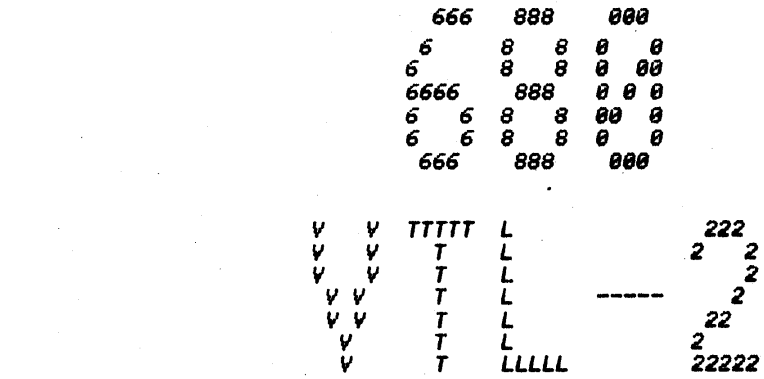
666 888 000
6 8 8 0 0
6 8 8 0 00
6666 888 0 0 0
6 6 8 8 00 0
6 6 8 8 0 0
666 888 000
V V TTTTT L 222
V V T L 2 2
V V T L 2
V V T L ----- 2
V V T L 22
V T L 2
V T LLLLL 22222

V T L - 2

A VERY TINY LANGUAGE

FOR THE ALTAIR 680

INTRODUCTION:

VTL-2 IS THE SECOND VERY TINY LANGUAGE DEVELOPED FOR THE ALTAIR 680 COMPUTER SYSTEM. VTL-2 REPRESENTS AN ENORMOUS IMPROVEMENT OVER THE EARLIER VTL-1 LANGUAGE, AND INCORPORATES SOME THIRTY ADDITIONAL FEATURES IN SPITE OF THESE ENHANCEMENTS, IT STILL REQUIRES ONLY 768 BYTES OF READ-ONLY-MEMORY, AND STILL FITS INTO THE THREE EMPTY PROM SOCKETS ALREADY ON THE 680 CPU BOARD.

THE STATEMENTS THAT MAY BE ENTERED AS INPUT TO THE VTL-2 INTERPRETER ARE OF TWO TYPES.

1. DIRECT STATEMENTS, WHICH HAVE NO LINE NUMBER, AND ARE EXECUTED IMMEDIATLY AFTER THEY ARE ENTERED.

2. PROGRAM STATEMENTS, WHICH ARE USED TO BUILD A PROGRAM, AND ARE NOT EXECUTED UNTIL THE PROGRAM IS RUN. PROGRAM STATEMENTS MUST HAVE LINE NUMBERS IDENTIFIYING THEIR LOCATION IN THE PROGRAM.

VTL-2 IS SIMPLE ENOUGH FOR THE BEGINNER TO USE EASILY, AND YET POWERFUL ENOUGH TO SERVE THE NEEDS OF THE MOST ADVANCED USERS. THE SUBSCRIPTED MEMORY REFERENCE COMMANDS, AND FULL INPUT-OUTPUT FORMAT CONTROL, MAKE VTL-2 A VERSATILE LANGUAGE SUITIBLE FOR SOLVING A WIDE RANGE OF COMPUTER PROBLEMS.

PRELIMINARY CONCEPTS:

LINE NUMBERS MUST PRECEDE EACH PROGRAM STATEMENT, AND MUST BE SEPARATED FROM THAT STATEMENT BY A SINGLE BLANK SPACE. THESE NUMBERS MUST BE IN THE RANGE OF 1-65535. LINE NUMBER ZERO IS NOT PERMITTED. EACH LINE ENDS WITH A CARRIAGE RETURN, AND MUST BE LESS THAN 73 CHARACTERS LONG.

IT IS RECOMMENDED THAT LINES BE NUMBERED IN STEPS OF TEN (10, 20, 30,.... ETC.) SO THAT NEW STATEMENTS MAY BE INSERTED IF NECESSARY.

VARIABLES MAY BE REPRESENTED BY ANY SINGLE ALPHABETIC OR SPECIAL CHARACTER (EG. PUNCTUATION MARK. !"#$%&'()=-+*:;?/>.<,[]). MOST OF THESE ARE AVAILABLE FOR THE USER TO DEFINE AS HE WISHES. A FEW OF THE VARIABLE NAMES HOWEVER, HAVE BEEN SET ASIDE FOR SPECIAL PURPOSES. THESE SO-CALLED "SYSTEM VARIABLES" WILL BE DISCUSSED IN DETAIL BELOW.

THE VALUE ASSIGNED TO A VARIABLE MAY BE EITHER A NUMERIC VALUE IN THE RANGE 0-65535, OR A SINGLE ASCII CHARACTER (INCLUDING CONTROL-CHARACTERS). NUMERIC AND STRING VALUES MAY BE FREELY INTERCHANGED, IN WHICH CASE, THE CHARACTERS ARE EQUIVALENT TO THE DECIMAL VALUE OF THEIR ASCII CODE REPRESENTATION. THUS, IT BECOMES POSSIBLE TO ADD 1 TO THE LETTER "A", GIVING AS A RESULT, THE LETTER "B".

THE ARITHMETIC OPERATIONS PERMITTED FOR USE IN EXPRESSIONS ARE:

```
+  (ADDITION)
-  (SUBTRACTION)
*  (MULTIPLICATION)
/  (DIVISION)
=  (TEST FOR EQUALITY)
>  (TEST FOR GREATER THAN OR EQUAL TO)
<  (TEST FOR LESS THAN)
```

THE TEST OPERATIONS, EQUAL TO, GREATER THAN OR EQUAL TO, AND LESS THAN, ALL RETURN A VALUE OF ZERO IF THE TEST FAILS, AND A VALUE OF ONE IF THE TEST IS SUCCESSFUL.

EXPRESSIONS IN VTL-2 MAY CONTAIN ANY NUMBER OF VARIABLES OR NUMERIC VALUES (LITERALS) CONNECTED BY ANY OF THE ABOVE OPERATION SYMBOLS. PARENTHESES MAY BE USED TO ALTER THE ORDER OF EXECUTION OF THE OPERATIONS. IF NO PARENTHESES ARE INCLUDED, THE OPERATIONS PROCEED IN STRICTLY RIGHT TO LEFT ORDER.

THE VALUE RESULTING FROM THE EXPRESSION MUST BE ASSIGNED TO SOME VARIABLE NAME. THIS IS DONE WITH THE EQUAL SIGN. NOTE THAT THE SYMBOL HAS TWO MEANINGS DEPENDING ON WHERE IT OCCURS IN THE EXPRESSION. THE EXPRESSION "A=B=C" MEANS TEST B AND C FOR EQUALITY. IF THEY ARE EQUAL, PUT A ONE IN A; IF THEY ARE UNEQUAL, PUT A ZERO IN A.

SOME EXAMPLES OF VALID ARITHMETIC EXPRESSIONS WOULD BE:

```
Y=A*(X*X)+B*X+C          WITH LEFT TO RIGHT EXECUTION THIS IS
                         EQUIVELANT TO:  Y=(A*X*X+B)*X+C
Y=(A*X*X)+(B*X)+C        WHICH IS EQUIVALENT TO:  AX↑2+BX+C
```

NOTICE HOW THE ABSENCE OF PARENTHESES AROUND THE QUUANTITY B*X IN THE FIRST EXPRESSION HAS COMPLETELY ALTERED ITS MEANING. KEEP THE LEFT TO RIGHT ORDER IN MIND, AND WHEN IN DOUBT, USE PARENTHESES TO CONTROL THE ORDER OF EVALUATION.

SYSTEM VARIABLES:

IN ORDER TO CONSERVE SPACE, AND TO PROVIDE A MORE CONSISTENT SYNTAX, VTL-2 USES "SYSTEM VARIABLES" TO ACCONPLISH FUNCTIONS USUALY DONE WITH SPECIAL KEY WORDS IN OTHER LANGUAGES. THIS CONVENTION IS PROBABLY THE SINGLE MOST IMPORTANT REASON FOR ITS TINY SIZE.

THESE SPECIAL VARIABLES ARE USED FOR SUCH FUNCTIONS AS THE BASIC 'PRINT, GOTO, GOSUB, RETURN, IF, AND RANDOM' FUNCTIONS.

THE SYSTEM VARIABLE "NUMBER" OR "POUND SIGN" (#) REPRESENTS THE LINE NUMBER OF THE LINE BEING EXECUTED. UNTIL THE STATEMENT HAS BEEN COMPLETED, IT WILL CONTAIN THE CURRENT LINE NUMBER. SO THAT THE STATEMENT:

```
100 A=#
```

IS EQUIVELENT TO SIMPLY WRITING "100 A=100". AFTER COMPLETION OF A LINE, THIS VARIABLE WILL CONTAIN THE NUMBER OF THE NEXT LINE TO BE EXECUTED. IF NOTHING IS DONE TO THE VARIABLE, THIS WILL BE THE NEXT LINE IN THE PROGRAM TEXT. IF A STATEMENT CHANGES #, HOWEVER, THE NEXT LINE EXECUTED WILL BE THE LINE WITH THE NUMBER THAT MATCHES THE VALUE OF #. THUS THE VARIABLE # MAY BE USED TO TRANSFER CONTROL TO A DIFFERENT PART OF THE PROGRAM.

THIS IS THE VTL-2 EQUIVELANT OF THE BASIC "GOTO" STATEMENT. FOR EXAMPLE:

```
#=300   MEANS   "GOTO 300"
```

IF THE # VARIABLE SHOULD EVER BE SET TO ZERO BY SOME STATEMENT, THIS VALUE WILL BE IGNORED, AND THE PROGRAM WILL PROCEED AS IF NO CHANGE HAD TAKEN PLACE. THIS FACT ALLOWS US TO WRITE "IF" STATEMENTS IN VTL-2. CONSIDER THE FOLLOWING EXAMPLE:

```
10 X=1            SET X EQUAL TO 1
20 #=(X=25)*50    IF X=25 GOTO 50
30 X=X+1          ADD 1 TO X
40 #=20           GOTO 20
50 .....          CONTINUE
```

NOTICE THAT THE QUANTITY (X=25) WILL HAVE THE VALUE ONE, IF IT IS TRUE THAT X IS EQUAL TO 25, AND THE VALUE ZERO IF IT IS FALSE. WHEN THIS LOGICAL VALUE IS MULTIPLIED TIMES 50, THE RESULT WILL BE EITHER ZERO, OR 50. IF IT IS 50 THE STATEMENT CAUSES A "GOTO 50" TO OCCUR. IF THE VALUE IS ZERO, A "GOTO 0" WHICH IS A DUMMY OPERATION, CAUSES THE NEXT STATEMENT DOWN (NUMBER 30) TO BE EXECUTED.

TAKING ADVANTAGE OF LEFT-TO-RIGHT EVALUATION, TWO BYTES OF MEMORY COULD BE SAVED BY WRITING:

```
20 #=X=25*50
```

EACH AND EVERY TIME THE VALUE OF # IS CHANGED BY A PROGRAM STATEMENT, THE OLD-VALUE+1 IS SAVED IN THE SYSTEM VARIABLE "EXCLAMATION POINT" (!). IN OTHER WORDS AFTER EXECUTING A GOTO, THE LINE NUMBER OF THE LINE THAT FOLLOWS THE GOTO IS SAVED SO THAT A SUBROUTINE WILL KNOW WHICH PROGRAM STATEMENT CALLED IT, AND WILL KNOW WHERE TO RETURN WHEN FINISHED. THUS THE # VARIABLE IS USED FOR BOTH GOTO AND GOSUB OPERATIONS. FOR EXAMPLE:

```
10 X=1
20 #=100
30 X=2
40 #=100
50 X=3
60 #=100
....
100 X=X*X
110 #=!          (GOTO BACK WHERE YOU CAME FROM)
```

IN THIS EXAMPLE, CONTROL PROCEEDS FROM LINE 20 TO LINE 100. AFTER THAT, LINE 110 CAUSES CONTROL TO RETURN TO LINE 30. WHEN LINE 40 IS EXECUTED, THE SUBROUTINE AT 100 WILL RETURN TO LINE 50.

THE ACTUAL VALUE STORED IN THE ! VARIABLE IS (OLD LINE NUMBER+1) BUT, VTL-2, IF IT DOES NOT FIND THE EXACT LINE NUMBER IT IS SEARCHING FOR, WILL TAKE THE NEXT HIGHER LINE NUMBER. THEREFORE, IF A PROGRAM STATEMENT SAYS "#=52" AND THERE ARE LINES NUMBERED 50 AND 60 WITH NOTHING IN BETWEEN, CONTROL PASSES TO THE NEXT HIGHER LINE NUMBER, 60.

THE SYSTEM VARIABLE "QUESTION MARK" (?) REPRESENTS THE USER'S TERMINAL. IT CAN BE EITHER AN INPUT, OR AN OUTPUT, DEPENDING ON WHICH SIDE OF THE EQUAL SIGN IT APPEARS.

THE STATEMENT "?=A" IS INTERPRETED AS "PRINT A", AND THE STATEMETNT "X=?"IS INTERPRETED AS "INPUT X". NOTE THAT THE "?" MAY BE INCLUDED ANYWHERE IN THE EXPRESSION. FOR EXAMPLE THE PROGRAM:

```
10 ?="ENTER THREE VALUES"
20 A=(?+?+?)/3
30 ?="THE AVERAGE IS"
40 ?=A
```

WILL REQUEST THREE INPUTS WHILE EXECUTING LINE 20.

