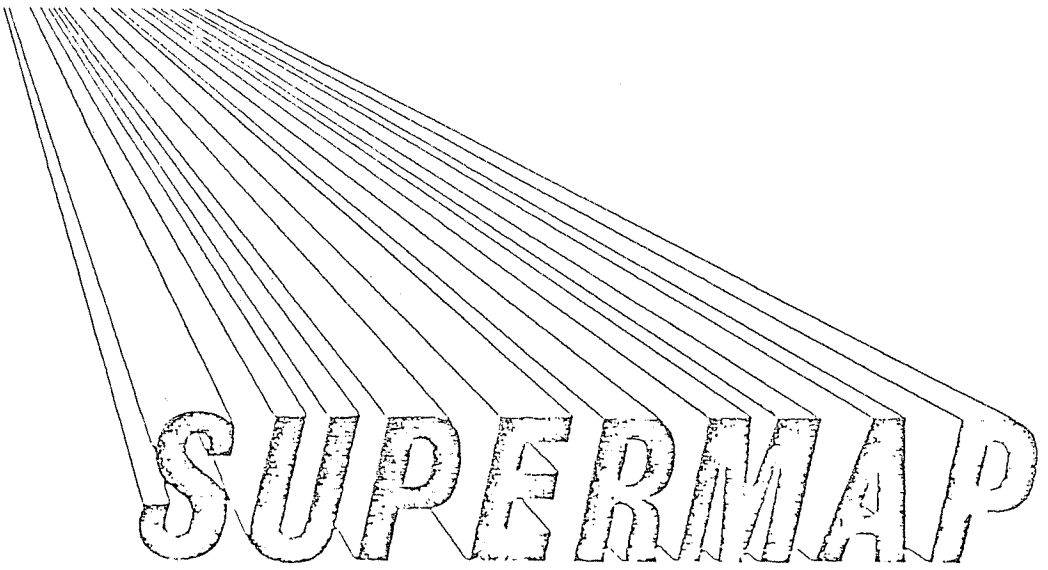

LEVEL II ROM DOCUMENTATION

by

FULLER SOFTWARE

Fuller Software is in no way attempting to infringe upon the proprietary rights of MICROSOFT by this publication. The expressed purpose of SUPERMAP is to aid users of LEVEL II microcomputers in using and understanding their machines.

IMPORTANT NOTICE

Fuller Software assumes no liability with respect to the use of this publication nor any damages resulting from te use of any information contained herein. SUPERMAP is sold in an "as is" condition and is not represented as being error free.

FULLER SOFTWARE
630 E. Springdale
Grand Prairie, TX 75051

SUPERMAP by ROGER FULLER

0000 POWER ON ROUTINE TURN OFF CLOCK ZERO A THEN JUMP

****

0008 RST 8H: (HL)-((SP)) SN ERROR IF NON ZERO

****

0010 RST 10H: INCREMENT HL, PASS TROUGH STRING IGNORE CR AND SPACES.
SET C IF NEXT CHARACTER NUMERIC. RESET C IF NOT.

****

0013 KEYBOARD ROUTINE (SEE 002B)

****

0018 RST 18H: HL-DE Z SET IF EQUAL. C SET IF DE>HL.

****

001B DISPLAY ROUTINE

****

0020 RST 20: IF NTF=8 C IS RESET ELSE C SET. A=NTF-3 S AND Z
FLAGS VALID. MAINTAINS BC, DE, HL.

****

0028 RST 28H BREAK VECTOR

****

002B SCAN KEYBOARD RETURN WITH CHAR IN A USES AF, DE

****

0033 DISPLAY BYTE IN A ON SCREEN

****

003B PRINTER DRIVER ENTRY
0049 SCAN KEYS WAIT FOR KEY PRESSED USES AF, DE

****

NOTE: SUPERMAP is designed to be transferred to a disassembled listing of LEVEL II BASIC. Three and five digit numbers are decimals.

```
    KEYBOARD LOOKUP TABLE
0050  (ENTER)
0051  (ENTER)  SHIFT
0052  (CLEAR)
0053  (CLEAR)  SHIFT
0054  (BREAK)
0055  (BREAK)  SHIFT
0056  (UP ARROW)
0057  (UP ARROW)  SHIFT
0058  (DOWN ARROW)
0059  (DOWN ARROW)  SHIFT
005A  (LEFT ARROW)
005B  (LEFT ARROW)  SHIFT
005C  (RIGHT ARROW)
005D  (RIGHT ARROW)  SHIFT
005E  (SPACE)
005F  (SPACE)  SHIFT

      ****************************************

0060  DELAY LOOP  BC IS COUNTER   14.65 MICROSECONDS EACH LOOP

      ****************************************

0066 NMI RESET

      ****************************************

0075  NON DOS INITIALIZATION AREA MOVE 18F7-191C  TO 4080-40A6
008B  41E8 TO INPUT BUFFER ADDRESS POINTER (40A7)
0091  LOAD DUMMY JUMP VECTORS IN DOS COMMANDS JUMP ADDRESSES.  JUMP
      WILL BE TO L3 ERROR (012D) INSTEAD OF DOS.  USED DURING LEVEL II
     INTERPRETATION OF BASIC PROGRAM. COMMAND   ENTRY POINTS.
009F  PLACE RETURN COMMANDS IN DOS LINK AREA (THESE ARE USED BY LEVEL
      II MACHINE ROUTINES)
00B2  'MEMORY SIZE' ROUTINE   CLEAR SCREEN
00B5  POINT TO 'MEMORY SIZE' MESSAGE
00B8  DISPLAY IT
00BB  WAIT FOR USER INPUT
00BE  IF (BREAK) ASK AGAIN
00C0  LOCATE 1ST CHAR
00C1  IS IT ANYTHING
00C2  IF SO SKIP MEMORY SIZE ROUTINE
00C4  TEST FOR END OF ACTUAL MEMORY.  USED WHEN (ENTER) IS GIVEN
      TO MEMORY SIZE QUESTION
00C7  HL=MEMORY POINTER
00CC  GET A BYTE IN MEMORY
00CD  SAVE IT IN B FOR LATER
00CE  COMPLEMENT IT
00CF  PUT IT BACK WHERE YOU GOT IT
00D0  SEE IF MEMORY WAS THERE TO RECEIVE IT
00D1  PUT BACK ORIGINAL BYTE
00D2  DO TIL MEMORY FAILS TEST
      """""""""""""""""""""""""""""""""""""""
```

```
00D6  CONVERT INPUT
00DA  SN ERROR IF NOT NUMERIC INPUT
00DF  LOAD TEST BYTE
00E1  SAVE CURRENT MEMORY BYTE
00E2  PUT IN TEST BYTE
00E3  WAS MEMORY THERE TO RECEIVE IT?
00E4  RESTORE MEMORY
00E5  GO BACK TO MEMORY SIZE ROUTINE IF USER WAS WISHFUL THINKING
      (NOTE: YOU END UP WITH LESS MEMORY AVAILABLE EVEN IF YOU
      EXCEED ACTUAL SIZE BY 1)
00E7  POINT TO END OF MEMORY -1
00E8  LOAD MINIMUM MEMORY SIZE
00EB  CHECK FOR UNDER SIZE
00EC  OM ERROR IF UNDER 17430   THEN SINCE BC=0000 RETURN IN ERROR
      ROUTINE WILL RESULT IN A JP TO 0000
00EF  PREPARE TO REDUCE MEMORY BY 050
00F2  SAVE END OF MEMORY
00F5  REDUCE BY 050 (CLEAR 050)
00F6  SAVE STRING SPACE POINTER
00F9  REVELATION 21:5
00FC  POINT TO 'RADIO SHACK...' MESSAGE
00FF  DISPLAY IT
0102  ON TO THE FARM
      """""""""""""""""""""""""""""""""""""""
0105  'MEMORY SIZE' MESSAGE
0111  'RADIO SHACK LEVEL II BASIC' MESSAGE

      *****************************************

012D  L3 ERROR ENTRY POINT

      *****************************************

0132* POINT
0135* SET
0138* RESET

      *****************************************

019D* INKEY$

      *****************************************

01C9* CLEAR SCREEN  DISPLAYS CODE FOR CLS

      *****************************************

01D3* RANDOM   USES REFRESH REGISTER

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

```
01D9  MAKE A CASSETTE PULSE
      """""""""""""""""""""""""""""""""""""

01F8  TURN CASSETTE OFF
01F9  BIT 2 CONTROLS MOTOR  1=ON  0=OFF
      """""""""""""""""""""""""""""""""""""

0212  DEFINE DRIVE 1,0 IN A
      """""""""""""""""""""""""""""""""""""

021E  RESET THE TAPE INPUT CIRCUIT
0221  OUT REG A TO PORT FF
      """""""""""""""""""""""""""""""""""""

