIC

By Roger Rauskolb

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 OOOOH 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 @(1). 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, 2/3=0.

* multiply.

− subtract.

+ add.

> compare if greater than.

< compare if less than.

= compare if equal to. Note than to certain versions of Basic "LET A=B=O" means "set both A and B to O". To this version of Tiny Basic, it means "set A to the result of comparing B with O".

compare if not equal to.

>= compare if greater than or equal to.

<= 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

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 ',

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

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-0, then turn on the paper tape reader. When the paper tape is finished, turn it off and type a Control-0 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-0, then turn on the paper tape reader. When the paper tape is finished, turn it off and type a Control-0 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
                ;                     VERSION 2.0
                ;                  BY LI-CHEN WANG
                ;               MODIFIED AND TRANSLATED
                ;                 TO INTEL MNEMONICS
                ;                  BY ROGER RAUSKOLB
                ;                   10 OCTOBER, 1976
                ;                     @COPYLEFT
                ;                 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 LLHH.  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.
                ;
                DWA     MACRO   WHERE
                        DB      (WHERE SHR 8) +128
                        DB      WHERE AND 0FFH
                        ENDM
0000                    ORG     0H
                START:
0000 310014             LXI     SP,STACK ;*** COLD START ***
0003 3EFF               MVI     A,0FFH
0005 C34206             JMP     INIT
                ;
0008 E3         RST1:   XTHL            ;*** TSTC OR RST 1 ***
0009 EF                 RST     5       ;IGNORE BLANKS AND
000A BE                 CMP     M       ;TEST CHARACTER
000B C36800             JMP     TC1     ;REST OF THIS IS AT TC1
                ;
000E 3E0D       CRLF:   MVI     A,CR    ;*** CRLF ***
                ;
0010 F5         RST2:   PUSH    PSW     ;*** OUTC OR RST 2 ***
0011 3A0010             LDA     OCSW    ;PRINT CHARACTER ONLY
0014 B7                 ORA     A       ;IF OCSW SWITCH IS ON
0015 C36C06             JMP     OC2     ;REST OF THIS IS AT OC2
                ;
0018 CD7103     RST3:   CALL    EXPR2   ;*** EXPR OR RST 3 ***
001B E5                 PUSH    H       ;EVALUATE AN EXPRESION
001C C32D03             JMP     EXPR1   ;REST OF IT IS AT EXPR1
                ;
001F 57                 DB      'W'
                ;
0020 7C         RST4:   MOV     A,H     ;*** COMP OR RST 4 ***
0021 BA                 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      'AN'
                ;
0028 1A         SS1:    LDAX    D       ;*** IGNBLK/RST 5 ***
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         RST6:   POP     PSW     ;*** FINISH/RST 6 ***
0031 CDB304             CALL    FIN     ;CHECK END OF COMMAND
0034 C3C604             JMP     QWHAT   ;PRINT "WHAT?" IF WRONG
0037 47                 DB      'G'
                ;
0038 EF         RST7:   RST     5       ;*** TSTV OR RST 7 ***
0039 D640               SUI     40H     ;TEST VARIABLES
003B D8                 RC              ;C:NOT A VARIABLE
003C C25800             JNZ     TV1     ;NOT "@" ARRAY
003F 13                 INX     D       ;IT IS THE "@" ARRAY
0040 CD1A04             CALL    PARN    ;@ SHOULD BE FOLLOWED
0043 29                 DAD     H       ;BY (EXPR) AS ITS INDEX
0044 DA9F00             JC      QHOW    ;IS INDEX TOO BIG?
0047 D5                 PUSH    D       ;WILL IT OVERWRITE
0048 EB                 XCHG            ;TEXT?
0049 CD5904             CALL    SIZE    ;FIND SIZE OF FREE
004C E7                 RST     4       ;AND CHECK THAT
004D DAF404             JC      ASORRY  ;IF SO, SAY "SORRY"
0050 216613             LXI     H,VARBGN ;IF NOT, GET ADDRESS
0053 CD7C04             CALL    SUBDE   ;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
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,0
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 0600               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 13         TC2:    INX     D       ;IF =, SKIP THOSE BYTES
0074 23                 INX     H       ;AND CONTINUE
0075 E3                 XTHL
0076 C9                 RET
                ;
                TSTNUM:
0077 210000             LXI     H,0     ;*** TSTNUM ***
007A 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                 RC              ;B AND HL
007F FE3A               CPI     3AH     ;IF NUMBERS, CONVERT
0081 D0                 RNC             ;TO BINARY IN HL AND
0082 3EF0               MVI     A,0F0H  ;SET A TO # OF DIGITS
0084 A4                 ANA     H       ;IF H>255, THERE IS NO
0085 C29F00             JNZ     QHOW    ;ROOM FOR NEXT DIGIT
0088 04                 INR     B       ;B COUNTS # OF DIGITS
0089 C5                 PUSH    B
008A 44                 MOV     B,H     ;HL=10*HL+(NEW DIGIT)
008B 4D                 MOV     C,L
008C 29                 DAD     H       ;WHERE 10* IS DONE BY
008D 29                 DAD     H       ;SHIFT AND ADD
008E 09                 DAD     B
008F 29                 DAD     H
0090 1A                 LDAX    D       ;AND (DIGIT) IS FROM
0091 13                 INX     D       ;STRIPPING THE ASCII
0092 E60F               ANI     0FH     ;CODE
0094 85                 ADD     L
0095 6F                 MOV     L,A
0096 3E00               MVI     A,0
0098 8C                 ADC     H
0099 67                 MOV     H,A
009A C1                 POP     B
009B 1A                 LDAX    D       ;DO THIS DIGIT AFTER
009C F27C00             JP      TN1     ;DIGIT  S SAYS OVERFLOW
009F D5         QHOW:   PUSH    D       ;*** ERROR: "HOW?" ***
00A0 11A600     AHOW:   LXI     D,HOW
00A3 C3CA04             JMP     ERROR
00A6 484F573F HOW:      DB      'HOW?'
00AA 0D                 DB      CR
00AB 4F4B       OK:     DB      'OK'
00AD 0D                 DB      CR
00AE 57484154 WHAT:     DB      'WHAT?'
00B2 3F
00B3 0D                 DB      CR
00B4 534F5252 SORRY:    DB      'SORRY'
00B8 59
00B9 0D                 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
                RSTART:
00BA 310014             LXI  SP,STACK
00BD CD0E00     ST1:    CALL CRLF       ;AND JUMP TO HERE
00C0 11AB00             LXI  D,OK       ;DE->STRING
00C3 97                 SUB  A          ;A=0
00C4 CD6005             CALL PRTSTG     ;PRINT STRING UNTIL CR

