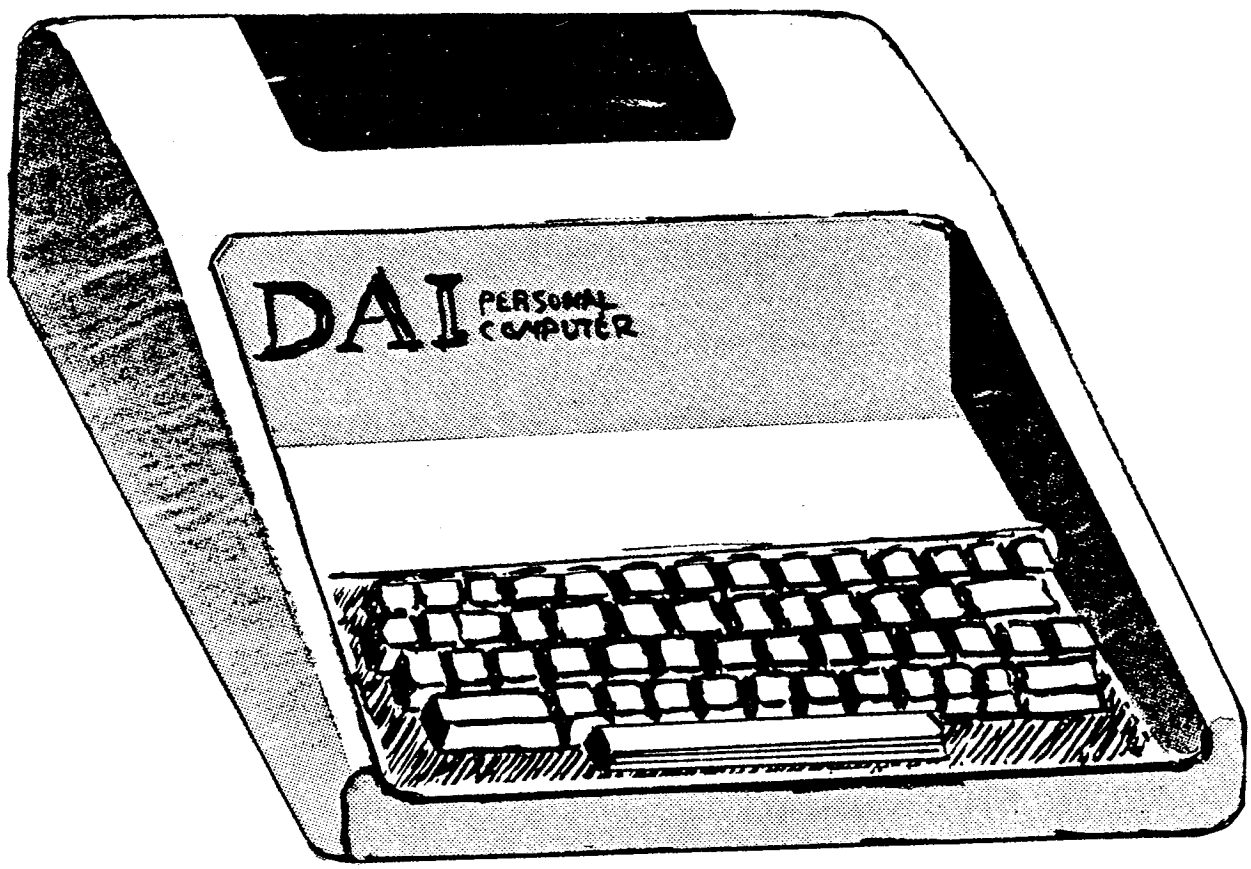
DAI PERSONAL COMPUTER



firmware manual

by B.J.Boerrigter

PREFACE

Well, the work is done now. After 2 years of investigation and typing the DAI personal computer software manual is ready. A countless number of hours resulted in about 500 pages of Assembler listings.

Going through the DAI firmware will show you all the good (and less good) features of the DAI personal computer. Many routines can be used in your own machine language programs. Therefore, entry and exit conditions are added to the routines.

Please don't blame me if you may find some wrong interpretations of parts of the firmware. It is sometimes very difficult to trace the ideas of the one who wrotes the program.
If you may have any comments, I woud be very grateful if you could transmit these to me in writing. It may result in updates of the manual, published seperately or in the DAInamic Newletter.

I would like to acknowledge the DAInamic Users Club for the support with all the information they had available.
I would like to especially thank Mr. Gordon Wassermann for using the results of his investigations of the firmware.
The chapter 'Updates BASIC V1.1' is a result of the compare Jos Schepens did on both BASIC versions.

Jan Boerrigter - September, 1982.

Published by:
'Micro Service'
Fabritiusstraat 15,
6174 RG Sweikhuizen
The Netherlands.

SUMMARY:
========

1. MEMORY MAP.

2. FIRMWARE MODULES:

C000 -	C6BF:	Math. utilities.
C6C0 -	C718:	Bank switching.
C719 -	D100:	BASIC handler.
D101 -	D194:	String handler.
D195 -	D23C:	Heap handler.
D23D -	D8FA:	I/O handler.
D8FB -	D9F4:	Interrupt handler.
D9F5 -	DAD3:	Error handler.
DAD4 -	DDD0:	Print routines.
DDD1 -	DE01:	Encoding service routines.
DE02 -	DEB4:	Single/double byte utilities.
DEB5 -	0ECAA:	BASIC execution/run-time module.
0ECAB -	0EFFF:	List handler.
1E000 -	1EE6D:	Math. package.
1EE6E -	1EFFF:	Sound module.
2E000 -	2EBF3:	Screen driving package.
2EBF4 -	2EFFF:	Editor package.
3E000 -	3E9FF:	Encoding package.
3EA00 -	3EFFF:	Utility package.

'Gaps' in these modules are filled with parts of routines from other modules.

3. UPDATES BASIC V1.1.

TABLES:
=======

CBBF:	Strings BASIC commands.
CD8B:	Pointers to strings BASIC commands (for LIST).
CF02:	Pointers to execution routines BASIC commands.
CF86:	Table prefixes unitary operations.
CF91:	Table binary operators.
CFD8:	Table unitary operators.
CFE6:	Table strings BASIC functions.
DA94:	Pointers to strings error messages.
DB6F:	Strings machine messages.
DC1C:	Strings error messages.
0E9F0:	Function indirection table.
0ECFB:	Pointers LIST handling routines.
2E030:	Screen constants.
3E8C5:	ASCII tables.

DAI pc MEMORY MAP

BASIC V1.0/V1.1 Revision 5.1 Date: 18.9.82

INTERRUPT VECTOR ROUTINES: 0000 - 003F

0000-07 Interrupt vector routine 0:
Used by Utility (LOOK).

0008-0F Interrupt vector routine 1:
Used by Utility and encoding Basic.
RST 1 + data: Switch to ROM-bank 3.

0010-17 Interrupt vector routine 2:
Used by stack interrupt.

0018-1F Interrupt vector routine 3:
Used by sound interrupt.

0020-27 Interrupt vector routine 4:
Used for math. routines.
RST 4 + data: Switch to ROM-bank 1.

0028-2F Interrupt vector routine 5:
Used for screen handling routines.
RST 5 + data: Switch to ROM-bank 2.

0030-37 Interrupt vector routine 6:
Used for keyboard service routines.

0038-3F Interrupt vector routine 7:
Used to flash the cursor.

Interrupt vector routines:

```
00  NOP
E5  PUSH H
2A  LHLD:
..  ) vector address location
..  ) see (#0062-#0071).
E9  PCHL
00  NOP
00  NOP
```

BANK SWITCHING AREA: 0040 - 0046

0040 !POROM:) Memory of last outputs to output ports.
!POR1M:) Duplicate of (#FD06).
POROM:)

0041/42 RSWK1: Save PSW during ROM bank switching.

0043/44 RSWK2: Save HL during ROM bank switching.

0045/46 Spare.

UTILITY WORK AREA: 0047 - 0061

0047 Store EI/DI instructions after using LOOK the first time (No clear occurs).

0048/49 High address trace window.

004A/4B Low address trace window.

004C-4F Store current instruction if LOOK is used, preceeded by EI. In case of a RST-instruction is stored: RSTx/data x; RST0. In case of an EI instruction is stored: EI, NOP, next instruction.

0050 Flag for Look initialisation:
#FF: init. Look, else: #00.

0051/52 IADR: I: Address current instruction.

0053 AFSAV: A: Contents A after execution of I.

0054 F: Idem status flags.

0055 BCSAV: B: Idem B register.

0056 C: Idem C register.

0057 DESAV: D: Idem D register.

0058 E: Idem E register.

0059 HLSAV: H: Idem H register.

005A L: Idem L register.

005B/5C SPSAV: S: Idem stackpointer.

005D/5E PCSAV: P: Address next instruction to be executed.

005F TICIM: M: Current interrupt mask.
Duplicate of (#FFF8)

0060 T: Value TICC control word.
(#FC after Z2).

0061 G: Value GIC control word.
(#1B after Z2).

INTERRUPT VECTOR ADDRESSES: 0062 - 0071

0062/63 I0USA: Vector address RST 0: set by UT (Z2): 3#EB5D.

0064/65 I1USA: Vector address RST 1: utility/encode: #C70E.

0066/67 I2USA: Vector address RST 2: stack interrupt: #D9E2.

0068/69 I3USA: Vector address RST 3: sound interrupt: #D755.

006A/6B I4USA: Vector address RST 4: math. restart: #C6C0.

006C/6D I5USA: Vector address RST 5: screen restart: #C6FD.

006E/6F I6USA: Vector address RST 6: keyb. int. serv: #D578.

0070/71 I7USA: Vector address RST 7: clock interrupt: #D9A9.

By changing the vector addresses, other interrupt routines can be used.

SCREEN VARIABLES: 0072 - 00CF

Character mode variables:

0072/73 CURSOR: Cursor position address.

0074 CURTY: Cursor type:
#00: cursor flashes in colour.
#01: cursor alternates between actual character and contents #0075.

```
0075      CURIN:  Cursor information:
                    If type = 0: Mask which is EXOR'ed with
                                 the colour byte for that
                                 character to flash it.
                    If type = 1: Cursor alterates between
                                 actual character and this
                                 information.
0076/77   CURSV:  Contents screen RAM location indicated by
                  the cursor:
                    #0076 contains the colour byte,
                    #0077 contains the data.
0078/79   LNSTR:  Address line mode byte of currently used
                  line of the screen RAM.
007A      LNEND:  Lobyte of end of cursor line.
                  Used to check if end of line is reached.
007B      LCONT:  Number of extended lines.
007C      COLMT:  ) colours for colour    #80+X3
007D              ) registers COLORT      #90+X2
007E              )                       #A0+X1
007F              )                       #B0+X0

          Variables set to describe the current state of
          the screen:

0080/81   SCREEN: Points to first byte of screen RAM (#BFFF).
0082/83   SCTOP:  Points after header (#BFEF).
0084/85   FFB:    First free byte in this mode.
0086/87   GRR:    Points to top of rolled area. Contains the
                  line mode byte of the line where split mode starts.
0088/89   GRE:    Points after end of graphics area.
008A/8B   CHS:    Points to start of character area.
008C/8D   GAE:    Unsplit: End archive area.
          CHE:    Split: After end of character area.
008E/8F   SCE:    End of screen (after trailer).
0090/91   GTE:    End area used splitting mode.
0092/93   GAS:    Unsplit: start archive area.
          GTS:    Split: start temporary save area.
0094/95   GRC:    Number of blobs horizontally in mode.
0096      GRL:    Number of lines of graphics in mode.
0097      GAL:    Number saved lines of graphics.
0098      GXB:    Number of bytes/line this mode.
0099/9A   GREO:   Previous end of graphics.
009B/9C   CHSO:   Previous start characters:
                  Was split: previous mode byte of 1st text line.
                  Was graphics: Previous last COLORT-byte.
009D      SMODE:  Current screen mode (updated after mode
                  changed):
                    #00 mode 1          #08 mode 5
                    #01 mode 1A         #09 mode 5A
                    #02 mode 2          #0A mode 6
                    #03 mode 2A         #0B mode 6A
                    #04 mode 3
                    #05 mode 3A         #10 during init.
                    #06 mode 4
                    #07 mode 4A         #FF mode 0
                      bits 4-7 are ignored;
                      #0C,0D,0E,0F are inhibited.

          Graphics mode variables (From #00A2-#00B5 also
          used by the EDIT mode):

009E      COLMG:  ) colours for colour #80+X3
009F              ) registers COLORG   #90+X2
00A0              )                    #A0+X1
00A1              )                    #B0+X0
00A2      SCVR:
00A3-AA   SCXBUF: Buffer used to hold contents of an 8 bit
                  field during 16 colour updates.
00AB      SBGOU:  Flags if colour is being carried out to
                  next field.
00AC      SBGOC:  Colour being carried out.
00AD-B4   COLS:   Buffer for impossible requests.

          !Edit variables:

00A2/A3 !EBUFR:   Address start EDIT buffer.
00A4/A5 !EBUFN:   Address end of text in EDIT buffer.
00A6/A7 !EBUFS:   End available space in EDIT buffer.
00A8    !EWINX:   Offset of left side of window.
00A9/AA !EWINY:   Offset of top of window from start buffer.
00AB    !ECURX:   X-offset of cursor in document (current
                  cursor position in text line).
00AC/AD !ECURY:   Y-offset of cursor in document (count of
                  current cursor line).
00AE/AF !CURPT:   Pointer to cursor position in buffer.
00B0/B1 !CURLS:   Pointer to line mode byte of cursor line on screen.
00B2/B3 !CURLB:   Pointer to start of cursor line in buffer.
00B4/B5 !TABTP:   Address tab position table.

          Line drawing variables:

00B5/B6   DELTA:  Amount to add into count.
00B7/B8   RT:     Count.
00B9/BA   COR:    Adjustments for long sectors.
00BB/BC   SECT:   Lower of 2 possible sector lengths.
00BD      SECTC:  Number of sectors.
00BE      TRIM:   Amount to trim off last sector.
00BF      DIRN1:  Set if Y-direction is negative.
00C0      DIRN2:  Set if swap X,Y directions.
00C1      ANIM:   Set if animate in 4 colour mode.
00C2/C3   FCOLR:  Details of colour required.


00C4/C5   ASMKRM: Address memory management routine (#CA01).
                  Checks available RAM space.
00C6/C7   AESTOP: Address emergency stop routine (#CA25).
                  Return-routine for 'Out of space for mode'.
00C8-CF           Spare.


MATH. WORKING AREA: 00D0 - 00FF
===============================

00D0/D1   EVECT:  Pointer to table with error routines (#C7F2).
00D2/D3   AGETC:  Pointer to input routine (#DDE0).
00D4      MVECA:  Math. chip flag: offset of start HW/SW vector;
                  (offset for RST 4 restart routines):
                    #00  No math. chip.
                    #7B  math. chip present.
```

```
00D5-D8  FPAC:    ) Arithmetic FPT/INT accumulator.
         IAC:     )
00D9     SF:      Subtraction flag.
00DA     OP4:     Operand 4th byte.
00DB     OP3:     Operand 3rd byte.
00DC     OP2:     Operand 2nd byte.
00DD     OP1:     Operand 1st byte.
00DE     EXPDF:   Difference in exponents for last FPT
                  add/sub operations.

         Work area for math. operations:

00DF/E0  FWORK:   ) Also used for data save during stack operations.
         XPRAS:   )
00E1/E2  XPHLS:   )

         FPOLY variables (RAM shared with SQRT):

00E3-E6  XN:      Running power of X (X^K).
00E7-EA  XK:      Power multiplier (X^J).
00EB-EE  SUM:     Running sum.

         SQRT variables (RAM shared with FPOLY):

00E3-E6  F:       Mantissa.
00E7-EA  P:       Polynomial approximation.

         EXP variables (RAM shared with TRIG, INVTRIG):

00EF     SIGN:    Input sign.

         TRIG variables (RAM shared with EXP, INVTRIG):

00EF-F2  FTWRK:   Work location for TAN.

         Inverse TRIG variables (RAM shared with EXP, TRIG):

00EF-F2  FATZX:   Z,X. Used by ATAN, ASIN, ACOS.

         Number input variables:

00E3-E6  ICBWK:   Number to add for each digit.

         Number output variables:

00E3-F1: DECBUF:  Decimal output buffer.
                     MAXSIG: #0A: Max. possible significant figures.
                     FPTSIG: #06: Number of significant digits for FPT.
00E4     DECBS:   Sign.
00E5     DECBD:   Decimal point.
00E6-F0  DECBF:   Digits, most significant one in 00E6.
00F1     DECBE:   Exponent.
00F2/F3  DECBP:   Buffer pointer.
00F4-FF           Spare variable space.
```

BASIC VARIABLES: 0100 - 02EB

```
         User state:

         Following are saved by soft break: (SFRAME = SYSTOP - SYSBOT)

0100/01  SYSBOT:  ) Start of current line. Points to first
         CURRNT:  ) byte of line number.
0102/03  BRKPT:   Start of current command.
0104/05  LOPVAR:  Points to current loop variable. Points to
                  position of variable in symbol table.
                     #00 if no running loop.
0106     LSTPF:   Flag for integer/fpt loop and
                  implicit/explicit step.
                     bit 0: 0 = implicit step.
                            1 = expicit step.
                     bit 7: 0 = FPT loop variable.
                            1 = INT loop variable.
0107-0A  LSTEP:   Step value if explicit.
010B-0E  LCOUNT:  Loop iteration count.
010F-10  LOPPT:   Pointer to start address loop.
0111/12  LOPLN:   Pointer to start loop line.
0113/14  STKGOS:  Stack level at last GOSUB.
                     #00 if no active call.
0115     SYSTOP:  )
        (STRFL:   ) Trace/step flag together)
         TRAFL:   ) Trace flag (#FF if set).
0116     STEPF:   Step flag (#FF if set).
0117     RDIPF:   Flag set while running input (set: #FF).
0118     RUNF:    Flag set while running program.
                  (Previous 2 bytes must be consecutive)

         Runtime scratch area:

0119/1A  GSNWK:   Scratch area for GOSUB/NEXT (2 bytes).
                  Points to destination address last GOSUB.
         LISW1:   Startaddress of listed area.
         COLWK:   Scratch area for SCOLG, SCOLT (4 bytes).
                  Contains last selected COLORT/COLORG values.
011B/1C  LISW2:   End address listed area.

         Save area for restart on error:

011D/1E  ERSSP:   Stack pointer.
011F-21
0122     ERSFL:   Set if encoding a stored line (set: #01).

         Data/read variables:

0123     DATAC:   Offset of next character to encode.
0124/25  DATAP:   Pointer to address current data line.
        !DATAQ:   Pointer after current data line.
0126     CONFL:   Set if there is a suspended program (set: #01).
0127/28  STACK:   Current base stack level.

         Scratch location for expression/function evaluation.

0129-2C  WORKE:   Scratch area. Contains also the argument A of
                  the last software random RND(A).
```

Random number kernel:

012D-30 RNUM: Random number kernel.
!RNDLY: Random number delay count (1 byte).

Output switching:

0131 OTSW: #00 output to screen + RS232.
#01 output to screen only.
#02 output to edit buffer.
#03 output via DOUTC.

Encoding input source switching:

0132/33 EFEPT: Encoding input pointer. Points to start-address of Basic-line just being encoded.
0134 EFECT: Encoded input count. Counts length of line.
0135 EFSW: Encoded input switching:
#00 Input from keyboard/DINC.
#01 Input from string.
#02 From edit buffer to program area.

Variables used during expression encoding (could overlap with runtime variables):

0136 TYPE: Type of latest expression or item:
#00 FPT
#10 INT
#20 STR
#30 Boolean

0137 RGTOP: Latest priority operator:

#00	no operation	#6A	IOR
#38	AND	#6C	IXOR
#39	OR	#8D	SHL
#50	>=	#8E	SHR
#51	>	#A0	+
#52	<>	#A1	-
#53	<=	#C2	/
#54	<	#C3	*
#55	=	#CF	MOD
#69	IAND	#E4	^

013B OLDOP: Old priority operator.
0139/3A HOPPT: Pointer to place in encoded input buffer for next operator.
013B/3C RGTPT: Pointer to place in encoded input buffer of operand latest operator.

Mask to select cassette 1 or 2:

013D CASSL: #10 Cassette 1 activated.
#20 Cassette 2 activated.

Encoded input buffer:

013E-BD EBUF: 128 bytes buffer. Also used by utility.

Interrupt handler variables:

01BE/BF TIMER: Timer location. Also used in WAIT TIME.
01C0 CTIMR: Cursor clock. Used for cursor flashing.
CTIMV: #0F: Flash time in 20 ms units.
If #00, cursor flashes.
01C1 KBXCT: Extend keyboard scan time counter. When #00, keyboard scan will be performed.
KBXCK: #02: Keyboard scan time (16 ms units). Also used by RAND routine.

Sound control block storage:

01C2-CF: Sound control block 0.
SCBL: Length of a sound block (14 bytes).
01C2 SCB0: Elapsed count of current volume:
#FF: channel off.
#FE: current volume forever.
01C3/C4 Pointer to required count at this volume in envelope table.
01C5/C6 Pointer to start envelope table being used.
01C7 Sound-volume *8. Multiplier for volume, between 0 and 16, shifted 3 places left.
01C8 Basic volume at this moment, calculated from sound-volume and present envelope volume.
01C9 Counter for tremolo. 0 if no tremolo.
01CA Actual volume, calculated from volume and tremolo fluctuations.
01CB Glissando flag:
#00 Endperiod reached.
#01 Set frequency.
#02 Endperiod not reached.
01CC/CD Current period of output.
01CE/CF Required final period of output.
01D0-DB SCB1: Sound control block 1 (see SCB0).
01DE-EB SCB2: Sound control block 2 (idem).
01EC-F4 NCB: Noise control block.
NCBL: Length of noise block (9 bytes).
The noise control block is identical to the sound control block, but without period-values and tremolo.

Envelope storage:

01F5--0274 ENVST: Envelope storage (128 bytes).
ENVLL: #40: Number of bytes/envelope
NUMENV: #02: Number of envelopes.
Two envelope tables of each 64 bytes:
#01F5-#0234 and #0235-#0274.

Type storage:

0275--28E IMPTAB: Implicit type table.

#0275	A	#027C	H	#0283	O	#028A	V
#0276	B	#027D	I	#0284	P	#028B	W
#0277	C	#027E	J	#0285	Q	#028C	X
#0278	D	#027F	K	#0286	R	#028D	Y
#0279	E	#0280	L	#0287	S	#028E	Z
#027A	F	#0281	M	#0288	T		
#027B	G	#0282	N	#0289	U		

```
028F      IMPTYP: Default number type. Selected by IMP command.
                  #00  FPT
                  #10  INT
                  #20  STR
0290      REQTYP: Required number type for present operation.
                  #00  FPT
                  #10  INT
                  #20  STR
                  #30  Variable name argument
                  #40  Array without arguments


0291/92   DATAQ:  Pointer to begin current data line.
0293      RNDLY:
0294      POROM:  Duplicate of (#FD04).
0295      POR1M:  Duplicate of (#FD05).
0296      INSW:   Input switching:
                     If #00, input from keyboard.
                     If <>#00, input from DINC (Default: RS232).
0297-9A           Spare.

          Heap/text buffer/symbol table pointers:

029B/9C   HEAP:   Start address of HEAP.
029D/9E   HSIZE:  Size of HEAP.
                     HSIZD: #100: Default size.
029F/A0   TXTBGN: Start address of text buffer.
02A1/A2   TXTUSE: End text buffer and.
          STBBGN: Start symbol table.
02A3/A4   STBUSE: End of symbol table.
02A5/A6   SCRBOT: Bottom screen RAM area (48K):
                     mode 0:      #B350
                     mode 1/2(A): #B7A0
                     mode 3/4(A): #A65C
                     mode 5/6(A): #63B8

          Keyboard variables + constants:

02A7/A8   KBTPT:  Pointer to table with ASCII-codes.
02A9-B0   MAP1:   Latest scan of keys (key-codes).
                  (row 0 in #02A9, row 7 in #02B0)
   02AF   RPLOC:  Byte containing REPT key.
                     RPMSK: #20: Rept key bit.
                     BRSEL: #40: Column select mask for BREAK.
                     BRMSK: #40: Break key bit.
   02B0   SHLOC:  Byte containing SHIFT.
                     SHMSK: #40: Shift key bit.
02B1-B8   MAP2:   Previous scanning of keyboard.
02B9      KNSCAN: Set to scan for BREAK only. When (#02B9)
                  is #FF: scan for BREAK only.
02BA-BD   KLIND:  4 byte circular buffer to store the ASCII
                  values for keys pressed.
                     KBLEN/KEYL: #04: length rollover buffer.
02BE/BF   KLIIN:  Next position for input to KLIND.
02C0/C1   KLIOU:  Next position for output from KLIND.
02C2      RPCNT:  Count for REPT. #01 if REPT is not
                  pressed. Else it is used as timer for the
                  repeat function.
02C3      SHLK:   Set to #FF if CTRL is pressed to
                  invert SHIFT. Else #00. Used to
                  calculate the offset for the ASCII code
                  table.

02C4      KBRFL:  Break flag. #FF indicates BREAK pressed
                  (Only if suspended program). If BREAK is pressed,
                  #02C4 counts from 00 to #0F before stopping
                  the program.

          Data/cassette switching vectors:

          Copy of ROM (#D7A4 - #D7CA) for cassette and RS232.
          Can be loaded with other I/O vectors.

02C5-EB   IOVEC:  02C5  WOPEN:  C3 B8 D2  JMP: D2B8
                  02C8  WBLK:   C3 F1 D2  JMP: D2F1
                  02CB  WCLOSE: C3 27 D4  JMP: D427
                  02CE  ROPEN:  C3 25 D3  JMP: D325
                  02D1  RBLK:   C3 40 D3  JMP: D340
                  02D4  RCLOSE: C3 45 D4  JMP: D445
                  02D7  MBLK:   C3 A2 D3  JMP: D3A2
                  02DA  RESET:  C9 00 00  RET
                  02DD  DOUTC:  C9 00 00  RET
                  02E0  DINC:   C3 B4 DD  JMP: DDB4
                  02E3          C9 00 00  RET
                  02E6  TAPSL:  24 24     Tape speed leader.
                  02E8  TAPSD:  24 3C     Tape speed data.
                  02EA  TAPST:  24 18     Tape speed trailer.


HEAP, PROGRAM AREA, SCREEN RAM: 02EC - BFFF
===========================================

02EC-   (RAM:      HEAP (Strings + arrays)  - See (#029B/9C).
 -BFFF  (VAREND:   Program (compiled Basic) - See (#029F/A0).
        (VARLAST:  Symbol table             - See (#02A1/A2).
                   Not used RAM             - See (#02A3/A4).
                   Screen RAM               - See (#02A5/A6).


ROM AND CPU AREA: C000 - F8FF
=============================

C000-             24K ROM:
  -EFFF              #C000-#DFFF: 8K non-switched ROM.
           VECA:     #E000-#EFFF: 4 banks of each 4K ROM.
                                  (switchable).


F000-             Can be used for ROM extension (reading only).
  -F7FF           Is already completely used by Memocom MDCR-D.


F800-             Microcomputer stack.
  -F8FF           Incl. vector for MDS jump instructions.
                     #F800  SRBOT    Bottom of stack RAM.
                     #F900  STTOP    Top of stack RAM.
```

I/O DEVICE ADDRESSES: F900 - FFFF

F900-FAFF Spare I/O device addresses. (Not wired on pC board).

MATH. CHIP AMD 9511: FB00 - FBFF

FB00 MTHAD:) Data math.chip.
      MDATA:)
FB02 MCOMD:) Command + status.
      MSTATUS:)

AMD9511 operator and status bytes:

ODADD:	#2C Int addition	OFADD:	#10 Fpt addition
ODSUB:	#2D Int subtract	OFSUB:	#11 Fpt subtract
ODMUL:	#2E Int multiply	OFMUL:	#12 Fpt multiply
ODDIV:	#2F Int division	OFDIV:	#13 Fpt division
OSQRT:	#01 Square root	OFIXD:	#1E Fix
OSIN:	#02 Sine	OFLTD:	#1C Float
OCOS:	#03 Cosine	OCHSD:	#34 Change sign int
OTAN:	#04 Tangent	OCHSF:	#15 Change sign fpt
OASIN:	#05 Arc sine	OPTOD:	#37 Push int/fpt
OACOS:	#06 Arc cosine	OPOPD:	#38 Pop int/fpt
OATAN:	#07 Arc tangent		
OLOG:	#08 Log base 10		
OLN:	#09 Log base e	MBUSY:	#80 Busy status bit
OEXP:	#0A Expotential	MERRB:	#1E All error bits
OPWR:	#0B X^Y	MZERO:	#20 Top of stack

PROGRAMMABLE INTERVAL TIMER 8253: FC00 - FCFF

Used for sound generator. 3 independent 16 bits down counters with programmable counter modes.

