

# Palo Alto Tiny BASIC Version Three

BY LI-CHEN WANG

Article Excerpt from

## PCC's REFERENCE BOOK of PERSONAL and HOME COMPUTING

Edited by  
DWIGHT McCABE

Copyright 1977 by People's Computer Company  
unless otherwise noted.  
All rights reserved.  
Library of Congress Catalog Number: 77-073021  
International Standard Book Number: 0-918790-01-6

First Printing: July 1977

# Palo Alto Tiny BASIC Version Three

BY LI-CHEN WANG

This latest version of Li-Chen Wang's Palo Alto Tiny BASIC will run on either the 8080 or Z-80, and only uses 2K of core memory. It contains a number of nice features including command abbreviations and error messages. At the end of the listing is a cross reference table for symbols used in the program and also the object code for the program. For further information on Tiny BASICs, see Dr. Dobb's Journal Volume 1 for which there is a card at the back. Tapes of version 1 are available from Community Computer Center (see p. 105).

If you are unfamiliar with reading listings, here is the format used for this one:

machine code	line reference numbers	assembly code	comments by line
F0A6 F1	1677	POP PSW	
F0A5	1679	PUSH PSW	
F0A6 Fc03	1681	CPI J	THE LENGTH OF NEW LINE IS 3 (LINE # AND CR) THEN DO NOT INSERT
F0AA CA53F0	1683	JZ RSTART	MUST CLEAR THE STACK COMPUTE NEW TXTUNF
F0A6 9E	1685	ADD A,A	
F0AE 9F	1687	MOV E,A	
F0AF 3E 00	1689	MVI A,0	
F0A6 9C	1691	ADC A,A	
F0B2 57	1693	MOV D,A	DE->NEW UNFILLED AREA CHECK TO SEE IF THERE
F0B3 24 d100	1693	LHLD TXTLMT	
F0B7 EB	1697	KC LD	
F0B7 0EDF4	1699	CALL C0MP	IS ENOUGH SPACE
F0B4 0250F0	1701	JNC GSURRY	SORRY, NO ROOM FOR IT
F0B4 020040	1703	SHLD TXTUNF	OK, UPDATE AND EXIT
F0C0 02	1705	POP D	DE->CLD LNFTILLED AREA
F0C1 C0BF0	1707	CALL MVDWN	
F0C4 D1	1709	POP H	DE->BEGIN, HL->END
F0C4 01	1711	POP M	
F0C6 C00F06	1713	CALL MVUP	MOVE NEW LINE TO SAVE
F0C9 C36BF0	1715	JMP ST2	AREA
*****			
*** DIRECT *** & EXEC ***			
THIS SECTION OF THE CODE TESTS A STRING AGAINST A TABLE. WHEN A MATCH IS FOUND, CONTROL IS TRANSFERRED TO THE SECTION OF CODE ACCORDING TO THE TABLE.			
section comments			

The line reference numbers start arbitrarily at 1438 and have blocks of numbers missing. Their main relevance is for cross referencing the symbols and for keeping the lines of code in order. —Ed.

Palo Alto Tiny BASIC (PATB) is one of the implementations of Tiny BASIC proposed in *People's Computer Company Newspaper* and *Dr. Dobb's Journal*. However, there are some differences between the original proposal and PATB.

- a) FOR - NEXT loop is implemented in PATB.
- b) PRINT format control is implemented in PATB.
- c) PATB uses semi-colons to separate commands and commas to separate items in the same command. E.g.:
- 120 INPUT A, B, C; LET D=5, E=7; PRINT F, G, H
- d) Compare operator can be used in any expression, not restricted to IF commands.
- e) PATB allows prompt strings in the INPUT command. Furthermore, if the input is not valid, the prompt is automatically repeated until a valid input is keyed in.
- f) PATB command keywords may be abbreviated.

---

Palo Alto Tiny BASIC, version 1, was originally published in *Dr. Dobb's Journal of Computer Calisthenics and Orthodontics*, Vol. 1 No. 5, May, 1976.

Version 1.0 of PATB was published in *Dr. Dobb's Journal* (Vol. 1, #5). Version 2.0 was published in *Interface Age* (Vol. 2, #1). The version presented here (Version 3.0) differs from the previous ones in that:

- a) The RST instructions are no longer used as CALL's. This makes the program longer, but enables one to relocate the program to anywhere in the address space. (RST instructions call subroutines on page 0.)
- b) A few JMP instructions are inserted so that the user can extend PATB by changing these JMP's.
- c) Some other small changes in PRINT command.
- d) In Version 1.0 there, are two known bugs: FOR I=1 TO 32767 will never end, and ABS(-32767-1) gives a negative result. In Version 2.0, there is one known bug: ABS(0) gives an error. These bugs are fixed in Version 3.0.

## THE LANGUAGE

### NUMBERS

All numbers are integers and must be between -32767 and 32767.

### VARIABLES

There are 26 variables denoted by letters A through Z. There is also a single array @(!). The dimension of this array (i.e., the range of values of the index !) 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

Without extension, there are only 3 functions. More functions can be added as extensions.

- |        |   |
|--------|---|
| 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) greater than.
<	(compare) less than.
=	(compare) equal to. (Note that to certain versions of BASIC 'LET A=B=0' means 'set both A and B to 0'. To this version of Tiny BASIC, it means 'set A to the result of comparing B with 0'.)
#	(compare) not equal to.
>=	(compare) greater than or equal to.
<=	(compare) less than or equal to.

+, -, \*, and / operations result in a value between -32767 and 32767. (-32768 is also allowed in some cases.) The result of any comparison is 1 if true, 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 - next, and compare operators the last. Parentheses can also be used to alter the order of evaluation.

### STATEMENTS

A Tiny BASIC statement must consist 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 commands to its right in the same statement to be skipped.

## COMMANDS

Tiny BASIC commands (un-extended) are listed below with examples. More commands can be added as extensions. 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 causes the interpreter to ignore the rest of the line. This allows the programmer to put remarks in the program.

### 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, respectively; and set the variable V to either X, Y or 0 depending on whether A is greater than, less than, or equal to B, respectively.

### PRINT COMMAND

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

The first command will print the value of the expression  $A * 3 + 1$  (e.g., 307), the string of characters ABC"123, and the string #'!@%, and then start a new line. Note that either single or double quotes can be used to quote strings, but pairs must be matched. The second command will produce the same output as the first, except that it will not start a new line after the last item is printed. This enables the program to continue printing on the same line with another PRINT command.

Numerical values are printed with leading blanks so that they take 8 spaces each. This field width can be changed by a # sign followed by a number indicating the new width. The new width will be effective until the end of this PRINT command unless changed again by # n. Note that no trailing space is printed. Extra spaces can be generated by repeated commas.

PRINT A, # 3, B,,, C + 1,

This will print the value of A in 8 spaces, the value of B in 3 spaces (more if B > 999 or B < -999), two extra spaces, and the value of C + 1 in 3 spaces (more if C > 998 or C < -100).

PRINT I L, I K, I I,

This command will print the control characters FF (control L), VT (control K), and HT (control I). Control characters can also appear inside quotes, but the method used here makes them visible in the program listing.

### INPUT COMMAND

INPUT A, B

When this command is executed, Tiny BASIC will print A, a space, and wait to read in an expression from the keyboard. The variable A will be set to the value of the input expression. Then B is printed with a space and variable B is set to the value of the next expression read from the keyboard.

INPUT 'WHAT IS YOUR WEIGHT?'A, "YOUR  
HEIGHT?"B

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

In both of the above examples, if the input (at run time) from the keyboard is not a valid expression, Tiny BASIC will reprint the prompt and wait again until a valid expression is entered. One may also choose to reprint only part of the prompt, e.g.:

```
INPUT "WHAT IS", "YOUR WEIGHT?"A,  
"YOUR HEIGHT?"B
```

In this case, WHAT IS YOUR WEIGHT? will be asked the first time; while only the part of YOUR WEIGHT? will be repeated if an invalid input is given.

## 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 and execution continues at the next statement. Note that the word 'THEN' is not used.

## GOTO COMMAND

```
GOTO 120
```

will jump to statement 120 and proceed from that statement on. 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 jump to a statement number as computed from the value of the expression and proceed from that point on.

## GOSUB AND RETURN COMMANDS

GOSUB command is different from GOTO command in that: a) the current statement number and position within the statement is remembered; and b) a semi-colon and other commands can follow GOSUB 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 the statement as computed from the given expression A \* 10 + B.

```
RETURN
```

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

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

## 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 sequence 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 the nest 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 (instead of saved) after you have typed the CR. All commands described above can be used as direct commands. There are three more commands that can be used as direct commands but not as part of a statement:

##### RUN

will start execution of the program starting at the lowest statement number.

##### LIST

will print out all statements in ascending numerical order.

##### LIST 120

will print out all the statements starting at statement 120.

##### LIST 120,5

will print out 5 lines starting at statement 120.

##### NEW

will delete all statements.

#### STOPPING THE EXECUTION

The execution of program or listing of program can be stopped by typing Control-C.

#### 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 with 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 key words are as follows:

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

#### ERROR REPORT

There are only three error conditions in Tiny BASIC. The statement that contains an 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  
where a close parenthesis is missing

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

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

HOW?  
380 GOTO 412?  
where statement 412 does 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 BS (Control H) key.

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

To delete a statement, type the statement number followed immediately by a CR. ■

```
*****
* T B I
* TINY BASIC INTERPRETER
* VERSION 3.0
* FOR 8080 SYSTEM
* LI-CHEN WANG
* 26 APRIL, 1977
*****
```

```
* *** MEMORY USAGE ***
*
```

```
* 0080-01FF ARE FOR VARIABLES, INPUT LINE, AND STACK
* 2000-3FFF ARE FOR TINY BASIC TEXT & ARRAY
* F000-F7FF ARE FOR TBI CODE
*
```

```
1438 BOTSCR EQU 00080H
1440 TOPSCR EQU 00200H
1442 BOTRAM EQU 02000H
1444 DFTLMT EQU 04000H
1446 EOTROM EQU CF000H
```

```
* * DEFINE VARIABLES, BUFFER, AND STACK IN RAM
*
```

C080	1451	ORG	BOTSCR	
C081	1453	KEYWRD	DS 1	WAS INIT DONE?
C083	1455	TXTLMT	DS 2	->LIMIT OF TEXT AREA
C087	1457	VARBGN	DS 2*26	TB VARIABLES A-Z
C089	1459	CURRNT	DS 2	POINTS TO CURRENT LINE
0088	1461	STKGOS	DS 2	SAVES SP IN "GOSUB"
00BD	1463	VARNXT	DS 0	TEMP STORAGE
00BF	1465	STKINP	DS 2	SAVES SP IN "INPUT"
00C1	1467	LOPVAR	DS 2	'FOR' LOOP SAVE AREA
00C3	1469	LOPINC	DS 2	INCREMENT
00C5	1471	LOPLMT	DS 2	LIMIT
00C7	1473	LOPLN	DS 2	LINE NUMBER
00C9	1475	LOPPT	DS 2	TEXT POINTER
00CA	1477	RANPNT	DS 2	RANDOM NUMBER POINTER
014E	1479	DS	1	EXTRA BYTE FOR BUFFER
014E	1481	BUFFER	DS 132	INPUT BUFFER
014E	1483	BUFEND	DS C	BUFFER ENDS
014E	1485	DS	4	EXTRA BYTES FOR STACK
014E	1487	STKLMT	DS 0	SOFT LIMIT FOR STACK
014E	1489	ORG	TOPSCR	
2000	1491	STACK	DS 0	STACK STARTS HERE
2002	1493	ORG	BOTRAM	
	1495	TXTUNF	DS 2	
	1497	TEXT	DS 2	

```
*****
*
```

```
* *** INITIALIZE ***
*
```

F000 310002	1504	ORG	BGTRDM	
F003 CD93F7	1506	INIT	LXI SP,STACK	
F006 218000	1508	CALL	CRLF	
F009 3EC3	1510	LXI	H,KEYWRD	AT POWER ON KEYWRD IS
FG0B BE	1512	MVI	A,0C3H	PROBABLY NOT C3
FC0C CA26F0	1514	CMP	M	
F00F 77	1516	JZ	TELL	IT IS C3, CONTINUE
F010 210040	1518	MOV	M,A	NO, SET IT TO C3
F013 228100	1520	LXI	H,DFTLMT	AND SET DEFAULT VALUE
F016 3EF0	1522	SHLD	TXTLMT	IN "TXTLNT".
F018 32C800	1524	MVI	A,BOTROM,<	INITIALIZE RANPNT
F01E 210620	1526	STA	RANPNT+1	
F01E 220020	1528	PURGE	LXI H,TEXT+4	PURGE TEXT AREA
F021 26FF	1530	SHLD	TXTUNF	
F023 220220	1532	MVI	H,OFFH	
F026 112FF0	1534	SHLD	TEXT	
F029 CD65F6	1536	TELL	LXI C,MSG	TELL USER
F02C C353F0	1538	CALL	PRTSTG	*****
F02F 5494E5920	1540	JMP	RSTART	**** JMP USER-INIT ****
F034 4241534943	1542	MSG	DB 'TINY'	*****
	1543		DB 'BASIC'	*****