00C7 21CE00             LXI  H,ST2+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:    MVI  A,3EH      ;PROMPT ">" AND
00D8 CDFA04             CALL GETLN      ;READ A LINE
00DB 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   DIRECT
00E9 1B                 DCX  D          ;BACKUP DE AND SAVE
00EA 7C                 MOV  A,H        ;VALUE OF LINE # THERE
00EB 12                 STAX D
00EC 1B                 DCX  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 93                 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
00FF C1                 POP  B          ;BC->LINE TO BE DELETED
0100 2A1510             LHLD TXTUNF     ;HL->UNFILLED SAVE AREA
0103 CDE505             CALL MVUP       ;MOVE UP TO DELETE
0106 60                 MOV  H,B        ;TXTUNF->UNFILLED AREA
0107 69                 MOV  L,C
0108 221510             SHLD TXTUNF     ;UPDATE
010B 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               CPI  3          ;THEN DO NOT INSERT
0113 CABA00             JZ   RSTART     ;MUST CLEAR THE STACK
0116 85                 ADD  L          ;COMPUTE NEW TXTUNF
0117 6F                 MOV  L,A
0118 3E00               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 CDEE05             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 TRANSFERED TO THESE POINTS VIA THE
                ; COMMAND TABLE LOOKUP CODE OF 'DIRECT' AND 'EXEC' IN LAST
                ; SECTION.  AFTER THE COMMAND IS EXECUTED, CONTROL IS
                ; TRANSFERED 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.
                ; 'RUNSML' CONTINUES THE EXECUTION ON SAME LINE.
                ;
                ; 'GOTO EXPR(CR)' EVALUATES THE EXPRESSION, FIND THE TARGET
                ; LINE, AND JUMP TO 'RUNTSL' TO DO IT.
                ;
0132 CDC204     NEW:    CALL ENDCHK     ;*** NEW(CR) ***
0135 211710             LXI  H,TXTBGN
0138 221510             SHLD TXTUNF
013B CDC204     STOP:   CALL ENDCHK     ;*** STOP(CR) ***
013E C3BA00             JMP  RSTART
                ;
0141 CDC204     RUN:    CALL ENDCHK     ;*** RUN(CR) ***
0144 111710             LXI  D,TXTBGN   ;FIRST SAVED LINE
                ;
                RUNNXL:
0147 210000             LXI  H,0        ;*** RUNNXL ***
014A CD4005             CALL FNDLP      ;FIND WHATEVER LINE #
014D DABA00             JC   RSTART     ;C:PASSED TXTUNF, QUIT
                ;
                RUNTSL:
0150 EB                 XCHG            ;*** RUNTSL ***
0151 220110             SHLD CURRNT     ;SET 'CURRNT'->LINE #
0154 EB                 XCHG
0155 13                 INX  D          ;BUMP PASS LINE #
0156 13                 INX  D
                ;
                RUNSML:
0157 CD8406             CALL CHKIO      ;*** RUNSML ***
015A 21BD06             LXI  H,TAB2-1   ;FIND COMMAND IN TAB2
015D C33B07             JMP  EXEC       ;AND EXECUTE IT
                ;
0160 DF         GOTO:   RST  3          ;*** GOTO EXPR ***
0161 D5                 PUSH D          ;SAVE FOR ERROR ROUTINE
0162 CDC204             CALL ENDCHK     ;MUST FIND A CR
0165 CD3805             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 EXPRESIONS, FORMATS, BACK-
                ; ARROWS, AND STRINGS.  THESE ITEMS ARE SEPERATED BY COMMAS.
                ;
                ; A FORMAT IS A FOUND SIGN FOLLOWED BY A NUMBER.  IT CONTROLS
                ; THE NUMBER OF SPACES THE VALUE OF A EXPRESION 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.
                ;
016F CD7700     LIST:   CALL TSTNUM     ;TEST IF THERE IS A #
0172 CDC204             CALL ENDCHK     ;IF NO # WE GET A 0
0175 CD3805             CALL FNDLN      ;FIND THIS OR NEXT LINE
0178 DABA00     LS1:    JC   RSTART     ;C:PASSED TXTUNF
017B CDD205             CALL PRTLN      ;PRINT THE LINE
017E CD8406             CALL CHKIO      ;STOP IF HIT CONTROL-C
0181 CD4005             CALL FNDLP      ;FIND NEXT LINE
0184 C37801             JMP  LS1        ;AND LOOP BACK
                ;
0187 0E06       PRINT:  MVI  C,6        ;C = # OF SPACES
0189 CF                 RST  1          ;IF NULL LIST & ";"
018A 3B                 DB   3BH
018B 06                 DB   PR2-$-1
018C CD0E00             CALL CRLF       ;GIVE CR-LF AND
018F C35701             JMP  RUNSML     ;CONTINUE SAME LINE
0192 CF         PR2:    RST  1          ;IF NULL LIST (CR)
0193 0D                 DB   CR
0194 06                 DB   PR0-$-1
0195 CD0E00             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 05                 DB   PR1-$-1
019E DF                 RST  3          ;YES, EVALUATE EXPR.
019F 4D                 MOV  C,L        ;AND SAVE IT IN C
01A0 C3A901             JMP  PR3        ;LOOK FOR MORE TO PRINT
01A3 CD6C05     PR1:    CALL QTSTG      ;OR IS IT A STRING?
01A6 C3B601             JMP  PR8        ;IF NOT, MUST BE EXPR.
01A9 CF         PR3:    RST  1          ;IF ",", GO FIND NEXT
01AA 2C                 DB   ','
01AB 06                 DB   PR6-$-1
01AC CDB304             CALL FIN        ;IN THE LIST.
01AF C39B01             JMP  PR0        ;LIST CONTINUES
01B2 CD0E00     PR6:    CALL CRLF       ;LIST ENDS
01B5 F7                 RST  6
01B6 DF         PR8:    RST  3          ;EVALUATE THE EXPR
01B7 C5                 PUSH B
01B8 CD9205             CALL PRTNUM     ;PRINT THE VALUE
01BB C1                 POP  B
01BC C3A901             JMP  PR3        ;MORE TO PRINT?

                ;*****************************************************************
                ;
                ; *** GOSUB *** & RETURN ***
                ;
                ; 'GOSUB EXPR;' OR 'GOSUB EXPR (CR)' IS LIKE THE 'GOTO'
                ; COMMAND, EXCEPT THAT THE CURRENT TEXT POINTER, STACK POINTER
                ; ETC. ARE SAVE 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)' UNDOS EVERYTHING THAT 'GOSUB' DID, AND THUS
                ; RETURN THE EXCUTION 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.
                ;
01BF CD1906     GOSUB:  CALL PUSHA      ;SAVE THE CURRENT "FOR"
01C2 DF                 RST  3          ;PARAMETERS
01C3 D5                 PUSH D          ;AND TEXT POINTER
01C4 CD3805             CALL FNDLN      ;FIND THE TARGET LINE
01C7 C2A000             JNZ  AHOW       ;NOT THERE.  SAY "HOW?"
01CA 2A0110             LHLD CURRNT     ;FOUND IT, SAVE OLD
01CD E5                 PUSH H          ;'CURRNT' OLD 'STKGOS'
01CE 2A0310             LHLD STKGOS
01D1 E5                 PUSH H
01D2 210000             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:
01DF CDC204             CALL ENDCHK     ;THERE MUST BE A CR
01E2 2A0310             LHLD STKGOS     ;OLD STACK POINTER
01E5 7C                 MOV  A,H        ;0 MEANS NOT EXIST
01E6 B5                 ORA  L
01E7 CAC604             JZ   QWHAT      ;SO, WE SAY: "WHAT?"
01EA F9                 SPHL            ;ELSE, RESTORE IT
01EB E1                 POP  H
01EC 220310             SHLD STKGOS     ;AND THE OLD 'STKGOS'
01EF E1                 POP  H
01F0 220110             SHLD CURRNT     ;AND THE OLD 'CURRNT'
01F3 D1                 POP  D          ;OLD TEXT POINTER
01F4 CDFD05             CALL POPA       ;OLD "FOR" PARAMETERS
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 NECESSARILLY 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 ARER
                ; IS PURGED AND EXECUTION CONTINUES.
                ;
01F8 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 211307             LXI  H,TAB5-1   ;USE 'EXEC' TO LOOK
0205 C33B07             JMP  EXEC       ;FOR THE WORD 'TO'
0208 DF         FR1:    RST  3          ;EVALUATE THE LIMIT
0209 220D10             SHLD LOPLMT     ;SAVE THAT
020C 211907             LXI  H,TAB6-1   ;USE 'EXEC' TO LOOK
020F C33B07             JMP  EXEC       ;FOR THE WORD 'STEP'
0212 DF         FR2:    RST  3          ;FOUND IT, GET STEP
0213 C31902             JMP  FR4
0216 210100     FR3:    LXI  H,1H       ;NOT FOUND, SET TO 1
0219 220B10     FR4:    SHLD LOPINC     ;SAVE THAT TOO
021C 2A0110     FR5:    LHLD CURRNT     ;SAVE CURRENT LINE #
021F 220F10             SHLD LOPLN
0222 EB                 XCHG            ;AND TEXT POINTER
0223 221110             SHLD LOPPT
0226 010A00             LXI  B,0AH      ;DIG INTO STACK TO
0229 2A0910             LHLD LOPVAR     ;FIND 'LOPVAR'
022C EB                 XCHG

022D 60                 MOV  H,B
022E 68                 MOV  L,B        ;HL=0 NOW
022F 39                 DAD  SP         ;HERE IS THE STACK
0230 3E                 DB   3EH
0231 09         FR7:    DAD  B          ;EACH LEVEL IS 10 DEEP
0232 7E                 MOV  A,M        ;GET THAT OLD 'LOPVAR'
0233 23                 INX  H
0234 B6                 ORA  M
0235 CA5202             JZ   FR8        ;0 SAYS NO MORE IN IT
0238 7E                 MOV  A,M
0239 2B                 DCX  H
023A BA                 CMP  D          ;SAME AS THIS ONE?
023B C23102             JNZ  FR7
023E 7E                 MOV  A,M        ;THE OTHER HALF?
023F BB                 CMP  E
0240 C23102             JNZ  FR7
0243 EB                 XCHG            ;YES, FOUND ONE
0244 210000             LXI  H,0H
0247 39                 DAD  SP         ;TRY TO MOVE SP
0248 44                 MOV  B,H
0249 4D                 MOV  C,L
024A 210A00             LXI  H,0AH
024D 19                 DAD  D
024E CDEE05             CALL MVDOWN     ;AND PURGE 10 WORDS
0251 F9                 SPHL            ;IN THE STACK
0252 2A1110     FR8:    LHLD LOPPT      ;JOB DONE, RESTORE DE
0255 EB                 XCHG
0256 F7                 RST  6          ;AND CONTINUE
                ;
0257 FF         NEXT:   RST  7          ;GET ADDRESS OF VAR.
0258 DAC604             JC   QWHAT      ;NO VARIABLE, "WHAT?"
025B 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 CAC704             JZ   AWHAT      ;SO WE ASK: "WHAT?"
0268 E7                 RST  4          ;ELSE WE CHECK THEM
0269 CA7602             JZ   NX3        ;OK, THEY AGREE
026C D1                 POP  D          ;NO, LET'S SEE
026D CDFD05             CALL POPA       ;PURGE CURRENT LOOP
0270 2A0510             LHLD VARNXT     ;AND POP ONE LEVEL
0273 C35E02             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 2A0B10             LHLD LOPINC
027C E5                 PUSH H
027D 7C                 MOV  A,H

