

```
*****
*          C O M M O D O R E   B A S I C   4 . 0
*          ****
```

This is a heavily commented listing of CBM BASIC 4.0. It covers the complete ROM-set, that is the BASIC-interpreter, the machine language monitor, and the operating system. A detailed description of the use of RAM is also given.

All versions of BASIC 4.0 are covered. This means that the two versions of the B-ROM are dealt with, because the first version of this ROM contained a few bugs, as you can read in the listing, on pages 59 through 61.

Furthermore the listing of all E-ROMs is given; you will find five listings for the area \$E000-\$E7FF. Each E-ROM is for a different machine, and the following machine types were made by Commodore:

1. 40 columns, 9" screen, no CRT-controller, N-keyboard
(model numbers 4008-N, 4016-N and 4032-N).
2. 40 columns, 9" screen, no CRT-controller, B-keyboard
(model numbers 4008-B, 4016-B and 4032-B).
3. 40 columns, 12" screen, CRT-controller, N-keyboard
(model numbers 4016-N and 4032-N, nicknamed 'FAT 40').
4. 80 columns, 12" screen, CRT-controller, B-keyboard
(model numbers 8016-B and 8032-B, 8016-B not marketed in Europe).

The fifth listing for the E-ROM is for a machine type that was not manufactured by Commodore. It is for a 12 inch screen, 40 column PET ('FAT 40') converted to have an 80 column display. This type of machine is sometimes referred to as 'Graphics 80' because of its graphics keyboard. The E-ROM presented here is fully compatible with the E-ROM for the 8016/32-B.

All ROM-listings are in Carl Mosers MAE assembler format, and a true effort has been made to add useful and complete comment to EVERY instruction. As far as they were known, Commodore's and/or Microsoft's original labels were used. In those cases the original label name was not available, the label is simply the address without the preceding dollar sign.

The listing was made by:

Nico de Vries
Mari Andriessenrade 49
2907 MA Capelle aan den IJssel
The Netherlands
Phone (010)-502239

The PBE ROM special was a great help making this listing. The author wishes to thank both Dirk Groot, the author of the PBE ROM special, and Johan Smilde, publisher of the ROM special and founder of PBE, the first dutch PET/CBM users group. The book 'Programming the PET/CBM' by Raeto Collin West also provided much useful information. The cross references were made using a program written by Ton Couwenberg.

Copyrights:

All machine code and official label names:

(C) 197? Microsoft Inc. and/or Commodore Business Machines Inc.

The MAE assembler:

(C) 197? Carl Moser

All comment, descriptions of RAM use and I/O, and the cross references:

(C) 1984 Nico de Vries.

```
*****
*          *
*   The zero page
*
*****
```

The table is given in four columns:
The first is for 9" N-keyboard machines,
the second is for 9" B-keyboard machines,
the third is for 12" N-keyboard machines,
and the fourth for the 80 column machines.

The fifth column gives a sample value in hex, where appropriate. Sample values are given hi-byte first if they concern a pointer.
Pointers are given in parentheses, that is (002A) is a pointer with the lo-byte stored in \$2A, and the hi-byte stored in \$2B.

9" N	9" B	FAT40	80-Col	Sam.	Comment
0000	0000	0000	0000	4C	;JMP-instruction for USR-routine
(0001)	(0001)	(0001)	(0001)	C373	;default: ?ILLEGAL QUANTITY ERROR
0003	0003	0003	0003		;00=INPUT/READ, other is GET's character
					;work area for AND/OR, low byte
					;1st delimiter in string set up
					;odd/even flag in power calculation/EXP
0004	0004	0004	0004		;flag in tokenise routine: REM=0
					;work area for AND/OR, hi byte
0005	0005	0005	0005		;2nd delimiter in string set up
					;length of current line in tokenisation
					;tokenconverter in tokenise inputbuffer
					;inverervalue for AND/OR; OR=FF, AND=00
0006	0006	0006	0006		;number of subscripts in current array
0007	0007	0007	0007	FF	;flag:DIM<>0, LET/get value of variable=0
0008	0008	0008	0008	00	;variable type: FF=string, 00=numeric
0009	0009	0009	0009		00;numeric type: 80=integer, 00=floating
					;quote flag in LIST
					;DATA-flag in tokenise routine
					;evaluation area for DS
					;garbage collect counter
000A	000A	000A	000A		;if non zero, no int. or arrays allowed
000B	000B	000B	000B	00	;flag: 00=INPUT, 40=GET, 98=READ
000C	000C	000C	000C		;save area for flag in evaluate express.
					;3rd/4th quadrant flag in trigon. func.
000D	000D	000D	000D		;length of DS\$
(000E)	(000E)	(000E)	(000E)		;pointer to DS\$
0010	0010	0010	0010	00	;current CMD output file number
(0011)	(0011)	(0011)	(0011)		;integer address for GOTO, SYS or GOSUB
					;DOS: record number hi, lo
0013	0013	0013	0013	16	;current descr. stack pointer
(0014)	(0014)	(0014)	(0014)		;pointer to last descr. on descr. stack
0016	0016	0016	0016		;string descriptor stack
!	!	!	!		;three strings deep
001E	001E	001E	001E		;
001F	001F	001F	001F		;save area for offset in array
(001F)	(001F)	(001F)	(001F)		;temp. pointer for memory move or string
					;temporary pointer in rechain
					;pointer to backpointer in LET
					;return address save in evaluate express.
					;pointer in search for string arrays
					;pointer in reserve space for string

(0021)	(0021)	(0021)	(0021)		;	save area for backpointer in garb. coll.
0023	0023	0023	0023		;	pointer for number movements and comp.
!	!	!	!		;	FLP accu #3 (fraction only)
0026	0026	0026	0026		;	for multiply and divide
(0025)	(0025)	(0025)	(0025)		;	used to store intermediate result
0027	0027	0027	0027		;	intermediate result in array size calc.
(0028)	(0028)	(0028)	(0028)	0401	;	unused
(002A)	(002A)	(002A)	(002A)		;	pointer: start of BASIC program
(002C)	(002C)	(002C)	(002C)		;	pointer: start of variables/end of PRG
(002E)	(002E)	(002E)	(002E)		;	pointer: start of arrays/end of var.'s
(0030)	(0030)	(0030)	(0030)		;	pointer: start of free RAM/end of arrays
(0032)	(0032)	(0032)	(0032)		;	pointer: start of strings (moving down)
(0034)	(0034)	(0034)	(0034)	8000	;	utility pointer for string allocation
(0036)	(0036)	(0036)	(0036)		;	pointer: top of RAM
					;	current BASIC line number
					;	\$37=FF: direct mode
(0038)	(0038)	(0038)	(0038)		;	BASIC line number for CONT
(003A)	(003A)	(003A)	(003A)		;	CHRGET pointer for CONT
(003C)	(003C)	(003C)	(003C)		;	line number of current DATA line
(003E)	(003E)	(003E)	(003E)		;	pointer to current DATA-value
(0040)	(0040)	(0040)	(0040)		;	save area for CHRGET pointer with
					;	INPUT, READ and GET
0042	0042	0042	0042		;	current variable name, 1st character
0043	0043	0043	0043		;	current variable name, 2nd character
0044	0044	0044	0044		;	number conv: save for table index (Y)
(0044)	(0044)	(0044)	(0044)		;	pointer to current variable
0046	0046	0046	0046		;	index save in LIST
					;	AND value in WAIT
(0046)	(0046)	(0046)	(0046)		;	variable name for FOR...NEXT
					;	ptr. for number move from accu to mem.
0047	0047	0047	0047		;	EOR value in WAIT
(0048)	(0048)	(0048)	(0048)		;	saved CHRGET ptr during READ/GET/INPUT
004A	004A	004A	004A		;	Symbol check: bits 0, 1, 2 are <,>,=,
(004B)	(004B)	(004B)	(004B)		;	pointer: temp. storage for FN, DEF, TAN
					;	source pointer in garbage collect
(004D)	(004D)	(004D)	(004D)		;	pointer: temporary storage
					;	temporary pointer in reserve string sp.
004F	004F	004F	004F		;	4B-4F also used as save area f. FLP acc
0050	0050	0050	0050		;	;unused
0051	0051	0051	0051	4C	;	JMP-instruction for functions
(0052)	(0052)	(0052)	(0052)		;	pointer: address of arithmetic function
0053	0053	0053	0053		;	save area for rounding byte of FLP acc1
0054	0054	0054	0054		;	54-59 also work area for polynome result
0055	0055	0055	0055		;	flag: string to be moved in garb. coll.
(0055)	(0055)	(0055)	(0055)		;	temp. destination ptr. for memory move
					;	temp. pointer to var. in array search
(0057)	(0057)	(0057)	(0057)		;	temp. source pointer for memory move
					;	pointer in create a variable
					;	search pointer in array values
0059	0059	0059	0059		;	end of work area for polynome result
005A	005A	005A	005A		;	bit counter in array size calculation
					;	string conv: # of digits after decpoint
					;	number conv: exponent after E or:
					;	number conv: position of period
0058	0058	0058	0058		;	string conv: exponent base 10
005C	005C	005C	005C		;	number conv: correct exponent after E
(005C)	(005C)	(005C)	(005C)		;	string conv: period flag in MSbit
					;	string conv: 2nd period flag in bit 6
					;	pointer in LET or search for linenumber
					;	pointer in search for variable

(005C)	(005C)	(005C)	(005C)		;destination pointer in garbage collect
005D	005D	005D	005D		;string conv: neg. exp. flag in MSbit
005E	005E	005E	005E		;exponent of FLP-accu #1
005F	005F	005F	005F		;length of string (1st descriptor byte)
0060	0060	0060	0060		;mantissa of FLP-accu, MSB, bit7 always 1
0061	0061	0061	0061		;pointer to string text lo (2nd descr.)
0062	0062	0062	0062		;hi++ byte of integer (if 4 byte int.)
0063	0063	0063	0063		;mantissa of FLP-accu #1
0064	0064	0064	0064		;pointer to string text hi (3rd descr.)
0065	0065	0065	0065		;hi+ byte of integer (if 4 byte int.)
0066	0066	0066	0066		;mantissa of FLP-accu #1
0067	0067	0067	0067		;hi byte of integer
0068	0068	0068	0068		;sign of FLP-accu #1 in bit 7
0069	0069	0069	0069		;number of terms of polynome
006A	006A	006A	006A		;sign of string number to be converted
006B	006B	006B	006B		;overflow byte on normalizing FLP-accu #1
006C	006C	006C	006C		;exponent of FLP-accu #2
006D	006D	006D	006D		;mantissa of FLP-accu, MSB, bit7 always 1
006E	006E	006E	006E		;mantissa of FLP-accu #2
006F	006F	006F	006F		;mantissa of FLP-accu #2
0070	0070	0070	0070		;mantissa of FLP-accu #2, LSB
0071	0071	0071	0071		;sign of FLP-accu #2 in bit 7
0072	0072	0072	0072		;sign comparison of FLP-acc:0=eq. FF=opp.
0073	0073	0073	0073		;rounding byte for FLP-accu #1
0074	0074	0074	0074		;save for output index in tokenise rout.
0075	0075	0075	0075		;number conv: save for index in string
(006E)	(006E)	(006E)	(006E)		;size of current array in bytes
					;pointer to end of string in set up str.
					;pointer to constants in series eval.
0076	0076	0076	0076		;BASIC's CHRGET routine, see \$D399-\$D3B0
!	!	!	!		;
0077	0077	0077	0077	0200	;BASIC's CHRGOT entry
!	!	!	!		;CHRGET's pointer
0078	0078	0078	0078		;
0079	0079	0079	0079		;end of CHRGET/CHRGOT
0080	0080	0080	0080		;current RND number seed, exponent
0081	0081	0081	0081		;current RND number seed, MSB mantissa
0082	0082	0082	0082		;current RND number seed, mantissa
0083	0083	0083	0083		;current RND number seed, middle byte
0084	0084	0084	0084		;current RND number seed, LSB mantissa
0085	0085	0085	0085		;jiffy clock (TI) MSB
0086	0086	0086	0086		;jiffy clock (TI) middle byte
0087	0087	0087	0087		;jiffy clock (TI) LSB
(0090)	(0090)	(0090)	(0090)	E455	;IRQ RAM vector: do cassettes & keyboard
(0092)	(0092)	(0092)	(0092)	D478	;BRK RAM vector: start MLM
(0094)	(0094)	(0094)	(0094)	B3FF	;NMI RAM vector: restart BASIC
0095	0095	0095	0095		;status byte ST
0096	0096	0096	0096		;current key value from keyboard
0097	0097	0097	0097		;shiftkey indicator: 1=shift pressed
0098	0098	0098	0098		;correction clock (slows TI)
(0099)	(0099)	(0099)	(0099)		;copy of \$E812 (key image)
009A	009A	009A	009A	FF	;cassette read: accumulated speed corr.
009B	009B	009B	009B	FF	;flag: 0=LOAD, 1=VERIFY
009C	009C	009C	009C	FF	;number of keys in keyboard buffer
009D	009D	009D	009D	00	;flag: 12=RVS-on, 0=normal
009E	009E	009E	009E	00	;IEEE flag: MSbit set if byte in buffer
009F	009F	009F	009F	00	;number of inputcharacters from screen
00A0	00A0	00A0	00A0		
00A1	00A1	00A1	00A1		

00A2	00A2	00A2	00A2		;unused
00A3	00A3	00A3	00A3		;copy of linenumber on screen
00A4	00A4	00A4	00A4		;copy of column number of cursor
00A5	00A5	00A5	00A5		;IEEE byte buffer for output
00A6	00A6				;last keyboard scan index (avoid doub.)
		00A6	00A6		;copy of last key (to avoid double dec.)
00A7	00A7	00A7	00A7		;flag: 0=cursor on, others=cursor off
00A8	00A8	00A8	00A8		;cursor flash countdown
00A9	00A9	00A9	00A9		;true character under cursor
00AA	00AA	00AA	00AA		;cursor's blink phase
00AB	00AB	00AB	00AB		;flag: end of tape
00AC	00AC	00AC	00AC		;flag: input from screen (3) or keyb. (0)
00AD	00AD	00AD	00AD		;save area during cassette read
00AE	00AE	00AE	00AE		;number of open files
00AF	00AF	00AF	00AF	00	;current input device number (0=keyb.)
00B0	00B0	00B0	00B0	03	;current output device number (3=screen)
00B1	00B1	00B1	00B1		;parity bit of tape character
00B2	00B2	00B2	00B2		;cassette read flag: tape byte received
00B3	00B3	00B3	00B3		;DOS: save for current file number
00B4	00B4	00B4	00B4	01	;cassette read: 1st character of buffer
					;cassette write: save area
					;offset in given filename
					;MLM: current command number
(00B4)	(00B4)	(00B4)	(00B4)		;pointer to filename
00B5	00B5	00B5	00B5	08	;MLM: counter for printed bytes
					;MLM: save for entry title character
					;cassette: offset in name in buffer
00B6	00B6	00B6	00B6		;index in current cassette buffer
00B7	00B7	00B7	00B7		;counter for cassette
00B8	00B8	00B8	00B8		;unused
00B9	00B9	00B9	00B9		;cycle counter for cassette
00BA	00BA	00BA	00BA		;header byte counter
					;DOS: save for current output dev. number
00BB	00BB	00BB	00BB		;number of bytes in cassette buffer #1
00BC	00BC	00BC	00BC		;number of bytes in cassette buffer #2
00BD	00BD	00BD	00BD		;cycle count per sync (cassette)
00BE	00BE	00BE	00BE		;save for timing on read (cassette)
00BF	00BF	00BF	00BF		;0/1 balance (cassette read)
00C0	00C0	00C0	00C0		;number of read errors pass 1 (cassette)
00C1	00C1	00C1	00C1		;number of read errors pass 2 (cassette)
00C2	00C2	00C2	00C2		;cassette flag: 0=scan, 1-15=count,
					;40=LOAD, 80=end of tape
00C3	00C3	00C3	00C3		;number of characters before block
					;checksum over LOADED program
(00C4)	(00C4)	(00C4)	(00C4)		;pointer to start of current screenline
00C6	00C6	00C6	00C6		;column number of cursor
(00C7)	(00C7)	(00C7)	(00C7)		;pointer for LOAD, utility pointer
(00C9)	(00C9)	(00C9)	(00C9)		;end address for tape load
					;end address for SAVE
00CB	00CB	00CB	00CB		;speed correction for tape read
00CC	00CC	00CC	00CC		;elapsed time for cassette read
00CD	00CD	00CD	00CD		;flag: odd number of quotes typed
00CE	00CE	00CE	00CE		;sync established (not 0) cassette read
00CF	00CF	00CF	00CF		;flag: byte valid (tape read)
00D0	00D0	00D0	00D0		;bit errors at cassette read
00D1	00D1	00D1	00D1		;length of current file name
					;DOS: word counter in DIRECTORY
00D2	00D2	00D2	00D2		;current file number
00D3	00D3	00D3	00D3		;current secondary address ORed #\$60
00D4	00D4	00D4	00D4		;current device number

0005	0005	0005		27 ;length of current line (39 or 79)
			0005	4F ;right margin of window
(0006)	(0006)	(0006)	(0006)	;pointer: start of current tape buffer
0008	0008	0008	0008	;current screen line number
0009	0009	0009	0009	;last key input, buffer checksum, ;character save in screen input ;cassette read: received bit ;cassette write: checksum over pass
(000A)	(000A)	(000A)	(000A)	;pointer to current filename ;DOS: pointer to commandstring
000C	000C	000C	000C	;number of inserts outstanding
000D	000D	000D	000D	;current byte of cassette I/O
000E	000E	000E	000E	;cassette: pass counter, 2=header ;MLM: address wrap around flag
000F	000F	000F	000F	;cassette read: build up area for byte
00E0	00E0	00E0		;table of 25 hi- bytes of starts of ;screenlines. If MSB set, line is 40
!	!	!		;characters long, if reset, line is
!	!	!		;extension of previous line
00F8	00F8	00F8		00E0 00 ;top line number of window
				00E1 18 ;bottom line number of window
				00E2 00 ;left margin of window
				00E3 09 ;maximum length of keyboard buffer
				00E4 00 ;repeatflag 0=on, 40=off, 80=REPEAT pr. ;G-80:80=cursor keys only
				00E5 ;repeat countdown
				00E6 ;new key marker (G-80:not used)
				00E7 10 ;bell timing; 0=off
				00E8 00 ;HOME-counter
			(00E9)	E110 ;vector: screen input
			(00EB)	E20C ;vector: screen output
			00ED	;unused
			!	; !
			00F7	;unused
			00F8	;counter to speed TI by 6/5
00F9	00F9	00F9	00F9	00 ;cassette flag #1, 0=motor off
00FA	00FA	00FA	00FA	00 ;cassette flag #2, 0=motor off
(00FB)	(00FB)	(00FB)	(00FB)	;pointer to start of SAVE ;MLM: start address
(00FD)	(00FD)	(00FD)	(00FD)	;MLM: end address ;DOS: (temp.) pointer to (2nd) filename
0OFF	0OFF	0OFF	0OFF	;output area for number to string conv.

```
*****
*          *
*  Page one, stack area  *
*          *
*****
```

9" N	9" B	FAT40	80-Col	Sam.	Comment
0100	0100	0100	0100		;MLM: temp. save for hex conversion
0100	0100	0100	0100		;tape error log, holds lo and hi
!	!	!	!		;bytes of addresses whose bytes were
!	!	!	!		;wrongly read on the first pass
013E	013E	013E	013E		;end of tape error log
0100	0100	0100	0100		;work area for # to string conversions
!	!	!	!		;
0100	0100	0100	0100		;end of work area for string conversion
0140	0140	0140	0140		;theoretical bottom of BASIC stack
!	!	!	!		;
01FB	01FB	01FB	01FB		;top of BASIC and machine stack
(01FC)	(01FC)	(01FC)	(01FC)	0101	;dummy link pointer for BASIC line in
(01FE)	(01FE)	(01FE)	(01FE)		;inputbuffer
					;line number for BASIC line in input
					;buffer
 layout on stack for a FOR block:					
SP+01	SP+01	SP+01	SP+01	A1	;FOR token
(SP+02)	(SP+02)	(SP+02)	(SP+02)		;pointer to FOR variable
SP+04	SP+04	SP+04	SP+04		;STEP size, exponent
SP+05	SP+05	SP+05	SP+05		;STEP size, MSB mantissa
SP+06	SP+06	SP+06	SP+06		;STEP size, mantissa
SP+07	SP+07	SP+07	SP+07		;STEP size, mantissa
SP+08	SP+08	SP+08	SP+08		;STEP size, LSB mantissa
SP+09	SP+09	SP+09	SP+09		;sign of STEP size
SP+0A	SP+0A	SP+0A	SP+0A		;TO-value, exponent
SP+0B	SP+0B	SP+0B	SP+0B		;TO-value, MSB mantissa
SP+0C	SP+0C	SP+0C	SP+0C		;TO-value, mantissa
SP+0D	SP+0D	SP+0D	SP+0D		;TO-value, mantissa
SP+0E	SP+0E	SP+0E	SP+0E		;TO-value, LSB mantissa
(SP+0F)	(SP+0F)	(SP+0F)	(SP+0F)		;line number of FOR statement
(SP+11)	(SP+11)	(SP+11)	(SP+11)		;CHRGET pointer of FOR statement
					; (hi and lo bytes reversed)
 layout on stack of a GOSUB block:					
SP+01	SP+01	SP+01	SP+01	8D	;GOSUB token
(SP+02)	(SP+02)	(SP+02)	(SP+02)		;pointer of saved statement
(SP+04)	(SP+04)	(SP+04)	(SP+04)		;CHRGET pointer of GOSUB statement
					; (hi and lo in correct order)

```
*****
*          *
*  Page two
*          *
*****
```

9" N	9" B	FAT40	80-Col	Sam.	Comment
0200	0200	0200	0200		;BASIC's inputbuffer
(0200)	(0200)	(0200)	(0200)		;MLM: save area for PC
0202	0202	0202	0202		;MLM: save area for status register
0203	0203	0203	0203		;MLM: save area for accumulator
0204	0204	0204	0204		;MLM: save area for X-register
0205	0205	0205	0205		;MLM: save area for Y-register
0206	0206	0206	0206		;MLM: save area for stack pointer
0207	0207	0207	0207		;MLM: save area for IRQ-vector, hi
0208	0208	0208	0208		;MLM: save area for IRQ-vector, lo
0209	0209	0209	0209		;MLM: start of filename buffer
!	!	!	!		;
0218	0218	0218	0218		;MLM: end of filename buffer
!	!	!	!		;
024F	024F	024F	024F		;end of inputbuffer
0250	0250	0250	0250	00	;terminator for inputbuffer
0251	0251	0251	0251		;table of max. 10 file numbers
!	!	!	!		;
025A	025A	025A	025A		;end of table of filenumbers
025B	025B	025B	025B		;table of max. 10 corr. devicenumbers
!	!	!	!		;
0264	0264	0264	0264		;end of table of devicenumbers
0265	0265	0265	0265		;table of max. 10 corr. sec. addresses
!	!	!	!		;ORed with \$60
026E	026E	026E	026E		;end of table of sec. addresses
026F	026F	026F	026F		;start of keyboard buffer
!	!	!	!		;
0278	0278	!	!		;end of buffer with 9" machines
		????	????		;end of buffer with 12" machines
					;determined by \$E3/\$03EB
0279	0279				;unused
027A	027A	027A	027A		;start of cassette #1 buffer
					;file type, 1=program, 2=seq. file,
					;5=last file on tape
(027B)	(027B)	(027B)	(027B)		;start address for load
(027D)	(027D)	(027D)	(027D)		;end address for load
027F	027F	027F	027F		;filename (with load/save only)
!	!	!	!		;
0339	0339	0339	0339		;end of buffer for cassette #1

```
*****
*
*   Page three, 2nd cassette buffer, work area for disk commands. *
*
*****
```

9" N	9" B	FAT40	80-Col	Sam.	Comment
0339	0339	0339	0339		;end of buffer for cassette #1
033A	033A	033A	033A		;start of cassette #2 buffer
					;cassette: file type, 1=program,
					;2=seq. file, 5=last file on tape
	033A		033A		;position of TAB-stop
033A	033A	033A	033A		;DOS: byte position for RECORD
					;DOS: save for file name length
(033B)	(033B)	(033B)	(033B)		;cassette: start address for load
033B	033B	033B	033B		;DOS: source drive number
033C	033C	033C	033C		;DOS: destination drive number
(033D)	(033D)	(033D)	(033D)		;cassette: end address for load
033D	033D	033D	033D		;DOS: record length/write flag
					;DOS: file type in ASCII (R or S)
033E	033E	033E	033E		;DOS: bitmask for syntaxcheck
					; bit7= at-sign in filename
					; bit6= write to file (L or W found)
					; bit5= destination drive# found
					; bit4= source or only drive# found
					; bit3= device number found
					; bit2= file number found
					; bit1= destination filename found
					; bit0= source or only filename found
	033E		033E		;bitmask for TAB
033F	033F	033F	033F		;cassette: filename (with load/save only)
					;DOS: 1st character of disk ID
0340	0340	0340	0340		;DOS: 2nd character of disk ID
0341	0341	0341	0341		;DOS: length of command string
					;DOS: length of filename
					;DOS: length of dummy command string
0342	0342	0342	0342		;DOS: first or only filename
!	!	!	!		;
0352	0352	0352	0352		;
0353	0353	0353	0353		;DOS: full command string buffer
!	!	!	!		;
037A	037A	037A	037A		;end of DOS string buffer
037B	037B	037B	037B		;used for cassette only
!	!	!	!		;
03F8	03ED	03E8	03ED		;end of used for cassette only
		03E9			;repeat key countdown
		03EA			;repeat rate (delay between repeats)
		03EB	9		;maximum keyboard buffer length
		03EC	10		;bell timing, 0=off
		03ED			;counter to speed TI by 6/5
		03EE	0		;repeat flag: 0=on, 40=off
		03EF			;bitmask for TAB
03EE	03F0	03EE			;table of 80 TABstop bits
!	!	!			;
03F7	03F9	03F7			;end of TABstop table
03F8		03F8			;repeat key countdown
		03F9			;used for cassette only
		03F9			;repeat rate (delay between repeats)
03F9	03F9	03F9	03F9		;end of 2nd cassette buffer

(03FA)	(03FA)	(03FA)	(03FA)	D7A4 ;MLM extension vector ;default prints ? in monitor
03FC	03FC	03FC	03FC	00 ;IEEE time out defeat, no timeout ;with MSB=1
03FD	03FD	03FD	03FD	;unused
03FE	03FE	03FE	03FE	;unused
03FF	03FF	03FF	03FF	;unused

```
*****
* Pages four through 127; user RAM area
*
* All machine types use this area in the same way.
* Only the end address of this area varies with the type of
* machine, and can be $1FFF (for a XX08), $3FFF (for a XX16) or
* $7FFF (for a XX32).
*
*****
```

Address	Value	Comment
0400	00	;start of user RAM area. Always filled with a zero ;byte (dummy end of line byte).
0401		;1st BASIC program line. ;lines are stored in the following format:
0400	00	;1. Zero byte. (indicates start of line/end of previous ;line).
(0401)	XXXX	;2. Link pointer. Points to link pointer of next ;program line. ; A link pointer with the hi-byte zero indicates ; the end of the program.
(0403)	????	;3. Line number. This is the BASIC linenumber, stored ; as a two byte integer, with lo-byte first, as usual ; with the 6502 processor. Smallest line number is ; zero, the largest possible is 63999 (\$F9FF).
0404		;4. Tokenised program text. All text is stored in the ; same way that it is typed, except for the BASIC ; keywords, which are replaced by a single byte that ; has the MSB set. These bytes are called tokens.
XXXX-1	00	;5. Zero byte, this is the same byte as mentioned at ; point 1.
(XXXX)	YYYY	; Link pointer to next line's link pointer.
(XXXX+2)	????	; Line number.
XXXX+4		; Tokenised BASIC program text.
!		
YYYY-1	00	; End/start of line zero byte.
!		; This pattern is repeated for all program lines, until:
(\$2A)-2	0000	; Link pointer with hi-byte zero.

```
($2A) ;Start of variables.  
! ;Variables are stored in the following manner:  
;  
;1. FLP variable  
; +0 :1st character of name.  
; +1 :2nd character of name, or zero byte if name  
;      is only one character long.  
; +2 :exponent of FLP number.  
; +3 :mantissa, MSB. (holds sign in MSbit).  
; +4 :mantissa, upper middle.  
; +5 :mantissa, lower middle.  
; +6 :mantissa, LSB.  
;  
;2. Integer variable.  
; +0 :1st character of name, MSbit set  
; +1 :2nd character of name, MSbit set ($80 if name  
;      is only one character long).  
; +2 :MSbyte.  
; +3 :LSbyte.  
; +4 :unused.  
; +5 :unused.  
; +6 :unused.  
;  
;3. String variable.  
; +0 :1st character of name.  
; +1 :2nd character of name, MSbit set ($80 if name  
;      is only one character long).  
; +2 :Length of string (1st descriptor byte).  
; +3 :Lo-byte of pointer to stringtext (2nd desc. byte).  
; +4 :Hi-byte of pointer to stringtext (3rd desc. byte).  
; +5 :no meaning, set to zero.  
; +6 :no meaning, set to zero.  
;  
;4. FN (function) variable.  
; +0 :1st character of name, MSbit set.  
; +1 :2nd character of name, or zero byte if name  
;      is only one character long.  
; +2 :Lo-byte of pointer to function description.  
; +3 :Hi-byte of pointer to function description.  
; +4 :Lo-byte of pointer +2 of formal FLP variable.  
; +5 :Hi-byte of pointer +2 of formal FLP variable.  
; +6 :Copy of 1st character of function text.  
;  
;End of variable area.  
($2C)-1
```

```

($2C)
    ;Start of array area.
    !
    ;Arrays of variables are stored in the following manner:
    ;
    ;Layout for 1 array:
    ; +0 :1st character of name (same as for single variable)
    ; +1 :2nd character of name (same as for single variable)
    ; +2 :Lo-byte of length of total array.
    ; +3 :Hi-byte of length of total array.
    ; +4 :Number of indices.
    ; +5 :maximum value+1 for 1st index, hi
    ; +6 :maximum value+1 for 1st index, lo
    ; +7-(n-1) :same as +5 and +6 for other indices.
    ; +n-??? :elements of array,
    ;             :length for FLP element      : 5 bytes (FLP#).
    ;             :length for integer element: 2 bytes (number).
    ;             :length for string element : 3 bytes (descr.).
    ;
    ;End of array variables.
    ;

    ($2E)
        ;Start of free, unused RAM space.
        !
        ;(The difference between these two pointers
        ;is returned by the dummy function FRE(0)).
    ($30)-1
        ;End of unused RAM space.
        ;
        ;Start of string text variables.
        ;
        ;This area may contain unused gaps.
        ;
        ;Strings are stored as litteral texts, without the
        ;quotes that surround string constants. Only calculated
        ;strings are held in this area.
        ;
        ;At the end of the string itself, a pointer is stored
        ;which points back to the length of the string in the
        ;variable or array area. This pointer is called the
        ;back pointer.
        ;
        ;Garbage strings are recognized by their invalid back
        ;pointer, which has the hi byte set to $FF. The lo byte
        ;represents the length of the garbage string in this case.
        ;
        ;End of string text area.
        ;
        ;The address of this point is usually top of RAM, and
        ;can be $1FFF, $3FFF or $7FFF dependent on the memory
        ;size of the machine in question.

```

```
*****
*          *
*  Pages 128 through 131; Screen RAM          *
*          *
*****
```

Address	Value	Comment
8000		;Start of screen RAM.
!		;Characters are stored in screen RAM in the following way:
!		;1. bits 0-5 determine the character itself.
!		;2. bit 6 determines shift or nonshift.
!		;3. bit 7 indicates reverse video, or not.
83FF		;End of screen RAM on a 40 column machine.
!		;The last 24 bytes aren't used for the screen, although
!		;they are set to \$20 when screen is cleared.
87FF		;End of screen RAM for an 80 column machine.
!		;The last 48 bytes are not used.
		;On a 40 column machine the area \$8400-\$87FF is a
		;duplicate of the area \$8000-\$83FF).
		;

```
*****
*          *
*  Pages 132 through 135; Unused area          *
*          *
*****
```

Address	Value	Comment
8800		;Area not decoded, and predetermined as alternate I/O
!		;area. On 40 column models, but not on the FAT40, a jumper
!		;is present to relocate all I/O chips for this purpose.
!		;Machines with a CRT controller chip do not have this
!		;feature. (It is not supported by ROM software).
88FF		;End of alternate I/O area.
8900		;Area, not decoded, and not predetermined for a typical
!		;use.
8FFF		;End of unused area.
		;

```
*****
*          *
*  Pages 136 through 143; ROM/EPROM expansion area      *
*          *
*****
```

Address	Value	Comment
9000		;Area preserved for ROM/EPROM expansion. For this purpose
!		;an empty socket for a ROM (type 2316E or 2332) is
!		;mounted on the mother board. Alternately, an EPROM,
!		; (type 2716 or 2532) may be used.
9FFF		;End of ROM/EPROM expansion area one.
A000		;Area preserved for ROM/EPROM expansion. For this purpose
!		;an empty socket for a ROM (type 2316E or 2332) is
!		;mounted on the mother board. Alternately, an EPROM,
!		; (type 2716 or 2532) may be used.
AFFF		;End of ROM/EPROM expansion area two.
B000		;Start of ROM area.

```
*****
* Pages 176 through 255; ROM and I/O area.
* All ROM machine code is listed on the pages hereafter.
*****
*****
```

The ROM contents is given in nine separate listings:

1. Listing for the ROMs 1465-23 and 1465-19 (Approx. \$8000-\$BFFF).
2. Listing for the ROM 1465-20 (Approx. \$C000-\$CFFF).
3. Listing for the ROM 1465-21 (Approx. \$D000-\$DFFF).
4. Listing for the ROM 1447-29 (Approx. \$E000-\$E7FF, 40 col. N-keyboard).
5. Listing for the ROM 1474-02 (Approx. \$E000-\$E7FF, 40 col. B-keyboard).
6. Listing for the ROM 1498-01 (Approx. \$E000-\$E7FF, FAT 40, N-keyboard).
7. Listing for the ROM 1474-04 (Approx. \$E000-\$E7FF, 80 col. B-keyboard).
8. Listing for an E-ROM for the Graphics 80 (Approx. \$E000-\$E7FF).
9. Listing for the ROM 1465-22 (\$F000-\$FFFF).

All listings are in Carl Mosers MAE Assembler format, because they were made using this assembler. This means that CBM BASIC 4.0 can be assembled using the files listed here.

The MAE assembler that was used to make the listings was slightly modified to suit the requirements for this special purpose. The changes were:

1. The maximum label size was reduced to 6 characters.
2. Lines printed during assembly are printed without a line number.
3. The position of the comment in the line was moved forward.
4. The memory space for the labels was expanded.

These changes lead to this line format:

```
BAA2- 20 20 BB  STRDON JSR STRPRT      ;print string from ($1F)
!   !   !   !   !   !   !
+---!---!---!---!-----!--Address in hexadecimal.
.   +---!---!-----!--Opcode in hexadecimal.
.   .   +---!-----!--1st or only operand in
.   .   .   +---!-----!--hexadecimal.
.   .   .   .   +---!-----!--Hi-byte address in hexadecimal.
.   .   .   .   +---!-----!--Label, max. 6 characters.
.   .   .   .   .   +---!-----!--Mnemonic for opcode or
.   .   .   .   .   !   !   assember directive.
.   .   .   .   .   +---!-----!--Operand.
.   .   .   .   .   .   .   +---Comment, always starting with
.   .   .   .   .   .   .   .   a semicolon.
1....7..10.13..17....23..27.....40.Column number.
```

Where known, the official Microsoft/Commodore label names where used, both in the label field and in the operand field after the opcode mnemonic. In cases where the official name was unknown, the label is simply the hexadecimal address, written in lower case, without a preceeding dollar sign.