F039	2056332E30	1544	DB * V3.0*,@CR	
F03E	0D	1545		
F03F	4F4B	1546	OK	DB "OK",@CR
F041	0D	1548		
F042	574841543F	1549	WHAT	DB "WHAT?",@CR
F047	0D	1551		
F048	484F573F	1552	HOW	DB "HW?",@CR
F04C	0D	1554		
F04D	534F525259	1555	SORRY	DB "SORRY",@CR
F052	0D	1557		
*				
*****				
*				
* *** DIRECT COMMAND / TEXT COLLECTER ***				
*				
*				
* TBI PRINTS OUT "OK(CR)", AND 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 ASK 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				
* "TEXT". THE END OF TEXT IS MARKED BY 2 BYTES XX FF. FOLLOWING				
* THESE ARE 2 BYTES RESERVED FOR THE ARRAY ELEMENT a(0). THE CONTENT				
* OF LOCATION LABELED "TXTUNF" POINTS TO ONE AFTER a(0).				
*				
* 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.				
*				
F053	310002	1593	RSTART	LXI SP,STACK
F056	215DF0	1595		LXI H,ST1+1
F059	22B700	1597	SHLD	CURRNT
F05C	210000	1599	ST1	LXI H,0
F05F	22BD00	1601	SHLD	LOPVAR
F062	22B900	1603	SHLD	STKGOS
F065	113FF0	1605	LXI	D,OK
F068	CD65F6	1607	CALL	PRTSTG
F06B	3E3E	1609	ST2	MVI A,>
F06D	CD9BF7	1611	CALL	GETLN
F070	D5	1613	PUSH	D
F071	11CA00	1615	LXI	D,BUFFER
F074	CDCFF5	1617	CALL	TSTNUM
F077	CD22F5	1619	CALL	IGNBLK
F07A	7C	1621	MOV	A,H
F07B	B5	1623	ORA	L
F07C	C1	1625	POP	B
F07D	CACCF0	1627	JZ	DIRECT
F080	18	1629	DCX	D
F081	7C	1631	MOV	A,H
F082	12	1633	STAX	D
F083	18	1635	DCX	D
F084	7D	1637	MOV	A,L
F085	12	1639	STAX	D
F086	C5	1641	PUSH	B
F087	D5	1643	PUSH	D
F088	79	1645	MOV	A,C
F089	93	1647	SUB	E
F08A	F5	1649	PUSH	PSW
F08B	CD64F5	1651	CALL	FNDLN
F08E	D5	1653	PUSH	D
F08F	C2A2F0	1655	JNZ	ST3
F092	D5	1657	PUSH	D
F093	CD7DF5	1659	CALL	FNDNXT
F096	C1	1661	POP	B
F097	2A0020	1663	LHLD	TXTUNF
				A=# OF BYTES IN LINE
				FIND THIS LINE IN SAVE
				AREA, DE->SAVE AREA
				NZ:NOT FOUND, INSERT
				Z:FOUND, DELETE IT
				SET DE->NEXT LINE
				BC->LINE TO BE DELETED
				HL->UNFILLED SAVE AREA

F09A	CD00F6	1665	CALL	MVUP	MOVE UP TO DELETE TXTUNF->UNFILLED AREA
F09D	60	1667	MOV	H,B	
F09E	69	1669	MOV	L,C	
F09F	220020	1671	SHLD	TXTUNF	UPDATE
F0A2	C1	1673	ST3	POP	GET READY TO INSERT
F0A3	2A0020	1675	LHLD	TXTUNF	BUT FIRST CHECK IF
F0A6	F1	1677	POP	PSW	THE LENGTH OF NEW LINE
F0A7	E5	1679	PUSH	H	IS 3 (LINE # AND CR)
F0A8	FE03	1681	CPI	3	THEN DO NOT INSERT
F0AA	CA53F0	1683	JZ	RSTART	MUST CLEAR THE STACK
F0AD	85	1685	ADD	L	COMPUTE NEW TXTUNF
F0AE	5F	1687	MOV	E,A	
F0AF	3E00	1689	MVI	A,0	
F0B1	8C	1691	ADC	H	
F0B2	57	1693	MOV	D,A	DE->NEW UNFILLED AREA
F0B3	2A8100	1695	LHLD	TXTLMT	CHECK TO SEE IF THERE
F0B6	EB	1697	XCHG	,	
F0B7	CDEF4	1699	CALL	COMP	IS ENOUGH SPACE
F0BA	D25DF5	1701	JNC	QSORRY	SORRY, NO ROOM FOR IT
F0BD	220020	1703	SHLD	TXTUNF	OK, UPDATE TXTUNF
F0C0	D1	1705	POP	D	DE->OLD UNFILLED AREA
F0C1	CDGBF6	1707	CALL	MVDOWN	
F0C4	D1	1709	POP	D	DE->BEGIN. HL->END
F0C5	E1	1711	POP	H	
F0C6	CD00F6	1713	CALL	MVUP	MOVE NEW LINE TO SAVE
F0C9	C36BF0	1715	JMP	ST2	AREA

\*\*\*\*\*

\*

\* \*\*\* 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 POINT TO THE  
\* TABLE-1. AT 'DIRECT', DE SHOULD POINT TO THE STRING, HL WILL BE SET  
\* UP TO POINT TO TAB1-1, WHICH IS THE TABLE OF ALL DIRECT AND  
\* STATEMENT COMMANDS.

\*

\* A '..' IN THE STRING WILL TERMINATE THE TEST AND THE PARTIAL 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.

\*

F0CC 2102F7 1747 DIRECT LXI H,TAB1-1 \*\*\* DIRECT \*\*\*

\*

F0CF CD22F5 1750 EXEC CALL IGNBLK \*\*\* EXEC \*\*\*

F0D2 D5 1752 PUSH D SAVE POINTER

F0D3 1A 1754 EX1 LDAX D IF FOUND '..' IN STRING

F0D4 13 1756 INX D BEFORE ANY MISMATCH

F0D5 FE2E 1758 CPI .. WE DECLARE A MATCH

F0D7 CAF0F0 1760 JZ EX3

F0DA 23 1762 INX H HL->TABLE

F0DB BE 1764 CMP M IF MATCH, TEST NEXT

F0DC CAD3F0 1766 JZ EX1

F0DF JE7F 1768 MVI A,07FH ELSE, SEE IF BIT 7

F0E1 1B 1770 DCX D OF TABLE IS SET, WHICH

F0E2 BE 1772 CMP M IS THE JUMP ADDR. (HI)

F0E3 DAF7F0 1774 JC EX5 C: YES, MATCHED

F0E6 23 1776 EX2 INX H NC: NO, FIND JUMP ADDR.

F0E7 BE 1778 CMP M

F0E8 D2E6F0 1780 JNC EX2

F0EB 23 1782 INX H BUMP TO NEXT TAB. ITEM

F0EC D1 1784 POP D RESTORE STRING POINTER

F0ED C3CFF0 1786 JMP EXEC TEST AGAINST NEXT ITEM

F0F0 3E7F 1788 EX3 MVI A,07FH PARTIAL MATCH, FIND

F0F2 23 1790 EX4 INX H JUMP ADDR., WHICH IS

F0F3 BE	1792	CMP M	FLAGGED BY BIT 7
F0F4 D2F2F0	1794	JNC EX4	
F0F7 7E	1796 EX5	MOV A,M	LOAD HL WITH THE JUMP
F0F8 23	1798	INX H	ADDRESS FROM THE TABLE
F0F9 6E	1800	MOV L,M	*****
F0FA E6FF	1802	ANI OFFH	***** ANI 07FH *****
F0FC 67	1804	MOV H,A	*****
F0FD F1	1806	POP PSW	CLEAN UP THE GARBAGE
F0FE E9	1808	PCHL ,	AND WE GO DO IT
*			*****
*			*****
* 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 'CURRENT' -> 0, GO TO 'RSTART', ELSE GO EXECUTE			
* NEXT COMMAND. (THIS IS DONE IN 'FINISH').			
*			*****
*			*****
* *** NEW *** STOP *** RUN (& FRIENDS) *** & GOTO ***			
*			
* 'NEW(CR)' RESETS 'TXTUNF'.			
*			
* 'STOP(CR)' GOES BACK TO 'RSTART'.			
*			
* 'RUN(CR)' FINDS THE FIRST STORED LINE, STORE ITS ADDRESS (IN			
* 'CURRENT'), 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.			
*			
F0FF CD2AF5	1847 NEW	CALL ENDCHK	*** NEW(CR) ***
F102 C31BF0	1849	JMP PURGE	
*			
F105 CD2AF5	1852 STOP	CALL ENDCHK	*** STOP(CR) ***
F108 C353F0	1854	JMP RSTART	
*			
F10B CD2AF5	1857 RUN	CALL ENDCHK	*** RUN (CR) ***
F10E 110220	1859	LXI D,TEXT	FIRST SAVED LINE
*			
F111 210000	1862 RUNNXL	LXI H,0	*** RUNNXL ***
F114 CD6CF5	1864	CALL FNLDL	FIND WHATEVER LINE #
F117 DAS3F0	1866	JC RSTART	C:PASSED TXTUNF, QUIT
*			
F11A EB	1869 RUNTSL	XCHG •	*** RUNTSL ***
F11B 228700	1871	SHLD CURRENT	SET 'CURRENT'->LINE #
F11E EB	1873	XCHG •	
F11F 13	1875	INX D	BUMP PASS LINE #
F120 13	1877	INX D	
*			
F121 CD98F7	1880 RUNSML	CALL CHKIO	*** RUNSML ***
F124 2112F7	1882	LXI H,TAB2-1	FIND COMMAND IN TAB2
F127 C3CCFO	1884	JMP EXEC	AND EXECUTE IT
*			
F12A CD5BF3	1887 GOTO	CALL EXPR	*** GOTO EXPR ***
F12D D5	1889	PUSH D	SAVE FOR ERROR ROUTINE
F12E CD2AF5	1891	CALL ENDCHK	MUST FIND A CR
F131 CD64F5	1893	CALL FNDLN	FIND THE TARGET LINE
F134 C2FAF5	1895	JNZ AHOM	NO SUCH LINE #
F137 F1	1897	POP PSW	CLEAR THE "PUSH DE"
F138 C31AF1	1899	JMP RUNTSL	GO DO IT

```

*****
*** LIST *** & PRINT ***
*
* LIST HAS THREE FORMS:
* "LIST(CR)" LISTS ALL SAVED LINES
* "LIST N(CR)" START LIST AT LINE N
* "LIST N1, N2(CR)" START LIST AT LINE N1 FOR N2 LINES YOU CAN STOP
* THE LISTING BY CONTROL C KEY
*
* PRINT COMMAND IS "PRINT ....;" OR "PRINT ....(CR)"
* WHERE "...." IS A LIST OF EXPRESSIONS, FORMATS, AND/OR STRINGS.
* THESE ITEMS ARE SEPERATED BY COMMAS.
*
* A FORMAT IS A POUND 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, 8 POSITIONS WILL BE
* USED.
*
* A STRING IS QUOTED IN A PAIR OF SINGLE QUOTES OR A PAIR OF DOUBLE
* QUOTES.
*
* CONTROL CHARACTERS AND LOWER CASE LETTERS CAN BE INCLUDED INSIDE THE
* QUOTES. ANOTHER (BETTER) WAY OF GENERATING CONTROL CHARACTERS ON
* THE OUTPUT IS USE THE UP-ARROW CHARACTER FOLLOWED BY A LETTER. |L
* MEANS FF, |I MEANS HT, |G MEANS BELL ETC.
*
* 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.
*
F13B CDCFF5      1935 LIST     CALL TSTNUM      TEST IF THERE IS A #
F13E E5          1937          PUSH H
F13F 21FFFF      1939          LXI H,0FFFFH
F142 CDBBF5      1941          TSTC "",LS1
F145 2C          1943
F146 03          1944
F147 CDCFF5      1945          CALL TSTNUM
F14A E3          1947 LS1       XTHL
F14B CD2AF5      1949          CALL ENDCHK
F14E CD64F5      1951          CALL FNDLN      IF NO # WE GET A 0
F151 DA53F0      1953 LS2       JC RSTART     FIND THIS OR NEXT LINE
F154 E3          1955          XTHL
F155 7C          1957          MOV A,H
F156 B5          1959          ORA L
F157 CA53F0      1961          JZ RSTART
F15A 2B          1963          DCX H
F15B E3          1965          XTHL
F15C CDF2F6      1967          CALL PRTLN      PRINT THE LINE
F15F CD65F6      1969          CALL PRTSTG
F162 CD98F7      1971          CALL CHKIO
F165 CD6CF5      1973          CALL FNDLP      FIND NEXT LINE
F168 C351F1      1975          JMP LS2       AND LCOP BACK
*
F16B 0E08          1978 PRINT    MVI C,8
F16D CDBBF5      1980          TSTC ";" .PR1   IF NULL LIST & ";"
F170 3B          1982
F171 06          1983
F172 CD93F7      1984          CALL CRLF      GIVE CR-LF AND
F175 C321F1      1986          JMP RUNSML    CONTINUE SAME LINE
F178 CDBBF5      1988 PR1       TSTC @CR,PR6  IF NULL LIST (CR)
F17B 0D          1990
F17C 24          1991
F17D CD93F7      1992          CALL CRLF      ALSO GIVE CR-LF AND
F180 C311F1      1994          JMP RUNNXL    GO TO NEXT LINE
F183 CDBBF5      1996 PR2       TSTC "#",PR4  ELSE IS IT FORMAT?
F186 23          1998
F187 0E          1999
F188 CD5BF3      2000 PR3       CALL EXPR      YES, EVALUATE EXPR.
F18B 3EC0          2002          MVI A,OCOH
F18D A5          2004          ANA L
F18E B4          2006          ORA H
F18F C2F9F5      2008          JNZ QHOW

```

F192	4D	2010	MOV	C,L	AND SAVE IT IN C
F193	C39CF1	2012	JMP	PR5	LOOK FOR MORE TO PRINT
F196	CD74F6	2014	PR4	CALL QTSTG	OR IS IT A STRING?
F199	C3BAF1	2016	JMP	PR9	IF NOT, MUST BE EXPR.
F19C	CDDBBF5	2018	PR5	TSTC "",".PR8	IF ",", GO FIND NEXT
F19F	2C	2020			
F1A0	13	2021			
F1A1	CDDBBF5	2022	PR6	TSTC "",".PR7	
F1A4	2C	2024			
F1A5	08	2025			
F1A6	3E20	2026	MVI	A, " "	
F1A8	CD95F7	2028	CALL	OUTCH	
F1AB	C3A1F1	2030	JMP	PR6	
F1AE	CD0FF5	2032	PR7	CALL FIN	IN THE LIST.
F1B1	C383F1	2034	JMP	PR2	LIST CONTINUES
F1B4	CD93F7	2036	PR8	CALL CRLF	LIST ENDS
F1B7	C309F5	2038	JMP	FINISH	
F1B8	CD5BF3	2040	PR9	CALL EXPR	EVALUATE THE EXPR
F1BD	C5	2042	PUSH	B	
F1BE	CDAEF6	2044	CALL	PRTNUM	PRINT THE VALUE
F1C1	C1	2046	POP	B	
F1C2	C39CF1	2048	JMP	PR5	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 SAVED SO  
 \* THAT EXECUTION CAN BE CONTINUED AFTER THE SUBROUTINE 'RETURN'. IN  
 \* ORDER THAT \*GOSUB\* CAN BE NESTED (AND EVEN RECURSIVE), THE SAVE AREA  
 \* MUST BE STACKED. THE STACK POINTER IS SAVED IN \*STKGOS\*. THE OLD  
 \* \*STKGOS\* IS SAVED IN THE STACK. IF WE ARE IN THE MAIN ROUTINE,  
 \* \*STKGOS\* IS ZERO (THIS WAS DONE BY THE "MAIN" SECTION OF THE CODE),  
 \* BUT WE STILL SAVE IT AS A FLAG FOR NO FURTHER 'RETURN'S.