027E AA                 XRA  D
027F 7A                 MOV  A,D
0280 19                 DAD  D          ;ADD ONE STEP
0281 FA8802             JM   NX4
0284 AC                 XRA  H
0285 FAAA02             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 CD9804     NX1:    CALL CKHLDE     ;COMPARE WITH LIMIT
029B D1                 POP  D          ;RESTORE TEXT POINTER
029C DAAC02             JC   NX2        ;OUTSIDE LIMIT
029F 2A0F10             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 CDFD05     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) SEPERATED 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 FOLLOWS 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 PROCEDED 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 SEPERATED BY COMMAS.
                ; EACH ITEM CONSISTS OF A VARIABLE, AN EQUAL SIGN, AND AN EXPR.
                ; TBI EVALUATES THE EXPR. AND SET THE VARIBLE TO THAT VALUE.
                ; TBI WILL ALSO HANDLE 'LET' COMMAND WITHOUT THE WORD 'LET'.
                ; THIS IS DONE BY 'DEFLT'.
                ;
02B0 210000     REM:    LXI  H,0H       ;*** REM ***
02B3 3E                 DB   3EH
                ;
02B4 DF         IFF:    RST  3          ;*** IF ***
02B5 7C                 MOV  A,H        ;IS THE EXPR.=0?
02B6 B5                 ORA  L
02B7 C25701             JNZ  RUNSML     ;NO, CONTINUE
02BA CD5605             CALL FNDSKP     ;YES, SKIP REST OF LINE
02BD D25001             JNC  RUNTSL
02C0 C3BA00             JMP  RSTART
                ;
                INPERR:
02C3 2A0710             LHLD STKINP     ;*** INPERR ***
02C6 F9                 SPHL            ;RESTORE OLD SP
02C7 E1                 POP  H          ;AND OLD 'CURRNT'
02C8 220110             SHLD CURRNT
02CB D1                 POP  D          ;AND OLD TEXT POINTER
02CC D1                 POP  D          ;REDO INPUT
                ;
                INPUT:                  ;*** INPUT ***
02CD D5         IP1:    PUSH D          ;SAVE IN CASE OF ERROR
02CE CD6C05             CALL QTSTG      ;IS NEXT ITEM A STRING?
02D1 C3DB02             JMP  IP2        ;NO
02D4 FF                 RST  7          ;YES, BUT FOLLOWED BY A
02D5 DA1503             JC   IP4        ;VARIABLE?  NO.
02D8 C3EB02             JMP  IP3        ;YES.  INPUT VARIABLE
02DB D5         IP2:    PUSH D          ;SAVE FOR 'PRTSTG'
02DC FF                 RST  7          ;MUST BE VARIABLE NOW
02DD DAC604             JC   QWHAT      ;"WHAT?" IT IS NOT?
02E0 1A                 LDAX D          ;GET READY FOR 'PRTSTG'
02E1 4F                 MOV  C,A
02E2 97                 SUB  A
02E3 12                 STAX D
02E4 D1                 POP  D
02E5 CD6005             CALL PRTSTG     ;PRINT STRING AS PROMPT

02E8 79                 MOV  A,C        ;RESTORE TEXT
02E9 1B                 DCX  D
02EA 12                 STAX D
02EB D5         IP3:    PUSH D          ;SAVE TEXT POINTER
02EC EB                 XCHG
02ED 2A0110             LHLD CURRNT     ;ALSO SAVE 'CURRNT'
02F0 E5                 PUSH H
02F1 21CD02             LXI  H,IP1      ;A NEGATIVE NUMBER
02F4 220110             SHLD CURRNT     ;AS A FLAG
02F7 210000             LXI  H,0H       ;SAVE SP TOO
02FA 39                 DAD  SP
02FB 220710             SHLD STKINP
02FE D5                 PUSH D          ;OLD HL
02FF 3E3A               MVI  A,3AH      ;PRINT THIS TOO
0301 CDFA04             CALL GETLN      ;AND GET A LINE
0304 119D13             LXI  D,BUFFER   ;POINTS TO BUFFER
0307 DF                 RST  3          ;EVALUATE INPUT
0308 00                 NOP             ;CAN BE 'CALL ENDCHK'
0309 00                 NOP
030A 00                 NOP
030B D1                 POP  D          ;OK, GET OLD HL
030C EB                 XCHG
030D 73                 MOV  M,E        ;SAVE VALUE IN VAR.
030E 23                 INX  H
030F 72                 MOV  M,D
0310 E1                 POP  H          ;GET OLD 'CURRNT'
0311 220110             SHLD CURRNT
0314 D1                 POP  D          ;AND OLD TEXT POINTER
0315 F1         IP4:    POP  PSW        ;PURGE JUNK IN STACK
0316 CF                 RST  1          ;IS NEXT CH. ',/?
0317 2C                 DB   ','
0318 03                 DB   IP5-$-1
0319 C3CD02             JMP  IP1        ;YES, MORE ITEMS.
031C F7         IP5:    RST  6
                ;
031D 1A         DEFLT:  LDAX D          ;*** DEFLT ***
031E FE0D               CPI  CR         ;EMPTY LINE IS OK
0320 CA2C03             JZ   LT1        ;ELSE IT IS 'LET'
                ;
0323 CDA004     LET:    CALL SETVAL     ;*** LET ***
0326 CF                 RST  1          ;SET VALUE TO VAR.
0327 2C                 DB   ','
0328 03                 DB   LT1-$-1
0329 C32303             JMP  LET        ;ITEM BY ITEM
032C F7         LT1:    RST  6          ;UNTIL FINISH
                ;
                ;*************************************************************
                ;
                ; *** EXPR ***
                ;

                ; 'EXPR' EVALUATES ARITHMETICAL OR LOGICAL EXPRESSIONS.
                ; <EXPR>::=<EXPR2>
                ;         <EXPR2><REL.OP.><EXPR2>
                ; WHERE <REL.OP.> IS ONE OF THE OPERATORS IN TAB8 AND THE
                ; RESULT OF THESE OPERATIONS IS 1 IF TRUE AND 0 IF FALSE.
                ; <EXPR2>::=(+ OR -)<EXPR3>(+ OR -<EXPR3>)(.....)
                ; WHERE () ARE OPTIONAL AND (.....) ARE OPTIONAL REPEATS.
                ; <EXPR3>::=<EXPR4>(* OR /><EXPR4>)(.....)
                ; <EXPR4>::=<VARIABLE>
```