This has led to the Microsoft BASIC part of the listing having a label in every operand field, except where constants are required.

In cases where the operand field is longer than 10 characters, comment may start at a later point on the line. If so, the comment may be truncated.

```

;name BASIC4.0B.CTL
;*****
;*      B-ROM for BASIC 4.0      *
;*      *
;* Date 10-12-83 Time 00:10   *
;*      *
;* Made by:                   *
;*      *
;* Nico de Vries             *
;* Mari Andriessenrade 49    *
;* 2907 MA Capelle a/d Yssel *
;* The Netherlands            *
;*      *
;* Original CBM ROM-number  *
;*      *
;*      1465-19 and 1465-23  *
;*      *
;*****
ROMLOC .BA $8000
.FI "BASIC4.0B.M01"

```

1A49 33FE-4E47 BASIC4.0B.M01

		;name BASIC4.0B.M01
		;*****
		;* *
		;* Table of addresses of main *
		;* keyword routines. *
		;* All are -1 because they are *
		;* reached through a RTS ins- *
		;* truction. *
		;*****
B000- C7 B7	STMDSR	.SI END-1 ;address for END
B002- DD B6		.SI FOR-1 ;address for FOR
B004- 18 BD		.SI NEXT-1 ;address for NEXT
B006- 82 B8		.SI DATA-1 ;address for DATA
B008- A3 BB		.SI INPUTN-1 ;address for INPUT#
B00A- BD BB		.SI INPUT-1 ;address for INPUT
B00C- 20 C1		.SI DIM-1 ;address for DIM
B00E- 01 BC		.SI READ-1 ;address for READ
B010- 2F B9		.SI LET-1 ;address for LET
B012- 2F B8		.SI GOTO-1 ;address for GOTO
B014- 07 B8		.SI RUN-1 ;address for RUN
B016- B2 B8		.SI IF-1 ;address for IF
B018- B6 B7		.SI RESTOR-1 ;address for RESTORE
B01A- 12 B8		.SI GOSUB-1 ;address for GOSUB
B01C- 5C B8		.SI RETURN-1 ;address for RETURN
B01E- C5 B8		.SI REM-1 ;address for REM
B020- C5 B7		.SI STOP-1 ;address for STOP
B022- D5 B8		.SI ONGOTO-1 ;address for ON
B024- 62 C9		.SI FNWAIT-1 ;address for WAIT
B026- D4 FF		.SI CLOAD-1 ;address for LOAD
B028- D7 FF		.SI CSAVE-1 ;address for SAVE
B02A- DA FF		.SI CVERF-1 ;address for VERIFY
B02C- DB C4		.SI DEF-1 ;address for DEF
B02E- 59 C9		.SI POKE-i ;address for POKE
B030- 87 BA		.SI PRINTN-1 ;address for PRINT#
B032- A7 BA		.SI PRINT-1 ;address for PRINT

B034-	ED B7	.SI CONT-1	;address for CONT
B036-	2F B6	.SI LIST-1	;address for LIST
B038-	ED B5	.SI CLEAR-1	;address for CLR
B03A-	8D BA	.SI CMD-1	;address for CMD
B03C-	DD FF	.SI CSYS-1	;address for SYS
B03E-	BF FF	.SI COPEN-1	;address for OPEN
B040-	C2 FF	.SI CCLOS-1	;address for CLOSE
B042-	79 BB	.SI GET-1	;address for GET
B044-	D1 B5	.SI SCRATH-1	;address for NEW
B046-	AB B7	.SI GO-1	;address for GO
B048-	92 FF	.SI CONCAT-1	;address for CONCAT
B04A-	95 FF	.SI DOPEN-1	;address for DOPEN
B04C-	98 FF	.SI DCLOSE-1	;address for DCLOSE
B04E-	9B FF	.SI RECORD-1	;address for RECORD
B050-	9E FF	.SI FORMAT-1	;address for HEADER
B052-	A1 FF	.SI COLECT-1	;address for COLLECT
B054-	A4 FF	.SI BACKUP-1	;address for BACKUP
B056-	A7 FF	.SI DCOPY-1	;address for COPY
B058-	AA FF	.SI APPEND-1	;address for APPEND
B05A-	AD FF	.SI DSAVE-1	;address for DSAVE
B05C-	B0 FF	.SI DLOAD-1	;address for DLOAD
B05E-	B3 FF	.SI DCAT-1	;address for CATALOG
B060-	B6 FF	.SI RENAME-1	;address for RENAME
B062-	B9 FF	.SI SCRATC-1	;address for SCRATCH
B064-	B3 FF	.SI DIRCAT-1	;address for DIRECTORY

;* * ;* Table of addresses of arithmetic function routines. ;* * ;* * ;*****			
B066-	6F CD	FUNDSP .SI SGN	;address for SGN
B068-	02 CE	.SI INT	;address for INT
B06A-	8E CD	.SI ABS	;address for ABS
B06C-	00 00	USRLOC .SI USRPOK	;address for USR
B06E-	A8 C4	.SI FRE	;address for FRE
B070-	C9 C4	.SI POS	;address for POS
B072-	08 D1	.SI SQR	;address for SQR
B074-	29 D2	.SI RND	;address for RND
B076-	20 CB	.SI LOG	;address for LOG
B078-	84 D1	.SI EXP	;address for EXP
B07A-	82 D2	.SI COS	;address for COS
B07C-	89 D2	.SI SIN	;address for SIN
B07E-	D2 D2	.SI TAN	;address for TAN
B080-	2C D3	.SI ATN	;address for ATN
B082-	43 C9	.SI PEEK	;address for PEEK
B084-	B2 C8	.SI LEN	;address for LEN
B086-	8E C5	.SI STRD	;address for STR\$
B088-	E3 C8	.SI VAL	;address for VAL
B08A-	C1 C8	.SI ASC	;address for ASC
B08C-	22 C8	.SI CHRD	;address for CHR\$
B08E-	36 C8	.SI LEFTD	;address for LEFT\$
B090-	62 C8	.SI RIGHTD	;address for RIGHT\$
B092-	6D C8	.SI MIDD	;address for MID\$

;* * ;* Table of addresses and priorities of operators. ;* All are -1 because they are reached through a RTS instruction. ;* * ;* * ;*****			
B094-	79	OPTAB .BY \$79	;priority for +

B095- 9F C9	.SI FADDT-1	;address for +
B097- 79	.BY \$79	;priority for -
B098- 88 C9	.SI FSUBT-1	;address for -
B09A- 7B	.BY \$7B	;priority for *
B09B- 60 CB	.SI FMULTT-1	;address for *
B09D- 7B	.BY \$7B	;priority for /
B09E- 47 CC	.SI FDIVT-1	;address for /
BOA0- 7F	.BY \$7F	;priority for ^
BOA1- 11 D1	.SI FPWRT-1	;address for ^
BOA3- 50	.BY \$50	;priority for AND
BOA4- 88 C0	.SI ANDOP-1	;address for AND
BOA6- 46	.BY \$46	;priority for OR
BOA7- 85 C0	.SI OROP-1	;address for OR
BOA9- 7D	NEGTAB .BY \$7D	;priority for unary -
BOAA- 4A D1	.SI NEGOP-1	;address for unary -
BOAC- 5A	NOTTAB .BY \$5A	;priority for NOT
BOAD- CB BE	.SI NOTOP-1	;address for NOT
BOAF- 64	PTDORL .BY \$64	;priority for <,>
BOB0- B5 C0	.SI DOREL-1	;address for <,>

	;*	
	;* Table of keywords in ASCII *	
	;* Last letter of each keyword *	
	;* has MSB set to indicate end *	
	;* of keyword. *	
	;*	

B0B2- 45 4E C4	RESLST .BY 'EN' \$C4	;token \$80, END
B0B5- 46 4F D2	.BY 'FO' \$D2	;token \$81, FOR
B0B8- 4E 45 58	.BY 'NEX' \$D4	;token \$82, NEXT
B0BB- D4		
B0BC- 44 41 54	.BY 'DAT' \$C1	;token \$83, DATA
B0BF- C1		
B0C0- 49 4E 50	.BY 'INPUT' \$A3	;token \$84, INPUT#
B0C3- 55 54 A3		
B0C6- 49 4E 50	.BY 'INPU' \$D4	;token \$85, INPUT
B0C9- 55 D4		
B0CB- 44 49 CD	.BY 'DI' \$CD	;token \$86, DIM
B0CE- 52 45 41	.BY 'REA' \$C4	;token \$87, READ
B0D1- C4		
B0D2- 4C 45 D4	.BY 'LE' \$D4	;token \$88, LET
B0D5- 47 4F 54	.BY 'GOT' \$CF	;token \$89, GOTO
B0D8- CF		
B0D9- 52 55 CE	.BY 'RU' \$CE	;token \$8A, RUN
B0DC- 49 C6	.BY 'I' \$C6	;token \$8B, IF
B0DE- 52 45 53	.BY 'RESTOR' \$C5	;token \$8C, RESTORE
B0E1- 54 4F 52		
B0E4- C5		
B0E5- 47 4F 53	.BY 'GOSU' \$C2	;token \$8D, GOSUB
B0E8- 55 C2		
B0EA- 52 45 54	.BY 'RETUR' \$CE	;token \$8E, RETURN
B0ED- 55 52 CE		
B0F0- 52 45 CD	.BY 'RE' \$CD	;token \$8F, REM
B0F3- 53 54 4F	.BY 'STO' \$D0	;token \$90, STOP
B0F6- D0		
B0F7- 4F CE	.BY 'O' \$CE	;token \$91, ON
B0F9- 57 41 49	.BY 'WAI' \$D4	;token \$92, WAIT
B0FC- D4		
B0FD- 4C 4F 41	.BY 'LOA' \$C4	;token \$93, LOAD
B100- C4		
B101- 53 41 56	.BY 'SAV' \$C5	;token \$94, SAVE
B104- C5		
B105- 56 45 52	.BY 'VERIF' \$D9	;token \$95, VERIFY
B108- 49 46 D9		

B10B-	44 45 C6	.BY 'DE' \$C6 ;token \$96, DEF
B10E-	50 4F 4B	.BY 'POK' \$C5 ;token \$97, POKE
B11·1-	C5	
B112-	50 52 49	.BY 'PRINT' \$A3 ;token \$98, PRINT#
B115-	4E 54 A3	
B118-	50 52 49	.BY 'PRIN' \$D4 ;token \$99, PRINT
B11B-	4E D4	
B11D-	43 4F 4E	.BY 'CON' \$D4 ;token \$9A, CONT
B120-	D4	
B121-	4C 49 53	.BY 'LIS' \$D4 ;token \$9B, LIST
B124-	D4	
B125-	43 4C D2	.BY 'CL' \$D2 ;token \$9C, CLR
B128-	43 4D C4	.BY 'CM' \$C4 ;token \$9D, CMD
B12B-	53 59 D3	.BY 'SY' \$D3 ;token \$9E, SYS
B12E-	4F 50 45	.BY 'OPE' \$CE ;token \$9F, OPEN
B131-	CE	
B132-	43 4C 4F	.BY 'CLOS' \$C5 ;token \$A0, CLOSE
B135-	53 C5	
B137-	47 45 D4	.BY 'GE' \$D4 ;token \$A1, GET
B13A-	4E 45 D7	.BY 'NE' \$D7 ;token \$A2, NEW
B13D-	54 41 42	.BY 'TAB' \$A8 ;token \$A3, TAB
B140-	A8	
B141-	54 CF	.BY 'T' \$CF ;token \$A4, TO
B143-	46 CE	.BY 'F' \$CE ;token \$A5, FN
B145-	53 50 43	.BY 'SPC' \$A8 ;token \$A6, SPC
B148-	A8	
B149-	54 48 45	.BY 'THE' \$CE ;token \$A7, THEN
B14C-	CE	
B14D-	4E 4F D4	.BY 'NO' \$D4 ;token \$A8, NOT
B150-	53 54 45	.BY 'STE' \$D0 ;token \$A9, STEP
B153-	D0	
B154-	AB	.BY \$AB ;token \$AA, +
B155-	AD	.BY \$AD ;token \$AB, ,
B156-	AA	.BY \$AA ;token \$AC, *
B157-	AF	.BY \$AF ;token \$AD, /
B158-	DE	.BY \$DE ;token \$AE, ^
B159-	41 4E C4	.BY 'AN' \$C4 ;token \$AF, AND
B15C-	4F D2	.BY 'O' \$D2 ;token \$B0, OR
B15E-	BE	.BY \$BE ;token \$B1, >
B15F-	BD	.BY \$BD ;token \$B2, =
B160-	BC	.BY \$BC ;token \$B3, <
B161-	53 47 CE	.BY 'SG' \$CE ;token \$B4, SGN
B164-	49 4E D4	.BY 'IN' \$D4 ;token \$B5, INT
B167-	41 42 D3	.BY 'AB' \$D3 ;token \$B6, ABS
B16A-	55 53 D2	.BY 'US' \$D2 ;token \$B7, USR
B16D-	46 52 C5	.BY 'FR' \$C5 ;token \$B8, FRE
B170-	50 4F D3	.BY 'PO' \$D3 ;token \$B9, POS
B173-	53 51 D2	.BY 'SQ' \$D2 ;token \$BA, SQR
B176-	52 4E C4	.BY 'RN' \$C4 ;token \$BB, RND
B179-	4C 4F C7	.BY 'LO' \$C7 ;token \$BC, LOG
B17C-	45 58 D0	.BY 'EX' \$D0 ;token \$BD, EXP
B17F-	43 4F D3	.BY 'CO' \$D3 ;token \$BE, COS
B182-	53 49 CE	.BY 'SI' \$CE ;token \$BF, SIN
B185-	54 41 CE	.BY 'TA' \$CE ;token \$C0, TAN
B188-	41 54 CE	.BY 'AT' \$CE ;token \$C1, ATN
B188-	50 45 45	.BY 'PEE' \$CB ;token \$C2, PEEK
B18E-	CB	
B18F-	4C 45 CE	.BY 'LE' \$CE ;token \$C3, LEN
B192-	53 54 52	.BY 'STR' \$A4 ;token \$C4, STR\$
B195-	A4	
B196-	56 41 CC	.BY 'VA' \$CC ;token \$C5, VAL
B199-	41 53 C3	.BY 'AS' \$C3 ;token \$C6, ASC
B19C-	43 48 52	.BY 'CHR' \$A4 ;token \$C7, CHR\$
B19F-	A4	

B1A0-	4C 45 46	.BY 'LEFT' \$A4 ;token \$C8, LEFT\$
B1A3-	54 A4	.BY 'RIGHT' \$A4 ;token \$C9, RIGHT\$
B1A5-	52 49 47	.BY 'MID' \$A4 ;token \$CA, MID\$
B1A8-	48 54 A4	.BY 'G' \$CF ;token \$CB, GO
B1AB-	4D 49 44	.FI "BASIC4.OB.M02"
B1AE-	A4	
B1AF-	47 CF	

17CA 33FE-4BC8 BASIC4.OB.M02

```

;name BASIC4.OB.M02
;*****
;*
;* Table of special keywords for *
;* the CBM diskdrive. The last *
;* letter of each has the MSB set*
;* to indicate the end of keyword.*
;*
;*****

B1B1- 43 4F 4E .BY 'CONCA' $D4 ;token $CC, CONCAT
B1B4- 43 41 D4 .BY 'DOPE' $CE ;token $CD, DOPEN
B1B7- 44 4F 50 .BY 'DCLOS' $C5 ;token $CE, DCLOSE
B1BA- 45 CE .BY 'RECOR' $C4 ;token $CF, RECORD
B1C2- 52 45 43 .BY 'HEADE' $D2 ;token $D0, HEADER
B1C5- 4F 52 C4 .BY 'COLLEC' $D4 ;token $D1, COLLECT
B1C8- 48 45 41 .BY 'BACKU' $D0 ;token $D2, BACKUP
B1CB- 44 45 D2 .BY ''COP' $D9 ;token $D3, COPY
B1DE- D9 .BY 'APPEN' $C4 ;token $D4, APPEND
B1E2- 45 4E C4 .BY 'DSAV' $C5 ;token $D5, DSAVE
B1E5- 44 53 41 .BY 'DLOA' $C4 ;token $D6, DLOAD
B1E8- 56 C5 .BY 'CATALO' $C7 ;token $D7, CATALOG
B1EA- 44 4C 4F .BY 'RENAM' $C5 ;token $D8, RENAME
B1ED- 41 C4 .BY 'SCRATC' $C8 ;token $D9, SCRATCH
B1EF- 43 41 54 .BY 'DIRECTOR' $D9 ;token $DA, DIRECTORY
B1F2- 41 4C 4F
B1F5- C7
B1F6- 52 45 4E
B1F9- 41 4D C5
B1FC- 53 43 52
B1FF- 41 54 43
B202- C8
B203- 44 49 52
B206- 45 43 54
B209- 4F 52 D9
B20C- 00 .BY $00 ;end of table byte
;*****
;*
;* Table of error messages. *
;* The last letter of each *
;* has the MSB set to indicate *
;* the end of the message. *
;*
;*****

```

B200- 4E 45 58 ERRTAB .BY 'NEXT WITHOUT FO' \$D2
B210- 54 20 57
B213- 49 54 48
B216- 4F 55 54
B219- 20 46 4F
B21C- D2
B210- 53 59 4E b21d .BY 'SYNTA' \$D8
B220- 54 41 D8
B223- 52 45 54 b223 .BY 'RETURN WITHOUT GOSU' \$C2
B226- 55 52 4E
B229- 20 57 49
B22C- 54 48 4F
B22F- 55 54 20
B232- 47 4F 53
B235- 55 C2
B237- 4F 55 54 b237 .BY 'OUT OF DAT' \$C1
B23A- 20 4F 46
B23D- 20 44 41
B240- 54 C1
B242- 49 4C 4C b242 .BY 'ILLEGAL QUANTIT' \$D9
B245- 45 47 41
B248- 4C 20 51
B24B- 55 41 4E
B24E- 54 49 54
B251- D9
B252- 4F 56 45 b252 .BY 'OVERFLO' \$D7
B255- 52 46 4C
B258- 4F D7
B25A- 4F 55 54 b25a .BY 'OUT OF MEMOR' \$D9
B250- 20 4F 46
B260- 20 40 45
B263- 4D 4F 52
B266- D9
B267- 55 4E 44 b267 .BY 'UNDEF' \$27 'D STATEMEN' \$D4
B26A- 45 46 27
B260- 44 20 53
B270- 54 41 54
B273- 45 40 45
B276- 4E D4
B278- 42 41 44 b278 .BY 'BAD SUBSCRIP' \$D4
B27B- 20 53 55
B27E- 42 53 43
B281- 52 49 50
B284- D4
B285- 52 45 44 b285 .BY 'REDIM' \$27 'D ARRA' \$D9
B288- 49 40 27
B28B- 44 20 41
B28E- 52 52 41
B291- D9
B292- 44 49 56 b292 .BY 'DIVISION BY ZER' \$CF
B295- 49 53 49
B298- 4F 4E 20
B29B- 42 59 20
B29E- 5A 45 52
B2A1- CF
B2A2- 49 4C 4C b2a2 .BY 'ILLEGAL DIREC' \$D4
B2A5- 45 47 41
B2A8- 4C 20 44
B2AB- 49 52 45
B2AE- 43 D4
B2B0- 54 59 50 b2b0 .BY 'TYPE MISMATC' \$C8
B2B3- 45 20 4D
B2B6- 49 53 4D
B2B9- 41 54 43

```

B2BC- C8
B2BD- 53 54 52 b2bd .BY 'STRING TOO LON' $C7
B2CO- 49 4E 47
B2C3- 20 54 4F
B2C6- 4F 20 4C
B2C9- 4F 4E C7
B2CC- 46 49 4C b2cc .BY 'FILE DAT' $C1
B2CF- 45 20 44
B2D2- 41 54 C1
B2D5- 46 4F 52 b2d5 .BY 'FORMULA TOO COMPLE' $D8
B2D8- 4D 55 4C
B2D9- 41 20 54
B2DE- 4F 4F 20
B2E1- 43 4F 4D
B2E4- 50 4C 45
B2E7- D8
B2E8- 43 41 4E b2e8 .BY 'CAN' $27 'T CONTINU' $C5
B2EB- 27 54 20
B2EE- 43 4F 4E
B2F1- 54 49 4E
B2F4- 55 C5
B2F6- 55 4E 44 b2f6 .BY 'UNDEF' $27 'D FUNCTIO' $CE
B2F9- 45 46 27
B2FC- 44 20 46
B2FF- 55 4E 43
B302- 54 49 4F
B305- CE
                                ;*****
                                ;*
                                ;* Table of system messages.      *
                                ;* Each message is terminated   *
                                ;* with a zero byte.           *
                                ;*
                                ;*****
B306- 20 45 52 ERR .BY ' ERROR' $00
B309- 52 4F 52
B30C- 00
B30D- 20 49 4E INTXT .BY ' IN ' $00
B310- 20 00
B312- 00 52 45 REDDY .BY $00 'READY.' $00 $00
B315- 41 44 59
B318- 2E 00 00
B31B- 00 42 52 BRKTXT .BY $00 'BREAK' $00
B31E- 45 41 4B
B321- 00
                                ;*****
                                ;*
                                ;* Search for a FOR-block on the *
                                ;* stack.                      *
                                ;* If correct FOR-block found, *
                                ;* returns with Z=1.          *
                                ;*
                                ;*****
B322- BA FNDFOR TSX ;get stack in X
B323- E8 INX
B324- E8 INX
B325- E8 INX ;skip return addresses
B326- E8 INX ;of JSR-calls
B327- BD 01 01 FFLOOP LDA $0101,X ;get token from the stack
B32A- C9 81 CMP #FORTK ;if not a FOR-token
B32C- D0 21 BNE FFRTS ;exit with Z=1
B32E- A5 47 LDA *FORPNT+1 ;get pointer to variable
B330- D0 0A BNE CMPFOR ;if none
B332- BD 02 01 LDA $0102,X ;get pointer to FOR var.

```

```

B335- 85 46      STA *FORPNT      ;and set up
B337- B0 03 01    LDA $0103,X
B33A- 85 47      STA *FORPNT+1   ;variabele
B33C- DD 03 01    CMPFOR  CMP $0103,X ;if not the same
B33F- D0 07      BNE ADDFRS     ;go remove FOR-block
B341- A5 46      LDA *FORPNT     ;check lo-byte also
B343- DD 02 01    CMP $0102,X
B346- F0 07      BEQ FFRTS      ;if equal, exit with Z=1
B348- 8A          ADDFRS TXA    ;else
B349- 18          CLC
B34A- 69 12      ADC #$12      ;change stackpointer
B34C- AA          TAX
B34D- D0 D8      BNE FFLOOP    ;to reflect previous block
B34F- 60          FFRTS RTS    ;and try again
;*****
;*
;* Test available RAM space and *
;* move portion of programtext   *
;* to open up space for merged   *
;* lines.                         *
;*
;*****
B350- 20 A0 B3    BLTU JSR REASON   ;check for available space
B353- 85 2E      STA *STREND    ;store new end of
B355- 84 2F      STY *STREND+1  ;arrays pointer
B357- 38          BLTC SEC        ;prepare for subtract
B358- A5 57      LDA *HIGHTR    ;get end address lo
B35A- E5 50      SBC *LOWTR     ;calculate length
B35C- 85 1F      STA *INDEX     ;and store in temp.
B35E- A8          TAY
B35F- A5 58      LDA *HIGHTR+1  ;get end address hi
B361- E5 50      SBC *LOWTR+1  ;calculate length
B363- AA          TAX
B364- E8          INX
B365- 98          TYA
B366- F0 23      BEQ DECBLT    ;use hi-bytes only
B368- A5 57      LDA *HIGHTR    ;get end address lo
B36A- 38          SEC
B36B- E5 1F      SBC *INDEX     ;subtract length lo
B36D- 85 57      STA *HIGHTR    ;and save
B36F- B0 03      BCS BLT1      ;if page crossed
B371- C6 58      DEC *HIGHTR+1  ;adapt hi-byte
B373- 38          SEC
B374- A5 55      BLT1 LDA *HIGHDS  ;($58) points to source now
B376- E5 1F      SBC *INDEX     ;get new end address
B378- 85 55      STA *HIGHDS    ;calculate start address
B37A- B0 08      BCS MOREN1   ;save it
B37C- C6 56      DEC *HIGHDS+1  ;if page crossed
B37E- 90 04      BCC MOREN1   ;adapt hi-byte
B380- B1 57      BLTLP LDA (HIGHTR),Y ;and adapt length lo
B382- 91 55      STA (HIGHDS),Y ;get source byte
B384- 88          MOREN1 DEY    ;move it
B385- D0 F9      BNE BLTLP    ;adapt counter and move
B387- B1 57      LDA (HIGHTR),Y ;until page boundary
B389- 91 55      STA (HIGHDS),Y ;move last byte
B38B- C6 58      DECBLT DEC *HIGHTR+1 ;of page also
B38D- C6 56      DEC *HIGHDS+1  ;adapt hi-bytes
B38F- CA          DEX
B390- D0 F2      BNE MOREN1   ;and length hi
B392- 60          RTS
;*****
;*
;* Test for 2 times Accu bytes   *
;* available on the stack.       *

```

```

        ;* Report OUT OF MEMORY if not      *
        ;* enough space.                  *
        ;*                                *
        ;*****  

B393- 0A    GETSTK ASL A           ;multiply A by two
B394- 69 3E   ADC #$3E          ;add bottom of stack
B396- B0 35   BCS OMERR         ;if more than 256, error
B398- 85 1F   STA *INDEX        ;save value
B39A- BA     TSX               ;get stackpointer
B39B- E4 1F   CPX *INDEX        ;if below calculated value
B39D- 90 2E   BCC OMERR         ;also error
B39F- 60     RTS               ;else return
        ;*****  

        ;*                                *
        ;* Test for overlap of arrays    *
        ;* and strings in memory.       *
        ;* If not enough room garbage   *
        ;* is collected. If still no    *
        ;* room, OUT OF MEMORY ERROR.   *
        ;*                                *
        ;*****  

B3A0- C4 31   REASON CPY *FRETOP+1 ;if start of strings hi
B3A2- 90 28   BCC REARTS        ;higher than Y, exit
B3A4- D0 04   BNE TRYMOR        ;if equal compare lo's
B3A6- C5 30   CMP *FRETOP        ;if lo-byte higher
B3A8- 90 22   BCC REARTS        ;than A exit as well
B3AA- 48     TRYMOR PHA         ;else save A (address lo)
B3AB- A2 09   LDX #9            ;get counter
B3AD- 98     TYA               ;save
B3AE- 48     REASAV PHA         ;Y too
B3AF- B5 54   LDA *TEMPF1,X     ;get result from FLP#3
B3B1- CA     DEX               ;adapt counter
B3B2- 10 FA   BPL REASAV        ;if positive, loop
B3B4- 20 6A C6  JSR GARBA2        ;result saved, collect garbage
B3B7- A2 F7   LDX #$F7          ;move 9 bytes
B3B9- 68     REASTO PLA         ;to FLP accu1
B3BA- 95 5E   STA *FAC,X        ;and address hi
B3BC- E8     INX               ;and address lo
B3BD- 30 FA   BMI REASTO        ;if hi now lower, exit
B3BF- 68     PLA               ;if higher, error
B3C0- A8     TAY               ;if equal compare lo's
B3C1- 68     PLA               ;if higher, error
B3C2- C4 31   CPY *FRETOP+1
B3C4- 90 06   BCC REARTS
B3C6- D0 05   BNE OMERR
B3C8- C5 30   CMP *FRETOP
B3CA- B0 01   BCS OMERR
B3CC- 60     REARTS RTS         ;else exit
.FI "BASIC4.0B.M03"

```

18AE . 33FE-4CAC BASIC4.0B.M03

```

;name BASIC4.0B.M03
*****  

        ;*                                *
        ;* ?OUT OF MEMORY ERROR, then    *
        ;* warmstart BASIC.              *
        ;*                                *
        ;*****  

B3CD- A2 40   OMERR LDX #b25a-ERRTAB ;get offset in error table
        ;*****  

        ;*                                *
        ;* Fatal errors in BASIC.        *

```

```

        ;* On entry, X holds offset in    *
        ;* error message table.          *
        ;*                                *
        ;*****  

B3CF- A5 10      ERROR   LDA *CHANNL      ;if cmd output file #
B3D1- F0 07      BEQ ERRCRD      ;is not default
B3D3- 20 CC FF    JSR CLSCHN      ;restore default I/O
B3D6- A9 00      LDA #0          ;and set default
B3D8- 85 10      STA *CHANNL      ;in cmd output file #
B3DA- 20 DF BA    JSR CRDO       ;do CR(LF)
B3DD- 20 44 BB    JSR OUTQST      ;print a question mark
B3E0- B0 00 B2    GETERR  LDA ERRTAB,X  ;get character of message
B3E3- 48          PHA           ;save it
B3E4- 29 7F      AND #$7F        ;set MSB to zero
B3E6- 20 46 BB    JSR OUTDO       ;print it
B3E9- E8          INX           ;adapt offset
B3EA- 68          PLA           ;reget character
B3EB- 10 F3      BPL GETERR      ;if MSB was clear, loop
B3ED- 20 0E B6    JSR STKINI      ;else reset stack and CHRGET
B3F0- A9 06      LDA #L,ERR      ;get pointer
B3F2- A0 B3      LDY #H,ERR      ;to ERROR message
B3F4- 20 10 BB    ERRFIN  JSR STROUT     ;print it
B3F7- A4 37      LDY *CURLIN+1   ;get current line# hi
B3F9- C8          INY           ;if FF (direct mode)
B3FA- F0 03      BEQ READY       ;warmstart BASIC
B3FC- 20 78 CF    JSR INPRT       ;else print 'IN' + line#
;*****  

;*                                *
;* BASIC warmstart, prints READY *
;* and waits for input.          *
;*                                *
;*****  

B3FF- A9 12      READY   LDA #L,REDDY    ;get pointer
B401- A0 B3      LDY #H,REDDY    ;to READY. message
B403- 20 10 BB    JSR STROUT      ;print it
B406- 20 E2 B4    MAIN   JSR INLIN       ;and get keyboard input
B409- 86 77      STX *TXTPTR      ;set up CHRGET
B40B- 84 78      STY *TXTPTR+1   ;get next character
B40D- 20 70 00    JSR CHRGET      ;save it
B410- AA          TAX            ;if end of statement, get another
B411- F0 F3      BEQ MAIN        ;flag direct mode
B413- A2 FF      LDX #$FF        ;tokenise the input buffer
B415- 86 37      STX *CURLIN+1   ;and execute the code
B417- 90 06      BCC MAIN1       ;if statement has no #
B419- 20 FB B4    JSR CRUNCH      ;look for line# in prg
B41C- 4C 7C B7    JMP GONE        ;if it does not exist, go add
;*****  

;*                                *
;* Tokenise BASIC programline in *
;* the inputbuffer (programline  *
;* in input buffer has a line   *
;* number).                      *
;*                                *
;*****  

B41F- 20 F6 B8    MAIN1   JSR LINGET      ;get decimal number in ($11)
B422- 20 FB B4    JSR CRUNCH      ;tokenise rest of buffer
B425- 84 05      STY *COUNT       ;save length of line
B427- 20 A3 B5    JSR FNDLIN      ;look for line# in prg
B42A- 90 44      BCC NODEL       ;Delete old line with line#
;* found in inputline.          *
;*                                *
;*****  


```

```

;*****
B42C- A0 01 LDY #1 ;line exists, fit it in
B42E- B1 5C LDA (LOWTR),Y ;get link byte hi
B430- 85 20 STA *INDEX1+1 ;set as src. pptr hi
B432- A5 2A LDA *VARTAB ;get start of variables lo
B434- 85 1F STA *INDEX1 ;set as src. pptr lo
B436- A5 5D LDA *LOWTR+1 ;get address of line hi
B438- 85 22 STA *INDEX2+1 ;set as dest. pptr hi
B43A- A5 5C LDA *LOWTR ;get address of line lo
B43C- 88 DEY
B43D- F1 5C SBC (LOWTR),Y ;calculate neg. line length
B43F- 18 CLC ;(must be less than 256, else error)
B440- 65 2A ADC *VARTAB ;correct end of prg pptr lo
B442- 85 2A STA *VARTAB ;and save pptr
B444- 85 21 STA *INDEX2 ;and set dest. pptr lo
B446- A5 2B LDA *VARTAB+1 ;get end of prg hi
B448- 69 FF ADC #$FF ;add carry
B44A- 85 2B STA *VARTAB+1 ;and complete pptr
B44C- E5 5D SBC *LOWTR+1 ;calculate length hi
B44E- AA TAX ;save it (X holds # of pages to move)
B44F- 38 SEC ;prepare for subtract
B450- A5 5C LDA *LOWTR ;get line address lo
B452- E5 2A SBC *VARTAB ;calculate offset for pptrs
B454- A8 TAY ;save in Y
B455- B0 03 BCS QDECT1 ;if negative
B457- E8 INX ;adapt # of pages to move
B458- C6 22 DEC *INDEX2+1 ;and dest. pointer
B45A- 18 QDECT1 CLC ;prepare for add
B45B- 65.1F ADC *INDEX1 ;check carry in dest.
B45D- 90 03 BCC MLOOP ;if carry
B45F- C6 20 DEC *INDEX1+1 ;adapt dest. pointer
B461- 18 CLC
B462- B1 1F MLOOP LDA (INDEX1),Y ;then delete
B464- 91 21 STA (INDEX2),Y ;old line
B466- C8 INY ;adapt byte counter
B467- D0 F9 BNE MLOOP ;if on page boundary
B469- E6 20 INC *INDEX1+1 ;adapt hi bytes
B46B- E6 22 INC *INDEX2+1
B46D- CA DEX ;and page counter
B46E- D0 F2 BNE MLOOP ;if not all done, loop
;*****
;*          *
;* Insert input line in program  *
;* according to its line number. *
;*          *
;*****


B470- 20 E9 B5 NODEL JSR RUNC ;perform CLR (set pointers)
B473- 20 B6 B4 JSR LNKPRG ;rechain (because of possible deletion)
B476- AD 00 02 LDA BUF ;if empty line
B479- F0 8B BEQ MAIN ;now ready, go get new input
B47B- 18 CLC ;prepare for add
B47C- A5 2A LDA *VARTAB ;get lo byte of pointer to end of prg
B47E- 85 57 STA *HIGHTR ;move to temp. pointer
B480- 65 05 ADC *COUNT ;add line length
B482- 85 55 STA *HIGHDS ;save as dest. pointer lo
B484- A4 28 LDY *VARTAB+1 ;get hi byte of end of prg
B486- 84 58 STY *HIGHTR+1 ;complete source pointer
B488- 90 01 BCC NODELC ;if carry
B48A- C8 . INY ;adapt hi byte
B48B- 84 56 NODELC STY *HIGHDS+1 ;and complete dest. pointer
B48D- 20 50 B3 JSR BLTU ;open up space in prg
B490- A5 11 LDA *LINNUM ;get line number lo
B492- A4 12 LDY *LINNUM+1 ;and hi
B494- 8D FE 01 STA BUF-2 ;precede input line