\* \*RETURN(CR)\* UNDOES EVERYTHING THAT \*GOSUB\* DID, AND THUS RETURN THE  
 \* EXECUTION TO THE COMMAND AFTER THE MOST RECENT \*GOSUB\*. IF \*STKGOS\*  
 \* IS ZERO, IT INDICATES THAT WE NEVER HAD A \*GOSUB\* AND IS THUS AN  
 \* ERROR.

F1C5	CD36F6	2070	GOSUB	CALL	PUSHA	SAVE THE CURRENT "FOR"
F1C8	CD5BF3	2072		CALL	EXPR	PARAMETERS
F1CB	D5	2074		PUSH	D	AND TEXT POINTER
F1CC	CD64F5	2076		CALL	FNDLN	FIND THE TARGET LINE
F1CF	C2FAF5	2078		JNZ	AHOW	NOT THERE. SAY "HOW?"
F1D2	2AB700	2080		LHLD	CURRNT	SAVE CLD
F1D5	E5	2082		PUSH	H	"CURRNT" OLD *STKGOS*
F1D6	2AB900	2084		LHLD	STKGOS	
F1D9	E5	2086		PUSH	H	AND LOAD NEW ONES
F1DA	210000	2088		LXI	H,0	
F1DD	22BD00	2090		SHLD	LOPVAR	
F1E0	39	2092		DAD	SP	
F1E1	22B900	2094		SHLD	STKGOS	
F1E4	C31AF1	2096		JMP	RUNTSL	THEN RUN THAT LINE
F1E7	CD2AF5	2098	RETURN	CALL	ENDCHK	THERE MUST BE A CR
F1EA	2AB900	2100		LHLD	STKGOS	OLD STACK POINTER
F1ED	7C	2102		MOV	A,H	0 MEANS NOT EXIST
F1EE	B5	2104		ORA	L	
F1EF	CA30F5	2106		JZ	CWHAT	SO, WE SAY: "WHAT?"
F1F2	F9	2108		SPHL	,	ELSE, RESTORE IT
F1F3	E1	2110	RESTOR	POP	H	
F1F4	22B900	2112		SHLD	STKGOS	AND THE OLD *STKGOS*
F1F7	E1	2114		POP	H	
F1F8	22B700	2116		SHLD	CURRNT	AND THE OLD *CURRNT*
F1FB	D1	2118		POP	D	OLD TEXT POINTER
F1FC	CD1AF6	2120		CALL	POPA	OLD "FOR" PARAMETERS
F1FF	C309F5	2122		JMP	FINISH	

\*\*\*\*\*

\*\*\* FOR \*\*\* & NEXT \*\*\*

\* \*FOR\* HAS TWO FORMS: \*FOR VAR=EXP1 TO EXP2 STEP EXP3\* AND \*FOR  
 \* VAR=EXP1 TO EXP2\* THE SECOND FORM MEANS THE SAME THING AS THE FIRST

\* FORM WITH EXP3=1. (I.E., WITH A STEP OF +1.) TBI WILL FIND THE  
 \* VARIABLE VAR. AND SET ITS VALUE TO THE CURRENT VALUE OF EXP1. IT  
 \* ALSO EVALUATES EXP2 AND EXP3 AND SAVE ALL THESE TOGETHER WITH THE  
 \* TEXT POINTER ETC. IN THE 'FOR' SAVE AREA, WHICH CONSISTS OF  
 \* 'LOPVAR', 'LOPINC', 'LOPLMT', 'LOPLN', AND 'LOPPT'. IF THERE IS  
 \* ALREADY SOME- THING IN THE SAVE AREA (THIS IS INDICATED BY A  
 \* NON-ZERO 'LOPVAR'), THEN THE OLD SAVE AREA IS SAVED IN THE STACK  
 \* BEFORE THE NEW ONE OVERWRITES IT. TBI WILL THEN DIG IN THE STACK  
 \* AND FIND OUT IF THIS SAME VARIABLE WAS USED IN ANOTHER CURRENTLY  
 \* ACTIVE 'FOR' LOOP. IF THAT IS THE CASE, THEN THE OLD 'FOR' LOOP IS  
 \* DEACTIVATED. (PURGED FROM THE STACK..)

\* 'NEXT VAR' SERVES AS THE LOGICAL (NOT NECESSARILY PHYSICAL) END OF  
 \* THE 'FOR' LOOP. THE CONTROL VARIABLE VAR. IS CHECKED WITH THE  
 \* 'LOPVAR'. IF THEY ARE NOT THE SAME, TBI DIGS IN THE STACK TO FIND  
 \* THE RIGHT ONE AND PURGES ALL THOSE THAT DID NOT MATCH. EITHER WAY,  
 \* TBI THEN ADDS THE 'STEP' TO THAT VARIABLE AND CHECK THE RESULT WITH  
 \* THE LIMIT. IF IT IS WITHIN THE LIMIT, CONTROL LOOPS BACK TO THE  
 \* COMMAND FOLLOWING THE 'FOR'. IF OUTSIDE THE LIMIT, THE SAVE ARER IS  
 \* PURGED AND EXECUTION CONTINUES.

F202	CD36F6	2156	FUR	CALL	PUSHA	SAVE THE OLD SAVE AREA
F205	CDF3F4	2158		CALL	SETVAL	SET THE CONTROL VAR.
F208	28	2160		DCX	H	HL IS ITS ADDRESS
F209	228D00	2162		SHLD	LOPVAR	SAVE THAT
F20C	216EF7	2164		LXI	H,TAB4-1	USE 'EXEC' TO LOOK
F20F	C3CF0	2166		JMP	EXEC	FOR THE WORD 'TO'
F212	CD58F3	2168	FR1	CALL	EXPR	EVALUATE THE LIMIT
F215	22C100	2170		SHLD	LOPLMT	SAVE THAT
F218	2174F7	2172		LXI	H,TAB5-1	USE 'EXEC' TO LOOK
F21B	C3CF0	2174		JMP	EXEC	FOR THE WORD 'STEP'
F21E	CD58F3	2176	FR2	CALL	EXPR	FOUND IT, GET STEP
F221	C327F2	2178		JMP	FR4	
F224	210100	2180	FR3	LXI	H,1	NOT FOUND, SET TO 1
F227	22BF00	2182	FR4	SHLD	LOPINC	SAVE THAT TOO
F22A	2AB700	2184		LHLD	CURRENT	SAVE CURRENT LINE #
F22D	22C300	2186		SHLD	LOPLN	
F230	EB	2188		XCHG	,	AND TEXT POINTER
F231	22C500	2190		SHLD	LOPPT	
F234	010A00	2192		LXI	B,10	DIG INTO STACK TO
F237	2ABD00	2194		LHLD	LOPVAR	FIND 'LOPVAR'
F23A	EB	2196		XCHG	,	
F23B	60	2198		MOV	H,B	
F23C	68	2200		MOV	L,B	HL=0 NOW
F23D	39	2202		DAD	SP	HERE IS THE STACK
F23E	C342F2	2204		JMP	FR6	
F241	09	2206	FR5	DAD	B	EACH LEVEL IS 10 DEEP
F242	7E	2208	FR6	MOV	A,M	GET THAT OLD 'LOPVAR'
F243	23	2210		INX	H	
F244	B6	2212		ORA	M	
F245	CA62F2	2214		JZ	FR7	O SAYS NC MORE IN IT
F248	7E	2216		MOV	A,M	
F249	28	2218		DCX	H	
F24A	BA	2220		CMP	D	SAME AS THIS ONE?
F248	C241F2	2222		JNZ	FR5	THE OTHER HALF?
F24E	7E	2224		MOV	A,M	
F24F	BB	2226		CMP	E	
F250	C241F2	2228		JNZ	FR5	
F253	EB	2230		XCHG	,	YES, FOUND ONE
F254	210000	2232		LXI	H,0	
F257	39	2234		DAD	SP	TRY TO MOVE SP
F258	44	2236		MOV	B,H	
F259	4D	2238		MOV	C,L	
F25A	210A00	2240		LXI	H,10	
F25D	19	2242		DAD	D	
F25E	C00BF6	2244		CALL	MVDOWN	AND PURGE 10 WORDS
F261	F9	2246		SPHL	,	IN THE STACK
F262	2AC500	2248	FR7	LHLD	LOPPT	JOB DONE, RESTORE DE
F265	EB	2250		XCHG	,	
F266	C309F5	2252		JMP	FINISH	AND CONTINUE
*						
F269	CD89F5	2255	NEXT	CALL	TSTV	GET ADDRESS OF VAR.
F26C	DA30F5	2257		JC	QWHAT	NO VARIABLE, "WHAT?"
F26F	22BB00	2259		SHLD	VARNXT	YES, SAVE IT
F272	D5	2261	NX1	PUSH	D	SAVE TEXT POINTER

F273 EB	2263	XCHG .	
F274 2ABD00	2265	LHLD LOPVAR	GET VAR. IN 'FOR'
F277 7C	2267	MOV A,H	O SAYS NEVER HAD ONE
F278 B5	2269	ORA L	SO WE ASK: "WHAT?"
F279 CA31F5	2271	JZ AWHAT	ELSE WE CHECK THEM
F27C CDEDF4	2273	CALL COMP	OK, THEY AGREE
F27F CA8CF2	2275	JZ NX2	NO, LET'S SEE
F282 D1	2277	POP D	PURGE CURRENT LOOP
F283 CD1AF6	2279	CALL POPA	AND PCP CNE LEVEL
F286 2ABB00	2281	LHLD VARNXT	GO CHECK AGAIN
F289 C372F2	2283	JMP NX1	COME HERE WHEN AGREED
F28C SE	2285 NX2	MOV E,M	DE=VALUE OF VAR.
F28D 23	2287	INX H	
F28E 56	2289	MOV D,M	
F28F 2ABF00	2291	LHLD LOPINC	
F292 E5	2293	PUSH H	
F293 7C	2295	MOV A,H	S=SIGN DIFFER
F294 AA	2297	XRA D	A=SIGN OF DE
F295 7A	2299	MOV A,D	ADD ONE STEP
F296 19	2301	DAD D	CANNOT OVERFLOW
F297 FA9EF2	2303	JM NX3	MAY OVERFLOW
F29A AC	2305	XRA H	AND IT DID
F29B FAC2F2	2307	JM NX5	
F29E EB	2309 NX3	XCHG .	
F29F 2ABD00	2311	LHLD LOPVAR	PUT IT BACK
F2A2 73	2313	MOV M,E	
F2A3 23	2315	INX H	
F2A4 72	2317	MOV M,D	
F2A5 2AC100	2319	LHLD LOPLMT	HL=LIMIT
F2A6 F1	2321	POP PSW	OLD HL
F2A9 B7	2323	ORA A	STEP > 0
F2AA F2AEF2	2325	JP NX4	STEP < 0
F2AD EB	2327	XCHG .	COMPARE WITH LIMIT
F2AE CDE3F4	2329 NX4	CALL CKHLDE	RESTORE TEXT POINTER
F2B1 D1	2331	POP D	OUTSIDE LIMIT
F2B2 DAC4F2	2333	JC NX6	WITHIN LIMIT, GO
F2B5 2AC300	2335	LHLD LOPLN	BACK TO THE SAVED
F2B8 22B700	2337	SHLD CURRNT	"CURRNT" AND TEXT
F2B6 2AC500	2339	LHLD LOPPT	POINTER
F2BE EB	2341	XCHG .	
F2BF C309F5	2343	JMP FINISH	
F2C2 E1	2345 NX5	POP H	OVERFLOW, PURGE
F2C3 D1	2347	POP D	GARBAGE IN STACK
F2C4 CD1AF6	2349 NX6	CALL POPA	PURGE THIS LOOP
F2C7 C309F5	2351	JMP FINISH	

\*

---

\* \*\*\* 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 AN UP-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 PROCEEDED 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 VARIABLE TO THAT VALUE. TBI WILL ALSO HANDLE  
 \* "LET" COMMAND WITHOUT THE WORD "LET". THIS IS DONE BY "DEFLT".