FC00/01 SNDAD:)
         SND0:) Counter 0 (oscillator channel 0).
         !PDLCH: Used as counter for paddle operations.
FC02/03 SND1: Counter 1 (oscillator channel 1).
FC04/05 SND2: Counter 2 (oscillator channel 2).
         (16 bit data; LSB first)
FC06   SNDC: Command 8253. To be loaded prior to freq. selection with resp. #36, #76 and #B6.
Command word format:
bit 0 : 0 binairy counter 16 digits.
1 BCD counter (4 decades).
3,2,1: 000 mode 0: Interrupt on end count.
001 mode 1: Programmable one shot.
x10 mode 2: Rate generator.
x11 mode 3: Sq.wave rate generator.
100 mode 4: SW trig. strobe.
101 mode 5: HW trig. strobe.
5,4 : 00 Counter latch operation.
01 Read/load MSB only.
10 Read/load LSB only.
11 Read/load LSB first, then MSB.
7,6 : 00 Select counter 0.
01 Select counter 1.
10 Select counter 2.
11 Illegal.

Several control words:

COFIX:	#00	Fix count on channel 0.
COM0:	#30	Chan.0, mode 0, 2 byte op.
COM1:	#32	Chan.0, mode 1, 2 byte op.
COM3:	#36	Chan.0, mode 3, 2 byte op.
C1M3:	#76	Chan.1, mode 3, 2 byte op.
C2M3:	#B6	Chan.2, mode 3, 2 byte op.

DISCRETE I/O DEVICE ADDRESSES: FD00 - FDFF

FD00 PORI: IN (1) bit 0: -
1: -
2: PIPGE: Page signal
3: PIDTR: Serial output ready
4: PIBU1: Button on paddle 1 (1 = closed)
5: PIBU2: Button on paddle 2 (1 = closed)
6: PIRPI: Random data
7: PICAI: Cassette input data

FD01 PDLST: OUT (3) Single pulse used to trigger paddle timer circuit.

FD04 PORO: OUT (2) bit 0 - 3: volume osc. channel 0
4 - 7: volume osc. channel 1
FD05 POR1: OUT (2) bit 0 - 3: volume osc. channel 2
4 - 7: volume random noise generator.
FD06 PORO: OUT (3) bit 0: POCAS: Cassette data output
1,2: PDLMSK: Paddle select
3: POPNA: Paddle enable
4: POCM1: Cassette 1 motor control.(0 = run)
5: POCM2: Cassette 2 motor control.(0 = run)
7,6: ROM bank switching:
00 bank 0
01 bank 1
10 bank 2
11 bank 3

PROGR. PERIPHERAL INTERFACE 8255 : FE00 - FEFF

Used for DCE-bus (GIC Controller).

FE00 GIC: (1) I/O port A
FE01 (1) I/O port B
FE02 (1) I/O port C
FE03 (6) Command word 8255:

Contr.	PA	PCH	PCL	PB	(mode 0)
#80	out	out	out	out	RWMOP
#81	out	out	in	out	
#82	out	out	out	in	
#83	out	out	in	in	
#88	out	in	out	out	
#89	out	in	in	out	
#8A	out	in	out	in	
#8B	out	in	in	in	
#90	in	out	out	out	RWMIP
#91	in	out	in	out	
#92	in	out	out	in	
#93	in	out	in	in	
#98	in	in	out	out	
#99	in	in	in	out	
#9A	in	in	out	in	
#9B	in	in	in	in	

TICC: TIMER + INTERRUPT CONTROLLER 5501: FF00-FFFF

FFF0 (4) Serial input buffer. Contains the last character received on the RS232 interface.

FFF1 (4) Keyboard input port. Bottom 7 bits are data input from the keyboard. Bit 7 is the IN7 line from the DCE-bus and is attached to the page-blanking signal for the TV. Every 20 ms. an impulse is present.

FFF2 (2) Interrupt address register:
bits 5,4,3: Number of pending interrupt.
7,6 :)
2,1,0:) always '1'

FFF3 (4) Status register:
bit 0: Frame error. Set by a BREAK on the RS232 input.
1: Overrun error. Set if a character has been received but not taken by the CPU.
2: Serial input. Set if no data is received.
3: Receive buffer loaded. Set if a character has been received.
4: Transmit buffer empty. Set if RS232 output is ready to accept another character.
5: Interrupt pending. Set if one or more of the enabled interrupts has occured.
6: Full bit detected. Set if the first data bit of an incoming character has been detected.
7: Start bit detected. Set if the start bit of an incoming character has been detected.

FFF4 (2) Command register:
bit 0: TICC reset.
1: Send Break. If set, the serial output is high impedance.
2: Interrupt 7 select. A '1' selects IN7 of the DCE-bus, a '0' selects Timer 5.
3: Interrupt acknowledge enable. A '1' enables TICC to accept a INTA signal from the CPU.
4 - 7: Always 0.

FFF5 (6) Communications rate register:
bit 0: 110 baud
1: 150 baud
2: 300 baud
3: 1200 baud
4: 2400 baud
5: 4800 baud
6: 9600 baud
7: 1 - one stop bit
0 - two stop bits

FFF6 (6) Serial output buffer. Write byte to this location to send it on the RS232 output. Use only when #FFF3-bit 4 is high.

FFF7 (7) Keyboard output port. Data output to scan keyboard.

FFF8 (2) Interrupt mask register:
bit 0: timer 1 has expired (UTIM).
1: timer 2 has expired.
2: External interrupt (STKIM).
3: Timer 3 has expired (SNDIM).
4: Serial receiver loaded.
5: Serial transmitter empty.
6: Timer 4 has expired (KBIM).
7: Timer 5 has expired or IN7 (CLKIM).
(react only on low-high transition)

FFF9 (2) UTIAD: Timer 1 address (UT).
FFFA (1) Timer 2 address.
FFFB (2) SNDIAD: Timer 3 address (sound).
FFFC (2) KBIAD: Timer 4 address (keyboard).
FFFD (1) Timer 5 address.
FFFE not used.
FFFF not used.

NOTES: (1) Read and write allowed.
(2) Reading allowed. Writing too, but may be overwritten by BASIC program.
(3) No writing allowed.
(4) Reading allowed, writing not.
(5) Should not be accessed.
(6) Writing allowed, reading not.
(7) Reading not allowed, writing is harmless but useless; keyboard scanner will overwrite it.

REMARKS:

ADDRESSES FB00 - FFFF:

The highest byte of the address is used for the chip select signal CS of the peripheral equipment 8253, 8255, 5501 etc. The lowest byte is used to address the several registers of the peripheral. Its high nibble does not have any value. So addresses in this range can be read as FBx0 - FFxF, in which x is a don't care.

```
002                         ORG   :C000
003                 *
004                 *
005                 *
006                 *  =================
007                 *** MATH. UTILITIES ***
008                 *  =================
009                 *
010                 *
011                 ***************
012                 * ENTRYPOINTS *
013                 ***************
014                 *
015 C000 C319C7     BASE    JMP   :C719     Reset (entry on hardware
016                                          reset)
017 C003 C335C0     XINIT   JMP   :C035     Math. package initialisation
018 C006 C3F3C0     XFINM   JMP   :C0F3     Incr. FPT number in memory
019 C009 C3FBC1     XFDCM   JMP   :C1FB     Decr. FPT number in memory
020                                          (not used)
021 C00C C379C0     XFCOMP  JMP   :C079     FPT Compare
022 C00F C3BBC0     XIINM   JMP   :C0BB     Incr. INT number in memory
023 C012 C3D5C0     XIDCM   JMP   :C0D5     Decr. INT number in memory
024                                          (not used)
025 C015 C3ACC0     XICOMP  JMP   :C0AC     INT Compare
026 C018 C31EC2     XPUSH   JMP   :C21E     Save MACC on stack
027 C01B C334C2     XPOP    JMP   :C234     Retrieve MACC from stack
028 C01E C349C2     XFCB    JMP   :C249     Input FPT number to MACC
029 C021 C361C3     XFBC    JMP   :C361     Conv. FPT number for output
030 C024 C373C5     XICB    JMP   :C573     Input INT number to MACC
031 C027 C3B2C5     XIBC    JMP   :C5B2     Conv. INT number for output
032 C02A C314C6     XHCB    JMP   :C614     Input Hex number to MACC
033 C02D C353C6     XHBC    JMP   :C653     Conv. MACC to Hex for output
034 C030 C386C4     XPRTY   JMP   :C486     Pretties up FPT/INT number
035 C033 E300       DECBUF  DBL   :00E3     Location output buffer
036                 *
037                 ********************************
038                 * MATH. PACKAGE INITIALISATION *
039                 ********************************
040                 *
041                 * Entry: HL: Address input encoding routine (DDE0).
042                 *         DE: Base address error routines (C7F2).
043                 * Exit:   AFDEHL corrupted, BC preserved.
044                 *
045 C035 22D200     MINIT   SHLD  :00D2     Init. (00D2/3)=DDE0
046 C038 EB                XCHG
047 C039 22D000             SHLD  :00D0     Init. (00D0/1)=C7F2
048 C03C 3A02FB             LDA   :FB02     Get math.chip status
049 C03F B7                 ORA   A         Check if math.chip present
050 C040 3E00               MVI   A,:00     Flag = 0 if not
051 C042 FA47C0             JM    :C047
052 C045 3E7B               MVI   A,:7B     Flag = #7B if present
053 C047 32D400     LC18    STA   :00D4     Set math.chip flag
054 C04A C9                 RET
055                 *
056                 ***************************
057                 * OVERFLOW ERROR ROUTINE *
058                 ***************************
059                 *
060                 * (From LC20 common part for various entries).
061                 *
062                 * Jump to (00D0/1) = Address 'overflow error'
063                 * routine (C7F2).
```



```
064                 *
065                 * Entry: If start at LC20: offset in HL.
066                 * Exit:  AFBCDEHL preserved.
067                 *        On stack original returnaddress.
068                 *
069 C04B E5           FPEOV   PUSH  H
070 C04C 210000             LXI   H,:0000   Init offset = 0
071                 *
072 C04F F5         LC20    PUSH  PSW
073 C050 D5                 PUSH  D
074 C051 EB                 XCHG            Offset in DE
075 C052 2AD000             LHLD  :00D0     Get addr pointer
076 C055 19                 DAD   D         Add offset
077 C056 7E                 MOV   A,M
078 C057 23                 INX   H
079 C058 66                 MOV   H,M
080 C059 6F                 MOV   L,A       Get addr routine in HL
081 C05A D1                 POP   D
082 C05B F1                 POP   PSW
083 C05C E3                 XTHL            New addr on stack
084 C05D C9                 RET             Continu with new address
085                 *
086                 ******************
087                 * ARGUMENT ERROR *
088                 ******************
089                 *
090                 * Jump to (00D0/1)+2 = Address 'number out of range'
091                 * routine (C7F4).
092                 *
093                 * Entry/exit: See FPEOV.
094                 *
095 C05E E5           FPEAE   PUSH  H
096 C05F 210200             LXI   H,:0002   Init. offset
097 C062 C34FC0             JMP   :C04F     Calc. new addr, go to it
098                 *
099                 ******************
100                 * UNDERFLOW ERROR *
101                 ******************
102                 *
103                 * Jump to (00D0/1)+4 = Return (C7F6).
104                 * Underflow gives 0 as result of operation.
105                 *
106                 * Entry/exit: See FPEOV.
107                 *
108 C065 E5           FPEUN   PUSH  H
109 C066 215EC4             LXI   H,:C45E   Addr. FPT(0)
110 C069 C30CD2             JMP   :D20C     Copy '0' into MACC
111                 *
112                 ************************
113                 * DIVIDE BY ZERO ERROR *
114                 ************************
115                 *
116                 * Jump to (00D0/1)+6 = Address 'divide by zero'
117                 * routine (C7F8).
118                 *
119                 * Entry/exit: See FPEOV.
120                 *
121 C06C E5           FPEDO   PUSH  H
122 C06D 210600             LXI   H,:0006   Init. offset
123 C070 C34FC0             JMP   :C04F     Calc. new addr, go to it
124                 *
125                 *
```

```
                        ****************************
                        * GET CHARACTER FROM LINE *
                        ****************************
                        *
                        * Entry: None.
                        * Exit : All registers preserved.
                        *        Address to continue on stack.
                        *
C073 E5         LC22    PUSH  H
C074 2AD200             LHLD  :00D2     Get addr 'Get char' routine
C077 E3                 XTHL            on stack; restore HL
C078 C9                 RET             Goto (00D0/1)+2
                        *
                        ****************************
                        * FLOATING POINT COMPARE *
                        ****************************
                        *
                        * Compares normalised FPT numbers in MACC
                        * and in M.
                        *
                        * Exit: ABCDEHL preserved.
                        *       Flags: CY=1,S=0,Z=1: both nrs. 0
                        *              CY=0,S=0,Z=1: both nrs. identical
                        *              CY=0,S=0,Z=0: MACC > M
                        *              CY=0,S=1,Z=0: MACC < M
                        *
C079 C5         FCOMP   PUSH  B
C07A F5                 PUSH  PSW
C07B D5                 PUSH  D
C07C E5                 PUSH  H
C07D E7                 RST   4         Copy MACC to reg A,B,C,D
C07E 15                 DATA  :15
C07F 5F                 MOV   E,A       Exp.byte in E
C080 AE                 XRA   M         XOR both exp.bytes
C081 FAB7C0             JM    :C0B7     Jump if different signs

                        * If equal signs:

C084 C3E8D1             JMP   :D1E8     Goto D1E8, return to C087
                        *
C087 17         LC23    RAL
C088 C2A3C0             JNZ   :C0A3
C08B 7B         LC24    MOV   A,E
C08C 96                 SUB   M         Comp. exp. bytes
C08D C2A2C0             JNZ   :C0A2     Jump if not equal
C090 23                 INX   H
C091 78                 MOV   A,B
C092 96                 SUB   M         Comp. 1st bytes mantissa's
C093 C2A2C0             JNZ   :C0A2     Jump if not equal
C096 23                 INX   H
C097 79                 MOV   A,C
C098 96                 SUB   M         Comp. 2nd bytes mantissa's
C099 C2A2C0             JNZ   :C0A2     Jump if not equal
C09C 23                 INX   H
C09D 7A                 MOV   A,D
C09E 96                 SUB   M         Comp. 3rd bytes mantissa's
C09F CAA6C0     LC25    JZ    :C0A6     Jump if not equal
C0A2 1F         LC26    RAR             ) Set flags for output
C0A3 AB         LC27    XRA   E         )
C0A4 F601       LC28    ORI   :01       Clear CY-flag
C0A6 E1         LC29    POP   H
C0A7 D1                 POP   D
```



```
C0A8 C1                 POP   B
C0A9 78                 MOV   A,B       Restore A
C0AA C1                 POP   B
C0AB C9                 RET
                        *
                        ********************
                        * INTEGER COMPARE *
                        ********************
                        *
                        * Compares INT numbers in MACC and M.
                        * REMARK: Routine is incorrect when both
                        *         numbers are negative ! Then result
                        *         is if MACC > M due to LC26/LC27.
                        *
                        * Exit:  ABCDEHL preserved. CY=0
                        *        Flags: S=0, Z=1: Both numbers equal
                        *               S=0, Z=0: MACC > M
                        *               S=1, Z=0: MACC < M
                        *
C0AC C5         ICOMP   PUSH  B
C0AD F5                 PUSH  PSW
C0AE D5                 PUSH  D
C0AF E5                 PUSH  H
C0B0 E7                 RST   4         Copy MACC to reg. A,B,C,D
C0B1 15                 DATA  :15
C0B2 5F                 MOV   E,A       Sign byte in E
C0B3 AE                 XRA   M         XOR both sign bytes
C0B4 F28BC0             JP    :C08B     If both nrs have same sign:
                                        compare

                        * If different signs:

C0B7 AE         LC241   XRA   M         Find out which one is neg:
                                        S=1: MEM pos; MACC neg
                                        S=0: MEM neg; MACC pos
C0B8 C3A4C0             JMP   :C0A4     Abort
                        *
                        ****************************************
                        * INCREMENT INTEGER NUMBER IN MEMORY *
                        ****************************************
                        *
                        * Entry: HL points to 1st byte of INT number.
                        * Exit:  All registers preserved.
                        *
C0BB F5         IINM    PUSH  PSW
C0BC E5                 PUSH  H
C0BD 23                 INX   H
C0BE 23                 INX   H
C0BF 23                 INX   H         HL pnts to last byte
C0C0 3E03               MVI   A,:03     Nr of bytes for INT nr
C0C2 34         LC30    INR   M         Incr. INT nr.
C0C3 C2D2C0             JNZ   :C0D2     Ready if no overflow
C0C6 2B                 DCX   H         Goto next byte
C0C7 3D                 DCR   A         1st byte reached?
C0C8 C2C2C0             JNZ   :C0C2     Incr. next byte
C0CB 34                 INR   M         Incr. 1st byte
C0CC 7E                 MOV   A,M       Get it
C0CD FE80               CPI   :80       msb=1?
C0CF CC4BC0             CZ    :C04B     Then overflow error
C0D2 E1         LC31    POP   H         Normal return
C0D3 F1                 POP   PSW
C0D4 C9                 RET
```

```
                        *
                        ****************************************************
                        * DECREMENT INTEGER NUMBER IN MEMORY - (not used) *
                        ****************************************************
                        *
                        * Entry: HL points to 1st byte of INT number.
                        * Exit:  All registers preserved.
                        *
C0D5 F5         IDCM    PUSH  PSW
C0D6 C5                 PUSH  B
C0D7 E5                 PUSH  H
C0D8 23                 INX   H
C0D9 23                 INX   H
C0DA 23                 INX   H         HL pnts to last byte
C0DB 0603               MVI   B,:03     Nr. of bytes of INT nr.
C0DD 35         LC32    DCR   M         Decr. INT nr
C0DE 7E                 MOV   A,M
C0DF 3C                 INR   A         Check for overflow
C0E0 C2EFC0             JNZ   :C0EF     Ready if no overflow
C0E3 2B                 DCX   H         Goto next byte
C0E4 05                 DCR   B         Decr. byte count
C0E5 C2DDC0             JNZ   :C0DD     Next byte if not ready
C0E8 35                 DCR   M         Decr. hibyte
C0E9 7E                 MOV   A,M
C0EA FE7F               CPI   :7F       Check for overflow
C0EC CC4BC0             CZ    :C04B     Then run overflow error
C0EF E1         LC33    POP   H         Normal return
C0F0 C1                 POP   B
C0F1 F1                 POP   PSW
C0F2 C9                 RET
                        *
                        ********************************************
                        * INCREMENT FLOATING POINT NUMBER IN MEMORY *
                        ********************************************
                        *
                        * If number = 0, or exponent < 0, a '1' is added
                        * to the lsb of the mantissa. Else, the position
                        * of the least significant '1' is looked up, and
                        * a '1' is added to this position.
                        * If the lsb of the mantissa is already a rounded
                        * value, no increment occurs.
                        *
                        * Entry: HL points to 1st byte of FPT number.
                        * Exit:  All registers preserved.
                        *
C0F3 F5         FINM    PUSH  PSW
C0F4 C5                 PUSH  B
C0F5 D5                 PUSH  D
C0F6 E5                 PUSH  H
C0F7 1162C4             LXI   D,:C462   Addr FPT(1)
C0FA 7E                 MOV   A,M       Get exp.byte
C0FB E67F               ANI   :7F       Mask sign bit
C0FD CAACC1             JZ    :C1AC     If nr=0: add 1, abort
C100 FE40               CPI   :40       Is exponent negative?
C102 D2ACC1             JNC   :C1AC     Then add 1, abort
C105 FE19               CPI   :19       Lsb of mantissa is not lsb
                                        of number ?
C107 D24DC1             JNC   :C14D     Then Popall, ret
C10A BE                 CMP   M         Check if nr. is negative
C10B 23                 INX   H
C10C C252C1             JNZ   :C152     Then jump
```



```
                        * From LC34 also used by XFDCM.
                        * Find lsb of mantissa if nr is positive:
C10F D609       LC34    SUI   :09       In 1st byte ?
C111 DCEEC1             CC    :C1EE     Then SHL bit into A (A) time
C114 DA36C1             JC    :C136     and jump
C117 23                 INX   H
C118 D608               SUI   :08       In 2nd byte ?
C11A DCEEC1             CC    :C1EE     Then SHL bit into A (A) time
C11D DA2EC1             JC    :C12E     and jump
C120 23                 INX   H
C121 D608               SUI   :08       In 3rd byte ?
C123 CDEEC1             CALL  :C1EE     Then SHL bit into A (A) time
C126 86                 ADD   M
C127 77                 MOV   M,A       Add 1 to 3rd byte mantissa
C128 D24DC1             JNC   :C14D     Ready if no overflow
C12B 2B                 DCX   H
C12C 3E01               MVI   A,:01     Overflow: add 1 to 2nd byte
C12E 86         LC35    ADD   M
C12F 77                 MOV   M,A       Add 1 to 2nd byte mantissa
C130 D24DC1             JNC   :C14D     Ready if no overflow
C133 2B                 DCX   H
C134 3E01               MVI   A,:01     Overflow: add 1 to 1st byte
C136 86         LC36    ADD   M
C137 77                 MOV   M,A       Add 1 to 1st byte mantissa
C138 D24DC1             JNC   :C14D     Ready if no overflow

                        * If overflow into exponent byte:

C13B 1F                 RAR             )
C13C 77                 MOV   M,A       )
C13D 23                 INX   H         )
C13E 7E                 MOV   A,M       ) Shift all bits in
C13F 1F                 RAR             ) mantissa right
C140 77                 MOV   M,A       ) one position
C141 23                 INX   H         )
C142 7E                 MOV   A,M       )
C143 1F                 RAR             )
C144 77                 MOV   M,A       )
C145 2B                 DCX   H
C146 2B                 DCX   H
C147 2B                 DCX   H         HL pnts to exp.byte
C148 3E01               MVI   A,:01
C14A CDBAC1     LC37    CALL  :C1BA     Add 1 to exponent
                        *
C14D E1         EXIT    POP   H
C14E D1                 POP   D
C14F C1                 POP   B
C150 F1                 POP   PSW
C151 C9                 RET

                        * Find lsb of mantissa if nr is negative:

C152 D609       LC39    SUI   :09       In 1st byte ?
C154 DCEEC1             CC    :C1EE     Then SHL bit into A (A) time
C157 DA7DC1             JC    :C17D     and jump
C15A 23                 INX   H
C15B D608               SUI   :08       In 2nd byte ?
C15D DCEEC1             CC    :C1EE     Then SHL bit into A (A) time
C160 DA73C1             JC    :C173     and jump
C163 23                 INX   H
C164 D608               SUI   :08       In 3rd byte ?
```

```
498 C1E5 37                 STC
499 C1E6 C3DEC1             JMP    :C1DE      Abort with CY=1
500                 *
501                 ***************
502                 * SIGN EXTEND *
503                 ***************
504                 *
505                 * Exponent byte is normalized.
506                 *
507                 * Entry: Exp.byte in A.
508                 * Exit:  BCDEHL preserved.
509                 *        Normalized exp.byte in A: Exp.value in
510                 *        bits 7-2, sign mantissa in bit 1, sign
511                 *        exponent in bit 0.
512                 *
513 C1E9 07          SEXT   RLC
514 C1EA 07                 RLC
515 C1EB 07                 RLC
516 C1EC 1F                 RAR
517 C1ED C9                 RET
518                 *
519                 ************************************
520                 * MOVE A BIT INTO A LEFT (A) TIMES *
521                 ************************************
522                 *
523                 * Used to place a '1' in the correct position in
524                 * a byte for adding/subtracting '1' to/from the
525                 * least significant '1' of a FPT mantissa.
526                 *
527                 * Entry: A contains a neg. number indicating
528                 *        how often RAL has to be performed.
529                 * Exit:  Result in A,B.
530                 *        FCDEHL preserved.
531                 *
532 C1EE F5          LC50   PUSH   PSW
533 C1EF 47                 MOV    B,A        Save nr of shifts
534 C1F0 AF                 XRA    A          Clear A
535 C1F1 37                 STC               Set CY
536 C1F2 17          LC51   RAL                SHL
537 C1F3 04                 INR    B          Update count
538 C1F4 C2F2C1             JNZ    :C1F2      Continu if not ready
539 C1F7 47                 MOV    B,A        Save result
540 C1F8 F1                 POP    PSW
541 C1F9 78                 MOV    A,B        Result in A
542 C1FA C9                 RET
543                 *
544                 *
545                 *
546 C1FB                    END
```

```
****************************
* S Y M B O L   T A B L E *
****************************
```

BASE	C000	DECBUF	C033	EXIT	C14D	FCOMP	C079
FINM	C0F3	FPEAE	C05E	FPEDO	C06C	FPEOV	C04B
FPEUN	C065	ICOMP	C0AC	IDCM	C0D5	IINM	C0BB
LC18	C047	LC20	C04F	LC22	C073	LC225	C1B7
LC23	C087	LC24	C08B	LC241	C0B7	LC25	C09F
LC26	C0A2	LC27	C0A3	LC28	C0A4	LC29	C0A6
LC30	C0C2	LC31	C0D2	LC32	C0DD	LC33	C0EF
LC34	C10F	LC35	C12E	LC36	C136	LC37	C14A