```

```

B497- 8C FF 01      STY BUF-1      ;with it
B49A- A5 2E          LDA *STREND    ;get new end of prg pointer lo
B49C- A4 2F          LDY *STREND+1  ;and hi
B49E- 85 2A          STA *VARTAB   ;store it
B4A0- 84 2B          STY *VARTAB+1
B4A2- A4 05          LDY *COUNT    ;get length of new line
B4A4- 88             DEY          ;adapt to counter
B4A5- B9 FC 01      STOLOP LDA BUF-4,Y ;get byte of input buffer
B4A8- 91 5C          STA (LOWTR),Y ;move to program
B4AA- 88             DEY          ;adapt counter
B4AB- 10 F8          BPL STOLOP   ;until whole line moved
;*****
;*          *
;* Reset BASIC to start, do CLR, *
;* and warmstart BASIC.           *
;*          *
;*****
B4AD- 20 E9 B5      FINI   JSR RUNC      ;perform CLR (set all pointers)
B4B0- 20 B6 B4      JSR LNKPRG    ;rechain all lines
B4B3- 4C 06 B4      JMP MAIN     ;and go get new input
;*****
;*          *
;* Rechain whole BASIC program  *
;* (compute all link pointers). *
;*          *
;*****
B4B6- A5 28          LNKPRG LDA *TXTTAB    ;get start of prg lo
B4B8- A4 29          LDY *TXTTAB+1  ;and hi
B4BA- 85 1F          STA *INDEX    ;move to temp. pointer
B4BC- 84 20          STY *INDEX+1
B4BE- 18             CLC          ;prepare for add
B4BF- A0 01          CHEAD  LDY #1       ;point to 1st link
B4C1- B1 1F          LDA (INDEX),Y ;get link hi
B4C3- F0 1C          BEQ LNKRTS   ;if end of prg, exit
B4C5- A0 04          LDY #4       ;point to
B4C7- C8             CZLOOP INY      ;tokenised statements
B4C8- B1 1F          LDA (INDEX),Y ;get byte from line
B4CA- D0 FB          BNE CZLOOP   ;if not end of line, loop
B4CC- C8             INY          ;if end of line, get length
B4CD- 98             TYA          ;in accu
B4CE- 65 1F          ADC *INDEX    ;add to start of line
B4D0- AA             TAX          ;save in X
B4D1- A0 00          LDY #0       ;point to link lo
B4D3- 91 1F          STA (INDEX),Y ;set up link lo
B4D5- 98             TYA          ;clear accu
B4D6- 65 20          ADC *INDEX+1 ;and add carry
B4D8- C8             INY          ;point to link hi
B4D9- 91 1F          STA (INDEX),Y ;set it up
B4DB- 86 1F          STX *INDEX    ;set pointer
B4DD- 85 20          STA *INDEX+1 ;to next line
B4DF- 90 DE          BCC CHEAD   ;and do next line (branch always)
B4E1- 60             LNKRTS RTS
.FI "BASIC4.OB.M04"

```

1800 33FE-4CCE BASIC4.OB.M04

```

;name BASIC4.OB.M04
;*****
;*          *
;* Get input in input buffer. *
;*          *
;*****
B4E2- A2 00          INLIN LDX #0       ;start with 1st character

```

```

B4E4- 20 CF FF  INLINC JSR INCHR      ;get a character
B4E7- C9 0D          CMP #$0D        ;if RETURN
B4E9- F0 0D          BEQ FININ1     ;add a null byte
B4EB- 9D 00 02      STA BUF,X      ;store in input buffer
B4EE- E8             INX            ;adapt counter
B4EF- E0 51          CPX #81        ;if buffer not full
B4F1- D0 F1          BNE INLINC     ;go loop
B4F3- A2 20          LDX #b2bd-ERRTAB ;else point to
B4F5- 4C CF B3      JMP ERROR       ;?STRING TOO LONG ERROR
B4F8- 4C D2 BA  FININ1 JMP FININL    ;add a null byte, do CR(LF)
;*****  

;*  

;* Tokenise the input buffer. *  

;*  

;*****  

B4FB- A6 77          CRUNCH LDX *TXTPTR   ;get index in buffer
B4FD- A0 04          LDY #4          ;and initial output index
B4FF- 84 09          STY *DORES     ;flag no DATA
B501- B0 00 02  KLOOP LDA BUF,X      ;get a character
B504- 10 07          BPL CMPSPC     ;if shifted
B506- C9 FF          CMP #PI         ;check for PI
B508- F0 4A          BEQ STUFFH     ;if PI do not tokenise
B50A- E8             INX            ;adapt pointer in buffer
B50B- D0 F4          BNE KLOOP     ;and get next character
B50D- C9 20          CMPSPC CMP #INDEX+1 ;not shifted, if blank
B50F- F0 43          BEQ STUFFH     ;do not tokenise
B511- 85 04          STA *ENDCHR    ;else save character
B513- C9 22          CMP #'''        ;if quote
B515- F0 62          BEQ STRNG      ;go move string (do not tokenise)
B517- 24 09          BIT *DORES     ;if in DATA statement
B519- 70 39          BVS STUFFH     ;go copy buffer (do not tokenise)
B51B- C9 3F          CMP #'?'       ;if ?
B51D- D0 04          BNE KLOOP1    ;replace with token for PRINT
B51F- A9 99          LDA #PRINTK    ;and do next character
B521- D0 31          BNE STUFFH     ;if numeral
B523- C9 30          KLOOP1 CMP #'0'    ;semicolon or colon
B525- 90 04          BCC MUSTCR    ;go, do not tokenise
B527- C9 3C          CMP #'<'      ;save output index
B529- 90 29          BCC STUFFH     ;get initial token number
B52B- 84 6E          MUSTCR STY *BUFPTR ;save it
B52D- A0 00          LDY #0          ;adapt CHRGET's pointer
B52F- 84 05          STY *COUNT     ;point (INDEX) to start
B531- 86 77          STX *TXTPTR    ;of keyword table
B533- A9 B0          LDA #H,RESLST  ;adapt input index
B535- 85 20          STA *INDEX+1  ;and keyword pointer
B537- A9 B2          LDA #L,RESLST
B539- 85 1F          STA *INDEX
B53B- D0 07          BNE RESCON
B53D- E8             RESER INX
B53E- E6 1F          INC *INDEX
B540- D0 02          BNE RESCON
B542- E6 20          INC *INDEX+1
B544- B0 00 02  RESCON LDA BUF,X  ;get character of keyword
B547- 38             SEC            ;compare with character
B548- F1 1F          SBC (INDEX),Y ;of keyword in table
B54A- F0 F1          BEQ RESER     ;if equal compare next char.
B54C- C9 80          CMP #$80        ;if only MSB was different
B54E- D0 30          BNE NTHIS
B550- 05 05.         ORA *COUNT    ;convert to token
B552- A4 6E          GETBPT LDY *BUFPTR ;get output index back
B554- E8             STUFFH INX    ;adapt input index
B555- C8             INY            ;adapt output index
B556- 99 FB 01      STA BUF-5,Y  ;store token in buffer
B559- B9 FB 01      LDA BUF-5,Y  ;get it back

```

```

B55C- F0 3B      BEQ CRDONE      ;if end of buffer, exit
B55E- 38          SEC            ;if not
B55F- E9 3A      SBC #'':       ;if colon,
B561- F0 04      BEQ COLIS      ;flag no DATA
B563- C9 49      CMP #$49       ;if token was $83 (DATA)
B565- D0 02      BNE NODATT    ;flag DATA
B567- 85 09      COLIS STA *DORES
B569- 38          NODATT SEC    ;prepare for subtract
B56A- E9 55      SBC #$55       ;if token was not $8F (REM)
B56C- D0 93      BNE KLOOP     ;do next input character
B56E- 85 04      STA *ENDCHR   ;if REM flag it
B570- B0 00 02      LDA BUF,X   ;get next input character
B573- F0 DF      BEQ STUFFH    ;if end of buffer,
B575- C5 04      CMP *ENDCHR   ;or terminator ("")
B577- F0 DB      BEQ STUFFH    ;move buffer forward
B579- C8          STRNG INY    ;else move
B57A- 99 FB 01      STA BUF-5,Y ;character forward
B57D- E8          INX            ;adapt output index
B57E- D0 F0      BNE STR1      ;and loop
B580- A6 77      NTHIS LDX *TXTPTR ;if not keyword, get input index
B582- E6 05      INC *COUNT    ;adapt token counter
B584- B1 1F      NTHIS1 LDA (INDEX),Y ;get keyword character
B586- 08          PHP            ;save flags
B587- E6 1F      INC *INDEX    ;adapt pointer
B589- D0 02      BNE NTHIS2    ;get flags back, if MSB clear
B58B- E6 20      INC *INDEX+1  ;and try next keyw. char.
B58D- 28          NTHIS2 PLP    ;get char. of next keyw.
B58E- 10 F4      BPL NTHIS1    ;if not at end of table try next keywor
B590- B1.1F      LDA (INDEX),Y ;get input character again
B592- D0 B0      BNE RESCON    ;and loop (branch always)
B594- B0 00 02      LDA BUF,X   ;end of buffer reached, set end
B597- 10 B9      BPL GETBPT    ;point CHRGET before
B599- 99 FD 01      CRDONE STA BUF-3,Y ;input buffer
B59C- C6 78      DEC *TXTPTR+1
B59E- A9 FF      LDA #L,BUF-1
B5A0- 85 77      STA *TXTPTR
B5A2- 60          RTS            ;and exit
;*****
;*                                     *
;* Search BASIC for line number. *
;*                                     *
;* If number found, carry is set *
;* and ($5C) points to line link *
;* On entry ($11) holds search   *
;* number.                         *
;*                                     *
;*****



B5A3- A5 28      FNDLIN LDA *TXTTAB   ;get start of program
B5A5- A6 29      LDX *TXTTAB+1 ;in A and X
B5A7- A0 01      FNDLNC LDY #1    ;point to link hi
B5A9- 85 5C      STA *LOWTR    ;set up pointer
B5AB- 86 5D      STX *LOWTR+1 ;for search
B5AD- B1 5C      LDA (LOWTR),Y ;get link hi
B5AF- F0 1F      BEQ FLINRT   ;if at end, exit w. CLC
B5B1- C8          INY            ;adapt index in line
B5B2- C8          INY            ;to point to line# hi
B5B3- A5 12      LDA *LINNUM+1 ;get search value hi
B5B5- D1 5C      CMP (LOWTR),Y ;compare it
B5B7- 90 18      BCC FLNRTS   ;if search value lower, exit
B5B9- F0 03      BEQ FNDL01   ;if same compare lo's
B5BB- 88          DEY            ;if search value higher
B5BC- D0 09      BNE AFFRTS   ;test previous line
B5BE- A5 11      FNDL01 LDA *LINNUM ;compare lo line#
B5C0- 88          DEY            ;adapt index to point to lo

```

```

B5C1- D1 5C      CMP (LOWTR),Y
B5C3- 90 0C      BCC FLNRTS      ;if search value lower, exit
B5C5- F0 0A      BEQ FLNRTS      ;if same, exit with SEC
B5C7- 88         AFFRTS DEY      ;if higher,
B5C8- B1 5C      LDA (LOWTR),Y  ;get link hi
B5CA- AA         TAX             ;in X
B5CB- 88         DEY             ;point to link lo
B5CC- B1 5C      LDA (LOWTR),Y  ;get it
B5CE- B0 D7      BCS FNDLNC     ;store pointer and loop (always)
B5D0- 18         FLINRT CLC     ;clear carry (line not found)
B5D1- 60         FLNRTS RTS     ;and exit
                                ;*****
                                ;*
                                ;* Perform NEW.          *
                                ;*
                                ;*****
B5D2- D0 FD      SCRATH BNE FLNRTS ;if parameters, SYNTAX ERROR
B5D4- A9 00      SCRTCH LDA #0   ;get end of prg marker
B5D6- A8         TAY             ;and index
B5D7- 91 28      STA (TXTTAB),Y ;set 1st link lo to zero
B5D9- C8         INY             ;point to link hi
B5DA- 91 28      STA (TXTTAB),Y ;set that to zero too
B5DC- A5 28      LDA *TXTTAB    ;get start of prg
B5DE- 18         CLC             ;calculate pointer for
B5DF- 69 02      ADC #2         ;start of variables
B5E1- 85 2A      STA *VARTAB   ;and store it
B5E3- A5 29      LDA *TXTTAB+1 ;get hi byte
B5E5- 69 00      ADC #0         ;add carry
B5E7- 85 28      STA *VARTAB+1 ;and store as.well
B5E9- 20 22 B6    RUNC   JSR STXTPT ;point CHRGET to start of prg
B5EC- A9 00      LDA #0         ;set Z-flag
                                ;*****
                                ;*
                                ;* Perform CLR.          *
                                ;*
                                ;*****
B5EE- D0 31      CLEAR  BNE STKRTS ;if parameters, SYNTAX ERROR
B5F0- A5 34      CLEARC LDA *MEMSIZ ;get top of RAM lo
B5F2- A4 35      LDY *MEMSIZ+1 ;and hi
B5F4- 85 30      STA *FRETOP   ;set start of strings
B5F6- 84 31      STY *FRETOP+1 ;to that (no strings left)
B5F8- A9 00      LDA #0         ;set length of DS$
B5FA- 85 00      STA *DSDESC   ;to zero
B5FC- 20 E7 FF    JSR CCALL     ;close all files
B5FF- A5 2A      LDA *VARTAB   ;get start of variables
B601- A4 2B      LDY *VARTAB+1 ;in A and Y
B603- 85 2C      STA *ARYTAB   ;set in start of arrays
B605- 84 2D      STY *ARYTAB+1 ;(clear variables)
B607- 85 2E      STA *STREND   ;and in end of arrays
B609- 84 2F      STY *STREND+1 ;(clear arrays)
B60B- 20 B7 B7    FLOAD  JSR RESTOR ;perform RESTORE
B60E- A2 16.      STKINI LDX #TEMPST ;reset string descriptor stack
B610- 86 13      STX *TEMPPT
B612- 68         PLA             ;get return address
B613- A8         TAY             ;in Y
B614- 68         PLA             ;and A
B615- A2 FA      LDX #L,STKEND-1 ;get top of stack
B617- 9A         TXS             ;load stack pointer
B618- 48         PHA             ;and move return address
B619- 98         TYA             ;back
B61A- 48         PHA
B61B- A9 00      LDA #0         ;clear:
B61D- 85 3B      STA *OLDTXT+1 ;pointer for CONT
B61F- 85 0A      STA *SUBFLG   ;and flag for arrays

```

B621- 60 STKRTS RTS ;and exit
.FI "BASIC4.0B.MOS"

1909 33FE-4007 BASIC4.0B.MOS

```
;name BASIC4.0B.MOS
;*****
;*
;* Reset CHRGET to start of *
;* program-1. *
;*
;*****
B622- 18 STXTPT CLC ;prepare for add
B623- A5 28 LDA *TXTTAB ;get start of program lo
B625- 69 FF ADC #$FF ;decrement it
B627- 85 77 STA *TXTPTR ;store in CHRGET's pointer lo
B629- A5 29 LDA *TXTTAB+1 ;get start of program hi
B62B- 69 FF ADC #$FF ;decrement it
B62D- 85 78 STA *TXTPTR+1 ;store in CHRGET's pointer hi
B62F- 60 RTS ;and exit
;*****
;*
;* Perform LIST. *
;*
;*****
B630- 90 06 LIST BCC GOLST ;parameter be a number
B632- F0 04 BEQ GOLST ;or no parameter
B634- C9 AB CMP #MINUTK ;or minus token
B636- D0 E9 BNE STKRTS ;if none of these, SYNTAX ERROR
B638- 20 F6 B8 GOLST JSR LINGET ;else get number in ($11)
B63B- 20 A3 B5 JSR FNDLIN ;search for that line number
B63E- 20 76 00 JSR CHRGOT ;reget current character
B641- F0 DC BEQ LSTEND ;if none, go list line
B643- C9 AB CMP #MINUTK ;if not minus token,
B645- D0 8A BNE FLNRTS ;give SYNTAX ERROR (why not STKRTS?)
B647- 20 70 00 JSR CHRGOT ;else get next character
B64A- 20 F6 B8 JSR LINGET ;get last line number in ($11)
B64D- D0 82 BNE FLNRTS ;if more parameters, SYNTAX ERROR
B64F- 68 LSTEND PLA
B650- 68 PLA ;pop return address
B651- A5 11 LDA *LINNUM ;if last parameter
B653- 05 12 ORA *LINNUM+1 ;zero
B655- D0 06 BNE LIST4
B657- A9 FF LDA #$FF ;set 65535
B659- 85 11 STA *LINNUM ;as last line to be
B65B- 85 12 STA *LINNUM+1 ;listed (LIST-till end of prg)
B65D- A0 01 LIST4 LDY #1 ;point to link hi
B65F- 84 09 STY *DORES ;clear quote flag
B661- B1 5C LDA (LOWTR),Y ;get link hi
B663- F0 43 BEQ GRODY ;if zero, stop listing
B665- 20 E1 FF JSR ISCNTC ;check for STOP key'
B668- 20 DF BA JSR CRDO ;do CR(LF)
B66B- C8 INY ;adapt index
B66C- B1 5C LDA (LOWTR),Y ;get line# lo
B66E- AA TAX ;save in X
B66F- C8 INY ;point to line# hi
B670- B1 5C LDA (LOWTR),Y ;get it
B672- C5 12 CMP *LINNUM+1 ;if past
B674- D0 04 BNE TSTDUN ;end line number
B676- E4 11 CPX *LINNUM
B678- F0 02 BEQ TYPLIN
B67A- B0 2C TSTDUN BCS GRODY ;stop listing
B67C- 84 46 TYPLIN STY *LSTPNT ;else save index
```

```

B67E- 20 83 CF      JSR LINPRT      ;print line# in decimal
B681- A9 20          LDA #$20        ;get a blank
B683- A4 46          PRIT4         ;and index
B685- 29 7F          LDY *LSTPNT    ;clear MSB (convert to unshift)
B687- 20 46 BB       PLOOP        ;print the character
B68A- C9 22          CMP #'"'      ;if it was a quote
B68C- D0 06          BNE PLOOP1    ;get quote flag
B68E- A5 09          LDA *DORES    ;invert it (flag quote mode)
B690- 49 FF          EOR #$FF      ;and store it
B692- 85 09          STA *DORES    ;adapt index
B694- C8             PLOOP1       ;if index >255, SYNTAX ERROR
B695- F0 11          BEQ GRODY     ;get character
B697- B1 5C          LDA (LOWTR),Y ;if not end of line, go list
B699- D0 10          BNE QPLOP     ;if end of line, zero Y
B69B- A8             TAY           ;get link lo
B69C- B1 5C          LDA (LOWTR),Y ;in X
B69E- AA             TAX            ;and
B69F- C8             INY            ;link hi
B6A0- B1 5C          LDA (LOWTR),Y ;set up pointer
B6A2- 86 5C          STX *LOWTR    ;for next line
B6A4- 85 50          STA *LOWTR+1 ;and do next line
B6A6- D0 B5          BNE LIST4    ;restart BASIC
B6A8- 4C FF B3       GRODY        ;if msb clear, go print
B6AB- 10 DA          QPLOP        ;char. is token or shifted, if PI
B6AD- C9 FF          CMP #PI        ;go print as PI
B6AF- F0 06          BEQ PLOOP    ;get quoteflag
B6B1- 24 09          BIT *DORES    ;if in quotes, go print
B6B3- 30 D2          BMI PLOOP    ;character is token, move to X
B6B5- AA             TAX            ;save index
B6B6- 84 46          STY *LSTPNT   ;set up
B6B8- A0 B0          LDY #H,RESLST ;pointer
B6BA- 84 20          STY *INDEX+1 ;to
B6BC- A0 B2          LDY #L,RESLST ;keyword table
B6BE- 84 1F          STY *INDEX    ;zero index in table
B6C0- A0 00          LDY #0        ;clear MSB, if 1st token (END)
B6C2- 0A             ASL A        ;go print keyword
B6C3- F0 10          BEQ PRIT3B   ;adapt token counter
B6C5- CA             RESRCH       ;if >$7F, go print keyword
B6C6- 10 0C          BPL PRIT3    ;else adapt pointer
B6C8- E6 1F          RESCR1      INC *INDEX
B6CA- D0 02          BNE RESCR2   ;get character of keyword
B6CC- E6 20          INC *INDEX+1 ;if not shifted, get next
B6CE- B1 1F          RESCR2      BPL RESCR1
B6D0- 10 F6          BPL RESCR1   ;if shifted do next keyword
B6D2- 30 F1          BMI RESRCH   ;adapt index
B6D4- C8             PRIT3       INY
B6D5- B1 1F          PRIT3B      LDA (INDEX),Y
B6D7- 30 AA          BMI PRIT4    ;get next character
B6D9- 20 46 BB       JSR OUTDO    ;if last go print and loop
B6DC- D0 F6          BNE PRIT3    ;else print
                                         ;and loop until last char. of keyw.
                                         ;*****
                                         ;*
                                         ;* Perform FOR. *
                                         ;*
                                         ;*****
B6DE- A9 80          FOR          LDA #$80        ;no integer or array
B6EO- 85 0A          STA *SUBFLG   ;variable allowed
B6E2- 20 30 B9       JSR LET       ;perform LET
B6E5- 20 22 B3       JSR FNDFOR   ;check block on stack
B6E8- D0 05          BNE NOTOL    ;if not there, set it up
B6EA- 8A             TXA           ;else get stack pointer
B6EB- 69 0F          ADC #$0F      ;set pointer below that block
B6ED- AA             TAX           ;load stack pointer
B6EE- 9A             TXS           ;with that value

```

B6EF- 68	NOTOL	PLA	;remove return address
B6F0- 68		PLA	;from stack
B6F1- A9 09		LDA #9	;18 parameters on stack wanted
B6F3- 20 93 B3		JSR GETSTK	;check for stack space
B6F6- 20 91 B8		JSR DATAN	;get offset to end of statement
B6F9- 18		CLC	;prepare for add
B6FA- 98		TYA	;get offset lo
B6FB- 65 77		ADC *TXTPTR	;add chrget's pointer
B6FD- 48		PHA	;save on stack
B6FE- A5 78		LDA *TXTPTR+1	;same for pointer hi
B700- 69 00		ADC #0	;add carry
B702- 48		PHA	;save (lo-hi reversed on stack!)
B703- A5 37		LDA *CURLIN+1	;get current line# hi
B705- 48		PHA	;save it
B706- A5 36		LDA *CURLIN	;get current line# lo
B708- 48		PHA	;save it
B709- A9 A4		LDA #TOTK	;get token for TO
B70B- 20 F7 BE		JSR SYNCNR	;check if next
B70E- 20 87 BD		JSR CHKNUM	;if so, check type of #
B711- 20 84 BD		JSR FRMNUM	;if numeric, get number
B714- A5 63		LDA *FACSGN	;get sign of FLP-accu
B716- 09 7F		ORA #\$7F	;get MSB
B718- 25 5F		AND *FACHO	;add to mantissa MSB
B71A- 85 5F		STA *FACHO	
B71C- A9 27		LDA #L,LDFONE	
B71E- A0 B7		LDY #H,LDFONE	
B720- 85 1F		STA *INDEX	
B722- 84 20		STY *INDEX+1	
B724- 4C 41 BE		JMP FORPSH	
B727- A9 F2	LDFONE	LDA #L,FONE	;round FLP accu1 and push on stack
B729- A0 CA		LDY #H,FONE	;continues here,
B72B- 20 D8 CC		JSR MOVFM	;get pointer to FLP# 1.
B72E- 20 76 00		JSR CHRGOT	
B731- C9 A9		CMP #STEPTK	
B733- D0 06		BNE ONEON	
B735- 20 70 00		JSR CHRGET	
B738- 20 84 BD		JSR FRMNUM	
B73B- 20 61 CD	ONEON	JSR SIGN	
B73E- 20 32 BE		JSR PUSHF	
B741- A5 47		LDA *FORPNT+1	
B743- 48		PHA	
B744- A5 46		LDA *FORPNT	
B746- 48		PHA	
B747- A9 81		LDA #FORTK	
B749- 48		PHA	

		/*	*
		/* Start BASIC: execute next	*
		/* statement.	*
		/*	*

B74A- 20 E1 FF	NEWSSTT	JSR ISCNTC	;check for STOP key
B74D- A5 77		LDA *TXTPTR	;get CHRGET's pointer
B74F- A4 78		LDY *TXTPTR+1	
B751- C0 02		CPY #H,BUF	
B753- F0 04		BEQ DIRCON	
B755- 85 3A		STA *OLDTXT	
B757- 84 3B		STY *OLDTXT+1	
B759- A0 00	DIRCON	LDY #0	
B75B- B1 77		LDA (TXTPTR),Y	
B75D- D0 46		BNE MORSTS	
B75F- A0 02		LDY #2	
B761- B1 77		LDA (TXTPTR),Y	
B763- 18		CLC	

```

B764- D0 03      BNE DIRCN1      ;go execute next statement in prg
B766- 4C E2 B7    JMP ENDCON      ;else perform END
B769- C8          DIRCN1 INY      ;point to line# lo
B76A- B1 77      LDA (TXTPTR),Y   ;get it
B76C- 85 36      STA *CURLIN     ;store it
B76E- C8          INY
B76F- B1 77      LDA (TXTPTR),Y   ;do same for hi
B771- 85 37      STA *CURLIN+1
B773- 98          TYA            ;get current offset
B774- 65 77      ADC *TXTPTR     ;adapt CHRGET's pointer
B776- 85 77      STA *TXTPTR     ;to point to 1st token of line
B778- 90 02      BCC GONE        ;if page crossed
B77A- E6 78      INC *TXTPTR+1   ;adapt hi too
B77C- 20 21 BF  GONE  JSR PATCHH   ;get next character
; (PATCH: if next char is PI, SYNTAX ERROR)
B77F- 20 85 B7    JSR GONE3       ;execute statement in A
B782- 4C 4A B7    JMP NEWSTT      ;and execute next statement
.FI "BASIC4.OB.M06"

```

1A4D 33FE-4E4B. BASIC4.OB.M06

```

;name BASIC4.OB.M06
;*****
;*          *
;* Execute command (token) in A. *
;*          *
;*****
B785- F0 3E  GONE3 BEQ ISCRTS    ;if end of statement, exit
B787- E9 80  GONE2 SBC #$80      ;convert token to keywordnumber
B789- 90 17      BCC GLET       ;if now less than zero, perform LET
B78B- C9 23      CMP #TABTK-$80  ;if token lower than A3 (TAB())
B78D- 90 06      BCC GONE4      ;go perform command
B78F- C9 48      CMP #GOTK-$80  ;if it was lower than CB (GO)
B791- 90 16      BCC SNERR1    ;go do SYNTAX ERROR
B793- E9 28      SBC #$28      ;else calculate offset (diskcommand)
B795- 0A          GONE4 ASL A      ;get offset in table
B796- A8          TAY           ;in Y
B797- B9 01 B0    LDA STMDSP+1,Y  ;get address of command hi
B79A- 48          PHA           ;push as dummy return address
B79B- B9 00 B0    LDA STMDSP,Y   ;get lo also
B79E- 48          PHA           ;complete dummy return address
B79F- 4C 70 00    JMP CHRGET     ;get next character
; (CHRGET's RTS will perform command)
B7A2- 4C 30 B9  GLET  JMP LET      ;perform LET
;*****
;*          *
;* Test for colon.          *
;*          *
;*****
B7A5- C9 3A  MORSTS CMP #':'      ;if colon (end of statement)
B7A7- F0 D3  BEQ GONE       ;go perform next statement
B7A9- 4C 00 BF  SNERR1  JMP SNERR    ;else go do SYNTAX ERROR
;*****
;*          *
;* Perform GO.          *
;*          *
;*****
B7AC- 20 76 00  GO   JSR CHRGOT    ;get current character
B7AF- A9 A4      LDA #TOTK      ;must be token for TO
B7B1- 20 F7 BE    JSR SYNCHR     ;test, if equal
B7B4- 4C 30 B8    JMP GOTO       ;perform GOTO
;*****
;*          *

```

```

        ;* Perform RESTORE.          *
        ;*                         *
        ;*****  

B7B7- 38      RESTOR SEC      ;prepare for subtract
B7B8- A5 28    LDA *TXTTAB     ;get start of prg lo
B7B9- E9 01    SBC #$01       ;decrement it
B7Bc- A4 29    LDY *TXTTAB+1   ;get pointer hi
B7Bd- B0 01    BCS RESFIN    ;if page crossed
B7C0- 88      DEY           ;adapt hi byte
B7C1- 85 3E    RESFIN STA *DATPTR  ;set pointer to current
B7C3- 84 3F    STY *DATPTR+1  ;DATA stmt to start of prg-1
B7C5- 60      ISCRTS RTS     ;and exit (warmstart basic)
        ;*****  

        ;*                         *
        ;* Perform STOP.          *
        ;*                         *
        ;*****  

B7C6- B0 01    STOP  BCS STOPC  ;if STOP, go do STOP
        ;*****  

        ;*                         *
        ;* Perform END.          *
        ;*                         *
        ;*****  

B7C8- 18      END   CLC       ;flag END
B7C9- D0 3C    STOPC BNE CONTRT ;if parameters, exit or SYNTAX ERROR
                                ;(STOP could enter here)
B7CB- A5 77    LDA *TXTPTR    ;get CHRGET's pointer
B7CD- A4 78    LDY *TXTPTR+1
B7CF- A6 37    LDX *CURLIN+1  ;and current line# hi
B7D1- E8      INX           ;if in direct mode
B7D2- F0 DC    BEQ DIRIS    ;skip saving the pointers
B7D4- 85 3A    STA *OLDTXT    ;else save CHRGET's pointer
B7D6- 84 3B    STY *OLDTXT+1
B7D8- A5 36    STPEND LDA *CURLIN ;get line#
B7DA- A4 37    LDY *CURLIN+1
B7DC- 85 38    STA *OLDLIN    ;save it too
B7DE- 84 39    STY *OLDLIN+1
B7E0- 68      DIRIS PLA     ;remove
B7E1- 68      PLA          ;return address
B7E2- A9 1B    ENDCON LDA #L,BRKTEXT ;point to BREAK message
B7E4- A0 B3    LDY #H,BRKTEXT
B7E6- 90 03    BCC GORDY    ;if STOP statement
B7E8- 4C F4 B3  JMP ERRFIN   ;print BREAK IN + line#, then restart
B7EB- 4C FF B3  GORDY JMP READY  ;else restart with READY
        ;*****  

        ;*                         *
        ;* Perform CONT.          *
        ;*                         *
        ;*****  

B7EE- D0 17    CONT  BNE CONTRT ;if parameters, SYNTAX ERROR
B7F0- A2 DB    LDX #b2e8-ERRRTAB ;point to CAN'T CONTINUE
B7F2- A4 3B    LDY *OLDTXT+1  ;if pointer is
B7F4- D0 03    BNE b7f9      ;invalid
B7F6- 4C CF B3 JMP ERROR    ;go do error
B7F9- A5 3A    b7f9  LDA *OLDTXT ;else get pointer lo
B7FB- 85 77    STA *TXTPTR   ;move it to CHRGET
B7FD- 84 78    STY *TXTPTR+1
B7FF- A5 38    LDA *OLDLIN   ;get line# lo
B801- A4 39    LDY *OLDLIN+1 ;and hi
B803- 85 36    STA *CURLIN   ;in current line# lo
B805- 84 37    STY *CURLIN+1 ;and hi
B807- 60      CONTRT RTS    ;and execute
        ;*****  

        ;*                         *

```

```

;* Perform RUN. *
;*
;*****
B808- D0 03    RUN    BNE b80d      ;if no parameters,
B80A- 4C E9 B5          JMP RUNC      ;go do CLR and execute
B80D- 20 F0 B5    b80d   JSR CLEARC   ;else do CLR
B810- 4C 27 B8          JMP RUNC2    ;and perform GOTO
;*****
;*
;* Perform GOSUB. *
;*
;*****
B813- A9 03    GOSUB   LDA #3       ;six items on stack wanted
B815- 20 93 B3          JSR GETSTK   ;check for space
B818- A5 78          LDA *TXTPTR+1 ;get CHRGET's pointer hi
B81A- 48            PHA           ;save it
B81B- A5 77          LDA *TXTPTR   ;get lo too
B81D- 48            PHA           ;save as well
B81E- A5 37          LDA *CURLIN+1 ;get current line# hi
B820- 48            PHA           ;save it
B821- A5 36          LDA *CURLIN   ;get lo too
B823- 48            PHA           ;save as well
B824- A9 8D          LDA #GOSUTK   ;and save a GOSUB token
B826- 48            PHA
B827- 20 76 00    RUNC2   JSR CHRGOT   ;get current character again
B82A- 20 30 B8          JSR GOTO     ;perform GOTO
B82D- 4C 4A B7          JMP NEWSTT   ;and execute next statement
;*****
;*
;* Perform GOTO. *
;*
;*****
B830- 20 F6 B8    GOTO    JSR LINGET   ;get line# in ($11)
B833- 20 94 B8          JSR REMN     ;get offset to end of line
B836- A5 37          LDA *CURLIN+1 ;get current line# hi
B838- C5 12          CMP *LINNUM+1 ;if lower than target
B83A- B0 0B          BCS LUK4IT   ;search from this line on
B83C- 98            TYA           ;get offset to end of line
B83D- 38            SEC           ;adapt sum
B83E- 65 77          ADC *TXTPTR   ;calculate start of
B840- A6 78          LDX *TXTPTR+1 ;next line
B842- 90 07          BCC LUKALL   ;if page crossed
B844- E8            INX           ;adapt hi byte
B845- B0 04          BCS LUKALL   ;and go search for line# from next line
B847- A5 28          LUK4IT    LDA *TXTTAB   ;get pointer to
B849- A6 29          LDX *TXTTAB+1 ;start of program
B84B- 20 A7 B5    LUKALL   JSR FNDLNC   ;search basic for line#
B84E- 90 1E          BCC USERR    ;if not found: UNDEF'D STATEMENT ERROR
B850- A5 5C          LDA *LOWTR    ;else get pointer to line
B852- E9 01          SBC #$01    ;decrement it (because of
B854- 85 77          STA *TXTPTR   ;CHRGET's increment)
B856- A5 5D          LDA *LOWTR+1 ;get hi too
B858- E9 00          SBC #0      ;subtract carry
B85A- 85 78          STA *TXTPTR+1 ;and set up CHRGET
B85C- 60            GORTS    RTS        ;and exit
;*****
;*
;* Perform RETURN *
;*
;*****
B85D- D0 FD    RETURN   BNE GORTS   ;if parameters, SYNTAX ERROR
B85F- A9 FF          LDA #$FF    ;set variable name of FOR...NEXT
B861- 85 47          STA *FORPNT+1 ;invalid
B863- 20 22 B3          JSR FNDFOR  ;get FOR block