WHEN TYPING IN A REPLY TO A REQUEST FOR INPUT, THE USER MAY ENTER ANY ONE OF THREE DIFFERENT TYPES OF DATA:

1. A DECIMAL NUMBER
2. A VARIABLE NAME
3. ANY VALID VTL-2 EXPRESSION

THUS, FOR EXAMPLE, THE USER MAY REPLY WITH SUCH THINGS AS "1004" OR "A+B*(9/X)". IN EACH CASE THE EXPRESSION IS COMPLETELY EVALUATED BEFORE THE RESULT IS PASSED TO THE INPUT STATEMENT. THE ONLY EXCEPTION IS THAT YOU ARE NOT ALLOWED TO RESPOND WITH ANOTHER QUESTION MARK AS THIS WILL MESS UP THE LINE POINTER IN THE INTERPRETER, CAUSING IT TO RETURN AN IMPROPER VALUE.

IF A CARRIAGE-RETURN, WITH NO VALUE, IS TYPED IN RESPONSE TO A REQUEST FOR INPUT, THE INTERPRETER WILL RETURN SOME UNDEFINED VALUE. THEREFORE, THIS IS NOT RECOMMENDED.

WHEN THE QUESTION MARK IS ON THE LEFT SIDE OF THE FIRST EQUAL SIGN IT REPRESENTS A PRINT STATEMENT. WHEN THIS OCCURS, EITHER OF TWO DIFFERENT THINGS MAY BE ON THE RIGHT SIDE OF THE EQUAL SIGN:

1. ANY VALID VTL-2 EXPRESSION (AS DEFINED ABOVE)
2. A STRING OF CHARACTERS ENCLOSED IN QUOTE MARKS ("")

WHEN THE EXPRESSION IS A NUMERIC ONE, THE VALUE IS COMPUTED, AND PRINTED AS A LEFT ADJUSTED, UNSIGNED, DECIMAL INTEGER, WITH NO LEADING OR TRAILING BLANKS. A CARRIAGE RETURN NEVER FOLLOWS THE PRINTING OF A DECIMAL VALUE.

WHEN THE EXPRESSION IS A QUOTED CHARACTER STRING, THE ACTUAL STRING OF CHARACTERS IS PRINTED WITH NO LEADING OR TRAILING BLANKS. A CARRIAGE-RETURN LINE-FEED SEQUENCE WILL FOLLOW THE PRINTING OF A STRING UNLESS A SEMICOLON FOLLOWS THE CLOSING QUOTE.

THE OMMISION OF LEADING AND TRAILING BLANKS ALLOWS COMPLETE CONTROL OF FORMATTING PRINTED OUTPUT. FOR EXAMPLE THE PROGRAM:

```
10 ?=50/2
20 ?=",";
30 ?=265+3
40 ?=".";
50 ?=16
```

WILL PRINT THE LINE: "25,268.16" WITH NO SPACES BETWEEN THE PIECES. THIS FEATURE IS MOST OFTEN USED IN FLOATING POINT, AND MULTIPLE PRECISION SUBROUTINES. (SEE "FACTORIALS" IN THE SAMPLE PROGRAM SECTION.)

IF, AT ANY TIME, IT IS DESIRED TO HAVE A CARRIAGE-RETURN LINE-FEED PRINTED, THE STATEMENT ?="" WILL ACOMPLISH THIS.

THE SYSTEM VARIABLE "PER-CENT" (%) CONTAINS THE VALUE OF THE REMAINDER OF THE LAST DIVIIDE OPERATION. THIS VALUE WILL REMAIN THE SAME UNTIL THE NEXT DIVIDE OPERATION.

THE SYSTEM VARIABLE "APOSTROPHE" (') REPRESENTS A RANDOM NUMBER. THIS NUMBER WILL HAVE AN UNPREDICTABLE VALUE IN THE RANGE 0-65535. IF CALLED TWICE ON THE SAME LINE, THE SAME VALUE WILL BE RETURNED BOTH TIMES. THE VALUE OF THE VARIABLE IS SCRAMBLED EACH TIME ANY STATEMENT IS EXECUTED. THEREFORE, FOR BEST RESULTS, IT IS HIGHLY RECOMMENDED THAT AT LEAST ONE OTHER COMPUTATION BE PERFORMED BEFORE THE VALUE IS AGAIN CALLED FOR. THIS MAY EVEN BE A SIMPLE DUMMY STATEMENT SUCH AS "Z=Z+7". FOR AN EXAMPLE OF THIS SEE "DON'T LOSE YOUR AT" IN THE SAMPLE PROGRAMS SECTION.

IN ADDITION TO DECIMAL NUMERIC INPUT AND OUTPUT, THE SYSTEM VARIABLE "DOLLAR SIGN" ($) IS USED TO INPUT AND OUTPUT SINGLE CHARACTERS AS WITH THE QUESTION MARK VARIABLE, "A=$" MEANS "INPUT A SINGLE ASCII CHARACTER AND PLACE ITS NUMERIC VALUE IN A". SIMILARLY, "$=X" MEANS "PRINT THE SINGLE ASCII CHARACTER WHOSE VALUE IS STORED IN X". FOR EXAMPLE THE PROGRAM:

```
10 A=65
20 $=A
30 A=A+1
40 #=A<91*20
50 ?=""
```

WILL PRINT OUT, AS ONE CONTINUOUS STRING, ALL THE LETTERS OF THE ALPHABET: ABCDEFGHIJKLMNOPQRSTUVWXYZ. IF YOU WISH TO FIND OUT WHAT DECIMAL VALUES CORRESPOND TO WHICH CHARACTERS, THESE CAN BE FOUND IN THE 680 REFERENCE MANUAL, OR SIMPLY COMPUTED BY TYPING THE DIRECT STATEMENT: "?=$" AND THEN ENTERING THE CHARACTER WHOSE DECIMAL VALUE IS TO BE FOUND

THE SYSTEM VARIABLE "ASTERISK" (*) REPRESENTS THE MEMORY SIZE OF YOUR COMPUTER. FOR A 1K SYSTEM THIS WOULD BE 1024, FOR A 17K SYSTEM IT WOULD BE 17*1024. WHEN THE MACHINE IS FIRST TURNED ON, AND VTL-2 IS CALLED FOR THE FIRST TIME, THE USER MUST TYPE IN THE VALUE OF THE * VARIABLE AS A DIRECT STATEMENT. "*=1024" FOR EXMAPLE.

IF A PERSON WISHES TO ALLOT SPACE FOR USER DEFINED MACHINE LANGUAGE SUBROUTINES, THEN THE VARIABLE * IS SET EQUAL TO THE BOTTOM OF THE FIRST BYTE REQUIRED BY THE USER DEFINED ROUTINE.

THE SYSTEM VARIABLE "AMPERSAND" (&) REPRESENTS THE NEXT AVAILABLE BYTE OF MEMORY IN THE PROGRAM BUFFER. WHEN FIRST CALLING VTL-2, OR WHEN IT IS DESIRED TO ERASE THE PRESENT PROGRAM, THIS MUST BE INITIALIZED TO THE VALUE 264. "&=264" AS A DIRECT STATEMENT.

AT ANY GIVEN TIME, THE USER MAY FIND OUT HOW MUCH OF HIS MEMORY STILL REMAINS UNUSED BY TYPING "?=*-&". THIS WILL CAUSE THE SYSTEM TO RESPOND WITH THE NUMBER OF BYTES REMAINING. A MINIMUM OF AT LEAST 3 BYTES ARE NEEDED FOR ANY LINE OF VTL-2. THE LINE NUMBERS ARE SAVED IN BINARY, AND REQUIRE TWO BYTES REGARDLESS OF THEIR DECIMAL VALUES. THE LINES "1 X=Y" AND "65000 X=Y" BOTH TAKE UP AN IDENTICAL 7 BYTES OF MEMORY, AND ARE EXAMPLES OF THE NORMAL MINIMUM VALID VTL-2 LINE.

ANY MEMORY REMAINING PAST THE END OF A PROGRAM MAY BE USED AS ARRAY STORAGE. THIS ARRAY STORAGE MAY BE USED FOR SAVING NUMERIC OR STRING VALUES. THE ARRAY DOES NOT HAVE A NAME, SINCE THERE IS ONLY ONE, BUT, IT CAN BE DIVIDED UP INTO SEVERAL PIECES AND USED FOR DIFFERENT GROUPS OF DATA. (SEE "CIPHER" IN THE SAMPLE PROGRAMS SECTION.) A SUBSCRIPT EXPRESSION IS IDENTIFIED BY A COLON AND A RIGHT PARENTHESES. THE COLON MARKS THE BEGINNING OF THE EXPRESSION AND THE RIGHT PARENTHESES MARKS THE END. THUS, FOR EXAMPLE, ":1)=0" PLACES A ZERO IN THE FIRST TWO BYTE WORD PAST THE END OF THE PROGRAM, AND ":2+7)=A" PLACES THE VALUE OF A IN THE 9TH TWO-BYTE WORD PAST THE END OF THE PROGRAM.

SUBSCRIPTS SHOULD NOT BE ALLOWED TO BE LESS THAN ONE (1) AS THIS WILL POINT THE SUBSCRIPT INTO THE PROGRAM AND COULD CAUSE IT TO BE WIPED OUT.

SUBSCRIPT EXPRESSIONS MAY BE ANY VALID VTL-2 NUMERIC EXPRESSIONS THIS EXMAPLE SHOULD CLARIFY THE USE OF SUBSCRIPT EXPRESSIONS:

```
10 I=1            SET POINTER
20 :I)=$          INPUT A CHARACTER TO NEXT ARRAY WORD
30 #=:1)=13*60    GOTO 60 IF ITS A CARRIAGE RETURN CHARACTER
40 I=I+1          POINT TO NEXT ARRAY WORD
50 #=20           GO GET ANOTHER CHARACTER
60 ?=""           PRINT CARRIAGE RETURN-LINE FEED
70 I=1            RESET POINTER
80 $=:I)          PRINT ITH CHARACTER
90 #=:I)=13*120   IF CARRIAGE RETURN THEN GOTO 120
100 I=I+1         POINT TO NEXT CHARACTER
110 #=80          GO GET NEXT CHARACTER
120 ?=""          PRINT CARRIAGE RETURN-LINE FEED
```

THE ABOVE EXAMPLE WILL READ IN ANY STRING OF CHARACTERS TYPED BY THE USER, SUCH AS A SENTANCE, OR PARAGRAPH, UNTIL A CARRIAGE RETURN IS TYPED. IT WILL THEN ECHO BACK THE COMPLETE STRING AS IT WAS TYPED.

FOR FURTHER EXAMPLES, STUDY THE GAME PROGRAMS WHICH USE CHARACTER INPUT AND THOSE THAT HAVE ARRAYS REPRESENTING THE PLAYING BOARD. THESE WILL BE FOUND IN THE SAMPLE PROGRAMS SECTION.

SINCE SUBSCRIPTS REFER TO TWO BYTE WORDS, AND SINCE VALUES AS LARGE AS 65535 ARE ALLOWED AS SUBSCRIPTS, IT IS POSSIBLE THAT LARGE VALUES IN THE SUBSCRIPT EXPRESSION MAY "WRAP AROUND" THE END OF MEMORY AND REACH LOCATIONS WITHIN THE PROGRAM TEXT. THEREFORE, THERE IS A DANGER THAT VTL-2 PROGRAMS USING COMPUTED SUBSCRIPTS MAY "CLOBBER" THEMSELVES. ON THE OTHER HAND, THIS ALSO MEANS THAT A VTL-2 PROGRAM MAY MODIFY ITSELF, ALTHOUGH THIS PRACTICE IS NOT RECOMMENDED.

THE SYSTEM VARIABLE "GREATER THAN" (>) IS USED TO PASS A VALUE TO A MACHINE LANGUAGE SUBROUTINE. WHEN ENCOUNTERED ON THE LEFT SIDE OF THE EQUAL SIGN, THE EXPRESSION IS EVALUATED, THE VALUE PLACED AS A 16 BIT INTEGER IN THE A AND B REGISTERS, AND A SOFTWARE INTERUPT IS GENERATED. (SEE 680 MANUAL FOR DETAILS ON USER INTERUPT HANDLING.) AT THE CONCLUSSION OF THE MACHINE LANGUAGE SUBROUTINE, A RETURN-FROM-INTERUPT INSTRUCTION RETURNS CONTROL TO VTL-2, AND PLACES THE VALUE FOUND IN THE A AND B REGISTERS INTO THE SYSTEM VARIABLE >.

NOTE THAT THE VALUE STORED IN > IS PULLED OFF THE STACK BY THE RTI INSTRUCTION, SO THAT IF YOU WISH TO CHANGE THE VALUE PLACED INTO THE VARIABLE > YOU SHOULD FIRST 'PUL' THE CONDITION CODES OFF THE STACK THEN 'PUL' THE B & A REGISTERS OFF THE STACK. THEN AFTER REPLACING THE VALUES IN THE A & B REGISTERS WITH THE NEW VALUES, YOU SHOULD 'PUS' THE A REGISTER ONCE AND THE B REGISTER TWICE. WHEN THIS ROUTINE IS FOLLOWED IT WILL PLACE THE PROPER VALUES INTO THE STACK FOR A RETURN-FROM-INTERUPT INSTRUCTION TO WORK CORRECTLY.

IF NO INTERUPT SERVICE ROUTINE IS USED, WHEN > IS ENCOUNTERED AS THE FIRST CHARACTER ON ANY LINE, CONTROL IS PASSED TO THE 680 MONITOR. FROM THERE, IF THE STACK HAS NOT BEEN DISTURBED, ONE MAY RETURN TO VTL-2 WITH THE MONITOR RESUME COMMAND. (P).

THERE IS NO "END" STATEMENT IN VTL-2. THE INTERPRETER SIMPLY CONTINUES SEQUENTIALY THROUGH THE PROGRAM UNTIL IT RUNS OUT OF LINES TO EXECUTE, OR UNTIL A STATEMENT IS ENCOUNTERED WHICH WILL TRY TO TRANSFER CONTROL TO A LINE THAT IS GREATER IN NUMBER THAN ANY IN THE PROGRAM.