```
                ;            <FUNCTION>
                ;            (<EXPR>)
                ; <EXPR> IS RECURSIVE SO THAT VARIABLE '@' CAN HAVE AN <EXPR>
                ; AS INDEX, FUNCTIONS CAN HAVE AN <EXPR> AS ARGUMENTS, AND
                ; <EXPR4> CAN BE AN <EXPR> IN PARANTHESE.
                ;
                ;                    EXPR   CALL EXPR2      THIS IS AT LOC. 18
                ;                           PUSH HL         SAVE <EXPR2> VALUE
022D 212107     EXPR1: LXI  H,TAB8-1 ;LOOKUP REL OP.
0230 C33B07            JMP  EXEC     ;GO DO IT
0233 CD5C03     XP11:  CALL XP18     ;REL. OP. ">="
0236 D8                RC            ;NO, RETURN HL=0
0237 6F                MOV  L,A      ;YES, RETURN HL=1
0238 C9                RET
0239 CD5C03     XP12:  CALL XP18     ;REL. OP. "#"
023C C8                RZ            ;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      ;SET HL=1
024A C8                RZ            ;REL. TRUE, RETURN
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           ;RETURN HL=<EXPR2>
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 CD7102            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
026A D1                POP  D        ;RESTORE TEXT POINTER
026B 210000            LXI  H,0H     ;SET HL=0, A=1
026E 3E01              MVI  A,1
0270 C9                RET
                ;
0271 CF         EXPR2: RST  1        ;NEGATIVE SIGN?
0272 2D                DB   '-'
0273 06                DB   XP21-$-1
0274 210000            LXI  H,0H     ;YES, FAKE '0-'
0277 C39B02            JMP  XP26     ;TREAT LIKE SUBTRACT
027A CF         XP21:  RST  1        ;POSITIVE SIGN?  IGNORE
027B 2B                DB   '+'
027C 00                DB   XP22-$-1
027D CDA503     XP22:  CALL EXPR3    ;1ST <EXPR3>
0280 CF         XP23:  RST  1        ;ADD?
0281 1B                DB   '+'
0282 15                DB   XP25-$-1
0283 E5                PUSH H        ;YES, SAVE VALUE
0284 CDA503            CALL EXPR3    ;GET 2ND <EXPR3>
0287 EB         XP24:  XCHG          ;2ND IN DE
0288 E3                XTHL          ;1ST IN HL
0289 7C                MOV  A,H      ;COMPARE SIGN
028A AA                XRA  D
028B 7A                MOV  A,D
028C 19                DAD  D
028D D1                POP  D        ;RESTORE TEXT POINTER
028E FA8002            JM   XP23     ;1ST 2ND SIGN DIFFER
0291 AC                XRA  H        ;1ST 2ND SIGN EQUAL
0292 F28002            JP   XP23     ;SO IS RESULT
0295 C39F00            JMP  OHOW     ;ELSE WE HAVE OVERFLOW
0298 CF         XP25:  RST  1        ;SUBTRACT?
0299 2D                DB   '-'
029A 86                DB   XP42-$-1
029B E5         XP26:  PUSH H        ;YES, SAVE 1ST <EXPR3>
029C CDA503            CALL EXPR3    ;GET 2ND <PR3>
029F CD8604            CALL CHGSGN   ;NEGATE
02A2 C38703            JMP  XP24     ;AND ADD THEM
                ;
02A5 CD0504     EXPR3: CALL EXPR4    ;GET 1ST <EXPR4>
02A8 CF         XP31:  RST  1        ;MULTIPLY?
02A9 2A                DB   '*'
02AA 2D                DB   XP34-$-1
02AB E5                PUSH H        ;YES, SAVE 1ST
02AC CD0504            CALL EXPR4    ;AND GET 2ND <EXPR4>
02AF 0600              MVI  B,0H     ;CLEAR B FOR SIGN
02B1 CD8704            CALL CHKSGN   ;CHECK SIGN
02B4 E3                XTHL          ;1ST IN HL
02B5 CD8304            CALL CHKSGN   ;CHECK SIGN OF 1ST
02B8 EB                XCHG
02B9 E3                XTHL
02BA 7C                MOV  A,H      ;IS HL > 255 ?
02BB B7                ORA  A
02BC CAC502            JZ   XP32     ;NO
02BF 7A                MOV  A,D      ;YES, HOW ABOUT DE
02C0 B2                ORA  D
02C1 EB                XCHG          ;PUT SMALLER IN HL
02C2 C2A000            JNZ  AHOW     ;ALSO >, WILL OVERFLOW
02C5 7D         XP32:  MOV  A,L      ;THIS IS DUMB
02C6 210000            LXI  H,0H     ;CLEAR RESULT
02C9 B7                ORA  A        ;ADD AND COUNT
02CA CAF702            JZ   XP35
02CD 19         XP33:  DAD  D
02CE DAA000            JC   AHOW     ;OVERFLOW
02D1 3D                DCR  A
02D2 C2CD02            JNZ  XP33
02D5 C3F703            JMP  XP35     ;FINISHED
02D8 CF         XP34:  RST  1        ;DIVIDE?
02D9 2F                DB   '/'
02DA 46                DB   XP42-$-1
02DB E5                PUSH H        ;YES, SAVE 1ST <EXPR4>
02DC CD0504            CALL EXPR4    ;AND GET 2ND ONE
02DF 0600              MVI  B,0H     ;CLEAR B FOR SIGN
02E1 CD8304            CALL CHKSGN   ;CHECK SIGN OF 2ND
02E4 E3                XTHL          ;GET 1ST IN HL
02E5 CD8304            CALL CHKSGN   ;CHECK SIGN OF 1ST
02E8 EB                XCHG
02E9 E3                XTHL
02EA EB                XCHG
02EB 7A                MOV  A,D      ;DIVIDE BY 0?
02EC B3                ORA  E
02ED CAA000            JZ   AHOW     ;SAY "HOW?"
02F0 C5                PUSH B        ;ELSE SAVE SIGN
02F1 CD6604            CALL DIVIDE   ;USE SUBROUTINE
02F4 60                MOV  H,B      ;RESULT IN HL NOW
02F5 69                MOV  L,C
02F6 C1                POP  B        ;GET SIGN BACK
02F7 D1         XP35:  POP  D        ;AND TEXT POINTER
02F8 7C                MOV  A,H      ;HL MUST BE +
02F9 B7                ORA  A
02FA FA9F00            JM   OHOW     ;ELSE IT IS OVERFLOW
02FD 78                MOV  A,B
02FE B7                ORA  A
02FF FC8604            CM   CHGSGN   ;CHANGE SIGN IF NEEDED
0302 C3A803            JMP  XP31     ;LOOK FOR MORE TERMS
                ;
0305 210107     EXPR4: LXI  H,TAB4-1 ;FIND FUNCTION IN TAB4
0308 C33B07            JMP  EXEC     ;AND GO DO IT
030B FF         XP40:  RST  7        ;NO, NOT A FUNCTION
030C DA1404            JC   XP41     ;NOR A VARIABLE
030F 7E                MOV  A,M      ;VARIABLE
0310 23                INX  H
0311 66                MOV  H,M      ;VALUE IN HL
0312 6F                MOV  L,A
0313 C9                RET
0314 CD7700     XP41:  CALL TSTNUM   ;OR IS IT A NUMBER
0317 78                MOV  A,B      ;# OF DIGIT
0318 B7                ORA  A
0319 C0                RNZ           ;OK
031A CF         PARN:  RST  1        ;NO DIGIT, MUST BE
031B 28                DB   '('
031C 05                DB   XP43-$-1
031D DF                RST  3        ;"(EXPR)"
031E CF                RST  1
031F 29                DB   ')'
0320 01                DB   XP43-$-1
0321 C9         XP42:  RET
0322 C3C604     XP43:  JMP  QWHAT     ;ELSE SAY: "WHAT?"
                ;
0325 CD1A04     RND:   CALL PARN      ;*** RND(EXPR) ***
0328 7C                MOV  A,H      ;EXPR MUST BE +
0329 B7                ORA  A
032A FA9F00            JM   OHOW
032D B5                ORA  L        ;AND NON-ZERO
032E CA9F00            JZ   OHOW
0331 D5                PUSH D        ;SAVE BOTH
0332 E5                PUSH H
0333 2A1310            LHLD RANPNT   ;GET MEMORY AS RANDOM
0336 116907            LXI  D,LSTROM ;NUMBER
0339 E7                RST  4
033A DA4004            JC   RA1      ;WRAP AROUND IF LAST
033D 210000            LXI  H,START
0340 5E         RA1:   MOV  E,M
0341 23                INX  H
0342 56                MOV  D,M
0343 221310            SHLD RANPNT
0346 E1                POP  H
0347 EB                XCHG
0348 C5                PUSH B
0349 CD6604            CALL DIVIDE   ;RND(N)=MOD(M,N)+1
034C C1                POP  B
034D D1                POP  D
034E 23                INX  H
                ;
034F C9                RET
                ;
0350 CD1A04     ABS:   CALL PARN      ;*** ABS(EXPR) ***
0353 1B                DCX  D
0354 CD8204            CALL CHKSGN   ;CHECK SIGN
0357 13                INX  D
0358 C9                RET
0359 2A1510     SIZE:  LHLD TXTUNF    ;*** SIZE ***
035C D5                PUSH D        ;GET THE NUMBER OF FREE
035D EB                XCHG          ;BYTES BETWEEN 'TXTUNF'
035E 216613            LXI  H,VARBGN ;AND 'VARBGN'
0361 CD7C04            CALL SUBDE
0364 D1                POP  D
0365 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' CHNGES 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
                ; CASE, HL DE ARE THEN COMPARED TO SET THE FLAGS.
                ;
                DIVIDE:
0366 E5                PUSH H        ;*** DIVIDE ***
0367 6C                MOV  L,H      ;DIVIDE H BY DE
0368 2600              MVI  H,0
036A CD7104            CALL DV1
036D 41                MOV  B,C      ;SAVE RESULT IN B
036E 7D                MOV  A,L      ;<REMAINDER+L>/DE
036F E1                POP  H
0370 67                MOV  H,A
0371 0EFF       DV1:   MVI  C,0FFH   ;RESULT IN C
0373 0C         DV2:   INR  C        ;DUMB ROUTINE
0374 CD7C04            CALL SUBDE    ;DIVIDE BY SUBTRACT
0377 D27304            JNC  DV2      ;AND COUNT
037A 19                DAD  D
037B C9                RET
                ;
037C 7D         SUBDE: MOV  A,L      ;*** SUBDE ***
037D 93                SUB  E        ;SUBTRACT DE FROM
037E 6F                MOV  L,A      ;HL
037F 7C                MOV  A,H
0380 9A                SBB  D
0381 67                MOV  H,A
0382 C9                RET
                ;
                CHKSGN:
0383 7C                MOV  A,H      ;*** CHKSGN ***
0384 B7                ORA  A        ;CHECK SIGN OF HL
0385 F0                RP            ;IF -, CHANGE SIGN
                ;
                CHGSGN:
0386 7C                MOV  A,H      ;*** CHGSGN ***
0387 F5                PUSH PSW
0388 2F                CMA           ;CHANGE SIGN OF HL
0389 67                MOV  H,A
038A 7D                MOV  A,L
038B 2F                CMA
038C 6F                MOV  L,A
038D 23                INX  H
038E F1                POP  PSW
038F AC                XRA  H
0390 F29F00            JP   OHOW
0393 78                MOV  A,B      ;AND ALSO FLIP B
0394 EE80              XRI  80H
0396 47                MOV  B,A
0397 C9                RET
```