F2CA	210000	2390	REM	LXI	H,0	*** REM ***
F2CD	C3D3F2	2392		JMP	IF1	THIS IS LIKE "IF 0"
*						
F2D0	CD5BF3	2395	IFF	CALL	EXPR	*** IF ***
F2D3	7C	2397	IF1	MOV	A,H	IS THE EXPR.=?
F2D4	85	2399		ORA	L	
F2D5	C221F1	2401		JNZ	RUNSM	NO, CONTINUE
F2D8	CD7FF5	2403		CALL	FNDSP	YES, SKIP REST OF LINE
F2D9	D21AF1	2405		JNC	RUNTS	AND RUN THE NEXT LINE
F2DE	C353F0	2407		JMP	RSTART	IF NC NEXT, RE-START
*						
F2E1	2AB800	2410	INPERR	LHLD	STKINP	*** INPERR ***
F2E4	F9	2412		SHLD	,	RESTORE GLD SP
F2E5	E1	2414		POP	H	AND OLD "CURRNT"
F2E6	22B700	2416		SHLD	CURRNT	
F2E9	D1	2418		POP	D	AND OLD TEXT POINTER
F2EA	D1	2420		POP	D	REDO INPUT
*						
F2EB	D5	2423	INPUT	DS	0	
F2EC	CD74F6	2425	IP1	PUSH	D	SAVE IN CASE OF ERROR
F2EF	C31AF3	2427		CALL	QTSTG	IS NEXT ITEM A STRING?
F2F2	CD89F5	2429		JMP	IP8	NO
F2F5	DA0EF3	2431	IP2	CALL	TSTV	YES, BUT FOLLOWED BY A
F2F8	CD2CF3	2433		JC	IP5	VARIABLE? NO.
F2FB	11CA00	2435	IP3	CALL	IP12	
F2FE	CD5BF3	2437		LXI	D,BUFFER	POINTS TO BUFFER
F301	CD2AF5	2439		CALL	EXPR	EVALUATE INPUT
F304	D1	2441		CALL	ENDCHK	
F305	E8	2443		POP	D	OK, GET GLD HL
F306	73	2445		XCHG	,	
F307	23	2447		MOV	M,E	SAVE VALUE IN VAR.
F308	72	2449		MOV	M,D	
F309	E1	2451	IP4	POP	H	GET GLD "CURRNT"
F30A	22B700	2453		SHLD	CURRNT	
F30D	D1	2455		POP	D	AND GLD TEXT POINTER
F30E	F1	2457		POP	PSW	PURGE JUNK IN STACK
F30F	CDBBF5	2459	IP5	POP	"",IP7	IS NEXT CH. "",?
F312	2C	2461	IP6	TSTC	"",IP7	
F313	03	2463				
F314	C3EBF2	2464				
F317	C309F5	2465		JMP	INPUT	YES, MORE ITEMS.
F31A	D5	2467	IP7	JMP	FINISH	
F31E	CD89F5	2469	IP8	PUSH	D	SAVE FOR "PRTSTG"
F31E	D224F3	2471		CALL	TSTV	MUST BE VARIABLE NOW
F321	C330F5	2473		JNC	IP11	"WHAT?" IT IS NOT?
F324	43	2475	IP10	JMP	QWHAT	
F325	D1	2477	IP11	MOV	B,E	
F326	CDA3F6	2479		POP	D	
F329	C3F8F2	2481		CALL	PRTCHS	PRINT THOSE AS PROMPT
F32C	C1	2483		JMP	IP3	YES, INPUT VARIABLE
F32D	DS	2485	IP12	POP	B	RETURN ADDRESS
F32E	EB	2487		PUSH	D	SAVE TEXT POINTER
F32F	2AB700	2489		XCHG	,	ALSO SAVE "CURRNT"
F332	E5	2491		LHLD	CURRNT	
F333	21EBF2	2493		PUSH	H	
F336	22B700	2495		LXI	H,IP1	A NEGATIVE NUMBER
F339	210000	2497		SHLD	CURRNT	AS A FLAG
F33C	39	2499		LXI	H,0	SAVE SP TOO
F33D	22B800	2501		DAD	SP	
F340	D5	2503		SHLD	STKINP	
F341	3E20	2505		PUSH	D	OLD HL
F343	C5	2507		MVI	A, " "	PRINT A SPACE
F344	C39BF7	2509		PUSH	B	
*		2511		JMP	GETLN	AND GET A LINE
F347	1A	2514	DEFLT	LDAX	D	*** DEFLT ***
F348	FE0D	2516		CPI	@CR	EMPTY LINE IS OK
F34A	CA5BF3	2518		JZ	LT4	ELSE IT IS "LET"
*						
F34D	CDF3F4	2521	LET	DS	0	*** LET ***
		2523	LT2	CALL	SETVAL	

F350 CDBBF5	2525 LT3	TSTC '+',LT4	SET VALUE TO VAR.
F353 2C	2527		
F354 03	2528		
F355 C34DF3	2529	JMP LET	ITEM BY ITEM
F358 C309F5	2531 LT4	JMP FINISH	UNTIL FINISH
*			
*****			
*			
*	*** EXPR ***		
*			
*	'EXPR' EVALUATES ARITHMETICAL OR LOGICAL EXPRESSIONS.		
*	<EXPR> ::= <EXPR1>		
*	<EXPR1><REL.OP.><EXPR1>		
*	WHERE <REL.OP.> IS ONE OF THE OPERATORS IN TAB6 AND THE RESULT OF		
*	THESE OPERATIONS IS 1 IF TRUE AND 0 IF FALSE.		
*	<EXPR1> ::= (+ OR -)<EXPR2>(+ OR -<EXPR2>)(....)		
*	WHERE () ARE OPTIONAL AND (....) ARE OPTIONAL REPEATS.		
*	<EXPR2> ::= <EXPR3>(<*> OR /><EXPR3>)(....)		
*	<EXPR3> ::= <VARIABLE>		
*	<FUNCTION>		
*	(<EXPR>)		
*	<EXPR> IS RECURSIVE SO THAT VARIABLE 'a' CAN HAVE AN <EXPR> AS		
*	INDEX, FUNCTIONS CAN HAVE AN <EXPR> AS ARGUMENTS. AND		
*	<EXPR3> CAN BE AN <EXPR> IN PARANTHESE.		
*			
F35B CDA3F3	2553 EXPR	CALL XPR1	*** EXPR ***
F35E E5	2555	PUSH H	SAVE <EXPR1> VALUE
F35F 217CF7	2557	LXI H,TAB6-1	LOOKUP REL.OP.
F362 C3CFF0	2559	JMP EXEC	GO DO IT
F365 CD8EF3	2561 XPR1	CALL XPR8	REL.OP.">=
F368 D8	2563	RC .	NO, RETURN HL=0
F369 6F	2565	MOV L,A	YES, RETURN HL=1
F36A C9	2567	RET .	
F36B CD8EF3	2569 XPR2	CALL XPR8	REL.OP."#"
F36E C8	2571	RZ .	FALSE, RETURN HL=0
F36F 6F	2573	MOV L,A	TRUE, RETURN HL=1
F370 C9	2575	RET .	
F371 CD8EF3	2577 XPR3	CALL XPR8	REL.OP.">"
F374 C8	2579	RZ .	FALSE
F375 D8	2581	RC .	ALSO FALSE, HL=0
F376 6F	2583	MOV L,A	TRUE, HL=1
F377 C9	2585	RET .	
F378 CD8EF3	2587 XPR4	CALL XPR8	REL.OP."<="
F378 6F	2589	MOV L,A	SET HL=1
F37C C8	2591	RZ .	REL. TRUE, RETURN
F37D D8	2593	RC .	
F37E 6C	2595	MOV L,H	ELSE SET HL=0
F37F C9	2597	RET .	
F380 CD8EF3	2599 XPR5	CALL XPR8	REL.OP."="
F383 C0	2601	RNZ .	FALSE, RETRUN HL=0
F384 6F	2603	MOV L,A	ELSE SET HL=1
F385 C9	2605	RET .	
F386 CD8EF3	2607 XPR6	CALL XPR8	REL.OP."<"
F389 D0	2609	RNC .	FALSE, RETURN HL=0
F38A 6F	2611	MOV L,A	ELSE SET HL=1
F38B C9	2613	RET .	
F38C E1	2615 XPR7	POP H	NOT REL.OP.
F38D C9	2617	RET .	RETURN HL=<EXPR1>
F38E 79	2619 XPR8	MOV A,C	SUBROUTINE FOR ALL
F38F E1	2621	POP H	REL.OP.'S
F390 C1	2623	POP B	
F391 E5	2625	PUSH H	REVERSE TOP OF STACK
F392 C5	2627	PUSH B	
F393 4F	2629	MOV C,A	
F394 CDA3F3	2631	CALL EXPR1	GET 2ND <EXPR1>
F397 EB	2633	XCHG .	VALUE IN DE NOW
F398 E3	2635	XTHL	1ST <EXPR1> IN HL
F399 CDE3F4	2637	CALL CKHLDE	COMPARE 1ST WITH 2ND
F39C D1	2639	POP D	RESTORE TEXT POINTER
F39D 210000	2641	LXI H,0	SET HL=0, A=1
F3A0 3E01	2643	MVI A,1	
F3A2 C9	2645	RET .	
*			
F3A3 CDBBF5	2648 EXPR1	TSTC '-',XP11	NEGATIVE SIGN?
F3A6 2D	2650		

F3A7	06	2651				
F3A8	210000	2652	LXI	H,O	YES. FAKE "0-"	
F3AB	C3D5F3	2654	JMP	XP16	TREAT LIKE SUBTRACT	
F3AE	CD8BF5	2656	XP11	TSTC "+",XP12	POSITIVE SIGN? IGNORE	
F3B1	2B	2658				
F3B2	00	2659				
F3B3	CDDFF3	2660	XP12	CALL EXPR2	1ST <EXPR2>	
F3B6	CD8BF5	2662	XP13	TSTC "+",XP15	ADD?	
F3B9	2B	2664				
F3B8	15	2665				
F3BB	E5	2666	PUSH	H	YES. SAVE VALUE	
F3BC	CDDFF3	2668	CALL	EXPR2	GET 2ND <EXPR2>	
F3BF	EB	2670	XP14	XCHG ,	2ND IN DE	
F3C0	E3	2672	XTHL		1ST IN HL	
F3C1	7C	2674	MOV	A,H	COMPARE SIGN	
F3C2	AA	2676	XRA	D		
F3C3	7A	2678	MOV	A,D		
F3C4	19	2680	DAD	D		
F3C5	D1	2682	POP	D		
F3C6	FAB6F3	2684	JM	XP13	RESTORE TEXT POINTER	
F3C9	AC	2686	XRA	H	1ST 2ND SIGN DIFFER	
F3CA	F2B6F3	2688	JP	XP13	1ST 2ND SIGN EQUAL	
F3CD	C3F9F5	2690	JMP	QHOB	SO IS RESULT	
F3D0	CD8BF5	2692	XP15	TSTC "--",XPR9	ELSE WE HAVE OVERFLOW	
F3D3	2D	2694			SUBTRACT?	
F3D4	92	2695				
F3D5	E5	2696	XP16	PUSH H	YES. SAVE 1ST <EXPR2>	
F3D6	CDDFF3	2698	CALL	EXPR2	GET 2ND <EXPR2>	
F3D9	CDCEF4	2700	CALL	CHKSGN	NEGATE	
F3DC	C3BFFF3	2702	JMP	XP14	AND ADD THEM	
*						
F3DF	CD43F4	2705	EXPR2	CALL EXPR3	GET 1ST <EXPR3>	
F3E2	CD8BF5	2707	XP21	TSTC "**",XP24	MULTIPLY?	
F3E5	2A	2709				
F3E6	2D	2710				
F3E7	E5	2711	PUSH	H	YES. SAVE 1ST	
F3E8	CD43F4	2713	CALL	EXPR3	AND GET 2ND <EXPR3>	
F3EB	0600	2715	MVI	B,O	CLEAR B FOR SIGN	
F3ED	CDCBF4	2717	CALL	CHKSGN	CHECK SIGN	
F3F0	E3	2719	XTHL		1ST IN HL	
F3F1	CDCBF4	2721	CALL	CHKSGN	CHECK SIGN OF 1ST	
F3F4	EB	2723	XCHG	,		
F3F5	E3	2725	XTHL			
F3F6	7C	2727	MOV	A,H	IS HL > 255 ?	
F3F7	B7	2729	ORA	A		
F3F8	CA01F4	2731	JZ	XP22	NO	
F3FB	7A	2733	MOV	A,D	YES. HOW ABOUT DE	
F3FC	B2	2735	ORA	D		
F3FD	EB	2737	XCHG	,	PUT SMALLER IN HL	
F3FE	C2FAF5	2739	JNZ	AHOW	ALSO >, WILL OVERFLOW	
F401	7D	2741	XP22	MOV	THIS IS DUMB	
F402	210000	2743	LXI	H,O	CLEAR RESULT	
F405	87	2745	ORA	A	ADD AND COUNT	
F406	CA35F4	2747	JZ	XP25		
F409	19	2749	XP23	DAD	OVERFLOW	
F40A	DAFAF5	2751	JC	AHOW		
F40D	3D	2753	DCR	A		
F40E	C209F4	2755	JNZ	XP23		
F411	C335F4	2757	JMP	XP25	FINISHED	
F414	CD8BF5	2759	XP24	TSTC "/",XPR9	DIVIDE?	
F417	2F	2761				
F418	4E	2762				
F419	E5	2763	PUSH	H	YES. SAVE 1ST <EXPR3>	
F41A	CD43F4	2765	CALL	EXPR3	AND GET 2ND ONE	
F41D	0600	2767	MVI	B,O	CLEAR B FOR SIGN	
F41F	CDCBF4	2769	CALL	CHKSGN	CHECK SIGN OF 2ND	
F422	E3	2771	XTHL		GET 1ST IN HL	
F423	CDCBF4	2773	CALL	CHKSGN	CHECK SIGN OF 1ST	
F426	EB	2775	XCHG	,		
F427	E3	2777	XTHL			
F428	EB	2779	XCHG	,	DIVIDE BY 0?	
F429	7A	2781	MOV	A,D		
F42A	B3	2783	ORA	E	SAY "HOW?"	
F42B	CAFAF5	2785	JZ	AHOW	ELSE SAVE SIGN	
F42E	CS	2787	PUSH	B		