OPERATIONAL CHARACTERISTICS

WHEN THE 680 IS FIRST TURNED ON THE FOLLOWING THINGS MUST BE DONE BEFORE ANY VTL-2 PROGRAM MAY BE ENTERED. THE SYSTEM COMES UP WITH THE MONITOR IN CONTROL. CONTROL IS PASSED TO THE VTL-2 INTERPRETER BY TYPING THE MONITOR COMMAND: JFC00. THE MONITOR WILL PLACE A SPACE BETWEEN THE "J" AND THE MACHINE ADDRESS "FC00". NO SPACES SHOULD BE TYPED, AND NO CARRIAGE RETURN IS NECESSARY.

ONCE VTL-2 IS IN CONTROL, THE MESSAGE "OK" WILL BE PRINTED. THE NEXT STEP IS TO SET YOUR MEMORY SIZE. THIS IS DONE BY TYPING *=1024 FOR A 1K SYSTEM, *=1024*17 FOR A 17K SYSTEM, AND SO ON.

FINALY, THE USER MUST SET THE "END OF PROGRAM" POINTER. THIS IS DONE BY TYPING &=264. THIS NUMBER WILL BE THE SAME FOR ALL SYSTEMS REGARDLESS OF MEMORY SIZE.

VTL-2 IS NOW READY TO BEGIN ACCEPTING PROGRAMS AND COMMANDS. IF AT ANY TIME IT IS DESIRED TO ERASE THE PROGRAM IN MEMORY, REPEAT THE LAST TWO STEPS GIVEN ABOVE. THIS WILL RE-INITIALIZE THE VTL-2 PROGRAM BUFFER SPACE.

WHEN A PROGRAM LINE IS ENTERED, IT WILL BE INSERTED INTO ITS PROPER PLACE IN THE PROGRAM TEXT. THE LINE NUMBER INDICATES WHERE IT WILL BE INSERTED. IF THE LINE JUST TYPED HAS THE SAME LINE NUMBER AS A LINE ALREADY IN THE TEXT, THE OLD LINE WILL BE REPLACED BY THE NEW LINE. IF THE LINE NUMBER ONLY IS TYPED, FOLLOWED IMMEDIATLY BY A CARRIAGE RETURN, THE LINE WITH THAT NUMBER WILL BE DELETED.

WHILE TYPING IN PROGRAM LINES, THE SYSTEM SHOULD SINGLE SPACE, AND MAKE NO REPLIES TO LINES ENTERED. IF, AFTER TYPING A LINE, THE SYSTEM DOUBLE SPACES DOWN, AND PRINTS "OK" THAT INDICATES THAT THERE WAS NOT ENOUGH MEMORY AVAILABLE TO INSERT THE NEW LINE JUST TYPED.

THE USER MAY CHECK TO SEE HOW MUCH MEMORY IS STILL AVAILABLE BY TYPING THE DIRECT STATEMENT (NO LINE NUMBER) "?=*-&". THE SYSTEM WILL RESPOND WITH THE NUMBER OF UNUSED BYTES REMAINING.

WHILE TYPING IN A LINE, THE BACK-ARROW KEY (SHIFT-O ON SOME TERMINALS, OR UNDERLINE ON OTHERS) WILL CAUSE THE LAST CHARACTER TYPED TO BE DELETED FROM THE INPUT BUFFER. THE CHARACTER WILL STILL APPEAR ON THE SCREEN OR PRINTOUT, BUT WILL NO LONGER BE IN MEMORY. THUS THE LINE "A=B*C__+N" GOES IN AS "A=B+N", WHERE THE "*C" WAS ERASED IN MEMORY BY TWO BACK-ARROW CHARACTERS.

AT ANY TIME BEFORE HITTING RETURN, THE ENTIRE LINE MAY BE ERASED FROM MEMORY BY TYPING THE AT-SIGN CHARACTER (@) (SHIFT-P OR "CANCEL" ON SOME TERMINALS.)

TYPING THE SINGLE CHARACTER ZERO (0) FOLLOWED BY A CARRIAGE RETURN, CAUSES THE SYSTEM TO PRINT OUT A LISTING OF THE PROGRAM.

WHILE PRINTING IS TAKING PLACE, WHETHER AS A PROGRAM LISTING, OR AS OUTPUT FROM A PROGRAM, THE OPERATION MAY BE CANCELLED, AND CONTROL RETURNED TO VTL-2 BY PRESSING CONTROL-C. WHEN THIS IS DONE, THE SYSTEM COMPLETES IT'S CURRENT PRINT STATEMENT, AND THEN PRINTS "OK" TO ACKNOWLEDGE THE INTERUPTION.

IN ADDITION TO THIS, ANY OTHER KEY (PREFERABLY A NON-PRINTING CONTROL CHARACTER SUCH AS CONTROL-A) MAY BE PRESSED. THIS WILL CAUSE THE SYSTEM TO TEMPORARILY SUSPEND OPERATION, AND WAIT FOR ANOTHER KEY TO BE PRESSED. (AGAIN PREFERABLY A NON PRINTING CHARACTER.)

THIS FEATURE ALLOWS USERS WITH VIDEO TERMINALS TO LIST THEIR PROGRAMS A SECTION AT A TIME, HITTING CONTROL-A TO STOP THE LISTING, AND HITTING IT AGAIN TO RESUME LISTING.

NOTE THAT THESE CHARACTERS ALSO AFFECT PRINTING BEING DONE BY A PROGRAM. YOU MAY TEMPORARILY HALT YOUR PROGRAM WITH A CONTROL-A, AND START IT UP AGAIN WITH ANOTHER CONTROL-A. THESE KEYS WORK ONLY DURING PRINTING WHICH USES THE QUESTION MARK SYSTEM VARIABLE. STRING PRINTING WITH THE DOLLAR SIGN VARIABLE WILL NOT INTERUPT IN THIS MANNER. THIS ALLOWS THE USER THE OPTION OF MAKING HIS PROGRAM INTERUPTABLE OR NON-INTERUPTABLE.

SHOULD AN UNINTERUPTABLE PROGRAM BECOME "LOCKED UP" IN A LOOP, THE ONLY WAY TO BREAK OUT IS WITH THE FRONT PANEL RESET SWITCH. WHEN THIS IS DONE, THE JFC00 MUST BE TYPED TO RETURN TO VTL-2, BUT THE REMAINING STEPS LISTED ABOVE TO CLEAR THE PROGRAM MUST NOT BE PERFORMED!

TO RUN A PROGRAM, THE USER SIMPLY TYPES THE DIRECT COMMAND "#=1" (GOTO 1). THIS CAUSES THE SYSTEM TO FIND THE LOWEST NUMBERED LINE AND BEGIN EXECUTING THERE. IF IT IS DESIRED TO BEGIN EXECUTING AT SOME OTHER LINE NUMBER, SAY 1000, SIMPLY TYPE "#=1000", OR WHATEVER LINE NUMBER IS DESIRED.

COMMENTS MAY BE INSERTED ON ANY LINE BY PRECEEDING THEM WITH A RIGHT PARENTHESES. THIS SYMBOL MUST FOLLOW THE EXPRESSION ON THE LINE IMMEDIATELY, WITH NO BLANKS IN BETWEEN. THIS CAUSES THE SYSTEM TO STOP EVALUATING THE LINE AND GO ON TO THE NEXT LINE. IF A LINE IS TO CONTAIN ONLY A COMMENT, THE FIRST CHARACTER ON THE LINE MUST BE A RIGHT PARENTHESES.

THERE ARE NO ERROR MESSAGES IN VTL-2. IF AN EXPRESSION IS WRONG THE RESULTS OF EVALUATING THAT EXPRESSION WILL ALSO BE WRONG. IN OTHER WORDS, VTL-2 ASSUMES THAT YOU KNOW WHAT YOU ARE DOING, AND WILL DO ITS BEST TO EXECUTE ANY STATEMENT THAT YOU GIVE IT. THIS LEAVES WIDE LATITUDE FOR TRYING VARIOUS PROGRAMMING "TRICKS", BUT ALSO LEAVES THE RESPONSIBILITY OF VERIFYING PROGRAM ACCURACY WITH THE PROGRAMMER.

VTL-2 PROGRAMS MAY BE SAVED ON CASSETE OR PAPER TAPE. FOR PAPER TAPE, SIMPLY TYPE "0", AND PUNCH A LISTING. FOR CASSETTE OPERATION, THIS METHOD WILL NOT WORK, SINCE THE CASSETTE OPERATES TOO FAST TO ALLOW VTL-2 TO FINISH PROCESSING EACH LINE IN TIME. CASSETTE TAPES MAY BE MADE USING THE MOTOROLA FORMAT TAPE PUNCH PROGRAM IN THE 680 PROGRAMMING MANUAL. BEFORE THIS IS DONE, YOU MUST NOTE THE VALUE OF THE & VARIABLE (TYPE ?=&, AND WRITE IT DOWN) SO THAT IT CAN BE SET BACK TO THIS VALUE AFTER READING THE PROGRAM BACK IN. AFTER LOADING FROM CASSETTE (MONITOR L COMMAND) YOU MUST TYPE &= AND THE VALUE NOTED. WITHOUT THIS, THE PROGRAM MAY NOT EXECUTE CORRECTLY.

SAMPLE OF PROGRAMMING

WHEN STARTING:
A. LIFT THE RUN-HALT SWITCH
B. TURN THE COMPUTER ON
C. LIFT THE RESET SWITCH
D. TURN THE RUN-HALT SWITCH TO RUN
E. THE MONITOR SHOULD REPLY WITH A PERIOD (.)
F. TYPE IN J FC00
G. VTL-2 SHOULD REPLY "OK"

```
OK                              VTL-2 PROMPT
*=1024                          SET MEMORY SIZE TO 1024 BYTES

OK                              VTL-2 PROMPT
$=264                           RESET PROGRAM POINTER

OK                              VTL-2 PROMPT
10 A=0                          SET A EQUAL TO ZERO
20 B=1                          SET B EQUAL TO ONE
30 ?=A                          PRINT THE VALUE OF A
40 ?="! = ";                    PRINT "FACTORIAL EQUALS"
50 ?=B                          PRINT THE VALUE OF B
60 ?=""                         PRINT A CARRIAGE RETURN LINE FEED
70 A=A+1                        INCREMENT A
80 B=A*B                        MULTIPLY B TIMES A
90 #=A<9*30                     IF A IS LESS THAN 9 THEN GOTO STEP # 30
#=10                            EXECUTE PROGRAM
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320

OK                              VTL-2 PROMPT SAYING IT'S DONE.
```

LIST OF NEW FEATURES OF VTL-2 COMPARED TO VTL-1

#1 11 MORE VARIABLES
#2 1 ARRAY
#3 COMPUTED RETURN ADDRESS
#4 END OF PROGRAM POINTER
#5 END OF MEMORY POINTER
#6 RANDOM NUMBER GENERATOR
#7 SINGLE CHARACTER STRING INPUT
#8 SINGLE CHARACTER STRING OUTPUT
#9 MACHINE LANGUAGE SUBROUTINES
#10 COMPUTED LINE INPUTS
#11 COMPUTED ARRAY
#12 FASTER OPERATION
#13 INPUT-OUTPUT COMPATABILITY ON LIST
#14 FASTER LINE INSERTION
#15 NO DOUBLE CARRIAGE RETURN ON LINE INPUTS
#16 "OK" PROMPT
#17 NO PROMPT ON LINE INSERTION
#18 NULLS AND CONTROL CHARACTERS EQUAL OR LESS THAN CARRIAGE-RETURN ARE NOT INSERTED INTO LINES
#19 CONTROL-C STOPS EXECUTION WHEN PRINTING
#20 ANY KEY PRESSED SUSPENDS OPERATION WHEN PRINTING
" " " RESUMES " " "
#21 ALL INTERUPT VECTORS ARE PRESERVED
#22 ON INITIALIZATION, SETTING THE LINE NUMBER EQUAL TO ZERO IS NO LONGER NEEDED
#23 MULTIPLE OPERATIONS ARE ALLOWED ON A VALUE ON ONE LINE
#24 SINGLE AND MULTIPLE PARENTHESES ARE ALLOWED
#25 PRINT STATEMENTS MAY BE WITH OR WITHOUT A FOLLOWING CARRIAGE-RETURN LINE FEED
#26 NO LEADING OR TRAILING SPACES ARE ADDED TO A PRINTED NUMBER
#27 # STANDS FOR PRESENT LINE NUMBER NOT PRESENT LINE NUMBER + 1
#28 MORE UNDERSTANDABLE SYSTEM VARIABLES, EI. # STANDS FOR LINE #
#29 33 MORE BYTES ARE ALOTTED FOR PROGRAM SPACE
#30 PROGRAM TAKES UP 10% LESS SPACE IN MEMORY
#31 UP TO 72 CHARACTERS ARE ALLOWED PER LINE

LIST OF FEATURES

VARIABLES

VARIABLE	MEANING
A-Z	COMMON VARIABLES USE FREELY FOR STORING VALUES

SYSTEM VARIABLES

!	RETURN ADDRESS POINTS TO THE LINE # AFTER THE LAST #= STATEMENT
"	POINTER FOR LITERAL PRINT STATEMENTS
#	LINE NUMBER
$	SINGLE CHARACTER STRING (INPUT OR OUTPUT)
%	REMAINDER AFTER THE LAST DIVIDE OPERATION
&	POINTS TO THE LAST BYTE OF PROGRAM
'	RANDOM NUMBER
(	SETS START OF PARENTHESIZED EXPRESSION
)	END SETS END OF LINE SETS END OF PARENTHESIZED EXPRESSION SETS END OF ARRAY DESCRIPTION USED ALSO FOR REMARK STATEMENT
*	POINTS TO END OF MEMORY
>	MACHINE LANGUAGE SUBROUTINE
?	PRINT STATEMENT WHEN ON LEFT OF EQUAL SIGN INPUT STATEMENT WHEN ON RIGHT OF EQUAL SIGN
:	DEFINES START OF ARRAY DESCRIPTION
;	WHEN FOLOWING A LITERAL PRINT STATEMENT, SAYS DO NOT PRINT CARRIAGE-RETURN LINE-FEED
. ~=; +, </↑][	MAY BE USED FREELY AS STANDARD VARIABLES BUT USE IS NOT RECOMMENDED FOR LEGIBILTY REASONS

OPERATORS

+	ADD TO PREVIOUS VALUE
-	SUBTRACT FROM PREVIOUS VALUE
*	MULTIPLY TIMES PREVIOUS VALUE
/	DIVIDE PREVIOUS VALUE BY
=	IS PREVIOUS VALUE EQUAL TO (YES = 1, NO = 0)
<	IS PREVIOUS VALUE LESS THAN (YES = 1, NO = 0)
>	IS PREVIOUS VALUE EQUAL TO OR GREATER THAN (Y=1,N=0)

THE DEFAULT OPERATOR IS THE LESS THAN TEST.