```
                CKHLDE:
0498 7C                 MOV  A,H
0499 AA                 XRA  D          ;SAME SIGN?
049A F29E04             JP   CK1        ;YES, COMPARE
049D EB                 XCHG            ;NO, XCH AND COMP
049E E7         CK1:    RST  4
049F 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 'CURRNT' WITH A "?"
                ; INSERTED AT WHERE THE OLD TEXT POINTER (SHOULD BE ON TOP
                ; OF THE STACK) POINTS TO.  EXECUTION OF TB IS STOPPED
                ; AND TBI IS RESTARTED.  HOWEVER, IF 'CURRNT' -> ZERO
                ; (INDICATING A DIRECT COMMAND), THE DIRECT COMMAND IS NOT
                ;  PRINTED.  AND IF 'CURRNT' -> 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:
                ; 'QWHAT' SAVES TEXT POINTER IN STACK AND GET MESSAGE "WHAT?"
                ; 'AWHAT' JUST GET MESSAGE "WHAT?" AND JUMP TO 'ERROR'.
                ; 'QSORRY' AND 'ASORRY' DO SAME KIND OF THING.
                ; 'QHOW' AND 'AHOW' IN THE ZERO PAGE SECTION ALSO DO THIS
                ;
                SETVAL:
04A0 FF                 RST  7          ;*** SETVAL ***
04A1 DAC604             JC   QWHAT      ;"WHAT?" NO VARIABLE
04A4 E5                 PUSH H          ;SAVE ADDRESS OF VAR.
04A5 CF                 RST  1          ;PASS "=" SIGN
04A6 3D                 DB   '='
04A7 08                 DB   SV1-$-1
04A8 DF                 RST  3          ;EVALUATE EXPR.
04A9 44                 MOV  B,H        ;VALUE IN BC NOW
04AA 4D                 MOV  C,L
04AB E1                 POP  H          ;GET ADDRESS
04AC 71                 MOV  M,C        ;SAVE VALUE
04AD 23                 INX  H
04AE 70                 MOV  M,B
04AF C9                 RET
04B0 C3C604     SV1:    JMP  QWHAT      ;NO "=" SIGN
                ;
04B3 CF         FIN:    RST  1          ;*** FIN ***
04B4 3B                 DB   3BH
04B5 04                 DB   FI1-$-1
04B6 F1                 POP  PSW        ;";", PURGE RET ADDR.
04B7 C35701             JMP  RUNSML     ;CONTINUE SAME LINE
04BA CF         FI1:    RST  1          ;NOT ";", IS IT CR?
04BB 0D                 DB   CR
04BC 04                 DB   FI2-$-1
04BD F1                 POP  PSW        ;YES, PURGE RET ADDR.
04BE C34701             JMP  RUNNXL     ;RUN NEXT LINE
04C1 C9         FI2:    RET             ;ELSE RETURN TO CALLER
                ;
                ENDCHK:
04C2 EF                 RST  5          ;*** ENDCHK ***
04C3 FE0D               CPI  CR         ;END WITH CR?
04C5 C8                 RZ              ;OK, ELSE SAY: "WHAT?"
                ;
04C6 D5         QWHAT:  PUSH D          ;*** QWHAT ***
04C7 11AE00     AWHAT:  LXI  D,WHAT     ;*** AWHAT ***
04CA 97         ERROR:  SUB  A          ;*** ERROR ***
04CB CD6005             CALL PRTSTG     ;PRINT 'WHAT??', 'HOW?'
04CE D1                 POP  D          ;OR 'SORRY'
04CF 1A                 LDAX D          ;SAVE THE CHARACTER
04D0 F5                 PUSH PSW        ;AT WHERE OLD DE ->
04D1 97                 SUB  A          ;AND PUT A 0 THERE
04D2 12                 STAX D
04D3 2A0110             LHLD CURRNT     ;GET CURRENT LINE #
04D6 E5                 PUSH H
04D7 7E                 MOV  A,M        ;CHECK THE VALUE
04D8 23                 INX  H
04D9 B6                 ORA  M
04DA D1                 POP  D
04DB CABA00             JZ   RSTART     ;IF ZERO, JUST RERSTART
04DE 7E                 MOV  A,M        ;IF NEGATIVE,
04DF B7                 ORA  A
04E0 FAC202             JM   INPERR     ;REDO INPUT
04E3 CDD205             CALL PRTLN      ;ELSE PRINT THE LINE
04E6 1B                 DCX  D          ;UPTO WHERE THE 0 IS
04E7 F1                 POP  PSW        ;RESTORE THE CHARACTER
04E8 12                 STAX D
04E9 3E3F               MVI  A,3FH      ;PRINT A "?"
04EB D7                 RST  2
04EC 97                 SUB  A          ;AND THE REST OF THE
04ED CD6005             CALL PRTSTG
04F0 C3BA00             JMP  RSTART
04F3 D5         QSORRY: PUSH D          ;*** QSORRY ***
04F4 11B400     ASORRY: LXI  D,SORRY    ;*** ASORRY ***
04F7 C3CA04             JMP  ERROR
                ;
                ;*****************************************************************
                ;
                ; *** GETLN *** FNDLN (& FRIENDS) ***
                ;
                ; 'GETLN' READS A INPUT LINE INTO 'BUFFER'.  IT FIRST PROMPT
                ; THE CHARACTER IN A (GIVEN BY THE CALLER), THEN IT FILLS THE
                ; THE BUFFER AND ECHOS.  IT IGNORES LF'S AND NULLS, BUT STILL
                ; ECHOS THEM BACK.  RUB-OUT IS USED TO CAUSE IT TO DELETE
                ; THE LAST CHARATER (IF THERE IS ONE), AND ALT-MOD IS USED TO
                ; CAUSE IT TO DELETE THE WHOLE LINE AND START IT ALL OVER.
                ; CR SIGNALS 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.
                ; 'FNDLNP' 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 STRART SEARCH.
                ;
04FA D7         GETLN:  RST  2          ;*** GETLN ***
04FB 119D12             LXI  D,BUFFER   ;PROMPT AND INIT.
04FE CD8406     GL1:    CALL CHKIO      ;CHECK KEYBOARD
0501 CAFE04             JZ   GL1        ;NO INPUT, WAIT
0504 FE7F               CPI  7FH        ;DELETE LAST CHARACTER?
0506 CA2305             JZ   GL3        ;YES
0509 D7                 RST  2          ;INPUT, ECHO BACK
050A FE0A               CPI  0AH        ;IGNORE LF
050C CAFE04             JZ   GL1
050F B7                 ORA  A          ;IGNORE NULL
0510 CAFE04             JZ   GL1
0513 FE7D               CPI  7DH        ;DELETE THE WHOLE LINE?
0515 CA3005             JZ   GL4        ;YES
0518 12                 STAX D          ;ELSE, SAVE INPUT
0519 13                 INX  D          ;AND BUMP POINTER
051A FE0D               CPI  0DH        ;WAS IT CR?
051C C8                 RZ              ;YES, END OF LINE
051D 7B                 MOV  A,E        ;ELSE, MORE FREE ROOM?
051E FEDD               CPI  BUFEND AND 0FFH
0520 C2FE04             JNZ  GL1        ;YES, GET NEXT INPUT
0523 7B         GL3:    MOV  A,E        ;DELETE LAST CHARACTER
0524 FE9D               CPI  BUFFER AND 0FFH
0526 CA3005             JZ   GL4        ;NO, REDO WHOLE LINE
0529 1B                 DCX  D          ;YES, BACKUP POINTER
052A 3E5C               MVI  A,5CH      ;AND ECHO A BACK-SLASH
052C D7                 RST  2
052D C3FE04             JMP  GL1        ;GO GET NEXT INPUT
0530 CD0E00     GL4:    CALL CRLF       ;REDO ENTIRE LINE
0533 3E5E               MVI  A,05EH     ;CR, LF AND UP-ARROW
0535 C3FA04             JMP  GETLN
                ;
0538 7C         FNDLN:  MOV  A,H        ;*** FNDLN ***
0539 B7                 ORA  A          ;CHECK SIGN OF HL
053A FA9F00             JM   QHOW       ;IT CANNOT BE -
053D 111710             LXI  D,TXTBGN   ;INIT. TEXT POINTER
                ;
                FNDLP:                  ;*** FNDLNP ***
0540 E5         FL1:    PUSH H          ;SAVE LINE #
                ;
0541 2A1510             LHLD TXTUNF     ;CHECK IF WE PASSED END
0544 2B                 DCX  H
0545 E7                 RST  4
0546 E1                 POP  H          ;GET LINE # BACK
0547 D8                 RC              ;C,NZ PASSED END
0548 1A                 LDAX D          ;WE DID NOT, GET BYTE 1
0549 95                 SUB  L          ;IS THIS THE LINE?
054A 47                 MOV  B,A        ;COMPARE LOW ORDER
054B 13                 INX  D
054C 1A                 LDAX D          ;GET BYTE 2
054D 9C                 SBB  H          ;COMPARE HIGH ORDER
054E DA5505             JC   FL2        ;NO, NOT THERE YET
0551 1B                 DCX  D          ;ELSE WE EITHER FOUND
0552 B0                 ORA  B          ;IT, OR IT IS NOT THERE
0553 C9                 RET             ;NC,Z:FOUND; NC,NZ:NO
                ;
                FNDNXT:                 ;*** FNDNXT ***
0554 13                 INX  D          ;FIND NEXT LINE
0555 13         FL2:    INX  D          ;JUST PASSED BYTE 1 & 2
                ;
                FNDSKP:
0556 1A         ;       LDAX D          ;*** FNDSKP ***
0557 FE0D               CPI  CR         ;TRY TO FIND CR
0559 C25505             JNZ  FL2        ;KEEP LOOKING
055C 13                 INX  D          ;FOUND CR, SKIP OVER
055D C34005             JMP  FL1        ;CHECK IF END OF TEXT
                ;
                ;*****************************************************************
                ;
                ; *** PRTSTG *** QTSTG *** 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.
                ;
                ; 'QTSTG' 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 PAD 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:
0560 47                 MOV  B,A        ;*** PRTSTG ***
0561 1A         PS1:    LDAX D          ;GET A CHARACTER
0562 13                 INX  D          ;BUMP POINTER
0563 B8                 CMP  B          ;SAME AS OLD A?
0564 C8                 RZ              ;YES, RETURN
0565 D7                 RST  2          ;ELSE PRINT IT
0566 FE0D               CPI  CR         ;WAS IT A CR?
0568 C26105             JNZ  PS1        ;NO, NEXT
056B C9                 RET             ;YES, RETURN
                ;
056C CF         QTSTG:  RST  1          ;*** QTSTG ***
056D 22                 DB   '"'
056E 0F                 DB   QT3-$-1
056F 3E22               MVI  A,22H      ;IT IS A "
0571 CD6005     QT1:    CALL PRTSTG     ;PRINT UNTIL ANOTHER
0574 FE0D               CPI  CR         ;WAS LAST ONE A CR?
0576 E1                 POP  H          ;RETURN ADDRESS
0577 CA4701             JZ   RUNNXL     ;WAS CR, RUN NEXT LINE
057A 23         QT2:    INX  H          ;SKIP 3 BYTES ON RETURN
057B 23                 INX  H
057C 23                 INX  H
057D E9                 PCHL            ;RETURN
057E CF         QT3:    RST  1          ;IS IT A ' ?
057F 27                 DB   27H
0580 05                 DB   QT4-$-1
0581 3E27               MVI  A,27H      ;YES, DO SAME
0583 C37105             JMP  QT1        ;AS IN "
0586 CF         QT4:    RST  1          ;IS IT BACK-ARROW?
0587 5F                 DB   5FH
0588 08                 DB   QT5-$-1
0589 3E8D               MVI  A,8DH      ;YES, CR WITHOUT LF
058B D7                 RST  2          ;DO IT TWICE TO GIVE
058C D7                 RST  2          ;TTY ENOUGH TIME
058D E1                 POP  H          ;RETURN ADDRESS
058E C37A05             JMP  QT2
0591 C9         QT5:    RET             ;NONE OF ABOVE
                ;
                PRTNUM:
0592 0600               MVI  B,0        ;*** PRTNUM ***
0594 CD8204             CALL CHKSGN     ;CHECK SIGN
0597 F29D05             JP   PN1        ;NO SIGN
059A 062D               MVI  B,'-'      ;B=SIGN
059C 0D                 DCR  C          ;'-' TAKES SPACE
059D D5         PN1:    PUSH D          ;SAVE
059E 110A00             LXI  D,0AH      ;DECIMAL
```