```

```

B866- 9A TXS ;adapt stack pointer
B867- C9 8D CMP #GOSUTK ;if block was GOSUB block
B869- F0 0B BEQ RETU1 ;go do RETURN
B86B- A2 16 LDX #b223-ERRTAB ;else point to RETURN WITHOUT GOSUB
B86D- 2C .BY $2C ;skip next instruction
B86E- A2 5A USERR LDX #b267-ERRTAB ;point to UNDEF'D STATEMENT
B870- 4C CF B3 JMP ERROR ;go do error
B873- 4C 00 BF SNERR2 JMP SNERR ;go do SYNTAX ERROR
B876- 68 RETU1 PLA ;GOSUB block found, pull token
B877- 68 PLA ;get saved pointer to statement
B878- 85 36 STA *CURLIN ;store it
B87A- 68 PLA ;hi also
B87B- 85 37 STA *CURLIN+1
B87D- 68 PLA ;get CHRGET's pointer lo
B87E- 85 77 STA *TXTPTR ;load CHRGET
B880- 68 PLA ;hi also
B881- 85 78 STA *TXTPTR+1
;*****
;*          *
;* Perform DATA.          *
;*          *
;*****
B883- 20 91 B8 DATA JSR DATAN ;get offset to end of statement
B886- 98 ADDON TYA ;in accu
B887- 18 CLC ;prepare for add
B888- 65 77 ADC *TXTPTR ;adapt CHRGET's pointer lo
B88A- 85 77 STA *TXTPTR
B88C- 90 02 BCC REMRTS ;if page crossed
B88E- E6 78 INC *TXTPTR+1 ;adapt CHRGET hi
B890- 60 REMRTS RTS ;and exit
.FI "BASIC4.0B.M07"

```

1B37 33FE-4F35 BASIC4.0B.M07

```

;name BASIC4.0B.M07
;*****
;*          *
;* Get offset to end of state-  *
;* ment in Y (search for :).  *
;*          *
;*****
B891- A2 3A DATAN LDX #':' ;get search value
B893- 2C .BY $2C ;skip next statement
;*****
;*          *
;* Get offset to end of state-  *
;* ment in Y (search for zero).  *
;*          *
;*****
B894- A2 00 REMN LDX #0 ;get search value
B896- 86 03 STX *CHARAC ;save it
B898- A0 00 LDY #0 ;get alternative end
B89A- 84 04 STY *ENDCHR ;save also
B89C- A5 04 EXCHQT LDA *ENDCHR ;get zero
B89E- A6 03 LDX *CHARAC ;and search value
B8A0- 85 03 STA *CHARAC ;swap them
B8A2- 86 04 STX *ENDCHR
B8A4- B1 77 REMER LDA (TXTPTR),Y ;get current byte
B8A6- F0 E8 BEQ REMRTS ;if end of line, exit
B8A8- C5 04 CMP *ENDCHR ;if search value
B8AA- F0 E4 BEQ REMRTS ;exit as well
B8AC- C8 INY ;else adapt offset
B8AD- C9 22 CMP #'"' ;if character is not quotes

```

```

B8AF- D0 F3      BNE REMER      ;loop
B8B1- F0 E9      BEQ EXCHQT     ;else swap values (search=0) and loop
;*****
;*          *
;* Perform IF.    *
;*          *
;*****          *
B8B3- 20 98 B0  IF   JSR FRMEVL    ;evaluate expression
B8B6- 20 76 00.    JSR CHRGOT     ;get current character
B8B9- C9 89        CMP #GOTOK     ;if token for GOTO
B8BB- F0 05        BEQ OKGOTO     ;go check if condition true
B8BD- A9 A7        LDA #THENTK    ;else get token for T0
B8BF- 20 F7 BE    JSR SYNCRR    ;must be current character
B8C2- A5 5E        OKGOTO LDA *FACEXP  ;if so, get result of expression
B8C4- D0 05        BNE DOCOND    ;if true, go do action, else
;*****
;*          *
;* Perform REM.    *
;*          *
;*****          *
B8C6- 20 94 B8  ·REM   JSR REMN       ;get offset to end of line
B8C9- F0 BB        BEQ ADDON      ;and go adapt CHRGET
;*****
;*          *
;* 2nd part of IF (Perform THEN).*
;*          *
;*****          *
B8CB- 20 76 00  DOCOND JSR CHRGOT    ;get current character
B8CE- B0 03        BCS DOCO       ;if it is a number,
B8D0- 4C 30 B8      JMP GOTO       ;perform GOTO
B8D3- 4C 85 B7  DOCO   JMP GONE3     ;else execute command in A
;*****
;*          *
;* Perform ON.      *
;*          *
;*****          *
B8D6- 20 D4 C8  ONGOTO JSR GETBYT    ;get next integer in X
B8D9- 48           PHA          ;save current character
B8DA- C9 8D        CMP #GOSUTK    ;if it is GOSUB
B8DC- F0 04        BEQ ONGLOP     ;go do ON..GOSUB
B8DE- C9 89        SNERR3      ;CMP #GOTOK     ;if it is not GOTO
B8E0- D0 91        BNE SNERR2    ;SYNTAX ERROR
;BUG: GO TO not allowed
B8E2- C6 62        ONGLOP DEC *FACLO    ;decrement LSB of FLP accu
B8E4- D0 04        BNE ONGLP1    ;until zero, then
B8E6- 68           PLA          ;reget character (token)
B8E7- 4C 87 B7      JMP GONE2     ;and execute command in A
B8EA- 20 70 00  ONGLP1 JSR CHRGET    ;else get next character
B8ED- 20 F6 B8      JSR LINGET    ;get integer in ($11)
B8F0- C9 2C        CMP #','
B8F2- F0 EE        BEQ ONGLOP    ;get next character
B8F4- 68           PLA          ;if not end of line, SYNTAX ERROR
B8F5- 60           ONGRTS RTS      ;if end of line, execute next line
;*****
;*          *
;* Get integer decimal number  *
;* in ($11).                  *
;*          *
;*****          *
B8F6- A2 00        LINGET LDX #0      ;zero result
B8F8- 86 11        STX *LINNUM
B8FA- 86 12        STX *LINNUM+1
B8FC- B0 F7        MORLIN BCS ONGRTS  ;if not numeric character, stop
B8FE- E9 2F        SBC #$2F      ;convert to binary number

```

```

B900- 85 03 STA *INTEGR ;save it
B902- A5 12 LDA *LINNUM+1 ;get result hi so far
B904- 85 1F STA *INDEX ;save it
B906- C9 19 CMP #H,6400 ;if result so far >=6400
B908- B0 04 BCS SNERR3 ;go do SYNTAX ERROR
;BUG: crash if hi is equal to $89
B90A- A5 11 LDA *LINNUM ;else get lo
B90C- 0A ASL A ;multiply by 2
B90D- 26 1F ROL *INDEX ;multiply hi also
B90F- 0A ASL A ;multiply by 4
B910- 26 1F ROL *INDEX ;multiply hi also
B912- 65 11 ADC *LINNUM ;add to itself (times 5)
B914- 85 11 STA *LINNUM ;store result-lo
B916- A5 1F LDA *INDEX ;add carry
B918- 65 12 ADC *LINNUM+1 ;to hi byte
B91A- 85 12 STA *LINNUM+1
B91C- 06 11 ASL *LINNUM ;multiply by 2 (10 in total now)
B91E- 26 12 ROL *LINNUM+1 ;hi byte also
B920- A5 11 LDA *LINNUM ;get number
B922- 65 03 ADC *INTEGR ;add current digit
B924- 85 11 STA *LINNUM ;save it
B926- 90 02 BCC NXTLGC ;if 'page' crossed
B928- E6 12 INC *LINNUM+1 ;adapt high byte
B92A- 20 70 00 NXTLGC JSR CHRGET ;get next character
B92D- 4C FC B8 JMP MORLIN ;and loop
;*****
;* *
;* Perform LET. *
;* *
;*****
B930- 20 2B C1 LET JSR PTRGET ;get name and pointer of variable
B933- 85 46 STA *FORPNT ;store pointer found
B935- 84 47 STY *FORPNT+1
B937- A9 B2 LDA #EQLTK ;get token for '='
B939- 20 F7 BE JSR SYNCHR ;must be next, if not SYNTAX ERROR
B93C- A5 08 LDA *INTFLG ;get integer flag
B93E- 48 PHA ;save it
B93F- A5 07 LDA *VALTYP ;get string flag
B941- 48 PHA ;save as well
B942- 20 98 BD JSR FRMEVL ;evaluate expression
B945- 68 PLA ;get string flag back
B946- 2A ROL A ;get flag bit in carry
B947- 20 8A BD JSR CHKVÁL ;check if var. and ex. same type
B94A- D0 18 BNE COPSTR ;if same, go assign string
B94C- 68 PLA ;get integer flag back
B94D- 10 12 QINTGR BPL COPFLT ;if floating, go assign
;*****
;* *
;* Assign integer. *
;* *
;*****
B94F- 20 51 CD JSR ROUND ;else assign integer, round FLP accu
B952- 20 EA C2 JSR AYINT ;convert FLP to integer
B955- A0 00 LDY #0 ;get index
B957- A5 61 LDA *FACMO ;get lo byte of integer
B959- 91 46 STA (FORPNT),Y ;and store hi byte
B95B- C8 INY
B95C- A5 62 LDA *FACLO ;get hi byte
B95E- 91 46 STA (FORPNT),Y ;store as well
B960- 60 RTS
;*****
;* *
;* Assign floating point. *
;* *

```

```

;*****
B961- 4C 06 CD COPFLT JMP MOVF      ;go assign FLP number
;*****
;*          *
;* Assign string.           *
;*          *
;*****  

B964- 68    COPSTR PLA      ;string assignment, get integer flag
B965- A4 47 INPCOM LDY *FORPNT+1 ;if high byte of pointer to variable
B967- C0 00   CPY #H,ZERO   ;not the same as dummy variable
B969- D0 4F   BNE GETSPT   ;go assign normal string variable
;*****
;*          *
;* Assign to TI$.           *
;*          *
;*****  

B96B- 20 B8 C7 JSR FREFAC   ;else assign TI$, discard temporary str
B96E- C9 06 CMP #6        ;length must be 6
B970- D0 40 BNE FCERR2   ;if not ILLEGAL QUANTITY ERROR
B972- A0 00 LDY #0        ;get zero
B974- 84 5E STY *FACEXP   ;store zero in exponent
B976- 84 63 STY *FACSGN   ;and in sign
B978- 84 6E TIMELP STY *STRNG2 ;and in pointer
B97A- 20 AB B9 JSR TIMNUM   ;add next character to FLP accu
B97D- 20 18 CC JSR MUL10   ;multiply FLP accu with 10
B980- E6 6E INC *STRNG2   ;adapt pointer
B982- A4 6E LDY *STRNG2   ;get it
B984- 20 AB B9 JSR TIMNUM   ;and add next character
B987- 20 42 CD JSR MOVAF   ;copy rounded FLP1 to FLP2
B98A- AA TAX        ;get exponent
B98B- F0 05 BEQ NOML6   ;if zero
B98D- E8 INX        ;adapt it (FLP2 times 2)
B98E- 8A TXA        ;get it in A (result times 2, total *6)
B98F- 20 23 CC JSR FINML6  ;add FLP2 to FLP1, then multiply by 2
B992- A4 6E NOML6 LDY *STRNG2 ;get index
B994- C8 INY        ;adapt it
B995- C0 06 CPY #6        ;if all digits done
B997- D0 DF BNE TIMELP   ;get value of time
B999- 20 18 CC JSR MUL10   ;convert value to four integers
B99C- 20 D1 CD JSR QINT    ;move 3 bytes
B99F- A2 02 LDX #2        ;disable interrupts
B9A1- 78 SEI        ;get value of time
B9A2- B5 60 TIMEST LDA *FAC+2,X ;in clock area
B9A4- 95 80 STA *CTIMR,X   ;adapt counter
B9A6- CA DEX        ;if not all done, loop
B9A7- 10 F9 BPL TIMEST   ;allow interrupts again
B9A9- 58 CLI        .FI "BASIC4.0B.M08"
B9AA- 60 RTS        ;get character of string

```

18E1 33FE-4CDF BASIC4.0B.M08

```

;name BASIC4.0B.M08
;*****
;*          *
;* Get next character for TI$.   *
;*          *
;*****  

B9AB- B1 1F TIMNUM LDA (INDEX),Y ;get character of string
B9AD- 20 70 00 JSR QNUM    ;set CHRGET flags
B9B0- 90 03 BCC GOTNUM   ;if not a digit
B9B2- 4C 73 C3 FCERR2 JMP FCERR  ;give ILLEGAL QUANTITY ERROR
B9B5- E9 2F GOTNUM SBC #$2F ;else get value

```

B9B7- 4C B4 CE	JMP FINLOG ;and add to FLP1 ;***** ;* * ;* Test for DS\$. * ;* * ;*****
B9BA- A0 02	GETSPT LDY #2 ;point to text pointer hi
B9BC- B1 61	LDA (FACMO),Y ;get it
B9BE- C5 0F	DSKXO CMP *DSDESC+2 ;if equal to pointer to DS\$
B9C0- D0 12	BNE DSKX2
B9C2- 48	PHA ;save the pointer
B9C3- 88	DEY ;point to pointer lo
B9C4- A5 0D	LDA *DSDESC ;get length of DS\$
B9C6- F0 DA	BEQ DSKX1 ;if not zero
B9C8- B1 61	LDA (FACMO),Y ;get pointer lo
B9CA- C5 0E	CMP *DSDESC+1 ;if equal to pointer to DS\$
B9CC- D0 04	BNE DSKX1 ;if not, assign normal string
B9CE- 68	PLA ;get pointer hi back
B9CF- 4C F6 B9	JMP COPY ;and assign DS\$;***** ;* * ;* Assign normal string. * ;* * ;*****
B9D2- 68	DSKX1 PLA ;get pointer hi back
B9D3- C8	INY ;point to pointer hi again
B9D4- C5 31	DSKX2 CMP *FRETOP+1 ;if below stringtext area
B9D6- 90 17	BCC DNTPY ;go copy descriptor
B9D8- D0 07	BNE QVARIA ;if in allocation area
B9DA- 88	DEY ;point to pointer lo
B9DB- B1 61	LDA (FACMO),Y ;get pointer lo
B9DD- C5 30	CMP *FRETOP ;if below allocation area
B9DF- 90 0E	BCC DNTPY ;go copy descriptor
B9E1- A4 62	QVARIA LDY *FACLO ;get pointer to string hi
B9E3- C4 28	CPY *VARTAB+1 ;if within program
B9E5- 90 08	BCC DNTPY ;go copy descriptor
B9E7- D0 00	BNE COPY ;if not in program, copy string
B9E9- A5 61	LDA *FACMO ;hi's equal, get pointer to string lo
B9EB- C5 2A	CMP *VARTAB ;if within program
B9ED- B0 07	BCS COPY
B9EF- A5 61	DNTPY LDA *FACMO ;get descr. pointer lo
B9F1- A4 62	LDY *FACLO ;and hi
B9F3- 4C 13 BA	JMP COPY ;and set up variable (don't move text) ;***** ;* * ;* Copy string to high memory. * ;* * ;*****
B9F6- A0 00	COPY LDY #0 ;point to length of string
B9F8- B1 61	LDA (FACMO),Y ;get length
B9FA- 20 9E C5	JSR STRINI ;allocate area for string
B9FD- A5 4D	LDA *DSCPNT ;get variable pointer
B9FF- A4 4E	LDY *DSCPNT+1
BA01- 85 6C	STA *STRNG1 ;in (\$6C)
BA03- 84 60	STY *STRNG1+1
BA05- 20 8C C7	JSR MOVINS ;move stringtext into hi-memory
BA08- A5 6C	LDA *STRNG1 ;get its var. pointer
BA0A- A4 60	LDY *STRNG1+1 ;and
BA0C- 20 11 C8	JSR FRETMS ;clean the descriptor stack
BA0F- A9 5E	LDA #L,FAC ;point A and Y
BA11- A0 00	LDY #H,FAC ;to FLP1 (current descriptor) ;***** ;* * ;* Set up backpointer. *

```

        ;*
        ;*****
        BA13- 85 4D    COPYC STA *DSCPNT      ;store pointer to new descriptor
        BA15- 84 4E    STY *DSCPNT+1   ;in ($4D)
        BA17- 85 1F    STA *INDEX       ;and in
        BA19- 84 20    STY *INDEX+1    ;(INDEX)
        BA1B- 20 11 C8  JSR FRETMS     ;clean descriptor stack
        BA1E- 20 4E BA  JSR STRADJ     ;set up pointer for backpointer
        BA21- 90 0B    BCC COPY00      ;if backpointer needed
        BA23- A0 00    LDY #0          ;point to backpointer
        BA25- A5 46    LDA *FORPNT     ;get value pointer lo
        BA27- 91 1F    STA (INDEX),Y  ;store it in hi-memory
        BA29- C8       INY             ;(after stringtext)
        BA2A- A5 47    LDA *FORPNT+1   ;get hi byte of value pointer
        BA2C- 91 1F    STA (INDEX),Y  ;and complete backpointer
        ;*****
        ;*
        ;* Set old backpointer invalid. *
        ;*
        ;*****
        BA2E- A5 46    COPY00 LDA *FORPNT     ;get pointer to variable again
        ;(old descriptor)
        BA30- 85 1F    STA *INDEX       ;save in (INDEX)
        BA32- A5 47    LDA *FORPNT+1   ;get hi byte
        BA34- 85 20    STA *INDEX+1    ;save too
        BA36- 20 4E BA  JSR STRADJ     ;set up pointer to old backpointer
        BA39- 90 09    BCC COPY01      ;if DS$ or TI$ copy descriptor only
        BA3B- 88       DEY             ;else point to old backpointer hi
        BA3C- A9 FF    LDA #$FF        ;get invalid flag
        BA3E- 91 1F    STA (INDEX),Y  ;set backpointer invalid
        BA40- 88       DEY             ;point to old backpointer lo
        BA41- 8A       TXA             ;get old length
        BA42- 91 1F    STA (INDEX),Y  ;in old backpointer lo
        ;*****
        ;*
        ;* Copy new descriptor to var. *
        ;* or array area.           *
        ;*
        ;*****
        BA44- A0 02    COPY01 LDY #2          ;copy three bytes (descriptor)
        BA46- B1 40    COPY02 LDA (DSCPNT),Y  ;get byte (of new descr.)
        BA48- 91 46    STA (FORPNT),Y    ;move to variable or array memory
        BA4A- 88       DEY             ;adapt counter
        BA4B- 10 F9    BPL COPY02      ;and loop
        BA4D- 60       RTS             ;exit
        ;*****
        ;*
        ;* Set up pointer for back-  *
        ;* pointer in ($1F) if needed. *
        ;* If set up, carry flag is  *
        ;* set. On entry, ($1F) must  *
        ;* point to descriptor.      *
        ;*
        ;*****
        BA4E- A0 00    STRADJ LDY #0          ;point to length
        BA50- B1 1F    LDA (INDEX),Y  ;get it
        BA52- 48       PHA             ;save on stack, if zero
        BA53- F0 30    BEQ ADJ01       ;exit with carry clear (no string)
        BA55- C8       INY             ;point to pointer lo
        BA56- B1 1F    LDA (INDEX),Y  ;get it
        BA58- AA       TAX             ;save it X
        BA59- C8       INY             ;point to pointer hi
        BA5A- B1 1F    LDA (INDEX),Y  ;get it, if it points to nonsense
        BA5C- 30 27    BMI ADJ01       ;exit with carry clear

```

```

BA5E- C9 B0      CMP #H,ROMLOC    ;if it points below ROM
BA60- B0 23      BCS ADJ01       ;exit with carry clear
BA62- C5 31      CMP *FRET0P+1  ;if not in string area
BA64- 90 1F      BCC ADJ01       ;exit with carry clear
BA66- D0 04      BNE ADJ       ;not in area, go check for DS$
BA68- E4 30      CPX *FRET0P   ;hi's equal, compare lo's
BA6A- 90 19      BCC ADJ01       ;if not in area, exit with carry clear
BA6C- 4C 10 BF   ADJ      JMP PABBO
BA6F- EA         NOP
BA70- E4 0E      ADJXX      CPX *DSDESC+1
BA72- F0 11      BEQ ADJ01       ;exit with carry clear
BA74- 86 1F      ADJ02      STX *INDEX
BA76- 85 20      STA *INDEX+1
BA78- 68         PLA
BA79- AA         TAX
BA7A- 18         CLC
BA7B- 65 1F      ADC *INDEX
BA7D- 85 1F      STA *INDEX
BA7F- 90 02      BCC ADJ00
BA81- E6 20      INC *INDEX+1
BA83- 38         ADJ00      SEC
BA84- 60         RTS
BA85- 68         ADJ01      PLA
BA86- 18         CLC
BA87- 60         RTS
BA88- 20 8E BA   PRINTN JSR CMD      ;perform CMD and PRINT
BA8B- 4C B4 BB   JMP IODONE     ;and reset file entry
BA8E- 20 D4 C8   CMD      JSR GETBYT   ;get integer (file#) in X
BA91- F0 05      BEQ SAVEIT     ;if none
BA93- A9 2C      LDA #','
BA95- 20 F7 BE   JSR SYNCHR
BA98- 08         SAVEIT      PHP
BA99- 20 C9 FF   JSR COOUT     ;set output device
BA9C- 86 10      STX *CHANNL   ;in current CMD device#
BA9E- 28         PLP
BA9F- 4C A8 BA   JMP PRINT     ;and go perform PRINT
.FI "BASIC4.DB.M09"

```

1ACF 33FE-4ECD BASIC4.DB.M09

```

;name BASIC4.DB.M09
;*****
;* Loop entries in PRINT. *
;*
;*****
BAA2- 20 20 BB   STRDON JSR STRPRT   ;print string from ($1F)
BAA5- 20 76 00   NEWCHR JSR CHRGOT   ;get current character
;*****
;* Perform PRINT. *
;*
;*****

```

```

BAA8- F0 35 PRINT BEQ CRDO ;if end of statement, go print CR(LF)
BAAA- F0 43 PRINTC BEQ PRTRTS ;if end of statement, return
BAAC- C9 A3 CMP #TABTK ;if TAB( is next
BAAE- F0 4D BEQ TABER ;go do TAB
BAB0- C9 A6 CMP #SPCTK ;if SPC( is next
BAB2- 18 CLC ;flag SPC
BAB3- F0 48 BEQ TABER ;and go SPC
BAB5- C9 2C CMP #', ' ;if comma
BAB7- F0 37 BEQ COMPRT ;advance cursor to next stop
BAB9- C9 3B CMP #' ';' ;if semicolon
BABB- F0 55 BEQ NOTABR ;get next character and loop
BABD- 20 98 BD JSR FRMEVL ;else evaluate expression
BAC0- 24 07 BIT *VALTYP ;if string type
BAC2- 30 DE BMI STRDON ;go print string and loop
BAC4- 20 93 CF JSR FOUT ;else convert FLP# to string
BAC7- 20 B0 C5 JSR STRLIT ;get descriptor of string in FLP accu
BACA- 20 20 BB JSR STRPRT ;print string from ($1F)
BACD- 20 3A BB JSR OUTSPC ;print space or cursor left
BAD0- D0 D3 BNE NEWCHR ;and loop
;*****
;*
;* End statement on screen and *
;* in inputbuffer. *
;*
;*****
BAD2- A9 00 FININL LDA #0 ;put terminator
BAD4- 9D 00 02 STA BUF,X ;in input buffer
BAD7- A2 FF LDX #L,BUF-1 ;point before inputbuffer
BAD9- A0 01 LDY #H,BUF-1
BADD- D0 10 LDA *CHANNL ;get file number
BADD- D0 10 BNE PRTRTS ;if default do
;*****
;*
;* Print CR (and LF if file- *
;* number is higher than 127). *
;*
;*****
BADF- A9 0D CRDO LDA #$0D ;get a Carriage Return
BAE1- 20 46 BB JSR OUTDO ;print it
BAE4- A5 10 LDA *CHANNL ;get current file number
BAE6- 10 05 BPL CRFIN ;if higher than 127
BAE8- A9 0A LDA #$0A ;get a Line Feed
BAEA- 20 46 BB JSR OUTDO ;print it
BAE0- 49 FF CRFIN EOR #$FF ;and set flags according to A
BAEF- 60 PRTRTS RTS ;then exit
;*****
;*
;* Comma found in PRINT state- *
;* ment. *
;*
;*****
BAF0- A5 C6 COMPRT LDA *TRMPOS ;get current column#
BAF2- 38 SEC ;prepare for subtraction
BAF3- E9 0A MORCO1 SBC #10 ;subtract ten
BAF5- B0 FC BCS MORCO1 ;until negative (modulo 10)
BAF7- 49 FF EOR #$FF ;get real remainder (amount to go forwa
BAF9- 69 01 ADC #1 ;add 1 (two's complement)
BAFB- D0 10 BNE ASPAC ;and generate spaces or RIGHTS
;*****
;*
;* Perform TAB( (carry set) or *
;* SPC( (carry clear). *
;*
;*****

```

BAFD- 08	TABER	PHP	;save flags (and carry)
BAFE- 20 D1 C8		JSR GTBYTC	;read number into X
BB01- C9 29		CMP #'')	;if not ended with closing par.
BB03- D0 59		BNE SNERR4	;give SYNTAX ERROR
BB05- 28		PLP	;get flags (and carry) back
BB06- 90 06		BCC XSPAC	;if SPC(), go do SPC()
BB08- 8A		TXA	;else do TAB(), get column# in A
BB09- E5 C6		SBC *TRMPOS	;calculate amount to go forward
BB0B- 90 05		BCC NOTABR	;if cursor must go back, loop in PRINT
BB0D- AA	ASPAC	TAX	;else get amount in X
BB0E- E8	XSPAC	INX	;adapt number
BB0F- CA	XSPAC2	DEX	;adapt counter
BB10- D0 06		BNE XSPAC1	;if not zero, generate space/RIGHT
BB12- 20 70 00	NOTABR	JSR CHRGET	;else get next character
BB15- 4C AA BA		JMP PRINTC	;and loop in PRINT
BB18- 20 3A BB	XSPAC1	JSR OUTSPC	;generate a space or RIGHT
BB1B- D0 F2		BNE XSPAC2	;and loop in TAB()//SPC()

		/*	*
		/* Print string from YA until	*
		/* zero byte found.	*
		/*	*

BB1D- 20 B0 C5	STROUT	JSR STRLIT	;get descriptor in FLP accu
BB20- 20 B8 C7	STRPR2	JSR FREFAC	;get pointer in (\$1F) and A
BB23- AA		TAX	;get length in X
BB24- A0 00		LDY #0	;get initial offset
BB26- E8		INX	;adapt length
BB27- CA	STRPR2	DEX	;adapt counter
BB28- F0 C5		BEQ PRTRTS	;if zero, exit
BB2A- B1 1F		LDA (INDEX),Y	;get character from string
BB2C- 20 46 BB		JSR OUTDO	;print it
BB2F- C8		INY	;adapt index
BB30- C9 00		CMP #\$00	;if character was Carriage Return
BB32- D0 F3		BNE STRPR2	
BB34- 20 ED BA		JSR CRFIN	;set flags (why???)
BB37- 4C 27 BB		JMP STRPR2	;and loop

		/*	*
		/* Print cursor right or space	*
		/* dependent on CMD output file	*
		/* number.	*
		/*	*

BB3A- A5 10	OUTSPC	LDA *CHANNEL	;get CMD output file#
BB3C- F0 03		BEQ CRTSKP	;if zero (default, screen) print right
BB3E- A9 20		LDA ##\$20	;else get space
BB40- 2C		.BY \$2C	;skip next instruction
BB41- A9 1D	CRTSKP	LDA ##\$10	;get a cursor right
BB43- 2C		.BY \$2C	;skip next instruction

		/*	*
		/* Print a question mark.	*
		/* (Used in error messages and	*
		/* in INPUT).	*
		/*	*

BB44- A9 3F	OUTQST	LDA #'?'	;get a question mark
BB46- 20 02 FF	OUTDO	JSR OUTCH	;print the character in A
BB49- 29 FF		AND ##\$FF	;set flags according to (A)
BB4B- 60	OUTRTS	RTS	;and exit

		/*	*
		/* Error exit from INPUT, GET or *	*

```

;* READ. *
;*
;*****
BB4C- A5 0B    TRMNOK LDA *INPFLG      ;get INPUT/READ/GET flag
BB4E- F0 11    BEQ TRMNO1      ;if INPUT, go input
BB50- 30 04    BMI GETDTL      ;if READ, go read
BB52- A0 FF    LDY #$FF       ;else GET, flag direct mode
BB54- D0 04    BNE STCURL     ;store in
BB56- A5 3C    GETDTL LDA *DATLIN     ;get line# lo of current DATA line
BB58- A4 3D    LDY *DATLIN+1   ;and hi also
BB5A- 85 36    STCURL STA *CURLIN     ;store in
BB5C- 84 37    STY *CURLIN+1   ;current BASIC line number
BB5E- 4C 00 BF  SNERR4 JMP SNERR      ;and give SYNTAX ERROR
;*****
;*
;* Error exit from INPUT. *
;*
;*****
BB61- A5 10    TRMNO1 LDA *CHANNEL    ;get current CMD file#
BB63- F0 05    BEQ DOAGIN      ;if default go do REDO FROM START
BB65- A2 BF    LDX #b2cc-ERRTAB   ;else point to FILE DATA ERROR
BB67- 4C CF B3  JMP ERROR       ;and go do error
BB6A- A9 07    DOAGIN LDA #L,TRYAGN   ;point to
BB6C- A0 B0    LDY #H,TRYAGN   ;?REDO FROM START
BB6E- 20 10 BB  JSR STROUT      ;print that string
BB71- A5 3A    LDA *OLDTXT      ;get saved pointer lo for CHRGET
BB73- A4 3B    LDY *OLDTXT+1   ;and hi also
BB75- 85 77    STA *TXTPTR      ;load CHRGET's
BB77- 84 78    STY *TXTPTR+1   ;pointer again
BB79- 60      RTS           ;and return
;*****
;*
;* Perform GET. *
;*
;*****
BB7A- 20 CF C4  GET    JSR ERRDIR      ;check if not in direct mode
BB7D- C9 23    CMP #'#'       ;if not followed by #
BB7F- D0 10    BNE GETTTY      ;GET from keyboard
BB81- 20 70 00  JSR CHRGET      ;else get 1st character
BB84- 20 D4 C8  JSR GETBYT      ;and read file# in X
BB87- A9 2C    LDA #','
BB89- 20 F7 BE  JSR SYNCHR     ;comma must follow
BB8C- 20 C6 FF  JSR COIN'      ;test that
BB8F- 86 10    STX *CHANNEL    ;set input device
BB91- A2 01    GETTTY LDX #L,BUF+1   ;and store current CMD file#
BB93- A0 02    LDY #H,BUF+1   ;point to end
BB95- A9 00    LDA #$00       ;of inputted value ($0201)
BB97- 8D 01 02  STA BUF+1      ;get end delimiter
BB9A- A9 40    LDA #$40       ;in inputbuffer
BB9C- 20 0B BC  JSR INPC01      ;get flag for GET
BB9F- A6 10    LDX *CHANNEL    ;perform 'READ' (1 char.)
BBA1- D0 13    BNE IORELE     ;if current file#
BBA3- 60      RTS           ;not default, go restore default I/O
                             ;else exit
.FI "BASIC4.0B.M10"

```

1980 33FE-4DBB BASIC4.0B.M10

```

;name BASIC4.0B.M10
;*****
;*
;* Perform INPUT#. *
;*
;*****

```

```

B8A4- 20 D4 C8 INPUTN JSR GETBYT      ;read file number in X
B8A7- A9 2C             LDA #','
B8A9- 20 F7 BE          JSR SYNCHR     ;comma must follow
B8AC- 20 C6 FF          JSR COIN       ;test that
B8AF- 86 10             STX *CHANNL   ;select input device
B8B1- 20 CD BB          JSR NOTQTI    ;and store current CMD file#
B8B4- A5 10             IODONE LDA *CHANNL ;go do INPUT
B8B6- 20 CC FF          IORELE JSR CLSCHN ;reget current CMD file#
B8B9- A2 00             LDX #$00      ;and restore default I/O
B8B8- 86 10             STX *CHANNL   ;get default file#
B8BD- 60               RTS          ;in current CMD file#
                                ;and exit
                                ;*****
                                ;*
                                ;* Perform INPUT.      *
                                ;*
                                ;*****
B8BE- C9 22             INPUT  CMP #'"'  ;if no quotes following
B8C0- 00 0B             BNE NOTQTI   ;go do input
B8C2- 20 B5 BE          JSR STRTXT    ;else get descriptor of prompt string
B8C5- A9 3B             LDA #';'     ;semicolon must follow string
B8C7- 20 F7 BE          JSR SYNCHR    ;test that
B8CA- 20 20 BB          JSR STRPRT    ;print the string
B8CD- 20 CF C4          NOTQTI JSR ERRDIR ;check if not in direct mode
B8D0- A9 2C             LDA #','
B8D2- 80 FF 01          STA BUF-1    ;precede input buffer
B8D5- 20 F5 BB          GETAGN JSR QINLIN ;with a comma
B8D8- A5 10             LDA *CHANNL  ;print a question mark and get input
B8DA- F0 0C             BEQ BUFFUL   ;if current CMD file#
B8DC- A5 96             LDA *CSTAT    ;is default, test for input given
B8DE- 29 03             AND #$03    ;else get status byte
B8E0- F0 06             BEQ BUFFUL   ;if no error occurred
B8E2- 20 B4 BB          JSR IODONE   ;go end input
B8E5- 4C 83 B8          JMP DATA     ;else restore default I/O
                                ;and go skip rest of statement
                                ;*****
                                ;*
                                ;* Test if any input given.      *
                                ;* If none, restart BASIC with  *
                                ;* 'READY.'.                  *
                                ;*
                                ;*****
B8E8- AD 00 02          BUFFUL LDA BUF     ;if any input
B8EB- 00 1C             BNE INPCON   ;perform 'READ'
B8ED- 4C 2E BF          JMP PATCHI   ;else PATCH: if no input, retry on IEEE
B8F0- EA               NOP          ;inserted because of patch
B8F1- 18               PTHRTI CLC     ;prevent BREAK message (select END)
B8F2- 4C D8 B7          JMP STPEND   ;and perform END
                                ;*****
                                ;*
                                ;* Output a question mark and  *
                                ;* get input from buffer.      *
                                ;*
                                ;*****
B8F5- A5 10             QINLIN LDA *CHANNL ;if current CMD file# is
B8F7- 00 06             BNE GINLIN   ;default,
B8F9- 20 44 BB          JSR OUTQST   ;output a question mark
B8FC- 20 3A BB          JSR OUTSPC   ;and a cursor right
B8FF- 4C E2 B4          GINLIN JMP INLIN ;and get input from buffer
                                ;*****
                                ;*
                                ;* Perform READ.            *
                                ;*
                                ;*****
BC02- A6 3E             READ   LDX *DATPTR  ;get pointer to current
BC04- A4 3F             LDY *DATPTR+1 ;DATA value in YX