0234  BLINK*
      """""""""""""""""""""""""""""""""""""

0235  READ A BYTE
      """""""""""""""""""""""""""""""""""""

0241  READ BIT (1 OR 0)
0243  WAIT FOR TIMING 'PIP'
0248  DELAY
024C  RESET INPUT CIRCUIT
0251  DELAY
0253  LOOK FOR 'PIP' IF PRESENT THEN BIT=1.  IF NOT THEN BIT=0
      """""""""""""""""""""""""""""""""""""

0264  WRITE A BYTE
0268  8 BITS TO SEND
026A  SAVE BYTE IN A
026B  SEND TIMING PIP
026E  RESTORE BYTE
026F  BIT TO CARRY
0270  SAVE BYTE
0271  JUMP IF BIT IS TO BE A 0
0273  SEND PIP FOR A BIT=1
0276  NEXT BIT
0277  LOOP TIL DONE
      """""""""""""""""""""""""""""""""""""

027E  DELAY FOR A BIT=0
0284  TURN ON CASSETTE
0287  WRITE LEADER AND SYNC BYTE   READY 255 COUNT
0289  BYTE TO SEND IS 00
028A  SEND BYTE
028D  LOOP TIL DONE
028F  SYNC BYTE
0291  SEND IT RETURN WHEN DONE
      """""""""""""""""""""""""""""""""""""

0293  TURN ON CASSETTE
0296  FIND LEADER AND SYNC BYTE
0297  ZERO A
0298  GET A BIT
029B  SYNC BIT YET?
029D  NO LOOP TIL SO
029F  GET A *
02A1  SEND LEFT *
02A4  SEND RIGHT *

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

```
02A9  GET 2 BYTES FROM TAPE
02AC  SAVE
02AF  TURN OFF CASSETTE
02B2* SYSTEM   RET IF NON DOS
02B5  INITIALIZE STACK IN INPUT BUFFER AREA
02B8  RETURN TO BEGINNING OF LINE
02BA  GET * PROMPT
02BD  DISPLAY IT
02C0  WAIT FOR INPUT
02C3  BAIL OUT IF (BREAK)
02C6  LOCATE FIRST CHARACTER
02C7  SN ERROR IF DRY RUN (NO INPUT)
02CA  CHECK FOR JUMP COMMAND  (/)
02CC  IF JUMP
02CE  FIND LEADER
02D1  READ A BYTE
02D4  SYSTEM HEADER?
02D6  NO?  LOOP BACK
02D8  FILE NAME BLOCK LENGTH (6)
02DA  FIRST CHAR
02DB  END OF BLOCK?
02DC  YES?  BLINK *
02DE  READ A BYTE
02E1  CORRECT CHAR?
02E2  POINT TO NEXT CHAR
02E3  IF NOT CORRECT START OVER
02E5  FINISH A 6 PACK
02E7  'TWINKLE TWINKLE LITTLE *'
02EA  READ A BYTE
02ED  ENTRY POINT HEADER?
02EF  IF YES
02F1  DATA HEADER?
02F3  IF NOT
02F5  READ BYTE
02F8  SAVE # DATA BYTES
02F9  GET LOAD ADDRESS
02FC  LOW ORDER BYTE OF ADDRESS TO INITIALIZE CHECK SUM
02FD  SAVE CHECK SUM
02FE  READ A BYTE
0301  PUT IT IN PLACE
0302  POINT TO NEXT PLACE
0303  RETREIVE CHECKSUM
0304  UPDATE IT
0305  LOOP TIL BLOCK READ
0307  READ CHECKSUM
030A  CHECK IT
030B  LOOP BACK IF OKAY
030D  GET A C
030F  DISPLAY IT
0312  LOOP BACK ANYWAY
      """""""""""""""""""""""""""""""""""""
0314  GET 2 BYTES AND PUT IN HL
031D  JUMP ROUTINE
0322  CHECK FOR AN ADDRESS EVALUATE INPUT
0329  JUMP TO IT

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

```
032A  OUTPUT BYTE TO TAPE, VIDEO, OR PRINTER   (409C)= -1,0,1
032B  SAVE BYTE
032C  RET IF NON DOS
032F  GET DEVICE TYPE FLAG
0333  RESTORE BYTE
0335  WRITE TO TAPE
0338  WRITE TO PRINTER
033A  WRITE TO VIDEO
0342  UPDATE CURSOR POSTION (0-3F LIMIT)
      """""""""""""""""""""""""""""""""""""
0348  CHECK FOR DOUBLE WIDTH. (403D)=8 FOR 32 CHAR LINE. (403D)=0 FOR
      64 CHARACTER LINE.
0355  RESET IF CURSOR POS BEYOND END OF LINE

      *********************************************

0361  KEYBOARD INPUT TO BUFFER  INPUT ROUTINE FOR KEYBOARD
0362  RESET LAST KEY STORAGE
0365  RESET LINE CURSOR POSITION
036C  LOAD BUFFER POINTER  (NORMALLY 41E8)
036F  BUFFER LENGTH= 240.  USE INSERT TO ADD MORE.
0371  RETURN WITH FULL BUFFER
0375  C= INPUT LENGTH
0376  BC=INPUT LENGTH
0378  HL= END OF INPUT LENGTH POINTER
0379  TERMINATE INPUT WITH 00 FLAG
037B  RETURN POINTER TO BEGINNING OF BUFFER
0380  BACK UP FOR RST 10
0381  IF (BREAK)

      *********************************************

0384  CALLED BY DOS EXIT 41AF.  LOOP TIL KEY PRESSED
038B  RETURN PRINTER CARRIAGE TO BEGINNING OF NEXT LINE IF REQUIRED
038C  SELECT VIDEO OUTPUT
038F  GET PRINTER POSITION
0392  CHECK FOR BEGINNING OF LINE
0393  IF SO RETURN
0394  GET CR
0397  SEND IT
      """""""""""""""""""""""""""""""""""""
039C  OUTPUT TO LPRINTER
03AD  GET A 0 TO RESET PRINT POSITION
03A4  IF TOP OF FORM
03A8  IF LINE FEED
03AA  GET CR
03AD  SEND IT
03B1  GET PRINTER POSITION
03BA  GET CHAR

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

```
03C2  DRIVER ENTRY ROUTINE
03C9  SAVE RETURN ADDRESS
03CD  PUT CHARACTER TO DISPLAY IN C
03CE  GET DEVICE TYPE FLAG,  (DE)= 1,7,6  FOR KEYBOARD,VIDEO,LPRINTER
03CF  B=1 KEYBOARD B=2 VIDEO OR PRINTER
03D1  NON DOS PUT 0 IN A
03DC  JUMP TO APPROPRIATE DRIVER ROUTINE

      ****************************************

03E3  KEYBOARD DRIVER  HL=KEYBOARD BUFFER POINTER
03E6  BC=ROW ADDRESS POINTER
03E9  D=ROW COUNTER (0 TO 6)
03EC  SAVE IN E
03EB  GET ROW BYTE AT (BC)
03ED  KEEP AMERICA BEAUTIFUL
03F0  GO IF KEY PRESSED
03F2  BUMP ROW COUNTER
03F3  BUMP KEYBOARD BUFFER POINTER
03F4  POINT TO NEXT ROW
03F6  GO NEXT ROW TIL ALL BUT (SHIFT)  TESTED
03F9  RETURN IF NO KEY PRESSED (A=0)
03FA  SAVE ROW BYTE
03FB  GET ROW COUNT
03FF  PUT ROW COUNT *8 IN D
0400  READY TESTING MASK   BIT POSITION = COLUMN #
0402  LOAD MASK
0403  TEST FOR KEY
0404  JUMP IF FOUND
0406  BUMP COLUMN # FOR TEST
0407  MOVE TEST BIT TO NEXT COLUMN
0409  DO AGAIN
      """""""""""""""""""""""""""""""""""""
0405  GET SHIFT BIT
040E  SAVE IN B
040F  LOAD ROW BIT * 8+COLUMN #
0410  ADD 064 TO IT (THIS ADJUSTS FOR ASCII LETTERS)
0412  TEST FOR NON LETTER CODE
0414  GO IF NON LETTER  (LAST 3 ROWS)
0416  SEND SHIFT BIT TO CARRY
0418  SKIP LOWER CASE ADJUSTMENT IF NO SHIFT
041A  CONVERT TO LOWER CASE
041C  SAVE IN D
041D  GET ROW 6 BITS
0420  TEST FOR DOWN ARROW
0422  GO IF NOT DOWN ARROW
0424  RETRIEVE CHAR
0425  ADJUST
      """""""""""""""""""""""""""""""""""""
```

```
0429  TEST FOR LAST ROW
042B  JUMP IF LAST ROW
042D  READJUST TO ROWS 4, 5  A=ROW * 8 + COLUMN# + 016
042F  CHECK FOR = OR ) OR ?
0431  JUMP IF NOT
0433  ADJUST ASCII
0435  GET SHIFT BIT
0437  JUMP IF NOT SHIFT
0439  ADJUST ASCII
043B  JUMP
      """""""""""""""""""""""""""""""""""
043D  A=(ROW * 8 + COLUMN #  - 048) * 2
043E  GET SHIFT BIT
0440  JUMP IF NOT
0442  A=COLUMN # * 2 + 1
0443  POINT TO CODE TABLE
0446  DISPLACEMENT
0449  COMPUTE POSITION IN TABLE
044A  GET ASCII CODE
044B  SAVE CHARACTER
044C  LOAD DELAY COUNT
044F  DELAY
0452  RETREIVE CHARACTER
0453  CHECK FOR (BREAK)
0455  LEAVE WITH CHAR
0457  RETURN

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

SYSTEM TAPE FORMAT

LEADER	256 zeros followed by a A5 sync byte
55	System format header byte
XX XX XX XX XX XX	6 char file name
3C	Data header
XX	Data length 00=256 bytes
LSB	Loading
MSB	Address
XX ... XX	Line itself
XX	Checksum of line bytes and load address
.	
.	
.	
78	End of file marker
LSB	Entry
MSB	Address