LC39	C152	LC40	C173	LC41	C17D	LC42	C186
LC43	C19F	LC44	C1A6	LC45	C1AC	LC46	C1BA
LC47	C1DE	LC48	C1E2	LC50	C1EE	LC51	C1F2
MINIT	C035	SEXT	C1E9	XFBC	C021	XFCB	C01E
XFCOMP	C00C	XFDCM	C009	XFINM	C006	XHBC	C02D
XHCB	C02A	XIBC	C027	XICB	C024	XICOMP	C015
XIDCM	C012	XIINM	C00F	XINIT	C003	XPOP	C01B
XPRTY	C030	XPUSH	C018				

```
                        ORG   :C1FB
                *
                *
                *
                ********************************************
                * DECREMENT FLOATING POINT NUMBER IN MEMORY *
                ********************************************
                *
                * Routine is not used.
                *
                * If the number is 0, or the exponent < 0, -1 is
                * added to the mantissa. Else, a -1 is added/
                * subtracted to/from the least significant '1' of
                * the mantissa.
                * If the lsb of the mantissa is already a rounded
                * value, no decrement occurs.
                *
                * Entry: HL points to FPT number in M.
                * Exit:  All registers preserved.
                *
C1FB F5         FDCM    PUSH  PSW
C1FC C5                 PUSH  B
C1FD D5                 PUSH  D
C1FE E5                 PUSH  H
C1FF 111AC2             LXI   D,:C21A   Addr. FPT(-1)
C202 7E                 MOV   A,M       Get exp.byte
C203 E67F               ANI   :7F       Mask sign bit
C205 CAACC1             JZ    :C1AC     If nr=0: add -1, abort
C208 FE40               CPI   :40       Is exp. negative ?
C20A D2ACC1             JNC   :C1AC     Then add -1, abort
C20D FE18               CPI   :18       Max. nr of mantissa bits
C20F D24DC1             JNC   :C14D     Abort if lsb mantissa is
                                        not lsb of number
C212 BE                 CMP   M         Check if nr. is negative
C213 23                 INX   H
C214 CA52C1             JZ    :C152     Into FINM for neg. nr
C217 C30FC1             JMP   :C10F     Idem for pos. nr.
                *
                * DATA - (not used):
                *
C21A 81         FPM1    DATA  :81       FPT (-1)
C21B 80                 DATA  :80
C21C 00                 DATA  :00
C21D 00                 DATA  :00
                *
                **********************
                * SAVE MACC ON STACK *
                **********************
                *
                * Contents MACC is placed on TOS. Returnaddress
                * is saved.
                *
                * Entry: None.
                * Exit:  All registers preserved.
                *        On stack: HL; returnaddress; MACC.
                *
C21E 22E100     PUSH    SHLD  :00E1     Save HL
C221 E3                 XTHL            Get returnaddress
C222 22DF00             SHLD  :00DF     Save it
C225 E5                 PUSH  H         and put it on stack again
C226 210000             LXI   H,:0000
C229 39                 DAD   SP        SP in HL
```



```
C22A E7                 RST   4         Copy MACC to TOS
C22B 0F                 DATA  :0F
C22C 2ADF00             LHLD  :00DF     Get returnaddr.
C22F E5                 PUSH  H         on stack
C230 2AE100     LC52    LHLD  :00E1     Get original HL
C233 C9                 RET
                *
                ****************************
                * RETRIEVE MACC FROM STACK *
                ****************************
                *
                * Gets data from TOS and place it in MACC.
                *
                * Entry: None.
                * Exit:  All registers preserved.
                *
C234 22E100     POP     SHLD  :00E1     Save HL
C237 E1                 POP   H         Get returnaddress
C238 22DF00             SHLD  :00DF     Save it
C23B 210000             LXI   H,:0000
C23E 39                 DAD   SP        Get SP in HL
C23F E7                 RST   4         Copy TOS to MACC
C240 0C                 DATA  :0C
C241 E1                 POP   H
C242 2ADF00             LHLD  :00DF     Get returnaddress
C245 E3                 XTHL            on stack
C246 C330C2             JMP   :C230     Restore HL, ret
                *
                ******************************************
                * INPUT A FLOATING POINT NUMBER TO MACC *
                ******************************************
                *
                * Converts a FPT number to binary into MACC.
                * The input string is converted as a integer FPT
                * number, then multiplied/divided by a power of
                * 10, corresponding to the explicit exponent and
                * placement of the decimal point.
                *
                * Entry: C points to 1st digit of FPT nr in input.
                * Exit:  CY=1: No error.
                *        CY=0: Over/underflow error.
                *        C points past FPT string in input.
                *        ABDEHL, rest of F preserved.
                *
C249 37         FCB     STC             CY=1
C24A F5                 PUSH  PSW
C24B D5                 PUSH  D
C24C E5                 PUSH  H
C24D CDAEC2             CALL  :C2AE     Clear MACC+DEH; L=2B
C250 CD2FC3     LC53    CALL  :C32F     Get bin.value of input char
C253 DCBAC2             CC    :C2BA     Value found: Move digit into
                                        MACC
C256 DA50C2             JC    :C250     and get next digit
C259 FE2E               CPI   :2E       '.' ?
C25B CA6BC2             JZ    :C26B     Then jump
C25E 1D                 DCR   E
C25F 1C                 INR   E
C260 CAA6C2             JZ    :C2A6     If error
C263 FE45               CPI   :45       'E' ?
C265 CA84C2             JZ    :C284     Then jump
C268 C39FC2             JMP   :C29F     Convert FPT exp; quit
```

```
                        * If digit is '.':
                        *
C26B CDD5C2     LC54    CALL  :C2D5     E=0, H=H+1
C26E CD2FC3     LC55    CALL  :C32F     Get bin.value of input char
C271 DCBAC2             CC    :C2BA     If found: Move digit into
                                        MACC
C274 DA6EC2             JC    :C26E     and get next digit
C277 1D                 DCR   E
C278 1C                 INR   E
C279 CAA6C2             JZ    :C2A6     If error
C27C FE45               CPI   :45       'E' ?
C27E C4D9C2             CNZ   :C2D9     H=0 if not
C281 C29FC2             JNZ   :C29F     If not: convert exp, quit
                        *
                        * If digit is 'E':
                        *
C284 CDD9C2     LC56    CALL  :C2D9     H=0
C287 CD2FC3             CALL  :C32F     Get bin.value of input char
C28A CCDCC2             CZ    :C2DC     If '+' or '-': char in L
C28D CC2FC3             CZ    :C32F     Then get bin.value of next
                                        char
C290 D2A6C2             JNC   :C2A6     Error if no char found
C293 CDDEC2             CALL  :C2DE     H = 10 * H + A
C296 CD2FC3             CALL  :C32F     Get bin.value of input char
C299 DCDEC2             CC    :C2DE     If found: H = 10 * H + A
C29C DC2FC3             CC    :C32F     Get bin.value of input char
                        *
                        * If digit is number:
                        *
C29F CDEBC2     LC57    CALL  :C2EB     Convert FPT exponent
C2A2 E1         LC58    POP   H
C2A3 D1                 POP   D
C2A4 F1                 POP   PSW       CY=1
C2A5 C9                 RET
                        *
                        * If error:
                        *
C2A6 CD2DC3     LC59    CALL  :C32D     DCR C
C2A9 E1         LC60    POP   H
C2AA D1                 POP   D
C2AB F1                 POP   PSW
C2AC 3F                 CMC             CY=0
C2AD C9                 RET
                        *
                        * CLEAR MACC AND REGISTERS D, E AND H:
                        *
                        * Exit: ABC preserved.
                        *       L = 2B ('+')
                        *
C2AE 215EC4     LC61    LXI   H,:C45E   Addr. FPT(0)
C2B1 E7                 RST   4         Copy FPT(0) to MACC
C2B2 0C                 DATA  :0C
C2B3 110000             LXI   D,:0000   Clear DE
C2B6 212B00             LXI   H,:002B   Clear H, L='+'
C2B9 C9                 RET
                        *
                        * MOVE A DIGIT INTO THE MACC:
                        *
                        * MACC = MACC * 10 + A.
                        *
                        * Entry: A: Digit 1 - 9.
                        * Exit:  AFBCHL preserved.
```



```
                        *       D=D-H; E=E-1.
                        *
C2BA F5         LC62    PUSH  PSW
C2BB E5                 PUSH  H
C2BC 214DC3             LXI   H,:C34D   Addr FPT (10)
C2BF E7                 RST   4         MACC = MACC * 10 (FPT)
C2C0 06                 DATA  :06
C2C1 D5                 PUSH  D
C2C2 87                 ADD   A         )
C2C3 87                 ADD   A         ) DE = 4 * A
C2C4 5F                 MOV   E,A       ) (calc offset to startaddr)
C2C5 1600               MVI   D,:00     )
C2C7 215EC4             LXI   H,:C45E   Addr table FPT(1-9)
C2CA 19                 DAD   D         Calc. addr nr to be added
C2CB D1                 POP   D
C2CC E7                 RST   4         MACC = MACC + (1-9) (FPT)
C2CD 00                 DATA  :00
C2CE E1                 POP   H
C2CF 7A                 MOV   A,D
C2D0 94                 SUB   H
C2D1 57                 MOV   D,A       D=D-H
C2D2 1D                 DCR   E         E=E-1
C2D3 F1                 POP   PSW
C2D4 C9                 RET
                        *
C2D5 1E00       LC63    MVI   E,:00
C2D7 24                 INR   H
C2D8 C9                 RET
                        *
C2D9 2600       LC64    MVI   H,:00
C2DB C9                 RET
                        *
C2DC 6F         LC65    MOV   L,A
C2DD C9                 RET
                        *
                        * H = 10 * H + A:
                        *
                        * Exit: AFBCDEL preserved.
                        *
C2DE F5         LC66    PUSH  PSW
C2DF 7C                 MOV   A,H       A=H
C2E0 87                 ADD   A         A=2*H
C2E1 87                 ADD   A         A=4*H
C2E2 84                 ADD   H         A=5*H
C2E3 87                 ADD   A         A=10*H
C2E4 67                 MOV   H,A
C2E5 F1                 POP   PSW
C2E6 F5                 PUSH  PSW
C2E7 84                 ADD   H         A=10*H+A
C2E8 67                 MOV   H,A
C2E9 F1                 POP   PSW
C2EA C9                 RET
                        *
                        * CONVERT A FPT EXPONENT:
                        *
                        * The MACC is multiplied/divided by a power of 10
                        * corresponding to the 'E'-exponent minus the number
                        * of digits after the decimal point.
                        *
                        * Entry: C:    Points beyond 1st nonuseable char of
                        *              FPT number in input.
                        *        L:    Contains sign of exponent.
```

```
                *       H:    Contains 'E....' exponent (10).
                *       MACC: Contains FPT conversion of string of
                *             digits.
                *       D:    Contains nr of digits after '.'.
                * Exit: BE preserved, AHL corrupted.
                *       C:    Decremented to after FPT nr in input.
                *       D:    Contains effective exponent.
                *
C2EB CD2DC3     LC67  CALL  :C32D     Decr C
C2EE 7D               MOV   A,L       Get exp. sign
C2EF FE2D             CPI   :2D       '-' ?
C2F1 7C               MOV   A,H       Get exponent
C2F2 C2F7C2           JNZ   :C2F7     If exp. positive
C2F5 2F               CMA             ) Else: make exponent
C2F6 3C               INR   A         ) positive
C2F7 82         LC68  ADD   D         Add nr of digits after '.'
C2F8 57               MOV   D,A       Save result
C2F9 F2FEC2           JP    :C2FE     If result positive
C2FC 2F               CMA             ) Else: make it
C2FD 3C               INR   A         ) positive
C2FE C5         LC69  PUSH  B
C2FF 0605             MVI   B,:05     Nr of times of multipl.
C301 214DC3           LXI   H,:C34D   Addr table powers of 10
C304 B7         LC70  ORA   A         Flags on result LC68
C305 1F               RAR             lsb in carry
C306 D217C3           JNC   :C317     If bit=0
C309 F5               PUSH  PSW
C30A 7A               MOV   A,D       Check if multipl/div
C30B B7               ORA   A
C30C FA11C3           JM    :C311     If division
C30F E7               RST   4         MACC = MACC * power of 10
C310 06               DATA  :06
C311 CA16C3     LC71  JZ    :C316     If multiplication
C314 E7               RST   4         MACC = MACC / power of 10
C315 09               DATA  :09
C316 F1         LC72  POP   PSW       Restore A
C317 23         LC73  INX   H
C318 23               INX   H
C319 23               INX   H
C31A 23               INX   H         HL pnts to next ^10
C31B 05               DCR   B
C31C C204C3           JNZ   :C304     Again if not ready
C31F B7               ORA   A
C320 CA2BC3           JZ    :C32B     If result OK

                * If error:

C323 7A               MOV   A,D
C324 B7               ORA   A         Set flags for error type
C325 F44BC0           CP    :C04B     If overflow error
C328 FC65C0           CM    :C065     If underflow error
C32B C1         LC74  POP   B         Normal return
C32C C9               RET
                *
C32D 0D         LC75  DCR   C
C32E C9               RET
                *
                * GET BINARY VALUE OF INPUT CHARACTER IN A:
                *
                * Entry: C points to character in input.
                * Exit:  C points to next character.
                *        BDEHL preserved.
```



```
                *       CY=1, Z=0: Value in A.
                *       CY=0, Z=1: Char is +/-.
                *       CY=0, Z=0: otherwise.
                *
C32F CD73C0     LC76  CALL  :C073     Get char from line
C332 0C               INR   C         Update pointer
C333 FE2B             CPI   :2B
C335 C8               RZ              Abort if '+'
C336 FE2D             CPI   :2D
C338 C8               RZ              Abort if '-'
C339 FE30             CPI   :30
C33B 3F               CMC
C33C D0               RNC             Abort if < #30
C33D FE3A             CPI   :3A
C33F D24BC3           JNC   :C34B     Abort if > #3A
C342 D630             SUI   :30       Convert ASCII to binary
C344 D5               PUSH  D
C345 57               MOV   D,A
C346 3C               INR   A         Set Z-flag correctly for
                                      reqd output
C347 7A               MOV   A,D
C348 D1               POP   D
C349 37               STC             CY=1: value in A
C34A C9               RET
C34B B7         LC77  ORA   A         Set flags correctly
C34C C9               RET
                *
                ****************************
                * TABLE FPT POWERS OF 10 *
                ****************************
                *
C34D 04         LC233 DATA  :04       FPT 10^1
C34E A0               DATA  :A0
C34F 00               DATA  :00
C350 00               DATA  :00
                *
C351 07               DATA  :07       FPT 10^2
C352 C8               DATA  :C8
C353 00               DATA  :00
C354 00               DATA  :00
                *
C355 0E               DATA  :0E       FPT 10^4
C356 9C               DATA  :9C
C357 40               DATA  :40
C358 00               DATA  :00
                *
C359 1B               DATA  :1B       FPT 10^8
C35A BE               DATA  :BE
C35B BC               DATA  :BC
C35C 20               DATA  :20
                *
C35D 36               DATA  :36       FPT 10^16
C35E 8E               DATA  :8E
C35F 1B               DATA  :1B
C360 CA               DATA  :CA
                *
                ***********************************************
                * CONVERT A FLOATING POINT NUMBER FOR OUTPUT *
                ***********************************************
                *
                * A FPT number in MACC is converted to ASCII in
                * outputbuffer 00E4-F1. The sign is in 00E4, the
```

```
374                     * decimal point in 00E5. The normalized value of
375                     * the mantissa is in 00E6-EC (7 digits). In 00F1
376                     * is the 10's exponent in 2-complement binairy
377                     * signed format.
378                     *
379                     * Exit: A=6 (number of significant digits).
380                     *       BCDEHL, MACC preserved.
381                     *
382 C361 C5           FBC     PUSH  B
383 C362 D5                   PUSH  D
384 C363 E5                   PUSH  H
385 C364 CD1EC2               CALL  :C21E     Save MACC on stack
386 C367 E7                   RST   4         Copy MACC to reg A,B,C,D
387 C368 15                   DATA  :15
388 C369 F5                   PUSH  PSW       Save exp.byte
389 C36A B0                   ORA   B
390 C36B B1                   ORA   C
391 C36C B2                   ORA   D
392 C36D CAD4C3               JZ    :C3D4     If FPT nr is zero
393 C370 F1                   POP   PSW       Get exp.byte
394 C371 F5                   PUSH  PSW
395 C372 2600                 MVI   H,:00
396 C374 E67F                 ANI   :7F       Mask sign bit mantissa
397 C376 FE40                 CPI   :40
398 C378 DA80C3               JC    :C380     If exp is positive
399 C37B 25                   DCR   H
400 C37C 2F                   CMA             ) Else: convert
401 C37D E67F                 ANI   :7F       ) exponent to
402 C37F 3C                   INR   A         ) positive
403 C380 F5           LC78    PUSH  PSW       Save value exponent
404 C381 AF                   XRA   A
405 C382 E7                   RST   4         Copy mantissa to MACC
406 C383 12                   DATA  :12
407 C384 F1                   POP   PSW       Get exp.value
408 C385 44                   MOV   B,H       B=FF (exp.<0), 00 (exp.>0)
409 C386 2137C4               LXI   H,:C437   Addr table powers FPT(2)
410 C389 0E00                 MVI   C,:00
411 C38B 1607                 MVI   D,:07     7 digits to be examined
412 C38D 0F           LC79    RRC             Shift exp. into carry
413 C38E F5                   PUSH  PSW       Save rest of exp.
414 C38F 7E                   MOV   A,M       Get 10's power byte
415 C390 23                   INX   H         Points to next
416 C391 D2A2C3               JNC   :C3A2     If n-th power of 2=0:
417                                           go to next
418 C394 81                   ADD   C
419 C395 4F                   MOV   C,A       Total 10's power in C
420 C396 05                   DCR   B
421 C397 04                   INR   B
422 C398 C29DC3               JNZ   :C39D     Jump if exp. negative
423 C39B E7                   RST   4         Multipl. mantissa by
424 C39C 06                   DATA  :06       (2^2^n)/10^m
425 C39D CAA2C3       LC80    JZ    :C3A2
426 C3A0 E7                   RST   4         Divide mantissa by
427 C3A1 09                   DATA  :09       (2^2^n)/10^m
428 C3A2 23           LC81    INX   H
429 C3A3 23                   INX   H
430 C3A4 23                   INX   H
431 C3A5 23                   INX   H         Pnts to next in table
432 C3A6 F1                   POP   PSW       Get rest exponent
433 C3A7 15                   DCR   D         Decr digit count
434 C3A8 C28DC3               JNZ   :C38D     Again if not 7 digits done
435 C3AB 05                   DCR   B
```



```
436 C3AC 04                   INR   B
437 C3AD 215AC4               LXI   H,:C45A   Addr FPT (0.1)
438 C3B0 C2C4C3               JNZ   :C3C4     If exp. negative
439
440                     * If exponent positive:
441
442 C3B3 E5           LC82    PUSH  H
443 C3B4 2162C4               LXI   H,:C462   Addr FPT(1)
444 C3B7 CD79C0               CALL  :C079     Compare with 1
445 C3BA E1                   POP   H
446 C3BB FAD4C3               JM    :C3D4     Jump if normalized
447 C3BE E7                   RST   4         MACC = MACC * 0.1 (FPT)
448 C3BF 06                   DATA  :06
449 C3C0 0C                   INR   C         Update 10's power
450 C3C1 C3B3C3               JMP   :C3B3     Cont. normalisation
451
452                     * If exponent negative:
453
454 C3C4 79           LC83    MOV   A,C       )
455 C3C5 2F                   CMA             ) Change 10's power
456 C3C6 3C                   INR   A         ) to neg. value
457 C3C7 4F                   MOV   C,A       )
458 C3C8 CD79C0       LC84    CALL  :C079     Compare with 0,1
459 C3CB F2D4C3               JP    :C3D4     Jump if normalized
460 C3CE E7                   RST   4         MACC = MACC / 0.1 (FPT)
461 C3CF 09                   DATA  :09
462 C3D0 0D                   DCR   C         Update 10's power
463 C3D1 C3C8C3               JMP   :C3C8     Cont. normalisation
464
465                     * Load output buffer:
466
467 C3D4 79           LC85    MOV   A,C       Get 10's power
468 C3D5 32F100               STA   :00F1     In output buffer
469 C3D8 F1                   POP   PSW       Get signbyte mantissa
470 C3D9 B7                   ORA   A         Set flags on it
471 C3DA 21E400               LXI   H,:00E4   Addr output buffer
472 C3DD 362B                 MVI   M,:2B     '+' in buffer
473 C3DF F2E4C3               JP    :C3E4     If mantissa is positive
474 C3E2 362D                 MVI   M,:2D     Else: '-' in buffer
475 C3E4 23           LC86    INX   H
476 C3E5 362E                 MVI   M,:2E     '.' in 00E5
477 C3E7 23                   INX   H
478 C3E8 E5                   PUSH  H
479 C3E9 E7                   RST   4         Copy MACC to reg A,B,C,D
480 C3EA 15                   DATA  :15
481 C3EB F5                   PUSH  PSW       Save exp.byte
482 C3EC AF                   XRA   A
483 C3ED E7                   RST   4         Copy mantissa to MACC
484 C3EE 12                   DATA  :12
485 C3EF F1                   POP   PSW       Get exp.byte
486 C3F0 2F                   CMA             )
487 C3F1 3C                   INR   A         ) 2-compl.
488 C3F2 E67F                 ANI   :7F       Mask sign bit mantissa
489 C3F4 2110C6               LXI   H,:C610   Addr INT(10)
490 C3F7 E7                   RST   4         MACC = MACC * 10 (INT)
491 C3F8 54                   DATA  :54
492 C3F9 2120C4               LXI   H,:C420   Addr. INT(1)
493 C3FC 3D                   DCR   A
494 C3FD FA02C4               JM    :C402     If exp. converted
495 C400 E7                   RST   4         Shift MACC right
496 C401 72                   DATA  :72
497 C402 F2FCC3       LC88    JP    :C3FC     If not ready
```

```
002                     ORG   :C437
003                 *
004                 *
005                 *
006                 ************************
007                 * FPT NUMBER CONSTANTS *
008                 ************************
009                 *
010                 * For the first 7 numbers, the 1st byte is the
011                 * power of 10 for division.
012                 *
013 C437 00         LC234   DATA  :00
014 C438 02                 DATA  :02       FPT (2^1)/10^0
015 C439 80                 DATA  :80
016 C43A 00                 DATA  :00
017 C43B 00                 DATA  :00
018                 *
019 C43C 00                 DATA  :00
020 C43D 03                 DATA  :03       FPT (2^2)/10^0
021 C43E 80                 DATA  :80
022 C43F 00                 DATA  :00
023 C440 00                 DATA  :00
024                 *
025 C441 01                 DATA  :01
026 C442 01                 DATA  :01       FPT (2^4)/10^1
027 C443 CC                 DATA  :CC
028 C444 CC                 DATA  :CC
029 C445 CD                 DATA  :CD
030                 *
031 C446 02                 DATA  :02
032 C447 02                 DATA  :02       FPT (2^8)/10^2
033 C448 A3                 DATA  :A3
034 C449 D7                 DATA  :D7
035 C44A 0A                 DATA  :0A
036                 *
037 C44B 04                 DATA  :04
038 C44C 03                 DATA  :03       FPT (2^16)/10^4
039 C44D D1                 DATA  :D1
040 C44E B7                 DATA  :B7
041 C44F 17                 DATA  :17
042                 *
043 C450 09                 DATA  :09
044 C451 03                 DATA  :03       FPT (2^32)/10^9
045 C452 89                 DATA  :89
046 C453 70                 DATA  :70
047 C454 5F                 DATA  :5F
048                 *
049 C455 13                 DATA  :13
050 C456 01                 DATA  :01       FPT (2^64)/10^19
051 C457 EC                 DATA  :EC
052 C458 1E                 DATA  :1E
053 C459 4A                 DATA  :4A
054                 *
055 C45A 7D         LC238   DATA  :7D       FPT (0.1)
056 C45B CC                 DATA  :CC
057 C45C CC                 DATA  :CC
058 C45D CD                 DATA  :CD
059                 *
060 C45E 00         FP0     DATA  :00       FPT (0)
061 C45F 00                 DATA  :00
062 C460 00                 DATA  :00
063 C461 00                 DATA  :00
```



```
064                 *
065 C462 01         FP1     DATA  :01       FPT (1)
066 C463 80                 DATA  :80
067 C464 00                 DATA  :00
068 C465 00                 DATA  :00
069                 *
070 C466 02         FP2     DATA  :02       FPT (2)
071 C467 80                 DATA  :80
072 C468 00                 DATA  :00
073 C469 00                 DATA  :00
074                 *
075 C46A 02         FP3     DATA  :02       FPT (3)
076 C46B C0                 DATA  :C0
077 C46C 00                 DATA  :00
078 C46D 00                 DATA  :00
079                 *
080 C46E 03         FP4     DATA  :03       FPT (4)
081 C46F 80                 DATA  :80
082 C470 00                 DATA  :00
083 C471 00                 DATA  :00
084                 *
085 C472 03         FP5     DATA  :03       FPT (5)
086 C473 A0                 DATA  :A0
087 C474 00                 DATA  :00
088 C475 00                 DATA  :00
089                 *
090 C476 03         FP6     DATA  :03       FPT (6)
091 C477 C0                 DATA  :C0
092 C478 00                 DATA  :00
093 C479 00                 DATA  :00
094                 *
095 C47A 03         FP7     DATA  :03       FPT (7)
096 C47B E0                 DATA  :E0
097 C47C 00                 DATA  :00
098 C47D 00                 DATA  :00
099                 *
100 C47E 04         FP8     DATA  :04       FPT (8)
101 C47F 80                 DATA  :80
102 C480 00                 DATA  :00
103 C481 00                 DATA  :00
104                 *
105 C482 04         FP9     DATA  :04       FPT (9)
106 C483 90                 DATA  :90
107 C484 00                 DATA  :00
108 C485 00                 DATA  :00
109                 *
110                 ***********************************
111                 * PRETTIES UP FPT OR INT NUMBER *
112                 ***********************************
113                 *
114                 * Entry: B:     Fix/float flag: (0=fix, 1=float).
115                 *        A:     Nr. of useable digits in string in
116                 *               00E4-F0 (not counting additional
117                 *               digit for rounding).
118                 *        00F1:  Nr. of digits before '.' (exponent).
119                 *        00E4:  Sign '+' or '-'.
120                 *        00E5:  Decimal point.
121                 *        E6-F0: Digits.
122                 * Exit:  All registers preserved.
123                 *        00E3:  Length of string.
124                 *        E4-F0: Output string.
125                 *
```

```
                    * Format: Sign in 00E4 is blank or '-'.
                    *         If exponent is 0:
                    *               real case: '0.digits'
                    *               int. case: no final '.'
                    *               real case if INT: '.0'
                    *         If exponent < -1:  E-format
                    *         If exponent too large: E-format
                    *         In E-format no '.0'
                    *
C486 F5     PRTY    PUSH  PSW
C487 C5             PUSH  B
C488 D5             PUSH  D
C489 E5             PUSH  H
C48A 21E600         LXI   H,:00E6   Startaddr digits
C48D 4F             MOV   C,A       Save nr. useable digits
C48E C5             PUSH  B
C48F 0600           MVI   B,:00
C491 09             DAD   B         HL pnt to last useable digit
C492 7E             MOV   A,M       Get last digit
C493 FE35           CPI   :35       Check for rounding
C495 DAAFC4         JC    :C4AF     If <5
C498 2B     LC91    DCX   H
C499 7E             MOV   A,M       Get digit before
C49A FE39           CPI   :39
C49C CAA3C4         JZ    :C4A3     If it is 9
C49F 34             INR   M         Rounding upwards
C4A0 C3AFC4         JMP   :C4AF     Abort rounding
C4A3 3630   LC92    MVI   M,:30     Make digit before 0
C4A5 0D             DCR   C         Decr. nr of digits
C4A6 C298C4         JNZ   :C498     Cont. check for rounding
C4A9 3631           MVI   M,:31     Make nr=1 if all digits 9
C4AB 21F100         LXI   H,:00F1
C4AE 34             INR   M         Incr nr of digits before '.'
C4AF C1     LC93    POP   B
C4B0 0C             INR   C         Incr. nr useable digits
C4B1 3AF100         LDA   :00F1     Get nr of digits before '.'
C4B4 B7             ORA   A
C4B5 CAD8C4         JZ    :C4D8     If 0
C4B8 FAE1C4         JM    :C4E1     If too many digits
C4BB 80             ADD   B         Add fix/float flag
C4BC B9             CMP   C
C4BD D2E1C4         JNC   :C4E1     If too many digits
C4C0 CD9CC6         CALL  :C69C     Restore A, insert '.'
                                    after number string
C4C3 79     LC94    MOV   A,C       Length string in A
C4C4 CD4BC5         CALL  :C54B     Calc nr of digits for output
C4C7 3C     LC95    INR   A         Add 1
C4C8 21E300         LXI   H,:00E3
C4CB 77             MOV   M,A       String length in outbuf
C4CC 23             INX   H
C4CD 7E             MOV   A,M       Get sign
C4CE FE2B           CPI   :2B       '+' ?
C4D0 C2D5C4         JNZ   :C4D5     Then abort
C4D3 3620           MVI   M,:20     Replace '+' by blank
C4D5 C34DC1 LC96    JMP   :C14D     Popall, ret

                    * If format '0.digits':

C4D8 CD1AC5 LC97    CALL  :C51A     Move string right 1 pos.
C4DB 3630           MVI   M,:30     Insert 0 in 00E5
C4DD 0C             INR   C         Update nr of digits
C4DE C3C3C4         JMP   :C4C3     Update string
```



```

                    * If too many digits:

C4E1 3E01   LC98    MVI   A,:01
C4E3 CD31C5         CALL  :C531     Move string left 1 pos,
                                    Insert '.' after string
C4E6 79             MOV   A,C       Get nr of digits
C4E7 0600           MVI   B,:00
C4E9 CD4BC5         CALL  :C54B     Calc nr of digits for output
C4EC 47             MOV   B,A       in B
C4ED 3AF100         LDA   :00F1     Get nr of digits before '.'
C4F0 3D             DCR   A         Minus 1
C4F1 3645           MVI   M,:45     'E' in buf after last digit
C4F3 23             INX   H
C4F4 04             INR   B         Incr. nr of digits
C4F5 B7             ORA   A         Flags on exponent
C4F6 F2FFC4         JP    :C4FF     If exp. positive

                    * If exponent negative:

C4F9 362D           MVI   M,:2D     Store '-' in buffer
C4FB 23             INX   H
C4FC 04             INR   B         Incr nr of digits
C4FD 2F             CMA
C4FE 3C             INR   A         2-compl of exponent

                    * Exponent to buffer:

C4FF 110A2F LC99    LXI   D,:2F0A
C502 93     LC100   SUB   E         Exp.-10 (unit value)
C503 14             INR   D         ASCII-count 10's-value
C504 D202C5         JNC   :C502     If rest exp. still >10
C507 C63A           ADI   :3A       Convert rest to Ascii
C509 5F             MOV   E,A       in E
C50A 7A             MOV   A,D       Get 10's value
C50B FE30           CPI   :30
C50D CA13C5         JZ    :C513     If exp. <10
C510 77             MOV   M,A       10's value exp. in buf
C511 23             INX   H
C512 04             INR   B         Incr nr of digits
C513 73     LC101   MOV   M,E       Unit value exp. in buf
C514 23             INX   H
C515 04             INR   B         Incr nr of digits
C516 78             MOV   A,B       into A
C517 C3C7C4         JMP   :C4C7     Prepare string for output
                    *
                    * MOVE STRING IN OUTPUTBUFFER RIGHT 1 POS.:
                    *
                    * The contents of 00E5-EF is moved up one