F42F	CDAEF4	2789	CALL	DIVIDE	USE SUBROUTINE
F432	60	2791	MOV	H,B	RESULT IN HL NOW
F433	69	2793	MOV	L,C	
F434	C1	2795	POP	B	GET SIGN BACK
F435	D1	2797	POP	D	AND TEXT POINTER
F436	7C	2799	MOV	A,H	HL MUST EE +
F437	B7	2801	ORA	A	ELSE IT IS OVERFLOW
F438	FAF9F5	2803	JM	QH0W	
F43B	78	2805	MOV	A,B	
F43C	B7	2807	ORA	A	
F43D	FCCEF4	2809	CM	CHGSGN	CHANGE SIGN IF NEEDED
F440	C3E2F3	2811	JMP	XP21	LOOK FOR MORE TERMS
*					
F443	2159F7	2814	EXPR3	LXI H,TAB3-1	FIND FUNCTION IN TAB3
F446	C3CFF0	2816	JMP	EXEC	AND GO DC IT
F449	CD89F5	2818	NOTF	CALL TSTV	NO, NOT A FUNCTION
F44C	DA54F4	2820	JC	XP32	NOR A VARIABLE
F44F	7E	2822	MOV	A,M	VARIABLE
F450	23	2824	INX	H	
F451	66	2826	MOV	H,M	VALUE IN HL
F452	6F	2828	MOV	L,A	
F453	C9	2830	RET	,	
F454	CDCFF5	2832	XP32	CALL TSTNUM	OR IS IT A NUMBER
F457	78	2834	MOV	A,B	# OF DIGIT
F458	87	2836	ORA	A	
F459	C0	2838	RNZ	,	OK
F45A	CUBBF5	2840	PARN	TSTC '(*,XPRO	NO DIGIT, MUST BE
F45D	28	2842			
F45E	09	2843			
F45F	CD5BF3	2844	PARNP	CALL EXPR	"(EXPR)"
F462	CD8BF5	2846		TSTC ')",XPRO	
F463	29	2848			
F466	01	2849			
F467	C9	2850	XPR9	RET ,	ELSE SAY: "WHAT?"
F468	C330F5	2852	XPRO	JMP QWHAT	
*					
F46B	CD5AF4	2855	RND	CALL PARN	*** RND(EXPR) ***
F46E	7C	2857	MOV	A,H	EXPR MUST BE +
F46F	B7	2859	ORA	A	
F470	FAF9F5	2861	JM	QH0W	
F473	B5	2863	ORA	L	AND NON-ZERO
F474	CAF9F5	2865	JZ	QH0W	
F477	D5	2867	PUSH	D	SAVE BOTH
F478	E5	2869	PUSH	H	
F479	2AC700	2871	LHLD	RANPNT	GET MEMORY AS RANDOM
F47C	1193F7	2873	LXI	D,RANEND	
F47F	CDED4	2875	CALL	COMP	
F482	DA88F4	2877	JC	RA1	WRAP AROUND IF LAST
F485	2100F0	2879	LXI	H,BOTROM	
F488	5E	2881	RANPNT	MOV	
F489	23	2883	E,M	INX	
F48A	56	2885	M	MOV	
F48B	22C700	2887	SHLD	RANPNT	
F48E	C1	2889	POP	H	
F48F	EB	2891	XCHG	,	
F490	C5	2893	PUSH	B	
F491	CDAEF4	2895	CALL	DIVIDE	RND(N)=MOD(M,N)+1
F494	C1	2897	POP	B	
F495	D1	2899	POP	D	
F496	23	2901	INX	H	
F497	C9	2903	RET	,	
*					
F498	CD5AF4	2906	ABS	CALL PARN	*** ABS(EXPR) ***
F49B	13	2908	DCX	D	CHECK SIGN
F49C	CDCBF4	2910	CALL	CHKSGN	
F49F	13	2912	INX	D	
F4A0	C9	2914	RET	,	
F4A1	2A0020	2916	SIZE	LHLD TXTUNF	*** SIZE ***
F4A4	D5	2918	PUSH	D	GET THE NUMBER OF FREE
F4A5	EB	2920	XCHG	,	BYTES BETWEEN 'TXTUNF'
F4A6	2A8100	2922	LHLD	TXTLMT	AND 'TXTLMT'
F4A9	CDC4F4	2924	CALL	SUBDE	
F4AC	D1	2926	POP	D	
F4AD	C9	2928	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.
*
F4AE E5          2948 DIVIDE  PUSH H      *** DIVIDE ***
F4AF 6C          2950 MOV   L,H      DIVIDE H BY DE
F4B0 2600        2952 MVI   H,O
F4B2 CDB9F4      2954 CALL  DV1
F4B5 41          2956 MOV   B,C
F4B6 7D          2958 MOV   A,L      SAVE RESULT IN B
F4B7 E1          2960 POP   H      (REMAINDER+L)/DE
F4B8 67          2962 MOV   H,A
F4B9 0EFF        2964 DV1   MVI   C,-1
F4B8 0C          2966 DV2   INR   C      RESULT IN C
F4BC CDC4F4      2968 CALL  SUBDE
F4BF D2B6F4      2970 JNC   DV2   INR   C
F4C2 19          2972 DAD   D      DUMB ROUTINE
F4C3 C9          2974 RET   *
F4C4 7D          2977 SUBDE  MOV   A,L      DIVIDE BY SUBTRACT
F4C5 93          2979 SUB   E      AND COUNT
F4C6 6F          2981 MOV   L,A
F4C7 7C          2983 MOV   A,H
F4C8 SA          2985 SBB   D
F4C9 67          2987 MOV   H,A
F4CA C9          2989 RET   *
F4CB 7C          2992 CHKSGN MOV   A,H      *** CHKSGN ***
F4CC B7          2994 ORA   A      CHECK SIGN OF HL
F4CD F0          2996 RP    *
F4CE 7C          2999 CHGSGN MOV   A,H      *** CHGSGN ***
F4CF B5          3001 ORA   L
F4D0 C8          3003 RZ    *
F4D1 7C          3005 MOV   A,H
F4D2 F5          3007 PUSH  PSW
F4D3 2F          3009 CMA   *
F4D4 67          3011 MOV   H,A      CHANGE SIGN OF HL
F4D5 7D          3013 MOV   A,L
F4D6 2F          3015 CMA   *
F4D7 6F          3017 MOV   L,A
F4D8 23          3019 INX   H
F4D9 F1          3021 POP   PSW
F4DA AC          3023 XRA   H
F4DB F2F9F5      3025 JP    QHOW
F4DE 78          3027 MOV   A,B      AND ALSO FLIP B
F4DF EE80        3029 XRI   080H
F4E1 47          3031 MOV   B,A
F4E2 C9          3033 RET   *
F4E3 7C          3036 CKHLDE MOV   A,H      SAME SIGN?
F4E4 AA          3038 XRA   D      YES, COMPARE
F4E5 F2E9F4      3040 JP    CK1      NO, XCH AND COMP
F4E8 EB          3042 XCHG  *
F4E9 CDEDFA      3044 CK1   CALL  COMP
F4EC C9          3046 RET   *
F4ED 7C          3049 COMP   MOV   A,H      *** CCMP ***
F4EE BA          3051 CMP   D      COMPARE HL WITH DE
F4EF C0          3053 RNZ   *
F4F0 7D          3055 MOV   A,L      RETURN CORRECT C AND
F4F1 BB          3057 CMP   E      Z FLAGS
                                         BUT OLD A IS LOST

```

F4F2 C9 3059 RET ,
 \*\*\*\*SETVAL\*\*\* FIN \*\*\* ENDCHK \*\*\* & ERROR (& FRIENDS) \*\*\*
 \*
 \* "SETVAL" EXPECTS A VARIABLE, FOLLOWED BY AN EQUAL SIGN AND THEN AN
 \* EXPR. IT EVALUATES THE EXPR. AND SET THE VARIABLE TO THAT VALUE.
 \*
 \* "FIN" CHECKS THE END OF A COMMAND. IF IT ENDED WITH ";" . EXECUTION
 \* CONTINUES. IF IT ENDED WITH A CR, IT FINDS THE NEXT LINE AND
 \* CONTINUE FROM THERE.
 \*
 \* "ENDCHK" CHECKS IF A COMMAND IS ENDED WITH CR. THIS IS REQUIRED IN
 \* CERTAIN COMMANDS. (GOTO, RETURN, AND STOP ETC.)
 \*
 \* "ERROR" PRINTS THE STRING POINTED BY DE (AND ENDS WITH CR). IT THEN
 \* PRINTS THE LINE POINTED BY "CURRENT" WITH A "?" INSERTED AT WHERE THE
 \* OLD TEXT POINTER (SHOULD BE ON TOP OF THE STACK) POINTS TO.
 \* EXECUTION OF TB IS STOPPED AND TBI IS RESTARTED. HOWEVER, IF
 \* "CURRENT" -> ZERO (INDICATING A DIRECT COMMAND), THE DIRECT COMMAND
 \* IS NOT
 \* PRINTED. AND IF "CURRENT" -> NEGATIVE # (INDICATING "INPUT"
 \* COMMAND, THE INPUT LINE IS NOT PRINTED AND EXECUTION IS NOT
 \* TERMINATED BUT CONTINUED AT "INPERR".
 \*
 \* RELATED TO "ERROR" ARE THE FOLLOWING: "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
 \*
 F4F3 CD89F5 3093 SETVAL CALL TSTV \*\*\* SETVAL \*\*\*
 F4F6 DA30F5 3095 JC QWHAT "WHAT?" NO VARIABLE
 F4F9 E5 3097 PUSH H SAVE ADDRESS OF VAR.
 F4FA CDBBF5 3099 TSTC "=",SV1 PASS "=" SIGN
 F4FD 3D 3101
 F4FE 0D 3102
 F4FF CD5BF3 3103 CALL EXPR EVALUATE EXPR.
 F502 44 3105 MOV B,H VALUE IN BC NOW
 F503 4D 3107 MOV C,L
 F504 E1 3109 POP H GET ADDRESS
 F505 71 3111 MOV M,C SAVE VALUE
 F506 23 3113 INX H
 F507 70 3115 MOV M,B
 F508 C9 3117 RET "
 \*
 F509 CD0FF5 3120 FINISH CALL FIN CHECK END OF COMMAND
 F50C C330F5 3122 SV1 JMP QWHAT PRINT "WHAT?" IF WRONG
 \*
 F50F CDBBF5 3125 FIN TSTC ";",FI1 \*\*\* FIN \*\*\*
 F512 3B 3127
 F513 04 3128
 F514 F1 3129 POP PSW
 F515 C321F1 3131 JMP RUNSML
 F518 CDBBF5 3133 FI1 TSTC @CR,FI2 NOT ";", IS IT CR?
 F51B 0D 3135
 F51C 04 3136
 F51D F1 3137 POP PSW YES, PURGE RET ADDR.
 F51E C311F1 3139 JMP RUNNXL RUN NEXT LINE
 F521 C9 3141 FI2 RET " ELSE RETURN TO CALLER
 \*
 F522 1A 3144 IGNBLK LDAX D \*\*\* IGNBLK \*\*\*
 F523 FE20 3146 CPI "
 F525 C0 3148 RNZ .
 F526 13 3150 INX D IN TEXT (WHERE DE->)
 F527 C322F5 3152 JMP IGNBLK AND RETURN THE FIRST
 \* NON-BLANK CHAR. IN A
 F52A CD22F5 3155 ENDCHK CALL IGNBLK \*\*\* ENDCHK \*\*\*
 F52D FE0D 3157 CPI @CR END WITH CR?
 F52F C8 3159 RZ " OK. ELSE SAY: "WHAT?"
 \*
 F530 D5 3162 QWHAT PUSH D \*\*\* QWHAT \*\*\*
 F531 1142F0 3164 AWHAT LXI D,WHAT \*\*\* AWHAT \*\*\*
 F534 CD93F7 3166 ERROR CALL CRLF
 F537 CD65F6 3168 CALL PRTSTG PRINT ERROR MESSAGE

F53A	2AB700	3170	LHLD	CURRNT	GET CURRENT LINE #
F53D	E5	3172	PUSH	H	CHECK THE VALUE
F53E	7E	3174	MOV	A,M	
F53F	23	3176	INX	H	
F54C	B6	3178	ORA	M	
F541	D1	3180	POP	D	
F542	CA26F0	3182	JZ	TELL	IF ZERO, JUST RESTART
F545	7E	3184	MOV	A,M	IF NEGATIVE,
F546	B7	3186	ORA	A	
F547	FAE1F2	3188	JM	INPERR	REDO INPUT
F54A	CDF2F6	3190	CALL	PRTLN	ELSE PRINT THE LINE
F54D	C1	3192	POP	B	
F54E	41	3194	MOV	B,C	
F54F	CDA3F6	3196	CALL	PRTCMS	
F552	3E3F	3198	MVI	A,"?"	PRINT A "?"
F554	CD95F7	3200	CALL	DUTCH	
F557	CD65F6	3202	CALL	PRTSTG	LINE
F55A	C326F0	3204	JMP	TELL	THEN RESTART
F55D	D5	3206	OSORRY	PUSH D	*** OSORRY ***
F55E	114DF0	3208	ASORRY	LXI D,SORRY	*** ASORRY ***
F561	C334F5	3210	JMP	ERROR	

\*  
\*\*\*\*\*  
\*

\* \*\*\* FNDLN (& FRIENDS) \*\*\*

\* '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. 'FNDLP'  
 \* WILL START WITH DE AND SEARCH FOR THE LINE #. 'FNDNXT' WILL BUMP DE  
 \* BY 2, FIND A CR AND THEN START SEARCH. 'FNDSKP' USE DE TO FIND A  
 \* CR, AND THEN START SEARCH.

F564	7C	3232	FNDLN	MOV A,H	*** FNDLN ***
F565	B7	3234	ORA	A	CHECK SIGN OF HL
F566	FAF9F5	3236	JM	QHOW	IT CANNOT BE -
F569	110220	3238	LXI	D,TEXT	INIT. TEXT PCINTER
*					
F56C	13	3241	FNDLP	INX D	IS IT EOT MARK?
F56D	1A	3243	LDAX	D	
F56E	1B	3245	DCX	D	
F56F	87	3247	ADD	A	
F570	D8	3249	RC	,	C,NZ PASSED END
F571	1A	3251	LDAX	D	WE DID NOT, GET BYTE 1
F572	95	3253	SUB	L	IS THIS THE LINE?
F573	47	3255	MOV	E,A	COMPARE LOW ORDER
F574	13	3257	INX	D	
F575	1A	3259	LDAX	D	GET BYTE 2
F576	9C	3261	SBD	H	COMPARE HIGH ORDER
F577	DA7EFS	3263	JC	FL1	NO, NOT THERE YET
F57A	1B	3265	DCX	D	ELSE WE EITHER FOUND
F578	B0	3267	ORA	B	IT, OR IT IS NOT THERE
F57C	C9	3269	RET	,	NC,Z:FOUND; NC,NZ:NO
*					
F57D	13	3272	FNDNXT	INX D	FIND NEXT LINE
F57E	13	3274	FL1	INX D	JUST PASSED BYTE 1 & 2
*					
F57F	1A	3277	FNDSKP	LDAX D	*** FNDSKP ***
F580	FE0D	3279	CPI	@CR	TRY TO FIND CR
F582	C27EF5	3281	JNZ	FL1	KEEP LOOKING
F585	13	3283	INX	D	FOUND CR, SKIP OVER
F586	C36CF5	3285	JMP	FNDLP	CHECK IF END OF TEXT
*					
F589	CD22F5	3288	TSTV	CALL IGNBLK	*** TSTV ***
F58C	D640	3290	SUI	"@"	TEST VARIABLES
F58E	D8	3292	RC	,	C:NOT A VARIABLE
F58F	C2ABF5	3294	JNZ	TV1	NOT "@" ARRAY
F592	13	3296	INX	D	IT IS THE "@" ARRAY
F593	CDSAF4	3298	CALL	PARN	@ SHOULD BE FOLLOWED

F596	29	3300	DAD	H	BY (EXPR) AS ITS INDEX
F597	DAF9F5	3302	JC	QH0W	IS INDEX TOO BIG?
F59A	D5	3304	PUSH	D	WILL IT FIT?