```
05A1 D5          PUSH D          ;SAVE AS A FLAG
05A2 0D          DCR  C          ;C=SPACES
05A3 C5          PUSH B          ;SAVE SIGN & SPACE
05A4 CD6604  PN2:  CALL DIVIDE   ;DEVIDE HL BY 10
05A7 78          MOV  A,B        ;RESULT 0?
05A8 B1          ORA  C
05A9 CAB405      JZ   PN3        ;YES, WE GOT ALL
05AC E3          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 C3A405      JMP  PN2        ;AND DIVIDE BY 10
05B4 C1      PN3:  POP  B        ;WE GOT ALL DIGITS IN
05B5 0D      PN4:  DCR  C        ;THE STACK
05B6 79          MOV  A,C        ;LOOK AT SPACE COUNT
05B7 B7          ORA  A
05B8 FAC105      JM   PN5        ;NO LEADING BLANKS
05BB 3E20        MVI  A,20H      ;LEADING BLANKS
05BD D7          RST  2
05BE C3B505      JMP  PN4        ;MORE?
05C1 78      PN5:  MOV  A,B      ;PRINT SIGN
05C2 B7          ORA  A
05C3 C41000      CNZ  10H
05C6 5D          MOV  E,L        ;LAST REMAINDER IN E
05C7 7B      PN6:  MOV  A,E      ;CHECK DIGIT IN E
05C8 FE0A        CPI  0AH        ;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      PRTLN: LDAX D       ;*** PRTLN ***
05D3 6F          MOV  L,A        ;LOW ORDER LINE #
05D4 13          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 #
05DA CD9205      CALL PRTNUM
05DD 3E20        MVI  A,20H      ;FOLLOWED BY A BLANK
05DF D7          RST  2
05E0 97          SUB  A          ;AND THEN THE TEXT
05E1 CD6005      CALL PRTSTG
05E4 C9          RET
                 ;
                 ;*****************************************************************
                 ;
                 ; *** MVUP *** MVDOWN *** POPA *** & PUSHA ***
                 ;
                 ; '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
                 ;
                 ; 'PUSHA' 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 13          INX  D          ;INCREASE BOTH POINTERS
05EA 03          INX  B
05EB C3E505      JMP  MVUP       ;UNTIL DONE
                 ;
                 MVDOWN:
05EE 78          MOV  A,B        ;*** MVDOWN ***
05EF 92          SUB  D          ;TEST IF DE = BC
05F0 C2F605      JNZ  MD1        ;NO, GO MOVE
05F3 79          MOV  A,C        ;MAYBE, OTHER BYTE?
05F4 93          SUB  E
05F5 C8          RZ              ;YES, RETURN
05F6 1B      MD1:  DCX  D        ;ELSE MOVE A BYTE
05F7 2B          DCX  H          ;BUT FIRST DECREASE
05F8 1A          LDAX D          ;BOTH POINTERS AND
05F9 77          MOV  M,A        ;THEN DO IT
05FA C3EE05      JMP  MVDOWN     ;LOOP BACK
                 ;
05FD C1      POPA:  POP  B       ;BC = RETURN ADDR.
05FE E1          POP  H          ;RESTORE LOPVAR, BUT
05FF 220910      SHLD LOPVAR     ;=0 MEANS NO MORE
0602 7C          MOV  A,H
0603 B5          ORA  L
0604 CA1706      JZ   PP1        ;YEP, GO RETURN
0607 E1          POP  H          ;NOP, RESTORE OTHERS
0608 220E10      SHLD LOPINC
060B E1          POP  H
060C 220D10      SHLD LOPLMT
060F E1          POP  H
0610 220F10      SHLD LOPLN
0613 E1          POP  H
0614 221110      SHLD LOPPT
0617 C5      PP1:  PUSH B        ;BC = RETURN ADDR.
0618 C9          RET
                 ;
0619 21DE12  PUSHA: LXI  H,STKLMT ;*** PUSHA ***
061C CD8604      CALL CHGSGN
                 ;
061F C1          POP  B          ;BC=RETURN ADDRESS
0620 39          DAD  SP         ;IS STACK NEAR THE TOP?
0621 D22F04      JNC  OSORRY     ;YES, SORRY FOR THAT.
0624 2A0910      LHLD LOPVAR     ;ELSE SAVE LOOP VAR.S
0627 7C          MOV  A,H        ;BUT IF LOPVAR IS 0
0628 B5          ORA  L          ;THAT WILL BE ALL
0629 CA2F06      JZ   PU1
062C 2A1110      LHLD LOPPT      ;ELSE, MORE TO SAVE
062F E5          PUSH H
0630 2A0F10      LHLD LOPLN
0633 E5          PUSH H
0634 2A0D10      LHLD LOPLMT
0637 E5          PUSH H
0638 2A0B10      LHLD LOPINC
063B E5          PUSH H
063C 2A0910      LHLD LOPVAR
063F E5      PU1:  PUSH H
0640 C5          PUSH B          ;BC = RETURN ADDR.
0641 C9          RET
                 ;
                 ;*****************************************************************
                 ;
                 ; *** OUTC *** & CHKIO ***
                 ;
                 ; 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 REG.
                 ; 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.  HOWERER, IF THE
                 ; INPUT IS A CONTROL-O, THE 'OCSW' SWITCH IS COMPLIMENTED, 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
0645 3ECF        MVI  A,0CFH'
0647 D3FB,       OUT  0FBH
0649 3E27        MVI  A,27H
064B D3FB        OUT  0FBH
064D 1619        MVI  D,19H
             PATLOP:
064F CD0E00      CALL CRLF
0652 15          DCR  D
0653 C24F06      JNZ  PATLOP
0656 97          SUB  A
0657 11A206      LXI  D,MSG1
065A CD6005      CALL PRTSTG
065D 210000      LXI  H,START
0660 221310      SHLD RANPNT
0663 211710      LXI  H,TXTBGN
0666 221510      SHLD TXTUNF
0669 C3BA00      JMP  RSTART
066C C27106  OC2:  JNZ  OC3        ;IT IS ON
066F F1          POP  PSW          ;IT IS OFF
0670 C9          RET               ;RESTORE AF AND RETURN
0671 DBFB    OC3:  IN   0FBH       ;COME HERE TO DO OUTPUT
0673 E601        ANI  1H           ;STATUS BIT
0675 CA7106      JZ   OC3          ;NOT READY, WAIT
0678 F1          POP  PSW          ;READY, GET OLD A BACK
0679 D3FA        OUT  0FAH         ;AND SEND IT OUT
067B FE0D        CPI  CR           ;WAS IT CR?
067D C0          RNZ               ;NO, FINISHED
067E 3E0A        MVI  A,LF         ;SEND LF
0680 D7          RST  2            ;THIS IS RECURSIVE
0681 3E0D        MVI  A,CR         ;GET CR BACK IN A
0683 C9          RET
0684 DBFB    CHKIO: IN   0FBH      ;*** CHKIO ***
0686 00          NOP               ;STATUS BIT FLIPPED?
0687 E602        ANI  2H           ;MASK STATUS BIT
0689 C8          RZ                ;NOT READY, RETURN "Z"
068A DBFA        IN   0FAH         ;READY, READ DATA
068C E67F        ANI  7FH          ;MASK BIT 7 OFF
068E FE0F        CPI  0FH          ;IS IT CONTROL-O?
0690 C29D06      JNZ  CI1          ;NO, MORE CHECKING
0693 3A0010      LDA  OCSW         ;CONTROL-O FLIPS OCSW
0696 2F          CMA               ;ON TO OFF, OFF TO ON
0697 320010      STA  OCSW
069A C38406      JMP  CHKIO        ;GET ANOTHER INPUT
069D FE03    CI1:  CPI  3H         ;IS IT CONTROL-C?
069F C0          RNZ               ;NO, RETURN "NZ"
06A0 C3BA00      JMP  RSTART       ;YES, RESTART TBI
06A3 54494E59 MSG1:  DB  'TINY '
06A7 20
06A8 42415349        DB  'BASIC'
06AC 43
06AD 0D              DB  CR
                 ;
                 ;*****************************************************************
                 ;
                 ; *** TABLES *** DIRECT *** & EXEC ***
                 ;
                 ; THIS SECTION OF THE CODE TESTS A STRING AGAINST A TABLE.
                 ; WHEN A MATCH IS FOUND, CONTROL IS TRANSFERED TO THE SECTION
                 ; OF CODE ACCORDING TO THE TABLE.
                 ;
                 ; AT 'EXEC', DE SHOULD POINT TO THE STRING AND HL SHOULD PO
                 ; TO THE TABLE-1.  AT 'DIRECT', DE SHOULD POINT TO THE STRI
                 ; HL WILL BE SET UP TO POINT TO TAB1-1, WHICH IS THE TABLE
                 ; ALL DIRECT AND STATEMENT COMMANDS.
                 ;
                 ; A '.' IN THE STRING WILL TERMINATE THE TEST AND THE PARTIA
                 ; MATCH WILL BE CONSIDERED AS A MATCH.  E.G., 'P.', 'PR.',
                 ; 'PRI.', 'PRIN.', OR 'PRINT' 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
06AE 4C495354            DB  'LIST'
                    +     DWA LIST
06B2 81             +     DB      (0016FH SHR 8) +128
06B3 6F             +     DB      0016FH AND 0FFH
                 ;
06B4 52554E              DB  'RUN'
                    +     DWA RUN
06B7 81             +     DB      (00141H SHR 8) +128
06B8 41             +     DB      00141H AND 0FFH
                 ;
06B9 4E4557              DB  'NEW'
                    +     DWA NEW
06BC 81             +     DB      (00132H SHR 8) +128
06BD 32             +     DB      00132H AND 0FFH
                 ;
                 TAB2:                ;DIRECT/STATEMENT
06BE 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 23             +     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 81             +     DB      (00160H SHR 8) +128
06D2 60             +     DB      00160H AND 0FFH
                 ;
06D3 474F5355            DB  'GOSUB'
06D7 42
                    +     DWA GOSUB
06D8 81             +     DB      (001BFH SHR 8) +128
06D9 BF             +     DB      001BFH AND 0FFH
```