                    * position to 00E6-F0.
                    *
                    * Entry: No conditions.
                    * Exit:  ABCDEF preserved.
                    *        HL points to 00E5.
                    *
C51A F5     LC102   PUSH  PSW
C51B C5             PUSH  B
C51C D5             PUSH  D
C51D 21F000         LXI   H,:00F0   Highest destination address.
C520 54             MOV   D,H
C521 5D             MOV   E,L
C522 1B             DCX   D         Highest source address.
```

```
C523 060B          MVI   B,:0B     Number of bytes.
C525 1A      LC103 LDAX  D         Get byte
C526 77            MOV   M,A       and move it.
C527 1B            DCX   D
C528 2B            DCX   H
C529 05            DCR   B
C52A C225C5        JNZ   :C525     Next byte if not ready
C52D D1            POP   D
C52E C1            POP   B
C52F F1            POP   PSW
C530 C9            RET
                   *
                   * MOVE STRING IN OUTPUTBUFFER LEFT 1 POS.:
                   *
                   * The string, beginning on 00E6, is moved one
                   * memory location downwards. A '.' is inserted
                   * after the string.
                   *
                   * Entry: A: number of bytes to be transferred.
                   * Exit:  All registers preserved.
                   *
C531 F5      LC104 PUSH  PSW
C532 C5            PUSH  B
C533 D5            PUSH  D
C534 E5            PUSH  H
C535 47            MOV   B,A       Store number of bytes
C536 21E500        LXI   H,:00E5   Lowest destination address
C539 11E600        LXI   D,:00E6   Lowest source address
C53C 1A      LC105 LDAX  D         Get byte
C53D 77            MOV   M,A       and move it
C53E 13            INX   D
C53F 23            INX   H
C540 05            DCR   B
C541 C23CC5        JNZ   :C53C     Next byte if not ready
C544 362E          MVI   M,:2E     Insert '.' after string
C546 E1            POP   H
C547 D1            POP   D
C548 C1            POP   B
C549 F1            POP   PSW
C54A C9            RET
                   *
                   * CALCULATE NUMBER OF DIGITS FOR OUTPUT:
                   *
                   * Entry: Total nr of string digits in A and C.
                   *        B: Flag for INT (0) or FPT (1).
                   *        Digits in 00E4 to 00E4 + A.
                   * Exit:  A:  Nr of bytes for output:
                   *                 INT: excl. trailing '.0'
                   *                 FPT: incl. trailing '.0'
                   *        HL: If last non-zero byte is not '.':
                   *            points after last byte.
                   *            Else: INT: points to '.'
                   *                  FPT: after '.0'
                   *
C54B C5      LC106 PUSH  B
C54C D5            PUSH  D
C54D 21E400        LXI   H,:00E4   Startaddr string
C550 5F            MOV   E,A       Total nr of digits in E
C551 1600          MVI   D,:00
C553 19            DAD   D         Calc end of string
C554 7E      LC107 MOV   A,M       Get digit
C555 FE30          CPI   :30
```



```
C557 C25FC5        JNZ   :C55F     If non-zero
C55A 2B            DCX   H         Points to previous digit
C55B 0D            DCR   C         Decr nr of digits
C55C C354C5        JMP   :C554     Again till non-zero found
                   *
                   * If non-zero digit found:
                   *
C55F FE2E    LC108 CPI   :2E       '.' ?
C561 23            INX   H
C562 C26FC5        JNZ   :C56F     Abort if not
C565 2B            DCX   H         Pnts after last non-zero,
                                   non-'.' digit
C566 0D            DCR   C         Excl. '.'
C567 05            DCR   B
C568 C26FC5        JNZ   :C56F     If INT case
C56B 23            INX   H         ) If FPT case: pnts
C56C 23            INX   H         ) after '.0'
C56D 0C            INR   C
C56E 0C            INR   C         Incl. '.0'
C56F 79      LC109 MOV   A,C       Nr of digits for output
C570 D1            POP   D
C571 C1            POP   B
C572 C9            RET
                   *
                   ********************************
                   * INPUT INTEGER NUMBER TO MACC *
                   ********************************
                   *
                   * Read string of digits from line and convert
                   * it to binary in MACC.
                   *
                   * Entry: BC points to input character.
                   * Exit:  BC points after INT number.
                   *        ADEHL preserved.
                   *        CY=1: there were digits.
                   *        CY=0: No digits.
                   *
C573 37      ICB   STC
C574 F5            PUSH  PSW
C575 D5            PUSH  D
C576 E5            PUSH  H
C577 CD98C5        CALL  :C598     Clear MACC and 00E3-E6.
C57A CD73C0  LC110 CALL  :C073     Get digit from line
C57D D630          SUI   :30       Convert ASCII to binary
C57F DA90C5        JC    :C590     )
C582 FE0A          CPI   :0A       ) Abort if no number
C584 D290C5        JNC   :C590     )
C587 2110C6        LXI   H,:C610   Addr INT(10)
C58A CDA5C5        CALL  :C5A5     MACC=MACC*10 + digit
C58D C37AC5        JMP   :C57A     Next digit
C590 15      LC111 DCR   D
C591 14            INR   D
C592 C2A2C2        JNZ   :C2A2     If digits: Pop, ret
C595 C3A9C2        JMP   :C2A9     If no digits: CY=0, Pop, ret
                   *
                   * CLEAR MACC AND 00E3-00E6:
                   *
                   * Both MACC and registers 00E3-E6 are loaded
                   * with the value of FPT (0).
                   *
                   * Exit: ABCE preserved. D=0.
                   *
```

```
374 C598 215EC4     LC112   LXI   H,:C45E   Addr. FPT(0)
375 C59B E7                 RST   4         Copy FPT(0) to MACC
376 C59C 0C                 DATA  :0C
377 C59D 21E300             LXI   H,:00E3
378 C5A0 E7                 RST   4         Copy FPT(0) to 00E3-E6
379 C5A1 0F                 DATA  :0F
380 C5A2 1600               MVI   D,:00     Clear D
381 C5A4 C9                 RET
382                 *
383                 * MACC = MACC*10 + DIGIT FROM LINE:
384                 *
385                 * Entry: HL: points to INT (10).
386                 *         A:  digit to be added.
387                 *
388 C5A5 E7         LC113   RST   4         MACC=MACC*10 (INT)
389 C5A6 54                 DATA  :54
390 C5A7 0C         LC114   INR   C
391 C5A8 15                 DCR   D
392 C5A9 32E600             STA   :00E6     Digit in lobyte E3-E6
393 C5AC 21E300             LXI   H,:00E3
394 C5AF E7                 RST   4         Add (E3-E6) to MACC (INT)
395 C5B0 4E                 DATA  :4E
396 C5B1 C9                 RET
397                 *
398                 ************************************
399                 * CONVERT INTEGER NUMBER FOR OUTPUT *
400                 ************************************
401                 *
402                 * Places ASCII string from INT MACC contents in
403                 * output buffer 00E4-F0.
404                 * 00E4 is sign, 00E5 is '.', 00E6-F0 is value,
405                 * 00F1 is nr of digits.
406                 *
407                 * Exit: A: Number of digits.
408                 *       BCDEHL preserved.
409                 *
410 C5B2 C5         IBC     PUSH  B
411 C5B3 D5                 PUSH  D
412 C5B4 E5                 PUSH  H
413 C5B5 CD1EC2             CALL  :C21E     Save MACC to TOS
414 C5B8 CDE0C5             CALL  :C5E0     Abs.value of MACC in reg
415                                          A,B,C,D; Prepare 00E4-E6
416 C5BB CD1EC2     LC115   CALL  :C21E     Save MACC to TOS
417 C5BE 2110C6             LXI   H,:C610   Addr INT(10)
418 C5C1 E7                 RST   4         MACC = remainder MACC/10
419 C5C2 5A                 DATA  :5A
420 C5C3 E7                 RST   4         Copy MACC to reg A,B,C,D
421 C5C4 15                 DATA  :15
422 C5C5 7A                 MOV   A,D       Lobyte in A
423 C5C6 CDFAC5             CALL  :C5FA     Digit into 00E5-F0
424 C5C9 CD34C2             CALL  :C234     Retrieve MACC from TOS
425 C5CC E7                 RST   4         MACC = MACC/10 (INT)
426 C5CD 57                 DATA  :57
427 C5CE E7                 RST   4         Copy MACC to reg A,B,C,D
428 C5CF 15                 DATA  :15
429 C5D0 B0                 ORA   B
430 C5D1 B1                 ORA   C
431 C5D2 B2                 ORA   D
432 C5D3 C2BBC5             JNZ   :C5BB     Again if <>0
433 C5D6 CD06C6             CALL  :C606     '.' in 00E5; length in 00F1
434 C5D9 CD34C2             CALL  :C234     Retrieve MACC from TOS
435 C5DC E1                 POP   H
```



```
436 C5DD D1                 POP   D
437 C5DE C1                 POP   B
438 C5DF C9                 RET
439                 *
440                 * PREPARE 00E4-E6:
441                 *
442                 * 00E4-E6 is set to +00 or -00, depending on sign
443                 * of contents MACC. In the MACC remains the absolute
444                 * value. The registers A,B,C,D contain the original
445                 * contents of the MACC.
446                 *
447                 * Exit: E=0. HL preserved. AFBCD corrupted.
448                 *
449 C5E0 E5         LC116   PUSH  H
450 C5E1 21E400             LXI   H,:00E4
451 C5E4 E7                 RST   4         Copy MACC to reg A,B,C,D
452 C5E5 15                 DATA  :15
453 C5E6 B7                 ORA   A         Set flags on sign
454 C5E7 362B               MVI   M,:2B     '+' in 00E4
455 C5E9 F2F0C5             JP    :C5F0     Jump if nr is positive
456 C5EC 362D               MVI   M,:2D     Else '-' in 00E4
457 C5EE E7                 RST   4         and make contents MACC pos.
458 C5EF 60                 DATA  :60
459 C5F0 23         LC117   INX   H
460 C5F1 3630               MVI   M,:30     0 in 00E5
461 C5F3 23                 INX   H
462 C5F4 3630               MVI   M,:30     0 in 00E6
463 C5F6 1E00               MVI   E,:00     Digit count is 0
464 C5F8 E1                 POP   H
465 C5F9 C9                 RET
466                 *
467                 * STORE DIGIT IN OUTPUT BUFFER 00E5-00F0:
468                 *
469                 * Entry: Digit in A.
470                 * Exit:  Digit in 00E5-F0 as most sign. digit.
471                 *        E:  Count of digit in buffer.
472                 *        BCDHL preserved. AF corrupted.
473                 *
474 C5FA E5         LC118   PUSH  H
475 C5FB F5                 PUSH  PSW
476 C5FC CD1AC5             CALL  :C51A     Move contents buffer right
477 C5FF F1                 POP   PSW
478 C600 C630               ADI   :30       Make digit ASCII
479 C602 77                 MOV   M,A       Digit in 00E5 inserted.
480 C603 1C                 INR   E         Update digit count.
481 C604 E1                 POP   H
482 C605 C9                 RET
483                 *
484                 * ADD A '.' TO A DIGIT STRING IN OUTPUTBUFFER:
485                 *
486                 * A '.' is placed at the beginning of a digit
487                 * string in the output buffer. The length of the
488                 * string is stored in 00F1.
489                 *
490                 * Entry: E:  Digit count.
491                 *        HL: Points to 00E5.
492                 * Exit:  A:  Count.
493                 *        BCDEHL preserved.
494                 *
495 C606 CD1AC5     LC119   CALL  :C51A     Move contents outbuf right
496                                          one position
497 C609 362E               MVI   M,:2E     '.' at begin of string
```

```
498 C60B 7B                 MOV   A,E       Get digit count
499 C60C 32F100             STA   :00F1     Store it in buffer
500 C60F C9                 RET
501                 *
502                 * DATA:
503                 *
504 C610 00         I10     DATA  :00       INT (10)
505 C611 00                 DATA  :00
506 C612 00                 DATA  :00
507 C613 0A                 DATA  :0A
508                 *
509                 *
510                 *
511 C614                    END

****************************
* S Y M B O L   T A B L E *
****************************

FP0     C45E    FP1     C462    FP2     C466    FP3     C46A
FP4     C46E    FP5     C472    FP6     C476    FP7     C47A
FP8     C47E    FP9     C482    I10     C610    IBC     C5B2
ICB     C573    LC100   C502    LC101   C513    LC102   C51A
LC103   C525    LC104   C531    LC105   C53C    LC106   C54B
LC107   C554    LC108   C55F    LC109   C56F    LC110   C57A
LC111   C590    LC112   C598    LC113   C5A5    LC114   C5A7
LC115   C5BB    LC116   C5E0    LC117   C5F0    LC118   C5FA
LC119   C606    LC234   C437    LC238   C45A    LC91    C498
LC92    C4A3    LC93    C4AF    LC94    C4C3    LC95    C4C7
LC96    C4D5    LC97    C4D8    LC98    C4E1    LC99    C4FF
PRTY    C486
```



```
002                         ORG   :C614
003                 *
004                 *
005                 *
006                 ****************************
007                 * INPUT HEX NUMBER TO MACC *
008                 ****************************
009                 *
010                 * Reads a sequence of hex digits and converts
011                 * them into MACC.
012                 *
013                 * Entry: C points to input.
014                 * Exit:  CY=1: There was a digit.
015                 *        CY=0: No digit.
016                 *        C points to next input.
017                 *        ABDEHL preserved.
018                 *
019 C614 37         HCB     STC
020 C615 F5                 PUSH  PSW
021 C616 D5                 PUSH  D
022 C617 E5                 PUSH  H
023 C618 CD98C5             CALL  :C598     Clear MACC and 00E3-E6
024 C61B CD73C0     LC120   CALL  :C073     Get digit from line
025 C61E D630               SUI   :30       )
026 C620 DA90C5             JC    :C590     )
027 C623 FE0A               CPI   :0A       ) Check if hex number
028 C625 DA34C6             JC    :C634     )
029 C628 D607               SUI   :07       ) Abort via C590 if not
030 C62A FE0A               CPI   :0A       )
031 C62C DA90C5             JC    :C590     )
032 C62F FE10               CPI   :10       )
033 C631 D290C5             JNC   :C590     )
034 C634 213DC6     LC121   LXI   H,:C63D   Addr INT(4)
035 C637 CD41C6             CALL  :C641     Insert digit at low end
036                                         MACC
037 C63A C31BC6             JMP   :C61B     Get next digit
038                 *
039                 * DATA:
040                 *
041 C63D 00         I4      DATA  :00       INT (4)
042 C63E 00                 DATA  :00
043 C63F 00                 DATA  :00
044 C640 04                 DATA  :04
045                 *
046                 * ENTER HEX DIGIT AT LOW END MACC:
047                 *
048                 * Entry: HL points to a 4-byte number.
049                 *        A  contains a digit.
050                 * Exit:  HL = 00E3.
051                 *        C is incremented, D decremented.
052                 *        ABE preserved.
053                 *
054 C641 F5         LC122   PUSH  PSW
055 C642 C5                 PUSH  B
056 C643 D5                 PUSH  D
057 C644 E7                 RST   4         Copy MACC to reg A,B,C,D
058 C645 15                 DATA  :15
059 C646 E6F0               ANI   :F0       Check if value too high
060 C648 C44BC0             CNZ   :C04B     Then overflow error
061 C64B D1                 POP   D
062 C64C C1                 POP   B
063 C64D F1                 POP   PSW
```

```
064 C64E E7                 RST   4          Shift left
065 C64F 6F                 DATA  :6F
066 C650 C3A7C5             JMP   :C5A7      Add digit to MACC
067                 *
068                 ****************************************
069                 * CONVERT HEX MACC TO ASCII FOR OUTPUT *
070                 ****************************************
071                 *
072                 * Converts a HEX number in MACC into its ASCII
073                 * representation into the output buffer.
074                 * Not significant leading zeroes are cancelled.
075                 *
076                 * Exit: BCDEHL preserved. AF corrupted.
077                 *       Length output string in 00E3.
078                 *       Output string in 00E4-00EB.
079                 *
080 C653 C5           HBC   PUSH  B
081 C654 D5                 PUSH  D
082 C655 E5                 PUSH  H
083 C656 CD8DC6             CALL  :C68D      Get startaddr DECBUF in HL
084 C659 E7                 RST   4          Copy MACC to reg A,B,C,D
085 C65A 15                 DATA  :15
086 C65B CD6AC6             CALL  :C66A      Convert A,B to ASCII
087                                          into DECBUF
088 C65E 79                 MOV   A,C
089 C65F 42                 MOV   B,D
090 C660 CD6AC6             CALL  :C66A      Idem for C,D
091 C663 CD91C6             CALL  :C691      Get string length in 00E3
092 C666 E1                 POP   H
093 C667 D1                 POP   D
094 C668 C1                 POP   B
095 C669 C9                 RET
096
097                 * Convert 2 hex digits:
098
099 C66A CD6EC6     LC123   CALL  :C66E      Convert 1st digit
100 C66D 78                 MOV   A,B        Get 2nd one
101 C66E F5         LC124   PUSH  PSW
102 C66F 1F                 RAR
103 C670 1F                 RAR
104 C671 1F                 RAR
105 C672 1F                 RAR              Shift high nibble in low
106 C673 CD77C6             CALL  :C677      Convert it to ASCII
107 C676 F1                 POP   PSW        Restore both nibbles
108 C677 E60F       LC125   ANI   :0F        Low nibble only
109 C679 FE0A               CPI   :0A
110 C67B DA80C6             JC    :C680      If 0 < digit < 9
111 C67E C607               ADI   :07        Add 7 for A < digit < F
112 C680 C630       LC126   ADI   :30        Convert to ASCII
113 C682 77                 MOV   M,A        Into DECBUF
114 C683 23                 INX   H          Incr pointer
115 C684 FE30               CPI   :30
116 C686 C0                 RNZ              Abort if digit <> 0
117
118                 * If 1st digit is zero:
119
120 C687 7D                 MOV   A,L        Get lsbyte buffer pointer
121 C688 FEE5               CPI   :E5        1st digit in buffer?
122 C68A C0                 RNZ              Abort if not
123 C68B 2B                 DCX   H          Else: cancel non-sign. 0's
124 C68C C9                 RET
125
```



```
126                 * Get startaddress output buffer:
127
128 C68D 21E400     LC127   LXI   H,:00E4    Startaddr in HL
129 C690 C9                 RET
130                 *
131                 * CALCULATE LENGTH OF STRING IN OUTPUT BUFFER:
132                 *
133                 * Entry: L: lobyte of address last digit in buffer.
134                 * Exit:  BCDEHL preserved. AF corrupted.
135                 *        Length is stored in 00E3.
136                 *
137 C691 7D          LC128   MOV   A,L        Get lobyte addr last digit
138 C692 D6E4               SUI   :E4        Minus beginaddr
139 C694 FE01               CPI   :01
140 C696 CE00               ACI   :00        Length min. 1
141 C698 32E300             STA   :00E3      Store length in DECBUF
142 C69B C9                 RET
143                 *
144                 ****************************************
145                 * RESTORE A, ADD '.' AFTER DIGIT STRING *
146                 ****************************************
147                 *
148                 * Part of PRTY (C486).
149                 *
150 C69C 90         LC129   SUB   B          Restore A
151 C69D C331C5             JMP   :C531      Move string left 1 pos,
152                                          insert '.'
153                 *
154                 ************************************
155                 * PRINT CHARACTER, INPUT A TEXT LINE *
156                 ************************************
157                 *
158                 * Part of Run 'INPUT' (0E3D6).
159                 *
160                 * Entry: Character in A.
161                 * Exit:  BC preserved.
162                 *
163 C6A0 C5           PINPLN  PUSH  B
164 C6A1 CD1FDD               CALL  :DD1F      Print char; input textline
165 C6A4 C1                   POP   B
166 C6A5 C9                   RET
167                 *
168 C6A6 FF                   DATA  :FF
169 C6A7 FF                   DATA  :FF
170                 *
171                 ********************
172                 * DATA FOR 'RANDOM' *
173                 ********************
174                 *
175 C6A8 00         RNDA    DATA  :00        Random number constant A
176 C6A9 00                 DATA  :00
177 C6AA 00                 DATA  :00
178 C6AB 3B                 DATA  :3B
179                 *
180 C6AC 07         RNDB    DATA  :07        Random number constant B
181 C6AD 73                 DATA  :73
182 C6AE 59                 DATA  :59
183 C6AF 41                 DATA  :41
184                 *
185 C6B0 01         IROR    DATA  :01        OR mask  (FPT (1))
186 C6B1 80                 DATA  :80
187 C6B2 00                 DATA  :00
```

```
C6B3 00                 DATA  :00
                *
                ******************************
                * part of READ BLOCK (D340) *
                ******************************
                *
                * Exit if no loading errors.
                *
C6B4 E3         MPT26   XTHL
C6B5 37                 STC             CY=1: no error
C6B6 E1         LBK30   POP   H
C6B7 D1                 POP   D
C6B8 C1                 POP   B
C6B9 C9                 RET
                *
                ******************
                * part of 2E8DE *
                ******************
                *
C6BA CD91CE     SPT02   CALL  :CE91     Go and set screen bits for
                                        mode 1
C6BD C338E1             JMP   :E138     (2) Pop all, ret.
                *
                *
                *  ================
                *** BANK SWITCHING ***
                *  ================
                *
                *
                ***************************
                * MATH. RESTART (RST 4) *
                ***************************
                *
                * This, and the following routines, switch the
                * paged banks of ROM. They are entered via
                * RST x; DATA xx instructions.
                *
C6C0 E1         MARST   POP   H
C6C1 F3                 DI
C6C2 224300             SHLD  :0043     Save HL
C6C5 F5                 PUSH  PSW
C6C6 E1                 POP   H
C6C7 224100             SHLD  :0041     Save PSW
C6CA 2640               MVI   H,:40     ROM bank 1 select bits
C6CC 3AD400             LDA   :00D4     Offset of start HW/SW vector
                *
                * ROM BANK SWITCHING:
                *
                * This routine is generally used by all Restarts
                * using ROM bank switching.
                *
C6CF E3         MRS10   XTHL
C6D0 86                 ADD   M         Add entry number
C6D1 23                 INX   H
C6D2 E3                 XTHL
C6D3 6F                 MOV   L,A       Complete entry point address
C6D4 3A4000             LDA   :0040     Old bank select port status
C6D7 F5                 PUSH  PSW       Save it
C6D8 E63F               ANI   :3F       Keep other bits
C6DA B4                 ORA   H         Add new select bits
C6DB 324000             STA   :0040     Update memory
C6DE 3206FD             STA   :FD06     Update port
```



```
C6E1 26E0               MVI   H,:E0
C6E3 CDF2C6             CALL  :C6F2     Restore HL, PSW; switch bank

                * Return from switched bank:

C6E6 E3                 XTHL            Return to old bank
C6E7 F5                 PUSH  PSW
C6E8 7C                 MOV   A,H       Get old bank
C6E9 324000             STA   :0040     Reinstate memory
C6EC 3206FD             STA   :FD06     Re-instate port
C6EF F1                 POP   PSW
C6F0 E1                 POP   H
C6F1 C9                 RET             Return to caller
                *
                * SWITCH TO ROM BANK:
                *
                * HL and PSW are restored. On exit, the
                * program switches to the selected ROM bank.
                * After return from this bank, the program
                * continues on C6E6.
                *
C6F2 E5         MRDCL   PUSH  H
C6F3 2A4100             LHLD  :0041
C6F6 E5                 PUSH  H
C6F7 F1                 POP   PSW       Restore A,F
C6F8 2A4300             LHLD  :0043     Restore H,L
C6FB FB                 EI
C6FC C9                 RET             Switch to selected bank
                *
                ***************************
                * SCREEN RESTART (RST 5) *
                ***************************
                *
                * From SRS10 also used by RST 1.
                *
C6FD E1         SCRST   POP   H
C6FE F3                 DI
C6FF 224300             SHLD  :0043     Save HL
C702 F5                 PUSH  PSW
C703 3E80               MVI   A,:80     ROM bank 2 select bits
C705 E1         SRS10   POP   H
C706 224100             SHLD  :0041     Save PSW
C709 67                 MOV   H,A
C70A AF                 XRA   A
C70B C3CFC6             JMP   :C6CF     Switch to new bank
                *
                ***************************
                * UTILITY/ENCODE (RST 1) *
                ***************************
                *
C70E E1         UTRST   POP   H
C70F F3                 DI
C710 224300             SHLD  :0043     Save HL
C713 F5                 PUSH  PSW
C714 3EC0               MVI   A,:C0     ROM bank 3 select bits
C716 C305C7             JMP   :C705     Switch to ROM bank 3
                *
                *
                *  ================
                *** BASIC HANDLER ***
                *  ================
                *
```

```
                        *
                        *********
                        * RESET *
                        *********
                        *
                        * BASIC entry point. Entry via hardware reset by
                        * means of a bootstrap on the addresslines to C000.
                        *
                        * This section is responsible for all 'once only'
                        * initialisation of the hardware and the software
                        * environment. It initialises pointers to all RAM
                        * areas required, the interrupt system and the
                        * software modules.
                        *
                        INIT
C719 3100F9             RESET   LXI   SP,:F900   Init. stackpointer
C71C 3E30                       MVI   A,:30     ) cassette motors off;
C71E 324000                     STA   :0040     ) paddle enable off;
C721 3206FD                     STA   :FD06     ) select ROM bank 0
C724 CDFBD8                     CALL  :D8FB     Init. interrupt system
C727 210000                     LXI   H,:0000
C72A 22A302                     SHLD  :02A3     Set for no Basic
C72D AF                         XRA   A
C72E 329302                     STA   :0293     RNDLY=0
C731 00                         NOP
C732 00                         NOP

                        * Init. math.package:

C733 11F2C7                     LXI   D,:C7F2   Addr table error vectors
C736 21E0DD                     LXI   H,:DDE0   Addr routine get char/line
C739 CD03C0                     CALL  :C003     Package initialisation

                        * Init. screen RAM:

C73C CDFBC7                     CALL  :C7FB     Check available RAM space
C73F 2B                         DCX   H         Highest RAM address
C740 11E0C7                     LXI   D,:C7E0   Addr screen default data
C743 EF                         RST   5
C744 00                         DATA  :00       Init. screen RAM

                        * Init. I/O:

C745 AF                         XRA   A
C746 CD8DEE                     CALL  :EE8D     (0) Init. I/O switching
                                                (input keyb; output screen
                                                + RS232)
C749 323501                     STA   :0135     Input from keyboard for
                                                encoding
C74C 3EC0                       MVI   A,:C0
C74E 32F5FF                     STA   :FFF5     Init. TICC baud rate

                        * Init. screen:

C751 CDFFDA                     CALL  :DAFF     Print 'DAI PERSONAL
C754 A8C7                       DBL   :C7A8     COMPUTER'
C756 2AA502                     LHLD  :02A5     Get bottom screen RAM
C759 117B09                     LXI   D,:097B
C75C 19                         DAD   D         Get line mode byte of line
                                                with 'DAI pC'
C75D 365F                       MVI   M,:5F     Set for medium resolution
C75F 11D0FF                     LXI   D,:FFD0
```



```
C762 19                         DAD   D         Create new line mode byte
                                                for line with 'COMPUTER'
C763 CDF9CE                     CALL  :CEF9     Place 'COMPUTER' on new line
C766 160F                       MVI   D,:0F     Nr of blanking lines
C768 CDCFCE             HD10    CALL  :CECF     Blank next 15 lines
C76B 15                         DCR   D
C76C C268C7                     JNZ   :C768     Next line

                        * Prepare BASIC:

C76F CD2DD7                     CALL  :D72D     Init. Soundgen/DCEbus/
                                                transfer cassette data/
                                                set start HEAP/get evt.
                                                DCE-inputs
C772 210001                     LXI   H,:0100
C775 229D02                     SHLD  :029D     HEAP size default value
C778 CDB5DE                     CALL  :DEB5     Run 'NEW'
C77B 3E10                       MVI   A,:10
C77D 323D01                     STA   :013D     Select cassette port 1
C780 21C5E8                     LXI   H,:E8C5   (3) Ptr. to ASCII table
C783 CD60D5                     CALL  :D560     Init keyboard pointers
C786 CD01D1                     CALL  :D101     Init string handler
C789 3E00                       MVI   A,:00
C78B 328F02                     STA   :028F     Default number type FPT
C78E 117502                     LXI   D,:0275   Begin IMPTAB
C791 218F02                     LXI   H,:028F   End IMPTAB
C794 CD7CDE                     CALL  :DE7C     Init. implicit type table
                                                with 0 (= FPT)
C797 FB                         EI
C798 CF                         RST   1         Wait for input keyboard
C799 15                         DATA  :15       or RS232
C79A 00                         NOP
C79B 21EEC7                     LXI   H,:C7EE   Ptr. to mode 0 colours
C79E EF                         RST   5         Set text colours
C79F 06                         DATA  :06

                        * Entry from utility:

C7A0 CDE4CE             RINIT   CALL  :CEE4     Select ROM bank 0 and print
C7A3 D3C7                       DBL   :C7D3     'BASIC V1.0'
C7A5 C318C8                     JMP   :C818     Into BASIC monitor
                        *
                        * INITIALISATION SCREEN DATA:
                        *
C7A8 0D                 MSGHDR  DATA  :0D       Frigged screen header
C7A9 0D                         DATA  :0D
C7AA 0D                         DATA  :0D
C7AB 0D                         DATA  :0D
C7AC 0D                         DATA  :0D
C7AD 0D                         DATA  :0D
C7AE 20                         DATA  :20
C7AF 44                         DATA  :44       D
C7B0 41                         DATA  :41       A
C7B1 49                         DATA  :49       I
C7B2 20                         DATA  :20
C7B3 50                         DATA  :50       P
C7B4 45                         DATA  :45       E
C7B5 52                         DATA  :52       R
C7B6 53                         DATA  :53       S
C7B7 4F                         DATA  :4F       O
C7B8 4E                         DATA  :4E       N
C7B9 41                         DATA  :41       A
```

```
436 C7BA 4C                 DATA  :4C       L
437 C7BB 20                 DATA  :20
438 C7BC 20                 DATA  :20
439 C7BD 20                 DATA  :20
440 C7BE 20                 DATA  :20
441 C7BF 20                 DATA  :20
442 C7C0 20                 DATA  :20
443 C7C1 20                 DATA  :20
444 C7C2 20                 DATA  :20
445 C7C3 20                 DATA  :20
446 C7C4 20                 DATA  :20