```
0458  VIDEO DRIVER
045B  SCREEN CURSOR POSITION IN HL
045E  JUMP IF JUST ENTERED FROM 3DC
      """""""""""""""""""""""""""""""""""""""

0467  GET CHAR
046A  IF CONTROL CHAR
046F  IF GRAPHIC OR SPACE COMPRESSION
0473  IF NOT A LETTER
0479  IF UPPER CASE
047B  CHANGE TO UPPER CASE
04A1  CURSOR TO BEGINNING OF LINE
      """""""""""""""""""""""""""""""""""""""""

04A6  CHECK FOR SPACE COMPRESSION CODE
04A8  IF GRAPHIC
04AA  REMOVE BIAS
04AF  GET SPACE
04B1  SEND IT
04B4  LOOP TIL DECOMPRESSED
      """""""""""""""""""""""""""""""""""""""

04B8  TURN ON CURSOR
04BD  TURN OFF CURSOR
04C0  HOME CURSOR
04C3  CHANGE TO 64 CHARACTER MODE
      """""""""""""""""""""""""""""""""""""""""

04CE  BACKSPACE  AND ERASE
04D2  CHECK FOR 64 CHARACTER
      """""""""""""""""""""""""""""""""""""""""

04DA  BACKSPACE CURSOR
      """""""""""""""""""""""""""""""""""""""

04EC  TAB
      """""""""""""""""""""""""""""""""""""""

04F6  GO DOUBLE WIDTH
04FB  SET MODE FLAG
04FE  GO DOUBLE
      """""""""""""""""""""""""""""""""""""""
      """""""""""""""""""""""""""""""""""""""

0506  SAVE RETURN ADDRESS
050C  IF BACKSPACE
050E  IF NO FUNCTION   2,3,4,5,6,7,9
0513  IF TURN ON CURSOR
051D  IF DOUBLE WIDTH
0521  IF SHIFT BACK ARROW
0523  IF RIGHT ARROW
0529  IF DOWN ARROW
052D  IF UP ARROW
0531  IF HOME CURSOR
0535  IF BEGINNING OF LINE
053A  IF ERASE TO END OF LINE
053C  IF ERASE TO END OF FRAME
      """""""""""""""""""""""""""""""""""""""

0554  SCROLL DISPLAY      POINT TO TOP OF DISPLAY
0557  POINT TO ROW 2
055B  15 ROWS TO MOVE
055E  HUP ONE TWO  HUP ONE TWO ........
0562  BLANK OUT ROW 16

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

```
058D  PRINTER DRIVER ROUTINE
0591  TOP OF FORM WANTED?
0595  TOP OF FORM WANTED?
0599  TOP OF FORM
059F   GET LINES PER PAGE (4028)
05A2  SUBTRACT CURRENT LINE POSITION (4029)
05A5  PLACE # REMAINING LINES IN B
05A6  PRINTER READY?
05A9  WAIT TIL SO
05AD  SEND A LF
05B0  TIL JOB IS DONE
05B2  ZERO COUNT
      """""""""""""""""""""""""""""""""""""""
05B4  SEND CHARACTER OUT
05B5  PRINTER READY?
05B8  WAIT TIL SO
05BB  SEND CHAR
05BE  CR?
05C0  IF NOT
05C1  BUMP LINE COUNTER
05C7  PAGE?
05CB  IF NOT
05CC  ZERO LINE COUNTER
      """""""""""""""""""""""""""""""""""""""
05D1  PRINTER HAND SHAKE. READY IF A=(0011XXXX). BIT 7=0 (NOT BUSY).
      BIT 6=0 (PAPER OK). BIT 5=1 (DEVICE SELECTED).
      BIT 4=1 (NO FAULT). BITS 3,2,1,0 NOT TESTED.

      ******************************************

05D9  INPUT ROUTINE  HL POINTS TO INPUT AREA
05DF  C=BUFFER BYTES REMAINING  (240 AT START)
05ED  RETURN WITH KEY PRESSED IN A
060C  PUT CHAR IN BUFFER
0613  PRINT IT
      """""""""""""""""""""""""""""""""""""""
0619  IF CLEAR SCREEN HIT, CLEAR BUFFER. (THIS IS WHY CLEARING SCREEN
      WITH KEYBOARD ERASES TYPED BUT UNENTERED CODE)
      """""""""""""""""""""""""""""""""""""""
0630  BACKUP CURSOR
      """""""""""""""""""""""""""""""""""""""
0641  DOUBLE WIDTH

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

```
0674  INITIALIZATION   MOVE 06D2-0707 TO 4000-4035
0683  DO ABOVE 128 TIMES (BEAT A DEAD HORSE)
0689  ZERO 4036 TO 405D .
068B  TEST FOR BREAK
0690  IF NOT BREAK
0693  LOAD NEW STACK
0696  GET DISC STATUS
069C  IF NON DOS
      """""""""""""""""""""""""""""""""""""""
06A1  SELECT DRIVE
06A4  POINT TO CONTROLLER
06AA  RESTORE
06AF  TWIDDLE THUMBS
06B2  TEST FOR READY
06B7  ZERO SECTOR REGISTER
06BD  READ
06C9  GO DOS LOADER

      ******************************************

06CC  ENTERED ON NMI RESET (SEE 0066)

      ******************************************

06D2-0707  RST JUMP ADDRESSES, I/O DEVICE CONTROL BLOCK

      ******************************************

070B  FLOATING POINT ADDITION      ARITH=(HL) + ARITH
      """""""""""""""""""""""""""""""""""""""
0710  FLOATING POINT SUBTRACTION  ARITH=(HL)-ARITH
0713  FLOATING POINT SUBTRACTION  ARITH=BCDE-ARITH
0716  FLOATING POINT ADDITION     ARITH=BCDE+ARITH
07B2  OV ERROR ENTRY POINT

      ******************************************

0809* LOG  ARITH=LOG(ARITH)
      """""""""""""""""""""""""""""""""""""""
0847  FLOATING POINT MULTIPLICATION  ARITH=BCDE*ARITH
      """""""""""""""""""""""""""""""""""""""
08A2  FLOATING POINT DIVISION  : ARITH=BCDE/ARITH
      """""""""""""""""""""""""""""""""""""""
0955  CHECK ARITH FOR ZERO

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

```
0977* ABS:  ARITH=ABS(ARITH).  INTEGER OR SINGLE IN AND OUT.
      NTF REQUIRED AND MAINTAINED.
0982  ARITH= -ARITH:  SINGLE ONLY.  MAINTAINS BC, DE
      """""""""""""""""""""""""""""""""""""

098A* SGN ACCEPT FLOATING POINT OR INTEGER.  OUTPUT INTEGER IN ARITH
      """""""""""""""""""""""""""""""""""""

0994  CHECK SIGN OF ARITH, FLOAT OR INTEGER.  REQUIRES NTF.  A=00  IF
      ARITH=0.  A=01  IF  ARITH GREATER  THAN 0.  A=FF IF ARITH LESS
      THAN 0.  S AND Z FLAGS ALSO VALID
      """""""""""""""""""""""""""""""""""""

09A4  LOAD SINGLE ARITH TO STACK.  TO RETRIEVE  POP BC, POP DE.
      A,BC,HL UNALTERED.
09B1  LOAD SINGLE:  ARITH=(HL)(HL+1)(HL+2)(HL+3)
09B4  LOAD SINGLE:  ARITH=BCDE.  HL UNALTERED
09BF  LOAD SINGLE:  BCDE=ARITH.
09C2  LOAD SINGLE:  BCDE=(HL)(HL+1)(HL+2)(HL+3)
09CB  MOVE FROM (ARITH) TO (HL) 4 BYTES
09CE  MOVE FROM (DE) TO (HL) 4 BYTES
      """""""""""""""""""""""""""""""""""""

09D2  MOVE FROM (HL) TO (DE) NTF BYTES
09D3  MOVE FROM (DE) TO (HL) NTF BYTES
09D6  MOVE FROM (DE) TO (HL) A BYTES
09D7  MOVE FROM (DE) TO (HL) B BYTES

      *****************************************

0A0C  SINGLE  COMPARE:  ARITH-BCDE
      """""""""""""""""""""""""""""""""""""

0A39  INTEGER COMPARE:  HL-DE
      """""""""""""""""""""""""""""""""""""

0A4F  DOUBLE  COMPARE:  ARITH-ARITHEX
0A78  DOUBLE  COMPARE:  ARITHEX-ARITH

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

EDITOR ASSEMBLER SOURCE TAPE FORMAT