F59B	EB	3306	XCHG	,	
F59C	CDA1F4	3308	CALL	SIZE	FIND SIZE OF FREE
F59F	CDEDFF4	3310	CALL	COMP	AND CHECK THAT
F5A2	DA5EF5	3312	JC	ASORRY	IF NOT, SAY "SORRY"
F5A5	CD5FF6	3314	CALL	LOCR	IF FITS, GET ADDRESS
F5A8	19	3316	DAD	D	OF @(EXPR) AND PUT IT
F5A9	D1	3318	POP	D	IN HL
F5AA	C9	3320	RET	,	C FLAG IS CLEARED
F5AB	FE1B	3322	TV1	CPI	NOT @. IS IT A TO Z?
F5AD	3F	3324	CMC	,	IF NOT RETURN C FLAG
F5AE	D8	3326	RC	,	
F5AF	13	3328	INX	D	IF A THRUH Z
F5B0	218100	3330	LXI	H,VARBGN-2	
F5B3	07	3332	RLC	,	HL->VARIABLE
F5B4	85	3334	ADD	L	RETURN
F5B5	6F	3336	MOV	L,A	WITH C FLAG CLEARED
F5B6	3E00	3338	MVI	A,0	
F5B8	8C	3340	ADC	H	
F5B9	67	3342	MOV	H,A	
F5BA	C9	3344	RET	,	

\*

---

\* \*\*\* TSTCH \*\*\* TSTNUM \*\*\*

\* TSTCH IS USED TO TEST THE NEXT NON-BLANK CHARACTER IN THE TEXT  
 \* (POINTED BY DE) AGAINST THE CHARACTER THAT FOLLOWS THE CALL. IF  
 \* THEY DO NOT MATCH, N BYTES OF CODE WILL BE SKIPPED OVER. WHERE N IS  
 \* BETWEEN 0 AND 255 AND IS STORED IN THE SECOND BYTE FOLLOWING THE  
 \* CALL.

\* TSTNUM IS USED TO CHECK WHETHER THE TEXT (POINTED BY DE) IS A  
 \* NUMBER. IF A NUMBER IS FOUND, B WILL BE NON-ZERO AND HL WILL  
 \* CONTAIN THE VALUE (IN BINARY) OF THE NUMBER. ELSE B AND HL ARE 0.

F5B8	E3	3362	TSTCH	XTHL	*** TSTCH ***
F5BC	CD22F5	3364	CALL	IGNBLK	IGNORE LEADING BLANKS
F5BF	BE	3366	CMP	M	AND TEST THE CHARACTER
F5C0	23	3368	INX	H	COMPARE THE BYTE THAT
F5C1	CACBF5	3370	JZ	TC1	FOLLOWS THE CALL INST.
F5C4	C5	3372	PUSH	B	WITH THE TEXT (DE->)
F5C5	4E	3374	MOV	C,M	IF NOT =, ADD THE 2ND
F5C6	0600	3376	MVI	B,0	BYTE THAT FOLLOWS THE
F5C8	09	3378	DAD	B	CALL TO THE OLD PC
F5C9	C1	3380	POP	B	I.E., DO A RELATIVE
F5CA	1B	3382	DCX	D	JUMP IF NOT =
F5CB	13	3384	TC1	INX	IF =, SKIP THOSE BYTES
F5CC	23	3386	INX	H	AND CONTINUE
F5CD	E3	3388	XTHL	,	
F5CE	C9	3390	RET	,	

F5CF	210000	3393	TSTNUM	LXI	*** TSTNUM ***
F5D2	44	3395	MOV	B,H	TEST IF THE TEXT IS
F5D3	CD22F5	3397	CALL	IGNBLK	A NUMBER
F5D6	FE3C	3399	TN1	CPI	IF NOT, RETURN 0 IN
F5D8	D8	3401	RC	,	B AND HL
F5D9	FE3A	3403	CPI	03AH	IF NUMBERS, CONVERT
F5DB	D0	3405	RNC	,	TO BINARY IN HL AND
F5DC	3EF0	3407	MVI	A,0FOH	SET B TO # OF DIGITS
F5DE	A4	3409	ANA	H	IF H>255, THERE IS NO
F5DF	C2F9F5	3411	JNZ	QH0W	ROOM FOR NEXT DIGIT
F5E2	04	3413	INR	B	B COUNTS # OF DIGITS
F5E3	C5	3415	PUSH	B	HL=10*HL+(NEW DIGIT)
F5E4	44	3417	MOV	B,H	
F5E5	4D	3419	MOV	C,L	WHERE 10* IS DONE BY
F5E6	29	3421	DAD	H	SHIFT AND ADD
F5E7	29	3423	DAD	H	
F5E8	09	3425	DAD	B	
F5E9	29	3427	DAD	H	
F5EA	1A	3429	LDAX	D	
F5EB	13	3431	INX	D	AND (DIGIT) IS FROM
F5EC	E60F	3433	ANI	00FH	STRIPPING THE ASCII

F5EE	85	3435	ADD	L	
F5EF	6F	3437	MOV	L,A	
F5F0	3E00	3439	MVI	A,0	
F5F2	8C	3441	ADC	H	
F5F3	67	3443	MOV	H,A	
F5F4	C1	3445	POP	B	
F5F5	1A	3447	LDAX	D	DO THIS DIGIT AFTER
F5F6	F2D6F5	3449	JP	TN1	DIGIT. S SAYS OVERFLOW
F5F9	D5	3451	QHOB	PUSH D	*** ERROR: "HOW?" ***
F5FA	1148F0	3453	AHOB	LXI D, HOW	
F5FD	C334F5	3455	JMP	ERROR	
*					*****
*					*****
*					*** MVUP *** MVDOWN *** POPA *** & PUSH A ***
*					
*					*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
*					
F600	CDEF4	3474	MVUP	CALL COMP	*** MVUP ***
F603	C8	3476	RZ	*	DE = HL, RETURN
F604	1A	3478	LDAX	D	GET ONE BYTE
F605	02	3480	STAX	B	MOVE IT
F606	13	3482	INX	D	INCREASE BOTH POINTERS
F607	03	3484	INX	B	
F608	C300F6	3486	JNP	MVUP	UNTIL DONE
*					
F60B	78	3489	MVDOWN	MOV A,B	*** MVDOWN ***
F60C	92	3491	SUB	D	TEST IF DE = BC
F60D	C213F6	3493	JNZ	MD1	NO, GO MOVE
F610	79	3495	MOV	A,C	MAYBE, OTHER BYTE?
F611	93	3497	SUB	E	
F612	C8	3499	RZ	*	YES, RETURN
F613	1B	3501	MD1	DCX D	ELSE MOVE A BYTE
F614	2B	3503	DCX	H	BUT FIRST DECREASE
F615	1A	3505	LDAX	D	BOTH POINTERS AND
F616	77	3507	MOV	M,A	THEN DO IT
F617	C30BF6	3509	JMP	MVDOWN	LOOP BACK
*					
F61A	C1	3512	POPA	POP B	BC = RETURN ADDR.
F61B	E1	3514	POP	H	RESTORE LOPVAR, BUT
F61C	22BD00	3516	SHLD	LOPVAR	=0 MEANS NO MORE
F61F	7C	3518	MOV	A,H	
F620	B5	3520	ORA	L	
F621	CA34F6	3522	JZ	PP1	YEP, GO RETURN
F624	E1	3524	POP	H	NOP, RESTORE OTHERS
F625	22BF00	3526	SHLD	LOPINC	
F628	E1	3528	POP	H	
F629	22C100	3530	SHLD	LGPLMT	
F62C	E1	3532	POP	H	
F62D	22C300	3534	SHLD	LOPLN	
F630	E1	3536	POP	H	
F631	22C500	3538	SHLD	LOPPT	
F634	C5	3540	PP1	PUSH B	BC = RETURN ADDR.
F635	C9	3542	RET	*	
*					
F636	215201	3545	PUSHA	LXI H,STKLMT	*** PUSHA ***
F639	CDCE4	3547	CALL	CHGSGN	
F63C	C1	3549	POP	B	BC=RETURN ADDRESS
F63D	39	3551	DAD	SP	IS STACK NEAR THE TOP?
F63E	D25DF5	3553	JNC	GSORRY	YES, SORRY FOR THAT.
F641	2ABD00	3555	LHLD	LOPVAR	ELSE SAVE LOOP VAR.S
F644	7C	3557	MOV	A,H	BUT IF LOPVAR IS 0
F645	B5	3559	ORA	L	THAT WILL BE ALL
F646	CA5CF6	3561	JZ	PP1	
F649	2AC500	3563	LHLD	LOPPT	ELSE, MORE TO SAVE
F64C	E5	3565	PUSH	H	
F64D	2AC300	3567	LHLD	LOPLN	
F650	E5	3569	PUSH	H	

F651	2AC100	3571	LHLD	LOPLMT
F654	E5	3573	PUSH	H
F655	2ABF00	3575	LHLD	LOPINC
F658	E5	3577	PUSH	H
F659	2ABD00	3579	LHLD	LCPVAR
F65C	E5	3581 PU1	PUSH	H
F65D	C5	3583	PUSH	B
F65E	C9	3585	RET	.
F65F	2A0020	3587 LOCR	LHLD	TXTUNF
F662	2B	3589	DCX	H
F663	2B	3591	DCX	H
F664	C9	3593	RET	.
*				
*****				
*				
*** 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 ZERO. REG. A AND B ARE CHANGED. REG. DE POINTS TO WHAT FOLLOWS THE CR OR TO THE ZERO.				
*				
'QTSTG' LOOKS FOR UP-ARROW, SINGLE QUOTE, OR DOUBLE QUOTE. IF NONE OF THESE, RETURN TO CALLER. IF UP-ARROW, OUTPUT A CONTROL CHARACTER. 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' FINDS A SAVED LINE, PRINTS THE LINE # AND A SPACE.				
*				
F665	97	3620	PRTSTG	SUB A
F666	47	3622	PS1	MOV B,A
F667	1A	3624	PS2	LDAX D
F668	13	3626	INX	D
F669	B8	3628	CMP	B
F66A	C8	3630	RZ	.
F66B	CD95F7	3632	CALL	CUTCH
F66E	FE0D	3634	CPI	@CR
F670	C267F6	3636	JNZ	PS2
F673	C9	3638	RET	.
*				*** PRTSTG ***
F674	CDBBF5	3641	QTSTG	TSTC 1**,QT3
F677	22	3643		*** QTSTG ***
F678	0F	3644		
F679	3E22	3645	MVI	A,***
F67B	CD66F6	3647 QT1	CALL	PS1
F67E	FE0D	3649 QT2	CPI	@CR
F680	E1	3651	PUP	H
F681	CA11F1	3653	JZ	RUNNXL
F684	23	3655	INX	H
F685	23	3657	INX	H
F686	23	3659	INX	H
F687	E9	3661	PCHL	.
F688	CDBBF5	3663 QT3	TSTC	027H,QT4
F68B	27	3665		RETURN
F68C	05	3666		IS IT A + ?
F68D	3E27	3667	MVI	A,027H
F68F	C37BF6	3669	JMP	QT1
F692	CDBBF5	3671 QT4	TSTC	05EH,QT5
F695	5E	3673		YES, DO SAME AS IN "
F696	0B	3674		IS IT AN UP-ARROW?
F697	1A	3675	LDAX	D
F698	EE40	3677	XRI	040H
F69A	CD95F7	3679	CALL	CUTCH
F69D	1A	3681	LDAX	D
F69E	13	3683	INX	D
F69F	C37EF6	3685	JMP	QT2
F6A2	C9	3687 QT5	RET	.
F6A3	7B	3689 PRTCHS	MOV	A,E

F6A4	B8	3691	CMP	B	
F6A5	C8	3693	RZ	,	
F6A6	1A	3695	LDAX	D	
F6A7	CD95F7	3697	CALL	CUTCH	
F6AA	13	3699	INX	D	
F6AB	C3A3F6	3701	JMP	PRTCHS	*
*					
F6AE	0600	3704	PRTNUM	DS	0
F6B0	CDCBF4	3706	PN3	MVI	E,0
F6B3	F2B9F6	3708		CALL	CHKSGN
F6B6	062D	3710		JP	PN4
F6B8	0D	3712		MVI	B,"--"
F6B9	D5	3714		DCR	C
F6BA	110A00	3716	PN4	PUSH	D
F6BD	D5	3718		LXI	D,10
F6BE	0D	3720		PUSH	D
F6BF	C5	3722		DCR	C
F6C0	CDAEF4	3724		PUSH	B
F6C3	78	3726	PN5	CALL	DIVIDE
F6C4	B1	3728		MOV	A,B
F6C5	CAD0F6	3730		ORA	C
F6C8	E3	3732		JZ	PN6
F6C9	2D	3734		XTHL	
F6CA	E5	3736		DCR	L
F6CB	60	3738		PUSH	H
F6CC	69	3740		MOV	H,B
F6CD	C3C0F6	3742		MOV	L,C
F6D0	C1	3744		JMP	PNS
F6D1	0D	3746	PN6	PJP	B
F6D2	79	3748	PN7	DCR	C
F6D3	B7	3750		MOV	A,C
F6D4	FADFF6	3752		ORA	A
F6D7	3E20	3754		JM	PN8
F6D9	CD95F7	3756		MVI	A," "
F6DC	C3D1F6	3758		CALL	CUTCH
F6DF	78	3760		JMP	PN7
F6E0	B7	3762	PN8	MOV	A,B
F6E1	C495F7	3764		ORA	A
F6E4	5D	3766		CNZ	CUTCH
F6E5	7B	3768		MOV	E,L
F6E6	FE0A	3770	PN9	MOV	A,E
F6E8	D1	3772		CPI	10
F6E9	C8	3774		POP	D
F6EA	C630	3776		RZ	:
F6EC	CD95F7	3778		ADI	"0"
F6EF	C3E5F6	3780		CALL	CUTCH
*		3782		JMP	PN9
F6F2	1A	3785	PRTLN	LDAX	D
F6F3	6F	3787		MOV	L,A
F6F4	13	3789		INX	D
F6F5	1A	3791		LDAX	D
F6F6	67	3793		MOV	H,A
F6F7	13	3795		INX	D
F6F8	0E04	3797		MVI	C,4
F6FA	CDAEF6	3799		CALL	PRTNUM
F6FD	3E20	3801		MVI	A," "
F6FF	CD95F7	3803		CALL	CUTCH
F702	C9	3805		RET	:
*					
*					
F703	4C495354	3809	TAB1	ITEM	"LIST",LIST
F707	F13B	3811			DIRECT COMMANDS
F709	4E4557	3812		ITEM	"NEW",NEW
F70C	F0FF	3814			
F70E	S2554E	3815		ITEM	"RUN",RUN
F711	F10B	3817			
F713	4E455854	3818	TAB2	ITEM	"NEXT",NEXT
F717	F269	3820			DIRECT/STATEMENT
F719	4C4554	3821		ITEM	"LET",LET
F71C	F34D	3823			
F71E	4946	3824		ITEM	"IF",IFF
F720	F2D0	3826			
F722	474F544F	3827		ITEM	"GOTO",GOTO
F726	F12A	3829			

\*\*\* PRTNUM \*\*\*  
 B=SIGN  
 CHECK SIGN  
 NO SIGN  
 B=SIGN  
 "--" TAKES SPACE  
 DECIMAL  
 SAVE AS A FLAG  
 C=SPACES  
 SAVE SIGN & SPACE  
 DEVIDE HL BY 10  
 RESULT 07  
 YES, WE GOT ALL  
 NO, SAVE REMAINDER  
 AND COUNT SPACE  
 HL IS OLD BC  
 MOVE RESULT TO BC  
 AND DIVIDE BY 10  
 WE GOT ALL DIGITS IN  
 THE STACK  
 LOOK AT SPACE COUNT  
 NO LEADING BLANKS  
 LEADING BLANKS  
 MCRC?  
 PRINT SIGN  
 MAYBE - OR NULL  
 LAST REMAINDER IN E  
 CHECK DIGIT IN E  
 10 IS FLAG FOR NO MORE  
 IF SO, RETURN  
 ELSE COVERT TO ASCII  
 AND PRINT THE DIGIT  
 GO BACK FOR MORE  
 \*\*\* PRTLN \*\*\*  
 LOW ORDER LINE #  
 HIGH CRDER  
 PRINT 4 DIGIT LINE #  
 FOLLOWED BY A BLANK  
 ITEM "LIST",LIST DIRECT COMMANDS  
 ITEM "NEW",NEW  
 ITEM "RUN",RUN  
 ITEM "NEXT",NEXT DIRECT/STATEMENT  
 ITEM "LET",LET  
 ITEM "IF",IFF  
 ITEM "GOTO",GOTO

F728	474F535542	3830	ITEM "GOSUB",GOSUB		
F72D	F1C5	3832			
F72F	52455455	3833	ITEM "RETURN",RETURN		
F733	524EF1E7	3835			
F737	52454D	3836	ITEM "REM",REM		
F73A	F2CA	3838			
F73C	464F52	3839	ITEM "FOR",FOR		
F73F	F202	3841			
F741	494E505554	3842	ITEM "INPUT",INPUT		
F746	F2EB	3844			
F748	5052494E54	3845	ITEM "PRINT",PRINT		
F74D	F16B	3847			
F74F	53544F50	3848	ITEM "STOP",STOP		
F753	F105	3850			
F755	F757	3851	ITEM ,MOREC		
F757	C347F3	3853	MOREC		
		JMP DEFLT	***** *** JMP USER-COMMAND *** *****		