HURKLE

```
100 ?=""
110 ?="A HURKLE IS HIDING ON A"
120 ?="10 BY 10 GRID.  HOMEBASE"
130 ?="ON THE GRID IS POINT 00"
140 ?="AND A GRIDPOINT IS ANY"
150 ?="PAIR OF WHOLE NUMBERS"
160 ?="TRY TO GUESS THE HURKLE'S"
170 ?="GRIDPOINT.  YOU GET 5 GUESSES"
180 ?=""
190 R='/100*0+%
200 A=R/10
210 B=%
220 K=1
230 ?="GUESS #";
240 ?=K
250 ?="  ?";
260 X=?/10
270 Y=%
280 ?=""
290 #=X*10+Y=R*540
300 K=K+1
310 #=K=6*440
320 ?="GO ";
330 #=Y=B*370+(Y<B*360)
340 ?="SOUTH";
350 #=370
360 ?="NORTH";
370 #=X=A*410+(X<A*400)
380 ?="WEST";
390 #=410
400 ?="EAST";
410 ?=""
420 ?=""
430 #=230
440 ?=""
450 ?="SORRY THAT'S 5 GUESSES"
460 ?="THE HURKLE IS AT ";
470 ?=A
480 ?=B
490 ?=""
500 ?=""
510 ?="LETS PLAY AGAIN."
520 ?="HURKLE IS HIDING"
530 #=180
540 ?="YOU FOUND HIM IN ";
550 ?=K
560 ?=" GUESSES"
570 #=490
```

TIME OF DAY DIGITAL CLOCK

FOR 300 BUAD TERMINALS

```
10 ?="HOUR ?";
20 H=?
30 ?="MINUTE ?";
40 M=?
50 ?="SECOND ?";
60 S=?
70 ?="READY"
80 A=$
90 S=S+1
100 M=S/60+M
110 S=%
120 H=M/60+H
130 M=%
140 H=H/24*0+%
150 ?="TIME:  ";
160 ?=H/10
170 ?=%
180 ?=":";
190 ?=M/10
200 ?=%
210 ?=":";
220 ?=S/10
230 ?=%
240 $=13
250 A=B
260 T=31
270 T=T-1
280 #=T=0*90
290 #=270
```

FOR 110 BAUD TERMINALS

```
10 ?="HOUR ?";
20 H=?
30 ?="MINUTE ?";
40 M=?
50 ?="SECOND ?";
60 S=?
70 ?="READY"
80 A=$
90 S=S+1
100 M=S/60+M
110 S=%
120 H=M/60+H
130 M=%
140 H=H/24*0+%
150 ?=H/10
160 ?=%
170 ?=":";
180 ?=M/10
190 ?=%
200 ?=":";
210 ?=S/10
220 ?=%
230 $=13
240 A=B
250 A=B
260 A=B
270 A=B+B
280 T=14
290 T=T-1
300 #=T=0*90
310 #=290
```

FACTORIALS

CALCULATES FACTORIALS UNTIL IT RUNS OUT OF MEMORY
FOR 1K OF MEMORY THIS IS ABOUT 208!

```
10 A=1
20 L=2
30 :1)=1
40 I=2
50 :I)=0
60 I=I+1
70 #=L>I*50
80 ?=""
90 ?=""
100 ?=A
110 ?="! ="
120 ?=""
130 I=L+1
140 I=I-1
150 #=:I)=0*140
160 ?=:I)
170 I=I-1
180 #=I=0*220
190 ?=:I)/10
200 ?=%
210 #=170
220 A=A+1
230 I=1
240 C=0
250 X=:I)
260 :I)=A*X
270 #=:I)<X*320
280 :I)=:I)+C
290 C=:I)/100
300 :I)=%
310 I=I+1
320 #=L>I*250
330 #=C=0*80
340 L=L+1
350 #=*-&/2<L*380
360 :I)=C
370 #=290
```

WEEKDAY

```
10 #=440
20 ?="DAY OF THE WEEK"
30 ?=""
40 ?="MONTH?  ";
50 M=?
60 #=M>13*40
70 #=M=0*40
80 ?=DAY OF MONTH?  ";
90 D=?
100 ?="YEAR?  "
110 Y=?
120 #=Y>1800*230
130 #=Y<100*150
140 #=70
150 ?=""
160 ?="IS THAT 19";
170 ?=Y
180 ?="?  ";
190 K=$
200 #=K=89=0*70
210 ?="ES"
220 Y=Y+1900
230 C=Y/100
240 Y=%
250 #=Y/4*0+%=0*280
260 :1)=6
270 :2)=2
280 W=Y/4+Y+D+:M)+(2*(C=18))/7*0+%
290 #=300+(20*W)
300 ?="SUN";
310 #=430
320 ?="MON";
330 #=430
340 ?="TUES";
350 #=430
360 ?="WEDNES";
370 #=340
380 ?="THURS";
390 #=430
400 ?="FRI";
410 #=430
420 ?="SATUR";
430 ?="DAY"
440 :1)=0
450 :2)=3
460 :3)=3
470 :4)=6
480 :5)=1
490 :6)=4
500 :7)=6
510 :8)=2
520 :9)=5
530 :10)=0
540 :11)=3
550 :12)=5
560 #=20
```

STARSHOOTER

```
10 I=0
20 I=I+1
30 :I)=46
40 #=I<41*20
50 :25)=42
60 I=8
70 J=1
80 $=I-1/7+64
90 ?=" - ";
100 S=I+J
110 $=:S)
120 J=J+1
130 #=J=6*160
140 ?="  ";
150 #=100
160 I=I+7
170 ?=""
180 ?=""
190 #=I<43*70
200 ?=""
210 ?="    1  2  3  4  5"
220 ?=""
230 ?="YOUR MOVE --";
240 I=42
250 I=I+1
260 :I)=$
270 #=:I)=13*320
280 #=:I)=3*580
290 #=:I)=95=0*250
300 I=I-1
310 #=260
320 A=:43)-64
330 ?=""
340 #=A>6*230
350 B=:44)-48
360 #=B>6*230
370 S=A*7+1+B
380 ?=""
390 #=:S)=42*420
400 ?="THAT'S NOT A STAR!"
410 #=230
420 :S)=46
430 C=S-7
440 #=520
450 C=S-1
460 #=520
470 C=S+1
480 #=520
490 C=S+7
500 #=520
510 #=60
520 ↑=!
530 #=:C)=42*560
540 :C)=42
550 #=↑
560 :C)=46
570 #=↑
```

OBJECT OF THE GAME IS TO CHANGE THIS:

```
A - . . . . .
B - . . . . .
C - . . * . .
D - . . . . .
E - . . . . .

    1 2 3 4 5
```

TO THIS:

```
A - * * * * *
B - *       *
C - *       *
D - *       *
E - * * * * *

    1 2 3 4 5
```

DON'T LOSE YOUR AT!

BY

ED VERNER

ADAPTED TO VTL-2 BY GARY SHANNON

(A GAME SIMILAR TO "BAGLES")

THE OBJECT OF THE GAME IS TO GUESS THE SECRET NUMBER PICKED BY THE COMPUTER. THE NUMBER HAS THREE DIGITS, NO ZEROES, AND NO DIGIT IS REPEATED. AFTER YOU TYPE YOUR GUESS, THE COMPUTER WILL PRINT AN "IT" FOR EVERY CORRECT DIGIT IN THE WRONG POSITION, AND AN "AT" FOR EVERY CORRECT DIGIT IN THE RIGHT POSITION. YOU WIN WHEN YOU GET 3 "AT'S".

EACH TIME THAT YOU GUESS INCORRECTLY, YOU LOSE 5% OF THE POINTS YOU HAVE LEFT.

```
10 T=0
20 L=0
30 ?="DON'T LOSE YOUR 'AT'"
40 X='/9*0+%+1
50 Y='/9*0+%+1
60 #=X=Y*40
70 Z='/9*0+%+1
80 #=X=Z*40
90 #=Y=Z*40
100 ?="I'VE GOT A NUMBER. "
105 L=L+1
110 P=10000
120 ?=""
130 ?="YOU HAVE ";
140 ?=P/100
150 ?=". ";
160 ?=%/10
170 ?=%
180 ?=" POINTS LEFT"
190 ?=""
200 ?="WHAT'S YOUR GUESS? -- ";
210 G=?
220 A=G/100
230 B=%/10
240 C=%
260 S=600
270 #=A=Y*S
280 #=A=Z*S
290 #=B=X*S
300 #=B=Z*S
310 #=C=X*S
320 #=C=Y*S
330 K=0
340 S=620
350 #=A=X*S
360 #=B=Y*S
370 #=C=Z*S
380 #=K<3*580
390 ?=""
400 ?="YOU WIN ";
410 ?=P/100
420 ?=". ";
430 ?=%/10
440 ?=%
450 ?=" POINTS FOR A TOTAL OF ";
460 T=T+P
490 ?=T/100
500 ?=". ";
510 ?=%/10
520 ?=%
540 ?=" POINTS IN ";
550 ?=L
560 ?=" GAMES. "
570 #=30
580 P=P/20*19
590 #=120
600 ?="IT ";
610 #=!
620 ?="AT ";
630 K=K+1
640 #=!
```

********** HAVE FUN! **********

CRAPS!

```
10 T=100
20 $=22
30 ?="CRAPS!"
40 ?=""
50 ?="HOW MUCH DO YOU BET? - ";
60 B=?
70 #=B=0*90
80 ?="GOOD LUCK!"
90 #=B=0*480
100 #=T>B*160
110 ?="TOO MUCH!"
120 ?="YOU HAVE $";
130 ?=T
140 ?=" LEFT."
150 #=40
160 ?=""
170 ?="ROLL-";
180 A=?
190 $=22
200 ?="FIRST ROLL: ";
210 #=500
220 #=R=7*360
230 #=R=11*360
240 #=R<4*390
250 #=R=12*390
260 ?=""
270 ?=R
280 ?=" IS YOUR POINT"
290 P=R
300 ?="ROLL-";
310 A=$
320 #=500
330 #=R=7*390
340 #=R=P*360
350 #=300
360 ?="YOU WIN"
370 T=T+B
380 #=120
390 T=T-B
400 ?="YOU LOSE"
410 #=T=0*430
420 #=120
430 ?="YOU ARE BUSTED!"
440 ?="MOVE OVER AND LET THE NEXT"
450 ?="SUCKER TRY."
460 ?=""
470 #=10
480 ?="BE SERIOUS"
490 #=40
500 R=⁄/6*0+%+1
510 ?=R
520 X=X+11213
530 ?=" AND ";
540 S=⁄/6*0+%+1
550 X=X*56001
560 ?=S
570 ?="  (";
580 R=R+S
590 ?=R
600 ?=")"
610 #=!
```

CIPHER GAME

```
10 I=0
20 I=I+1
30 :I)=I+64
40 #=I<26*20
50 I=1
60 ?=""
70 M='/26*0+%+1
80 H=:M)
90 :M)=:I)
100 :I)=H
110 I=I+1
120 #=I<27*70
130 ?="TEXT?"
140 ?=""
150 I=27
160 :I)=$
170 #=:I)=13*220
180 #=:I)=95=0*200
190 I=I-2
200 I=I+1
210 #=160
220 ?=""
230 I=27
240 #=:I)<64*270
250 T=:I)-64
260 :I)=:T)
270 I=I+1
280 #=:I)>14*240
290 ?=""
300 ?="CODE:"
310 ?=""
320 I=27
330 $=:I)
340 #=:I)=13*370
350 I=I+1
360 #=330
370 ?=""
380 ?="SWITCH? - ";
390 A=$
400 B=$
410 #=B=64*370
420 I=27
430 #=:I)=A*490
440 #=:I)=B=0*460
450 :I)=A
460 I=I+1
470 #=:I)=13*290
480 #=430
490 :I)=B
500 #=460
```

PHRASE SORT

```
10 $=22
20 I=0
30 I=I+1
40 :I)=$
50 L=:I)=95*2
60 I=I-L
70 #=:I)>14*30
80 ?=""
90 I=1
100 K=I
110 J=K
120 #=:K)=32*160
130 #=:J)=32*150
140 #=:K)>:J)*160
150 J=K
160 K=K+1
170 #=:K)>14*120
180 H=:I)
190 :I)=:J)
200 :J)=H
210 I=I+1
220 #=:I)>14*100
230 I=0
240 I=I+1
250 $=:I)
260 #=:I)>14*240
270 ?=""
```