```
06DA 52455455          DB  'RETURN'
06DE 524E
                +      DWA RETURN
06E0 81         +      DB      (001DFH SHR 8) +128
06E1 DF         +      DB      001DFH AND 0FFH

06E2 52454D            DB  'REM'
                +      DWA REM
06E5 82         +      DB      (002B0H SHR 8) +128
06E6 B0         +      DB      002B0H AND 0FFH

06E7 464F52            DB  'FOR'
                +      DWA FOR
06EA 81         +      DB      (001F8H SHR 8) +128
06EB F8         +      DB      001F8H AND 0FFH

06EC 494E5055          DB  'INPUT'
06F0 54
                +      DWA INPUT
06F1 82         +      DB      (002CDH SHR 8) +128
06F2 CD         +      DB      002CDH AND 0FFH

06F3 5052494E          DB  'PRINT'
06F7 54
                +      DWA PRINT
06F8 81         +      DB      (00187H SHR 8) +128
06F9 87         +      DB      00187H AND 0FFH

06FA 53544F50          DB  'STOP'
                +      DWA STOP
06FE 81         +      DB      (0013BH SHR 8) +128
06FF 3B         +      DB      0013BH AND 0FFH

                +      DWA DEFLT
0700 83         +      DB      (0031DH SHR 8) +128
0701 1D         +      DB      0031DH AND 0FFH

                TAB4:                  ;FUNCTIONS
0702 524E44            DB  'RND'
                +      DWA RND
0705 84         +      DB      (00425H SHR 8) +128
0706 25         +      DB      00425H AND 0FFH

0707 414253            DB  'ABS'
                +      DWA ABS
070A 84         +      DB      (00450H SHR 8) +128
070B 50         +      DB      00450H AND 0FFH

070C 53495A45          DB  'SIZE'
                +      DWA SIZE
0710 84         +      DB      (00459H SHR 8) +128
0711 59         +      DB      00459H AND 0FFH

                +      DWA XP40
0712 84         +      DB      (0040BH SHR 8) +128
0713 0B         +      DB      0040BH AND 0FFH

                TAB5:                  ;"TO" IN "FOR"
0714 544F              DB  'TO'
                +      DWA FR1
0716 82         +      DB      (00208H SHR 8) +128
0717 08         +      DB      00208H AND 0FFH

                +      DWA QWHAT
0718 84         +      DB      (004C6H SHR 8) +128
0719 C6         +      DB      004C6H AND 0FFH

                TAB6:                  ;"STEP" IN "FOR"
071A 53544550          DB  'STEP'
                +      DWA FR2
071E 82         +      DB      (00212H SHR 8) +128
071F 12         +      DB      00212H AND 0FFH