F75A	524E44	3855	TAB3	ITEM "RND",RND	FUNCTIONS
F75D	F46B	3857			
F75F	414253	3858			
F762	F498	3860			
F764	53495A45	3861			
F768	F4A1	3863			
F76A	F76C	3864			
F76C	C349F4	3866	MOREF	JMP NOTF	***** *** JMP USER-FUNCTION *** *****
F76F	544F	3868	TAB4	ITEM "TO",FR1	"FOR" COMMAND
F771	F212	3870			
F773	F530	3871			
F775	53544550	3873	TAB5	ITEM "QWHAT",	
F779	F21E	3875			
F77B	F224	3876			
F77D	3E3D	3878	TAB6	ITEM "STEP",FR2	"FOR" COMMAND
F77F	F365	3880			
F781	23	3881			
F782	F36B	3883			
F784	3E	3884			
F785	F371	3886			
F787	3D	3887			
F788	F380	3889			
F78A	3C3D	3890			
F78C	F378	3892			
F78E	3C	3893			
F78F	F366	3895			
F791	F38C	3896			
		3898	RANEND	EQU *	

\*  
\*\*\*\*\*  
\*\*\*\*\*

\* \*\*\* INPUT OUTPUT ROUTINES \*\*\*

\* USER MUST VERIFY AND/OR MODIFY THESE ROUTINES

\*  
\*\*\*\*\*  
\*\*\*\*\*

\* \*\*\* CRLF \*\*\* OUTCH \*\*\*

\* CRLF WILL OUTPUT A CR. ONLY A & FLAGS MAY CHANGE AT RETURN

\* OUTCH WILL OUTPUT THE CHARACTER IN A. IF THE CHARACTER IS CR, IT  
\* WILL ALSO OUTPUT A LF AND THREE NULLS. FLAGS MAY CHANGE AT RETURN.  
\* OTHER REGISTERS DO NOT.

\* \*\*\* CHKIO \*\*\* GETLN \*\*\*

\* CHKIO CHECKS TO SEE IF THERE IS ANY INPUT. IF NO INPUT, IT RETURNS  
\* WITH Z FLAG. IF THERE IS INPUT, IT FURTHER CHECKS WHETHER INPUT IS  
\* CONTROL-C. IF NOT CONTROL-C, IT RETURNS THE CHARACTER IN A WITH Z  
\* FLAG CLEARED. IF INPUT IS CONTROL-C, CHKIO JUMPS TO "INIT" AND WILL  
\* NOT RETURN. ONLY A & FLAGS MAY CHANGE AT RETURN.

\* \*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 ECHUS. BACK-SPACE IS USED TO DELETE THE LAST CHARACTER (IF THERE  
 \* IS ONE). CR SIGNALS THE END OF THE LINE, AND CAUSE "GETLN" TO  
 \* RETURN. WHEN BUFFER IS FULL, "GETLN" WILL ACCEPT BACK-SPACE OR CR  
 \* ONLY AND WILL IGNORE (AND WILL NOT ECHO) OTHER CHARACTERS. AFTER  
 \* THE INPUT LINE IS STORED IN THE BUFFER, TWO MORE BYTES OF FF ARE  
 \* ALSO STORED AND DE POINTS TO THE LAST FF. A & FLAGS ARE ALSO  
 \* CHANGED AT RETURN.

F793 3E00	3937 CRLF	MVI A,00DH	CR IN A
F795 C3D6F7	3939 OUTCH	JMP @OUTA	***** *** JMP USER-OUTPUT ***
F798 C3ECF7	3941 CHKIO	JMP @INA	***** *** JMP USER-INPUT ***
F79B 11CA00	3943 GETLN	LXI D,BUFFER	***** **** MCDIFY THIS ****
F79E CD95F7	3945 GL1	CALL OUTCH	PROMPT OR ECHO
F7A1 CD98F7	3947 GL2	CALL CHKIO	GET A CHARACTER
F7A4 CAA1F7	3949	JZ GL2	WAIT FOR INPUT
F7A7 FE0A	3951	CPI @LF	
F7A9 CAA1F7	3953	JZ GL2	
F7AC 12	3955 GL3	STAX D	SAVE CH.
F7AD FE06	3957	CPI 008H	IS IT BACK-SPACE?
F7AF C28DF7	3959	JNZ GL4	NO, MORE TESTS
F7B2 78	3961	MOV A,E	YES, DELETE?
F7B3 FECA	3963	CPI BUFFER,>	
F7B5 CAA1F7	3965	JZ GL2	NOTHING TO DELETE
F7B8 1A	3967	LDAX D	DELETE
F7B9 1B	3969	DCX D	
F7BA C39EF7	3971	JMP GL1	
F7BD FE0D	3973 GL4	CPI @CR	WAS IT CR?
F7BF CACDF7	3975	JZ GL5	YES, END OF LINE
F7C2 7B	3977	MOV A,E	ELSE, MORE FREE ROOM?
F7C3 FE4E	3979	CPI BUFEND,>	
F7C5 CAA1F7	3981	JZ GL2	NO, WAIT FOR CR/RUB-OUT
F7C8 1A	3983	LDAX D	YES, BUMF POINTER
F7C9 13	3985	INX D	
F7CA C39EF7	3987	JMP GL1	
F7CD 13	3989 GL5	INX D	END OF LINE
F7CE 13	3991	INX D	BUMP POINTER
F7CF 3EFF	3993	MVI A,OFFH	PUT MARKER AFTER IT
F7D1 12	3995	STAX D	
F7D2 1B	3997	DCX D	
F7D3 C393F7	3999	JMP CRLF	
F7D6 FS	4001 @OUTA	PUSH PSW	OUTPUT ROUTINE
F7D7 DB00	4003 UT1	IN 0	PRINT WHAT IS IN A
F7D9 E601	4005	ANI 001H	TBE BIT
F7DB CAD7F7	4007	JZ OT1	WAIT UNTIL READY
F7DE F1	4009	POP PSW	
F7DF D301	4011	OUT 1	
F7E1 FE0D	4013	CPI @CR	WAS IT CR?
F7E3 C0	4015	RNZ *	NO, RETURN
F7E4 3E0A	4017	MVI A,@LF	YES, GIVE LF
F7E6 CDD6F7	4019	CALL @OUTA	
F7E9 3E0D	4021	MVI A,@CR	
F7EB C9	4023	RET *	
F7EC DB00	4025 @INA	IN 0	
F7EE E602	4027	ANI 002H	DAV BIT
F7FO C8	4029	RZ *	NO INPUT, RETURN ZERO
F7F1 DB01	4031	IN 1	CHECK INPUT
F7F3 E67F	4033	ANI 07FH	
F7F5 FE03	4035	CPI 003H	IS IT CONTROL-C?
F7F7 C0	4037	RNZ *	NO, RETURN CH.
F7F8 C300FD	4039	JMP INIT	YES, RESTART
	4041	END	

## PALO ALTO TINY BASIC V3.0 CROSS REFERENCE

SYMBOL	VALUE	DEFN	REFERENCES
aCR	000D	1388	1545 1548 1551 1554 1557 1990 2517 3135 3158 3280 3635 3650 3974 4014 4022
aind	F7EC	4026	3942
ALF	000A	1385	3952 4018
OUTD	F7D6	4002	3940 4020
ABS	F498	2907	3860
AHOU	F5FA	3454	1896 2079 2740 2752 2786
ASORRY	F55E	3209	3313
AWHAT	F531	3165	2272
BOTRAM	2000	1443	1494
BOTROM	F000	1447	1505 1525 2880
EOTSCR	0080	1439	1452
BUFEND	014E	1484	3980
BUFFER	00CA	1482	1616 2438 3944 3964
CHGSGN	F4CE	3000	2701 2810 3548
CHKIO	F798	3942	1881 1972 3948
CHKSGN	F4CB	2993	2718 2722 2774 2774 2911 3709
CKHLDDE	F4E3	3037	2330 2638
CK1	F4E9	3045	3C41
COMP	F4ED	3050	1700 2274 2876 3045 3311 3475
CRLF	F793	3938	1509 1985 1993 2037 3167 4000
CURRNT	00B7	1460	1598 1872 2081 2117 2185 2338 2417 2456 2492 2498 3171
LEFLT	F347	2515	3854
DFTLMT	4000	1445	1521
DIRECT	F0CC	1748	1628
DIVIDE	F4AE	2949	2730 2896 3727
DV1	F4B9	2965	2955
DV2	F4BB	2967	2971
ENDCHK	F52A	3156	1648 1853 1858 1892 1950 2099 2442
ERROR	F534	3167	3211 3456
EXEC	F0CF	1751	1787 1885 2167 2175 2560 2817
EXPR	F35B	2554	1688 2001 2041 2073 2169 2177 2396 2440 2845 3104
EXPR1	F3A3	2649	2554 2632
EXPR2	F3DF	2706	2661 2669 2699
EXPR3	F443	2815	2706 2714 2766
EX1	F0D3	1755	1767
EX2	F0E6	1777	1781
EX3	F0F0	1789	1761
EX4	F0F2	1791	1795
EX5	F0F7	1797	1775
FIN	F50F	3126	2033 3121
FINISH	F509	3121	2039 2123 2253 2344 2352 2468 2532
FI1	F518	3134	3128
FI2	F521	3142	3136
FL1	F57E	3275	3264 3282
FNDLN	F564	3233	1652 1894 1952 2077
FNDLP	F56C	3242	1665 1974 3286
FNDNXT	F57D	3273	1660
FNDSKP	F57F	3278	2404
FOR	F202	2157	3841
FR1	F212	2169	3670
FR2	F21E	2177	3675
FR3	F224	2181	3877
FR4	F227	2183	2179
FR5	F241	2207	2223 2229
FR6	F242	2209	2205
FR7	F262	2249	2215
GETLN	F79B	3944	1612 2512
GL1	F79E	3946	3972 3988
GL2	F7A1	3948	3950 3954 3966 3982
GL4	F7BD	3974	3960
GL5	F7CD	3990	3976
GOSUB	F1C5	2071	3832
GOTO	F12A	1888	3629
HOW	F048	1553	3454
IFF	F2D0	2396	3826
IF1	F2D3	2398	2393
IGNBLK	F522	3145	1620 1751 3153 3156 3289 3365 3398
INIT	F000	1507	4040

SYMBOL	VALUE	DEFN	REFERENCES
INPERR	F2E1	2411	3189
INPUT	F2EB	2424	2466 3E44
IP1	F2EB	2426	2496
IP11	F324	2478	2474
IP12	F32C	2486	2436
IP3	F2F8	2436	2484
IP5	F30E	2460	2434
IP7	F317	2468	2464
IP8	F31A	2470	2430
KEYWRD	0080	1454	1511
LET	F34D	2522	2530 3E23
LIST	F13B	1936	3811
LOCK	F65F	3588	3315
LCPINC	008F	147C	2183 2292 3527 3576
LQPLMT	00C1	1472	2171 2320 3531 3572
LQPLN	00C3	1474	2187 2336 3535 3568
LUPFT	00C5	1476	2191 2249 2340 3539 3564
LCPVAR	00BD	1468	1602 2091 2163 2195 2266 2312 3517 3556 3580
LS1	F14A	1948	1944
LS2	F151	1954	1976
LT4	F358	2532	2519 2528
MD1	F613	3502	3494
MOREC	F757	3854	3852
MOREF	F76C	3867	3865
MSG	FC2F	1543	1537
MVDDWN	F60B	3490	1708 2245 3510
MVUP	F600	3475	1666 1714 3487
NEW	F0FF	1848	3814
NEXT	F269	2256	3820
NCTF	F449	2819	3867
NX1	F272	2262	2284
NX2	F28C	2286	2276
NX3	F29E	2310	2304
NX4	F2AE	2330	2326
NX5	F2C2	2346	2308
NX6	F2C4	2350	2334
OK	F03F	1547	1606
CT1	F7D7	4004	4008
CUTCH	F795	3940	2029 3201 3633 3680 3698 3759 3767 3781 3804 3946
PARN	F45A	2841	2856 2907 3299
PN4	F6B9	3717	3711
PN5	F6C0	3727	3745
PN6	F6D0	3747	3733
FN7	F6D1	3749	3761
PN8	F6DF	3763	3755
FN9	F6E5	3771	3763
POPA	F61A	3513	2121 2280 2350
PP1	F634	3541	3523
PRINT	F16B	1979	3847
PRTCHS	F6A3	3690	2482 3197 3702
PRTLN	F6F2	3786	1968 3191
PRTNUM	F6AE	3705	2045 3800
PRTSTG	F665	3621	1539 1608 1970 3169 3203
PR1	F178	1989	1983
PR2	F183	1997	2035
PR4	F196	2015	1999
PR5	F19C	2019	2013 2049
PR6	F1A1	2023	1991 2031
PR7	F1AE	2033	2025
PR8	F1B4	2037	2021
PR9	F1BA	2041	2017
PS1	F666	3623	3648
PS2	F667	3625	3637
PURGE	F01B	1529	1850
PUSHA	F636	3546	2071 2157
PU1	F65C	3582	3562
QHJW	F5F9	3452	2009 2691 2804 2862 2866 3026 3237 3303 3412
QSORRY	F55D	3207	1702 3554
QTSTG	F674	3642	2015 2428
QTL	F673	3648	3670
QT2	F67E	3650	3686

SYMBOL	VALUE	DEFN	REFERENCES
GT3	F688	3664	3644
GT4	F692	3672	3666
GT5	F6A2	3688	3674
WHAT	F530	3163	2107 2258 2476 2853 3096 3123 3872
RANEND	F793	3899	2874
RANPNT	00C7	1478	1527 2872 2888
RA1	F488	2882	2878
REM	F2CA	2391	3838
RETURN	F1E7	2099	3835
HND	F46B	2856	3857
START	F053	1594	1541 1684 1855 1867 1954 1962 2408
RUN	F108	1858	3817
RUNNXL	F111	1863	1995 3140 3654
RUNSM	F121	1881	1987 2402 3132
RUNTSL	F11A	1870	1900 2097 2406
SETVAL	F4F3	3094	2159 2524
SIZE	F4A1	2917	3309 3263
SORRY	F04D	1556	3209
STACK	0200	1492	1507 1594
STKGDS	00B9	1462	1604 2085 2095 2101 2113
STKINP	00BB	1466	2411 2504
STKLMT	0152	1488	3546
STOP	F105	1853	3850
ST1	F05C	1600	1596
ST2	F06B	1610	1716
ST3	F0A2	1674	1656
SUBDE	F4C4	2978	2925 2969
SV1	F50C	3123	3102
TAB1	F703	3810	1748
TAB2	F713	3819	1683
TAB3	F75A	3856	2815
TAB4	F76F	3869	2165
TAB5	F775	3874	2173
TAB6	F77D	3879	2558
TC1	F5CB	3385	3371
TELL	F026	1537	1517 3183 3205
TEXT	2002	1498	1529 1535 1860 3239
TNI	F5D6	3400	3450
TOPSCR	0200	1441	1490
TSTCH	F5B8	3363	1942 1981 1989 1997 2019 2023 2462 2526 2649 2657 2663 2693 2708 2760 2841 2847 3100 3126 3134 3642 3664 3672
1STNUM	F5CF	3394	1618 1936 1946 2833
TSTV	F589	3289	2256 2432 2472 2819 3094
TV1	F5AB	3323	3295
TXTLMT	0081	1456	1523 1696 2923
TXTUNF	2000	1496	1531 1664 1672 1676 1704 2917 3588
VARBGN	0083	1458	3331
VARNXT	00BB	1464	2260 2282
WHAT	F042	1550	3165
XPRO	F468	2853	2643 2849
XPR1	F365	2562	3880
XPR2	F36B	2570	3883
XPR3	F371	2578	3886
XPR4	F378	2588	3892
XPR5	F380	2600	3889
XPR6	F386	2608	3895
XPR7	F38C	2616	3897
XPR8	F38E	2620	2562 2570 2578 2588 2600 2608
XPR9	F467	2851	2695 2762
XP11	F3AE	2657	2651
XP12	F3B3	2661	2659
XP13	F3B6	2663	2685 2689
XP14	F3BF	2671	2703
XP15	F3D0	2693	2665
XP16	F3D5	2697	2655
XP21	F3E2	2708	2812
XP22	F401	2742	2732
XP23	F409	2750	2756
XP24	F414	2760	2710
XP25	F435	2798	2748 2758
XP32	F454	2833	2821

PALO ALTO TINY BASIC V3.0 OBJECT CODE