447 C7C5 20                 DATA  :20
448 C7C6 20                 DATA  :20
449 C7C7 20                 DATA  :20
450 C7C8 20                 DATA  :20
451 C7C9 43                 DATA  :43       C
452 C7CA 4F                 DATA  :4F       O
453 C7CB 4D                 DATA  :4D       M
454 C7CC 50                 DATA  :50       P
455 C7CD 55                 DATA  :55       U
456 C7CE 54                 DATA  :54       T
457 C7CF 45                 DATA  :45       E
458 C7D0 52                 DATA  :52       R
459 C7D1 0D                 DATA  :0D
460 C7D2 00                 DATA  :00
461               *
462 C7D3 0C       MSGIN     DATA  :0C
463 C7D4 42                 DATA  :42       B
464 C7D5 41                 DATA  :41       A
465 C7D6 53                 DATA  :53       S
466 C7D7 49                 DATA  :49       I
467 C7D8 43                 DATA  :43       C
468 C7D9 20                 DATA  :20
469 C7DA 56                 DATA  :56       V
470 C7DB 31                 DATA  :31       1
471 C7DC 2E                 DATA  :2E       .
472 C7DD 30                 DATA  :30       0
473 C7DE 0D                 DATA  :0D
474 C7DF 00                 DATA  :00
475               *
476               * SCREEN INITIALISATION PARAMETERS:
477               *
478 C7E0 01       SIPAR     DATA  :01       Default cursor type
479 C7E1 5F                 DATA  :5F       Default cursor ASCII value
480               *
481 C7E2 05                 DATA  :05       )
482 C7E3 0F                 DATA  :0F       ) Colours COLORT
483 C7E4 0F                 DATA  :0F       ) during Reset
484 C7E5 05                 DATA  :05       )
485               *
486 C7E6 00                 DATA  :00       )
487 C7E7 05                 DATA  :05       ) Default colours
488 C7E8 0A                 DATA  :0A       ) COLORG
489 C7E9 0F                 DATA  :0F       )
490               *
491 C7EA 01CA               DBL   :CA01     Addr. memory management
492                                         routine
493 C7EC 25CA               DBL   :CA25     Addr. emergency stop routine
494               *
495 C7EE 08       STCOL     DATA  :08       )
496 C7EF 00                 DATA  :00       ) Default colours
497 C7F0 00                 DATA  :00       ) COLORT
```



```
498 C7F1 08                 DATA  :08       )
499               *
500               * MATH. ERROR ROUTINE VECTORS:
501               *
502 C7F2 1FDA     MEVEC     DBL   :DA1F     Addr. Overflow error routine
503 C7F4 15DA               DBL   :DA15     Addr. Number out of range
504                                         error routine
505 C7F6 FAC7               DBL   :C7FA     Addr. Return
506 C7F8 24DA               DBL   :DA24     Addr. Error routine Division
507                                         by zero
508               *
509 C7FA C9       MERET     RET             Return
510               *
511               *
512               *
513 C7FB                    END
```

```
*****************************
* S Y M B O L   T A B L E *
*****************************

HBC     C653    HCB     C614    HD10    C768    I4      C63D
INIT    C719    IROR    C6B0    LBK30   C6B6    LC120   C61B
LC121   C634    LC122   C641    LC123   C66A    LC124   C66E
LC125   C677    LC126   C680    LC127   C68D    LC128   C691
LC129   C69C    MARST   C6C0    MERET   C7FA    MEVEC   C7F2
MPT26   C6B4    MRDCL   C6F2    MRS10   C6CF    MSGHDR  C7A8
MSGIN   C7D3    PINPLN  C6A0    RESET   C719    RINIT   C7A0
RNDA    C6A8    RNDB    C6AC    SCRST   C6FD    SIPAR   C7E0
SPT02   C6BA    SRS10   C705    STCOL   C7EE    UTRST   C70E
```

```
002                     ORG   :C7FB
003                 *
004                 *
005                 *
006                 ***********************************
007                 * CHECK FOR HIGHEST RAM ADDRESS *
008                 ***********************************
009                 *
010                 * Entry: No conditions.
011                 * Exit:  HL points after RAM.
012                 *        BC preserved, ADEF corrupted.
013                 *
014 C7FB 110010     MEMCHK  LXI   D,:1000
015 C7FE 210000             LXI   H,:0000    Start at 0000
016 C801 19         MCK10   DAD   D          Incr. with #1000
017 C802 7E                 MOV   A,M        Get what is in memory
018 C803 2F                 CMA              Take its complement
019 C804 77                 MOV   M,A        and store it back
020 C805 BE                 CMP   M          Then compare
021 C806 2F                 CMA
022 C807 77                 MOV   M,A        Restore original value
023 C808 CA01C8             JZ    :C801      Next block if still RAM
024 C80B C9                 RET
025                 *
026                 ***********************
027                 * START FROM SCRATCH *
028                 ***********************
029                 *
030                 * Entry to Basic monitor.
031                 *
032
033                 * If out of a Hard BREAK:
034
035 C80C 3100F9     RSTART  LXI   SP,:F900   Reset stackpointer
036 C80F CDE4CE             CALL  :CEE4      Select ROM bank 0,
037 C812 B8DB               DBL   :DBB8      print '*** BREAK'
038
039                 * Re-enter Basic after run-time error,
040                 * except on input:
041
042 C814 AF           START   XRA   A
043 C815 323501             STA   :0135      Input from keyb/DINC
044
045                 * Entry on reset, after encoding program line,
046                 * after END:
047
048 C818 2100F9     ST10    LXI   H,:F900
049 C81B 222701             SHLD  :0127      Reset current base stack
050 C81E F9                 SPHL             Reset stackpointer
051 C81F AF                 XRA   A
052 C820 322601             STA   :0126      No suspended program
053
054                 * Restart interpreter; entry after end of program,
055                 * after direct command, after soft BREAK, after
056                 * direct command error, after STOP:
057
058 C823 2A2701     ST20    LHLD  :0127      Get saved stackpointer
059 C826 F9                 SPHL             Set it to saved value
060 C827 210000             LXI   H,:0000
061 C82A 220001             SHLD  :0100      Reset current line nr.
062 C82D 220401             SHLD  :0104      No running loops
063 C830 221301             SHLD  :0113      No active subroutine call
```



```
064 C833 7C                 MOV   A,H
065 C834 322201             STA   :0122      No encoding of stored line
066 C837 00                 NOP
067 C838 00                 NOP
068 C839 00                 NOP
069 C83A 211701             LXI   H,:0117
070 C83D 77                 MOV   M,A        No running of inputs
071 C83E 23                 INX   H
072 C83F 77                 MOV   M,A        No running of program
073 C840 CD88D9             CALL  :D988      Enable keyboard interrupt
074 C843 CDDBD9             CALL  :D9DB      Enable clock interrupt
075 C846 3A3501     ST23    LDA   :0135      Get input direction
076 C849 FE02               CPI   :02
077 C84B CC79D8             CZ    :D879      EFSW=2: input from editbuf
078 C84E D267C8             JNC   :C867      EFSW>=2: encode  ENCODE TEXTLINE IF EDITBUF NOT EMPTY
079 C851 3E2A               MVI   A,:2A
080 C853 CD1ADD             CALL  :DD1A      Print '*', scan keyboard (ERRATUM DANAMIC 83-17 P231)
081                                          and display characters
082                                          until Break or car.ret
083                                          (If no input is given,
084                                          the DAI remains here in a
085                                          endless loop).
086 C856 DA46C8             JC    :C846      If BREAK: new inputs
087 C859 CDD2DD             CALL  :DDD2      Get char from line, neglect
088                                          TAB and space
089 C85C FE0D               CPI   :0D
090 C85E CA46C8             JZ    :C846      If car.ret: new inputs
091 C861 CD0DDE             CALL  :DE0D      Check if char is number
092 C864 D26DC8             JNC   :C86D      If no leading nr: encode cmd
093
094                 * Encode program line (if 1st char is number):
095
096 C867 CD18C9     ST24    CALL  :C918      Encode program line, update
097                                          program
098 C86A C318C8             JMP   :C818      Get next input line; kill
099                                          any suspended program
100
101                 * Encode direct command (if 1st char is no number):
102
103 C86D 1680       ST25    MVI   D,:80      Mask for direct command
104 C86F E5                 PUSH  H          Pointer to RUNF
105 C870 213F01             LXI   H,:013F    Addr EBUF
106 C873 E5                 PUSH  H          Save it on stack
107 C874 CF                 RST   1          Encode immediate cmd line
108 C875 00                 DATA  :00
109 C876 3600               MVI   M,:00      Dummy end of program
110 C878 CD5EDD             CALL  :DD5E      Print car.ret
111 C87B C1                 POP   B          Get EBUF pntr
112 C87C E1                 POP   H          Pntr to RUNF
113 C87D 36FF               MVI   M,:FF      Set flag running programs
114
115                 * Run a Basic line:
116
117 C87F 0A         ST30    LDAX  B          Get 1st byte from EBUF:
118                                          <  #80: length,
119                                          >= #80: Token.
120 C880 03         ST35    INX   B
121 C881 87                 ADD   A          Calc offset from CF00
122 C882 D2E5C8             JNC   :C8E5      Jump if length byte
123 C885 6F                 MOV   L,A        Get table address in HL
124 C886 26CF               MVI   H,:CF
125 C888 7E                 MOV   A,M        ) Get addr Basic routine
```

```
126 C889 23              INX   H          ) from table in HL
127 C88A 66              MOV   H,M        )
128 C88B 6F              MOV   L,A        )
129 C88C CDA9C8          CALL  :C8A9      Perform this routine
130
131                 * Commands return here:
132
133 C88F DAAAC8     ENDCOM JC    :C8AA     Jump if special action
134
135                 * If suspended:
136
137 C892 60                MOV   H,B
138 C893 69                MOV   L,C
139 C894 220201            SHLD  :0102     Remember start next cmd
140 C897 3AC402            LDA   :02C4
141 C89A B7                ORA   A         BREAK flag set?
142 C89B CA7FC8            JZ    :C87F     Run Basic line if not
143 C89E 00                NOP
144 C89F 00                NOP
145 C8A0 00                NOP
146 C8A1 3EFF              MVI   A,:FF
147 C8A3 32C402            STA   :02C4     Set BREAK flag 'serviced'
148 C8A6 C3C0C8            JMP   :C8C0     Handle break
149
150                 * Run a BASIC line:
151
152 C8A9 E9         DCALL  PCHL             Addr Basic routine in PC
153
154                 * If special end of action:
155
156 C8AA FE02         ST40   CPI   :02
157 C8AC CAC0C8            JZ    :C8C0     If soft break (2)
158 C8AF D2B8C8            JNC   :C8B8     If STOP (3)
159 C8B2 EA18C8            JPE   :C818     If can't continu (1)
160 C8B5 C308C9            JMP   :C908     If after LOAD (0)
161
162                 * If 'STOP':
163
164 C8B8 60         ST45   MOV   H,B
165 C8B9 69                MOV   L,C
166 C8BA 220201            SHLD  :0102     Remember where next cmd
167 C8BD C3C5C8            JMP   :C8C5     Print 'IN LINE ..' and
168                                         handle a break
169
170                 * If suspended (soft Break handling):
171
172 C8C0 CDFFDA     ST50   CALL  :DAFF     Print car.ret; 'BREAK'.
173 C8C3 C5DB              DBL   :DBC5
174 C8C5 CD75DA     ST55   CALL  :DA75     Print 'IN LINE .....'
175                                         or car.ret
176 C8C8 CA23C8            JZ    :C823     Jump if immediate cmd
177
178                 * Only if 'break' in program:
179
180 C8CB 21EBFF              LXI   H,:FFEB   Frame length
181 C8CE 39                DAD   SP          New stack level
182 C8CF 44                MOV   B,H
183 C8D0 4D                MOV   C,L
184 C8D1 222701            SHLD  :0127     Set new base stack
185 C8D4 F9                SPHL            Set stackpointer
186 C8D5 110001            LXI   D,:0100   ) Boundaries frame
187 C8D8 211501            LXI   H,:0115   )
```



```
188 C8DB CD4FDE            CALL  :DE4F     Save program status
189                                        (FRAME) on stack.
190 C8DE 212601            LXI   H,:0126
191 C8E1 34                INR   M         Set flag existence saved
192                                        program
193 C8E2 C323C8            JMP   :C823     Run again
194
195                 * Length byte or end flag:
196
197 C8E5 CA23C8     ST60   JZ    :C823     If end immediate cmd
198                                        line or end program
199 C8E8 60                MOV   H,B
200 C8E9 69                MOV   L,C
201 C8EA 220001            SHLD  :0100     Store start current line
202 C8ED 2A1501            LHLD  :0115     Get trace + step flag
203 C8F0 7C                MOV   A,H
204 C8F1 B5                ORA   L
205 C8F2 CA00C9            JZ    :C900     If no step/trace flag
206
207                 * If step/trace flag set:
208
209 C8F5 E5                PUSH  H
210 C8F6 C5                PUSH  B
211 C8F7 CDA4CE            CALL  :CEA4     List current line
212 C8FA C1                POP   B
213 C8FB F1                POP   PSW       Get step flag
214 C8FC B7                ORA   A         If set:
215 C8FD C4DAD6            CNZ   :D6DA     Wait for spacebar pressed
216 C900 03         ST65   INX   B
217 C901 03                INX   B         Pnts after linenr
218 C902 DACBC8            JC    :C8CB     If Break
219 C905 C37FC8            JMP   :C87F     Run next BASIC line
220
221                 * Special action after LOAD:
222
223 C908 2A0001     ST70   LHLD  :0100     Get start current line
224 C90B 7C                MOV   A,H
225 C90C B5                ORA   L         Direct cmd?
226 C90D CA7FC8            JZ    :C87F     Then run Basic line
227 C910 3100F9            LXI   SP,:F900  Else: reset stackpointer
228 C913 3E87              MVI   A,:87     Simulate Token 'RUN'
229 C915 C380C8            JMP   :C880     Pretend RUN cmd
230                 *
231                 * PROGRAM INPUT:
232                 *
233                 * Encodes a program line and updates the stored
234                 * program.
235                 *
236                 * Entry: C: Input count / offset.
237                 * Exit:   C: Offset after line.
238                 *         AFBDEHL preserved.
239                 *
240 C918 213F01     PROGI  LXI   H,:013F   Addr buf for encoded cmds
241 C91B CF                RST   1         Get linenr
242 C91C 03                DATA  :03
243 C91D CDD2DD            CALL  :DDD2     Get char from line;
244                                        neglect TAB + space
245 C920 FE0D              CPI   :0D       Car.ret ?
246 C922 CCA2C9            CZ    :C9A2     Delete old version if
247                                        only linenr given
248 C925 CA3BC9            JZ    :C93B     Jump if linenr only
249 C928 D5                PUSH  D         Remember linenr
```

```
C929 1640          MVI   D,:40     Mask for 'stored cmd'
C92B CD3CC9        CALL  :C93C     Encode a line
C92E 7D            MOV   A,L
C92F D63F          SUI   :3F       Length string in A
C931 323E01        STA   :013E     Length in EBUF
C934 D1            POP   D         Get linenr
C935 CDA2C9        CALL  :C9A2     Delete old line
C938 CDBDC9        CALL  :C9BD     Insert new line
C93B C9      PGI20 RET
             *
             * ENCODE A LINE:
             *
             * Exit: DE restored.
             *       HL points to 1st free byte in EBUF.
             *        C points after car.ret in input.
             *        A=0, F corrupted.
             *
C93C D5      ELINA PUSH  D
C93D C5            PUSH  B
C93E E5            PUSH  H
C93F 210000        LXI   H,:0000
C942 39            DAD   SP        Stackpointer in HL
C943 221D01        SHLD  :011D     Save stackpointer
C946 E1            POP   H
C947 E5            PUSH  H
C948 3E01          MVI   A,:01
C94A 322201        STA   :0122     Set encoding a stored line
C94D CF            RST   1         Encode inputs
C94E 00            DATA  :00
C94F D1            POP   D         ) Cancel Push B,H
C950 D1            POP   D         )
C951 AF      ELA10 XRA   A
C952 322201        STA   :0122     No encoding stored line
C955 D1            POP   D
C956 C9            RET
             *
             ****************************************
             * ERROR WHILE ENCODING A STORED LINE *
             ****************************************
             *
             * Restores stackpointer, adds '***' to begin of
             * line, adds '?' to place of error. Line is entered
             * into the encoded inputbuffer (EBUF).
             *
             * Entry: B: Errorcode.
             *        C: Place of error.
             *        On stack: BC points to input.
             *                  HL points to EBUF.
             *
C957 2A1D01  ELARS LHLD  :011D     Get ERSSP
C95A F9            SPHL            Restore stackpointer
C95B E1            POP   H         Get buffer pointer
C95C 78            MOV   A,B       Errorcode in A
C95D 51            MOV   D,C       Place of error in D
C95E C1            POP   B         Get input pointer
C95F 47            MOV   B,A
C960 36B1          MVI   M,:B1     Token for '***' in EBUF
C962 23            INX   H
C963 E5            PUSH  H         Save buf pointer
C964 23            INX   H
C965 79      ELA20 MOV   A,C       ) Place of error
C966 BA            CMP   D         ) reached ?
```



```
C967 3E3F          MVI   A,:3F
C969 CC95C9        CZ    :C995     Then insert '?'
C96C CDE0DD        CALL  :DDE0     Get char. from line
C96F 0C            INR   C         Update inputpointer
C970 FE0D          CPI   :0D       Line done ?
C972 C495C9        CNZ   :C995     Insert char in EBUF if not
C975 C265C9        JNZ   :C965     Next char if not ready
C978 7D            MOV   A,L       Lobyte EBUF pntr in A
C979 D1            POP   D         Addr after '***'
C97A 93            SUB   E
C97B 3D            DCR   A
C97C 12            STAX  D         Store length in EBUF
C97D 3600          MVI   M,:00     0 after string
C97F C5            PUSH  B         Save errormessage pntr
C980 42            MOV   B,D       ) EBUF pntr in BC
C981 4B            MOV   C,E       )
C982 0B            DCX   B
C983 0B            DCX   B
C984 0B            DCX   B
C985 0B            DCX   B         Pnts to begin EBUF
C986 CD5EDD        CALL  :DD5E     Print car.ret
C989 CDABEC        CALL  :ECAB     (0) List current line
C98C C1            POP   B         Get errormessage pntr
C98D E5            PUSH  H
C98E CD50DA        CALL  :DA50     Print errormessage
C991 E1            POP   H
C992 C351C9        JMP   :C951     Store 0 in ERSFL, Pop D,
                                   and ret.
             *
             * INSERT CHARACTER IN ENCODED INPUT BUFFER:
             *
             * A character is inserted in the EBUF only if
             * there is space available.
             *
             * Entry: HL: 1st free location in EBUF.
             *        A:  Character to be inserted.
             * Exit:  HL updated. AFBCDE preserved.
             *
C995 F5      ELAIN PUSH  PSW
C996 7D            MOV   A,L       Get lobyte of EBUF pntr
C997 FEBC          CPI   :BC       Buffer full ?
C999 CAA0C9        JZ    :C9A0     Then abort
C99C F1            POP   PSW
C99D 77            MOV   M,A       Char into EBUF
C99E 23            INX   H         Update pntr
C99F C9            RET

             * If EBUF full:

C9A0 F1      EAI10 POP   PSW       No action
C9A1 C9            RET
             *
             ********************************
             * DELETE OLD VERSION OF A LINE *
             ********************************
             *
             * A textline is deleted by moving the rest of the
             * textbuffer and the symboltable 'downwards'.
             *
             * Entry: DE: requested linenumber.
             * Exit:  DE points to linenr after deleted line.
             *        AFBCHL preserved.
```

```
374                     *
375 C9A2 F5             LDEL    PUSH  PSW
376 C9A3 C5                     PUSH  B
377 C9A4 E5                     PUSH  H
378 C9A5 EB                     XCHG            Linenr in HL
379 C9A6 CDF6CA                 CALL  :CAF6     Addr line in textbuf in HL
380 C9A9 D2B8C9                 JNC   :C9B8     Abort if not found
381 C9AC 7E                     MOV   A,M       Get line length
382 C9AD 2F                     CMA
383 C9AE 5F                     MOV   E,A       Compl. value in E
384 C9AF 16FF                   MVI   D,:FF
385 C9B1 CD39DE                 CALL  :DE39     HL=HL-line length
386 C9B4 EB                     XCHG
387 C9B5 CDD1C9                 CALL  :C9D1     Move program buffers
388 C9B8 EB             LDL10   XCHG
389 C9B9 E1                     POP   H
390 C9BA C1                     POP   B
391 C9BB F1                     POP   PSW
392 C9BC C9                     RET
393                     *
394                     **********************
395                     * INSERT A NEW LINE *
396                     **********************
397                     *
398                     * Inserts a encoded line in the textbuffer.
399                     * Required space for the textline is made by
400                     * shifting the rest of the textbuffer and the
401                     * symboltable 'upwards'.
402                     *
403                     * Entry: DE: Destination address in textbuffer.
404                     *        HL: Points after string in EBUF.
405                     *        A:  Length string in EBUF.
406                     *
407 C9BD C5             LINS    PUSH  B
408 C9BE E5                     PUSH  H
409 C9BF 6F                     MOV   L,A
410 C9C0 2600                   MVI   H,:00     String length in HL
411 C9C2 23                     INX   H         Required space in HEAP
412 C9C3 D5                     PUSH  D
413 C9C4 CDD1C9                 CALL  :C9D1     Move program buffers
414 C9C7 C1                     POP   B
415 C9C8 E1                     POP   H
416 C9C9 113E01                 LXI   D,:013E   Startaddr. EBUF
417 C9CC CD4FDE                 CALL  :DE4F     Transfer data from EBUF
418                                             into textbuffer
419 C9CF C1                     POP   B
420 C9D0 C9                     RET
421                     *
422                     ************************
423                     * MOVE PROGRAM BUFFERS *
424                     ************************
425                     *
426                     * Moves a part (or the whole) textbuffer and the
427                     * whole symboltable up or down.
428                     * The startaddress of the textbuffer and the end
429                     * of the symboltable are set depending on the
430                     * heap size. The heap pointers are updated.
431                     *
432                     * Entry: DE: Address from where to update.
433                     *        HL: Length of area to be inserted/deleted.
434                     * Entry if running 'NEW' only:
435                     *        BC: 0.
```



```
436                     *        DE: startaddress Heap.
437                     *        HL: Size Heap.
438                     *        At entry, the pointers for textbuf and
439                     *        symtab are as if HEAPsize is zero.
440                     * Exit:  AFBCDE preserved.
441                     *        HL: New startaddress.
442                     *
443 C9D1 C5             PROGM   PUSH  B
444 C9D2 D5                     PUSH  D
445 C9D3 42                     MOV   B,D       ) Addr from where to
446 C9D4 4B                     MOV   C,E       ) update in BC
447 C9D5 EB                     XCHG            Length area in DE
448 C9D6 2AA302                 LHLD  :02A3     Get end symtab
449 C9D9 E5                     PUSH  H
450 C9DA D5                     PUSH  D
451 C9DB 19                     DAD   D         New end symtab
452 C9DC EB                     XCHG
453 C9DD 2AA502                 LHLD  :02A5     Get bottom screen RAM
454 C9E0 CD14DE                 CALL  :DE14     Check for overflow
455 C9E3 EB                     XCHG            End symtab in HL
456 C9E4 DA10DA                 JC    :DA10     Evt. run 'OUT OF MEMORY'
457 C9E7 22A302                 SHLD  :02A3     Store end symbol table
458 C9EA D1                     POP   D
459 C9EB D5                     PUSH  D
460 C9EC 2AA102                 LHLD  :02A1     Get begin symtab
461 C9EF 19                     DAD   D         New begin symtab
462 C9F0 22A102                 SHLD  :02A1     Store begin symbol table
463 C9F3 E1             PRGM1   POP   H
464 C9F4 50                     MOV   D,B
465 C9F5 59                     MOV   E,C       Startaddr in DE
466 C9F6 19                     DAD   D         New startaddr
467 C9F7 44                     MOV   B,H
468 C9F8 4D                     MOV   C,L       New startaddr in BC
469 C9F9 E3                     XTHL            Get old end symtab
470 C9FA CD4FDE                 CALL  :DE4F     Move textbuf + symtab
471                                             from old to new addr.
472 C9FD E1                     POP   H
473 C9FE D1                     POP   D
474 C9FF C1                     POP   B
475 CA00 C9                     RET
476                     *
477                     *
478                     *
479 CA01                        END
```

```
**************************
* S Y M B O L   T A B L E *
**************************

DCALL   C8A9    EAI10   C9A0    ELA10   C951    ELA20   C965
ELAIN   C995    ELARS   C957    ELINA   C93C    ENDCOM  C88F
LDEL    C9A2    LDL10   C9B8    LINS    C9BD    MCK10   C801
MEMCHK  C7FB    PGI20   C93B    PRGM1   C9F3    PROGI   C918
PROGM   C9D1    RSTART  C80C    ST10    C818    ST20    C823
ST23    C846    ST24    C867    ST25    C86D    ST30    C87F
ST35    C880    ST40    C8AA    ST45    C8B8    ST50    C8C0
ST55    C8C5    ST60    C8E5    ST65    C900    ST70    C908
START   C814
```

```
002                         ORG   :CA01
003                 *
004                 *
005                 *
006                 ****************************
007                 * MEMORY MANAGEMENT ROUTINE *
008                 ****************************
009                 *
010                 * This routine is used to obtain and release
011                 * memory for its display, as the mode changes.
012                 *
013                 * Entry: HL: Lowest screen RAM byte required.
014                 *        CY=1: Space to this point at least is
015                 *              now required. Additional space
016                 *              may not be released.
017                 *        CY=0: Space is held to or below this
018                 *              point. Any space held below this
019                 *              point is no longer required.
020                 * Exit:  CY=1: O.K.
021                 *        CY=0: No space available.
022                 *        AFBCDE preserved.
023                 *
024                 ASKRM
025 CA01 00         SMKRM   NOP
026 CA02 F5                 PUSH  PSW
027 CA03 D5                 PUSH  D
028 CA04 D21BCA             JNC   :CA1B     If space not reqd anymore
029 CA07 EB                 XCHG            Lowest byte in DE
030 CA08 2AA502             LHLD  :02A5     Get bottom screen RAM
031 CA0B CD14DE             CALL  :DE14     Check for overflow
032 CA0E DA1ECA             JC    :CA1E     If OK
033 CA11 2AA302             LHLD  :02A3     Get begin free RAM
034 CA14 EB                 XCHG            and store it in DE
035 CA15 CD14DE             CALL  :DE14     Still free RAM available?
036 CA18 DA21CA             JC    :CA21     If not
037 CA1B 22A502     ASR10   SHLD  :02A5     Update bottom screen RAM
038 CA1E C3ADCE     ASR20   JMP   :CEAD     Return, set CY=1
039
040                 * If no RAM space available:
041
042 CA21 D1         ASR30   POP   D
043 CA22 C3B1CE             JMP   :CEB1     Return, set CY=0
044                 *
045                 ******************************************
046                 * EMERGENCY STOP ROUTINE (Graphics modes) *
047                 ******************************************
048                 *
049                 * This routine is used if no sufficient space for
050                 * a A-mode is available.
051                 *
052 CA25 CDC6CE     EMSTP   CALL  :CEC6     Set up HEAPsize + buffers
053                                         to default values
054 CA28 3EFF             MVI   A,:FF
055 CA2A EF               RST   5           Change to mode 0
056 CA2B 18                 DATA  :18
057 CA2C CDE4CE             CALL  :CEE4     Select ROM-bank 0; Run error
058 CA2F 89DB               DBL   :DB89     'OUT OF SPACE FOR MODE'
059 CA31 C318C8             JMP   :C818     Return to BASIC monitor
060                 *
061                 ******************************************
062                 * FIND STRING BASIC INSTRUCTION IN TABLE *
063                 ******************************************
```



```
064                 *
065                 * Looks for table entry whose name is the initial
066                 * string of input, beginning at C'th position.
067                 * REMARK: Variables, beginning with a reserved
068                 *         string are not allowed.
069                 *
070                 * Entry: HL: Startaddress table.
071                 *        C : Position char on current line.
072                 *        E : Number of info bytes -1.
073                 * Exit:  If found: CY=1:
074                 *            HL: Address in table where string
075                 *                can be found.
076                 *            C : Position on current line after
077                 *                last char.
078                 *            A : Last byte of string typed in.
079                 *            BE preserved, D=0.
080                 *        If not found: CY=0:
081                 *            C : Points to next char.
082                 *            HL: Points after end of table.
083                 *            BE preserved. A,D=0.
084                 *
085 CA34 CDD2DD     LOOKC   CALL  :DDD2     Get char from line,
086                                         neglect TAB and space
087 CA37 56         LKC10   MOV   D,M       Get length byte of string
088 CA38 23                 INX   H         Points to 1st stringchar.
089 CA39 7A                 MOV   A,D
090 CA3A B7                 ORA   A         Is length zero?
091 CA3B C8                 RZ              Then abort
092 CA3C C5                 PUSH  B         Save position of 1st char
093 CA3D CDE0DD     LKC20   CALL  :DDE0     Get char from line
094 CA40 0C                 INR   C         Points to next char on line
095 CA41 BE                 CMP   M         Is it identical to the
096                                         one in table?
097 CA42 23                 INX   H         Points to next char in table
098 CA43 C24ECA             JNZ   :CA4E     If not identical
099 CA46 15                 DCR   D         Else: decr string length
100 CA47 C23DCA             JNZ   :CA3D     Get evt. next byte to check
101 CA4A E3                 XTHL            cancell PUSH B
102 CA4B E1                 POP   H
103 CA4C 37                 STC             CY=1
104 CA4D C9                 RET
105
106                 * If strings not identical:
107
108 CA4E 7A         LKC30   MOV   A,D       Get string length
109 CA4F 83                 ADD   E         Add 2
110 CA50 CD30DE             CALL  :DE30     Add A to HL; HL points now
111                                         to next string in table.
112 CA53 C1                 POP   B         Restore C=1
113 CA54 C337CA             JMP   :CA37     Start check on next string
114                 *
115                 ****************
116                 * TABLE LOOK UP *
117                 ****************
118                 *
119                 * Finds an entry in a look-up table.
120                 * LOOK used for symboltable, LOOKX for table of
121                 * Basic functions (CFE6).
122                 *
123                 * Table format:
124                 * [type/length][name][type/length][info]
125                 *
```