```
LEADER              256 zeros followed by an A5 sync byte

D3                  Source header
XX XX XX XX XX XX   File name

   #1 #2 #3 #4 #5   Line # in ASCII  (bit 7 is set)
   20               Data header
   XX ..... XX      Line (128 bytes maximum)
   0D               End of line marker

   .
   .
   .

1A                  End of file marker
```

```
0A7F* CINT
0A83  ALREADY INTEGER
0A84  TM ERROR IF STRING
0A87  IF DOUBLE CONVERT TO SINGLE

      ****************************************

0A9A  RETURN TO BASIC WITH OUTPUT OF USER ROUTINE IN HL.
0A9D  FLAG IT INTEGER

      ****************************************

0AB1* CSNG
      """""""""""""""""""""""""""""""""""""
0ACC  INTEGER ARITH TO  SINGLE ARITH  CONVERSION
0ACF  INTEGER HL    TO  SINGLE ARITH  CONVERSION

      ****************************************

0ADB* CDBL:  ARITH(DOUBLE)=ARITH(INTEGER OR SINGLE).  REQUIRES NTF

      ****************************************

0AF4  TEST NTF=3 (STRING).  IF STRING RETURN ELSE ERROR.
      BC,DE,HL UNALTERED.
      """""""""""""""""""""""""""""""""""""
0AF6  TM ERROR ENTRY POINT

      ****************************************

0B26* FIX:  IF FLOATING POINT TRUNCATE  TO INTEGER AND RETURN FLOATING
      POINT.  IF INTEGER RETURN.  IF STRING ERROR.
      """""""""""""""""""""""""""""""""""""
0B37* INT
0B38  IF INTEGER
0B39  IF DOUBLE
0B3B  TM ERROR IF STRING

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

```
0C70  DBL PRECISION SUBTRACTION: ARITH=ARITH-ARITHEX
0C77  DBL PRECISION ADDITION    : ARITH=ARITH+ARITHEX
      """""""""""""""""""""""""""""""""""""""""""""
0DA1  DBL PRECISION MULTIPLICATION: ARITH=ARITH*ARITHEX
0DE5  DBL PRECISION DIVISION      : ARITH=ARITH/ARITHEX
0E65  LOAD dOUBLE PRECISION ASCII CONSTANT TO aRITH. POINT HL TO INPUT
      STRING DELIMITED BY 0 OR COMMA.  AFTER LOAD HL POINTS TO
      DELIMITER
0E6C  LOAD ASCII CONSTANT TO ARITH. RETURN THE LEAST NECESSARY NUMBER
      TYPE (SEE LEVEL II MANUAL FOR RULES) POINT HL TO INPUT STRING
      DELIMITED BY 0 OR COMMA
0E7B  IF -
0E80  IF +
0E84  IF NUMERIC
0E89  IF .
0E8E  IF E
0E92  IF %
0E97  IF #
0E9C  IF !

      ********************************************

0F40  MULTIPLY HL BY TEN

      ********************************************

0FAB  OUTPUT 'IN ' MESSAGE
      """""""""""""""""""""""""""""""""""""""""""""
0FAF  OUTPUT A LINE #

      ********************************************

0FBD  ARITH AND NTF TO ASCII CONVERSION HL POINTS TO STRING

      ********************************************

13E7* SQR
1439* EXP

      ********************************************

14C9* RND

      ********************************************

1541* COS
1547* SIN
15A8* TAN
      """""""""""""""""""""""""""""""""""""""""""""
15BD* ATN

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

1608 TABLE OF ENTRY POINTS FOR LEVEL II BASIC COMMANDS

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

1650 RESERVED WORD LIST FOR LEVEL II COMMANDS

ABS	D9 0977	GOSUB	91 1EB1	READ	8B 21EF
AND	D2 25FD	GOTO	8D 1EC2	REM	93 1F07
ASC	F6 2A0F	1F	8F 2039	RESET	82 0138
ATN	E4 15BD	INKEY$	C9 019D	RESTORE	90 1D91
AUTO	B7 2008	INP	DB 2AEF	RESUME	9F 1FAF
CDBL	F1 0ADB	INPUT	89 219A	RETURN	92 1EDE
CHR$	F7 2A1F	INSTR	C5 419D	RIGHT$	F9 2A91
CINT	EF 0A7F	INT	D8 0B37	RND	DE 14C9
CLEAR	B8 1E7A	KILL	AA 4191	RSET	AC 419A
CLOAD	B9 2C1F	LEFT$	F8 2A61	RUN	8E 1EA3
CLOSE	A6 4185	LEN	F3 2A03	SAVE	AD 41A0
CLS	84 01C9	LET	8C 1F21	SET	83 0135
CMD	85 4173	LINE	9C 41A3	SGN	07 098A
CONT	B3 1DE4	LIST	B4 2B2E	SIN	E2 1547
COS	E1 1541	LLIST	B5 2B29	SQR	CD 13E7
CSAVE	BA 2BF5	LOAD	A7 4188	STEP	CC 2B01
CSNG	F0 0AB1	LOC	EA 4164	STOP	94 1DA9
CVD	E8 415E	LOF	EB 4167	STR$	F4 2836
CVI	E6 4152	LOG	DF 0809	STRING$	C4 2A2F
CVS	E7 4158	LPRINT	AF 2067	SYSTEM	AE 02B2
DATA	88 1F05	LSET	AB 4197	TAB(	BC 2137
DEF	BD 415B	MEM	C8 27C9	TAN	E3 15A8
DEFDBL	9B 1E09	MERGE	A8 418B	THEN	CA ----
DEFINT	99 1E03	MID$	FA 2A9A	TIME$	C7 4176
DEFSNG	9A 1E06	MKD$	EE 4170	TO	BD ----
DEFSTR	98 1E00	MKI$	EC 416A	TROFF	97 1DF8
DELETE	B6 2BC6	MKS$	ED 416D	TRON	96 1DF8
DIM	8A 2608	NAME	A9 418E	USING	BF 2CBD
EDIT	9D 2E60	NEW	BB 1B49	USR	C1 27FE
ELSE	95 1F07	NEXT	87 22B6	VAL	FF 2AC5
END	80 1DAE	NOT	CB 25C4	VARPTR	C0 24EB
EOF	E9 4161	ON	A1 1F6C	+	CD 249F
ERL	C2 24DD	OPEN	A2 4179	-	CE 2532
ERR	C3 24CF	OR	D3 25F7	*	CF ----
ERROR	9E 1FF4	OUT	A0 2AFB	/	D0 ----
EXP	E0 1439	PEEK	E5 2CAA	↑	D1 ----
FIELD	A3 417C	POINT	C6 0132	>	D4 ----
FIX	F2 0B26	POKE	B1 2CB1	=	D5 ----
FN	BE 4155	POS	DC 27F5	<	D6 ----
FOR	81 1CA1	PRINT	B2 206F	&	26 4194
FRE	DA 27D4	PUT	A5 4182	'	3A 93 FB
GET	A4 4174	RANDOM	86 01D3		

1821 END OF TABLE MARKER

```
""""""""""""""""""""""""""""""""""""""""""
```

1822 TABLE OF JUMP ADDRESSES FOR ENTRY POINTS OF BASIC INSTRUCTIONS.

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