```

```

BC06- A9 98      LDA #$98      ;get flag for READ (MSB set)
BC08- 2C          .BY $2C      ;skip next instruction
BC09- A9 00      INPCON LDA #$00 ;entry for INPUT, get flag
BC0B- 85 0B      INPC01 STA *INPFLG ;set up flag (entry w. GET)
BC0D- 86 40      STX *INPPTR   ;store pointer to value'(YX)
BC0F- 84 41      STY *INPPTR+1 ;in temporary read pointer
BC11- 20 2B C1    INLOOP JSR PTRGET
BC14- 85 46      STA *FORPNT  ;store variable
BC16- 84 47      STY *FORPNT+1 ;pointer
BC18- A5 77      LDA *TXTPTR  ;get CHRGET's pointer
BC1A- A4 78      LDY *TXTPTR+1
BC1C- 85 48      STA *VARTXT ;in save
BC1E- 84 49      STY *VARTXT+1 ;area for pointer
BC20- A6 40      LDX *INPPTR  ;get pointer to
BC22- A4 41      LDY *INPPTR+1 ;value back
BC24- 86 77      STX *TXTPTR  ;point CHRGET
BC26- 84 78      STY *TXTPTR+1 ;to value
BC28- 20 76 00    JSR CHRGOT  ;get value, 1st character
BC2B- D0 20      BNE DATBK1 ;if end of statement
BC2D- 24 0B      BIT *INPFLG ;test flag
BC2F- 50 0C      BVC QDATA   ;if GET
BC31- 20 E4 FF    JSR CGETL   ;get a character
BC34- 8D 00 02    STA BUF     ;store in input buffer
BC37- A2 FF      LDX #L,BUF-1 ;set up
BC39- A0 01      LDY #H,BUF-1 ;pointer to character
BC3B- D0 0C      BNE DATBK   ;and get character via CHRGET
BC3D- 30 75      QDATA BMI DATLOP ;if READ, go search DATA line
BC3F- A5 10      LDA *CHANNL ;INPUT(#), get current CMD file#
BC41- D0 03      BNE GETNTH ;if default
BC43- 20 44 BB    JSR OUTQST  ;output a 2nd question mark
BC46- 20 F5 BB    GETNTH JSR QINLIN ;get input from buffer
BC49- 86 77      DATBK STX *TXTPTR ;store pointer to input
BC4B- 84 78      STY *TXTPTR+1 ;in CHRGET
BC4D- 20 70 00    DATBK1 JSR CHRGOT ;get character from inputted value
BC50- 24 07      BIT *VALTYP ;if variable type
BC52- 10 31      BPL NUMINS ;is string
                                ;*****
                                ;*
                                ;* Input and assign string. *
                                ;*
                                ;*****
BC54- 24 0B      BIT *INPFLG ;and mode is
BC56- 50 09      BVC SETQUT ;GET
BC58- E8          INX         ;adapt pointer
BC59- 86 77      STX *TXTPTR ;in CHRGET
BC5B- A9 00      LDA #$00   ;set terminator
BC5D- 85 03      STA *CHARAC ;in save
BC5F- F0 0C      BEQ RESETC ;and go assign string
BC61- 85 03      SETQUT STA *CHARAC ;else if READ or INPUT(#)
BC63- C9 22      CMP #'"' ;if quotes read
BC65- F0 07      BEQ NOWGET ;store them
BC67- A9 3A      LDA #':' ;else get dummy end of statement
BC69- 85 03      STA *CHARAC ;store that as terminator
BC6B- A9 2C      LDA #',,' ;and a comma as alternative
BC6D- 18          RESETC CLC
BC6E- 85 04      NOWGET STA *ENDCHR ;save alternative terminator
BC70- A5 77      LDA *TXTPTR ;get CHRGET's pointer
BC72- A4 78      LDY *TXTPTR+1 ;in A and Y
BC74- 69 00      ADC #$00   ;add carry (if numeric,
BC76- 90 01      BCC NOWGE1 ;increment pointer)
BC78- C8          INY         ;if page crossed, adapt hi
BC79- 20 B6 C5    NOWGE1 JSR STRLT2 ;get descriptor in FLP accu
BC7C- 20 18 C9    JSR ST2TXT ;get pointer to string CHRGET
BC7F- 20 65 B9    JSR INPCOM ;assign string

```

BC82- 4C 8D BC	JMP STRDN2 ;and check for more ;***** ;* * ;* Input and assign a numeral. * ;* * ;*****
BC85- 20 29 CE	NUMINS JSR FIN ;get input number in FLP accu
BC88- A5 08	LDA *INTFLG ;get integer/FLP flag
BC8A- 20 4D B9	JSR QINTGR ;and assign number
BC8D- 20 76 00	STRDN2 JSR CHRGOT ;get current character
BC90- F0 07	BEQ TRMOK ;if end of statement, quit input
BC92- C9 2C	CMP #',,' ;if comma
BC94- FD 03	BEQ TRMOK ;go loop
BC96- 4C 4C BB	JMP TRMNOK ;else give SYNTAX ERROR
BC99- A5 77	TRMOK LDA *TXTPTPR ;get CHRGET's pointer
BC9B- A4 78	LDY *TXTPTPR+1 ;save the pointer
BC9D- 85 40	STA *INPPTR ;in (\$40)
BC9F- 84 41	STY *INPPTR+1 ;get saved CHRGET pointer
BCA1- A5 48	LDA *VARTXT ;lo-hi
BCA3- A4 49	LDY *VARTXT+1 ;in CHRGET
BCA5- 85 77	STA *TXTPTPR ;again
BCA7- 84 78	STY *TXTPTPR+1 ;get current character
BCA9- 20 76 00	JSR CHRGOT ;if end of statement, test if more
BCAC- FD 2C	BEQ VAREND ;if not, test for comma
BCAE- 20 F5 BE	JSR CHKCOM ;if comma, go and repeat
BCB1- 4C 11 BC	JMP INLOOP ;***** ;* * ;* Search for next DATA line in * ;* BASIC program. * ;* (end of line found in current * ;* DATA line). * ;* * ;*****
BCB4- 20 91 B8	DATLOP JSR DATAN ;get offset to end of statement
BCB7- C8	INY ;point to next statement
BCB8- AA	TAX ;get end of statement character
BCB9- D0 12	BNE NOWLIN ;if zero found
BCBB- A2 2A	LDX #b237-ERRTAB ;point to OUT OF DATA
BCBD- C8	INY
BCBE- B1 77	LDA (TXTPTPR),Y ;get link
BCC0- FD 6B	BEQ ERRGOS ;if end of program, go do error
BCC2- C8	INY ;else point to line# lo
BCC3- B1 77	LDA (TXTPTPR),Y ;get it
BCC5- 85 3C	STA *DATLIN ;put in line# of
BCC7- C8	INY ;current DATA line
BCC8- B1 77	LDA (TXTPTPR),Y ;get line# hi
BCCA- C8	INY ;point to 1st statement
BCCB- 85 3D	STA *DATLIN+1 ;save line# hi
BCCD- B1 77	NOWLIN LDA (TXTPTPR),Y ;colon found, get statement
BCCF- AA	TAX ;in X
BCD0- 20 86 B8	JSR ADDON ;point CHRGET to next statement
BCD3- ED 83	CPX #DATATK ;if statement is not DATA
BCD5- D0 DD	BNE DATLOP ;go try next line
BCD7- 4C 4D BC	JMP DATBK1 ;if DATA statement found, perform READ .FI "BASIC4.DB.M11"

1AD9 33FE-4ED7 BASIC4.DB.M11

```
;name BASIC4.DB.M11
;*****
;* *  
;* End in INPUT(#/GET/READ. *
```

```

        ;* *
        ;*****  

BCDA- A5 40    VAREND LDA *INPPTR      ;get temporary
BCDC- A4 41    LDY *INPPTR+1   ;READ pointer in YA
BCDE- A6 0B    LDX *INPFLG      ;get flag
BCEO- 10 03    BPL VARYO       ;if READ
BCE2- 4C C1 B7  JMP RESFIN      ;go do next statement
BCE5- A0 00    VARYO LDY #$00       ;if INPUT(#) or GET
BCE7- B1 40    LDA (INPPTR),Y  ;get character
BCE9- F0 0B    BEQ INPRTS      ;if not end of line
BCEB- A5 10    LDA *CHANNL      ;and CMD output file#
BCED- D0 07    BNE INPRTS      ;is default
BCEF- A9 F7    LDA #L,EXIGNT    ;then
BCF1- A0 BC    LDY #H,EXIGNT    ;point to EXTRA IGNORED
BCF3- 4C 1D BB  JMP STROUT      ;and print that string
BCF6- 60      INPRTS RTS       ;else exit to caller
        ;*****  

        ;* *
        ;* Message: ?EXTRA IGNORED  *
        ;* *
        ;*****  

BCF7- 3F 45 58  EXIGNT .BY '?EXTRA IGNORED' $00 $00
BCFA- 54 52 41
BCFD- 20 49 47
BD00- 4E 4F 52
BD03- 45 44 0D
BD06- 00
        ;*****  

        ;* *
        ;* Message: ?REDO FROM START  *
        ;* *
        ;*****  

BD07- 3F 52 45  TRYAGN .BY '?REDO FROM START' $00 $00
BD0A- 44 4F 20
BD0D- 46 52 4F
BD10- 40 20 53
BD13- 54 41 52
BD16- 54 00 00
        ;*****  

        ;* *
        ;* Perform NEXT.          *
        ;* *
        ;*****  

BD19- D0 04    NEXT  BNE GETFOR     ;if no variable
BD1B- A0 00    LDY #$00       ;set pointer to variable
BD1D- F0 03    BEQ STXFOR      ;invalid
BD1F- 20 2B C1 GETFOR JSR PTRGET   ;else get pointer to variable
BD22- 85 46    STXFOR STA *FORPNT   ;store it
BD24- 84 47    STY *FORPNT+1  ;in ($46)
BD26- 20 22 B3  JSR FNDFOR      ;check for FOR block on stack
BD29- F0 04    BEQ HAVFOR      ;if not found
BD2B- A2 00    LDX #$00       ;point to NEXT WITHOUT FOR
BD2D- F0 66    ERRG05 BEQ ERRG04   ;and give error
BD2F- 9A      HAVFOR TXS       ;else load stackpointer
BD30- 8A      TXA           ;and get it in A
BD31- 18      CLC           ;prepare for add
BD32- 69 04    ADC #$04       ;point to STEP mantissa
BD34- 48      PHA           ;save pointer
BD35- 69 06    ADC #$06       ;point to T0 value
BD37- 85 21    STA *INDEX2    ;save that pointer too
BD39- 68      PLA           ;reget STEP pointer to
BD3A- A0 01    LDY #$01       ;and hi (stack area!)
BD3C- 20 D8 CC  JSR MOVFM      ;load FLP accu with step
BD3F- BA      TSX           ;get stackpointer again

```

```

BD40- BD 09 01      LDA $0109,X      ;get sign of STEP
BD43- 85 63          STA *FACSGN    ;in FLP accu
BD45- A5 46          LDA *FORPNT    ;get pointer to FOR variable
BD47- A4 47          LDY *FORPNT+1 ;in YA
BD49- 20 90 C9      JSR FADD      ;add STEP to variable
BD4C- 20 06 CD      JSR MOVVF     ;and store new value in variable
BD4F- A0 01          LDY #$01      ;get hi-byte again
BD51- 20 93 CD      JSR FCOPMN    ;subtract T0 value
BD54- BA             TSX           ;get stackpointer again
BD55- 38             SEC           ;prepare for subtract
BD56- FD 09 01      SBC $0109,X  ;if signs equal
BD59- F0 17          BEQ LOOPDN    ;loop finished, do next statement
BD5B- BD 0F 01      LDA $010F,X  ;else get FOR-line# lo
BD5E- 85 36          STA *CURLIN   ;in current line#
BD60- BD 10 01      LDA $0110,X  ;get hi also
BD63- 85 37          STA *CURLIN+1
BD65- BD 12 01      LDA $0112,X  ;get CHRGET pointer for FOR, lo
BD68- 85 77          STA *TXTPTR   ;in CHRGET
BD6A- BD 11 01      LDA $0111,X  ;get hi also
BD6D- 85 78          STA *TXTPTR+1 ;and complete CHRGET
BD6F- 4C 4A B7      NEWSGO JMP NEWSTT ;and execute next (FOR) statement
BD72- 8A             LOOPDN TXA   ;get stackpointer in A
BD73- 69 11          ADC #$11      ;remove FOR block
BD75- AA             TAX           ;and load
BD76- 9A             TXS           ;stackpointer again
BD77- 20 76 00      JSR CHRGOT    ;get current character
BD7A- C9 2C          CMP #','
BD7C- D0 F1          BNE NEWSGO   ;go do next statement
BD7E- 20 70 00      JSR CHRGET    ;else get next character (variablename)
BD81- 20 1F BD      JSR GETFOR   ;and perform NEXT again
                                ;*****
                                ;*
                                ;* Get next non-string value. *
                                ;*
                                ;*****
BD84- 20 98 BD      FRMNUM JSR FRMEVL ;evaluate expression
                                ;*****
                                ;*
                                ;* Check if value is numeric. *
                                ;*
                                ;*****
BD87- 18             CHKNUM CLC      ;prepare for numeric test
BD88- 24             .BY $24      ;skip next instruction
                                ;*****
                                ;*
                                ;* Check if value is string *
                                ;* type.                 *
                                ;*
                                ;*****
BD89- 38             CHKSTR SEC      ;prepare for string test
BD8A- 24 07          CHKVAL BIT *VALTYP ;get variable type
BD8C- 30 03          BMI DOCSTR    ;if string, go test
BD8E- B0 03          BCS CHKERR   ;if numeric and string wanted, error
BD90- 60             CHKKOK RTS      ;else exit
BD91- B0 FD          DOCSTR BCS CHKKOK ;if string wanted, exit
BD93- A2 A3          CHKERR LDX #b260-ERRTAB ;else point to TYPE MISMATCH
BD95- 4C CF B3      ERRG04 JMP ERROR  ;and give error, restart
                                ;*****
                                ;*
                                ;* Evaluate any expression. *
                                ;*
                                ;*****
BD98- A6 77          FRMEVL LDX *TXTPTR ;get CHRGET pointer lo
BD9A- D0 02          BNE FRMEV1   ;if equal to zero

```

```

BD9C- C6 78      DEC *TXTPTR+1 ;adapt hi
BD9E- C6 77      FRMEV1 DEC *TXTPTR ;and lo
BDA0- A2 00      LDX #$00 ;get initial priority
BDA2- 24          .BY $24 ;skip next instruction
BDA3- 48          LPOPER PHA ;save previous priority
BDA4- 8A          TXA ;get current priority in A
BDA5- 48          PHA ;save it
BDA6- A9 01      LDA #$01 ;check if there is
BDA8- 20 93 B3      JSR GETSTK ;space for 2 bytes on stack
BDA9- 20 81 BE      JSR EVAL ;get value in FLP #1
BDAE- A9 00      LDA #$00 ;clear
BDB0- 85 4A      STA *OPMASK ;symbol check flag
BDB2- 20 76 00    TSTOP JSR CHRGOT ;get current character
BDB5- 38          LOPREL SEC ;prepare for subtract
BDB6- E9 B1      SBC #GREATK ;subtract token for '>'
BDB8- 90 17      BCC ENDREL ;if operator, go handle operator
BDBA- C9 03      CMP #$03 ;if higher than '>'
BDBC- B0 13      BCS ENDREL ;go do function
BDBE- C9 01      CMP #$01 ;else set carry if '>'
BDC0- 2A          ROL A ;get carry in A
BDC1- 49 01      EOR #$01 ;bit0='<', bit1='=', bit2='>'
BDC3- 45 4A      EOR *OPMASK ;add to flag
BDC5- C5 4A      CMP *OPMASK ;if that bit set already
BDC7- 90 61      BCC SNERRS ;do SYNTAX ERROR
BDC9- 85 4A      STA *OPMASK ;else store new flag
BDCB- 20 70 00    JSR CHRGET ;get next character
BDCE- 4C B5 BD    JMP LOPREL ;and loop
BDD1- A6 4A      ENDREL LDX *OPMASK ;other than <=> found, get flags
BDD3- .D0 2C      BNE FINREL ;if <, = or >, perform that
BDD5- B0 ?F      BCS QOP ;if function, 'go perform it
BDD7- 69 07      ADC #$07 ;else calculate operator index in A
BDD9- 90 78      BCC QOP ;if STEP or lower go perform that
BDD8- 65 07      ADC *VALTYP ;else if stringtype and '+'
BDD0- D0 03      BNE bde2 ;go perform string concatenation
BDDF- 4C 4F C7    JMP CAT ;else decrement index
BDE2- 69 FF      bde2 ADC #$FF ;save it
BDE4- 85 1F      STA *INDEX ;multiply it by two
BDE6- 0A          ASL A ;add original (times three)
BDE7- 65 1F      ADC *INDEX ;and put in Y (index in table)
BDE9- A8          TAY ;get current priority back
BDEA- 68          QPREC PLA ;if current priority
BDEB- D9 94 B0    CMP OPTAB,Y ;is higher, perform it
BDEE- B0 6B      BCS QCHNUM ;else check if numeric type
BDF0- 20 87 BD    JSR CHKNUM ;save priority again
BDF3- 48          DOPREC PHA ;stack operation for later processing
BDF4- 20 1A BE    NEGPRC JSR DOPRE1 ;get current priority back
BDF7- 68          PLA ;if not at start of expression
BDF8- A4 48      LDY *OPPTR ;check priorities
B DFA- 10 17      BPL QPREC1 ;if begin of expression, get priority
BDFC- AA          TAX ;if initial, stop evaluation
BDFD- F0 5A      BEQ QOPGO ;else retrieve result and loop
Bdff- D0 63      BNE PULSTK ;*****
                                ;*
                                ;* Perform <, = or >. *
                                ;*
                                ;*****
BEO1- 46 07      FINREL LSR *VALTYP ;get var. type in carry
BEO3- 8A          TXA ;and get flags in A
BEO4- 2A          ROL A ;add type to A
BEO5- A6 77      LDX *TXTPTR ;get CHRGOT pointer lo
BEO7- D0 02      BNE FENRE2 ;if equal to zero
BEO9- C6 78      DEC *TXTPTR+1 ;adapt hi byte
BEOB- C6 77      FENRE2 DEC *TXTPTR ;decrement CHRGOT pointer

```

```

BE00- A0 1B LDY #PTDORL-OPTAB ;point to <=> in table
BE0F- 85 4A STA *OPMASK ;store flag and type
BE11- D0 07 BNE QPREC ;and perform <,= or >
;*****
;* *
;* Check for current priority. *
;* *
;*****
BE13- D9 94 B0 QPREC1 CMP OPTAB,Y ;if current priority
BE16- B0 4C BCS PULSTK ;is higher, perform it
BE18- 90 09 BCC DOPREC ;else loop
.FI "BASIC4.DB.M12"

```

1903 33FE-4001 BASIC4.DB.M12

```

;name BASIC4.DB.M12
;*****
;* *
;* Save address of operation and *
;* FLP accu1 on stack and *
;* evaluate further. *
;* *
;*****
BE1A- B9 96 B0 DOPRE1 LDA OPTAB+2,Y ;get address hi of operator
BE1D- 48 PHA ;push it on stack
BE1E- B9 95 B0 LDA OPTAB+1,Y ;do the same
BE21- 48 PHA ;for the lo-byte
BE22- 20 20 BE JSR PUSHF1 ;push result so far on stack
BE25- A5 4A LDA *OPMASK ;get flag back
BE27- 4C A3 B0 JMP LOPER ;and loop
;*****
;* *
;* SYNTAX ERROR exit. *
;* *
;*****
.BE2A- 4C 00 BF SNERRS JMP SNERR ;go do SYNTAX ERROR
;*****
;* *
;* Push FLP accu1 (result so *
;* far) on stack. *
;* *
;*****
BE20- A5 63 PUSHF1 LDA *FACSGN ;get sign of FLP accu
BE2F- BE 94 B0 LDX OPTAB,Y ;get priority of operator in X
BE32- A8 PUSHF TAY ;save sign in Y
BE33- 68 PLA ;get return address lo
BE34- 85 1F STA *INDEX ;save in $1F
BE36- 68 PLA ;get hi also
BE37- 85 20 STA *INDEX+1 ;complete address in ($1F)
BE39- 98 TYA ;get sign back
BE3A- 48 PHA ;save on stack
BE3B- E6 1F INC *INDEX ;calculate
BE3D- D0 02 BNE FORPSH ;correct
BE3F- E6 20 INC *INDEX+1 ;return address in ($1F)
BE41- 20 51 CD FORPSH JSR ROUND ;round FLP accu1
BE44- A5 62 LDA *FACLO ;and
BE46- 48 PHA ;push
BE47- A5 61 LDA *FACMO ;FLP
BE49- 48 PHA ;accu1
BE4A- A5 60 LDA *FACMOH ;on
BE4C- 48 PHA ;stack
BE4D- A5 5F LDA *FACHO
BE4F- 48 PHA

```

```

BE50- A5 SE      LDA *FACEXP
BE52- 48          PHA
BE53- 6C 1F 00    JMP (INDEX)      ;then return to caller
;*****  

;*          *  

;* Flag start of expression and *  

;* check if at initial priority. *  

;* If so, do not load FLP2 with *  

;* result so far.             *  

;*          *  

;*****  

BE56- A0 FF      QOP    LDY #$FF      ;flag start of expression
BE58- 68          PLA    ;get priority back
BE59- F0 23      QOPGO BEQ QOPRTS   ;if at initial, do not load FLP2
BE5B- C9 64      QCHNUM CMP #$64     ;if not priority for <=>
BE5D- F0 03      BEQ UNPSTK
BE5F- 20 87 BD    JSR CHKNUM   ;check if numeric type
;*****  

;*          *  

;* Get result so far in FLP      *  

;* accu2 (from stack) and per-  *  

;* form operation.            *  

;*          *  

;*****  

BE62- 84 48      UNPSTK STY *OPPTR  ;save index in table
BE64- 68          PULSTK PLA      ;get flag and type back
BE65- 4A          LSR A       ;calculate flag
BE66- 85 DC      STA *DOMASK  ;and save it
BE68- 68          PLA      ;then
BE69- 85 66      STA *ARGEXP  ;load
BE6B- 68          PLA      ;FLP
BE6C- 85 67      STA *ARGHO   ;accu2
BE6E- 68          PLA      ;from
BE6F- 85 68      STA *ARGMOH  ;stack
BE71- 68          PLA
BE72- 85 69      STA *ARGMO
BE74- 68          PLA
BE75- 85 6A      STA *ARGLO
BE77- 68          PLA
BE78- 85 6B      STA *ARGSGN
BE7A- 45 63      EOR *FACSGN  ;get EOR of signs
BE7C- 85 6C      STA *ARISGN  ;in $6C
BE7E- A5 SE      QOPRTS LDA *FACEXP ;get exponent of FLP accu1
BE80- 60          UNPRTS RTS      ;and perform operation or exit
;*****  

;*          *  

;* Get value of constant or      *  

;* variable in FLP accu1.        *  

;*          *  

;*****  

BE81- A9 00      EVAL   LDA #$00      ;set variable type
BE83- 85 07.     STA *VALTYP  ;to numeric
BE85- 20 70 00    EVAL0  JSR CHRGET  ;get next character
BE88- B0 03      BCS EVAL2   ;if numeric
BE8A- 4C 29 CE    EVAL1  JMP FIN     ;convert to number in FLP1, and exit
BE8D- 20 B6 C1    EVAL2  JSR ISLETC  ;if variable, test if same type
BE90- B0 7B      BCS ISVJMP  ;if same, get value in FLP1
BE92- C9 FF      CMP #PI     ;if not PI
BE94- D0 0F      BNE QDOT   ;go check for other char's
BE96- A9 A0      LDA #L,PIVAL ;else get pointer to
BE98- A0 BE      LDY #H,PIVAL ;value of PI
BE9A- 20 D8 CC    JSR MOVFM   ;get it in FLP accu1
BE9D- 4C 70 00    JMP CHRGET  ;and get next character
;*****  


```

```

        ;*
        ;* PI as FLP number.          *
        ;* (Value= 3.1415791064)      *
        ;*                         *
        ;*****  

BEAD- 82      PIVAL  .BY $82           ;exponent
BEA1- 49      .BY $49           ;mantissa, MSB
BEA2- 0F      .BY $0F           ;mantissa
BEA3- DA      .BY $DA           ;mantissa
BEA4- A1      .BY $A1           ;mantissa, LSB
        ;*****  

        ;*
        ;* Check for other characters *
        ;* than variable names or    *
        ;* numerals.                 *
        ;*                         *
        ;*****  

BEA5- C9 2E    QDOT   CMP #'.'       ;if period
BEA7- F0 E1    BEQ EVAL1        ;go get constant in FLP accu1
BEA9- C9 AB    CMP #MINUTK     ;if token for -
BEAB- F0 58    BEQ DOMIN        ;perform unary minus
BEAD- C9 AA    CMP #PLUSTK     ;if token for +
BEAF- F0 D4    BEQ EVAL0        ;skip and get constant
BEB1- C9 22    CMP #'/'        ;if not string delimiter
BEB3- D0 0F    BNE EVAL3        ;check for NOT, FN or (
BEB5- A5 77    STRTXT LDA *TXTPTR  ;else get CHRGET's pointer
BEB7- A4 78    LDY *TXTPTR+1   ;in YA
BEB9- 69 00    ADC #$00        ;add carry
BEBB- 90 01    BCC STRTX2      ;if page crossed
BEBD- C8      INY              ;adapt hi
BEBE- 20 B0 C5  STRTX2 JSR STRLIT  ;set up string in memory
BEC1- 4C 18 C9  JMP ST2TXT      ;and point CHRGET after constant
BEC4- C9 A8    EVAL3  CMP #NOTTK     ;if token for NOT
BEC6- D0 13    BNE EVAL4        ;and point to NOT in table
BEC8- A0 18    LDY #NOTTAB-OPTAB ;point to NOT in table
Beca- D0 3B    BNE GONPRC      ;and perform NOT
        ;*****  

        ;*                         *
        ;* Perform NOT.            *
        ;*                         *
        ;*****  

BECC- 20 EA C2  NOTOP  JSR AYINT    ;convert FLP1 to 2 integers
BECF- A5 62    LDA *FACLO      ;get LSB of FLP accu1
BED1- 49 FF    EOR #$FF        ;invert it
BED3- A8      TAY              ;get it in Y
BED4- A5 61    LDA *FACMO      ;get MSB of FLP accu1
BED6- 49 FF    EOR #$FF        ;invert that too
BED8- 4C BC C4  JMP GIVAYF      ;and convert integer to floating
        ;*****  

        ;*                         *
        ;* Test for other tokens.  *
        ;*                         *
        ;*****  

BEDB- C9 A5    EVAL4  CMP #FNTK     ;if token for FN
BEDD- D0 03    BNE bee2        ;and point to FN
BEDF- 4C 10 C5  JMP FNDOER      ;go expand function
BEE2- C9 B4    bee2  CMP #ONEFUN    ;if token for function
BEE4- 90 03    BCC PARCHK      ;and point to function
BEE6- 4C 47 C0  JMP ISFUN        ;go perform function
BEE9- 20 F2 BE  PARCHK JSR CHKOPN    ;test for (, if not SYNTAX ERROR
BEEC- 20 98 BD  PARCHK JSR FRMEVL    ;else evaluate expression in ()
        ;*****  

        ;*                         *
        ;* Test if current character is *

```

```

        ;* closing parenthesis. If not      *
        ;* give SYNTAX ERROR.           *
        ;*
        ;*****  

BEEF- A9 29     CHKCLS LDA #'')'      ;get closing parenthesis
BEF1- 2C         .BY $2C          ;skip next instruction
        ;*****  

        ;*
        ;* Test if current character is  *
        ;* an open parenthesis. If not   *
        ;* give SYNTAX ERROR.          *
        ;*
        ;*****  

BEF2- A9 28     CHKOPN LDA #'('      ;get open parenthesis
BEF4- 2C         .BY $2C          ;skip next instruction
        ;*****  

        ;*
        ;* Test if current character is  *
        ;* a comma. If not give SYNTAX  *
        ;* ERROR.                      *
        ;*
        ;*****  

BEF5- A9 2C     CHKCOM LDA #','
                .FI "BASIC4.OB.M13"

```

1712 33FE-4B10 BASIC4.OB.M13

```

        ;name BASIC4.OB.M13
        ;*****  

        ;*
        ;* Test if current character is  *
        ;* the character in A. If not,   *
        ;* give SYNTAX ERROR.          *
        ;*
        ;*****  

BEF7- A0 00     SYNCRR LDY #$00       ;get offset zero
BEF9- D1 77     CMP (TXTPTR),Y    ;if current char. not equal to A
BEFB- D0 03     BNE SNERR        ;give SYNTAX ERROR
BEFD- 4C 70 00  JMP CHRGET        ;else read next character
        ;*****  

        ;*
        ;* Exit with SYNTAX ERROR and  *
        ;* restart BASIC.              *
        ;*
        ;*****  

BF00- A2 10     SNERR  LDX #b21d-ERRTAB ;point to SYNTAX ERROR
BF02- 4C CF B3  JMP ERROR         ;and go do error
        ;*****  

        ;*
        ;* Perform unary minus.        *
        ;*
        ;*****  

BF05- A0 15     DOMIN  LDY #NEGTAB-OPTAB ;point to unary minus in table
        ;*****  

        ;*
        ;* Entry for perform NOT.     *
        ;*
        ;*****  

BF07- 68        GONPRC PLA          ;remove return
BF08- 68        PLA             ;address
BF09- 4C F4 BD  JMP NEGPRC       ;and perform operation
        ;*****  

        ;*

```

```

        ;* Checksum byte (over B-ROM). *
        ;*
        ;*****  

BF0C- B8      CKSMBO .BY $B8          ;checksum byte
        ;*****
        ;*
        ;* Extra jump to reach get value *
        ;* of constant or variable via   *
        ;* relative branch.             *
        ;*
        ;*****  

BF0D- 4C 8C BF  ISVJMP JMP ISVAR      ;go get value
        ;
PABBO ;begin of patch area one
        ;*****
        ;*
        ;* Patch for set up backpointer *
        ;* if needed.                  *
        ;* (this code not in -19 ROM). *
        ;*
        ;*****  

BF10- 48      PATCHG PHA           ;save pointer hi
BF11- A5 0D    LDA *DSDESC        ;get length of DS$
BF13- F0 08    BEQ PTCHO         ;if zero, go do normal string
BF15- 68      PLA              ;else get pointer hi back
BF16- C5 0F    CMP *DSDESC+2    ;if pointer not for DS$
BF18- D0 04    BNE PTCH1         ;go do normal string
BF1A- 4C 70 BA JMP ADJXX        ;else go do DS$
BF1D- 68      PTCHO            PLA           ;get pointer hi back
BF1E- 4C 74 BA PTCH1            JMP ADJ02       ;and go do normal string
        ;*****
        ;*
        ;* Patch for execute next state-
        ;* ment.                      *
        ;* (this code not in -19 ROM). *
        ;*
        ;*****  

BF21- 20 70 00  PATCHH JSR CHRGET     ;get next character
BF24- C9 FF    CMP #PI           ;if code for PI
BF26- D0 03    BNE bf2b         ;go do
BF28- 4C 00 BF  JMP SNERR         ;SYNTAX ERROR (why not with BEQ?)
BF2B- 4C 76 00  bf2b            JMP CHRGOT      ;else get current character back
        ;*****
        ;*
        ;* Patch for INPUT, entered if
        ;* nothing has been input.
        ;* (Like RETURN only).
        ;* (this code not in -19 ROM).
        ;*
        ;*****  

BF2E- A5 10      PATCHI LDA *CHANNL    ;if current CMD file#
BF30- D0 03    BNE bf35         ;is default
BF32- 4C F1 BB  JMP PTHRTI      ;go end INPUT through END
BF35- A5 96      bf35            LDA *CSTAT      ;else get status byte
BF37- 29 40    AND #$40        ;if not end of file
BF39- D0 03    BNE bf3e         ;go and try get new input
BF3B- 4C 05 BB  JMP GETAGN      ;else perform 'READ'
BF3E- 4C 09 BC  bf3e            JMP INPCON      ;*****  

        ;*
        ;* Area $BF41-$BF8B filled with
        ;* $AA bytes.
        ;*
        ;*****  


```

```

.BA CKSM80+$80
;*****
;*                               *
;* Get value of variable.      *
;*                               *
;*****
BF8C- 20 2B C1 ISVAR JSR PTRGET      ;search for variable
BF8F- 85 61 ISVRET STA *FACMO      ;store pointer lo
BF91- 84 62 STY *FACLO          ;and hi
BF93- A5 42 LDA *VARNAME        ;get 1st character of name
BF95- A4 43 LDY *VARNAME+1      ;and second also
BF97- A6 07 LDX *VALTYP         ;and type
BF99- F0 39 BEQ G000           ;if numeric, if handle numeric
BF9B- A2 00 LDX #$00           ;else
BF9D- 86 60 STX *FACOV          ;clear overflow byte
BF9F- A6 62 LDX *FACLO          ;get pointer lo
BFA1- E0 8D CPX #L,CTIMR       ;if it points below jiffy clock
BFA3- 90 2E BCC STRRTS        ;exit (normal string variable)
BFA5- C9 54 CMP #'T'           ;else if 1st character is T
BFA7- D0 18 BNE ISVDS         ;and 2nd is 'I$'
BFA9- CO C9 CPY #$C9           ;then read and assign TI$
BFB0- D0 14 BNE ISVDS         ;clear exponent
BFAD- 20 03 CO JSR GETTIM        ;now Y=$FF
BFB0- 84 5B STY *TENEXP        ;and set initial index
BF82- 88 DEY                  ;set length to 6
BFB3- 84 6E STY *FBUFPT        ;get initial constant offset
BFB5- A0 06 LDY #$06           ;convert FLP accu to string
BFB7- 84 5A STY *DECCNT        ;and get descriptor in FLP accu
BF89- A0 24 LDY #$24           ;if 1st character is D
BFBB- 20 1E D0 JSR FOUTIM        ;then read DS$
BFBE- 4C 98 C5 JMP TIMSTR        ;get pointer to
BFC1- C9 44 ISVDS CMP #'D'           ;DS$
BFC3- D0 0E BNE STRRTS        ;and set it up in memory
BFC5- CO D3 CPY #$03           ;universal exit for normal strings
BFC7- D0 DA BNE STRRTS        ;get numeric type
BFC9- 20 FC BF JSR CHKDS          ;if floating, go handle FLP#
BFCC- A5 DE LDA *DSDESC+1      ;get offset
BFCE- A4 0F LDY *DSDESC+2      ;get lo byte of value
BFDD- 4C B0 C5 JMP STRLIT         ;in X
BFD3- 60 STRRTS RTS            ;adapt offset
BFD4- A6 08 G000 LDX *INTFLG        ;get hi in A
BFD6- 10 0D BPL G00000         ;and in Y
BFD8- A0 00 LDY #$00           ;get lo in A
BFDA- B1 61 LDA (FACMO),Y      ;and convert integer to floating
BFDC- AA TAX                  ;entry for floating, get pointer hi
BFDD- C8 INY                  ;if it points below jiffy clock
BFDE- B1 61 LDA (FACMO),Y      ;go assign
BFE0- A8 TAY                  ;else if 1st character is T
BFE1- 8A TXA                  ;and 2nd is I
BFE2- 4C BC C4 JMP GIVAYF        ;*****
BFE5- A6 62 G00000 LDX *FACLO        ;*
BFE7- E0 8D CPX #L,CTIMR       ;*
BFE9- 90 55 BCC GOMOVF          ;*
BFEB- C9 54 CMP #'T'           ;*
BFED- D0 20 BNE QSTATV          ;*
BFEF- CO 49 CPY #'I'           ;*
BFF1- D0 1C BNE QSTATV          ;*****
;*****
;*                               *
;* Evaluate TI.                *
;*                               *
;*****
BFF3- 20 03 CO bff3 JSR GETTIM        ;read time in FLP accu
BFF6- 98 TYA

```

BFF7- A2 A0	LDX #\$AO	;get exponent
BFF9- 4C 85 CD	JMP FLOATB	;and convert to floating