(ERRATUM DAINAMIC 83-17 P 231)

```
                        * Entry: B:  Points to start name in input.
                        *        D:  Type/length of name in input
                        *            high nibble: type; low nibble: length.
                        *        HL: Startaddress look-up table.
                        * Exit:   If not found: CY=0:
                        *            HL points to 0 byte at table end.
                        *            ABCD preserved, E corrupted.
                        *        If found: CY=1:
                        *            HL points to T/L of entry found.
                        *            E indicates how manyth entry.
                        *            ABC preserved.
                        *
CA57 2AA102             LOOK    LHLD  :02A1      Get startaddr symtab
                        *
CA5A 37                 LOOKX   STC             CY=1
CA5B F5                         PUSH  PSW
CA5C C5                         PUSH  B
CA5D 1EFF                       MVI   E,:FF
CA5F 1C                 LK10    INR   E         Entry count
CA60 7E                         MOV   A,M       Get T/L name in table
CA61 B7                         ORA   A
CA62 CA8BCA                     JZ    :CA8B     Abort if end table reached
CA65 BA                         CMP   D         Compare with wanted T/L
CA66 CA6FCA                     JZ    :CA6F     Jump if found
CA69 CDAECA             LK15    CALL  :CAAE     Calc addr next entry
CA6C C35FCA                     JMP   :CA5F     Check next entry

                        * If T/L of name O.K.:

CA6F C1                 LK20    POP   B
CA70 C5                         PUSH  B
CA71 48                         MOV   C,B
CA72 D5                         PUSH  D
CA73 7A                         MOV   A,D       Get wanted T/L of name
CA74 E60F                       ANI   :0F       Length name only
CA76 57                         MOV   D,A       in D
CA77 E5                         PUSH  H         Save begin table entry
CA78 23                 LK30    INX   H
CA79 CDE0DD                     CALL  :DDE0     Get char from line
CA7C BE                         CMP   M         Compare char of name
CA7D C28FCA                     JNZ   :CA8F     If not correct name

                        * If char. identical:

CA80 0C                         INR   C         Points to next char
CA81 15                         DCR   D         Decr length
CA82 C278CA                     JNZ   :CA78     Check next char if not
                                                ready
CA85 23                         INX   H         Points after name in table
CA86 D1                         POP   D
CA87 D1                         POP   D
CA88 C1                         POP   B
CA89 F1                         POP   PSW       CY=1: Entry found
CA8A C9                         RET

                        * If end of look-up table reached:

CA8B C1                 LK40    POP   B
CA8C F1                         POP   PSW
CA8D 3F                         CMC             CY=0: No entry found
CA8E C9                         RET
```



```
                        * If characters not identical:

CA8F E1                 LK50    POP   H
CA90 D1                         POP   D
CA91 7A                         MOV   A,D       Length in A
CA92 C369CA                     JMP   :CA69     Look further
                        *
                        ************************************
                        * FIND A VARIABLE IN THE SYMBOLTABLE *
                        ************************************
                        *
                        * Routine skips through successive symtab entries
                        * from beginning till past the place pointed by HL.
                        *
                        * Entry: HL points to 1st byte required variable.
                        * Exit:  HL points to (if found) or past (if not
                        *        found) address required in symbol table.
                        *        AFBCDE preserved.
                        *
CA95 F5                 FNAME   PUSH  PSW
CA96 D5                         PUSH  D
CA97 EB                         XCHG            Reqd addr in DE
CA98 2AA102                     LHLD  :02A1     Get startaddr symtab
CA9B E5                 FNM10   PUSH  H
CA9C CDAECA                     CALL  :CAAE     Calc addr next variable
CA9F CD14DE                     CALL  :DE14     Reqd variable reached ?
CAA2 D2AACA                     JNC   :CAAA     Quit if true
CAA5 33                         INX   SP        ) Cancel Push H
CAA6 33                         INX   SP        )
CAA7 C39BCA                     JMP   :CA9B     Skip next variable
CAAA E1                 FNM20   POP   H
CAAB D1                         POP   D
CAAC F1                         POP   PSW
CAAD C9                         RET
                        *
                        ************************************************
                        * CALCULATE ADDRESS NEXT VARIABLE IN SYMBOLTABLE *
                        ************************************************
                        *
                        * Adds length of name of variable + length of value
                        * of variable to beginaddress.
                        *
                        * DADD: variable = length/name/length/value.
                        * DADR: variable = length/name.
                        *
                        * Entry: HL points to 1st byte of current variable.
                        * Exit:  HL points to next variable in symtab.
                        *
CAAE CDB1CA             DADD    CALL  :CAB1     Add length name to HL
CAB1 7E                 DADR    MOV   A,M       Get info T/L byte
CAB2 23                         INX   H         Add 1
CAB3 E60F                       ANI   :0F       Length only
CAB5 C330DE                     JMP   :DE30     Add length info to HL
                        *
                        ********************************************
                        * INSERT A VARIABLE NAME IN THE SYMBOL TABLE *
                        ********************************************
                        *
                        * Entry: HL points to end symtab.
                        *        B  points to start of name in input.
                        *        E  number of bytes of info to reserve.
                        *        D  T/L byte of name.
```

```
                        * Exit:  HL points to info T/L byte.
                        *        AFBCDE preserved.
                        *
CAB8 F5         LOOKI   PUSH  PSW
CAB9 C5                 PUSH  B
CABA 48                 MOV   C,B       Input pos. in C
CABB D5                 PUSH  D
CABC D5                 PUSH  D
CABD E5                 PUSH  H
CABE 7A                 MOV   A,D       T/L name in A
CABF E60F               ANI   :0F       Name length only
CAC1 83                 ADD   E         Add length info
CAC2 3C                 INR   A
CAC3 3C                 INR   A         +2 (length new entry)
CAC4 CD30DE             CALL  :DE30     Calc new end symtab -1
CAC7 EB                 XCHG
CAC8 2AA502             LHLD  :02A5     Get bottom screen RAM
CACB EB                 XCHG
CACC CD14DE             CALL  :DE14     Compare DE-HL
CACF 3E1B               MVI   A,:1B
CAD1 D2F5D9             JNC   :D9F5     Run 'OUT OF MEMORY' if
                                        not sufficient free RAM
CAD4 3600               MVI   M,:00     New 'end table' flag
CAD6 23                 INX   H         HL is new end symtab
CAD7 22A302             SHLD  :02A3     Store end symtab
CADA E1                 POP   H         Get old end symtab
CADB D1                 POP   D         Get T/L info
CADC 72                 MOV   M,D       Into symtab
CADD 23                 INX   H
CADE 7A                 MOV   A,D
CADF E670               ANI   :70       Get type only
CAE1 B3                 ORA   E         Set low nibble for length
CAE2 5F                 MOV   E,A
CAE3 7A                 MOV   A,D       Get length name
CAE4 E60F               ANI   :0F       Max. 15 bytes
CAE6 57                 MOV   D,A       Length in D for count
CAE7 CDE0DD     LKI10   CALL  :DDE0     Get char from line
CAEA 77                 MOV   M,A       Char into symtab
CAEB 0C                 INR   C         Pnts to next char on line
CAEC 23                 INX   H         Next pos in symtab
CAED 15                 DCR   D         Decr length name
CAEE C2E7CA             JNZ   :CAE7     Next char if not ready
CAF1 73                 MOV   M,E       Info T/L into symtab
CAF2 D1                 POP   D
CAF3 C1                 POP   B
CAF4 F1                 POP   PSW
CAF5 C9                 RET
                        *
                        ********************************
                        * FIND LINENUMBER IN TEXTBUFFER *
                        ********************************
                        *
                        * Entry: HL: requested linenumber.
                        * Exit:  ABCDE preserved. F corrupted.
                        *        CY=1: Linenr found:
                        *              HL points to address textline.
                        *        CY=0: Not found:
                        *              Z=0: HL points to address textline
                        *                   with next higher linenumber.
                        *              Z=1: End of textbuffer reached.
                        *
CAF6 C5         FINDL   PUSH  B
```



```
CAF7 F5                 PUSH  PSW
CAF8 D5                 PUSH  D
CAF9 0600               MVI   B,:00
CAFB EB                 XCHG            Req. linenr in DE
CAFC 2A9F02             LHLD  :029F     Get startaddr textbuf
CAFF 48         FDL05   MOV   C,B       Length prev. instr in C
CB00 0600               MVI   B,:00
CB02 09                 DAD   B         Add this length to beginaddr
CB03 46         FDL10   MOV   B,M       Get length current instr
CB04 78                 MOV   A,B       in A
CB05 B7                 ORA   A
CB06 23                 INX   H         Pnts to hibyte linenr
CB07 CA1DCB             JZ    :CB1D     Abort if at end textbuf
CB0A 7A                 MOV   A,D       Get hibyte reqd linenr.
CB0B BE                 CMP   M         Test high order bits
CB0C DA1CCB             JC    :CB1C     Abort if reqd nr lower than
                                        current one (nr > reqd)
CB0F C2FFCA             JNZ   :CAFF     Next textline (nr < reqd)
CB12 23                 INX   H         Pnts to lobyte linenr
CB13 7B                 MOV   A,E       Get lobyte reqd linenr
CB14 BE                 CMP   M
CB15 2B                 DCX   H
CB16 DA1CCB             JC    :CB1C     Abort if reqd nr lower than
                                        current one (nr > reqd)
CB19 C2FFCA             JNZ   :CAFF     Next textline if not found
                                        (nr < reqd)
CB1C 3F         FDL20   CMC             Line found: CY=1
CB1D 2B         FDL30   DCX   H
CB1E D1                 POP   D
CB1F C1                 POP   B
CB20 78                 MOV   A,B
CB21 C1                 POP   B
CB22 C9                 RET
                        *
                        ********************************
                        * EMPTY SYMBOLTABLE AND HEAP *
                        ********************************
                        *
                        * Zeroes all variables, all pointers in the symtab,
                        * kill all arrays, strings or stringarrays (zero the
                        * pointers) referenced by the symtab by setting the
                        * msb of the sizebit =1. Basic program is moved to a
                        * location corresponding to the Heapsize.
                        *
                        * Exit: All registers preserved.
                        *
CB23 F5         SCRATC  PUSH  PSW
CB24 C5                 PUSH  B
CB25 D5                 PUSH  D
CB26 E5                 PUSH  H
CB27 2AA102             LHLD  :02A1     Get startaddr symtab
CB2A 7E         SCT10   MOV   A,M       ) get name length
CB2B E60F               ANI   :0F       )
CB2D CA53CB             JZ    :CB53     Abort if at end symtab
CB30 CDB1CA             CALL  :CAB1     HL pnts to info length byte
CB33 7E                 MOV   A,M       ) Get type
CB34 E6F0               ANI   :F0       )
CB36 FE40               CPI   :40
CB38 D24DCB             JNC   :CB4D     If array
CB3B 23                 INX   H
CB3C FE20               CPI   :20
CB3E CA47CB             JZ    :CB47     If string
```

```
                        * If numeric variable:
                        
CB41 CD9ECB                     CALL    :CB9E       Set value is 0 (4 bytes)
CB44 C32ACB                     JMP     :CB2A       Next entry
                        
                        * If string variable:
                        
CB47 CDA8CB     SCT30   CALL    :CBA8       Erase string reference in
                                                    symbtab and Heap
CB4A C32ACB             JMP     :CB2A       Next entry
                        
                        * If array:
                        
CB4D CD5BCB     SCT40   CALL    :CB5B       Erase array
CB50 C32ACB             JMP     :CB2A       Next entry
                        
                        * If ready:
                        
CB53 CDCADE     SCT90   CALL    :DECA       Organise HEAP + buffers
CB56 E1         LC188   POP     H
CB57 D1                 POP     D
CB58 C1                 POP     B
CB59 F1                 POP     PSW
CB5A C9                 RET
                        *
                        ***************
                        * ERASE ARRAY *
                        ***************
                        *
                        * The pointer is zeroed, the array size is erased
                        * (msb=1). For stringarrays: zeroes all pointers
                        * in the array and erase the string in the Heap.
                        *
                        * Entry: HL points to T/L byte of info after a
                        *          symtab name of an array.
                        * Exit:  HL points to next symtab entry.
                        *        AFBCDE preserved.
                        *
CB5B D5         EARRAY  PUSH    D
CB5C C5                 PUSH    B
CB5D F5                 PUSH    PSW
CB5E 7E                 MOV     A,M         Get type info
CB5F E630               ANI     :30
CB61 F5                 PUSH    PSW         Save type only
CB62 CD51CE             CALL    :CE51       ) Get addr of array in
CB65 23                 INX     H           ) Heap in DE,
CB66 56                 MOV     D,M         ) Kill pointer in the
CB67 3600               MVI     M,:00       ) symboltable
CB69 23                 INX     H           Pnts after symtab entry
CB6A 7A                 MOV     A,D
CB6B B3                 ORA     E
CB6C CA99CB             JZ      :CB99       Abort if entry was already 0
CB6F EB                 XCHG                Arrayaddr in HL
CB70 2B                 DCX     H
CB71 2B                 DCX     H           Pnts to 1st byte Heap entry
CB72 46                 MOV     B,M         Get 1st byte
CB73 CD36D2             CALL    :D236       Clear heap entry by msb=1
CB76 F1                 POP     PSW         Get type info
CB77 F5                 PUSH    PSW
CB78 FE20               CPI     :20         String ?
CB7A D5                 PUSH    D           Save stringpointer
```



```
CB7B C298CB                     JNZ     :CB98       Abort if not string
                        
                        * If string array:
                        
CB7E 23                 EAR10   INX     H
CB7F 4E                         MOV     C,M         Get length Heap entry
CB80 23                         INX     H
CB81 5E                         MOV     E,M         Get dimension
CB82 1C                         INR     E
CB83 7B                         MOV     A,E
CB84 CD30DE                     CALL    :DE30       Calc beginaddr stringpntrs
CB87 79                         MOV     A,C
CB88 93                         SUB     E           Calc length pntr area
CB89 4F                         MOV     C,A         in C
CB8A D28ECB                     JNC     :CB8E
CB8D 05                         DCR     B
CB8E CDA8CB             EAR10   CALL    :CBA8       Erase stringreference in
                                                    symtab and Heap
CB91 0B                         DCX     B
CB92 0B                         DCX     B           Update length pntr area
CB93 78                         MOV     A,B
CB94 B1                         ORA     C
CB95 C28ECB                     JNZ     :CB8E       Next string if not ready
                        
                        * If ready:
                        
CB98 E1                 EAR20   POP     H           Get symtab pntr
CB99 F1                 EAR30   POP     PSW
CB9A F1                         POP     PSW
CB9B C1                         POP     B
CB9C D1                         POP     D
CB9D C9                         RET
                        *
                        ************************************************
                        * CLEAR A NUMERIC VARIABLE IN THE SYMBOLTABLE *
                        ************************************************
                        *
                        * Loads '0' into 4 consecutive memory locations.
                        *
                        * Entry: Startaddress in HL.
                        * Exit:  A=0, HL points to next byte.
                        *        BCDEF preserved.
                        *
CB9E AF                 ZFPINT  XRA     A
CB9F 77                         MOV     M,A
CBA0 23                         INX     H
CBA1 77                         MOV     M,A
CBA2 23                         INX     H
CBA3 77                         MOV     M,A
CBA4 23                         INX     H
CBA5 77                         MOV     M,A
CBA6 23                         INX     H
CBA7 C9                         RET
                        *
                        ******************************************
                        * ERASE STRINGREFERENCE IN HEAP AND SYMTAB *
                        ******************************************
                        *
                        * The pointer in the symtab is set to '0', the
                        * msb of the sizebyte of the Heap entry is set
                        * to 1.
                        *
```

```
                    * Entry: HL points to stringpointer in symtab.
                    * Exit:  HL points after this pointer.
                    *        DE stringpointer.
                    *        BC preserved, AF corrupted
                    *
CBA8 5E             RSVHL   MOV   E,M       ) Stringpointer
CBA9 3600                   MVI   M,:00     ) in DE and then
CBAB 23                     INX   H         ) erased.
CBAC 56                     MOV   D,M       )
CBAD 3600                   MVI   M,:00     )
CBAF 23                     INX   H
CBB0 E5                     PUSH  H
CBB1 2A9F02                 LHLD  :029F     Get startaddr textbuf
CBB4 EB                     XCHG            in DE; stringpntr in HL
CBB5 7C                     MOV   A,H
CBB6 B5                     ORA   L         Stringpntr is already 0 ?
CBB7 C414DE                 CNZ   :DE14     If not: test if end of
                                            heap reached
CBBA DC87D1                 CC    :D187     If not: clear heap entry
CBBD E1                     POP   H
CBBE C9                     RET
                    *
                    *
                    *
CBBF                        END

**************************
* S Y M B O L   T A B L E *
**************************

ASKRM  CA01   ASR10  CA1B   ASR20  CA1E   ASR30  CA21
DADD   CAAE   DADR   CAB1   EAR10  CB8E   EAR20  CB98
EAR30  CB99   EARRAY CB5B   EMSTP  CA25   FDL05  CAFF
FDL10  CB03   FDL20  CB1C   FDL30  CB1D   FINDL  CAF6
FNAME  CA95   FNM10  CA9B   FNM20  CAAA   LC188  CB56
LK10   CA5F   LK15   CA69   LK20   CA6F   LK30   CA78
LK40   CA8B   LK50   CA8F   LKC10  CA37   LKC20  CA3D
LKC30  CA4E   LKI10  CAE7   LOOK   CA57   LOOKC  CA34
LOOKI  CAB8   LOOKX  CA5A   RSVHL  CBA8   SCRATC CB23
SCT10  CB2A   SCT30  CB47   SCT40  CB4D   SCT90  CB53
SMKRM  CA01   ZFPINT CB9E
```



```
                            ORG   :CBBF
                    *
                    *
                    *
                    **************************
                    * STRINGS BASIC COMMANDS *
                    **************************
                    *
                    * The first byte of each string is a length byte.
                    *
                    * The first byte after the string is the 'type'
                    * byte. It is used to compose the TOKEN of the
                    * particular Basic command:
                    *    type byte, ANI :3F, ORI :80, gives TOKEN.
                    *
                    * Commands with type bytes bit 7=1 can be executed
                    * during a program run. If bit 6=1, commands are
                    * valid as direct command.
                    *
                    * The address given is the location of the encoding
                    * routine for this particular command. These
                    * routines can be found in ROM bank 3.
                    *
                    CMDTB
CBBF 03             SNEW    DATA  :03
CBC0 4E                     DATA  :4E       N
CBC1 45                     DATA  :45       E
CBC2 57                     DATA  :57       W
CBC3 81                     DATA  :81
CBC4 69E3                   DBL   :E369
                    *
CBC6 04             SCONT   DATA  :04
CBC7 43                     DATA  :43       C
CBC8 4F                     DATA  :4F       O
CBC9 4E                     DATA  :4E       N
CBCA 54                     DATA  :54       T
CBCB 82                     DATA  :82
CBCC 69E3                   DBL   :E369
                    *
CBCE 04             SSTOP   DATA  :04
CBCF 53                     DATA  :53       S
CBD0 54                     DATA  :54       T
CBD1 4F                     DATA  :4F       O
CBD2 50                     DATA  :50       P
CBD3 43                     DATA  :43
CBD4 69E3                   DBL   :E369
                    *
CBD6 03             SEND    DATA  :03
CBD7 45                     DATA  :45       E
CBD8 4E                     DATA  :4E       N
CBD9 44                     DATA  :44       D
CBDA 44                     DATA  :44
CBDB 69E3                   DBL   :E369
                    *
CBDD 07             SREST   DATA  :07
CBDE 52                     DATA  :52       R
CBDF 45                     DATA  :45       E
CBE0 53                     DATA  :53       S
CBE1 54                     DATA  :54       T
CBE2 4F                     DATA  :4F       O
CBE3 52                     DATA  :52       R
CBE4 45                     DATA  :45       E
```

```
064 CBE5 C5              DATA  :C5
065 CBE6 69E3            DBL   :E369
066              *
067 CBE8 06      SRET    DATA  :06
068 CBE9 52              DATA  :52      R
069 CBEA 45              DATA  :45      E
070 CBEB 54              DATA  :54      T
071 CBEC 55              DATA  :55      U
072 CBED 52              DATA  :52      R
073 CBEE 4E              DATA  :4E      N
074 CBEF 46              DATA  :46
075 CBF0 69E3            DBL   :E369
076              *
077 CBF2 03      SRUN    DATA  :03
078 CBF3 52              DATA  :52      R
079 CBF4 55              DATA  :55      U
080 CBF5 4E              DATA  :4E      N
081 CBF6 87              DATA  :87
082 CBF7 95E2            DBL   :E295
083              *
084 CBF9 04      SGOTO   DATA  :04
085 CBFA 47              DATA  :47      G
086 CBFB 4F              DATA  :4F      O
087 CBFC 54              DATA  :54      T
088 CBFD 4F              DATA  :4F      O
089 CBFE 49              DATA  :49
090 CBFF 6AE3            DBL   :E36A
091              *
092 CC01 05      SGOSUB  DATA  :05
093 CC02 47              DATA  :47      G
094 CC03 4F              DATA  :4F      O
095 CC04 53              DATA  :53      S
096 CC05 55              DATA  :55      U
097 CC06 42              DATA  :42      B
098 CC07 4A              DATA  :4A
099 CC08 6AE3            DBL   :E36A
100              *
101 CC0A 03      SIMP    DATA  :03
102 CC0B 49              DATA  :49      I
103 CC0C 4D              DATA  :4D      M
104 CC0D 50              DATA  :50      P
105 CC0E B5              DATA  :B5
106 CC0F 9FE2            DBL   :E29F
107              *
108 CC11 05      SSAVA   DATA  :05
109 CC12 53              DATA  :53      S
110 CC13 41              DATA  :41      A
111 CC14 56              DATA  :56      V
112 CC15 45              DATA  :45      E
113 CC16 41              DATA  :41      A
114 CC17 F9              DATA  :F9
115 CC18 A9E4            DBL   :E4A9
116              *
117 CC1A 05      SLODA   DATA  :05
118 CC1B 4C              DATA  :4C      L
119 CC1C 4F              DATA  :4F      O
120 CC1D 41              DATA  :41      A
121 CC1E 44              DATA  :44      D
122 CC1F 41              DATA  :41      A
123 CC20 FA              DATA  :FA
124 CC21 A9E4            DBL   :E4A9
125              *
```



```
126 CC23 03      SOUT    DATA  :03
127 CC24 4F              DATA  :4F      O
128 CC25 55              DATA  :55      U
129 CC26 54              DATA  :54      T
130 CC27 CE              DATA  :CE
131 CC28 02E3            DBL   :E302
132              *
133 CC2A 04      SPOKE   DATA  :04
134 CC2B 50              DATA  :50      P
135 CC2C 4F              DATA  :4F      O
136 CC2D 4B              DATA  :4B      K
137 CC2E 45              DATA  :45      E
138 CC2F CF              DATA  :CF
139 CC30 02E3            DBL   :E302
140              *
141 CC32 04      SWAIT   DATA  :04
142 CC33 57              DATA  :57      W
143 CC34 41              DATA  :41      A
144 CC35 49              DATA  :49      I
145 CC36 54              DATA  :54      T
146 CC37 D0              DATA  :D0
147 CC38 59E2            DBL   :E259
148              *
149 CC3A 04      SLIST   DATA  :04
150 CC3B 4C              DATA  :4C      L
151 CC3C 49              DATA  :49      I
152 CC3D 53              DATA  :53      S
153 CC3E 54              DATA  :54      T
154 CC3F D3              DATA  :D3
155 CC40 28E2            DBL   :E228
156              *
157 CC42 04      SEDIT   DATA  :04
158 CC43 45              DATA  :45      E
159 CC44 44              DATA  :44      D
160 CC45 49              DATA  :49      I
161 CC46 54              DATA  :54      T
162 CC47 B6              DATA  :B6
163 CC48 28E2            DBL   :E228
164              *
165 CC4A 05      SSOUND  DATA  :05
166 CC4B 53              DATA  :53      S
167 CC4C 4F              DATA  :4F      O
168 CC4D 55              DATA  :55      U
169 CC4E 4E              DATA  :4E      N
170 CC4F 44              DATA  :44      D
171 CC50 D6              DATA  :D6
172 CC51 17E3            DBL   :E317
173              *
174 CC53 05      SNOISE  DATA  :05
175 CC54 4E              DATA  :4E      N
176 CC55 4F              DATA  :4F      O
177 CC56 49              DATA  :49      I
178 CC57 53              DATA  :53      S
179 CC58 45              DATA  :45      E
180 CC59 D7              DATA  :D7
181 CC5A 25E3            DBL   :E325
182              *
183 CC5C 08      SENV    DATA  :08
184 CC5D 45              DATA  :45      E
185 CC5E 4E              DATA  :4E      N
186 CC5F 56              DATA  :56      V
187 CC60 45              DATA  :45      E
```

```
188 CC61 4C                DATA  :4C       L
189 CC62 4F                DATA  :4F       O
190 CC63 50                DATA  :50       P
191 CC64 45                DATA  :45       E
192 CC65 D8                DATA  :D8
193 CC66 F6E1              DBL   :E1F6
194                *
195 CC68 06        SCURS   DATA  :06
196 CC69 43                DATA  :43       C
197 CC6A 55                DATA  :55       U
198 CC6B 52                DATA  :52       R
199 CC6C 53                DATA  :53       S
200 CC6D 4F                DATA  :4F       O
201 CC6E 52                DATA  :52       R
202 CC6F D9                DATA  :D9
203 CC70 02E3              DBL   :E302
204                *
205 CC72 04        SMODE   DATA  :04
206 CC73 4D                DATA  :4D       M
207 CC74 4F                DATA  :4F       O
208 CC75 44                DATA  :44       D
209 CC76 45                DATA  :45       E
210 CC77 DA                DATA  :DA
211 CC78 D1E1              DBL   :E1D1
212                *
213 CC7A 03        SDOT    DATA  :03
214 CC7B 44                DATA  :44       D
215 CC7C 4F                DATA  :4F       O
216 CC7D 54                DATA  :54       T
217 CC7E DB                DATA  :DB
218 CC7F 8FE2              DBL   :E28F
219                *
220 CC81 04        SDRAW   DATA  :04
221 CC82 44                DATA  :44       D
222 CC83 52                DATA  :52       R
223 CC84 41                DATA  :41       A
224 CC85 57                DATA  :57       W
225 CC86 DC                DATA  :DC
226 CC87 8CE2              DBL   :E28C
227                *
228 CC89 04        SFILL   DATA  :04
229 CC8A 46                DATA  :46       F
230 CC8B 49                DATA  :49       I
231 CC8C 4C                DATA  :4C       L
232 CC8D 4C                DATA  :4C       L
233 CC8E DD                DATA  :DD
234 CC8F 8CE2              DBL   :E28C
235                *
236 CC91 06        SCOLT   DATA  :06
237 CC92 43                DATA  :43       C
238 CC93 4F                DATA  :4F       O
239 CC94 4C                DATA  :4C       L
240 CC95 4F                DATA  :4F       O
241 CC96 52                DATA  :52       R
242 CC97 54                DATA  :54       T
243 CC98 DE                DATA  :DE
244 CC99 0BE3              DBL   :E30B
245                *
246 CC9B 06        SCOLG   DATA  :06
247 CC9C 43                DATA  :43       C
248 CC9D 4F                DATA  :4F       O
249 CC9E 4C                DATA  :4C       L
```



```
250 CC9F 4F                DATA  :4F       O
251 CCA0 52                DATA  :52       R
252 CCA1 47                DATA  :47       G
253 CCA2 DF                DATA  :DF
254 CCA3 0BE3              DBL   :E30B
255                *
256 CCA5 05        SINPUT  DATA  :05
257 CCA6 49                DATA  :49       I
258 CCA7 4E                DATA  :4E       N
259 CCA8 50                DATA  :50       P
260 CCA9 55                DATA  :55       U
261 CCAA 54                DATA  :54       T
262 CCAB 60                DATA  :60
263 CCAC 15E1              DBL   :E115
264                *
265 CCAE 04        SDATA   DATA  :04
266 CCAF 44                DATA  :44       D
267 CCB0 41                DATA  :41       A
268 CCB1 54                DATA  :54       T
269 CCB2 41                DATA  :41       A