FACTORS OF A NUMBER
THIS VERSION FOR THE TVT

```
10 #=200
20 D=D+2/3*0+%=0*2+(D>3)+D+1
30 Q=N/D
40 #=Q<D*300
50 #=%>1*20
60 ?=""
70 ?=D
80 N=Q
90 Q=N/D
100 #=%>1*20
110 ?="↑";
130 P=1
140 N=Q
150 Q=N/D
160 P=P+1
170 #=%=0*140
180 ?=P
190 #=20
200 ?="NUMBER? ";
210 N=?
220 X=N
230 $=22
240 ?="      "
250 ?=N
260 ?=" IS ";
270 D=2
280 #=30
300 #=N=X*370
310 #=N=1*340
320 ?=""
330 ?=N
340 ?=""
350 ?="DONE"
360 #=200
370 ?="PRIME"
380 #=200
```

PRIMES
THIS VERSION FOR 32 CHAR TERMINAL

```
10 #=100
20 #=D>Q*150
30 D=D+2/3*0+%=0*2+(D>3)+D+1
40 Q=N/D
50 #=%>1*20
60 N=N+2/3*0+%=0*2+(N>3)+N+1
70 D=2
80 #=N<65533*40
90 #=N
100 $=28
101 #=102
102 ?="              PRIMES"
104 ?=""
106 ?="    ";
110 ?=1
115 ?=" ";
120 N=2
130 A=1
150 B=N
160 #=B>10000*200
170 $=32
180 #=B>1000*200
190 B*B*10
195 #=170
200 ?=N
205 ?=" ";
210 A=A+1
220 #=A<5*60
230 ?=""
240 A=0
250 #=60
```

LIFE
FAST VERSION

```
10 #=370
20 S=Y<F*Y+(Y=0*E)+(Y=F)-1*0+(X<Q*X+(X=0*0)+(X=Q))
25 :S)=:S)+2
30 X=X+1-(J<X*3)+(J-1=X*(Y=I))
40 Y=J-1=X+Y
50 #=I+1>Y*20
70 #=90
80 #=:I-1*0+J)/2*0+%*20
90 J=J+1-(0=J*0)
100 I=J=1+I
110 ?="";
120 X=J-1
130 Y=I-1
140 #=I<F*80
150 I=1
160 J=1
180 ?=""
190 P=0
200 K=I-1*0+J
210 :K)=:K)<5+(:K)>8)=0
220 P=P+:K)
230 $=:K)*10+32
240 J=J+1-(J=0*0)
250 #=1<J*200
260 ?=""
270 I=I+1
280 #=I<F*200
290 ?="GEN = ";
300 ?=G
310 G=G+1
320 ?="   POP = ";
330 ?=P
340 I=1
350 J=1
360 #=0<P*110+(P=0*650)
370 I=1
380 G=0
390 ?="SIZE? ";
400 0=?
410 Q=0+1
420 ?="BY?   ";
430 E=?
440 F=E+1
450 J=0*E+2
460 #=J*2+&)**390
470 :I)=0
480 I=I+1
490 #=J>I*470
495 #=631
500 I=1
510 ?=""
520 J=1
530 #=I>10*550
540 ?=" ";
550 ?=I
560 ?=" ";
```

```
580 :I-1*O+J)=L=32+(L=13)+(L=95)+(L=64)=0*6
590 J=J+1-(L=95*2)
600 #=L=13*620+(L=64*510)
610 #=J<Q*570
620 I=I+1
625 #=I<F*510
626 #=631
627 #=150
631 $=22
632 $=18
633 $=32
634 $=18
635 $=22
636 $=18
637 $=32
638 $=18
640 #=!
```

THIS PROGRAM TAKES AT LEAST 2 K OF MEMORY TO OPERATE SATISFACTORILY.

THIS VERSION WAS WRITTEN FOR THE SWTP TVT BUT WILL RUN ON ANY NORMAL TERMINAL. FOR BEST RESULTS (ON THE TVT) TRY FOR A 31 BY 15 MATRIX.

```
                    *MINI-TREK*
                BY FRANK MCCOY 1/7/77

10 ?=""                              PRINT A CARRIAGE RETURN-LINE FEED
20 ?="          *MINI-TREK*"         PRINT HEADING
30 W='/2500+10                       SETUP NUMBER OF KLINGONS (10-36)
40 D=0-('/6000+31*W/19)              SETUP NUMBER OF STARDATES
50 L=10000                           SET INITIAL ENERGY TO 10000
60 X=0                               INITIALIZE LOOP COUNTER
70 S=10                              POINT STARBASE OUTSIDE OF QUADRANT
80 T=10                                "      "        "    "     "
90 A=0                               INITIALIZE KLINGON COUNTER
100 X=X+1                            POINT TO NEXT SECTOR IN QUADRANT
110 :X)=0                            CLEAR SECTOR
120 #=X<64*100                       HAVE ALL 64 SECTORS BEEN CLEARED?
130 X=1                              RESET LOOP COUNTER
140 #='/13*0+%>2*170                 PROB.  OF 2 IN 13 OF A STAR OR KLINGON
150 :X)='/5*0+%=0*(A<W)+1            PROB OF 1 IN 5 OF STAR BEING KLINGON
160 A=:X)=2+A                        IF KLINGON THEN INCREMENT COUNTER
170 X=X+1                            INCREMENT SECTOR COUNTER
180 #=X<65*140                       SEE IF ALL 64 SECTORS HAVE BEEN SETUP
190 X='/64*0+%+1                     POSITION ENTERPRISE AT RANDOM
200 :X)=3
210 E=X-1/8                          SAVE SECTOR COORDINATES
220 F=%+1
230 #='>20000*290                    PROB.  OF A STARBASE APPROX 1 IN 4
240 J='/64*0+%+1                     POSITION STARBASE AT RANDOM
250 #=J=X*240                        DO IT AGAIN IF ENTERPRISE IN SAME PLACE
260 :J)=4
270 S=J-1/8                          SAVE STARBASE COORDINATES
280 T=%+1
290 C=S<E*(E-S)+(E<S*(S-E))          FIND OUT HOW CLOSE THE ENTERPRISE
300 G=T<F*(F-T)+(F<T*(T-F))          IS TO A STARBASE
310 Q=C<2*(G<2)                      IF CLOSE ENOUGH ENTERPRISE IS DOCKED
320 D=D+1                            INCREMENT STARDATE
340 L=Q*10000+(Q=0*L)*(L<10001)      SET ENERGY IF DOCKED OR ENERGY IS NEG.
350 ?=""                             PRINT CRLF
370 ?="##########"                   PRINT TOP BORDER OF SCAN
380 X=1                              INITIALIZE LINE COUNT OF SCAN
390 K=0                              SET UP TO COUNT KLINGONS
400 J=1                              INITIALIZE COLLUMN COUNT
420 ?="#";                           PRINT LEFT BORDER
430 C=:X-1*8+J)                      FIND OUT WHATS IN THAT SECTOR
440 $=C*14+32+(C=2*15)-(C=3*5)-(C=4*22)   PRINT SPACE,.,E,B,OR K
450 K=C=2+K                          INCREMENT IF KLINGON
460 J=J+1                            NCREMENT SECTOR
470 #=J<9*430                        IF NOT LAST IN ROW THEN GET NEXT
480 ?="#   ";                        PRINT RIGHT BORDER
490 #=X<4*750                        FIRST FOUR LINES BLANK
500 #=X>5*(X*30+400)                 PRINT THE APPROPRIATE DATA FOR EACH LINE
510 ?="SECTOR     ";                 PRINT THE SECTOR NUMBER
520 ?=E+1
530 ?=F
540 #=750
550 ?="STARDATE   ";                 PRINT THE STARDATE
560 ?=D
570 #=750
580 ?="ENERGY     ";                 PRINT THE ENERGY REMAINING
590 ?=L
```

```
600 #=750
610 ?="KLINGONS  ";              PRINT THE # OF KLINGONS REMAINING
620 ?=W
630 #=750
640 ?="CONDITION ";              PRINT THE CONDITION (RED, GREEN, ETC.)
650 #=Q*690                      IF DOCKED
660 #=K=0*710                    IF NO KLINGONS IN VICINITY
670 ?="RED";                     IF KLINGONS ARE PRESENT
680 #=750
690 ?="DOCKED";                  IF NEXT TO A STARBASE
700 #=750
710 #=L<2000*740                 IF ENERGY IS LOW GOTO 740
720 ?="GREEN";                   IF NO KLINGONS IN QUADRANT
730 #=750
740 ?="YELLOW";                  IF ENERGY IS LOW
750 ?=""                         PRINT A CRLF
760 X=X+1                        INCREMENT LINE NUMBER
770 #=X<9*400                    IF NOT LAST LINE THEN REPEAT
780 ?="##########"               PRINT BOTTOM BORDER OF SCAN
790 ?=""                         PRINT CRLF
800 #=K=0*840                    IF NO KLINGONS PRESENT THEN SKIP NEXT
810 H='/250*K                    FIND OUT HOW HARD YOU GOT ZAPPED
820 ?=H                          PRINT THE VALUE
830 ?=" UNIT HIT FROM KLINGONS"  PRINT MESSAGE
835 L=L-H                        SUBTRACT VALUE OF HIT FROM ENERGY LEFT
840 #=W=0*1190                   IF NO KLINGONS LEFT THEN YOU WON
850 #=D=0+(:E*8+F)=0)+(L-1>10000)>1*1210   SEVERAL WAYS TO LOSE
860 ?="COMMAND? ";               PROMPT
870 A=?                          INPUT THE COMMAND #
880 #=A-1>3*860                  IF ILLEGAL COMMAND THEN REPEAT
890 #=A>2*(A*20+940)             GOTO APPRORIATE COMMAND ROUTINE
900 #=1060                       GOTO THE SECTOR SUBROUTINE
910 #=:O)>1*900                  YOU CAN'T JUMP WHERE SOMETHING IS
920 :E*8+F)=0                    YOU ARE NO LONGER THERE
930 :O)=3                        YOUR NEW LOCATION
940 E=M                          SAVE NEW COORDINATES
950 F=N                            "   "
960 L=L-G                        SUBTRACT ENERGY NEEDED TO MOVE
970 #=290                        PRINT OUT NEW MAP
980 L=L-('/250+300)              LESS ENERGY TO MOVE TO A NEW QUADRANT
990 #=60                         SETUP NEW QUADRANT
1000 #=1060                      GOTO THE SECTOR SUBROUTINE
1010 #='<7800*1040               RANDOM MISS
1020 W=W-(:O)=2)                 IF ITS A KLINGON THEN ONE LESS KLINGON
1030 :O)=0                       WHATEVER IT WAS IT'S DEAD
1040 L=L-(G*2)                   SUBTRACT ENERGY NEEDED TO SHOOT
1050 #=290                       PRINT OUT NEW MAP
1060 ?="SECTOR? ";               SECTOR SUBROUTINE
1070 M=?/10-1                    INPUT COORDINATES
1080 N=%
1090 O=M*8+N                     FIND WHERE THEY ARE IN THE ARRAY
1100 #=O-1>64*840                RETURN TO COMMAND IF ILLEGAL COORDINATES
1110 C=M-E*(M-E)+(N-F*(N-F))*100   SUM OF SQUARES OF TWO DISTANCES
1120 R=!                         SAVE RETURN ADDRESS
1130 G=C/10                      SETUP FOR SQUARE ROOT
1140 #=G=0*R                     RETURN IF ZERO DISTANCE
1150 J=G                         SAVE APPROXIMATION
1160 G=C/G+G/2                   CALCULATE SQUARE ROOT
1170 #=G<J*1150                  IF NEW APPROXIMATION IS BETTER THEN CONT
1180 #=R                         RETURN
```

```
1190 ?="YOU WIN!!"              IF YOU WON
1200 #=1220                     SKIP NEXT
1210 ?="YOU LOSE!"              IF YOU LOST
```

THIS MINI VERSION OF STARTREK HAS ONLY THREE COMMANDS:

1. MOVE TO A DIFFERENT SECTOR
2. MOVE TO A DIFFERENT QUADRANT
3. FIRE AT A GIVEN SECTOR

NOTES: THE FURTHER YOU MOVE OR THE FURTHER AWAY YOUR TARGET THE MORE ENERGY IT TAKES

IF YOU RUN OUT OF STARDATES OR ENERGY YOU LOSE

IF YOU ZAP YOURSELF, YOU LOSE

NOT ALL QUADRANTS HAVE STARBASES IN THEM

A SAMPLE PRINTOUT LOOKS LIKE THIS:

```
##########
#.     .. #
#   K    #
#  E   .  #
#     .   #  SECTOR    33
#KK   .   #  STARDATE  65437
#         #  ENERGY    5736
#   B   .#  KLINGONS  15
#..       #  CONDITION RED
##########

586 UNIT HIT FROM KLINGONS
COMMAND?
```