	;*	
	;* Evaluate DS\$.	
	;*	
	;*****	
BFFC- A5 0D	LDA *DSDESC	;get length of DS\$
BFFE- D0 D3	BNE STRRTS	;if not zero, exit
C000- 4C BD FF	JMP READDS	;else read DS\$ and set it up
	.FI "BASIC4.OB.M14"	

1137 33FE-4535 BASIC4.OB.M14

```

;name BASIC4.OB.M14
*****
;*
;* Differences between the *
;* 1465-23 and the 1465-19 ROMs. *
;*
;* The code presented in the *
;* modules 1 through 13 is for *
;* the ROM 1465-23. This ROM was *
;* the second release for the *
;* B-ROM. The original -19 ROM *
;* had a few bugs, which are *
;* shown here. *
;*
;* The patch area ($BF0E-$BF40) *
;* is filled with $AA bytes in *
;* the 1465-19 ROM. *
;*
;* The following routines are *
;* different: *
;*
*****
```

.BA NEWSTT

```

;*****
;*
;* Start BASIC: execute next *
;* statement. *
;*
*****
```

B74A- 20 E1 FF JSR ISCNTC ;check for STOP key

B74D- A5 77 LDA *TXTPTR ;get CHRGET's pointer

B74F- A4 78 LDY *TXTPTR+1

B751- C0 02 CPY #2 ;if not in direct mode

B753- F0 04 BEQ DIRCON

B755- 85 3A STA *OLDTXT ;save pointer

B757- 84 3B STY *OLDTXT+1 ;for CONT

B759- A0 00 LDY #0 ;get offset

B75B- B1 77 LDA (TXTPTR),Y ;get current character

B75D- D0 46 BNE MORSTS ;if not end of statement, execute

B75F- A0 02 LDY #2 ;point to link hi

B761- B1 77 LDA (TXTPTR),Y ;get it

B763- 18 CLC ;if not end of program

B764- D0 03 BNE DIRCN1 ;go execute next statement in prg

B766- 4C E2 B7 JMP ENDCON ;else perform END

B769- C8 INY ;point to line# lo

B76A- B1 77 LDA (TXTPTR),Y ;get it

B76C- 85 36 STA *CURLIN ;store it

B76E- C8 INY

B76F- B1 77 LDA (TXTPTR),Y ;do same for hi

```

B771- 85 37 STA *CURLIN+1
B773- 98 TYA ;get current offset
B774- 65 77 ADC *TXTPTR ;adapt chrget's
B776- 85 77 STA *TXTPTR ;pointer
B778- 90 02 BCC GONE ;if page crossed
B77A- E6 78 INC *TXTPTR+1 ;adapt hi too
B77C- 20 70 00 JSR CHRGET ;get next character
B77F- 20 85 B7 JSR GONE3 ;execute statement in A
;BUG: if character was PI, this leads
;      to offset $AE, which executes from
;      address $64BF on. This will usually
;      end in a crash.
B782- 4C 4A B7 JMP NEWSTT ;and execute next statement
.BA STRADJ
;*****
;*
;* Set up pointer for backpoint- *
;* ter if needed. If set up,   *
;* carry flag is set.          *
;* On entry, ($1F) must point to *
;* the descriptor of the string. *
;*
;*****
LDY #0 ;point to length
LDA (INDEX),Y ;get it
PHA ;save on stack, if zero
BEQ ADJ01 ;exit with carry clear (no string)
INY ;point to pointer lo
LDA (INDEX),Y ;get it
TAX ;save it X
INY ;point to pointer hi
LDA (INDEX),Y ;get it, if it points to nonsense
BMI ADJ01 ;exit with carry clear
CMP #H,ROMLOC ;if it points below ROM
BCS ADJ01 ;exit with carry clear
CMP *FRET0P+1 ;if not in string area
BCC ADJ01 ;exit with carry clear
BNE ADJ ;not in area, go check for DS$
CPX *FRET0P ;hi's equal, compare lo's
BCC ADJ01 ;if not in area, exit with carry clear
CMP *DSDESC+2 ;if pointer hi of DS$
BNE ADJ02
CPX *DSDESC+1 ;and if pointer lo of DS$
BEQ ADJ01 ;exit with carry clear
;BUG: if DS$ has length zero, diskdrive
;      is not present. Because length is not
;      tested, the system tries to read DS$,
;      which leads to a DEVICE NOT PRESENT
;      error, and stops execution of the program.
STX *INDEX ;else set up pointer lo
STA *INDEX+1 ;and hi
PLA ;get length back
TAX ;save in X
CLC ;prepare for add
ADC *INDEX ;calculate address of backpointer
STA *INDEX ;in ($1F)
BCC ADJ00 ;if page crossed
INC *INDEX+1 ;adapt hi byte
SEC ;and exit with carry set
RTS
PLA ;clean stack (get 'length')
CLC ;clear carry
RTS ;and exit
.BA BUFFUL

```

```

;*****
;*          *
;* Test if any input given.      *
;* If none, restart BASIC with  *
;* 'READY.'.                   *
;*          *
;*****  

BEE8- AD 00 02    LDA BUF           ;if any input
BEBB- D0 1C       BNE INPCON        ;perform 'READ'
BBED- A5 10       LDA *CHANNL       ;if current CMD file#
BBEF- D0 E4       BNE GETAGN        ;is not default, loop
;BUG: does not test for end of file
;      on IEEE or cassette.
;      If end of file missed, will read
;      in next value, although this was
;      not asked for. If file exhausted,
;      loops forever, without the possibility
;      to stop with the STOP key.  

BFF1- 18          CLC              ;else prevent BREAK message (select END
BFF2- 4C D8 B7   JMP STPEND        ;and perform END
.FI "BASIC4.OB.M15"

```

17CB 33FE-4BC9 BASIC4.OB.M15

```

;name BASIC4.OB.M15
;*****
;*          *
;* External labels, part 1.  *
;*          *
;*****  

USRPOK .DE $0000      ;vector for USR
CHARAC .DE $0003      ;end delimiter in get end of statement
INTEGR .DE $0003      ;integer read by QINT
ENDCHR .DE $0004      ;end delimiter in LIST, CRUNCH
COUNT .DE $0005      ;general counter for BASIC
VALTYP .DE $0007      ;variable type: numeric or string
INTFLG .DE $0008      ;numeric type: integer or floating
DORES .DE $0009      ;flag: print tokens as keywords
SUBFLG .DE $000A      ;flag: arrays and integers allowed
INPFLG .DE $000B      ;flag: type of input
DOMASK .DE $000C      ;mask used by <,=,> operators
DSDESC .DE $000D      ;descriptor of DS$
CHANNL .DE $0010      ;current CMD output file#
LINNUM .DE $0011      ;(line)number read by LINGET
TEMPPT .DE $0013      ;current descriptor stack pointer
TEMPST .DE $0016      ;top of descriptor stack
INDEX .DE $001F        ;indirect index pointer
INDEX1 .DE $001F       ;indirect index pointer #1
INDEX2 .DE $0021       ;indirect index pointer #2
TXTTAB .DE $0028       ;pointer: start of program text
VARTAB .DE $002A       ;pointer: start of variables
ARYTAB .DE $002C       ;pointer: start of arrays
STREND .DE $002E       ;pointer: end of arrays
FRETOP .DE $0030       ;pointer: start of real strings
FRESPC .DE $0032       ;pointer: top of free string space
MEMSIZ .DE $0034       ;pointer: top of RAM
CURLIN .DE $0036       ;current line number
OLDLIN .DE $0038       ;last line number (for CONT)
OLDXTX .DE $003A       ;old TXTPTR (for CONT)
DATLIN .DE $003C       ;current DATA line
DATPTR .DE $003E       ;pointer to current DATA statement
INPPTR .DE $0040       ;pointer to inputted value
VARNAM .DE $0042       ;current variable name

```

FORPNT .DE \$0046	;current FOR variable
LSTPNT .DE \$0046	;pointer for LIST
OPPTR .DE \$0048	;pointer in table
VARTXT .DE \$0048	;index to current operator
OPMASK .DE \$004A	;mask created by current operator
DSCPNT .DE \$004D	;pointer to descriptor
TEMPPF1 .DE \$0054	;temporary FLP accu
HIGHDS .DE \$0055	;destination pointer in BLT
HIGHTR .DE \$0057	;source pointer in BLT
DECCTN .DE \$005A	:# of digits before decimal point
TENEXP .DE \$005B	;exponent base ten in FOUT
LOWTR .DE \$005C	;pointer for LIST
FAC .DE \$005E	;floating point accu #1
FACEXP .DE \$005E	;exponent of FLP accu #1
FACHO .DE \$005F	;MSB mantissa of FLP accu #1
FACMOH .DE \$0060	;mantissa, upper middle, of FLP accu #1
FACMO .DE \$0061	;mantissa, lower middle, of FLP accu #1
FACLO .DE \$0062	;LSB mantissa of FLP accu #1
FACSGN .DE \$0063	;sign of FLP accu #1
ARGEXP .DE \$0066	;exponent of FLP accu #2
ARGHO .DE \$0067	;MSB mantissa of FLP accu #2
ARGMOH .DE \$0068	;mantissa, upper middle, of FLP accu #2
ARGMO .DE \$0069	;mantissa, lower middle, of FLP accu #2
ARGLO .DE \$006A	;LSB mantissa of FLP accu #2
ARGSGN .DE \$006B	;sign of FLP accu #2
ARISGN .DE \$006C	;sign comparison of FLP accus
STRNG1 .DE \$006C	;pointer to a string
FACOV .DE \$006D	;overflow byte of FLP accu #1
BUFPTR .DE \$006E	;index save in CRUNCH
STRNG2 .DE \$006E	;2nd pointer to a string
FBUFPT .DE \$006E	;index in output in FOUT
CHRGET .DE \$0070	;gets next character from program text
CHRGOT .DE CHRGET+6	;gets current character from program te
TXTPTR .DE CHRGET+7	;pointer to current character
QNUM .DE CHRGET+13	;label in CHRGET
CTIMR .DE \$0080	;jiffy clock
CSTAT .DE \$0096	;I/O STatusbyte
TRMPOS .DE \$00C6	;column number of cursor
PI .DE \$00FF	;character for PI
STKEND .DE \$01FB	;top of stack for BASIC
BUF .DE \$0200	;BASIC's input buffer
GETTIM .DE \$C003	;get value of TI\$
QSTATV .DE \$C00F	;check for special variables
GOMOVF .DE \$C040	;exit for normal variable
ISFUN .DE \$C047	;perform arithmetic function
OROP .DE \$C086	;perform OR
ANDOP .DE \$C089	;perform AND
DOREL .DE \$C0B6	;perform comparison (<,>)
DIM .DE \$C121	;perform DIM
PTRGET .DE \$C12B	;get name and pointer of current variab
ISLETC .DE \$C1B6	;set carry if A holds letter
AYINT .DE \$C2EA	;convert FLP# to integer in (\$61)
FCERR .DE \$C373	;ILLEGAL QUANTITY ERROR exit
FRE .DE \$C4A8	;perform FRE
GIVAYF .DE \$C4BC	;convert YA to FLP
POS .DE \$C4C9	;perform POS
ERRDIR .DE \$C4CF	;check for program mode
DEF .DE \$C4DC	;perform DEF
FNDOER .DE \$C51D	;evaluate FN
STRD .DE \$C58E	;perform STR\$
TIMSTR .DE \$C598	;move string at \$100 to memory
STRINI .DE \$C59E	;allocate pointer and length of new str
STRLIT .DE \$C5B0	;set up string in memory
STRLT2 .DE \$C5B6	;same, with other delimiters

```

GARBA2 .DE $C66A      ;collect all garbage strings
CAT    .DE $C74F      ;perform string concatenation
MOVINS .DE $C78C      ;store string in hi RAM
FREFAC .DE $C7B8      ;discard temporary string
FRETMIS .DE $C811     ;clean descriptor stack
CHRD   .DE $C822      ;perform CHR$
LEFTD  .DE $C836      ;perform LEFT$
RIGHTD .DE $C862      ;perform RIGHT$
MIDD   .DE $C86D      ;perform MID$
LEN    .DE $C8B2      ;perform LEN
ASC    .DE $C8C1      ;perform ASC
GTBYTC .DE $C8D1      ;input and evaluate a 1-byte number
GETBYT .DE $C8D4      ;same, but CHRGET called already
VAL    .DE $C8E3      ;perform VAL
ST2TXT .DE $C918      ;get pointer to string in CHRGET
PEEK   .DE $C943      ;perform PEEK
POKE   .DE $C95A      ;perform POKE
FNWAIT .DE $C963      ;perform WAIT
FSUBT  .DE $C989      ;perform subtraction
FADD   .DE $C99D      ;perform addition
FADDT  .DE $C9A0      ;add to zero
FONE   .DE $CAF2      ;table of constants
LOG    .DE $CB20      ;perform LOG
FMULTT .DE $CB61      ;multiply FLP accu1 with FLP accu2
MUL10  .DE $CC18      ;multiply FLP accu1 by 10
FINML6 .DE $CC23      ;same
FDIVT  .DE $CC48      ;perform division
MOVFM  .DE $CCD8      ;load FLP accu1 from memory
MOVVF  .DE $CD06      ;store FLP accu1 in memory
MOVAF  .DE $CD42      ;round and copy FLP accu1 to FLP accu2
ROUND  .DE $CD51      ;round FLP accu1
SIGN   .DE $CD61      ;find sign of FLP accu1
SGN    .DE $CD6F      ;perform SGN
FLOATB .DE $CD85      ;same
ABS    .DE $CD8E      ;perform ABS
FCOMPN .DE $CD93      ;compare FLP accu1 with FLP number
QINT   .DE $CD01      ;convert FLP accu1 to integer
INT    .DE $CE02      ;perform INT
FIN    .DE $CE29      ;convert ASCII string to # in FLP accu1
FINLOG .DE $CEB4      ;add A-to FLP accu1
INPRT  .DE $CF78      ;print IN plus a line number
LINPRT .DE $CF83      ;print XA in decimal on next line
FOUT   .DE $CF93      ;convert FLP accu1 to string
.FI "BASIC4.DB.M16"

```

089D 33FE-3C9B BASIC4.DB.M16

```

;name BASIC4.DB.M16
;*****
;*
;* External labels, part 2. *
;*
;*****
FOUTIM .DE $D01E      ;convert jiffy clock to TI$
ZERO   .DE $D009      ;dummy value for zero
SQR    .DE $D108      ;perform SQR
FPWRT  .DE $D112      ;perform power calculation (^)
NEGOP  .DE $D14B      ;negate FLP accu1
EXP    .DE $D184      ;perform EXP
RND    .DE $D229      ;perform RND
COS    .DE $D282      ;perform COS
SIN    .DE $D289      ;perform SIN
TAN    .DE $D202      ;perform TAN

```

ATN .DE \$D32C	;perform ATN
CONCAT .DE \$FF93	;perform CONCAT
DOPEN .DE \$FF96	;perform DOPEN
DCLOSE .DE \$FF99	;perform DCLOSE
RECORD .DE \$FF9C	;perform RECORD
FORMAT .DE \$FF9F	;perform HEADER
COLECT .DE \$FFA2	;perform COLLECT
BACKUP .DE \$FFA5	;perform BACKUP
DCOPY .DE \$FFA8	;perform COPY
APPEND .DE \$FFAB	;perform APPEND
DSAVE .DE \$FFAE	;perform DSAVE
DLOAD .DE \$FFB1	;perform DLOAD
DIRCAT .DE \$FFB4	;perform DIRECTORY
DCAT .DE \$FFB4	;perform CATALOG
RENAME .DE \$FFB7	;perform RENAME
SCRATC .DE \$FFBA	;perform SCRATCH
READDS .DE \$FFBD	;read and assign DS\$
COPEN .DE \$FFC0	;perform OPEN
CCLOS .DE \$FFC3	;perform CLOSE
COIN .DE \$FFC6	;set input device
COOUT .DE \$FFC9	;set output device
CLSCHN .DE \$FFCC	;restore default I/O
INCHR .DE \$FFCF	;get a character in A
OUTCH .DE \$FFD2	;print the character in A
CLOAD .DE \$FFD5	;perform LOAD
CSAVE .DE \$FFD8	;perform SAVE
CVERF .DE \$FFDB	;perform VERIFY
CSYS .DE \$FFDE	;perform SYS
ISCNTC .DE \$FFE1	;check for STOP key
CGETL .DE \$FFE4	;get 1 character
CCALL .DE \$FEF7	;close all files
FORTK .DE \$0081	;token for FOR
DATATK .DE \$0083	;token for DATA
GOTOTK .DE \$0089	;token for GOTO
GOSUTK .DE \$008D	;token for GOSUB
REMTK .DE \$008F	;token for REM
PRINTK .DE \$0099	;token for PRINT
TABTK .DE \$00A3	;token for TAB
TOTK .DE \$00A4	;token for TO
FNTK .DE \$00A5	;token for FN
SPCTK .DE \$00A6	;token for SPC
THENTK .DE \$00A7	;token for THEN
NOTTK .DE \$00A8	;token for NOT
STEPTK .DE \$00A9	;token for STEP
PLUSTK .DE \$00AA	;token for +
MINUTK .DE \$00AB	;token for -
GREATK .DE \$00B1	;token for >
EQULTK .DE \$00B2	;token for =
ONEFUN .DE \$00B4	;lowest token for function
GOTK .DE \$00CB	;token for GO
.EN	

END MAE PASS

LABELFILE

ABS =CD8E	ADDFRS =B348	ADDON =B886
ADJ =BA6C	ADJOO =BA83	ADJ01 =BA85
ADJ02 =BA74	ADJXX =BA70	AFFRTS =B5C7
ANDOP =C089	APPEND =FFAB	ARGEXP =0066
ARGHO =0067	ARGLO =006A	ARGMO =0069
ARGMOH =0068	ARGSGN =006B	ARISGN =006C

ARYTAB =002C	ASC =C8C1	ASPAC =BB0D
ATN =D32C	AYINT =C2EA	BACKUP =FFA5
BLT1 =B374	BLTC =B357	BLTLP =B380
BLTU =B350	BRKTXT =B31B	BUF =0200
BUFFUL =B8E8	BUFPTR =006E	CAT =C74F
CCALL =FFE7	CCLOS =FFC3	CGETL =FFE4
CHANNL =0010	CHARAC =0003	CHEAD =B4BF
CHKCLS =BEEF	CHKCOM =BEF5	CHKDS =BFFC
CHKERR =B093	CHKNUM =BD87	CHKOK =BD90
CHKOPN =BEF2	CHKSTR =BD89	CHKVAL =BD8A
CHRD =C822	CHRGET =0070	CHRGOT =0076
CKSMBO =BFOC	CLEAR =B5EE	CLEARC =B5F0
CLOAD =FFD5	CLSNCHN =FFCC	CMD =BA8E
CMPFOR =B33C	CMPSPC =B50D	COIN =FFC6
COLECT =FFA2	COLIS =B567	COMPRT =BAF0
CONCAT =FF93	CONT =B7EE	CONTRT =B807
COOUT =FFC9	COPEN =FFC0	COPFLT =B961
COPSTR =B964	COPY =B9F6	COPYOO =B.A2E
COPYO1 =BA44	COPYO2 =BA46	COPYC =BA13
COS =D282	COUNT =0005	CRDO =BADF
CRDONE =B599	CRFIN =BAED	CRTSKP =BB41
CRUNCH =B4FB	CSAV =FFD8	CSTAT =0096
CSYS =FFDE	CTIMR =008D	CURLIN =0036
CVERF =FFDB	CZLOOP =B4C7	DATA =B883
DATAJN =B891	DATATK =0083	DATBK =BC49
DATBK1 =BC4D	DATLIN =003C	DATLOP =BCB4
DATPTR =003E	DCAT =FFB4	DCLOSE =FF99
DCOPY =FFA8	DECBLT =B38B	DECCNT =D05A
DEF =C4DC	DIM =C121	DIRCAT =FFB4
DIRCN1 =B769	DIRCON =B759	DIRIS =B7E0.
DLOAD =FFB1	DNTCPY =B9EF	DOAGIN =BB6A
DOCO =B8D3	DOCOND =B8CB	DOCSTR =BD91
DOMASK =000C	DOMIN =BF05	DOPEN =FF96
DOPRE1 =BE1A	DOPREC =BDF3	DOREL =C0B6
DORES =0009	DSAVE =FFAE	DSCPNT =004D
DSDESC =000D	DSKX0 =B9BE	DSKX1 =B9D2
DSKX2 =B9D4	END =B7C8	ENDCHR =0004
ENDCON =B7E2	ENDREL =BDD1	EQLUTK =0082
ERR =B306	ERRCRD =B3DA	ERRDIR =C4CF
ERRFIN =B3F4	ERRG04 =BD95	ERRG05 =BD2D
ERROR =B3CF	ERRTAB =B20D	EVAL =BE81
EVAL0 =BE85	EVAL1 =BE8A	EVAL2 =BE8D
EVAL3 =BEC4	EVAL4 =BEDB	EXCHQT =B89C
EXIGNT =B0F7	EXP =D184	FAC =005E
FACEXP =005E	FACH0 =005F	FACLO =0062
FACMO =D061	FACMOH =0060	FACOV =006D
FACSGN =0063	FADD =C99D	FADDT =C9A0
FBUFPT =006E	FCERR =C373	FCERR2 =B9B2
FCOMP =CD93	FDIVT =CC48	FENRE2 =BEOB
FFLOOP =B327	FFRTS =B34F	FIN =CE29
FINI =B4AD	FININ1 =B4F8	FININL =BAD2
FINLOG =CEB4	FINML6 =CC23	FINREL =BEO1
FLINRT =B5D0	FLNRTS =B5D1	FLOAD =B60B
FLOATB =CD85	FMULTT =CB61	FNDFOR =B322
FNDLIN =B5A3	FNDLNC =B5A7	FNDL01 =B5B8
FNDOER =C51D	FNTK =00A5	FNWAIT =C963
FONE =CAF2	FOR =B6DE	FORMAT =FF9F
FORPNT =0046	FORPSH =BE41	FORTK =0081
FOUT =CF93	FOUTIM =D01E	FPWRT =D112
FRE =C4A8	FREFAC =C7B8	FRESPC =0032
FRETMS =C811	FRETOP =0030	FRMEV1 =BD9E
FRMEVL =B098	FRMNUM =BD84	FSUBT =C989
FUNDSP =B066	GARBA2 =C66A	GET =BB7A
GETAGN =BB05	GETBPT =B552	GETBYT =C8D4

GETDTL =B856	GETERR =B3E0	GETFOR =B01F
GETNTH =BC46	GETSPT =B9BA	GETSTK =B393
GETTIM =C003	GETTTY =BB91	GINLIN =BBFF
GIVAYF =C4BC	GLET =B7A2	GO =B7AC
GOLST =B638	GOMOVF =C040	GONE =B77C
GONE2 =B787	GONE3 =B785	GONE4 =B795
GONPRC =BF07	G000 =BF04	G00000 =BF05
GORDY =B7EB	GORTS =B85C	GOSUB =B813
GOSUTK =008D	GOTK =00CB	GOTNUM =B9B5
GOTO =B830	GOTOTK =0089	GREATK =00B1
GROODY =B6A8	GTBYTC =C8D1	HAVFOR =B02F
HIGHDS =0055	HIGHTR =0057	IF =B8B3
INCHR =FFCF	INDEX =001F	INDEX1 =001F
INDEX2 =0021	INLIN =B4E2	INLINC =B4E4
INLOOP =BC11	INPC01 =BC0B	INPCOM =B965
INPCON =BC09	INPFLG =000B	INPPTR =0040
INPRT =CF78	INPRTS =BCF6	INPUT =B8BE
INPUTN =B8A4	INT =CE02	INTEGR =0003
INTFLG =0008	INTXT =B300	IODONE =B8B4
IORELE =B8B6	ISCNTC =FFE1	ISCRTS =B7C5
ISFUN =CD47	ISLETC =C1B6	ISVAR =BF8C
ISVDS =BFC1	ISVJMP =BF00	ISVRET =BF8F
KLOOP =B501	KLOOP1 =B523	LOFONE =B727
LEFTD =C836	LEN =C8B2	LET =B930
LINGET =B8F6	LINNUM =0011	LINPRT =CF83
LIST =B630	LIST4 =B650	LNKPRG =B4B6
LNKRTS =B4E1	LOG =C820	LOOPDN =B072
LOPREL =B0B5	LOWTR =005C	LPOPER =BDA3
LSTEND =B64F	LSTPNT =0046	LUK4IT =B847
LUKALL =B84B	MAIN =B406	MAIN1 =B41F
MEMSIZ =0034	MIDD =C86D	MINUTK =00AB
MLOOP =B462	MORCO1 =BAF3	MOREN1 =B384
MORLIN =B8FC	MORSTS =B7A5	MOVAF =CD42
MOVFM =CCD8	MOVINS =C78C	MOVVF =CD06
MUL10 =CC18	MUSTCR =B52B	NEGOP =D14B
NEGPRC =B0F4	NEGTAB =B0A9	NEWCHR =BA05
NEWSGO =B06F	NEWSTT =B74A	NEXT =B019
NODATT =B569	NODEL =B470	NODELC =B48B
NOML6 =B992	NOTABR =B812	NOTOL =B6EF
NOTOP =BECC	NOTQTI =BBCD	NOTTAB =B0AC
NOTTK =00A8	NOWGE1 =BC79	NOWGET =BC6E
NOWLIN =BCCD	NTHIS =B580	NTHIS1 =B584
NTHIS2 =B580	NUMINS =BC85	NXTLGC =B92A
OKGOTO =B8C2	OLDLIN =0038	OLDTXT =003A
OMERR =B3CD	ONEFUN =00B4	ONEON =B73B
ONGLOP =B8E2	ONGLP1 =B8EA	ONGOTO =B8D6
ONGRTS =B8F5	OPMASK =004A	OPPTR =0048
OPTAB =B094	OROP =C086	OUTCH =FF02
OUTDO =B846	OUTQST =B844	OUTRTS =B84B
OUTSPC =B83A	PABBO =BF10	PARCHK =BEE9
PATCHG =BF10	PATCHH =BF21	PATCHI =BF2E
PEEK =C943	PI =00FF	PIVAL =BEAD
PL0OP =B687	PL0OP1 =B694	PLUSTK =00AA
POKE =C95A	POS =C4C9	PRINT =BAA8
PRINTC =BAAA	PRINTK =0099	PRINTN =BA88
PRIT3 =B6D4	PRIT3B =B6D5	PRIT4 =B683
PRTRTS =BAEF	PTCHO =BF10	PTCH1 =BF1E
PTD0RL =B0AF	PTHRTI =B8F1	PTRGET =C12B
PULSTK =BE64	PUSHF =BE32	PUSHF1 =BE20
QCHNUM =BESB	QDATA =BC3D	QDECT1 =B45A
QDOT =BEA5	QINLIN =BBF5	QINT =CDD1
QINTGR =B94D	QNUM =007D	QOP =BE56
QOPGO =BES9	QOPRTS =BE7E	QPL0P =B6AB
QPREC =BDEA	QPREC1 =BE13	QSTATV =COOF

QVARIA =B9E1	READ =BC02	READD\$ =FFBD
READY =B3FF	REARTS =B3CC	REASAV =B3AE
REASON =B3A0	REASTO =B3B9	RECORD =FF9C
REDDY =B312	REM =B8C6	REMER =B8A4
REMN =B894	REMRTS =B890	REMTK =008F
RENAME =FFB7	RESCON =B544	RESCR1 =B6C8
RESCR2 =B6CE	RESER =B53D	RESETC =BC6D
RESFIN =B7C1	RESLST =B0B2	RESRCH =B6C5
RESTOR =B7B7	RETU1 =B876	RETURN =B85D
RIGHTD =C862	RND =D229	ROMLOC =B000
ROUND =CD51	RUN =B808	RUNC =B5E9
RUNC2 =B827	SAVEIT =BA98	SCRATC =FFBA
SCRATH =B5D2	SCRTCH =B5D4	SETQUT =BC61
SGN =CD6F	SIGN =CD61	SIN =D289
SNERR =BF00	SNERR1 =B7A9	SNERR2 =B873
SNERR3 =B8DE	SNERR4 =B85E	SNERR5 =BE2A
SPCTK =00A6	SQR =D108	ST2TXT =C918
STCURL =B85A	STEPTK =00A9	STKEND =01FB
STKINI =B60E	STKRTS =B621	STMDSP =B000
STOLOP =B4A5	STOP =B7C6	STOPC =B7C9
STPEND =B708	STR1 =B570	STRADJ =BA4E
STRD =C58E	STRDN2 =BC8D	STRDON =BAA2
STREND =002E	STRINI =C59E	STRLIT =C5B0
STRLT2 =C5B6	STRNG =B579	STRNG1 =006C
STRNG2 =006E	STROUT =B81D	STRPR2 =B827
STRPRT =B820	STRRTS =BFD3	STRTX2 =BEBE
STRTXT =BE85	STUFFH =B554	STXFOR =B022
STXTPT =B622	SUBFLG =000A	SYNCHR =BEF7
TABER =BAFD	TABTK =00A3	TAN =D2D2
TEMPF1 =0054	TEMPPT =0013	TEMPST =0016
TENEXP =0058	THENTK =00A7	TIMELP =B978
TIMEST =B9A2	TIMNUM =B9AB	TIMSTR =C598
TOTK =00A4	TRMNO1 =B861	TRMNOK =BB4C
TRMOK =BC99	TRMPOS =00C6	TRYAGN =BD07
TRYMOR =B3AA	TSTDUN =B67A	TSTOP =BDB2
TXTPTR =0077	TXTTAB =0028	TYPLIN =B67C
UNPRTS =BE80	UNPSTK =BE62	USERR =B86E
USRLOC =B06C	USRPOK =0000	VAL =C8E3
VALTYP =0007	VAREND =BCDA	VARNAM =0042
VARTAB =002A	VARTXT =0048	VARYO =BCE5
XSPAC =B80E	XSPAC1 =B818	XSPAC2 =B80F
ZERO =D0D9	b21d =B21D	b223 =B223
b237 =B237	b242 =B242	b252 =B252
b25a =B25A	b267 =B267	b278 =B278
b285 =B285	b292 =B292	b2a2 =B2A2
b2b0 =B2B0	b2bd =B2BD	b2cc =B2CC
b2d5 =B2D5	b2e8 =B2E8	b2f6 =B2F6
b7f9 =B7F9	b80d =B80D	bde2 =BDE2
bee2 =BEE2	bf2b =BF2B	bf35 =BF35
bf3e =BF3E	bf13 =BFF3	
//0000,BBF5,BBF5		
]		

```
;name BASIC4.OC.CTL
;*****
;*          *
;*      C-ROM for BASIC 4.0      *
;*          *
;* Date 10-12-83  Time 00:25  *
;*          *
;* Made by:          *
;*          *
;* Nico de Vries          *
;* Mari Andriessenrade 49  *
;* 2907 MA Capelle a/d Yssel *
;* The Netherlands          *
;*          *
;* Original CBM ROM-number  *
;*          *
;*          1465-20          *
;*          *
;*****.BA $C003
.FI "B4C.M01"
```

19ED 338C-4079 B4C.M01

```
;name BASIC4.OC.M01
;*****
;*          *
;* This assembly starts at      *
;* $C003. The instruction left  *
;* out at $C000 is assembled    *
;* in the last module of the    *
;* B-ROM. This instruction is:  *
;*          *
;*****.c000 JMP READD$      ;go assign DS$
;*****
;*          *
;* Continuation of get value    *
;* of variable.                *
;* Read TI$.                   *
;*          *
;*****.c003- A9 8B   GETTIM LDA #L,CTIMR      ;point YA to
.C005- A0 00           LDY #H,CTIMR      ;zero page jiffy clock
.C007- 78              SEI             ;prevent clock from updating
.C008- 20 08 CC         JSR MOVFM        ;assign TI$
.C00B- 58              CLI             ;allow clock update again
.C00C- 84 5F           STY *FACHO
.C00E- 60              RTS             ;and exit
;*****
;*          *
;* Check for other special     *
;* variables.                 *
;*          *
;*****.c00F- C9 53   QSTATV CMP #'S'       ;if 1st character is S
.C011- D0 09           BNE QDSAV
.C013- C0 54           CPY #'T'       ;and 2nd character is T
.C015- D0 05           BNE QDSAV
.C017- A5 96           LDA *CSTAT
.C019- 4C 72 CD         JMP FLOAT      ;variable is ST
;get status byte
;and assign integer
```

```

C01C- C9 44      QDSAV  CMP #'D'          ;if 1st character is D
C01E- D0 20      BNE GOMOVF
C020- C0 53      CPY #'S'          ;and 2nd is S
C022- D0 1C      BNE GOMOVF
C024- 20 FC BF  JSR CHKDS          ;then variable is DS
C027- A0 00      LDY #$00          ;evaluate and assign DS$
C029- B1 0E      LDA (DSDESC+1),Y ;get offset
C02B- 29 0F      AND #$0F          ;get 1st character of DS$
C02D- 0A          ASL A           ;remove ASCII
C02E- 85 09      STA *GARBFL        ;multiply by 2
C030- 0A          ASL A           ;save intermediate result
C031- 0A          ASL A           ;multiply by 8
C032- 65 09      ADC *GARBFL        ;add intermediate result (times 10)
C034- 85 09      STA *GARBFL        ;save tens
C036- C8          INY              ;point to units
C037- B1 0E      LDA (DSDESC+1),Y ;get units in ASCII
C039- 29 0F      AND #$0F          ;remove ASCII
C03B- 65 09      ADC *GARBFL        ;add to tens (DS complete in $09)
C03D- 4C 72 CD  JMP FLOAT          ;and assign DS
C040- A5 61      GOMOVF LDA *FACMO ;no special var., get
C042- A4 62      LDY *FACLO          ;pointer to value
C044- 4C D8 CC  JMP MOVFM          ;and assign value
;*****
;* Perform arithmetic function. *
;* Entered with A holding token *
;* of function, and CHRGET   *
;* pointing before the argument*
;* of the function.          *
;*****
C047- 0A          ISFUN  ASL A           ;multiply token by 2
C048- 48          PHA               ;save it on stack
C049- AA          TAX               ;and in X
C04A- 20 70 00    JSR CHRGET        ;get next character
C04D- E0 8F      CPX #$8F          ;if token was lower than C8 (LEFT$)
C04F- 90 20      BCC OKNORM        ;go do function
C051- 20 F2 BE  JSR CHKOPN        ;else test for open parenthesis
C054- 20 98 BD  JSR FRMEVL        ;if found, evaluate (string) expression
C057- 20 F5 BE  JSR CHKCOM         ;test for comma
C05A- 20 89 BD  JSR CHKSTR         ;if found, check if value was string
C05D- 68          PLA               ;if so get multiplied token back
C05E- AA          TAX               ;in X
C05F- A5 62      LDA *FACLO        ;push pointer
C061- 48          PHA               ;to string
C062- A5 61      LDA *FACMO        ;on stack
C064- 48          PHA               ;get token*2 back
C065- 8A          TXA               ;save it too
C066- 48          PHA               ;evaluate a one byte parameter
C067- 20 D4 C8  JSR GETBYT        ;get token*2 back
C06A- 68          PLA               ;put it in Y
C06B- A8          TAY               ;get parameter
C06C- 8A          TXA               ;on stack
C06D- 48          PHA               ;and perform LEFT$, MID$ or RIGHT$
C06E- 4C 76 CD  JMP FINGO         ;test for ( and evaluate expression
C071- 20 E9 BE  OKNORM JSR PARCHK ;get token*2 back
C074- 68          PLA               ;in Y
C075- A8          TAY               ;get lo byte of address
C076- B9 FE AF  FINGO LDA FUNDSP-$68,Y ;STA *JMPER+1 ;in pointer
C079- 85 52      STA *JMPER+1        ;LDA FUNDSP-$68+1,Y ;get hi also
C07B- B9 FF AF  STA *JMPER+2        ;JSR JMPER ;complete pointer
C07E- 85 53      JSR JMPER          ;perform function