270 CCB3 62                DATA  :62
271 CCB4 A2E8              DBL   :E8A2
272                *
273 CCB6 04        SREAD   DATA  :04
274 CCB7 52                DATA  :52       R
275 CCB8 45                DATA  :45       E
276 CCB9 41                DATA  :41       A
277 CCBA 44                DATA  :44       D
278 CCBB 63                DATA  :63
279 CCBC 27E1              DBL   :E127
280                *
281 CCBE 03        SLET    DATA  :03
282 CCBF 4C                DATA  :4C       L
283 CCC0 45                DATA  :45       E
284 CCC1 54                DATA  :54       T
285 CCC2 E4                DATA  :E4
286 CCC3 FEE0              DBL   :E0FE
287                *
288 CCC5 02        SIF     DATA  :02
289 CCC6 49                DATA  :49       I
290 CCC7 46                DATA  :46       F
291 CCC8 66                DATA  :66
292 CCC9 BCE0              DBL   :E0BC
293                *
294 CCCB 03        SREM    DATA  :03
295 CCCC 52                DATA  :52       R
296 CCCD 45                DATA  :45       E
297 CCCE 4D                DATA  :4D       M
298 CCCF 69                DATA  :69
299 CCD0 66E3              DBL   :E366
300                *
301 CCD2 03        SFOR    DATA  :03
302 CCD3 46                DATA  :46       F
303 CCD4 4F                DATA  :4F       O
304 CCD5 52                DATA  :52       R
305 CCD6 EA                DATA  :EA
306 CCD7 5FE0              DBL   :E05F
307                *
308 CCD9 04        SNEXT   DATA  :04
309 CCDA 4E                DATA  :4E       N
310 CCDB 45                DATA  :45       E
311 CCDC 58                DATA  :58       X
```

```
312 CCDD 54                DATA  :54       T
313 CCDE EB                DATA  :EB
314 CCDF A9E0              DBL   :E0A9
315                *
316 CCE1 05        SPRINT  DATA  :05
317 CCE2 50                DATA  :50       P
318 CCE3 52                DATA  :52       R
319 CCE4 49                DATA  :49       I
320 CCE5 4E                DATA  :4E       N
321 CCE6 54                DATA  :54       T
322 CCE7 ED                DATA  :ED
323 CCE8 9FE1              DBL   :E19F
324                *
325 CCEA 01        S?      DATA  :01
326 CCEB 3F                DATA  :3F       ?   (abbr. for PRINT)
327 CCEC ED                DATA  :ED
328 CCED 9FE1              DBL   :E19F
329                *
330 CCEF 02        SON     DATA  :02
331 CCF0 4F                DATA  :4F       O
332 CCF1 4E                DATA  :4E       N
333 CCF2 6E                DATA  :6E
334 CCF3 76E1              DBL   :E176
335                *
336 CCF5 03        SDIM    DATA  :03
337 CCF6 44                DATA  :44       D
338 CCF7 49                DATA  :49       I
339 CCF8 4D                DATA  :4D       M
340 CCF9 F0                DATA  :F0
341 CCFA 66E1              DBL   :E166
342                *
343 CCFC 03                DATA  :03
344 CCFD 2A                DATA  :2A       *
345 CCFE 2A                DATA  :2A       *
346 CCFF 2A                DATA  :2A       *
347 CD00 71                DATA  :71
348 CD01 66E3              DBL   :E366
349                *
350 CD03 02        SUT     DATA  :02
351 CD04 55                DATA  :55       U
352 CD05 54                DATA  :54       T
353 CD06 B2                DATA  :B2
354 CD07 69E3              DBL   :E369
355                *
356 CD09 05        SCALM   DATA  :05
357 CD0A 43                DATA  :43       C
358 CD0B 41                DATA  :41       A
359 CD0C 4C                DATA  :4C       L
360 CD0D 4C                DATA  :4C       L
361 CD0E 4D                DATA  :4D       M
362 CD0F F3                DATA  :F3
363 CD10 44E3              DBL   :E344
364                *
365 CD12 05        SCLEAR  DATA  :05
366 CD13 43                DATA  :43       C
367 CD14 4C                DATA  :4C       L
368 CD15 45                DATA  :45       E
369 CD16 41                DATA  :41       A
370 CD17 52                DATA  :52       R
371 CD18 F4                DATA  :F4
372 CD19 14E3              DBL   :E314
373                *
```



```
374 CD1B 04        SLOAD   DATA  :04
375 CD1C 4C                DATA  :4C       L
376 CD1D 4F                DATA  :4F       O
377 CD1E 41                DATA  :41       A
378 CD1F 44                DATA  :44       D
379 CD20 CB                DATA  :CB
380 CD21 55E3              DBL   :E355
381                *
382 CD23 04        SSAVE   DATA  :04
383 CD24 53                DATA  :53       S
384 CD25 41                DATA  :41       A
385 CD26 56                DATA  :56       V
386 CD27 45                DATA  :45       E
387 CD28 8C                DATA  :8C
388 CD29 55E3              DBL   :E355
389                *
390 CD2B 05        SCHECK  DATA  :05
391 CD2C 43                DATA  :43       C
392 CD2D 48                DATA  :48       H
393 CD2E 45                DATA  :45       E
394 CD2F 43                DATA  :43       C
395 CD30 4B                DATA  :4B       K
396 CD31 8D                DATA  :8D
397 CD32 69E3              DBL   :E369
398                *
399 CD34 05                DATA  :05       (Cancelled instr.)
400 CD35 00                DATA  :00
401 CD36 52                DATA  :52       R
402 CD37 41                DATA  :41       A
403 CD38 53                DATA  :53       S
404 CD39 45                DATA  :45       E
405 CD3A FB                DATA  :FB
406 CD3B 73E8              DBL   :E873
407                *
408 CD3D 04        SSTEP   DATA  :04
409 CD3E 53                DATA  :53       S
410 CD3F 54                DATA  :54       T
411 CD40 45                DATA  :45       E
412 CD41 50                DATA  :50       P
413 CD42 BC                DATA  :BC
414 CD43 69E3              DBL   :E369
415                *
416 CD45 04        STRON   DATA  :04
417 CD46 54                DATA  :54       T
418 CD47 52                DATA  :52       R
419 CD48 4F                DATA  :4F       O
420 CD49 4E                DATA  :4E       N
421 CD4A FD                DATA  :FD
422 CD4B 69E3              DBL   :E369
423                *
424 CD4D 05        STROF   DATA  :05
425 CD4E 54                DATA  :54       T
426 CD4F 52                DATA  :52       R
427 CD50 4F                DATA  :4F       O
428 CD51 46                DATA  :46       F
429 CD52 46                DATA  :46       F
430 CD53 FE                DATA  :FE
431 CD54 69E3              DBL   :E369
432                *
433 CD56 04        STALK   DATA  :04
434 CD57 54                DATA  :54       T
435 CD58 41                DATA  :41       A
```

```
436 CD59 4C                DATA  :4C       L
437 CD5A 4B                DATA  :4B       K
438 CD5B FB                DATA  :FB
439 CD5C 14E3              DBL   :E314
440                *
441 CD5E 00                DATA  :00       End of table
442 CD5F E5                DATA  :E5       Assignment (= LET)
443 CD60 FEE0              DBL   :E0FE
444                *
445 CD62 FF                DATA  :FF
446 CD63 FF                DATA  :FF
447                *
448                **********************
449                * RUN basiccmd TALK *
450                **********************
451                *
452 CD64 CDF8E6    RTALK   CALL  :E6F8     (0) Get addr parameter
453                                        block in HL
454 CD67 7E        RTK10   MOV   A,M       Get code
455 CD68 23                INX   H
456 CD69 B7                ORA   A
457 CD6A F8                RM              Ready if code = FF (end)
458 CD6B FE05              CPI   :05
459 CD6D DA45CE            JC    :CE45     Jump if freq. code
460 CD70 FE0C              CPI   :0C
461 CD72 DA94EE            JC    :EE94     (0) Jump if volume code
462 CD75 C36DEC            JMP   :EC6D     (0) Continu
463                *
464                ***************
465                * STRING DATA *
466                ***************
467                *
468 CD78 08        LC236   DATA  :08
469 CD79 57                DATA  :57       W
470 CD7A 41                DATA  :41       A
471 CD7B 49                DATA  :49       I
472 CD7C 54                DATA  :54       T
473 CD7D 20                DATA  :20
474 CD7E 4D                DATA  :4D       M
475 CD7F 45                DATA  :45       E
476 CD80 4D                DATA  :4D       M
477                *
478 CD81 09        LC237   DATA  :09
479 CD82 57                DATA  :57       W
480 CD83 41                DATA  :41       A
481 CD84 49                DATA  :49       I
482 CD85 54                DATA  :54       T
483 CD86 20                DATA  :20
484 CD87 54                DATA  :54       T
485 CD88 49                DATA  :49       I
486 CD89 4D                DATA  :4D       M
487 CD8A 45                DATA  :45       E
488                *
489                *
490                *
491 CD8B                   END
```



```
**************************
* S Y M B O L   T A B L E *
**************************

CMDTB  CBBF   LC236  CD78   LC237  CD81   RTALK  CD64
RTK10  CD67   S?     CCEA   SCALM  CD09   SCHECK CD2B
SCLEAR CD12   SCOLG  CC9B   SCOLT  CC91   SCONT  CBC6
SCURS  CC68   SDATA  CCAE   SDIM   CCF5   SDOT   CC7A
SDRAW  CC81   SEDIT  CC42   SEND   CBD6   SENV   CC5C
SFILL  CC89   SFOR   CCD2   SGOSUB CC01   SGOTO  CBF9
SIF    CCC5   SIMP   CC0A   SINPUT CCA5   SLET   CCBE
SLIST  CC3A   SLOAD  CD1B   SLODA  CC1A   SMODE  CC72
SNEW   CBBF   SNEXT  CCD9   SNOISE CC53   SON    CCEF
SOUT   CC23   SPOKE  CC2A   SPRINT CCE1   SREAD  CCB6
SREM   CCCB   SREST  CBDD   SRET   CBE8   SRUN   CBF2
SSAVA  CC11   SSAVE  CD23   SSOUND CC4A   SSTEP  CD3D
SSTOP  CBCE   STALK  CD56   STROF  CD4D   STRON  CD45
SUT    CD03   SWAIT  CC32
```

```
002                             ORG   :CD8B
003                     *
004                     *
005                     *
006                     ****************************************
007                     * POINTERS TO STRINGS OF BASIC COMMANDS *
008                     ****************************************
009                     *
010                     * This table, with base at CC08, is used for
011                     * printing the Basic instructions during a
012                     * listing.
013                     *
014                     * From the TOKEN, the address in this table can
015                     * be found by:
016                     *      CC08 + 3x TOKEN = address in table
017                     * This is done by a routine on 0ECCC.
018                     *
019                     * The address given points to the memory location
020                     * on which the particular string can be found.
021                     *
022                     * The data byte after the address is an offset
023                     * with base at 0ECF8. Therewith the instructions
024                     * can be found about what to print after the
025                     * Basic statement (when performing LIST).
026                     *
027 CD8B BFCB        CDTAB   DBL   :CBBF     NEW
028 CD8D 00                 DATA  :00
029                     *
030 CD8E C6CB               DBL   :CBC6     CONT
031 CD90 00                 DATA  :00
032                     *
033 CD91 CECB               DBL   :CBCE     STOP
034 CD93 00                 DATA  :00
035                     *
036 CD94 D6CB               DBL   :CBD6     END
037 CD96 00                 DATA  :00
038                     *
039 CD97 DDCB               DBL   :CBDD     RESTORE
040 CD99 00                 DATA  :00
041                     *
042 CD9A E8CB               DBL   :CBE8     RETURN
043 CD9C 00                 DATA  :00
044                     *
045 CD9D F2CB               DBL   :CBF2     RUN
046 CD9F 00                 DATA  :00
047                     *
048 CDA0 F2CB               DBL   :CBF2     RUN
049 CDA2 01                 DATA  :01
050                     *
051 CDA3 F9CB               DBL   :CBF9     GOTO
052 CDA5 01                 DATA  :01
053                     *
054 CDA6 01CC               DBL   :CC01     GOSUB
055 CDA8 01                 DATA  :01
056                     *
057 CDA9 1BCD               DBL   :CD1B     LOAD
058 CDAB 04                 DATA  :04
059                     *
060 CDAC 23CD               DBL   :CD23     SAVE
061 CDAE 04                 DATA  :04
062                     *
063 CDAF 2BCD               DBL   :CD2B     CHECK
```



```
064 CDB1 00                 DATA  :00
065                     *
066 CDB2 23CC               DBL   :CC23     OUT
067 CDB4 05                 DATA  :05
068                     *
069 CDB5 2ACC               DBL   :CC2A     POKE
070 CDB7 05                 DATA  :05
071                     *
072 CDB8 32CC               DBL   :CC32     WAIT
073 CDBA 0A                 DATA  :0A
074                     *
075 CDBB 78CD               DBL   :CD78     WAIT MEM
076 CDBD 0A                 DATA  :0A
077                     *
078 CDBE 81CD               DBL   :CD81     WAIT TIME
079 CDC0 04                 DATA  :04
080                     *
081 CDC1 3ACC               DBL   :CC3A     LIST
082 CDC3 00                 DATA  :00
083                     *
084 CDC4 3ACC               DBL   :CC3A     LIST
085 CDC6 01                 DATA  :01
086                     *
087 CDC7 3ACC               DBL   :CC3A     LIST
088 CDC9 0B                 DATA  :0B
089                     *
090 CDCA 4ACC               DBL   :CC4A     SOUND
091 CDCC 0C                 DATA  :0C
092                     *
093 CDCD 53CC               DBL   :CC53     NOISE
094 CDCF 0D                 DATA  :0D
095                     *
096 CDD0 5CCC               DBL   :CC5C     ENVELOPE
097 CDD2 0E                 DATA  :0E
098                     *
099 CDD3 68CC               DBL   :CC68     CURSOR
100 CDD5 05                 DATA  :05
101                     *
102 CDD6 75CC               DBL   :CC75     MODE
103 CDD8 0F                 DATA  :0F
104                     *
105 CDD9 7ACC               DBL   :CC7A     DOT
106 CDDB 07                 DATA  :07
107                     *
108 CDDC 81CC               DBL   :CC81     DRAW
109 CDDE 08                 DATA  :08
110                     *
111 CDDF 89CC               DBL   :CC89     FILL
112 CDE1 08                 DATA  :08
113                     *
114 CDE2 91CC               DBL   :CC91     COLORT
115 CDE4 09                 DATA  :09
116                     *
117 CDE5 9BCC               DBL   :CC9B     COLORG
118 CDE7 09                 DATA  :09
119                     *
120 CDE8 A5CC               DBL   :CCA5     INPUT
121 CDEA 11                 DATA  :11
122                     *
123 CDEB A5CC               DBL   :CCA5     INPUT
124 CDED 10                 DATA  :10
125                     *
```

```
126 CDEE AECC               DBL   :CCAE     DATA
127 CDF0 03                 DATA  :03
128                 *
129 CDF1 B6CC               DBL   :CCB6     READ
130 CDF3 11                 DATA  :11
131                 *
132 CDF4 BECC               DBL   :CCBE     LET
133 CDF6 13                 DATA  :13
134                 *
135 CDF7 0000               DBL   :0000     Assignment (= LET)
136 CDF9 13                 DATA  :13
137                 *
138 CDFA C5CC               DBL   :CCC5     IF
139 CDFC 14                 DATA  :14
140                 *
141 CDFD C5CC               DBL   :CCC5     IF
142 CDFF 15                 DATA  :15
143                 *
144 CE00 C5CC               DBL   :CCC5     IF
145 CE02 16                 DATA  :16
146                 *
147 CE03 CBCC               DBL   :CCCB     REM
148 CE05 03                 DATA  :03
149                 *
150 CE06 D2CC               DBL   :CCD2     FOR
151 CE08 17                 DATA  :17
152                 *
153 CE09 D9CC               DBL   :CCD9     NEXT
154 CE0B 00                 DATA  :00
155                 *
156 CE0C D9CC               DBL   :CCD9     NEXT
157 CE0E 18                 DATA  :18
158                 *
159 CE0F E1CC               DBL   :CCE1     PRINT
160 CE11 19                 DATA  :19
161                 *
162 CE12 EFCC               DBL   :CCEF     ON
163 CE14 1A                 DATA  :1A
164                 *
165 CE15 EFCC               DBL   :CCEF     ON
166 CE17 1B                 DATA  :1B
167                 *
168 CE18 F5CC               DBL   :CCF5     DIM
169 CE1A 11                 DATA  :11
170                 *
171 CE1B FCCC               DBL   :CCFC     '***'
172 CE1D 03                 DATA  :03
173                 *
174 CE1E 03CD               DBL   :CD03     UT
175 CE20 00                 DATA  :00
176                 *
177 CE21 09CD               DBL   :CD09     CALLM
178 CE23 1C                 DATA  :1C
179                 *
180 CE24 12CD               DBL   :CD12     CLEAR
181 CE26 04                 DATA  :04
182                 *
183 CE27 0000               DBL   :0000     IMP
184 CE29 00                 DATA  :00
185                 *
186 CE2A 3ACC               DBL   :CC3A     LIST
187 CE2C 00                 DATA  :00
```



```
188                 *
189 CE2D 3ACC               DBL   :CC3A     LIST
190 CE2F 01                 DATA  :01
191                 *
192 CE30 3ACC               DBL   :CC3A     LIST
193 CE32 0B                 DATA  :0B
194                 *
195 CE33 11CC               DBL   :CC11     SAVEA
196 CE35 1E                 DATA  :1E
197                 *
198 CE36 1ACC               DBL   :CC1A     LOADA
199 CE38 1E                 DATA  :1E
200                 *
201 CE39 56CD               DBL   :CD56     TALK
202 CE3B 04                 DATA  :04
203                 *
204 CE3C 3DCD               DBL   :CD3D     STEP
205 CE3E 00                 DATA  :00
206                 *
207 CE3F 45CD               DBL   :CD45     TRON
208 CE41 00                 DATA  :00
209                 *
210 CE42 4DCD               DBL   :CD4D     TROFF
211 CE44 00                 DATA  :00
212                 *
213                 ******************************
214                 * part of Run 'TALK' (CD6D) *
215                 ******************************
216                 *
217                 * Set frequencies of channels 0,1 or 2.
218                 *
219 CE45 5F         RTK60   MOV   E,A       Code in E (=lobyte
220                                         osc. addr)
221 CE46 16FC               MVI   D,:FC
222 CE48 7E                 MOV   A,M       Get 1st byte freq.code
223 CE49 12                 STAX  D         into osc.
224 CE4A 23                 INX   H
225 CE4B 7E                 MOV   A,M       Get 2nd byte freq.code
226 CE4C 12                 STAX  D         into osc.
227 CE4D C347EA             JMP   :EA47     (0) Handle next code
228                 *
229 CE50 FF                 DATA  :FF
230                 *
231                 ******************************
232                 * GET (M+1) IN E, ZERO M+1 *
233                 ******************************
234                 *
235                 * Part of EARRAY (CB5B).
236                 *
237                 * Entry: HL points to M.
238                 * Exit:  HL points to M+1. (M+1) in E.
239                 *        AFBCD preserved.
240                 *
241 CE51 23         MPT09   INX   H
242 CE52 5E                 MOV   E,M
243 CE53 3600               MVI   M,:00
244 CE55 C9                 RET
245                 *
246                 ***************
247                 * STRING DATA *
248                 ***************
249                 *
```

```
CE56 05          MSPACE  DATA  :05
CE57 53                  DATA  :53       S
CE58 50                  DATA  :50       P
CE59 41                  DATA  :41       A
CE5A 43                  DATA  :43       C
CE5B 45                  DATA  :45       E
                 *
                 ****************************
                 * part of RUN DIM (0E639) *
                 ****************************
                 *
CE5C 2B          MPT41   DCX   H
CE5D C35BCB              JMP   :CB5B     Erase array if exists
                 *
                 ***********************************
                 * GET TABNUMBER IN L, DOUTC IN A *
                 ***********************************
                 *
                 * Part of Run 'TAB'.
                 *
CE60 CD1DE7      MPT50   CALL  :E71D     (0) Get nr of tabs in A
CE63 6F                  MOV   L,A      save it in L
CE64 3A3101              LDA   :0131    Get output direction
CE67 C9                  RET
                 *
                 ****************************************
                 * PRINT EXPRESSION FOLLOWED BY A SPACE *
                 ****************************************
                 *
                 * Entry SCHSP frequently used to print a space.
                 *
CE68 CDA2EE      LC230   CALL  :EEA2     (0) Print expression
CE6B 3E20        SCHSP   MVI   A,:20
CE6D C360DD              JMP   :DD60     Print space
                 *
                 *************
                 * PRINT ',' *
                 *************
                 *
                 * Entry: None.
                 * Exit:  FBCDEHL preserved.
                 *
CE70 3E2C        SCHCO   MVI   A,:2C
CE72 C360DD              JMP   :DD60     Print ','
                 *
                 ********************************
                 * PRINT A STRING BETWEEN SPACES *
                 ********************************
                 *
                 * Entry: Pointer to stringpointer on stack.
                 * Exit:  BC preserved. AFDEHL corrupted.
                 *
CE75 CD6BCE      STXSS   CALL  :CE6B     Print space
CE78 E3          STXTS   XTHL            Get stringpntr from stack
CE79 5E                  MOV   E,M      ) Store addr string in DE
CE7A 23                  INX   H        )
CE7B 56                  MOV   D,M      )
CE7C 23                  INX   H        )
CE7D E3                  XTHL           Addr after pntr on stack
CE7E EB                  XCHG           Addr string in HL
CE7F CD32DB              CALL  :DB32    Print string pointed by HL
CE82 C36BCE              JMP   :CE6B    Print space
```



```
                 *
                 ****************************
                 * EDIT: PRINT TEXT COMPLETE *
                 ****************************
                 *
CE85 CD17EF      LC216   CALL  :EF17     (2) Print text complete
CE88 C338E1              JMP   :E138    (2) Popall, ret
                 *
                 *********************************
                 * LIST ARRAY NAME - (not used) *
                 *********************************
                 *
CE8B D5          LC217   PUSH  D
CE8C CDF7EE              CALL  :EEF7    (0) List array name
CE8F D1                  POP   D
CE90 C9                  RET
                 *
                 ****************
                 * part of C6BA *
                 ****************
                 *
CE91 E5          LC196   PUSH  H
CE92 C32DE9              JMP   :E92D    (2) Now set up screen bits
                                        for mode 1
                 *
                 **************
                 * (not used) *
                 **************
                 *
CE95 3AA200      LC218   LDA   :00A2     Get startaddr edit buffer
CE98 C37CE9              JMP   :E97C    (0)
                 *
                 ****************************
                 * CONVERT MACC FOR OUTPUT *
                 ****************************
                 *
                 * The MACC contents is converted from FPT to ASCII.
                 *
                 * Exit: AF corrupted, BCDEHL preserved.
                 *
CE9B CD21C0      FBCP    CALL  :C021     Convert FPT nr for output
CE9E C5                  PUSH  B
CE9F 0601                MVI   B,:01    Cannot trim last dec.place
CEA1 C365DB              JMP   :DB65    Tidy up into external form
                 *
                 **********************
                 * LIST CURRENT LINE *
                 **********************
                 *
                 * Lists a program line if trace flag set.
                 * Part of C8F5.
                 *
CEA4 0B          MPT06   DCX   B
CEA5 CD55DD              CALL  :DD55    Cursor to begin next line
CEA8 C3ABEC              JMP   :ECAB    (0) List current line
                 *
CEAB D1          LC219   POP   D        (Not used)
CEAC C9                  RET
                 *
                 ************************
                 * part of SMKRM (CA01) *
                 ************************
```

```
                        *
CEAD D1         MPT07   POP   D         Return, CY=1
CEAE F1                 POP   PSW
CEAF 37                 STC
CEB0 C9                 RET
                        *
CEB1 F1         MPT08   POP   PSW       Return, CY=0
CEB2 37                 STC
CEB3 3F                 CMC
CEB4 C9                 RET
                        *
                        ************************
                        * CHANGE SCREEN MODE   *
                        ************************
                        *
                        * Part of Run 'MODE' (0E5BB).
                        *
                        * Entry: New mode in A.
                        *
CEB5 EF         MPT40   RST   5         Change mode
CEB6 18                 DATA  :18
CEB7 DA10DA             JC    :DA10     If insufficient memory:
                                        error 'OUT OF MEMORY'.
CEBA C9                 RET
                        *
                        ******************************
                        * part of RUN CLEAR (0E6B5)  *
                        ******************************
                        *
                        * Checks if more than 4 bytes are cleared.
                        *
                        * Entry: HL: Number of bytes to be cleared.
                        *        F : flags on hibyte HL.
                        *
CEBB 110400     MPT43   LXI   D,:0004   Must be at least 4 bytes
CEBE F414DE             CP    :DE14     Compare HL-DE if not >32k
CEC1 DA15DA             JC    :DA15     Run error 'NUMBER OUT
                                        OF RANGE'if < 4.
CEC4 C9                 RET
                        *
CEC5 FF                 DATA  :FF
                        *
                        ************************************
                        * SET HEAP SIZE TO DEFAULT VALUE   *
                        ************************************
                        *
                        * Part of emergency stop routine (CA25).
                        * Also runs a NEW command.
                        *
CEC6 210001     HRNEW   LXI   H,:0100
CEC9 229D02             SHLD  :029D     Store HEAP default value
CECC C3B5DE             JMP   :DEB5     Run 'NEW'
                        *
                        ****************************************
                        * RE-ORGANISE SCREEN AFTER 'DAI-PC'    *
                        ****************************************
                        *
                        * Lines after 'DAI PERSONAL COMPUTER' are set in
                        * unit colour mode.
                        * Set line mode byte to 7F and first char.
                        * byte to 20, colour 00 during screen init.
                        *
```



```
                        * Entry: HL line mode byte of part. line.
                        *
CECF 367F       MPT04   MVI   M,:7F     Wide char line contr byte
CED1 2B                 DCX   H
CED2 2B                 DCX   H
CED3 3620               MVI   M,:20     Load space
CED5 2B                 DCX   H
CED6 3600               MVI   M,:00     Colour no change
CED8 2B                 DCX   H
CED9 C9                 RET
                        *
                        ****************************************
                        * part of RUN 'WAIT(TIME)' (DFD5/DFF7) *
                        ****************************************
                        *
CEDA 0B         REX1D   DCX   B
CEDB C31DE7             JMP   :E71D     (0) Get value of argument
                                        in A (max. FF)
                        *
                        ********************************************
                        * EVALUATE ARGUMENTS IN NUMERIC EXPRESSION *
                        ********************************************
                        *
                        * Not used.
                        *
CEDE F5         LC231   PUSH  PSW
CEDF CD19E8             CALL  :E819     (0) Evaluate arguments
CEE2 F1                 POP   PSW
CEE3 C9                 RET
                        *
                        ************************************
                        * SELECT ROM BANK 0; PRINT MESSAGE *
                        ************************************
                        *
                        * When CEE4 is called, the 2 bytes following
                        * the CALL-instruction indicate the message to
                        * be printed by the routine DAFF.
                        *
CEE4 3A4000     SELB0   LDA   :0040     Get POROM
CEE7 E63F               ANI   :3F       Select ROM bank 0
CEE9 324000             STA   :0040     Set POROM
CEEC 3206FD             STA   :FD06     and PORO
CEEF C3FFDA             JMP   :DAFF     Print message
                        *
                        ****************************************
                        * EDIT: RETURN FROM 'DELETE CHARACTER' *
                        ****************************************
                        *
                        * Part of 2EFCC.
                        *
CEF2 E1         LC232   POP   H
CEF3 CD30E3             CALL  :E330     (2) Put cursor on screen
CEF6 C395EF             JMP   :EF95     (2) Popall, ret, CY=1
                        *
                        ****************************************
                        * PRINT 'COMPUTER' UNDER 'DAI PERSONAL' *
                        ****************************************
                        *
                        * Part of RESET (C751).
                        * During screen initialisation used to set a
                        * new line mode byte between both parts of the
                        * message. Line colour bytes are set for medium
```

```
498                 * resolution.
499                 *
500                 * Entry: HL: line mode byte to be changed.
501                 *        DE: offset for calculation next
502                 *            line mode byte.
503                 * Exit:  HL: next line mode byte.
504                 *
505 CEF9 365F      MPT03   MVI   M,:5F     Set line mode byte
506 CEFB 2B                DCX   H
507 CEFC 3640              MVI   M,:40     Set line colour byte
508 CEFE 23                INX   H
509 CEFF 19                DAD   D         Addr. next line mode byte
510 CF00 C9                RET
511                 *
512 CF01 FF                DATA  :FF
513                 *
514                 *
515                 *
516 CF02                   END
```