TIC TAC TOE
BY FRANK MCCOY 1/17/77

```
1000 Q=0
1010 H=0
1020 J=0
1030 I=0
1040 U=0
1050 S=1
1060 I=I+1
1070 :I)=I+48
1080 #=I<9*1060
1090 #=1680
1100 ?=""
1110 ?=""
1120 U=1
1130 ?="YOUR MOVE - ";
1140 M=$-48
1150 #=3-48=M*2040
1160 ?=""
1170 #=M=0*1030
1180 #=9<M*1220
1190 #=:M)<65*1240
1200 ?="SOMEBODY ALREADY THERE"
1210 #=1130
1220 ?="ILLEGAL MOVE!"
1230 #=1130
1240 :M)=88
1250 #=1680
1260 X=1
1270 L=0
1280 K=0
1290 N=1
1300 A=N
1310 B=X+N
1320 C=2*X+N
1330 #=:A)=:B)+(:A)=:C))=2*1880
1340 #=:A)=:B)+(:C)<65)=2*1410
1350 D=A
1360 A=B
1370 B=C
1380 C=D
1390 #=A=N*1440
1400 #=1340
1410 #=K>1*(:L)=79)*1440
1420 L=A
1430 K=C
1440 #=X=4+(2*X+N=9+(X=2))*30+#
1450 N=X=1*2+1+N
1460 #=1300
1470 X=X+1
1480 N=X=2*2+1
1490 #=1300
1500 #=K>1*1620
1510 I=0
1520 P=0
1530 I=I+1
1540 P=:I)>65+P
1550 #=I<9*1530
1560 #=P=9*1950
```

```
1570 K=M
1575 X=0
1580 K=:5)>65*(K+5/2*0+%+('/16384*2+1)+K/10*0+%/9*0+%+1)
1590 K=:5)<65*5+K
1600 X=X+1
1605 S=X>9+S
1610 #=:K)>65*1580
1620 :K)=79
1625 S=:5)=79*M+5/2*0+%
1630 #=L>1*(:L)=79)*1910
1640 !=1100
1650 ?=""
1660 ?="MY MOVE - ";
1670 ?=K
1680 ?=""
1690 ?=""
1700 R=!
1710 I=1
1720 ?="   I   I"
1730 ?=" ";
1740 #=U*1770
1750 $=:I)
1760 #=1780
1770 $=:I)<65*32+(:I)>65*:I))
1780 P=:I)>65+P
1790 I=I+1
1800 #=I/3*0+%=1*1830
1810 ?=" I";
1820 #=1730
1830 ?=""
1840 ?="   I   I"
1850 #=I=10*R
1860 ?="-----------"
1870 #=1720
1880 ?="!!!YOU WIN!!!"
1890 H=H+1
1900 #=1970
1910 #=1650
1920 ?="YOU LOSE"
1930 J=J+1
1940 #=1970
1950 ?="CAT GOT THIS ONE. "
1960 Q=Q+1
1970 ?="PLAY AGAIN? ";
1980 S=$
1990 #=S=89=0*2020
2000 ?="ES"
2010 #=1030
2020 #=S=78=0*1970
2030 ?="O"
2040 ?=""
2050 ?="  I WON ";
2060 ?=J
2070 ?=" GAMES"
2080 ?="YOU WON ";
2090 ?=H
2100 ?=" GAMES"
2110 ?="WE TIED ";
2120 ?=Q
2130 ?=" GAMES"
```

CASSETTE SAVER FOR 680

```
65000 ?="# OF NULLS? ";
65010 N=?
65020 ?="GO -";
65030 Z=$
65040 Z=&
65050 U=0
65060 I=132
65070 #=65230
65080 &=U
65090 X=:I)
65100 &=Z
65110 ?=X
65120 I=U+2/2+I
65130 U=%
65140 &=U
65150 L=:I)/256
65160 &=Z
65170 #=L=0*65210
65180 $=L
65190 U=U+1
65200 #=65140
65210 ?=""
65220 U=U+1
65230 M=N
65240 #=M=0*65280
65250 M=M-1
65260 $=0
65270 #=65240
65280 #=I*2+U<(Z-292)*65080
```

TO SAVE THE SAVER ITSELF, ADD ONE MORE LINE

```
65290 &=565
```

THEN TYPE:

```
&=848
#=1
```

THE ROUTINE WILL THEN SAVE ITSELF.

TIC TAC TOE
BY FRANK MCCOY 1/17/77
THIS VERSION FOR THE SWTP TVT

```
1000 Q=0
1010 H=0
1020 J=0
1030 I=0
1035 $=22
1040 U=0
1050 S=1
1060 I=I+1
1070 :I)=I+48
1080 #=I<9*1060
1090 #=1670
1100 $=19
1110 ?=""
1120 U=1
1130 ?="YOUR MOVE - ?";
1135 $=18
1140 M=$-48
1150 #=3-48=M*2040
1160 ?=""
1170 #=M=0*1030
1180 #=9<M*1220
1190 #=:M)<65*1240
1200 ?="SOMEBODY ALREADY THERE";
1210 #=1100
1220 ?="ILLEGAL MOVE!";
1230 #=1100
1240 :M)=88
1250 #=1690
1252 $=19
1254 ?="  MY MOVE -  ";
1256 $=18
1260 X=1
1270 L=0
1280 K=0
1290 N=1
1300 A=N
1310 B=X+N
1320 C=2*X+N
1330 #=:A)=:B)+(:A)=:C))=2*1880
1340 #=:A)=:B)+(:C)<65)=2*1410
1350 D=A
1360 A=B
1370 B=C
1380 C=D
1390 #=A=N*1440
1400 #=1340
1410 #=K>1*(:L)=79)*1440
1420 L=A
1430 K=C
1440 #=X=4+(2*X+N=9+(X=2))*30+#
1450 N=X=1*2+1+N
1460 #=1300
1470 X=X+1
1480 N=X=2*2+1
1490 #=1300
1500 #=K>1*1620
```

```
1510 I=0
1520 P=0
1530 I=I+1
1540 P=:I))65+P
1550 #=I<9*1530
1560 #=P=9*1950
1570 K=M
1575 X=0
1580 K=:5))65*(K+S/2*0+%+('/16384*2+1)+K/10*0+%/9*0+%+1)
1590 K=:5)<65*5+K
1600 X=X+1
1605 S=X>9+S
1610 #=:K))65*1580
1620 :K)=79
1625 S=:5)=79*M+S/2*0+%
1630 #=L>1*(:L)=79)*1910
1640 !=1100
1660 ?=K
1670 ?=""
1680 ?=""
1690 ?=""
1700 R=!
1710 I=1
1720 ?="   I   I"
1730 ?=" ";
1740 #=U*1770
1750 $=:I)
1760 #=1780
1770 $=:I)<65*32+(:I))65*:I))
1780 P=:I))65+P
1790 I=I+1
1800 #=I/3*0+%=1*1830
1810 ?=" I";
1820 #=1730
1830 ?=""
1840 ?="   I   I"
1850 #=I=10*R
1860 ?="-----------"
1870 #=1720
1880 $=19
1885 ?="!!!YOU WIN!!!"
1890 H=H+1
1900 #=1970
1910 #=1650
1920 $=19
1925 ?="YOU LOSE     "
1930 J=J+1
1940 #=1970
1950 $=19
1955 ?="CAT GOT THIS ONE. "
1960 Q=Q+1
1970 ?="PLAY AGAIN?  ";
1980 S=$
1990 #=S=89*1030
2020 #=S=78=0*1970
2030 ?="O"
2040 ?=""
2050 ?="  I WON ";
2060 ?=J
2070 ?=" GAMES"
```

```
2080 ?="YOU WON ";
2090 ?=H
2100 ?=" GAMES"
2110 ?="WE TIED ";
2120 ?=Q
2130 ?=" GAMES"
2140 ?=""
```

RENUMBER

680 VERSION

```
64000 A=#
64010 B=&
64020 C=#
64030 &=B
64040 ?="STARTING #? ";
64050 D=?
64060 ?="STEP SIZE?  ";
64070 E=?
64080 &=1
64090 G=131
64100 J=0
64110 H=#+1
64120 G=&+1/2+G
64130 &=%
64140 #=:G)>A*5*(C-A)+#
64150 #=D-1>(A-1)+(J>D)>1*C
64160 :G)=D
64170 &=&+1
64180 J=D
64190 D=D+E
64200 K=#+1
64210 &=&+1
64220 #=:G)>256*K
64230 #=H
64240 &=B
64250 ?="DONE"
```

8080 VERSION

```
64000 A=#
64010 B=&
64020 C=#
64030 &=B
64040 ?="STARTING #? ";
64050 D=?
64060 ?="STEP SIZE?  ";
64070 E=?
64080 &=1
64090 G=159
64100 J=0
64110 H=#+1
64120 G=&+1/2+G
64130 &=%
64140 #=:G)>A*5*(C-1)+#
64150 #=D-1>(A-1)+(J>D)>1*C
64160 :G)=D
64170 &=&+1
64180 J=D
64190 D=D+E
64200 K=#+1
64210 &=&+1
64220 #=:G)*256>1*K
64230 #=H
64240 &=B
64250 ?="DONE"
```