```
191D  'ERROR','IN','READY','BREAK' STRINGS FOR BASIC MESSAGES
1955  IQ TESTING SERVICE
1930  IF (409A)=2  SN ERROR OUTPUT GOTO EDIT MODE
197A  OM ERROR ENTRY POINT
      """""""""""""""""""""""""""""""""""
199A  /0 ERROR ENTRY POINT
199D  NF ERROR ENTRY POINT
19A0  RW ERROR ENTRY POINT
19A2  ERROR OUTPUT ROUTINE.  ERROR CODE IN E
19E6  RETURN TO BEGINNING OF LINE.  ZERO A
19E9  POINT HL TO BOTTOM OF ERROR MESSAGE TABLE
19EF  NON DOS RET
19EF  ZERO D
19F0  GET A '?'
19F2  DISPLAY IT
19F5  ADD ERROR CODE DISPLACEMENT TO POINTER
19F6  GET ERROR MESSAGE
19F7  DISPLAY 1ST CHAR
19FA  NUMERIC CHECK
19FB  DISPLAY 2ND CHAR
19FE  POINT TO 'ERROR'
1A06  DISPLAY IT
1A0D  IF DE=(40EA)
1A0E  THEN POWER UP RESET
1A14  DISPLAY ' IN ' LINE #
1A17  LOAD A TAB(1)
1A19  RETURN TO BASIC COMMAND MODE ( 'READY'  ROUTINE)  PRINT A
1A1C  RETURN IF NON DOS
1A1F  TURN OFF CASSETTE
1A22  RETURN TO BEGINNING OF LINE OR CR.  ZERO A
1A25  POINT TO 'READY' MESSAGE
1A28  OUTPUT IT
1A2B  GET ERROR CODE
1A2E  TEST FOR SN ERROR
1A30  CALL IF SN ERROR
1A33  RETURN ADDRESS
1A39  GET AUTO FLAG ( AUTO=NON ZERO)
1A3D  JUMP FOR NON AUTO
      """""""""""""""""""""""""""""""""""
1A3F  GET CURRENT LINE #
1A43  OUTPUT LINE #
1A48  SEE IF LINE # OCCUPIED   (CARRY SET)  READY TO DISPLAY IF MAT
1A4B  GET A ASTERISK
1A4D  PRINT A SPACE OR AN ASTERISK
1A4F  GET A SPACE OTHERWISE
1A51  DISPLAY CORRECT CHARACTER
1A54  INPUT INTO BUFFER
1A58  (BREAK) SETS CARRY
      """""""""""""""""""""""""""""""""""
```

```
1A5A  TURN OFF AUTO AND JUMP BACK
      """""""""""""""""""""""""""""""""""""
1A60  GET LINE INCREMENT
1A63  ADD TO CURRENT LINE #
1A69  (BREAK) IF OVERSIZE LINE # RESULTS
1A6C  (BREAK) IF LINE # > 65529X
      """""""""""""""""""""""""""""""""""""
1A76  GET PROMPT
1A78  DISPLAY IT
1A7B  INPUT INTO BUFFER
1A7E  JUMP BACK IF BREAK
1A81  FIND FIRST CHARACTER
1A84  JUMP BACK IF NULL
1A88  CHECK FOR NUMERIC THEN SCAN PAST LINE #  (LINE # IS IN DE)
1A8B  SCAN
1A99  ENCODE INPUT INTO LEVEL II TOKENS
1A9D  FLAGS DECIDE IF COMMAND MODE
1A9E  ENCODED STATEMENT POINTER
1AA1  NON DOS RET
1AA4  IF COMMAND MODE?
1AA7  SAVE LINE #
1AA8  SAVE LINE LENGTH
1AA9  ZERO
1AAA  RESET RESUME + RETURN FLAG
1AAD  SCAN 1ST TOKEN
1AB1  SAVE LINE #
1ABF  SAVE THE LINE #
1AB5  SEARCH FOR A MATCHING LINE #  C=NONE  Z=FOUND
1AB9  IF NONE MAKE ROOM
1ABF  JUMP FOR MATCH
1AC6  LINE LENGTH
1AC7  HL= NEW END OF BASIC PROGRAM
1AC9  MAKE SURE BRAIN WON'T OVERFLOW
1ACD  STORE END OF BASIC PROGRAM POINTER
1AD0  HL=LINE TO BE MOVED
1AD3  HL=LINE # POINTER
1AD6  DE=LINE #
1AD9  HL=LINE POINTER  (TEXT)
1AE1  MOVE NEW LINE INTO PLACE
1AE6  LOOP TIL LINE IS MOVED
1AE9  FIX LINE POINTERS
1AEC  NON DOS RETURN
1AEF  HERE'S WHY EDITING A PROGRAM DESTROYS VARIABLES, ETC.
1AF2  NON DOS RETURN
1AF5  BACK TO THE FARM
      ******************************************
```

The LEVEL II ROM does not use the alternate registers.

```
1AFC  FIX THE LINE POINTERS ROUTINE
1B01  RETURN IF END OF OF BASIC
1B02  MOVE
1B03  PAST
1B04  THE NEXT LINE POINTER AND LINE #
1B06  CHECK FOR END OF LINE
1B08  DON'T QUIT TIL YOU SUCCEED
1B0A  GET THE JOB DONE
1B0E  RERUN ROADRUNNER CARTOON

      ****************************************

1B2C  SEARCH FOR MATCHING LINE # IN BASIC. DE=DESIRED LINE #.  GET
      FIRST LINE FROM (40A4)
1B2F  SAVE POINTER IN BC
1B31  CHECK FOR END OF BASIC PROGRAM  (STAY OUT OF JUNKYARDS)
1B35  RETURN IF END
1B36  POINT TO CURRENT LINE #
1B3B  HL=CURRENT LINE #
1B3C  MATCH? Z=YES  C=HL<DE
1B3D  NEXT LINE POINTER TO HL
1B44  RETURN IF MATCH FOUND (CARRY SET)
1B46  RETURN (NO SUCH LINE)
1B47  TRY NEXT LINE #

      ****************************************

1B49* NEW
1B4A  CLEAR SCREEN
1B4D  START OF BASIC PROGRAM
1B50  TROFF
1B53  TURN OFF AUTO
1B56  ERASE PROGRAM BY MAKING ITS LEADERS ZILCH
1B5A  RESET END OF PROGRAM POINTER
1B64  26 VARIABLES ARE
1B6C  SET TO SINGLE PRECISION HERE
1B6F  RESET RESUME FLAG
1B74  RESET ON ERROR STORAGE
1B77  RESET CONT LOCATION
1B7A  GET END OF MEMORY
1B7D
1B88  RESTORE DATA POINTER
1B83  GET END OF BASIC LOCATION
1B86  RESET VARIABLES POINTER
1B89  RESET ARRAYS POINTER
1B90  GET START OF STRING SPACE POINTER
1B95  SET STACK POINTER TO START OF STRING SPACE - 2
1B9A  SP=STRING SPACE POINTER
1BA1  SELECT VIDEO  FINISH PRINTING
1BA4  TURN OFF CASSETTE

      ****************************************

1BB3  PRINT '? ' AND INPUT FROM KEYBOARD  GO ON CR

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

```
1BC0  ENCODE BUFFER INTO TOKENS
1BC1  RESET FLAG
1BC6  HL=INPUT BUFFER POINTER
1BCC  GET 1ST CHAR FROM BUFFER
1BCD  IF SPACE
1BD3  IF STRING
1BD9  IF END OF LINE
1BE0  GET BUFFER CHAR
1BE4  CHECK FOR PRINT ABBREVIATION
1BE6  GET PRINT TOKEN
1BE8  SUBSTITUTE
1BEB  GET CHAR
1BEE  IF NON ALPHA NUMERIC
1BF2  IF NUMERIC
1BF5  SAVE INPUT POINTER-2
1BF6  LOAD RESERVED WORD LIST POINTER-1
1BF9  SAVE LINE LENGTH
1BFD  SAVE CONTINUATION ADDRESS
1BFE  END OF RESERVED WORD LIST TEST MASK
1C00  GET CHAR
1C03  IF NOT LOWERCASE
1C07  IF NOT LOWERCASE
1C09  CONVERT TO UPPERCASE
1C0C  SAVE CHAR
1C0E  POINT TO RESERVED WORD LIST
1C0F  CHECK FOR BEFINNING OF WORD (CHAR+80)
1C10  TRY AGAIN IF NOT
1C13  BUMP COUNT
1C14  GET RESERVED WORD CHAR
1C15  CHECK FOR END OF LIST
1C17  CONTINUATION IF WORD NOT FOUND
1C18  IS IT SAME AS BUFFER CHAR
1C19  NEXT RESERVED WORD IF NOT
1C27  IF NOT GOTO
1C31  MAKE UPPER CASE IF NEEDED
1C37  NEXT CHAR
      """""""""""""""""""""""""""""""""""""""
1C46  IF NOT ELSE
1C4C  IF NOT '
1C4E  LOAD COLON
1C50  NEXT
1C53  LOAD REM (1 FOR THE PRICE OF 3)

      ********************************************

1C90  RST 18H CODE

      ********************************************

1C96  RST 08H CODE

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