****
* S Y M B O L T A B L E *
****

CDTAB	CD8B	FBCP	CE9B	HRNEW	CEC6	LC196	CE91
LC216	CE85	LC217	CE8B	LC218	CE95	LC219	CEAB
LC230	CE68	LC231	CEDE	LC232	CEF2	MPT03	CEF9
MPT04	CECF	MPT06	CEA4	MPT07	CEAD	MPT08	CEB1
MPT09	CE51	MPT40	CEB5	MPT41	CE5C	MPT43	CEBB
MPT50	CE60	MSPACE	CE56	REX1D	CEDA	RTK60	CE45
SCHCO	CE70	SCHSP	CE6B	SELBO	CEE4	STXSS	CE75
STXTS	CE78						


```
002                         ORG   :CF02
003                 *
004                 *
005                 *
006                 ****************************************
007                 * POINTERS TO ROUTINES BASICCOMMANDS *
008                 ****************************************
009                 *
010                 * This table, with base at CF00, gives the
011                 * addresses of the routines for execution of the
012                 * Basic statements.
013                 * The offset from baseaddress CF00 can be found
014                 * by adding 2x TOKEN to the base address.
015                 * Address indicates begin subroutine (ROM bank 0).
016                 *
017                 * The number given between brackets is the TOKEN.
018                 *
019 CF02 B5DE      CITAB   DBL   :DEB5     (81) NEW
020 CF04 D5DE              DBL   :DED5     (82) CONT
021 CF06 03DF              DBL   :DF03     (83) STOP
022 CF08 0CDF              DBL   :DF0C     (84) END
023 CF0A 01E4              DBL   :E401     (85) RESTORE
024 CF0C 4CDF              DBL   :DF4C     (86) RETURN
025 CF0E 9EDF              DBL   :DF9E     (87) RUN
026 CF10 BADF              DBL   :DFBA     (88) RUN <linenumber>
027 CF12 63DF              DBL   :DF63     (89) GOTO
028 CF14 2ADF              DBL   :DF2A     (8A) GOSUB
029 CF16 70D2              DBL   :D270     (8B) LOAD
030 CF18 3DD2              DBL   :D23D     (8C) SAVE
031 CF1A C3D2              DBL   :D2C3     (8D) CHECK
032 CF1C C9DF              DBL   :DFC9     (8E) OUT
033 CF1E C0DF              DBL   :DFC0     (8F) POKE
034 CF20 D5DF              DBL   :DFD5     (90) WAIT
035 CF22 F7DF              DBL   :DFF7     (91) WAIT MEM
036 CF24 16E0              DBL   :E016     (92) WAIT TIME
037 CF26 97E1              DBL   :E197     (93) LIST <whole program>
038 CF28 AAE1              DBL   :E1AA     (94) LIST <linenumber>
039 CF2A B6E1              DBL   :E1B6     (95) LIST <part of progr>
040 CF2C BCE4              DBL   :E4BC     (96) SOUND
041 CF2E 0CE5              DBL   :E50C     (97) NOISE
042 CF30 70E5              DBL   :E570     (98) ENVELOPE
043 CF32 B2E5              DBL   :E5B2     (99) CURSOR
044 CF34 BBE5              DBL   :E5BB     (9A) MODE
045 CF36 C1E5              DBL   :E5C1     (9B) DOT
046 CF38 CEE5              DBL   :E5CE     (9C) DRAW
047 CF3A D7E5              DBL   :E5D7     (9D) FILL
048 CF3C 0EE6              DBL   :E60E     (9E) COLORT
049 CF3E 15E6              DBL   :E615     (9F) COLORG
050 CF40 02E3              DBL   :E302     (A0) INPUT
051 CF42 FCE2              DBL   :E2FC     (A1) INPUT <with prompt>
052 CF44 8FE1              DBL   :E18F     (A2) DATA
053 CF46 23E3              DBL   :E323     (A3) READ
054 CF48 5AE4              DBL   :E45A     (A4) LET
055 CF4A 5AE4              DBL   :E45A     (A5) assignment
056 CF4C 20DF              DBL   :DF20     (A6) IF THEN <statement>
057 CF4E 15DF              DBL   :DF15     (A7) IF GOTO
058 CF50 15DF              DBL   :DF15     (A8) IF THEN <linenumber>
059 CF52 8FE1              DBL   :E18F     (A9) REM
060 CF54 2BE0              DBL   :E02B     (AA) FOR .. TO
061 CF56 E5E0              DBL   :E0E5     (AB) NEXT
062 CF58 C5E0              DBL   :E0C5     (AC) NEXT <variable>
063 CF5A B3E2              DBL   :E2B3     (AD) PRINT
```

```
064 CF5C 6ADF                DBL   :DF6A     (AE) ON GOTO
065 CF5E 71DF                DBL   :DF71     (AF) ON GOSUB
066 CF60 2FE6                DBL   :E62F     (B0) DIM
067 CF62 33DA                DBL   :DA33     (B1) *** (error line run)
068 CF64 9EE6                DBL   :E69E     (B2) UT
069 CF66 A4E6                DBL   :E6A4     (B3) CALLM
070 CF68 B5E6                DBL   :E6B5     (B4) CLEAR
071 CF6A 95E1                DBL   :E195     (B5) IMP
072 CF6C F5E1                DBL   :E1F5     (B6) EDIT <total>
073 CF6E 53E2                DBL   :E253     (B7) EDIT <linenumber>
074 CF70 5CE2                DBL   :E25C     (B8) EDIT <part>
075 CF72 1DD8                DBL   :D81D     (B9) SAVEA
076 CF74 5ED8                DBL   :D85E     (BA) LOADA
077 CF76 64CD                DBL   :CD64     (BB) TALK
078 CF78 FEDE                DBL   :DEFE     (BC) STEP
079 CF7A CEE6                DBL   :E6CE     (BD) TRON
080 CF7C D5E6                DBL   :E6D5     (BE) TROFF
081                     *
082                     ******************************
083                     * part of RUN 'ASC' (OEACF) *
084                     ******************************
085                     *
086 CF7E B7             MPT51  ORA   A          Nulstring ?
087 CF7F CA7CEB                JZ    :EB7C     (0) Then 0 into MACC
088 CF82 23                    INX   H         Else:
089 CF83 C3C7EA                JMP   :EAC7     (0) Next byte from MEM
090                                             into MACC
091                     *
092                     ******************************************
093                     * TABLE PREFIXES FOR UNITARY OPERATIONS *
094                     ******************************************
095                     *
096                     * Only used by LIST for decoding.
097                     * The prefix gives a code for unitary operators,
098                     * which on encoding are directly determined, not
099                     * looked up in a table.
100                     *
101                     * Format: length of name / name / 5-bit opcode
102                     *
103                     OPTBB
104 CF86 01             SBRA   DATA  :01
105 CF87 28                    DATA  :28       (
106 CF88 1A                    DATA  :1A
107                     *
108 CF89 01                    DATA  :01
109 CF8A 2D                    DATA  :2D       -
110 CF8B 1D                    DATA  :1D
111                     *
112 CF8C 01                    DATA  :01
113 CF8D 2B                    DATA  :2B       +
114 CF8E 1C                    DATA  :1C
115                     *
116 CF8F 00                    DATA  :00       Fixed FPT conversion
117 CF90 1F                    DATA  :1F
118                     *
119                     ***************************
120                     * TABLE BINARY OPERATORS *
121                     ***************************
122                     *
123                     * Format: 1 byte:  length of name.
124                     *         n bytes: name.
125                     *         1 byte:  code byte:
```



```
126                     *                 highest 3 bits: priority.
127                     *                 lowest  5 bits: opcode.
128                     *
129                     OPTAB
130 CF91 01             SDIV   DATA  :01
131 CF92 2F                    DATA  :2F       /
132 CF93 C2                    DATA  :C2
133                     *
134 CF94 01                    DATA  :01
135 CF95 2A                    DATA  :2A       *
136 CF96 C3                    DATA  :C3
137                     *
138 CF97 03                    DATA  :03
139 CF98 4D                    DATA  :4D       M
140 CF99 4F                    DATA  :4F       O
141 CF9A 44                    DATA  :44       D
142 CF9B CF                    DATA  :CF
143                     *
144 CF9C 01                    DATA  :01
145 CF9D 5E                    DATA  :5E       ^
146 CF9E E4                    DATA  :E4
147                     *
148 CF9F 04                    DATA  :04
149 CFA0 49                    DATA  :49       I
150 CFA1 41                    DATA  :41       A
151 CFA2 4E                    DATA  :4E       N
152 CFA3 44                    DATA  :44       D
153 CFA4 69                    DATA  :69
154                     *
155 CFA5 03                    DATA  :03
156 CFA6 49                    DATA  :49       I
157 CFA7 4F                    DATA  :4F       O
158 CFA8 52                    DATA  :52       R
159 CFA9 6A                    DATA  :6A
160                     *
161 CFAA 04                    DATA  :04
162 CFAB 49                    DATA  :49       I
163 CFAC 58                    DATA  :58       X
164 CFAD 4F                    DATA  :4F       O
165 CFAE 52                    DATA  :52       R
166 CFAF 6C                    DATA  :6C
167                     *
168 CFB0 03                    DATA  :03
169 CFB1 53                    DATA  :53       S
170 CFB2 48                    DATA  :48       H
171 CFB3 4C                    DATA  :4C       L
172 CFB4 8D                    DATA  :8D
173                     *
174 CFB5 03                    DATA  :03
175 CFB6 53                    DATA  :53       S
176 CFB7 48                    DATA  :48       H
177 CFB8 52                    DATA  :52       R
178 CFB9 8E                    DATA  :8E
179                     *
180 CFBA 02                    DATA  :02
181 CFBB 3E                    DATA  :3E       >
182 CFBC 3D                    DATA  :3D       =
183 CFBD 50                    DATA  :50
184                     *
185 CFBE 01                    DATA  :01
186 CFBF 3E                    DATA  :3E       >
187 CFC0 51                    DATA  :51
```

```
                        *
CFC1 02                 DATA  :02
CFC2 3C                 DATA  :3C       <
CFC3 3E                 DATA  :3E       >
CFC4 52                 DATA  :52
                        *
CFC5 02                 DATA  :02
CFC6 3C                 DATA  :3C       <
CFC7 3D                 DATA  :3D       =
CFC8 53                 DATA  :53
                        *
CFC9 01                 DATA  :01
CFCA 3C                 DATA  :3C       <
CFCB 54                 DATA  :54
                        *
CFCC 01                 DATA  :01
CFCD 3D                 DATA  :3D       >
CFCE 55                 DATA  :55
                        *
CFCF 03                 SAND  DATA  :03
CFD0 41                 DATA  :41       A
CFD1 4E                 DATA  :4E       N
CFD2 44                 DATA  :44       D
CFD3 38                 DATA  :38
                        *
CFD4 02                 DATA  :02
CFD5 4F                 DATA  :4F       O
CFD6 52                 DATA  :52       R
CFD7 39                 DATA  :39
                        *
                        ****************************
                        * TABLE UNITARY OPERATORS *
                        ****************************
                        *
                        * Format: 1 byte:  Length of name.
                        *         n bytes: Name.
                        *         1 byte:  Code byte:
                        *                  3 highest bits: priority.
                        *                  5 lowest bits:  opcode.
                        *
CFD8 04                 OPTBM DATA  :04
CFD9 49                 DATA  :49       I
CFDA 4E                 DATA  :4E       N
CFDB 4F                 DATA  :4F       O
CFDC 54                 DATA  :54       T
CFDD 1E                 DATA  :1E
                        *
CFDE 01                 DATA  :01
CFDF 2B                 DATA  :2B       +
CFE0 A0                 DATA  :A0
                        *
CFE1 01                 DATA  :01
CFE2 2D                 DATA  :2D       -
CFE3 A1                 DATA  :A1
                        *
CFE4 00                 DATA  :00       End of table
CFE5 00                 DATA  :00
                        *
                        ********************************
                        * TABLE STRINGS BASIC FUNCTIONS *
                        ********************************
                        *
```



```
                        * Format: 1 byte:  Length of name.
                        *         n bytes: Name.
                        *         1 byte:  High nibble: type of info.
                        *                  Low nibble: nr of arguments.
                        *         1 byte:  High nibble: required type
                        *                  of variable expected:
                        *                  (0=FPT, 1=INT, 2=STR).
                        *
                        FUNTB
CFE6 03                 SABS  DATA  :03
CFE7 41                 DATA  :41       A
CFE8 42                 DATA  :42       B
CFE9 53                 DATA  :53       S
CFEA 01                 DATA  :01
CFEB 00                 DATA  :00
                        *
CFEC 04                 SALOG DATA  :04
CFED 41                 DATA  :41       A
CFEE 4C                 DATA  :4C       L
CFEF 4F                 DATA  :4F       O
CFF0 47                 DATA  :47       G
CFF1 01                 DATA  :01
CFF2 00                 DATA  :00
                        *
CFF3 03                 SASC  DATA  :03
CFF4 41                 DATA  :41       A
CFF5 53                 DATA  :53       S
CFF6 43                 DATA  :43       C
CFF7 11                 DATA  :11
CFF8 20                 DATA  :20
                        *
CFF9 04                 SCHR  DATA  :04
CFFA 43                 DATA  :43       C
CFFB 48                 DATA  :48       H
CFFC 52                 DATA  :52       R
CFFD 24                 DATA  :24       $
CFFE 21                 DATA  :21
CFFF 10                 DATA  :10
                        *
D000 04                 SCURX DATA  :04
D001 43                 DATA  :43       C
D002 55                 DATA  :55       U
D003 52                 DATA  :52       R
D004 58                 DATA  :58       X
D005 10                 DATA  :10
                        *
D006 04                 SCURY DATA  :04
D007 43                 DATA  :43       C
D008 55                 DATA  :55       U
D009 52                 DATA  :52       R
D00A 59                 DATA  :59       Y
D00B 10                 DATA  :10
                        *
D00C 03                 SEXP  DATA  :03
D00D 45                 DATA  :45       E
D00E 58                 DATA  :58       X
D00F 50                 DATA  :50       P
D010 01                 DATA  :01
D011 00                 DATA  :00
                        *
D012 04                 SFRAC DATA  :04
D013 46                 DATA  :46       F
```

```
312 D014 52                DATA  :52       R
313 D015 41                DATA  :41       A
314 D016 43                DATA  :43       C
315 D017 01                DATA  :01
316 D018 00                DATA  :00
317              *
318 D019 03      SFRE      DATA  :03
319 D01A 46                DATA  :46       F
320 D01B 52                DATA  :52       R
321 D01C 45                DATA  :45       E
322 D01D 10                DATA  :10
323              *
324 D01E 04      SFREQ     DATA  :04
325 D01F 46                DATA  :46       F
326 D020 52                DATA  :52       R
327 D021 45                DATA  :45       E
328 D022 51                DATA  :51       Q
329 D023 11                DATA  :11
330 D024 00                DATA  :00
331              *
332 D025 04      SGETC     DATA  :04
333 D026 47                DATA  :47       G
334 D027 45                DATA  :45       E
335 D028 54                DATA  :54       T
336 D029 43                DATA  :43       C
337 D02A 10                DATA  :10
338              *
339 D02B 04      SHEX      DATA  :04
340 D02C 48                DATA  :48       H
341 D02D 45                DATA  :45       E
342 D02E 58                DATA  :58       X
343 D02F 24                DATA  :24       $
344 D030 21                DATA  :21
345 D031 10                DATA  :10
346              *
347 D032 03      SINP      DATA  :03
348 D033 49                DATA  :49       I
349 D034 4E                DATA  :4E       N
350 D035 50                DATA  :50       P
351 D036 11                DATA  :11
352 D037 10                DATA  :10
353              *
354 D038 03      SINT      DATA  :03
355 D039 49                DATA  :49       I
356 D03A 4E                DATA  :4E       N
357 D03B 54                DATA  :54       T
358 D03C 01                DATA  :01
359 D03D 00                DATA  :00
360              *
361 D03E 05      SLEFT     DATA  :05
362 D03F 4C                DATA  :4C       L
363 D040 45                DATA  :45       E
364 D041 46                DATA  :46       F
365 D042 54                DATA  :54       T
366 D043 24                DATA  :24       $
367 D044 22                DATA  :22
368 D045 20                DATA  :20
369 D046 10                DATA  :10
370              *
371 D047 03      SLEN      DATA  :03
372 D048 4C                DATA  :4C       L
373 D049 45                DATA  :45       E
```



```
374 D04A 4E                DATA  :4E       N
375 D04B 11                DATA  :11
376 D04C 20                DATA  :20
377              *
378 D04D 06      SVPT      DATA  :06
379 D04E 56                DATA  :56       V
380 D04F 41                DATA  :41       A
381 D050 52                DATA  :52       R
382 D051 50                DATA  :50       P
383 D052 54                DATA  :54       T
384 D053 52                DATA  :52       R
385 D054 11                DATA  :11
386 D055 30                DATA  :30
387              *
388 D056 03      SLOG      DATA  :03
389 D057 4C                DATA  :4C       L
390 D058 4F                DATA  :4F       O
391 D059 47                DATA  :47       G
392 D05A 01                DATA  :01
393 D05B 00                DATA  :00
394              *
395 D05C 04      SLOGT     DATA  :04
396 D05D 4C                DATA  :4C       L
397 D05E 4F                DATA  :4F       O
398 D05F 47                DATA  :47       G
399 D060 54                DATA  :54       T
400 D061 01                DATA  :01
401 D062 00                DATA  :00
402              *
403 D063 04      SXMAX     DATA  :04
404 D064 58                DATA  :58       X
405 D065 4D                DATA  :4D       M
406 D066 41                DATA  :41       A
407 D067 58                DATA  :58       X
408 D068 10                DATA  :10
409              *
410 D069 04      SYMAX     DATA  :04
411 D06A 59                DATA  :59       Y
412 D06B 4D                DATA  :4D       M
413 D06C 41                DATA  :41       A
414 D06D 58                DATA  :58       X
415 D06E 10                DATA  :10
416              *
417 D06F 04      SMID      DATA  :04
418 D070 4D                DATA  :4D       M
419 D071 49                DATA  :49       I
420 D072 44                DATA  :44       D
421 D073 24                DATA  :24       $
422 D074 23                DATA  :23
423 D075 20                DATA  :20
424 D076 10                DATA  :10
425 D077 10                DATA  :10
426              *
427 D078 03      SPDL      DATA  :03
428 D079 50                DATA  :50       P
429 D07A 44                DATA  :44       D
430 D07B 4C                DATA  :4C       L
431 D07C 11                DATA  :11
432 D07D 10                DATA  :10
433              *
434 D07E 04      SPEEK     DATA  :04
435 D07F 50                DATA  :50       P
```

```
0D80 45              DATA  :45      E
0D81 45              DATA  :45      E
0D82 4B              DATA  :4B      K
0D83 11              DATA  :11
0D84 10              DATA  :10
                *
0D85 02      SPI     DATA  :02
0D86 50              DATA  :50      P
0D87 49              DATA  :49      I
0D88 00              DATA  :00
                *
0D89 06      SRIGHT  DATA  :06
0D8A 52              DATA  :52      R
0D8B 49              DATA  :49      I
0D8C 47              DATA  :47      G
0D8D 48              DATA  :48      H
0D8E 54              DATA  :54      T
0D8F 24              DATA  :24      $
0D90 22              DATA  :22
0D91 20              DATA  :20
0D92 10              DATA  :10
                *
0D93 03      SRND    DATA  :03
0D94 52              DATA  :52      R
0D95 4E              DATA  :4E      N
0D96 44              DATA  :44      D
0D97 01              DATA  :01
0D98 00              DATA  :00
                *
0D99 04      SSCRN   DATA  :04
0D9A 53              DATA  :53      S
0D9B 43              DATA  :43      C
0D9C 52              DATA  :52      R
0D9D 4E              DATA  :4E      N
0D9E 12              DATA  :12
0D9F 10              DATA  :10
0DA0 10              DATA  :10
                *
0DA1 03      SSGN    DATA  :03
0DA2 53              DATA  :53      S
0DA3 47              DATA  :47      G
0DA4 4E              DATA  :4E      N
0DA5 01              DATA  :01
0DA6 00              DATA  :00
                *
0DA7 03      SSPC    DATA  :03
0DA8 53              DATA  :53      S
0DA9 50              DATA  :50      P
0DAA 43              DATA  :43      C
0DAB 21              DATA  :21
0DAC 10              DATA  :10
                *
0DAD 03      SSQR    DATA  :03
0DAE 53              DATA  :53      S
0DAF 51              DATA  :51      Q
0DB0 52              DATA  :52      R
0DB1 01              DATA  :01
0DB2 00              DATA  :00
                *
0DB3 04      SSTR    DATA  :04
0DB4 53              DATA  :53      S
0DB5 54              DATA  :54      T
```



```
0DB6 52              DATA  :52      R
0DB7 24              DATA  :24      $
0DB8 21              DATA  :21
0DB9 00              DATA  :00
                *
0DBA 03      STAB    DATA  :03
0DBB 54              DATA  :54      T
0DBC 41              DATA  :41      A
0DBD 42              DATA  :42      B
0DBE 21              DATA  :21
0DBF 10              DATA  :10
                *
0DC0 03      SVAL    DATA  :03
0DC1 56              DATA  :56      V
0DC2 41              DATA  :41      A
0DC3 4C              DATA  :4C      L
0DC4 01              DATA  :01
0DC5 20              DATA  :20
                *
0DC6 03      SSIN    DATA  :03
0DC7 53              DATA  :53      S
0DC8 49              DATA  :49      I
0DC9 4E              DATA  :4E      N
0DCA 01              DATA  :01
0DCB 00              DATA  :00
                *
0DCC 03      SCOS    DATA  :03
0DCD 43              DATA  :43      C
0DCE 4F              DATA  :4F      O
0DCF 53              DATA  :53      S
0DD0 01              DATA  :01
0DD1 00              DATA  :00
                *
0DD2 03      STAN    DATA  :03
0DD3 54              DATA  :54      T
0DD4 41              DATA  :41      A
0DD5 4E              DATA  :4E      N
0DD6 01              DATA  :01
0DD7 00              DATA  :00
                *
0DD8 04      SASIN   DATA  :04
0DD9 41              DATA  :41      A
0DDA 53              DATA  :53      S
0DDB 49              DATA  :49      I
0DDC 4E              DATA  :4E      N
0DDD 01              DATA  :01
0DDE 00              DATA  :00
                *
0DDF 04      SACOS   DATA  :04
0DE0 41              DATA  :41      A
0DE1 43              DATA  :43      C
0DE2 4F              DATA  :4F      O
0DE3 53              DATA  :53      S
0DE4 01              DATA  :01
0DE5 00              DATA  :00
                *
0DE6 03      SATN    DATA  :03
0DE7 41              DATA  :41      A
0DE8 54              DATA  :54      T
0DE9 4E              DATA  :4E      N
0DEA 01              DATA  :01
0DEB 00              DATA  :00
```

```
                    *
                            DATA  :00       End of table
                    *
                    ********
                    * DATA *
                    ********
                    *
                    ENDFT
DOED 15             FPOSC   DATA  :15       Sound constant
DOEE F4                     DATA  :F4
DOEF 24                     DATA  :24
DOF0 00                     DATA  :00
                    *
DOF1 81             FPM1    DATA  :81       FPT (-1)
DOF2 80                     DATA  :80
DOF3 00                     DATA  :00
DOF4 00                     DATA  :00
                    *
DOF5 02             FPPI    DATA  :02       FPT (PI)
DOF6 C9                     DATA  :C9
DOF7 0F                     DATA  :0F
DOF8 DB                     DATA  :DB
                    *
DOF9 00             I4      DATA  :00       INT (4)  (not used)
DOFA 00                     DATA  :00
DOFB 00                     DATA  :00
DOFC 04                     DATA  :04
                    *
DOFD 00             IRAND   DATA  :00       AND mask
DOFE FF                     DATA  :FF
DOFF FF                     DATA  :FF
D100 FF                     DATA  :FF
                    *
                    *
                    *
D101                        END

****************************
* S Y M B O L   T A B L E *
****************************

CITAB  CF02   ENDFT  DOED   FPM1   DOF1   FPOSC  DOED
FPPI   DOF5   FUNTB  CFE6   I4     DOF9   IRAND  DOFD
MPT51  CF7E   OPTAB  CF91   OPTBB  CF86   OPTBM  CFD8
SABS   CFE6   SACOS  DODF   SALOG  CFEC   SAND   CFCF
SASC   CFF3   SASIN  DOD8   SATN   D0E6   SBRA   CF86
SCHR   CFF9   SCOS   D0CC   SCURX  D000   SCURY  D006
SDIV   CF91   SEXP   D00C   SFRAC  D012   SFRE   D019
SFREQ  D01E   SGETC  D025   SHEX   D02B   SINP   D032
SINT   D038   SLEFT  D03E   SLEN   D047   SLOG   D056
SLOGT  D05C   SMID   D06F   SPDL   D078   SPEEK  D07E
SPI    D085   SRIGHT D089   SRND   D093   SSCRN  D099
SSGN   D0A1   SSIN   D0C6   SSPC   D0A7   SSQR   D0AD
SSTR   D0B3   STAB   D0BA   STAN   D0D2   SVAL   D0C0
SVPT   D04D   SXMAX  D063   SYMAX  D069
```



```
                            ORG   :D101
                    *
                    *
                    *
                    *  ================
                    *** STRING HANDLER ***
                    *  ================
                    *
                    *
                    ****************************
                    * INITIALISE STRING HANDLER *
                    ****************************
                    *
                    * Initialises a given size of string area.
                    * This routine is used once by RESET (C719), but
                    * without purpose. It belongs to another BASIC
                    * package of the firm DAI.
                    *
                    * Entry: None.
                    * Exit:  A=0, BCDEHL preserved. F corrupted.
                    *
D101 AF             SHINIT  XRA   A
D102 320000                 STA   :0000     Zero 0000
D105 C9                     RET
                    *
                    **********************
                    * APPEND TWO STRINGS *
                    **********************
                    *
                    * Appends two strings together to one new string
                    * by adding the 2nd string to the end of the 1st
                    * string.
                    *
                    * Entry: DE: Startaddress 1st string.
                    *            (1st byte is length byte).
                    *        HL: Startaddress 2nd string.
                    * Exit:  AFBCDE preserved.
                    *        HL: Points to new string.
                    *        Error if string too long.
                    *
D106 F5             SHAPP   PUSH  PSW
D107 D5                     PUSH  D
D108 1A                     LDAX  D         Get length 1st string
D109 86                     ADD   M         Calc length new string
D10A DA38DA                 JC    :DA38     Run error 'STRING TOO LONG'
                                              if length >255.
D10D E5                     PUSH  H         Save pntr 2nd string
D10E CD8BD1                 CALL  :D18B     Find place in Heap for
                                              new string
D111 E5                     PUSH  H         Save pntr to new string
D112 23                     INX   H
D113 CD6DD1                 CALL  :D16D     Move 1st string to new loc.
D116 D1                     POP   D
D117 E3                     XTHL
D118 EB                     XCHG
D119 E3                     XTHL
D11A CD6DD1                 CALL  :D16D     Move 2nd string to new loc.
D11D E1                     POP   H         Get pntr to new string
D11E D1                     POP   D
D11F F1                     POP   PSW
D120 C9                     RET
                    *
```