NOTE: THIS PROGRAM IS RELOCATABLE. I.E. IT CAN BE RENUMBERED AND IT WILL STILL RUN. HOWEVER, THE STEP SIZE BETWEEN PROGRAM STEPS MUST REMAIN CONSTANT OR LINE 64140 WILL NOT WORK RIGHT. ALSO, THE LARGEST NUMBER OF THE PROGRAM TO BE RENUMBERED MUST BE LESS THAN THE FIRST NUMBER OF THE RENUMBER PROGRAM.

```
 1  0000            /VTL-2
 2  0000            /V-3.6
 3  0000            /9-23-76
 4  0000            /BY GARY SHANNON
 5  0000            /& FRANK MCCOY
 6  0000            /COPYRIGHT 1976, THE COMPUTER STORE
 7  0000            /
 8  0000            /DEFINE LOCATIONS IN MONITOR
 9  0000            ORG FFX 0
10  FF00            INCH, STR 0
11  FF00            ORG FFX 24X
12  FF24            POLCAT, STR 0
13  FF24            ORG FFX 81X
14  FF81            OUTCH, STR 1
15  FF82            OUTS, STR 0
16  FF82            /
17  FF82            /SET ASIDE FOUR BYTES FOR USER
18  FF82            /DEFINED INTERUPT ROUTINE IF NEEDED
19  FF82            ORG 0 0
20  0000            ZERO, STR 4             /INTERUPT VECTOR
21  0004            AT, STR 2               /CANCEL & C-R
22  0006            /
23  0006            /GENERAL PURPOSE STORAGE
24  0006            VARS, STR 52            /VARIABLES (A-Z)
25  003A            BRAK, STR 2             /[
26  003C            SAVE10, STR 1           /BACK SLASH
27  003D            SV10+1, STR 1
28  003E            BRIK, STR 2             /]
29  0040            UP, STR 2               /↑
30  0042            SAVE11, STR 1
31  0043            SV11+1, STR 1
32  0044            /
33  0044            SAVE14,STR 2            /SPACE
34  0046            EXCL, STR 2             /!
35  0048            QUOTE, STR 2            /"
36  004A            DOLR, STR 1             /# 1ST HALF
37  004B            DOLR+1, STR 1           /# 2ND HALF
38  004C            DOLLAR, STR 2           /$
39  004E            REMN1, STR 1            /% 1ST HALF
40  004F            REMN2, STR 1            /% 2ND HALF
41  0050            AMPR, STR 1             /& 1ST HALF
42  0051            AMPR+1, STR 1           /& 2ND HALF
43  0052            QUITE, STR 1            /' 1ST HALF
44  0053            QUITE+1, STR 1          /' 2ND HALF
45  0054            PAREN, STR 2            /(
46  0056            PARIN, STR 2            /)
47  0058            STAR, STR 1             /* 1ST HALF
48  0059            STAR+1, STR 1           /* 2ND HALF
49  005A            PLUS, STR 2             /+
50  005C            COMA, STR 2             /,
51  005E            MINS, STR 2             /-
52  0060            PERD, STR 2             /.
53  0062            SLASH, STR 2            //
54  0064            /
55  0064            SAVE0, STR 2
56  0066            SAVE1, STR 1
57  0067            SV1+1, STR 1
58  0068            SAVE2, STR 1
59  0069            SV2+1, STR 1
```

```
 60  006A                SAVE3, STR 1
 61  006B                SV3+1, STR 1
 62  006C                SAVE4, STR 1
 63  006D                SV4+1, STR 1
 64  006E                SAVE5, STR 2
 65  0070                SAVE6, STR 1
 66  0071                SV6+1, STR 1
 67  0072                SAVE7, STR 2
 68  0074                SAVE8, STR 2
 69  0076                SAVE9, STR 1
 70  0077                SV9+1, STR 1
 71  0078                COLN, STR 2          /:
 72  007A                SEMI, STR 2          /;
 73  007C                LESS, STR 2          /<
 74  007E                EQAL, STR 2          /=
 75  0080                GRAT, STR 1          />
 76  0081                DECBUF-1, STR 1
 77  0082                /
 78  0082                DECBUF, STR 4
 79  0086                LASTD, STR 1
 80  0087                DELIM, STR 1
 81  0088                LINBUF, STR 73       /LINE LENGTH +1
 82  00D1                /
 83  00D1                ORG 0 F1X
 84  00F1                STACK, STR 0         /SPACE FOR MONITOR
 85  00F1                /                    /STORAGE
 86  00F1                ORG 1 0
 87  0100                MI, STR 4            /INTERUPT VECTORS
 88  0104                NMI, STR 4
 89  0108                PRGM, STR 0          /PROGRAM STARTS HERE
 90  0108                /
 91  0108                ORG FCX 0
 92  FC00                /
 93  FC00  8E 00 F1      START, LDSI STACK
 94  FC03  4F            CLRA
 95  FC04  CE FE FB      LDXI OKM
 96  FC07  8D 42         BSR STRGT
 97  FC09                /
 98  FC09  4F            LOOP, CLRA
 99  FC0A  97 4A         STAD DOLR
100  FC0C  97 4B         STAD DOLR+1
101  FC0E  BD FE A3      JSRX CVTLN
102  FC11  24 3B         BCC STMNT            /NO LINE# THEN EXEC.
103  FC13  8D 24         BSR EXEC
104  FC15  27 E9         BEQ START
105  FC17                /
106  FC17  8D 65         LOOP2, BSR FIND      /FIND LINE
107  FC19  27 E5         EQSTRT, BEQ START    /IF END THEN STOP
108  FC1B  EE 00         LDXN 0               /LOAD REAL LINE #
109  FC1D  DF 4A         STXD DOLR            /SAVE IT
110  FC1F  DE 42         LDXD SAVE11          /GET LINE
111  FC21  08            INX                  /BUMP PAST LINE #
112  FC22  08            INX                  /  "    "    "   "
113  FC23  08            INX                  /BUMP PAST SPACE
114  FC24  8D 13         BSR EXEC             /EXECUTE IT
115  FC26  27 0D         BEQ LOOP3            /IF ZERO, CONTINUE
116  FC28  DE 42         LDXD SAVE11          /FIND LINE #
117  FC2A  EE 00         LDXN 0               /GET IT
118  FC2C  9C 4A         CPXD DOLR            /HAS IT CHANGED?
119  FC2E  27 05         BEQ LOOP3            /IF NOT GET NEXT
```

```
120  FC30                  /
121  FC30  08              INX                    /INCREMENT OLD LINE#
122  FC31  DF 46           STXD EXCL              /SAVE FOR RETURN
123  FC33  20 E2           BRA LOOP2              /CONTINUE
124  FC35                  /
125  FC35  8D 5A           LOOP3, BSR FND3        /FIND NEXT LINE
126  FC37  20 E0           BRA EQSTRT             /CONTINUE
127  FC39                  /
128  FC39  DF 72           EXEC, STXD SAVE7       /EXECUTE LINE
129  FC3B  BD FD EC        JSRX VAR2
130  FC3E  08              INX
131  FC3F                  /
132  FC3F  A6 00           SKIP, LDAN 0           /GET FIRST TERM
133  FC41  8D 03           BSR EVIL               /EVALUATE EXPRESSION
134  FC43  DE 4A           OUTX, LDXD DOLR        /GET LINE #
135  FC45  39              RTS
136  FC46                  /
137  FC46  81 22           EVIL, CPAI :"          /IF " THEN BRANCH
138  FC48  26 4E           BNE EVALU
139  FC4A  08              INX
140  FC4B  7E FD 75        STRGT, JMPX STRNG      /TO PRINT IT
141  FC4E                  /
142  FC4E  DF 74           STMNT, STXD SAVE8      /SAVE LINE #
143  FC50  97 4A           STAD DOLR
144  FC52  D7 4B           STBD DOLR+1
145  FC54  DE 4A           LDXD DOLR
146  FC56  26 69           BNE SKP2               /IF LINE# <> 0
147  FC58                  /
148  FC58  CE 01 08        LDXI PRGM              /LIST PROGRAM
149  FC5B  9C 50           LST2, CPXD AMPR        /END OF PROGRAM
150  FC5D  27 BA           BEQ EQSTRT
151  FC5F  DF 42           STXD SAVE11            /LINE# FOR CVDEC
152  FC61  A6 00           LDAN 0
153  FC63  E6 01           LDBN 1
154  FC65  BD FD 20        JSRX PRNT2
155  FC68  DE 42           LDXD SAVE11
156  FC6A  08              INX
157  FC6B  08              INX
158  FC6C  BD FD 58        JSRX PNTMSG
159  FC6F  BD FE F2        JSRX CRLF
160  FC72  20 E7           BRA LST2
161  FC74                  /
162  FC74  DE 42           NXTXT, LDXD SAVE11     /GET POINTER
163  FC76  08              INX                    /BUMP PAST LINE#
164  FC77  08              LOOKAG, INX            /FIND END OF LINE
165  FC78  6D 00           TSTN 0
166  FC7A  26 FB           BNE LOOKAG
167  FC7C  08              INX
168  FC7D  39              RTS
169  FC7E                  /
170  FC7E  CE 01 08        FIND, LDXI PRGM        /FIND LINE #
171  FC81  DF 42           FND2, STXD SAVE11
172  FC83  9C 50           CPXD AMPR
173  FC85  27 10           BEQ RTS1
174  FC87  A6 01           LDAN 1
175  FC89  90 4B           SUAD DOLR+1
176  FC8B  A6 00           LDAN 0
177  FC8D  92 4A           SBAD DOLR
178  FC8F  24 04           BCC SET
179  FC91  8D F1           FND3, BSR NXTXT
```

```
180  FC93  20 EC      BRA FND2
181  FC95            /
182  FC95  86 FF      SET, LDAI FFX          /SET NOT EQUAL
183  FC97  39         RTS1, RTS
184  FC98            /
185  FC98  BD FD 80   EVALU, JSRX EVAL       /EVALUATE LINE
186  FC9B  37         PSHB
187  FC9C  36         PSHA
188  FC9D  DE 72      LDXD SAVE7
189  FC9F  BD FD FF   JSRX CONVP
190  FCA2  32         PULA
191  FCA3  C1 24      CPBI :$                /STRING?
192  FCA5  26 04      BNE AR1
193  FCA7  33         PULB
194  FCA8  7E FF 81   JMPX OUTCH             /THEN PRINT IT
195  FCAB            /
196  FCAB  C0 3F      AR1, SUBI :?           /PRINT?
197  FCAD  27 70      BEQ PRNT               /THEN DO IT
198  FCAF  5C         INCB                   /MACHINE LANGUAGE?
199  FCB0  33         PULB
200  FCB1  26 01      BNE AR2
201  FCB3  3F         SWI                    /THEN INTERUPT
202  FCB4            /
203  FCB4  A7 00      AR2, STAN 0            /STORE NEW VALUE
204  FCB6  E7 01      STBN 1
205  FCB8  DB 52      ADBD QUITE             /RANDOMIZER
206  FCBA  99 53      ACAD QUITE+1
207  FCBC  97 52      STAD QUITE
208  FCBE  D7 53      STBD QUITE+1
209  FCC0  39         RTS
210  FCC1            /
211  FCC1  8D BB      SKP2, BSR FIND         /FIND LINE
212  FCC3  27 18      BEQ INSRT              /IF NOT THERE
213  FCC5  EE 00      LDXN 0                 /THEN INSERT
214  FCC7  9C 4A      CPXD DOLR              /NEW LINE
215  FCC9  26 12      BNE INSRT
216  FCCB            /
217  FCCB  8D A7      BSR NXTXT              /SETUP REGISTERS
218  FCCD  9E 42      LDSD SAVE11            /FOR DELETE
219  FCCF            /
220  FCCF  9C 50      DELT, CPXD AMPR        /DELETE OLD LINE
221  FCD1  27 08      BEQ FITIT
222  FCD3  A6 00      LDAN 0
223  FCD5  36         PSHA
224  FCD6  08         INX
225  FCD7  31         INS
226  FCD8  31         INS
227  FCD9  20 F4      BRA DELT
228  FCDB            /
229  FCDB  9F 50      FITIT, STSD AMPR       /STORE NEW END
230  FCDD            /
231  FCDD  DE 74      INSRT, LDXD SAVE8      /COUNT NEW LINE LENGTH
232  FCDF  C6 03      LDBI 3
233  FCE1  6D 00      TSTN 0
234  FCE3  27 31      BEQ GOTIT              /IF NO LINE THEN STOP
235  FCE5  5C         CNTLN, INCB
236  FCE6  08         INX
237  FCE7  6D 00      TSTN 0
238  FCE9  26 FA      BNE CNTLN
239  FCEB            /
```

```
240  FCEB  4F        OPEN, CLRA              /CALCULATE NEW END
241  FCEC  DB 51      ADBD AMPR+1
242  FCEE  99 50      ACAD AMPR
243  FCF0  97 3C      STAD SAVE10
244  FCF2  D7 3D      STBD SV10+1
245  FCF4  D0 59      SUBD STAR+1
246  FCF6  92 58      SBAD STAR
247  FCF8  24 22      BCC RSTRT               /IF TOO BIG THEN STOP
248  FCFA  DE 50      LDXD AMPR
249  FCFC  9E 3C      LDSD SAVE10
250  FCFE  9F 50      STSD AMPR
251  FD00             /
252  FD00  08         INX                     /SLIDE OPEN GAP
253  FD01  09         SLIDE, DEX
254  FD02  E6 00      LDBN 0
255  FD04  37         PSHB
256  FD05  9C 42      CPXD SAVE11
257  FD07  26 F8      BNE SLIDE
258  FD09             /
259  FD09  9E 4A      DON, LDSD DOLR          /STORE LINE #
260  FD0B  AF 00      STSN 0
261  FD0D  9E 74      LDSD SAVE8              /GET NEW LINE
262  FD0F  34         DES
263  FD10             /
264  FD10  08         MOVL, INX               /INSERT NEW LINE
265  FD11  33         PULB
266  FD12  E7 01      STBN 1
267  FD14  26 FA      BNE MOVL
268  FD16             /
269  FD16  8E 00 F1   GOTIT, LDSI STACK
270  FD19  7E FC 09   JMPX LOOP
271  FD1C             /
272  FD1C  7E FC 00   RSTRT, JMPX START
273  FD1F             /
274  FD1F  33         PRNT, PULB              /PRINT DECIMAL
275  FD20  CE 00 82   PRNT2, LDXI DECBUF      /CONVERT TO DECIMAL
276  FD23  DF 6C      STXD SAVE4
277  FD25  CE FE 5B   LDXI PWRS10
278  FD28  DF 6E      CVD1, STXD SAVE5
279  FD2A  EE 00      LDXN 0
280  FD2C  DF 70      STXD SAVE6
281  FD2E  CE 00 70   LDXI SAVE6
282  FD31  BD FE 65   JSRX DIVIDE
283  FD34  36         PSHA
284  FD35  DE 6C      LDXD SAVE4
285  FD37  96 69      LDAD SV2+1
286  FD39  8B 30      ADAI :0
287  FD3B  A7 00      STAN 0
288  FD3D  08         INX
289  FD3E  DF 6C      STXD SAVE4
290  FD40  DE 6E      LDXD SAVE5
291  FD42  32         PULA
292  FD43  08         INX
293  FD44  08         INX
294  FD45  6D 01      TSTN 1
295  FD47  26 DF      BNE CVD1
296  FD49             /
```

```
297  FD49  CE 00 81   LDXI DECBUF-1
298  FD4C  63 05      COMN 5                  /ZERO SUPPRESS
299  FD4E  08         ZRSUP, INX
300  FD4F  E6 00      LDBN 0
301  FD51  C1 30      CPBI :0
302  FD53  27 F9      BEQ ZRSUP
303  FD55  73 00 86   COMX LASTD
304  FD58             /
305  FD58  4F         PNTMSG, CLRA            /ZERO FOR DELIM
306  FD59  97 87      STRTMS, STAD DELIM      /STORE DELIMITER
307  FD5B             /
308  FD5B  E6 00      OUTMSG, LDBN 0          /GENERAL PURPOSE PRINT
309  FD5D  08         INX
310  FD5E  D1 87      CPBD DELIM
311  FD60  27 05      BEQ CTLC
312  FD62  BD FF 81   JSRX OUTCH
313  FD65  20 F4      BRA OUTMSG
314  FD67             /
315  FD67  BD FF 24   CTLC, JSRX POLCAT       /POL FOR CHARACTER
316  FD6A  24 5B      BCC RTS2
317  FD6C  8D 04      BSR INCH2
318  FD6E  C1 03      CPBI 3                  /CONTROL-C?
319  FD70  27 AA      BEQ RSTRT
320  FD72             /
321  FD72  7E FF 00   INCH2, JMPX INCH
322  FD75             /
323  FD75  8D E2      STRNG, BSR STRTMS       /PRINT STRING LITERAL
324  FD77  A6 00      LDAN 0
325  FD79  81 3B      CPAI :;
326  FD7B  27 5D      BEQ OUTD
327  FD7D  7E FE F2   CRLF2, JMPX CRLF
328  FD80             /
329  FD80  8D 46      EVAL, BSR GETVAL        /EVALUATE EXPRESSION
330  FD82             /
331  FD82  36         NXTRM, PSHA
332  FD83  A6 00      LDAN 0                  /END OF LINE?
333  FD85  27 02      BEQ OUTN
334  FD87  81 29      CPAI :)
335  FD89  32         OUTN, PULA
336  FD8A  27 4E      BEQ OUTD
337  FD8C  8D 04      BSR TERM
338  FD8E  DE 64      LDXD SAVE0
339  FD90  20 F0      BRA NXTRM
340  FD92             /
341  FD92  36         TERM, PSHA              /GET VALUE
342  FD93  37         PSHB
343  FD94  A6 00      LDAN 0
344  FD96  36         PSHA
345  FD97  08         INX
346  FD98  8D 2E      BSR GETVAL
347  FD9A  97 6A      STAD SAVE3
348  FD9C  D7 6B      STBD SV3+1
349  FD9E  DF 64      STXD SAVE0
350  FDA0  CE 00 6A   LDXI SAVE3
351  FDA3  32         PULA
352  FDA4  33         PULB
353  FDA5             /
```

```
354  FDA5  81 2A      CPAI :*                 /SEE IF *
355  FDA7  26 6E      BNE EVAL2
356  FDA9  32         PULA                    /MULTIPLY
357  FDAA  97 68      MULTIP, STAD SAVE2
358  FDAC  D7 69      STBD SV2+1              /2'S COMPLEMENT
359  FDAE  C6 10      LDBI 16
360  FDB0  D7 66      STBD SAVE1
361  FDB2  4F         CLRA
362  FDB3  5F         CLRB
363  FDB4             /
364  FDB4  74 00 68   MULT, LSRX SAVE2
365  FDB7  76 00 69   RORX SV2+1
366  FDBA  24 02      BCC NOAD
367  FDBC  8D 5E      MULTI, BSR ADD
368  FDBE  68 01      NOAD, ASLN 1
369  FDC0  69 00      ROLN 0
370  FDC2  7A 00 66   DECX SAVE1
371  FDC5  26 ED      BNE MULT                /LOOP TIL DONE
372  FDC7  39         RTS2, RTS
373  FDC8             /
374  FDC8  BD FE A5   GETVAL, JSRX CVBIN      /GET VALUE
375  FDCB  24 0E      BCC OUTV
376  FDCD  C1 3F      CPBI :?                 /OF LITERAL
377  FDCF  26 0B      BNE VAR
378  FDD1  DF 76      STXD SAVE9              /OR INPUT
379  FDD3  BD FE D6   JSRX INLN
380  FDD6  8D A8      BSR EVAL
381  FDD8  DE 76      LDXD SAVE9
382  FDDA  08         OUTD, INX
383  FDDB  39         OUTV, RTS
384  FDDC             /
385  FDDC  C1 24      VAR, CPBI :$            /OR STRING
386  FDDE  26 05      BNE VAR1
387  FDE0  8D 90      BSR INCH2
388  FDE2  4F         CLRA
389  FDE3  08         INX
390  FDE4  39         RTS
391  FDE5             /
392  FDE5  C1 28      VAR1, CPBI :(
393  FDE7  26 03      BNE VAR2
394  FDE9  08         INX
395  FDEA  20 94      BRA EVAL
396  FDEC             /
397  FDEC  8D 11      VAR2, BSR CONVP         /OR VARIABLE
398  FDEE  A6 00      LDAN 0                  /OR ARRAY ELEMENT
399  FDF0  E6 01      LDBN 1
400  FDF2  DE 70      LDXD SAVE6              /LOAD OLD INDEX
401  FDF4  39         RTS
402  FDF5             /
403  FDF5  8D 89      ARRAY, BSR EVAL         /LOCATE ARRAY ELEMENT
404  FDF7  58         ASL8
405  FDF8  49         ROLA
406  FDF9  DB 51      ADBD AMPR+1
407  FDFB  99 50      ACAD AMPR
408  FDFD  20 0E      BRA PACK
409  FDFF             /
```

```
410  FDFF  E6 00     CONVP, LDBN 0          /GET LOCATION
411  FE01  08        INX
412  FE02  37        PSHB
413  FE03  C1 3A     CPBI ::
414  FE05  27 EE     BEQ ARRAY              /OF VARIABLE OR
415  FE07  4F        CLRA                   /ARRAY ELEMENT
416  FE08  C4 3F     NDBI 3FX
417  FE0A  CB 02     ADBI 2
418  FE0C  58        ASLB
419  FE0D            /
420  FE0D  DF 70     PACK, STXD SAVE6       /STORE OLD INDEX
421  FE0F  97 6C     STAD SAVE4
422  FE11  D7 6D     STBD SV4+1
423  FE13  DE 6C     LDXD SAVE4             /LOAD NEW INDEX
424  FE15  33        PULB
425  FE16  39        RTS
426  FE17            /
427  FE17  81 2B     EVAL2, CPAI :+         /ADDITION
428  FE19  26 06     BNE EVAL3
429  FE1B  32        PULA
430  FE1C  EB 01     ADD, ADBN 1
431  FE1E  A9 00     ACAN 0
432  FE20  39        RTS
433  FE21            /
434  FE21  81 2D     EVAL3, CPAI :-         /SUBTRACTION
435  FE23  26 06     BNE EVAL4
436  FE25  32        PULA
437  FE26  E0 01     SUBTR, SUBN 1
438  FE28  A2 00     SBAN 0
439  FE2A  39        RTS
440  FE2B            /
441  FE2B  81 2F     EVAL4, CPAI 2FX        /SEE IF IT'S DIVIDE
442  FE2D  26 0C     BNE EVAL5
443  FE2F  32        PULA
444  FE30  8D 33     BSR DIVIDE
445  FE32  97 4E     STAD REMN1
446  FE34  D7 4F     STBD REMN2
447  FE36  96 68     LDAD SAVE2
448  FE38  D6 69     LDBD SV2+1
449  FE3A  39        RTS
450  FE3B            /
451  FE3B  80 3D     EVAL5, SUBI :=         /SEE IF EQUAL TEST
452  FE3D  26 0C     BNE EVAL6
453  FE3F  32        PULA
454  FE40  8D E4     BSR SUBTR
455  FE42  26 03     BNE NOTEQ
456  FE44  5D        TSTB
457  FE45  27 02     BEQ EQL
458  FE47  C6 FF     NOTEQ, LDBI FFX
459  FE49  20 0D     EQL, BRA COMBOUT
460  FE4B            /
461  FE4B  4A        EVAL6, DECA            /SEE IF LESS THAN TEST
462  FE4C  32        PULA
463  FE4D  27 07     BEQ EVAL7
464  FE4F            /
465  FE4F  8D D5     SUB2, BSR SUBTR
466  FE51  59        ROLB
467  FE52  4F        COMOUT, CLRA
468  FE53  C4 01     NDBI 1
469  FE55  39        RTS
```

```
470  FE56                  /
471  FE56  8D F7           EVAL7, BSR SUB2          /GT TEST
472  FE58  53              COMBOUT, COMB
473  FE59  20 F7           BRA COMOUT
474  FE5B                  /
475  FE5B  27              PWRS10, 27X              /10000
476  FE5C  10              10X
477  FE5D  03              3                        /1000
478  FE5E  E8              E8X
479  FE5F  00              0                        /100
480  FE60  64              100
481  FE61  00              0                        /10
482  FE62  0A              10
483  FE63  00              0                        /1
484  FE64  01              1
485  FE65                  /
486  FE65  7F 00 66        DIVIDE, CLRX SAVE1       /DIVIDE 16-BITS
487  FE68  7C 00 66        GOT, INCX SAVE1
488  FE6B  68 01           ASLN 1
489  FE6D  69 00           ROLN 0
490  FE6F  24 F7           BCC GOT
491  FE71  66 00           RORN 0
492  FE73  66 01           RORN 1
493  FE75  7F 00 68        CLRX SAVE2
494  FE78  7F 00 69        CLRX SV2+1
495  FE7B  8D A9           DIV2, BSR SUBTR
496  FE7D  24 04           BCC OK
497  FE7F  8D 9B           BSR ADD
498  FE81  0C              CLC
499  FE82  9C              9CX
500  FE83  0D              OK, SEC
501  FE84  79 00 69        ROLX SV2+1
502  FE87  79 00 68        ROLX SAVE2
503  FE8A  7A 00 66        DECX SAVE1
504  FE8D  27 12           BEQ DONE
505  FE8F  64 00           LSRN 0
506  FE91  66 01           RORN 1
507  FE93  20 E6           BRA DIV2
508  FE95                  /
509  FE95  E6 00           TSTN, LDBN 0             /TEST FOR NUMERIC
510  FE97  C1 3A           CPBI 3AX
511  FE99  2A 04           BPL NOTDEC
512  FE9B  C1 30           CPBI :0
513  FE9D  2C 02           BGE DONE
514  FE9F  0D              NOTDEC, SEC
515  FEA0  39              RTS
516  FEA1  0C              DONE, CLC
517  FEA2  39              DUN, RTS
518  FEA3                  /
519  FEA3  8D 31           CVTLN, BSR INLN
520  FEA5                  /
521  FEA5  8D EE           CVBIN, BSR TSTN          /CONVERT TO BINARY
522  FEA7  25 F9           BCS DUN
523  FEA9  4F              CONT, CLRA
524  FEAA  5F              CLRB
525  FEAB  EB 00           CBLOOP, ADDN 0
526  FEAD  89 00           ACAI 0
527  FEAF  C0 30           SUBI :0
528  FEB1  82 00           SBAI 0
529  FEB3  97 66           STAD SAVE1
```

```
530  FEB5  D7 67      STBD SV1+1
531  FEB7  08         INX
532  FEB8  37         PSHB
533  FEB9  8D DA      BSR TSTN
534  FEBB  33         PULB
535  FEBC  25 E3      BCS DONE
536  FEBE  58         ASLB
537  FEBF  49         ROLA
538  FEC0  58         ASLB
539  FEC1  49         ROLA
540  FEC2  DB 67      ADBD SV1+1
541  FEC4  99 66      ACAD SAVE1
542  FEC6  58         ASLB
543  FEC7  49         ROLA
544  FEC8  20 E1      BRA CBLOOP
545  FECA             /
546  FECA  C1 40      INLN6, CPBI 40X          /CANCEL
547  FECC  27 06      BEQ NEWLIN
548  FECE  08         INX
549  FECF  8C 00 4A   CPXI 0 74                /LINE LENGTH +2
550  FED2  26 08      BNE INLN2
551  FED4  8D 1C      NEWLIN, BSR CRLF
552  FED6             /
553  FED6  CE 00 02   INLN, LDXI 0 2           /INPUT LINE FROM TERMINAL
554  FED9  09         INLN5, DEX
555  FEDA  27 F8      BEQ NEWLIN
556  FEDC             /
557  FEDC  BD FF 00   INLN2, JSRX INCH         /INPUT CHARACTER
558  FEDF  E7 87      STBN 87X                 /STORE IT
559  FEE1  C1 5F      CPBI 5FX                 /BACKSPACE?
560  FEE3  27 F4      BEQ INLN5
561  FEE5             /
562  FEE5  C1 0D      INLN3, CPBI DX           /CARRIAGE RETURN
563  FEE7  28 F3      BMI INLN2
564  FEE9  26 DF      BNE INLN6
565  FEEB             /
566  FEEB  6F 87      INLN4, CLRN 87X          /CLEAR LAST CHAR
567  FEED  CE 00 88   LDXI LINBUF
568  FEF0  20 04      BRA LF
569  FEF2             /
570  FEF2  C6 0D      CRLF, LDBI DX            /CARR-RET
571  FEF4  8D 02      BSR OUTCH2
572  FEF6  C6 0A      LF, LDBI 10              /LINE-FEED
573  FEF8  7E FF 81   OUTCH2, JMPX OUTCH
574  FEFB             /
575  FEFB  0D         OKM, DX                  /"OK" MESSAGE
576  FEFC  0A         10
577  FEFD  4F         :O
578  FEFE  4B         :K
579  FEFF  00         0
580  FF00             /
581  FF00             END
582  FF00
```

```
INCH        FF00
POLCAT      FF24
OUTCH       FF81
OUTS        FF82
ZERO        0000
AT          0004
VARS        0006
BRAK        003A
SAVE10      003C
SV10+1      003D
BRIK        003E
UP          0040
SAVE11      0042
SV11+1      0043
SAVE14      0044
EXCL        0046
QUOTE       0048
DOLR        004A
DOLR+1      004B
DOLLAR      004C
REMN1       004E
REMN2       004F
AMPR        0050
AMPR+1      0051
QUITE       0052
QUITE+1     0053
PAREN       0054
PARIN       0056
STAR        0058
STAR+1      0059
PLUS        005A
COMA        005C
MINS        005E
PERD        0060
SLASH       0062
SAVE0       0064
SAVE1       0066
SV1+1       0067
SAVE2       0068
SV2+1       0069
SAVE3       006A
SV3+1       006B
SAVE4       006C
SV4+1       006D
SAVE5       006E
SAVE6       0070
SV6+1       0071
SAVE7       0072
SAVE8       0074
SAVE9       0076
SV9+1       0077
COLN        0078
SEMI        007A
LESS        007C
EQAL        007E
GRAT        0080
DECBUF-1    0081
DECBUF      0082
LASTD       0086
DELIM       0087
```