```
1CA1* FOR
1CFB  CHECK FOR 'STEP ' TOKEN
1CFD  DEFAULT VALUE OF 1
1CFF  IF NOT 'STEP '
1D4A  '(LINE NUMBER)' TRON USAGE
      """""""""""""""""""""""""""""""""""""""
1D5A  BASIC INTERPETER
1D60  REMOVE BIAS
1D62  CHECK FOR A TOKEN
1D62  JUMP IF NOT
1D6A  DOUBLE REMAINDER  (REQUIRED FOR 2 BYTE ADDRESSES)
1D6B  SAVE OFFSET
1D6C  IN B
1D6F  POINT TO VECTORS
1D72  LOCATE DESIRED ROUTINE ADDRESS
1D73  LOW BYTE TO C
1D75  HIGH BYTE TO B
1D76  SAVE ON STACK
      """""""""""""""""""""""""""""""""""""""
1D78  RST 16

      ******************************************

1D91* RESTORE
1D92  GET BEGINNING OF PROGRAM
1D96  RESTORE DATA POINTER

      ******************************************

1D9B  DISPLAY LINE NUMBER

      ******************************************

1DA0  CHECK FOR PAUSE
1DA2  WAIT FOR KEYSTROKE TO RESUME
1DA5  SAVE IT
1DA8  01= BREAK
1DA9* STOP
1DD4  SELECT VIDEO
1DD7  RETURN TO BEGINNING OF LINE

      ******************************************

1DAE* END

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

```
1DE4* CONT
1DEB  OUTPUT CN ERROR IF (40F7)=0

      ******************************************

1DF7* TRON    AF=TRON
1DF8* TROFF

      ******************************************

1E00* DEFSTR
1E03* DEFINT
1E06* DEFSNG
1E09* DEFDBL
1E0B  CHECK FOR SYNTAX  (LETTER NEEDED IN DEF---)
1E0E  GET ADDRESS OF SN ERROR
1E11  SAVE ON STACK FOR POSSIBLE USE
1E12  SN ERROR IF NO LETTER
1E13  CONVERT ASCII LETTER TO DISPLACEMENT INTO TABLE OF 26 LETTERS
1E15  SAVE DISPLACEMENT IN C
1E16  SAVE DISLACEMENT IN B
1E17  GET NEXT CHAR
1E18  IS IT -
1E1A  IF NOT DON'T USE A RANGE
1E1D  CHECK FOR LETTER
1E20  SN ERROR IF NOT
1E21  GET DISPLACEMENT
1E23  PUT IN B
1E24  GET TO NEXT CHAR
1E25  LOAD ENDING POINT
1E26  SUBTRACT BEGINNING POINT
1E27  SN ERROR IF VARIABLES REVERSED
1E28  BUMP COUNT (IN CASE VARIABLES SAME)
1E29  SAVE NEXT CHAR POINTER AND CLEAR SN ERROR VECTOR
1E2A  LOAD START OF VARIABLE DEFINITION AREA
1E2D  ZERO B
1E2F  DETERMINE ENDING POINT
1E30  SET VARIABLE TYPE FLAG
1E31  BUMP TABLE POINTER
1E32  REDUCE COUNT
1E33  LOOP TIL COUNT ZERO
1E35  RETURN NEXT CHAR POINTER
1E36  GET NEXT CHAR
1E37  IS IT A COMMA
1E39  IF NOT
1E3A  GET NEXT VARIABLE
1E3B  DEF---  AGAIN
      """""""""""""""""""""""""""""""""""""
1E3D  CHECK FOR LETTER IN (HL).  SET C IF NOT ELSE RESET

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

```
1E4F  GET CHAR
1E50  IS IT A PERIOD
1E53  GET PERIOD ADDRESS
1E57  JUMP IF PERIOD
1E5A  FOR RST 10
1E5B  INITIALIZE DE      DE=LINE # ON EXIT
1E5E  LOCATE 1ST CHAR AND NUMERIC CHECK
1E5F  RETURN IF NON NUMERIC
1E60  SAVE LOCATION
1E61  SAVE ASCII NUMERIC DIGIT
1E62  OVERSIZE LIMIT (65520)
1E65  PRE-FLIGHT
1E66  SN ERROR IF DE > 1998
1E69  HL=DE
1E6B  HL=(HL + DE)
1E6C  HL=(HL + DE) + (HL + DE)
1E6D  HL=(HL + DE + HL + DE) + DE
1E6E  HL=(HL+DE+HL+DE+DE) + (HL+DE+HL+DE+DE)=4*HL + 6*DE=010*DE
1E6F  RETREIVE ASCII NUMERIC DIGIT
1E70  CONVERT ASCII CODE TO #
1E72  SAVE # IN E
1E73  ZERO D  SO  DE=#
1E75  ADD # TO SUBTOTAL (HL)
1E76  DE=010*DE + #
1E77  RESTORE POINTER

      ********************************************

1E7A* CLEAR
1E7D  COMPUTE THE AMOUNT AS AN INTEGER
1E84  END OF MEM POINTER
1E8D  OM ERROR IF HL<DE
1E9C  LOAD STRING POINTER

      ********************************************

1EA3* RUN

      ********************************************

1EB1* GOSUB

      ********************************************

1EC2* GOTO    EVALUATE LINE #
1ED9  UL ERROR ENTRY POINT

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

```
1EDE* RETURN
1EEC  RG ERROR ENTRY POINT
1F05* DATA
1F07* REM
    * ELSE

      ****************************************

1F21* LET

      ****************************************

1F6C* ON
1F70* ON ERROR
1F80  IF UL ERROR
1F89  GET ERROR FLAG
1F8E  GET ERROR CODE
1F91  LOAD INTO E FOR ERROR ROUTINE
1F92  JUMP

      ****************************************

1FAF* RESUME  POINT TO ERROR FLAG
1FB3  CHECK IT
1FB4  RW ERROR IF ZERO

      ****************************************

1FF4* ERROR
2003  UE ERROR ENTRY POINT

      ****************************************

2008* AUTO
200B  SAVE DEFAULT VALVE OF 10
2019  SAVE LINE # INCREMENT
2022  IF SN ERROR
2025  CHECK FOR ZERO INCREMENT
2028  FC ERROR IF Z
202B  SAVE INCREMENT
202E  SET AUTO FLAG
2036  BACK TO THE FARM

      ****************************************

2039* IF
2044  IF   THEN
2060  IF NOT   ELSE

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

```
2067* LPRINT    SELECT LPRINTER FOR OUTPUT
207B  IF EXPRESSION ISN'T INTEGER
207F  POINT TO DISPLAY
2082  COMPUTE LOCATION
2083  SAVE IT
2089  UPDATE CURSOR
      """""""""""""""""""""""""""""""""""""

206F* PRINT
2076* PRINT @    EVALUATE EXPRESSION
2093* PRINT #     WRITE LEADER AND SYNC BYTE
20A5  IF PRINT USING
20AA  IF PRINT TAB(
20B0  IF COMMA
20B5  IF SEMI COLON
20BE  IF STRING
20FE  USED TO RETURN TO BEGINNING OF LINE AND ZERO A
2137* TAB(

      *****************************************

2169  OUTPUT DEVICE T, V, P -1, 0, 1
      """""""""""""""""""""""""""""""""""""""

216D  TURN OFF CASSETTE IF NEEDED
2171  SELECT VIDEO
      """""""""""""""""""""""""""""""""""""

2178  'REDO ' MESSAGE STRING

      *******************************************

219A* INPUT
219D  REDO
21A9* INPUT #
21AD  250 BYTES LIMIT
21AF  POINT
21B2  READ A BYTE
21B5  INTO THE BUFFER
21B7  CR YET?
21BB  LOOP BACK
21BE  END OF FILE MARKER
21C0  TURN OFF TAPE
      """""""""""""""""""""""""""""""""""""

21EF* READ
      """""""""""""""""""""""""""""""""""

227C  'EXTRA IGNORED ' STRING
      """""""""""""""""""""""""""""""""""""""