```

:1CF 00000310002CD93F72180003EC3BECA26F0772100402281003EF032C8002166
:1CF 01C00062022002026FF220220112FF0CD65F6C353F054494E59204241534926
:1CF 03800432056332E300D4F4B0D574841543F0D484F573F0D534F5252590D3127
:1CF 0540000002215DF022870021000022BD00228900113FF0CD65F63E3ECD98F739
:1CF 07000D511CA00CDCF5CD22F57C85C1ACCCF01B7C121B7D12C5C57993F5CD2C
:1CF 08C0064F5D5C2A2F0D5CD7DF5C12A0020CD00F66069220020C12A0020F1E518
:1CF 0A800FE03CA53F0855F3E008C572A8100EBCDEDF4D25DF5220020D1CD08F6F0
:1CF 0C400D1E1CD00F6C36BF02102F7CD22F5D51A13FE2ECAF0F023BE CAD3F03E1B
:1CF 0E0007F1B8BEDAF7F0238ED2E6F023D1C3CFF03E7F238ED2F2F07E236EE6FB86
:1CF 0FC0067F1E9CD2AF5C31BF0CD2AF5C353F0CD2AF5110220210000CD6CF5DAC3
:1CF 1180053F0EB22B700EB1313C098F72112F7C3CFF0CD5BF3D5CD2AF5CD64F5B9
:1CF 13400C2FAFS1C31AF1CDCFF5E521FFFFCDDBBF52C03CDCFF5E3CD2AF5CD647D
:1CF 1500GF5DA53F0E37C85C53F02BE3CFD2F6CD65F6CD9F87CD65F5C351F100E8
:1CF 16C0008CD86F53B806CD93F7C321F1CD8BF50D24CD93F7C311F1CD8BF5230E1D
:1CF 18800CD5BF33EC0A5B4C2F9F54DC39CF1CD74F6C3BAF1CDBEF52C13CD8BF5CE
:1CF 1A44002C083E20CD95F7C3A1F1CD0FFSC383F1CD93F7C309F5CD5BF3C5CDAE94
:1CF 1C000F6C1C39CF1CD36F6CD5BF3D5CD64F5C2FAF52AB700E52AE900E52100BD
:1CF 1DC000022BD003922B900C31AF1CD2AF52AB9007C5B5CA30F5F9E122B900E1D1
:1CF 1F80022B700D1CD1AF6C309F5CD36F6CDF3F42B22BD00216EF7C3CFF0CD5BCC
:1CF 21400F322C1002174F7C3CFF0CD5BF3C327F221010022BF002AB70022C3003A
:1CF 23000EB22C500010A002ABD00E8606839C342F2097E23B6CA62F27E2BBAC278
:1CF 24C0041F27EBBC241F2EB21000039444D210A0019CD0BF6F92AC500EBC309BE
:1CF 26800F5CD89F5DA30F522BB00D5EB2ABD007CB5CA31F5CDED4F4CAF2D1CD12
:1CF 284001AF62AB800C372F25E23562ABF00E57CAA19FAF2ACFAC2F2EB2AFB
:1CF 2A000BD007323722AC100F1B7F2AEF2EBCDE3F4D1DAC4F22AC30022B7002A88
:1CF 2B800C500EBC309F5E1D1C1AF6C309F5210000C3D3F2CD5BF37CB5C221F1AC
:1CF 2D800CD7FF5D21AF1C353F02AB800F9E122B700D1D1D5CD74F6C31AF3CD898A
:1CF 2F4000F5DA0E5F3CD2C311CA00CD5BF3CD2AF5D1EB73237E122B700D1F1CD53
:1CF 31000BBF52C03C3EBF2C309F5D5CD89F5D224F3C330F543D1CDA3F6C3F8F289
:1CF 32C001C1D5EB2AB700E521EBF22B700210003922BB00D53E20C5C39BF71A09
:1CF 34800FE0DC5A58F3CDF3F4CD8BF52C03C34DF3C309F5CD3F3E5217CF7C3CFF7
:1CF 36400F0CD8EF3D86FC9CD8EF3C86FC9CD8EF3C8D86FC9CD8EF36FC8D86CC939
:1CF 38000CD8EF3C06FC9CD8EF3D06FC9E1C979E1C1E5C54FDA3F3EBE3CDE3F442
:1CF 39C00D12100003E01C5CDBBF52D06210000C3D5F3CDBBF52800CDDFF3CD8B30
:1CF 3B800F52B15E5CDDF3F3EBE37CAA7A19D1FAB6F3ACF2B6F3C3F9F5CDBBF52DE3
:1CF 3D40092E5C5D0FF3CD3CEF4C38FF3CD43F4CD8BF52A2DE5CD43F40600CDCBF4B0
:1CF 3F000E3CDCBF4EBE37C87CA01F47AB2EBC2FAF57D21000087CA35F419DAFAD5
:1CF 40C00F53DC209F4C335F4CD8BF52F4E5CD43F40600CDCBF4E3CDCBF4EBE355
:1CF 42600EB7AB3CAFAF5C5CDAEF46069C1D17CB7FAF9F57887FCCEF4C3E2F321A7
:1CF 4440059F7C3CFF0CD89F5D5A4F7E2366FFC9CDCFF57887C0CD8BF52809CD33
:1CF 460005BF3CD8BF52901C9C330F5CD5AF47CB7FAF9F5B5CAF9F5D5E52AC7009C
:1CF 47C001193F7CDED4F4A88F42100F05E235622C700E1EBC5CDAEF4C1D123C986
:1CF 49800CD5A4F1BDCBF4B13C92A0020D5EB2A8100CDC4F4D1C9E56C2600CDCB9E9
:1CF 4B400F4417DE1670EFF0CDC4F4D28BF419C97D936F7C9A67C97CB7F07CB528
:1CF 4D000C87CF52F677D2F6F23F1ACF2F9F5788EE8047C97CAAFAE9F4EBCDEF412
:1CF 4EC00C97CBAC07DBBC9CD89F5D430F5E5CD8BF53D0DCD5BF3444DE1712370BD
:1CF 50800C9D0FF5C330F5CD8BF53B04F1C321F1CD8BF50D04F1C311F1C91AFEBE
:1CF 52400020C013C322F5CD22F5FE0DC8D51142F0CD93F7CD65F62AE700E57E2349
:1CF 5400086D1CA26F07E87FAE1F2CDF2F6C141CDA3F63E3FC95F7CD65F6C32642
:1CF 55C00F0D5114D0F3C34F57CB7FAF9F51102202131A1B87D81A9547131A9CDA06
:1CF 578007EF51B80C913131AF1E0DC27EF513C36CF5CD22F5D0640D8C2ABF513CD5
:1CF 594005AF429DAF9F5D5EBCDA1F4CDEDF4DA5EF5CD5FF619D1C9FE1B3FD81307
:1CF 5B800021810007856F3E008C67C9E3CD22F5B2E3CACBF5C54E060009C11B1365
:1CF 5CC0023E3C921000044CD22F5F30D8F3EAD03E0F4A2C9F504C5444D2929CF
:1CF 5E8009291A13E60F856F3E008C67C11AF2D6F5D5114F0C334F5CDED4F4C876
:1CF 604001A021303C300F67892C213F67993C81B2B1A77C308F6C1E122BD007CBE
:1CF 62000B5C34F6E122B00E122C100E122C300E122C500C59215201CDCEF480
:1CF 63C00C139D25DF52ABD007C85C5ACF62AC500E52AC300E52AC100E52ABF0001
:1CF 65800E52ABD00E5C5C92A00202B2BC997471A1388C8CD95F7FE0DC267F6C917
:1CF 67400CDCBF5220F3E22CD66F6FE0DE1CA11F123232E9CDBBF527053E27C368
:1CF 690007BF6CDCBF55E0B1AEE40CD95F71A13C37EF6C97B88C81ACD95F713C3F5
:1CF 6AC00A3F60600CDCBF4F2B9F5062D0D5110A00D50DC5CDAEF47881CAD0F677
:1CF 6C800E32DE5609C3C0F6C10D7987FADFF63E20CD95F7C3D1F67887C495F762
:1CF 6E4005D7BFE0AD1C8C630CD95F7C3E5F61A6F131A67130E04CDAEF63E20CD6
:1CF 7000095F7C94C495354F13B4E4557F0FF52554EF10B4E455854F2694C4554E7
:1CF 71C00F34D4946F2D0474F544FF12A474F53554F21C552455455524EF1E752AC
:1CF 73800454DF2CA464F52F202494E505554F2EB5052494E54F16B53544F50F1FF
:1CF 7540005F757C347F3524E4F46B414253F49853495A45F4A1F76C349F4541D
:1CF 770004FF212F53053544550F21EF2243E3DF36523F3683EF3713DF3803C3D84
:1CF 78C00F3783CF386F38C3E0DC3D6F3ECF711CA00CD95F7CD98F7CAA1F7FE51
:1CF 7A8000CAA1F712FE08C280D77BFECACAA1F71A1B3C39EF7FE0DCACDF77BF07
:1CF 7C4004ECAA1F71A13C39EF713133EFF121BC393F7F5D800E601CAD7F7F1D304
:1BF 7E00001FE0DC03E0ACDD6F73E0DC90B00E602C8DB01E67FFE03C0C300F00C

```