```

C083- 4C 87 BD	JMP CHKNUM ;and check if value is numeric ;***** ;* * ;* Perform OR. ;* * ;* A OR B is evaluated as: ;* NOT(NOT(A) AND NOT(B)) ;* which is De Morgan's rule. ;* * ;*****
C086- A0 FF	OROP LDY #\$FF ;get inverter value
C088- 2C	.BY \$2C ;and skip next instruction ;***** ;* * ;* Perform AND. ;* * ;*****
C089- A0 00	ANDOP LDY #\$00 ;get inverter value
C08B- 84 05	STY *COUNT ;store inverter value
C08D- 20 EA C2	JSR AYINT ;evaluate integer
C090- A5 61	LDA *FACMO ;get integer lo
C092- 45 05	EOR *COUNT ;invert if OR
C094- 85 03	STA *CHARAC ;store result
C096- A5 62	LDA *FACLO ;get hi also
C098- 45 05	EOR *COUNT ;invert if OR
C09A- 85 04	STA *ENDCHR ;and complete result
C09C- 20 32 CD	JSR MOVFA ;copy FLP accu 1 to FLP accu 2
C09F- 20 EA C2	JSR AYINT ;evaluate 2nd integer
COA2- A5 62	LDA *FACLO ;get hi
COA4- 45 05	EOR *COUNT ;invert if OR
COA6- 25 04	AND *ENDCHR ;AND with 1st integer hi
COA8- 45 05	EOR *COUNT ;invert result if OR
COAA- A8	TAY ;and save result in Y
COAB- A5 61	LDA *FACMO ;get lo
COAD- 45 05	EOR *COUNT ;invert if OR
COAF- 25 03	AND *CHARAC ;AND with 1st integer lo
COB1- 45 05	EOR *COUNT ;invert result if OR
COB3- 4C BC C4	JMP GIVAYF ;convert integer to floating and exit ;***** ;* * ;* Perform comparison (<, =, >). * ;* * ;*****
COB6- 20 8A BD	DOREL JSR CHKVAL ;check for correct type
COB9- B0 13	BCS STRCMP ;if string type go do string
COBB- A5 6B	LDA *ARGSGN ;get sign of FLP accu 2 (in MSB)
COBD- 09 7F	ORA #\$7F ;set all other bits
COBF- 25 67	AND *ARGHO ;add mantissa MSB
COC1- 85 67	STA *ARGHO ;and set up FLP# in accu 2
COC3- A9 66	LDA #L,ARGEXP ;get pointer to
COC5- A0 00	LDY #H,ARGEXP ;FLP accu 2 in YA
COC7- 20 91 CD	JSR FCOMP ;compare FLP accu's
COCA- AA	TAX ;get result in X
COCB- 4C 01 C1	JMP QCOMP ;and complete comparison
COCE- A9 00	STRCMP LDA #\$00 ;string comparison,
COD0- 85 07	STA *VALTYP ;set variable type to numeric
COD2- C6 4A	DEC *OPMASK ;adapt flags
COD4- 20 B8 C7	JSR FREFAC ;discard temporary string
COD7- 85 5E	STA *DSCTMP ;and store
COD9- 86 5F	STX *DSCTMP+1 ;descriptor
CODB- 84 60	STY *DSCTMP+2 ;in FLP accu1
CODD- A5 69	LDA *ARGMO ;get pointer
CODF- A4 6A	LDY *ARGLO ;to 2nd string
COE1- 20 BC C7	JSR FRETMP ;discard that string too

C0E4- 86 69	STX *ARGMO	;and move pointer
C0E6- 84 6A	STY *ARGLO	;to FLP accu 2
C0E8- AA	TAX	;get length of 2nd string in X
C0E9- 38	SEC	;prepare for subtract
C0EA- E5 5E	SBC *DSCTMP	;subtract length of 1st string
C0EC- F0 08	BEQ STASGN	;if lengths equal, set sign
C0EE- A9 01	LDA #\$01	;get flag '1st is bigger'
C0F0- 90 04	BCC STASGN	;and set sign
C0F2- A6 5E	LDX *DSCTMP	;else get length of 1st
C0F4- A9 FF	LDA #\$FF	;and flag '1st is smaller'
C0F6- 85 63	STASGN STA *FACSGN	;and set sign of FLP accu1
C0F8- A0 FF	LDY #\$FF	;get initial offset
C0FA- E8	INX	;adapt character counter
C0FB- C8	NXTCMP INY	;adapt offset
C0FC- CA	DEX	;and character counter
C0FD- D0 07	BNE GETCMP	;if not all done, compare
C0FF- A6 63	LDX *FACSGN	;else get flag
C101- 30 0F	QCOMP BMI DOCMP	;if 1st smaller, go
C103- 18	CLC	
C104- 90 0C	BCC DOCMP	;else clear carry, go
C106- B1 69	GETCMP LDA (ARGMO),Y	;get character of 2nd string
C108- D1 5F	CMP (DSCTMP+1),Y	;compare with 1st string
C10A- F0 EF	BEQ NXTCMP	;if equal, try next character
C10C- A2 FF	LDX #\$FF	;else get flag '1st is smaller'
C10E- B0 02	BCS DOCMP	;if smaller go finish
C110- A2 01	LDX #\$01	;else get flag '1st is bigger'
C112- E8	DOCMP INX	;adapt flag
C113- 8A	TXA	;get it in A
C114- 2A	ROL A	;bit0=<, bit1=equal, bit2=>
C115- 25 0C	AND *DOMASK	;compare with original comparison
C117- F0 02	BEQ GOFLLOT	;if not the same, go reflect 'false'
C119- A9 FF	LDA #\$FF	;else get dummy value for 'true'
C11B- 4C 72 CD	GOFLLOT JMP FLOAT	;and store value in FLP accu 1
	.FI "B4C.M02"	

19CC 338C-4D58 B4C.M02

;name BASIC4.OC.M02		
;*****		
;*		
;* Loop entry for DIM.		
;*		
;*****		
C11E- 20 F5 BE	DIM3	JSR CHKCOM ;check for comma
;*****		
;*		
;* Perform DIM.		
;*		
;*****		
C121- AA	DIM	TAX ;flag DIM (\$06 non zero)
C122- 20 30 C1		JSR PTRGT1 ;go allocate space for array
C125- 20 76 00		JSR CHRGOT ;get current character
C128- 00 F4		BNE DIM3 ;if not end of statement, loop
C12A- 60		RTS ;else exit
;*****		
;*		
;* Get name and pointer to *		
;* current variable by searching *		
;* memory for the desired name.. *		
;* If variable not found, it is *		
;* set up, with value zero, or *		
;* the empty string. *		
;* (Entry used by LET). *		

```

        ;*
        ;*****
C12B- A2 00    PTRGET LDX #$00      ;flag search for variable
C12D- 20 76 00  JSR CHRGOT       ;get current character
        ;*****
        ;*
        ;* Get name and pointer to      *
        ;* current variable by searching *
        ;* memory for the desired name. *
        ;* If variable not found, it is *
        ;* set up, with value zero, or   *
        ;* the empty string.           *
        ;* (Entry used by DIM).        *
        ;*
        ;*****
C130- 86 06    PTRGT1 STX *DIMFLG   ;store DIM/LET flag
C132- 85 42    PTRGT2 STA *VARNAM   ;store current var. name
C134- 20 76 00  JSR CHRGOT       ;get current character
C137- 20 86 C1  JSR ISLETC       ;set carry if letter
C13A- B0 03    BCS PTRGT3       ;if not a letter
C13C- 4C 00 BF  INTERR JMP SNERR   ;go do SYNTAX ERROR
C13F- A2 00    PTRGT3 LDX #$00     ;else get numeric flag
C141- 86 07    STX *VALTYP      ;and set var. type to numeric
C143- 86 08    STX *INTFLG      ;and flag floating point number
C145- 20 70 00  JSR CHRGET       ;get next character
C148- 90 05    BCC ISSEC        ;if not a numeral
C14A- 20 B6 C1  JSR ISLETC       ;check for letter
C14D- 90 08    BCC NOSEC        ;if letter or numeral
C14F- AA       ISSEC TAX         ;save it in X
C150- 20 70 00  EATEM JSR CHRGET   ;get next character
C153- 90 FB    BCC EATEM        ;if numeral
C155- 20 B6 C1  JSR ISLETC       ;or letter,
C158- B0 F6    BCS EATEM        ;get next character (skip rest of name)
C15A- C9 24    NOSEC CMP #'$'     ;if sign for string
C15C- D0 06    BNE NOTSTR      ;get flag for string
C15E- A9 FF    LDA #$FF        ;and store it
C160- 85 07    STA *VALTYP      ;and skip test for integer
C162- D0 10    BNE TURNON      ;if sign for integer
C164- C9 25    NOTSTR CMP #'%'   ;get flag for integers allowed
C166- D0 13    BNE STRNAM       ;if not allowed, SYNTAX ERROR
C168- A5 0A    LDA *SUBFLG      ;else get flag
C16A- D0 D0    BNE INTERR      ;and flag integer variable
C16C- A9 80    LDA #$80        ;flag in name also
C16E- 85 08    STA *INTFLG      ;and store 1st character
C170- 05 42    ORA *VARNAM      ;get 2nd character of name
C172- 85 42    STA *VARNAM      ;flag string type
C174- 8A       TURNON TXA        ;save in X again
C175- 09 80    ORA #$80        ;point CHRGET after %,$; get (
C177- AA       TAX             ;and store 2nd name character
C178- 20 70 00  JSR CHRGET      ;prepare for subtract
C17B- 86 43    STRNAM STX *VARNAM+1 ;if arrays allowed
C17D- 38       SEC             ;and character is (
C17E- 05 0A    ORA *SUBFLG      ;go get pointer to dimensioned var.
C180- E9 28    SBC #'('        ;else flag all types allowed
C182- D0 03    BNE c187        ;in flag byte
C184- 4C FC C2  JMP ISARY       ;get start of var's. lo
C187- A0 00    c187 LDY #$00      ;and hi
        ;*****
        ;*
        ;* Search variable area for      *
        ;* variable name in ($42).        *

```

```

        ;* If the variable is found,      *
        ;* ($44) points to its value or  *
        ;* descriptor.                  *
        ;* If the variable is not found, *
        ;* it is set up with value zero *
        ;* or the empty string, with    *
        ;* ($44) pointing to value or   *
        ;* descriptor.                  *
        ;*
        ;*****



C18F- 86 5D      STXFND STX *LOWTR+1      ;get pointer
C191- 85 5C      LOPFND STA *LOWTR       ;in ($5C)
C193- E4 2D      CPX *ARYTAB+1      ;if equal to start of arrays
C195- D0 04      BNE LOPFN
C197- C5 2C      CMP *ARYTAB      ;in lo byte too, variable not found
C199- F0 25      BEQ NOTFNS      ;go allocate space for variable
C19B- A5 42      LOPFN  LDA *VARNAM      ;else get 1st char. of name
C19D- D1 5C      CMP (LOWTR),Y      ;if not equal to name of searched var.
C19F- D0 0B      BNE NOTIT       ;go try next variable
C1A1- A5 43      LDA *VARNAM+1     ;else get 2nd name character
C1A3- C8          INY              ;point to 2nd name character
C1A4- D1 5C      CMP (LOWTR),Y      ;if name exists
C1A6- D0 03      BNE NXTPTR      ;point to value
C1A8- 4C B9 C2      JMP FINPTR      ;else adapt index
C1AB- 88          NXTPTR DEY      ;prepare for add
C1AC- 18          NOTIT CLC
C1AD- A5 5C      LDA *LOWTR      ;get current pointer
C1AF- 69 07      ADC #$07      ;point to next variable
C1B1- 90 DE      BCC LOPFND      ;if no page crossed, loop
C1B3- E8          INX              ;else adapt hi also
C1B4- D0 D9      BNE STXFND      ;and loop
        ;*****
        ;*
        ;* Set carry if character in A  *
        ;* is an unshifted letter.      *
        ;*
        ;*****



C1B6- C9 41      ISLETC CMP #'A'      ;if lower than A
C1B8- 90 05      BCC ISLRTS      ;exit with carry clear
C1BA- E9 5B      SBC #'C'      ;subtract value for Z+1
C1BC- 38          SEC              ;prepare for subtract
C1BD- E9 A5      SBC #$A5      ;get value back, set C if letter
C1BF- 60          ISLRTS RTS      ;and exit
        ;*****
        ;*
        ;* Create a new variable      *
        ;* (entered if variable not  *
        ;* found in variable area).   *
        ;*
        ;*****



C1C0- 68          NOTFNS PLA      ;get last return address to
C1C1- 48          PHA              ;restore stack
C1C2- C9 8E      CMP #L,ISVRET-1 ;if not called from get value
C1C4- D0 05      BNE NOTEVL      ;go get pointer to value
C1C6- A9 C9      LDZR  LDA #L,ZERO   ;else point YA to dummy
C1C8- A0 D0      LDY #H,ZERO
C1CA- 60          RTS              ;and exit
C1CB- A5 42      NOTEVL LDA *VARNAM ;get 1st name character in A
C1CD- A4 43      LDY *VARNAM+1   ;and 2nd in Y
C1CF- C9 54      CMP #'T'      ;if name is
C1D1- D0 0B      BNE QSTAVR
C1D3- C0 C9      CPY #$C9      ;TI$
C1D5- F0 EF      BEQ LDZR       ;exit with pointer to dummy
C1D7- C0 49      CPY #'I'      ;if name is TI

```

```

C1D9- D0 03      BNE QSTAVR
C1DB- 4C 00 BF   GOBADV JMP SNERR      ;go do SYNTAX ERROR
C1DE- C9 53      QSTAVR CMP #'S'       ;if name
C1E0- D0 04      BNE QDSVAR
C1E2- C0 54      CPY #'T'            ;is ST
C1E4- F0 F5      BEQ GOBADV      ;SYNTAX ERROR
C1E6- C9 44      QDSVAR CMP #'D'       ;if name
C1E8- D0 08      BNE VAROK
C1EA- C0 53      CPY #'S'            ;is DS
C1EC- F0 E0      BEQ GOBADV      ;SYNTAX ERROR
C1EE- C0 D3      CPY #$D3          ;if name is DS$
C1F0- F0 E9      BEQ GOBADV      ;SYNTAX ERROR
C1F2- A5 2C      VAROK  LDA *ARYTAB
C1F4- A4 2D      LDY *ARYTAB+1    ;get start of arrays
C1F6- 85 5C      STA *LOWTR
C1F8- 84 50      STY *LOWTR+1    ;in source start
C1FA- A5 2E      LDA *STREND
C1FC- A4 2F      LDY *STREND+1    ;pointer (for block move amount)
C1FE- 85 57      STA *HIGHTR
C200- 84 58      STY *HIGHTR+1    ;get end of arrays
C202- 18         CLC
C203- 69 07      ADC #$07          ;in source end pointer
C205- 90 01      BCC NOTEVE      ;for block move
C207- C8         INY
C208- 85 55      NOTEVE STA *HIGHDS
C20A- 84 56      STY *HIGHDS+1    ;pointer for block move
C20C- 20 50 B3   JSR BLTU        ;test space and move block in memory
;*****
;*
;* Set new start of arrays/end   *
;* of variables pointer.        *
;*                               *
;*****
C20F- A5 55      LDA *HIGHDS      ;get new start of arrays
C211- A4 56      LDY *HIGHDS+1    ;in YA
C213- C8         INY
C214- 85 2C      STA *ARYTAB
C216- 84 20      STY *ARYTAB+1    ;store it in
C218- 85 55      STA *HIGHDS
C21A- 84 56      STY *HIGHDS+1    ;start of arrays/end of var's
.FI "B4C.M03"

```

1A25 338C-4DB1 B4C.M03

```

;name BASIC4.0C.M03
;*****
;*
;* Search array area for string  *
;* arrays. If string array found *
;* recalculate all backpointers  *
;* because arrays moved when    *
;* new variable was added.      *
;*
;*****
C21C- A5 55      ARYVA2 LDA *ARYPNT      ;get pointer to
C21E- A6 56      LDX *ARYPNT+1    ;variable in XA
C220- E4 2F      ARYVA3 CPX *STREND+1  ;if at start of free RAM
C222- D0 04      BNE ARYVGO
C224- C5 2E      CMP *STREND      ;all arrays tested
C226- F0 75      BEQ ARYDON      ;go set up array variable in RAM
C228- 85 1F      ARYVGO STA *INDEX
C22A- 86 20      STX *INDEX+1    ;else save current array pointer
                                ;in ($1F)
C22C- A0 00      LDY #$00        ;point to 1st name character

```

```

C22E- B1 1F      LDA (INDEX),Y    ;get it
C230- AA          TAX             ;save it in X
C231- C8          INY             ;point to 2nd name character
C232- B1 1F      LDA (INDEX),Y    ;get that too
C234- 08          PHP             ;save the flags
C235- C8          INY             ;point to length of array lo
C236- B1 1F      LDA (INDEX),Y    ;get it
C238- 65 55      ADC *ARYPNT     ;calculate pointer to
C23A- 85 55      STA *ARYPNT     ;next array lo in $55
C23C- C8          INY             ;point to length hi
C23D- B1 1F      LDA (INDEX),Y    ;get it
C23F- 65 56      ADC *ARYPNT+1   ;add carry to pointer hi
C241- 85 56      STA *ARYPNT+1   ;and complete pointer to next array
C243- 28          PLP             ;get flags back (for 2nd name char.)
C244- 10 06      BPL ARYVA2    ;if FLP, loop
C246- 8A          TXA             ;get 1st character of name back
C247- 30 03      BMI ARYVA2    ;if integer, loop
                                ;*****
                                ;*
                                ;* String array found, recal- *
                                ;* culate all backpointers of *
                                ;* its strings in hi memory. *
                                ;*
                                ;*****
C249- C8          INY             ;if string array,
C24A- B1 1F      LDA (INDEX),Y    ;get number of indices
C24C- A0 00      LDY #$00        ;point to name again
C24E- 0A          ASL A           ;multiply number of indices by 2
C24F- 69 05      ADC #$05        ;get offset to 1st item
C251- 65 1F      ADC *INDEX      ;point lo to 1st item
C253- 85 1F      STA *INDEX      ;and store pointer lo
C255- 90 02      BCC ARYGET     ;if page crossed
C257- E6 20      INC *INDEX+1   ;adapt hi
C259- A6 20      ARYGET         LDX *INDEX+1   ;get pointer hi in X (A holds lo)
C25B- E4 56      CPX *ARYPNT+1   ;if not at end of array
C25D- D0 04      BNE GOGO       ;go do next string
C25F- C5 55      CMP *ARYPNT     ;if lo and hi at end of current array
C261- F0 B0      BEQ ARYVA3    ;go test next array
C263- A0 00      GOGO          LDY #0          ;else point to length of string
C265- B1 1F      LDA (INDEX),Y    ;get it
C267- F0 27      BEQ DVARTS    ;if string empty, do next string
C269- C8          INY             ;else point to textpointer lo
C26A- 18          CLC             ;prepare for add
C26B- 71 1F      ADC (INDEX),Y   ;calculate pointer lo to backpointer
C26D- 85 57      STA *HIGHTR     ;save it
C26F- AA          TAX             ;in X too
C270- C8          INY             ;point to pointer to text hi
C271- B1 1F      LDA (INDEX),Y    ;get it
C273- 69 00      ADC #$00        ;add carry
C275- 85 58      STA *HIGHTR+1   ;and complete pointer to backpointer
C277- C5 31      CMP *FRETOP+1   ;if backpointer not in string area
C279- 90 15      BCC DVARTS    ;go do next string
C27B- D0 04      BNE GOGO1     ;else adapt backpointer
C27D- E4 30      CPX *FRETOP     ;if lo byte not in string area
C27F- 90 0F      BCC DVARTS    ;do next string
C281- A0 00      GOGO1         LDY #$00        ;else point to backpointer lo
C283- B1 57      LDA (HIGHTR),Y   ;get it
C285- 69 06      ADC #$06        ;add 7 (carry set already)
C287- 91 57      STA (HIGHTR),Y   ;and store it
C289- C8          INY             ;point to backpointer hi
C28A- B1 57      LDA (HIGHTR),Y   ;get it
C28C- 69 00      ADC #$00        ;add carry
C28E- 91 57      STA (HIGHTR),Y   ;and complete new backpointer
C290- A9 03      DVARTS        LDA #$03        ;get offset to next string item

```

```

C292- 18      CLC          ;prepare for add
C293- 65 1F    ADC *INDEX   ;calculate pointer lo
C295- 85 1F    STA *INDEX   ;store it
C297- 90 C0    BCC ARYGET  ;if no page crossed, do next string
C299- E6 20    INC *INDEX+1 ;else adapt pointer hi
C29B- D0 BC    BNE ARYGET  ;and do next string
;***** *
;* Set up variable in variable *
;* area and fill it with a zero *
;* value or the empty string.   *
;*                                *
;***** *
C29D- A0 00    ARYDON LDY #$00  ;point to 1st name character
C29F- A5 42    LDA *VARNAM  ;get that character
C2A1- 91 5C    STA (LOWTR),Y ;store it in memory
C2A3- C8       INY          ;point to 2nd name character
C2A4- A5 43    LDA *VARNAM+1 ;get that too
C2A6- 91 5C    STA (LOWTR),Y ;and move to memory
C2A8- A9 00    LDA #$00  ;get initial zero value
C2AA- C8       INY          ;point to exponent/lo byte/length
C2AB- 91 5C    STA (LOWTR),Y ;point to mantissa MSB/hi byte/pointer
C2AD- C8       INY          ;point to mantissa/???/pointer hi
C2AE- 91 5C    STA (LOWTR),Y ;point to mantissa/???/?
C2B0- C8       INY          ;point to mantissa/???/?
C2B1- 91 5C    STA (LOWTR),Y ;point to mantissa LSB/???/?
C2B3- C8       INY          ;point to mantissa LSB/???/?
C2B4- 91 5C    STA (LOWTR),Y ;point to mantissa LSB/???/?
C2B6- C8       INY          ;point to mantissa LSB/???/?
C2B7- 91 5C    STA (LOWTR),Y ;point to mantissa LSB/???/?
;***** *
;* Point ($44) to value or   *
;* descriptor in variable or *
;* array area.               *
;*                                *
;***** *
C2B9- A5 5C    FINPTR LDA *LOWTR ;get pointer lo
C2BB- 18       CLC          ;prepare for add
C2BC- 69 02    ADC #$02  ;point to value or descriptor
C2BE- A4 5D    LDY *LOWTR+1 ;get pointer hi
C2C0- 90 01    BCC FINNOW ;if page crossed
C2C2- C8       INY          ;adapt hi
C2C3- 85 44    FINNOW STA *VARPNT ;store pointer lo
C2C5- 84 45    STY *VARPNT+1 ;and hi
C2C7- 60       RTS          ;and exit
;***** *
;* Calculate pointer to first *
;* array item in ($55).        *
;*                                *
;***** *
C2C8- A5 05    FMAPTR LDA *COUNT ;get number of subscripts
C2CA- 0A       ASL A        ;multiply by 2
C2CB- 69 05    ADC #$05  ;add five (points to next array)
C2CD- 65 5C    ADC *LOWTR ;and correct pointer lo
C2CF- A4 5D    LDY *LOWTR+1 ;get pointer hi
C2D1- 90 01    BCC JSRGM  ;if page crossed
C2D3- C8       INY          ;adapt hi
C2D4- 85 55    JSRGM STA *ARYPNT ;store pointer lo
C2D6- 84 56    STY *ARYPNT+1 ;and hi
C2D8- 60       RTS          ;and exit
;***** *
;*                                *

```

```

;* FLP number for minimum sub- *
;* script allowed. *
;* Bug: *
;* The mantissa LSB is missing; *
;* the opcode for JSR ($20) at *
;* $C2DD is used for this. *
;* The resulting value is *
;* -32768.0005. *
;*
;*****  

C2D9- 90      N32768 .BY $90          ;exponent
C2DA- 80      .BY $80          ;mantissa, MSB, and sign
C2DB- 00      .BY $00          ;mantissa
C2DC- 00      .BY $00          ;mantissa
;*****  

;*
;* Read and check one maximum *
;* value for a subscript in a *
;* DIM statement. *
;*
;*****  

C2D0- 20 70 00 .INTIDX JSR CHRGET      ;get next character
C2E0- 20 98 BD  JSR FRMEVL      ;evaluate expression for subscript
C2E3- 20 87 BD  POSINT   JSR CHKNUM      ;check if numeric type
C2E6- A5 63    LDA *FACSGN      ;get sign of FLP accu 1
C2E8- 30 00    BMI NONONO      ;if negative, ILLEGAL QUANTITY ERROR
;*****  

;*
;* Read one number and evaluate *
;* it to an integer in ($61). *
;* Maximum value allowed for *
;* the integer is (-)32768. *
;*
;*****  

C2EA- A5 5E      AYINT    LDA *FACEXP      ;get exponent of FLP accu1
C2EC- C9 90      CMP #$90          ;if 32767.9998 or lower,
C2EE- 90 09      BCC QINTGO      ;go convert FLP to integer in ($61)
C2F0- A9 D9      LDA #L,N32768     ;else get pointer to
C2F2- A0 C2      LDY #H,N32768     ;minimum number (-32768)
;BUG: minimum number is equal to -32768.0005,
;so -32768 will be rejected instead of accepted.
C2F4- 20 91 CD  JSR FCOMP      ;compare FLP accu1 with number
C2F7- D0 7A      NONONO   BNE FCERR      ;if not equal, ILLEGAL QUANTITY ERROR
C2F9- 4C D1 CD  QINTGO   JMP QINT      ;else convert floating to integer
.FI "B4C.M04"  


```

19AC 338C-4D38 B4C.M04

```

;name BASIC4.0C.M04
;*****  

;*
;* Get pointer to dimensioned *
;* variable. If the array of the *
;* variable does not exist, it *
;* is set up with default max. *
;* subscript(s) of 10. *
;*
;*****  

C2FC- A5 06      ISARY    LDA *DIMFLG      ;get DIM/LET flag
C2FE- 05 08      ORA *INTFLG      ;add numeric type
C300- 48          PHA          ;save result on stack
C301- A5 07      LDA *VALTYP      ;get variable type
C303- 48          PHA          ;save that too

```

```

C304- A0 00 LDY #$00 ;get initial
C306- 98 INDLOP TYA ;number of subscripts in accu
C307- 48 PHA ;save number of subscripts
C308- A5 43 LDA *VARNAM+1 ;get 2nd character of name
C30A- 48 PHA ;save it
C30B- A5 42 LDA *VARNAM ;get 1st character of name
C30D- 48 PHA ;save that too
C30E- 20 DD C2 JSR INTIDX ;read expression for max. subscript in
C311- 68 PLA ;get 1st name character back
C312- 85 42 STA *VARNAM ;restore it
C314- 68 PLA ;get 2nd character too
C315- 85 43 STA *VARNAM+1 ;complete name
C317- 68 PLA ;get number of subscripts
C318- A8 TAY ;in Y again
C319- BA TSX ;get stack pointer
C31A- BD 02 01 LDA $0102,X ;get variable type
C31D- 48 PHA ;on correct stack position
C31E- BD 01 01 LDA $0101,X ;do the same
C321- 48 PHA ;for the numeric type/DIM LET flag
C322- A5 61 LDA *INDICE ;get subscript lo
C324- 9D 02 01 STA $0102,X ;on stack
C327- A5 62 LDA *INDICE+1 ;get hi
C329- 9D 01 01 STA $0101,X ;on stack too
C32C- C8 INY ;adapt number of subscripts
C32D- 20 76 00 JSR CHRGOT ;get current character
C330- C9 2C CMP #',,' ;if comma
C332- F0 02 BEQ INDLOP ;loop for next subscript
C334- 84 05 STY *COUNT ;else save number of subscripts
C336- 20 EF BE JSR CHKCLS ;test if current character is )
C339- 68 PLA ;if so, get variable type back
C33A- 85 07 STA *VALTYP ;in its flag
C33C- 68 PLA ;get numeric type and DIM/LET flag
C33D- 85 08 STA *INTFLG ;in its flag too
C33F- 29 7F AND #$7F ;get DIM/LET flag back ent
C341- 85 06 STA *DIMFLG ;in its flag
C343- A6 2C LDX *ARYTAB ;get start of arrays
C345- A5 2D LDA *ARYTAB+1
C347- 86 5C LOPFDA STX *LOWTR ;in current pointer
C349- 85 50 STA *LOWTR+1
C34B- C5 2F CMP *STREND+1 ;if at end of arrays
C34D- D0 04 BNE LOPFDV
C34F- E4 2E CPX *STREND
C351- F0 39 BEQ NOTFDD ;go set array up
C353- A0 00 LOPFDV LDY #0 ;else point to 1st name character
C355- B1 5C LDA (LOWTR),Y ;get it
C357- C8 INY ;point to 2nd character
C358- C5 42 CMP *VARNAM ;if not the desired name
C35A- D0 06 BNE NMARY1 ;try next array
C35C- A5 43 LDA *VARNAM+1 ;get 2nd character searched for
C35E- D1 5C CMP (LOWTR),Y ;if array name is the same
C360- F0 16 BEQ GOTARY ;go test if double dimension
C362- C8 NMARY1 INY ;else point to length lo
C363- B1 5C LDA (LOWTR),Y ;get it
C365- 18 CLC ;prepare for add
C366- 65 5C ADC *LOWTR ;calculate pointer to next array lo
C368- AA TAX ;save it in X
C369- C8 INY ;point to length hi
C36A- B1 5C LDA (LOWTR),Y ;get it
C36C- 65 5D ADC *LOWTR+1 ;add carry
C36E- 90 07 BCC LOPFDA ;and try next array (branch always)
;*****  

;* *  

;* ?BAD SUBSCRIPT ERROR, *  

;* then READY. (restart BASIC). *

```

```

        ;* *
        ;*****BSERR*****
C370- A2 6B    BSERR   LDX #b278-ERRTAB ;point to ?BAD SUBSCRIPT
C372- 2C          .BY $2C           ;skip next instruction
        ;***** *
        ;* ?ILLEGAL QUANTITY ERROR, *
        ;* then READY. (restart BASIC). *
        ;* *
        ;*****FCERR*****
C373- A2 35    FCERR   LDX #b242-ERRTAB ;point to ?ILLEGAL QUANTITY
C375- 4C CF B3  ERRG03  JMP ERROR      ;go print message and restart BASIC
        ;***** *
        ;* Array found. *
        ;* Check if not called by DIM *
        ;* and get value of item *
        ;* requested. *
        ;* *
        ;*****GOTARY*****
C378- A2 78    GOTARY  LDX #b285-ERRTAB ;point to ?REDIM'ED ARRAY
C37A- A5 06          LDA *DIMFLG    ;if called by DIM
C37C- D0 F7          BNE ERRG03    ;go do error and restart
C37E- 20 C8 C2          JSR FMAPTR   ;else compute pointer to 1st item
C381- A5 05          LDA *COUNT    ;get number of subscripts in var.
C383- A0 04          LDY #4        ;point to number of s. in array
C385- D1 5C          CMP (LOWTR),Y ;if not equal,
C387- D0 E7          BNE BSERR    ;go do BAD SUBSCRIPT ERROR and restart
C389- 4C 15 C4          JMP GETDEF   ;else get value of element
        ;***** *
        ;* Array not found, set it up. *
        ;* If not called by DIM, use *
        ;* default max. subscripts of *
        ;* ten. *
        ;* *
        ;*****NOTFDD*****
C38C- 20 C8 C2  NOTFDD   JSR FMAPTR   ;compute pointer to 1st array item
C38F- 20 A0 B3          JSR REASON   ;test for overlap of arrays and strings
C392- A0 00          LDY #0        ;get initial size hi
C394- 84 6F          STY *CURTOL+1 ;in $6f
C396- A2 05          LDX #5        ;get item length for FLP array
C398- A5 42          LDA *VARNAME  ;get 1st name character
C39A- 91 5C          STA (LOWTR),Y ;move to array area
C39C- 10 01          BPL NOTFLT   ;if integer array
C39E- CA            DEX         ;adapt item length
C39F- C8            NOTFLT    INY       ;point to 2nd char. of name
C3A0- A5 43          LDA *VARNAME+1 ;get 2nd name character
C3A2- 91 5C          STA (LOWTR),Y ;move to array area
C3A4- 10 02          BPL STOMLT   ;if string or integer array
C3A6- CA            DEX         ;set item length
C3A7- CA            DEX         ;to 2 or 3
C3A8- 86 6E          STOMLT    STX *CURTOL ;and store item length
C3AA- A5 05          LDA *COUNT   ;get number of indices+1
C3AC- C8            INY         ;INY
C3AD- C8            INY         ;INY
C3AE- C8            INY         ;point to indices save area
C3AF- 91 5C          STA (LOWTR),Y ;move to array area
C3B1- A2 0B          LOPPTA    LDX #11     ;get default lo of max. index
C3B3- A9 00          LDA #$00     ;get default hi also
C3B5- 24 06          BIT *DIMFLG   ;if not called by DIM
C3B7- 50 08          BVC NOTDIM   ;use defaults
C3B9- 68            PLA         ;else get subscript lo
C3BA- 18            CLC         ;prepare for add

```

C3BB- 69 01	ADC #1	;increment it
C3BD- AA	TAX	;save in X
C3BE- 68	PLA	;get subscript hi
C3BF- 69 00	ADC #0	;add carry
C3C1- C8	NOTDIM INY	;point to max. value of index hi
C3C2- 91 5C	STA (LOWTR),Y	;move to array area
C3C4- C8	INY	;point to max. value lo
C3C5- 8A	TXA	;get lo byte
C3C6- 91 5C	STA (LOWTR),Y	;move to array area
C3C8- 20 77 C4	JSR UMULT	;compute size of this subscript
C3CB- 86 6E	STX *CURTOL	;store size lo
C3CD- 85 6F	c3cd STA *CURTOL+1	;and hi
C3CF- A4 1F	LDY *INDEX	;get index back
C3D1- C6 05	DEC *COUNT	;if not all indices done
C3D3- D0 DC	BNE LOPPTA	;repeat for others
C3D5- 65 56	ADC *ARYPNT+1	;else add size hi to pointer
C3D7- B0 5D	BCS OMERR1	;if overflow, ?OUT OF MEMORY ERROR
C3D9- 85 56	STA *ARYPNT+1	;else store pointer hi to end
C3DB- A8	TAY	;move to Y too
C3DC- 8A	TXA	;get size lo
C3DD- 65 55	ADC *ARYPNT	;complete pointer to end
C3DF- 90 03	BCC GREASE	;if page crossed
C3E1- C8	INY	;adapt hi byte
C3E2- F0 52	BEQ OMERR1	;if overflow, ?OUT OF MEMORY ERROR
C3E4- 20 A0 B3	GREASE JSR REASON	;check for overlap
C3E7- 85 2E	STA *STREND	;if space, store new end
C3E9- 84 2F	STY *STREND+1	;of arrays
C3EB- A9 00	LDA #0	;get initial value
C3ED- E6 6F	INC *CURTOL+1	;adapt length hi
C3EF- A4 6E	LDY *CURTOL	;get length lo
C3F1- F0 05	BEQ DECCUR	;if zero, do previous page
C3F3- 88	ZERITA DEY	;else adapt length lo
C3F4- 91 55	STA (ARYPNT),Y	;store zero in array area
C3F6- D0 FB	BNE ZERITA	;until lo byte of length zero
C3F8- C6 56	DECCUR DEC *ARYPNT+1	;then adapt page number
C3FA- C6 6F	DEC *CURTOL+1	;adapt length hi
C3FC- D0 F5	BNE ZERITA	;and do previous 256 bytes
C3FE- E6 56	INC *ARYPNT+1	;if array filled with zeroes
C400- 38	SEC	;prepare for subtract
C401- A5 2E	LDA *STREND	;get end of arrays lo
C403- E5 5C	SBC *LOWTR	;subtract start of current (last) array
C405- A0 02	LDY #2	;point to length of array lo
C407- 91 5C	STA (LOWTR),Y	;store length lo
C409- A5 2F	LDA *STREND+1	;get end of arrays hi
C40B- C8	INY	;point to length hi
C40C- E5 5D	SBC *LOWTR+1	;calculate length hi
C40E- 91 5C	STA (LOWTR),Y	;and store that too
C410- A5 06	LDA *DIMFLG	;if called by DIM
C412- D0 62	BNE DIMRTS	;exit now
	.FI "B4C.MOS"	