22B6* NEXT

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

```
2337  EVALUATE EXPRESSION  PUT IN ARITH : POINT HL TO ADDRESS OF
      1ST CHAR.  TERMINATE WITH 00 OR , OR ) OR :
      """""""""""""""""""""""""""""""""""""""""

2490  INTEGER DIVIDE. OUTPUT IN SINGLE PRECISION
      """""""""""""""""""""""""""""""""""""""""

249F* +
24A2  MO ERROR IF Z
24A5  IF NUMERIC
24A8  CHECK FOR LETTER
24AB  IF LETTER
24B0  IF +
24B4  IF .
24B9  IF -
24BE  IF QUOTE
24C3  IF NOT
24C8  IF &
      """""""""""""""""""""""""""""""""""""""""

24CD  IF NOT ERR
24CF* ERR
      """""""""""""""""""""""""""""""""""""""""

24DB  IF NOT ERL
24DD* ERL
      """""""""""""""""""""""""""""""""""""""""

24E9  IF NOT VARPTR
24EB* VARPTR
      """""""""""""""""""""""""""""""""""""""""

2501  IF USR
2506  IF INSTR
250B  IF MEM
2510  IF TIME$
2515  IF POINT
251A  IF INKEY$
251F  IF STRING$
2524  IF FN

      *****************************************

2532* -

      *****************************************

2540  ASCII VARIBLE TO ARITH :  PUT VARIABLE IN ARITH AND SET NTF.
      POINT HL TO 1ST CHAR. RETURNS WITH HL POINTING TO NEXT CHARACTER
      AFTER VARIABLE.

      *****************************************

25D9  RST 20H
25F7* OR
      """""""""""""""""""""""""""""""""""""""""

25FD* AND

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

```
2608* DIM
260D  LOCATE OR CREATE IF NOT FOUND VARIABLES: POINT HL TO 1ST CHAR OF
      VARIABLE. RETURNS WITH DE POINTING TO VARIABLE'S ADDRESS AND HL
      POINTING TO THE NEXT CHARACTER AFTER VARIABLE.
2612  CHECK FOR LETTER
2615  SN ERROR IF C
261B  IF NUMERIC
2620  IF NOT A LETTER
2624  IF NUMERIC
2626  CHECK FOR LETTER
2629  IF NUMERIC
262E  SET RETURN ADDRESS TO 2652 BEFORE GOING
2633  IF % (INTEGER D=2)
2637  IF $ (STRING D=3)
263B  IF ! (SINGLE D=4)
2640  IF # (DOUBLE  D=8)
273D  BS ERROR ENTRY POINT

      ****************************************

27C9* MEM
27CB  ZERO NTF
27CE  CALL FRE ROUTINE

      ****************************************

27D4* FRE   GET FREE SPACE POINTER
27DD  IF NOT A STRING
27E5  GET START OF STRING SPACE
27E9  GET END OF STRING SPACE
27F1  HL=END - START

      ****************************************

27F5* POS   GET CURSOR POSITION
27FB  RETURN VIA USR RENTRY  HL IS POSITION

      ****************************************

27FE* USR
2801  NEXT CHAR
2806  SET REENTRY POINT
280A  GET NTF
2810  CALL AF STRING
2815  (408E) CONTAINS ENTRY POINT TO USR ROUTINE
      """""""""""""""""""""""""""""""""""""""
2831  ID ERROR ENTRY POINT

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

```
2836* STR$
2866  QUOTE
2891  NTF = STRING
28A1  ST ERROR ENTRY POINT

      ****************************************

28A7  OUTPUT A MESSAGE; POINT HL TO STARTING ADDRESS OR STRING; MARK
      END WITH A 00 OR 22  OUTPUT DEVICE SELECTED BY (409C).
      UPDATES LINE CURSOR POSITION

      ****************************************

2A03* LEN

      ****************************************

2A0F* ASC

      ****************************************

2A1F* CHR$

      ****************************************

2A2F* STRING$

      ****************************************

2A61* LEFT$

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

```
2A91* RIGHT$

      ****************************************

2A9A* MID$

      ****************************************

2AC5* VAL

      ****************************************

2AEF* INP  GET PORT ADDRESSES
2AF2  LOAD IT
2AF5  INPUT FROM CORRECT PORT
2AF8  rETURN VIA USR CODE

      ****************************************

2AFB* OUT  GET BYTE
2AFE  OUTPUT IT
      """"""""""""""""""""""""""""""""""""

2B01* STEP
2B02  COMPUTE VALUE OF EXPRESSION
2B06  CONVERT IT TO INTEGER
2B0B  MSB TO A
2B0C  CHECK FOR OVERFLOW
      """"""""""""""""""""""""""""""""""""

2B0E  GET PORT #
2B11  SET PORT # FOR INPUT
2B14  SET PORT # FOR OUTPUT
2B17  SYNTAX CHECK
2B1C  COMPUTE VALUE
2B1F  CONVERT TO INTEGER
2B22  FC ERROR IF OVERFLOW

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

Notes on the EDITOR ASSEMBLER

4113	Top of memory pointer
4115	Start of memory pointer
41C3	Start of symbol table pointer
4301	Keyboard driver entry address pointer
4309	Video driver address
4311	Lprinter driver address
45AA	Lprinter driver (patch your printer here)
4905	Command table (have a feast)
4925	B command (go someplace new)

```
2BC6* DELETE

      ****************************************

2BF5* CSAVE  WRITE LEADER AND SYNC BYTE
2BF8  EVALUATE CHAR FOLLOWING CSAVE
2BFF  D3= HEADER FOR BASIC TAPE
2C04  WRITE ONCE
2C04  WRITE TWICE
2C07  SEND FILE NAME TO TAPE
2C0B  START OF PROGRAM
2C0F  END OF PROGRAM
2C12  GET A BYTE
2C13  POINT TO NEXT
2C14  SEND BYTE
2C17  DONE?
2C18  LOOP TIL DONE
2C1A  OFF CASSETTE

      ****************************************

2C1F* CLOAD
2C40  TROFF + NEW
2C47  3 COUNT FOR HEADER
2C49  GET A BYTE
2C4C  D3?
2C4E  IF NOT RESET COUNT AND TRY AGAIN
2C50  LOOP TIL HEADER FOUND
2C52  GET FILE NAME (1 BYTE)
2C57  IF ONLY CLOAD
2C59  IS TAPE FILE NAME CORRECT
2C5A  GO BAD IF NOT
2C5C  BEGINNING OF BASIC ID  HL
2C5F  LOAD COUNT FOR 3 ZEROS WHICH DETERMINE END OF BASIC TAPE
2C61  READ A BYTE
2C64  SAVE IT FOR LATER
2C65  SAME AS ORIGINAL?
2C66  SAME LOCATION?
2C67  NZ MEANS BAD
2C69  PUT BYTE INTO MEMORY NOTE: VALIDITY IS CHECKED BEFORE PLACEMENT
2C6A  CHECK FOR OM ERROR
2C6D  RETREIVE BYTE
2C6E  CHECK FOR A ZERO
2C6F  POINT TO NEXT MEMORY LOCATION
2C70  IF NOT A ZERO (END OF LINE) RESETS ZERO COUNT
2C72  TWINKLE TWINKLE (END OF LINE)
2C75  LOOP TIL END OF PROGRAM (3 ZEROS)
2C77  UPDATE END OF PROGRAM POINTER
2C74  POINT TO 'READY' MESSAGE
2C7D  DISPLAY IT
2C80  TURN OF CASSETTE
2C83  BEGINNING OF BASIC TO HL
2C86  SAVE IT
2C87  RETURN TO FARM AFTER FIXING ADDRESSES
      """""""""""""""""""""""""""""""""""""""
```

```
2C8A  POINT TO 'BAD' MESSAGE
2C93  GIVE THE NEWS
2C90  BACK TO THE FARM
2C93  DISPLAY FILE LETTER
2C96  3 COUNT FOR END OF PROGRAM TEST
2C98  READ BYTE
2C9B  ZERO?
2C9C  IF NOT RESET COUNT TRY AGAIN
2C9E  LOOP TIL 3 ZEROS
2CA0  SEARCH FOR NEXT PROGRAM
2CA3  TRY NEXT PROGRAM
      """""""""""""""""""""""""""""""""""""
2CAA* PEEK   EVALUATE EXPRESSION AS INTEGER
2CAD  GET BYTE
2CAE  RETURN VIA USR CODE

      ******************************************

2CA5  'BAD ' MESSAGE STRING

      ******************************************

2CB1* POKE  COMPUTE ADDRESS
2CB4  SAVE IT
2CB5  CHECK SYNTAX
2CB6  NEXT
2CB7  COMPUTE VALUE OF OPERAND
2CBA  RETRIEVE ADDRESS
2CBB  POKE IT IN

      ******************************************

2CBD* USING
      """""""""""""""""""""""""""""""""""""
2E49  PRINT A PLUS IF D NON ZERO

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