                +      DWA FR3
0720 82         +      DB      (00216H SHR 8) +128
0721 16         +      DB      00216H AND 0FFH

                TAB8:                  ;RELATION OPERATORS
0722 3E3D              DB  '>='
                +      DWA XP11
0724 83         +      DB      (00333H SHR 8) +128
0725 33         +      DB      00333H AND 0FFH

0726 23                DB  '#'
                +      DWA XP12
0727 83         +      DB      (00339H SHR 8) +128
0728 39         +      DB      00339H AND 0FFH

0729 3E                DB  '>'
                +      DWA XP13
072A 83         +      DB      (0033FH SHR 8) +128
072B 3F         +      DB      0033FH AND 0FFH

072C 3D                DB  '='
                +      DWA XP15
072D 83         +      DB      (0034EH SHR 8) +128
072E 4E         +      DB      0034EH AND 0FFH

072F 3C3D              DB  '<='
                +      DWA XP14
0731 83         +      DB      (00346H SHR 8) +128
0732 46         +      DB      00346H AND 0FFH

0733 3C                DB  '<'
                +      DWA XP16
0734 83         +      DB      (00354H SHR 8) +128
0735 54         +      DB      00354H AND 0FFH

                +      DWA XP17
0736 83         +      DB      (0035AH SHR 8) +128
0737 5A         +      DB      0035AH AND 0FFH

                ;
                ;
                DIRECT:
0738 21AD06             LXI  H,TAB1-1 ;*** DIRECT ***
                ;
                EXEC:                  ;*** EXEC ***
073B EF         EX0:    RST  5         ;IGNORE LEADING BLANKS
073C D5                 PUSH D         ;SAVE POINTER
073D 1A         EX1:    LDAX D         ;IF FOUND '.' IN STRING
073E 13                 INX  D         ;BEFORE ANY MISMATCH
073F FE2E               CPI  2EH       ;WE DECLARE A MATCH
0741 CA5A07             JZ   EX3
0744 23                 INX  H         ;HL->TABLE
0745 BE                 CMP  M         ;IF MATCH, TEST NEXT
0746 CA2D07             JZ   EX1
0749 3E7F               MVI  A,07FH    ;ELSE, SEE IF BIT 7
074B 1B                 DCX  D         ;OF TABLE IS SET, WHICH
074C BE                 CMP  M         ;IS THE JUMP ADDR. (HI)
074D DA6107             JC   EX5       ;C:YES, MATCHED
0750 23         EX2:    INX  H         ;NC:NO, FIND JUMP ADDR.
0751 BE                 CMP  M
0752 D25007             JNC  EX2
0755 23                 INX  H         ;BUMP TO NEXT TAB. ITEM
0756 D1                 POP  D         ;RESTORE STRING POINTER
0757 C33B07             JMP  EX0       ;TEST AGAINST NEXT ITEM

075A 3E7F       EX3:    MVI  A,07FH    ;PARTIAL MATCH, FIND
075C 23         EX4:    INX  H         ;JUMP ADDR., WHICH IS
075D BE                 CMP  M         ;FLAGGED BY BIT 7
075E D25C07             JNC  EX4
0761 7E         EX5:    MOV  A,M       ;LOAD HL WITH THE JUMP
0762 23                 INX  H         ;ADDRESS FROM THE TABLE
0763 6E                 MOV  L,M
0764 E67F               ANI  7FH       ;MASK OFF BIT 7
0766 67                 MOV  H,A
0767 F1                 POP  PSW       ;CLEAN UP THE GARBAGE
0768 E9                 PCHL           ;AND WE GO DO IT
                ;
                LSTROM:
1000                    ORG  1000H     ;HERE ON MUST BE RAM
1000            OCSW:   DS   1         ;SWITCH FOR OUTPUT
1001            CURRNT: DS   2         ;POINTS TO CURRENT LINE
1003            STKGOS: DS   2         ;SAVES SP IN 'GOSUB'
1005            VARNXT: DS   2         ;TEMP STORAGE
1007            STKINP: DS   2         ;SAVES SP IN 'INPUT'
1009            LOPVAR: DS   2         ;'FOR' LOOP SAVE AREA
100B            LOPINC: DS   2         ;INCREMENT
100D            LOPLMT: DS   2         ;LIMIT
100F            LOPLN:  DS   2         ;LINE NUMBER
1011            LOPPT:  DS   2         ;TEXT POINTER
1013            RANPNT: DS   2         ;RANDOM NUMBER POINTER
1015            TXTUNF: DS   2         ;->UNFILLED TEXT AREA
1017            TXTBGN: DS   2         ;TEXT SAVE AREA BEGINS
1366                    ORG  1366H
1366            TXTEND: DS   0         ;TEXT SAVE AREA ENDS
1366            VARBGN: DS   55        ;VARIABLE @(0)
139D            BUFFER: DS   64        ;INPUT BUFFER
13DD            BUFEND: DS   1         ;BUFFER ENDS
13DE            STKLMT: DS   1         ;TOP LIMIT FOR STACK
1400                    ORG  1400H
1400            STACK:  DS   0         ;STACK STARTS HERE
                ;
000D            CR      EQU  0DH
000A            LF      EQU  0AH
0000                    END

ABS    0450     AHOW   00A0     ASORR 04F4     AWHAT 04C7
BUFEN 13DD      BUFFE 139D      CHGSG 0486     CHKIO 0684
CHKSG 0483      CI1    069D     CK1    049E    CKHLD 0498
CR     000D     CRLF   000E     CURRN 1001     DEFLT 031D
DIREC 0738      DIVID 0466      DV1    0471    DV2   0473
DWA    BF78     ENDCH 04C2      ERROR 04CA     EX0   073B
EX1    073D     EX2    0750     EX3    075A    EX4   075C
EX5    0761     EXEC   073B     EXPR1 032D     EXPR2 0371
EXPR3 03A5      EXPR4 0405      FI1    04BA    FI2   04C1
FIN    04B3     FL1    0540     FL2    0555    FNDLN 0538
FNDLP 0540      FNDNX 0554      FNDSK 0556     FOR   01F8
FR1    0208     FR2    0212     FR3    0216    FR4   0219
FR5    021C     FR7    0231     FR8    0252    GETLN 04FA
GL1    04FE     GL3    0523     GL4    0530    GOSUB 01BF
GOTO   0160     HOW    00A6     IFF    02B4    INIT  0642
INPER 02C3      INPUT 02CD      IP1    02CD    IP2   02DB
IP3    02EB     IP4    0315     IP5    031C    LET   0323
LF     000A     LIST   016F     LOPIN 100B     LOPLM 100D
LOPLN 100F      LOPPT 1011      LOPVA 1009     LS1   0178
LSTRO 0769      LT1    032C     MD1    05F6    MSG1  06A3
MVDOW 05EE      MVUP   05E5     NEW    0132    NEXT  0257
NX0    025E     NX1    0298     NX2    02AC    NX3   0276
NX4    0288     NX5    02AA     OC2    066C    OC3   0671
OCSW   1000     OK     00AB     PARN   041A    PATLO 064F
PN1    059D     PN2    05A4     PN3    05B4    PN4   05B5
PN5    05C1     PN6    05C7     POPA   05FD    PP1   0617
PR0    019B     PR1    01A3     PR2    0192    PR3   01A9
PR6    01B2     PR8    01B6     PRINT 0187     PRTLN 05D2
PRTNU 0592      PRTST 0560      PS1    0561    PU1   063F
PUSHA 0619      QHOW   009F     QSORR 04F3     QT1   0571
QT2    057A     QT3    057E     QT4    0586    QT5   0591
QTSTG 056C      QWHAT 04C6      RA1    0440    RANPN 1013
REM    02B0     RETUR 01DF      RND    0425    RSTAR 00BA
RUN    0141     RUNNX 0147      RUNSM 0157     RUNTS 0150
SETVA 04A0      SIZE   0459     SORRY 00B4     SS1   0028
ST1    00BD     ST2    00CD     ST3    00D6    ST4   010B
STACK 1400      START 0000      STKGO 1003     STKIN 1007
STKLM 13DE      STOP   013B     SUBDE 047C     SV1   04B0
TAB1   06AE     TAB2   06BE     TAB4   0702    TAB5  0714
TAB6   071A     TAB8   0722     TC1    0068    TC2   0073
TN1    007C     TSTNU 0077      TV1    0058    TXTBG 1017
TXTEN 1366      TXTUN 1015      VARBG 1366     VARNX 1005
WHAT   00AE     XP11   0333     XP12   0339    XP13  033F
XP14   0346     XP15   034E     XP16   0354    XP17  035A
XP18   035C     XP21   037A     XP22   037D    XP23  0380
XP24   0387     XP25   0398     XP26   039B    XP31  03A8
XP32   03C5     XP33   03CD     XP34   03D8    XP35  03F7
XP40   040B     XP41   0414     XP42   0421    XP43  0422
```

Merry Christmas

Happy New Year