LINBUF	0088
STACK	00F1
MI	0100
NMI	0104
PRGM	0108
START	FC00
OKM	FEF8
STRGT	FC4B
LOOP	FC09
CVTLN	FEA3
STMNT	FC4E
EXEC	FC39
LOOP2	FC17
FIND	FC7E
EQSTRT	FC19
LOOP3	FC35
FND3	FC91
VAR2	FDEC
SKIP	FC3F
EVIL	FC46
OUTX	FC43
EVALU	FC98
STRNG	FD75
SKP2	FCC1
LST2	FC58
PRNT2	FD20
PNTMSG	FD58
CRLF	FEF2
NXTXT	FC74
LOOKAG	FC77
FND2	FC81
RTS1	FC97
SET	FC95
EVAL	FD80
CONVP	FDFF
AR1	FCAB
PRNT	FD1F
AR2	FCB4
INSRT	FCDD
DELT	FCCF
FITIT	FCD8
GOTIT	FD16
CNTLN	FCE5
OPEN	FCEB
RSTRT	FD1C
SLIDE	FD01
DON	FD09
MOVL	FD10
PWRS10	FE5B
CVD1	FD28
DIVIDE	FE65
ZRSUP	FD4E
STRTMS	FD59
OUTMSG	FD5B
CTLC	FD67
RTS2	FDC7
INCH2	FD72
OUTD	FDDA
CRLF2	FD7D
GETVAL	FDC8

NXTRM	FD82
OUTN	FD89
TERM	FD92
EVAL2	FE17
MULTIP	FDAA
MULT	FDB4
NOAD	FDBE
MULTI	FDBC
ADD	FE1C
CVBIN	FEA5
OUTV	FDDB
VAR	FDDC
INLN	FED6
VAR1	FDE5
ARRAY	FDF5
PACK	FE0D
EVAL3	FE21
EVAL4	FE2B
SUBTR	FE26
EVAL5	FE3B
EVAL6	FE4B
NOTEQ	FE47
EQL	FE49
COMBOUT	FE58
EVAL7	FE56
SUB2	FE4F
COMOUT	FE52
GOT	FE68
DIV2	FE7B
OK	FE83
DONE	FEA1
TSTN	FE95
NOTDEC	FE9F
DUN	FEA2
CONT	FEA9
CBLOOP	FEAB
INLN6	FECA
NEWLIN	FED4
INLN2	FEDC
INLN5	FED9
INLN3	FEE5
INLN4	FEEB
LF	FEF6
OUTCH2	FEF8