```
2E60* EDIT  GET LINE
2E64  DE=LINE #
2E66  PUT LINE # IN STORAGE
2E6A  SEARCH FOR MATCHING LINE #
2E6D  UL ERROR IF NOT FOUND (NC)
2E70  ADDRESS OF LINE TO HL
2E72  POINT TO LINE #
2E74  PUT LINE# IN BC
2E78  SAVE LINE # ON STACK
2E79  CONVERT LINE TO ASCII
2E7E  OUTPUT LINE #
2E81  OUTPUT BLANK
2E86  POINT TO BUFFER
2E89  SEND CURSOR
2E8E  SAVE BUFFER POINTER
2E91  C=LENGTH OF LINE
2E95  SEARCH FOR END OF LINE
2E98  ZERO A
2E99  ZERO D
2E9B  GET A KEY
"""""""""""""""""""""""""""""""""""
2EBB  (BACK SPACE)
2EC0  (ENTER)?
2EC7  (SPACE)?
2EC9  UPPER CASE?
2ECB  CHANGE TO LOWERCASE
2ECF  Q UIT?
2ED4  L IST LINE?
2ED9  S EARCH?
2EDD  I NSERT?
2EE2  D ELETE?
2EE7  C HANGE?
2EEC  E ND?
2EF1  X TRA?
2EF6  K ILL?
2EFA  H ACK?
2EFF  A GAIN?
2F01  NOT VALID CHARACTER IF NZ GET ANOTHER
2F07  START OVER
"""""""""""""""""""""""""""""""""""
```

BASIC RAM STORAGE FORMAT

LSB	Address of
MSB	next line
LSB	Line # in
MSB	binary form
XX ... XX	Line
00	End of line marker
.	
.	
.	
00 00	End of file marker

```
2F0A  SPACE ROUTINE
"""""""""""""""""""""""""""""""""""""
2F16  KILL ROUTINE
2F1C  SEARCH ROUTINE
2F1D  GET THE CHARACTER
"""""""""""""""""""""""""""""""""""""
2F40  OUTPUT AREA POINTED BY HL
"""""""""""""""""""""""""""""""""""""
2F4A  DELETE ROUTINE
2F4D  PRINT '!'
2F5F  PRINT '!'
"""""""""""""""""""""""""""""""""""""
2F65  CHANGE ROUTINE
2F68  GET CHARACTER
2F6C  PRINT IT
2F71  REDUCE COUNT
2F72  LOOP TIL DONE
"""""""""""""""""""""""""""""""""""""
2F75  HACK ROUTINE
2F78  XTRA ROUTINE
2F7D  INSERT ROUTINE
2F84  BACK SPACE
2F88  CR?
2F8D  ESCAPE?
"""""""""""""""""""""""""""""""""""""
2FE0  LEAVE EDIT MODE AFTER DISPLAYING LINE
"""""""""""""""""""""""""""""""""""""
2FF6  QUIT ROUTINE
```

****

BASIC TAPE FORMAT

LEADER	256 zeros followed by an A5 sync byte
D3 D3 D3	BASIC header
XX	File name
LSB	Next line's address
MSB	Pointer
LSB	Line
MSB	#
XX ... XX	Line
00	End of line marker
.	
.	
.	
00 00	End of file markers

3000-37DD RESERVED THERE IS NOTHING HERE NO MEMORY AT ALL
37DE DOS COMMUNICATION STATUS ADDRESS
37DF DOS COMMUNICATION DATA ADDRESS
37E0 INTERRUPT LATCH ADDRESS
37E1 DISC DRIVE SELECT LATCH ADDRESS
37E2 CASSETTE DRIVE LATCH ADDRESS
37E8 LPRINTER PORT ADDRESS
37EC FLOPPY DISC CONTROLLER ADDRESS

----------KEYBOARD MEMORY--------

	COL # =	0	1	2	3	4	5	6	7
3801	ROW 0	@	A	B	C	D	E	F	G
3802	ROW 1	H	I	J	K	L	M	N	O
3804	ROW 2	P	Q	R	S	T	U	V	W
3808	ROW 3	X	Y	Z					
3810	ROW 4		!	"	#	$	%	&	'
		0	1	2	3	4	5	6	7
3820	ROW 5	(	)	*	+	<	=	>	?
		8	9	:	;	,	-	.	/
3840	ROW 6	ENT	CLS	BRK	↑	↓	←	→	SPS
3880	ROW 7	SHIFT							

****

3C00-3FFF VIDEO MEMORY

****

```
4000  RST   8
4003  RST  10
4006  RST  24
400C  RST  32
400C  RST  40
400F  RST  48
4012  RST  56
------------KEYBOARD CONTROL BLOCK
4015  DEVICE TYPE
4016  DRIVER ADDRESS (INTERCEPT HERE FOR DEBOUNCE)
4018  0
4019  0
401A  0
401B  'K '
401C  'I '
------------VIDEO CONTROL BLOCK
401D  DEVICE TYPE
401E  DRIVER ADDRESS
4020  CURSOR POSITION IN MEMORY (2 BYTES)
4022  CURSOR CHARACTER
4023  'D '
4024  'O '
------------LPRINTER CONTROL BLOCK
4025  DEVICE TYPE
4026  DRIVER ADDRESS
4028  #  LINES PER PAGE KEPT HERE
4029  CURRENT LINE # PRINTER IS ON
402A  0
402B  'P '
402C  'R '
-----------------------------------
4036-403C  7 BYTE WORK AREA FOR KEYBOARD ROUTINE
-----------------------------------
403D  PRINT SIZE FLAG  0=64 CHAR  8=32 CHAR   ALSO   USED IN TAPE
       OUTPUT TO PREVENT RESETING SIZE DURING AN OUT 255
```

```
          ----------TIME$ STORAGE AREA
4040  25 MS TICKS
4041  SECONDS
4042  MINUTES
4043  HOURS
4044  YEAR
4045  DAY
4046  MONTH
408E  ENTRY POINTER TO USR ROUTINES
4093  INP ROUTINE
4094  PORT #
4096  OUT
4097  PORT #
4099  INKEY$ STORAGE
409A  ERROR CODE STORAGE FOR RESUME USE
409B  PRINTER CARRIAGE POSITION
409C  DEVICE TYPE FLAG  -1=TAPE  0=VIDEO  1=LPRINTER
409D  PRINT# USE
40A0  START OF STRING SPACE POINTER
40A2  CURRENT LINE BEING PROCESSED
40A4  START OF BASIC PROGRAM POINTER
40A6  LINE CURSOR POS  USED FOR TAB
40A7  INPUT BUFFER POINTER
40AA  LSB OF SEED FOR RND
40AB  LSB OF SEED FOR RND   ALSO USED IN RANDOM
40AC  MSB OF SEED FOR RND
40AF  NTF (NUMBER TYPE FLAG)  2=INTEGER  3=STRING  4=SINGLE  8=DOUBLE
40B1  TOP OF BASIC MEMORY POINTER
40B3  STRING WORK AREA POINTER
40B5--STRING WORK AREA
40D6  ~~MEMORY SIZE~~ ERRATA 4006 SHOULD BE NEXT STRING STORAGE POINTER.
40DC  DIM USE
40DE  PRINT USING
40DF  ENTRY POINT STORAGE FOR SYSTEM TAPES
40E1  AUTO FLAG  0=NOT AUTO  ELSE AUTO
40E2  CURRENT LINE #
40E4  AUTO INCREMENT SIZE
40E6  ENCODED STATEMENT POINTER
40E8  S T A C K  P O I N T E R  POINTER
40EA  USED DURING RESUME
40EC  EDIT LINE #
40EE  USED DURING RESUME
40F5  LAST LINE # EXECUTED
40F7  USED TO CONT
40F9  SIMPLE VARIABLES POINTER
40FB  ARRAYS POINTER
40FD  FREE SPACE
40FF  DATA POINTER
          ----------VARIABLE TYPE DECLARATION TABLE
4101-411B  2=INTEGER 4=SINGLE 8=DOUBLE  3=STRING
411B  TRON FLAG  0=TROFF
```

```
----------ARITH-------------
      INTGER   SINGLE    DOUBLE
411D                     LSB
411E                     LSB
411F                     LSB
4120                     LSB
4121   LSB       LSB     LSB
4122   MSB       LSB     LSB
4123             MSB     MSB
4124             EXP     EXP
-------------------------------
----------ARITHEX-----------
4127   LSB       LSB     LSB
4128   MSB       LSB     LSB
4129   MSB       MSB     LSB
412A             EXP     LSB
412B                     LSB
412C                     LSB
412D                     MSB
412E                     EXP
-------------------------------
4130  LINE # WORK AREA POINTER
----------DOS ENTRY POINTS--
4152  CVI
4155  FN
4158  CVS
415B  DEF
415E  CVD
4161  EOF
4164  LOC
4167  LOF
416A  MKI$
416D  MKS$
4170  MKD$
4173  CMD
4176  TIME$
4179  OPEN
417C  FIELD
417F  GET
4182  PUT
4185  CLOSE
4188  LOAD
418B  MERGE
418E  NAME
4191  KILL
4194  &
4197  LSET
419A  RSET
419D  INSTR
41A0  SAVE
41A3  LINE
----------INPUT BUFFER AREA------
41E8-42E8   NOTE STACK POINTER IS 4288 FOR SYSTEM
-----------------------------------
42E9  BASICALLY ONLY THE BEGINNING!
```