1909 338C-4065 B4C.MOS

```

;name BASIC4.0C.MOS
;*****
;* Get pointer to dimensioned *
;* variable. If the array of the *
;* variable does not exist, it   *
;* is set up with default max.   *
;* subscript(s) of 10.           *
;*                               *
;*****

```

C414- C8	INY	;point to number of indices in array
C415- B1 5C	GETDEF LDA (LOWTR),Y	;get number of indices
C417- 85 05	STA *COUNT	;save it
C419- A9 00	LDA #0	;zero
C41B- 85 6E	STA *CURTOL	;length lo
C41D- 85 6F	INLPNM STA *CURTOL+1	;and hi
C41F- C8	INY	;point to max value+1 of index hi
C420- 68	PLA	;get index lo from stack
C421- AA	TAX	;save it in X
C422- 85 61	STA *INDICE	;save
C424- 68	PLA	;get index hi
C425- 85 62	STA *INDICE+1	;complete index in (\$61)
C427- D1 5C	CMP (LOWTR),Y	;if lower than declared value+1
C429- 90 0E	BCC INLPN2	;go get pointer to value
C42B- D0 06	BNE BSERR7	;if not equal, ?BAD SUBSCRIPT ERROR
C42D- C8	INY	;else point to max. lo
C42E- 8A	TXA	;get index lo
C42F- D1 5C	CMP (LOWTR),Y	;if lower or same than declared
C431- 90 07	BCC INLPN1	;go get pointer to value
C433- 4C 70 C3	BSERR7 JMP BSERR	;else ?BAD SUBSCRIPT ERROR, restart
C436- 4C CD B3	OMERR1 JMP OMERR	?OUT OF MEMORY ERROR, restart
C439- C8	INLPN2 INY	;point to index lo
C43A- A5 6F	INLPN1 LDA *CURTOL+1	;get length hi
C43C- 05 6E	ORA *CURTOL	;if current length zero
C43E- 18	CLC	;prepare for add
C43F- F0 0A	BEQ ADDIND	;add initial index
C441- 20 77 C4	JSR UMULT	;else compute # of items/subscript
C444- 8A	TXA	;get number of items lo
C445- 65 61	ADC *INDICE	;add to total number lo
C447- AA	TAX	;get total lo in X
C448- 98	TYA	;get number of items hi
C449- A4 1F	LDY *INDEX	;get index back
C44B- 65 62	ADDIND ADC *INDICE+1	;add to total number hi
C44D- 86 6E	STX *CURTOL	;store total lo
C44F- C6 05	DEC *COUNT	;if not all indices done
C451- D0 CA	BNE INLPNM	;loop for next index
C453- 85 6F	STA *CURTOL+1	;else store total hi
C455- A2 05	LDX #5	;get item length for FLP array
C457- A5 42	LDA *VARNAM	;if type is integer
C459- 10 01	BPL NOTFL1	
C45B- CA	DEX	
C45C- A5 43	NOTFL1 LDA *VARNAM+1	
C45E- 10 02	BPL STOML1	
C460- CA	DEX	
C461- CA	DEX	
C462- 86 25	STOML1 STX *ADDEND	
C464- A9 00	LDA #0	
C466- 20 80 C4	JSR UMULTD	
C469- 8A	TXA	
C46A- 65 55	ADC *ARYPNT	
C46C- 85 44	STA *VARPNT	
C46E- 98	TYA	
C46F- 65 56	ADC *ARYPNT+1	
C471- 85 45	STA *VARPNT+1	
C473- A8	TAY	
C474- A5 44	LDA *VARPNT	
C476- 60	DIMRTS RTS	
		;and exit

		;
		*
		;
		* Compute value of indexlimit *
		;
		* multiplied by (\$6E) in YX and *
		;
		* YA. *
		;
		* *****

```

C477- 84 1F    UMULT   STY *INDEX      ;save offset in array
C479- B1 5C    LDA (LOWTR),Y   ;get index limit lo
C47B- 85 25    STA *ADDEND    ;in $25
C47D- 88       DEY          ;point to indexlimit hi
C47E- B1 5C    LDA (LOWTR),Y   ;get it
;***** *
;* Compute value of A (hi) and *
;* $25 (lo) multiplied by ($6E) *
;* in YX and YA.               *
;*                                *
;***** *
C480- 85 26    UMULTD  STA *ADDEND+1  ;save value for hi
C482- A9 10    LDA #16        ;get bit counter
C484- 85 5A    STA *LOWDS    ;in $5A
C486- A2 00    LDX #0        ;clear initial lo
C488- A0 00    LDY #0        ;clear initial hi also
C48A- 8A       UMULTC   TXA        ;get result so far, lo
C48B- 0A       ASL A        ;multiply by 2
C48C- AA       TAX          ;move to X again
C48D- 98       TYA          ;get result so far, hi
C48E- 2A       ROL A        ;multiply by 2 also
C48F- A8       TAY          ;move to Y again
C490- B0 A4    BCS OMERR1   ;if overflow, OUT OF MEMORY ERROR
C492- 06 6E    ASL *CURTOL   ;get current MSbit
C494- 26 6F    ROL *CURTOL+1 ;of multiplicand
C496- 90 0B    BCC UMLCNT   ;if bit was set
C498- 18       CLC          ;prepare for add
C499- 8A       TXA          ;get lo so far
C49A- 65 25    ADC *ADDEND   ;add value lo
C49C- AA       TAX          ;save in X again
C49D- 98       TYA          ;get hi so far
C49E- 65 26    ADC *ADDEND+1 ;add value hi
C4A0- A8       TAY          ;move to Y again
C4A1- B0 93    BCS OMERR1   ;if overflow, OUT OF MEMORY ERROR
C4A3- C6 5A    UMLCNT   DEC *LOWDS   ;adapt bit counter
C4A5- D0 E3    BNE UMULTC   ;if not all done, loop
C4A7- 60       UMLRTS   RTS        ;else exit
;***** *
;*                                *
;* Perform FRE.                 *
;*                                *
;***** *
C4A8- A5 07    FRE     LDA *VALTYP   ;if current type is string
C4AA- F0 03    BEQ NOREF
C4AC- 20 B8 C7    JSR FREFAC   ;clean descriptor stack
C4AF- 20 6A C6    NOREF   JSR GARBA2   ;collect garbage strings
C4B2- 38       SEC          ;prepare for subtract
C4B3- A5 30    LDA *FRETOP   ;get start of strings lo
C4B5- E5 2E    SBC *STREND   ;subtract end of arrays lo
C4B7- A8       TAY          ;save result in Y
C4B8- A5 31    LDA *FRETOP+1 ;get start of strings hi
C4BA- E5 2F    SBC *STREND+1 ;subtract end of arrays hi
;***** *
;*                                *
;* Convert AY to floating point *
;* number in FLP accur1.        *
;*                                *
;***** *
C4BC- A2 00    GIVAYF   LDX #0        ;set variable type
C4BE- 86 07    STX *VALTYP   ;to numeric
C4C0- 85 5F    STA *FACHO    ;and move result
C4C2- 84 60    STY *FACMOH   ;to FLP accur1
C4C4- A2 90    LDX #$90      ;get exponent

```

C4C6- 4C 7A CD	JMP FLOATS ;and convert to floating ;***** ;* * ;* Perform POS. * ;* * ;*****
C4C9- A4 C6	POS LDY *TRMPOS ;get current column number
C4CB- A9 00	SNGFLT LDA #0 ;get hi byte also
C4CD- F0 ED	BEQ GIVAYF ;convert AY to FLP and exit ;***** ;* * ;* Check for program mode. * ;* If in direct mode, ?ILLEGAL * ;* DIRECT ERROR is printed and * ;* BASIC is restarted. * ;* * ;*****
C4CF- A6 37	ERRDIR LDX *CURLIN+1 ;get current line number hi
C4D1- E8	INX ;if it was not \$FF
C4D2- D0 A2	BNE DIMRTS ;exit
C4D4- A2 95	LDX #b2a2-ERRTAB ;else point to ILLEGAL DIRECT .
C4D6- 2C	.BY \$2C ;skip next statement ;***** ;* * ;* UNDEF'D FUNCTION ERROR, then * ;* restart BASIC. * ;* * ;*****
C4D7- A2 E9	ERRGUF LDX #b2f6-ERRTAB ;point to UNDEF'D FUNCTION
C4D9- 4C CF B3	JMP ERROR ;do error and restart BASIC ;***** ;* * ;* Perform DEF. * ;* * ;*****
C4DC- 20 DA C5	DEF JSR GETFNM ;check for FN and variable
C4DF- 20 CF C4	JSR ERRDIR ;check if in program mode
C4E2- 20 F2 BE	JSR CHKOPN ;check for (
C4E5- A9 80	LDA #\$80 ;set up flag for FN variable
C4E7- 85 DA	STA *SUBFLG ;no integers or arrays allowed
C4E9- 20 2B C1	JSR PTRGET ;get pointer to global variable
C4EC- 20 87 BD	JSR CHKNUM ;check if numeric type
C4EF- 20 EF BE	JSR CHKCLS ;test for)
C4F2- A9 B2	LDA #EQLUTK ;get token for =
C4F4- 20 F7 BE	JSR SYNCHR ;must be current character
C4F7- 48	PHA ;save 1st character of function
C4F8- A5 45	LDA *VARPNT+1 ;get pointer hi to function
C4FA- 48	PHA ;save it
C4FB- A5 44	LDA *VARPNT ;get pointer lo too
C4FD- 48	PHA ;save it
C4FE- A5 78	LDA *TXTPTPTR+1 ;get CHRGET's pointer hi
C500- 48	PHA ;save it
C501- A5 77	LDA *TXTPTPTR ;get CHRGET's pointer lo
C503- 48	PHA ;save that too
C504- 20 83 B8	JSR DATA ;perform DATA
C507- 4C 78 C5	JMP DEFFIN ;and store function parameters .FI "B4C.M06"

1BAE 338C-4F3A B4C.M06

```
;name BASIC4.0C.M06
;*****
;* *
```

```

;* Check if FN and variable are  *
;* following DEF. Store pointer  *
;* to the variable in ($4B).    *
;*                                *
;*****  

C50A- A9 A5      GETFNM LDA #FNTK      ;get token for FN.  

C50C- 20 F7 BE    JSR SYNCNR      ;must be current character  

C50F- 09 80      ORA #$80       ;set MSbit (flag FN variable)  

C511- 85 0A      STA *SUBFLG     ;and allow no integer or arrays  

C513- 20 32 C1    JSR PTRGT2     ;set variable up  

C516- 85 4B      STA *DEFPNT     ;and move pointer to variable  

C518- 84 4C      STY *DEFPNT+1   ;to ($4B)  

C51A- 4C 87 BD    JMP CHKNUM     ;test if numeric and exit  

;*****  

;*                                *
;* Evaluate FN.                  *
;*                                *
;*****  

C51D- 20 0A C5    FNDOER JSR GETFNM    ;check for FN and type  

C520- A5 4C      LDA *DEFPNT+1   ;get pointer hi  

C522- 48          PHA             ;save on stack  

C523- A5 4B      LDA *DEFPNT     ;get lo also  

C525- 48          PHA             ;on stack  

C526- 20 E9 BE    JSR PARCHK     ;test for ( and evaluate expression  

C529- 20 87 BD    JSR CHKNUM     ;if result was numeric  

C52C- 68          PLA             ;get pointer lo back  

C52D- 85 4B      STA *DEFPNT     ;get hi also  

C52F- 68          PLA             ;point to pointer to formal var.  

C530- 85 4C      STA *DEFPNT+1   ;get pointer to formal value lo  

C532- A0 02      LDY #2         ;store it  

C534- B1 4B      LDA (DEFPNT),Y  ;in X also  

C536- 85 44      STA *VARPNT     ;point to hi  

C538- AA          TAX             ;get pointer hi  

C539- C8          INY             ;if zero, UNDEF'D FUNCTION ERROR, restart  

C53A- B1 4B      LDA (DEFPNT),Y  ;else store it  

C53C- F0 99      BEQ ERRGUF     ;Y=4  

C53E- 85 45      STA *VARPNT+1   ;push real value of  

C540- C8          INY             ;formal variable on stack  

C541- B1 44      DEFSTF LDA (VARPNT),Y ;until 5 bytes  

C543- 48          PHA             ;done  

C544- 88          DEY             ;set YX to point to value  

C545- 10 FA      BPL DEFSTF     ;move FLP accu1 to formal variable  

C547- A4 45      LDY *VARPNT+1   ;get CHRGET's pointer hi  

C549- 20 0A CD    JSR MOVMF      ;on stack  

C54C- A5 78      LDA *TXTPTR+1   ;do the same  

C54E- 48          PHA             ;for CHRGET lo  

C54F- A5 77      LDA *TXTPTR     ;get pointer to function desc.  

C551- 48          PHA             ;in CHRGET  

C552- B1 4B      LDA (DEFPNT),Y ;point to function pointer hi  

C554- 85 77      STA *TXTPTR     ;get that too  

C556- C8          INY             ;and complete CHRGET  

C557- B1 4B      LDA (DEFPNT),Y ;get pointer to formal variable  

C559- 85 78      STA *TXTPTR+1   ;value  

C55B- A5 45      LDA *VARPNT+1   ;on the stack  

C55D- 48          PHA             ;evaluate function in FLP accu1  

C55E- A5 44      LDA *VARPNT     ;restore pointer to original  

C560- 48          PHA             ;value of formal  

C561- 20 84 BD    JSR FRMNUM     ;variable  

C564- 68          PLA             ;in ($4B)  

C565- 85 4B      STA *DEFPNT     ;get current character  

C567- 68          PLA             ;if not end of statement  

C568- 85 4C      STA *DEFPNT+1   ;JSR CHRGOT  

C56A- 20 76 00    JSR CHRGOT     ;BEQ c572

```

```

C56F- 4C 00 8F      JMP SNERR      ;SYNTAX ERROR, then restart
C572- 68             c572 PLA      ;else get CHRGET lo back
C573- 85 77          STA *TXTPTR   ;store in CHRGET
C575- 68             PLA      ;get hi also
C576- 85 78          STA *TXTPTR+1 ;and complete CHRGET
C578- A0 00          DEFFIN LDY #0   ;point to exponent/pointer to function
C57A- 68             PLA      ;get it back
C57B- 91 4B          STA (DEFPNT),Y ;(re)store it
C57D- 68             PLA      ;get mantissa MSB/pointer to function
C57E- C8             INY      ;point to it
C57F- 91 4B          STA (DEFPNT),Y ;(re)store it
C581- 68             PLA      ;get mantissa/pointer+2 of formal var.
C582- C8             INY      ;point to it
C583- 91 4B          STA (DEFPNT),Y ;(re)store it
C585- 68             PLA      ;get mantissa/pointer+2 of formal var.
C586- C8             INY      ;point to it
C587- 91 4B          STA (DEFPNT),Y ;(re)store it
C589- 68             PLA      ;get mantissa LSB/copy of 1st FN char.
C58A- C8             INY      ;point to it
C58B- 91 4B          STA (DEFPNT),Y ;(re)store it
C58D- 60             RTS      ;and exit
;*****
;*          *
;* Perform STR$.          *
;*          *
;*****
C58E- 20 87 BD      STRD      JSR CHKNUM    ;test if numeric type
C591- A0 00          LDY #0       ;skip leading space (initial index)
C593- 20 95 CF      JSR FOUTC    ;convert number to string at $0100
C596- 68             PLA       ;pull return address
C597- 68             PLA
;*****
;*          *
;* Move string at $0100 to hi      *
;* memory.                      *
;*          *
;*****
C598- A9 FF          TIMSTR    LDA #L,FBUFFR-1 ;point YA
C59A- A0 00          LDY #H,FBUFFR-1 ;before the string
C59C- F0 12          BEQ STRLIT   ;and move string to memory
;*****
;*          *
;* Preserve space for the string  *
;* text in hi memory and set up  *
;* the start of string pointer   *
;* accordingly. Leave the des-   *
;* criptor in FLP accu 1.        *
;* On entry, A holds the length  *
;* of the text, and ($61) points  *
;* to it.                         *
;*          *
;*****
C59E- A6 61          STRINI    LDX *FACMO   ;get pointer to text
CSA0- A4 62          LDY *FACLO   ;in YX
CSA2- 86 40          STX *DSCPNT  ;move it to
CSA4- 84 4E          STY *DSCPNT+1 ;temporary pointer
CSA6- 20 10 C6      STRSPA    JSR GETSPA  ;set up space for stringtext
CSA9- 86 5F          STX *DSCTMP+1 ;store pointer to
CSAB- 84 60          STY *DSCTMP+2 ;and hi
CSAD- 85 5E          STA *DSCTMP   ;and length (descriptor in FLP accu1)
CSAF- 60             RTS      ;then exit
;*****
;*          *
;* Set up the descriptor of a      *

```

```

        ;* string in FLP accu 1. Move      *
        ;* the string to hi memory and   *
        ;* leave its descriptor on the   *
        ;* descriptor stack.             *
        ;* The string is searched until  *
        ;* either a the closing quotes   *
        ;* or a zero byte is found.      *
        ;*
        ;*****  

C5B0- A2 22    STRLIT LDX #'"/'          ;get string initial
C5B2- 86 03    STX *CHARAC            ;store in start flag
C5B4- 86 04    STX *ENDCHR            ;and in end flag
        ;*****  

        ;*
        ;* Set up the descriptor of a    *
        ;* string in FLP accu 1. Move    *
        ;* the string to hi memory and   *
        ;* leave its descriptor on the   *
        ;* descriptor stack.             *
        ;* The string is searched until  *
        ;* either the contents of $03,    *
        ;* $04 or a zero byte is found.  *
        ;*
        ;*****  

C5B6- 85 6C    STRLT2 STA *STRNG1       ;set pointer to
C5B8- 84 6D    STY *STRNG1+1         ;start of text in ($6C)
C5BA- 85 5F    STA *DSCTMP+1         ;set pointer in
C5BC- 84 60    STY *DSCTMP+2         ;descriptor also
C5BE- A0 FF    LDY #$FF              ;get initial offset
C5C0- C8      STRGET INY             ;point to next character
C5C1- B1 6C    LDA (STRNG1),Y       ;get character
C5C3- F0 DC    BEQ STRFI1           ;if end of line, store length
C5C5- C5 03    CMP *CHARAC          ;if same as initial
C5C7- F0 04    BEQ STRFIN           ;test for quotes
C5C9- C5 04    CMP *ENDCHR          ;if not same as terminator
C5CB- D0 F3    BNE STRGET           ;get next character
C5CD- C9 22    STRFIN CMP #'"/'      ;if quotes
C5CF- F0 01    BEQ STRFI2           ;store length
C5D1- 18      STRFI1 CLC             ;else prepare for add
C5D2- 84 5E    STRFI2 STY *DSCTMP     ;store length in $5E
C5D4- 98      TYA                 ;get length
C5D5- 65 6C    ADC *STRNG1          ;adapt start pointer lo
C5D7- 85 6E    STA *STRNG2          ;point ($6E) to end of string
C5D9- A6 6D    LDX *STRNG1+1         ;get pointer hi
C5DB- 90 01    BCC STRST2           ;if page crossed
C5DD- E8      INX                 ;adapt hi byte
C5DE- 86 6F    STRST2 STX *STRNG2+1  ;and complete pointer to end
C5E0- A5 6D    LDA *STRNG1+1         ;get start pointer hi
C5E2- F0 04    BEQ STRCP            ;if $100 is start of string,
C5E4- C9 02    CMP #H,BUF           ;or string in inputbuffer
C5E6- D0 0B    BNE PUTNEW           ;  

C5E8- 98      STRCP   TYA             ;get length in A
C5E9- 20 9E C5  JSR STRINI          ;preserve space for text
C5EC- A6 6C    LDX *STRNG1          ;get pointer to
C5EE- A4 6D    LDY *STRNG1+1         ;text in YX
CSFD- 20 9A C7 JSR MOVSTR          ;and move string to hi memory
CSF3- A6 13    PUTNEW LDX *TEMPPT     ;get desc. stack pointer
CSF5- E0 1F    CPX #INDEX           ;if at end of stack
CSF7- D0 05    BNE PUTNW1           ;  

CSF9- A2 C8    c5f9   LDX #b2d5-ERRTAB ;point to FORMULA TOO COMPLEX
CSFB- 4C CF B3 ERRG02 JMP ERROR      ;do error and restart BASIC
CSFE- A5 5E    PUTNW1 LDA *DSCTMP     ;else get length
C600- 95 00    STA *$00,X             ;store in descriptor stack
C602- A5 5F    LDA *DSCTMP+1         ;get pointer to text lo

```

C604- 95 01	STA *\$00+1,X	;store in descriptor stack
C606- A5 60	LDA *DSCTMP+2	;get pointer to text hi
C608- 95 02	STA *\$00+2,X	;store in descriptor stack
C60A- A0 00	LDY #0	;get desc. stack pointer hi
C60C- 86 61	STX *FACMO	;save current desc. stack
C60E- 84 62	STY *FACLO	;pointer in (\$61)
C610- 84 60	STY *STRNG1+1	;set start pointer invalid
C612- 88	DEY	;Y=\$FF
C613- 84 07	STY *VALTYP	;set type to string
C615- 86 14	STX *LASTPT	;set pointer to last desc.
C617- E8	INX	
C618- E8	INX	
C619- E8	INX	;point to next stack item
C61A- 86 13	STX *TEMPPT	;set up new desc. stackpointer
C61C- 60	RTS	;and exit
	.FI "B4C.M07"	

1A06 338C-4D92 B4C.M07

	;name BASIC4.OC.M07	
	;*****	
	;*	
	;* Reserve space in memory for *	
	;* the string in the input *	
	;* buffer. *	
	;*	
	;*****	
C610- 46 09	GETSPA LSR *GARBFL	;clear MSB in garbage collect counter
C61F- AA	TRYAG2 TAX	;get length in X
C620- F0 38	BEQ GETRTS	;if zero, exit
C622- 48	PHA	;else save length
C623- A5 30	LDA *FRETOP	;get start of strings lo
C625- 38	SEC	;prepare for subtract
C626- E9 02	SBC #2	;subtract length of backpointer
C628- A4 31	LDY *FRETOP+1	;get hi byte
C62A- B0 01	BCS TRYAG3	;if page crossed
C62C- 88	DEY	;adapt hi also
C62D- 85 1F	TRYAG3 STA *INDEX	;set up temporary pointer
C62F- 84 20	STY *INDEX+1	;to new backpointer in (\$1F)
C631- 8A	TXA	;get length back
C632- 49 FF	EOR #\$FF	;invert it
C634- 38	SEC	;add one (2's complement number now)
C635- 65 1F	ADC *INDEX	;calculate start of text in memory
C637- B0 01	BCS TRYAG4	;if page crossed
C639- 88	DEY	;adapt hi also
C63A- C4 2F	TRYAG4 CPY *STREND+1	;if below end of arrays now
C63C- 90 10	BCC GARBAG	;go do garbage collect
C63E- D0 04	BNE STRFRE	;if hi bytes different, skip lo
C640- C5 2E	CMP *STREND	;else if lo byte below start of arrays
C642- 90 17	BCC GARBAG	;go do garbage collect
C644- 85 32	STRFRE STA *FRESPC	;else set up utility
C646- 84 33	STY *FRESPC+1	;pointer for string
C648- A0 01	LDY #\$01	;point to backpointer hi
C64A- A9 FF	LDA #\$FF	;get invalid hi byte
C64C- 91 1F	STA (INDEX),Y	;set backpointer hi invalid
C64E- 88	DEY	;point to backpointer lo
C64F- 68	PLA	;get length back
C650- 91 1F	STA (INDEX),Y	;store in backpointer
C652- A6 32	LDX *FRESPC	;now move temporary
C654- A4 33	LDY *FRESPC+1	;pointer
C656- 86 30	STX *FRETOP	;to new start of strings
C658- 84 31	STY *FRETOP+1	;pointer
C65A- 60	GETRTS RTS	;and exit

```

;*****
;*                                     *
;* Collect all garbage strings      *
;* if not collected before. If     *
;* collected before, do OUT OF    *
;* MEMORY ERROR.                   *
;*                                     *
;*****



C65B- A2 4D      GARBAG LDX #b25a-ERRTAB ;point to OUT OF MEMORY
C65D- A5 09      LDA *GARBFL   ;get garbage collect counter
C65F- 30 9A      BMI ERRGO2  ;if collected already, OUT OF MEMORY ER
C661- 20 6A C6      JSR GARBA2  ;else collect garbage strings
C664- 38          SEC        ;and flag collection
C665- 66 09      ROR *GARBFL ;in counter
C667- 68          PLA        ;get length of string back
C668- D0 B5      BNE TRYAG2 ;and try again
;*****
;*                                     *
;* Collect all garbage strings.   *
;*                                     *
;*****



C66A- A0 00      GARBA2 LDY #$00   ;flag
C66C- 84 55      STY *HIGHDS ;no string to be moved
C66E- A5 34      LDA *MEMSIZ ;get end of strings lo
C670- A4 35      LDY *MEMSIZ+1 ;and hi
C672- 85 5C      STA *GRBTOP ;move to destination pointer
C674- 85 4B      STA *GRBPNT ;and to source pointer
C676- 85 32      STA *FRESPC ;and to temporary pointer
C678- 84 50      STY *GRBTOP+1 ;move
C67A- 84 4C      STY *GRBPNT+1 ;hi bytes
C67C- 84 33      STY *FRESPC+1 ;also
C67E- C4 31      GLOOP  CPY *FRETOP+1 ;if below end of strings hi
C680- 90 7E      BCC GRBEND ;copy pointer and exit
C682- D0 06      BNE COLOO ;if hi bytes the same
C684- C5 30      CMP *FRETOP ;if lo's are the same
C686- F0 78      BEQ GRBEND ;copy pointer and exit
C688- 90 76      BCC GRBEND ;if below that, do the same
C68A- A6 55      COLOO LDX *HIGHDS ;else get start flag
C68C- 30 05      BMI COOOB ;if not started
C68E- A9 02      LDA #$02 ;get length of backpointer
C690- 20 35 C7      JSR MOVTOP ;adapt destination pointer
C693- 20 24 C7      COOOB JSR SKIP2A ;adapt source pointer also
C696- A0 D1      LDY #1 ;point to backpointer hi
C698- B1 4B      LDA (GRBPNT),Y ;get it
C69A- C9 FF      CMP #$FF ;if valid string
C69C- D0 0B      BNE COLO1 ;go move to top
C69E- 88          COOOA DEY ;else point to backpointer lo (length)
C69F- B1 4B      LDA (GRBPNT),Y ;get length
C6A1- 20 26 C7      JSR MOVPNT ;adapt source pointer
C6A4- 38          SEC ;get 'MSB'
C6A5- 66 55      ROR *HIGHDS ;and flag strings to be moved
C6A7- D0 05      BNE GLOOP ;and do next string
C6A9- 20 44 C7      COLO1 JSR SETINX ;valid string found, move backpointer
C6AC- A6 55      LDX *HIGHDS ;if no string to be moved
C6AE- 10 53      BPL COLO3 ;adapt pointers and loop
C6B0- 46 55      LSR *HIGHDS ;else set no stringmove
C6B2- A0 00      COLO2 LDY #0 ;point to backpointer lo
C6B4- B1 4B      LDA (GRBPNT),Y ;get it
C6B6- 91 5C      STA (GRBTOP),Y ;move to new location
C6B8- C8          INY ;point to backpointer hi
C6B9- B1 4B      LDA (GRBPNT),Y ;get that too
C6BB- 91 5C      STA (GRBTOP),Y ;and move also
C6BD- 88          DEY ;Y=0
C6BE- B1 1F      LDA (INDEX),Y ;get length of string

```

```

C6C0- AA      TAX          ;in X
C6C1- 20 35 C7 JSR MOVTOP   ;adapt destination pointer
C6C4- 85 32    STA *FRESPC  ;set that pointer
C6C6- 84 33    STY *FRESPC+1 ;in temporary pointer
C6C8- 8A       TXA          ;get length back
C6C9- 20 26 C7 JSR MOVPTN   ;adapt source pointer too
C6CC- 8A       TXA          ;get length again
C6CD- A8       TAY          ;as index in Y
C6CE- 88       DEY          ;point to previous character
C6CF- B1 4B     LDA (GRBPNT),Y ;get that
C6D1- 91 5C     STA (GRBTOP),Y ;and move it upwards
C6D3- CA       DEX          ;adapt counter
C6D4- D0 F8     BNE GLOP1    ;and move until all characters done
C6D6- A0 02     LDY #2      ;then adapt pointer to text
C6D8- B9 5B 00  COLO2B    LDA GRBTOP-1,Y ;get byte from pointer
C6DB- 91 1F     STA (INDEX),Y ;move to var. area
C6DD- 88       DEY          ;adapt index
C6DE- D0 F8     BNE COLO2B  ;if pointer not complete, loop
C6E0- A5 4B     LDA *GRBPNT ;else get current source pointer
C6E2- A4 4C     LDY *GRBPNT+1 ; in YA
C6E4- C4 31     CPY *FRET0P+1 ;if hi below start of strings hi
C6E6- 90 18     BCC GRBEND  ;adapt pointer and exit
C6E8- D0 06     BNE COLO2A  ;if higher,
C6EA- C5 30     CMP *FRET0P  ;if hi's the same
C6EC- F0 12     BEQ GRBEND  ;and lo's too, adapt pointer and exit
C6EE- 90 10     BCC GRBEND  ;if below start of string, do the same
C6F0- 20 1F C7  COLO2A    JSR SKIP2   ;else skip backpointer of next
C6F3- A0 01     LDY #1      ;point to backpointer hi
C6F5- B1 4B     LDA (GRBPNT),Y ;get it
C6F7- C9 FF     CMP #$FF    ;if invalid string
C6F9- F0 A3     BEQ COLO0A  ;'remove' that too
C6FB- 20 44 C7  JSR SETINX   ;else copy backpointer to ($1F)
C6FE- 30 B2     BMI COLO2   ;and loop
C700- 4C 16 C7  GRBEND    JMP ENDGRB ;intermediate jump for exit
;*****
;*
;* Lower both source and desti- *
;* nation pointers by the length *
;* of the string currently       *
;* handled.                      *
;*
;*****
C703- A0 00     COLO3     LDY #0      ;pointer to length in var. area
C705- B1 1F     COLO3     LDA (INDEX),Y ;get length
C707- AA       COLO3     TAX         ;in X
C708- 20 35 C7  COLO3     JSR MOVTOP   ;adapt destination pointer
C708- 85 32     COLO3     STA *FRESPC  ;set new temporary
C70D- 84 33     COLO3     STY *FRESPC+1 ;pointer
C70F- 8A       COLO3     TXA          ;get length again
C710- 20 26 C7  COLO3     JSR MOVPTN   ;and adapt source pointer
C713- 4C 7E C6  COLO3     JMP GLOOP   ;then do next string
;*****
;*
;* Exit from garbage collect,   *
;* set new start of strings   *
;* pointer.                   *
;*
;*****
C716- A5 32     ENDGRB   LDA *FRESPC ;get current real
C718- A4 33     ENDGRB   LDY *FRESPC+1 ;start of strings
C71A- 85 30     ENDGRB   STA *FRET0P  ;and move to start
C71C- 84 31     ENDGRB   STY *FRET0P+1 ;of strings pointer
C71E- 60       ENDGRB   RTS          ;then exit
;*****

```

```

        ;*
        ;* Point both source and desti- *
        ;* nation pointer to end of next *
        ;* string. (Skip backpointer of *
        ;* that string). *
        ;*
        ;*****
C71F- A9 02      SKIP2   LDA #$02          ;get length of backpointer
C721- 20 35 C7    JSR MOVTOP         ;adapt source pointer by that amount
        ;*****
        ;*
        ;* Set source pointer ($4B) to *
        ;* end of next string (skip *
        ;* backpointer of that string). *
        ;*
        ;*****
C724- A9 02      SKIP2A  LDA #$02          ;get length of backpointer
.FI "B4C.M08"

```

1A42 338C-4DCE B4C.M08

```

        ;name BASIC4.OC.M08
        ;*****
        ;*
        ;* Lower source pointer ($4B) *
        ;* with the amount in A. *
        ;*
        ;*****
C726- 49 FF      MOVPNT EOR #$FF          ;invert it
C728- 38          SEC                 ;add one (2's complement number now)
C729- 65 4B      ADC *GRBPNT        ;add to lo byte
C72B- A4 4C      LDY *GRBPNT+1     ;get hi byte
C72D- B0 01      BCS MOV00          ;if page crossed
C72F- 88          DEY                 ;adapt hi byte also
C730- 85 4B      STA *GRBPNT        ;and store new
C732- 84 4C      STY *GRBPNT+1     ;source pointer
C734- 60          RTS                 ;then exit
        ;*****
        ;*
        ;* Lower destination pointer *
        ;* ($5C) with the amount in A. *
        ;*
        ;*****
C735- 49 FF      MOVTOP EOR #$FF          ;invert length
C737- 38          SEC                 ;add one (2's complement number now)
C738- 65 5C      ADC *GRBTOP        ;add to pointer lo
C73A- A4 5D      LDY *GRBTOP+1     ;get hi in Y
C73C- B0 01      BCS MOV01          ;if page crossed
C73E- 88          DEY                 ;adapt hi byte also
C73F- 85 5C      STA *GRBTOP        ;and store new
C741- 84 5D      STY *GRBTOP+1     ;destination pointer
C743- 60          RTS                 ;then exit
        ;*****
        ;*
        ;* Move backpointer to ($1F). *
        ;*
        ;*****
C744- A0 01      SETINX LDY #1          ;point to backpointer hi
C746- B1 4B      SET00   LDA (GRBPNT),Y  ;get backpointer byte
C748- 99 1F 00    STA INDEX,Y       ;move to save area
C74B- 88          DEY                 ;if not both bytes done
C74C- 10 F8      BPL SET00          ;loop
C74E- 60          RTS                 ;else exit

```

```

;*****
;*
;* Perform string concatenation *
;* (adding two strings). *
;*
;*****
C74F- A5 62      CAT    LDA *FACLO      ;get pointer to descriptor
C751- 48          PHA      ;of 1st string
C752- A5 61      LDA *FACMO      ;on stack
C754- 48          PHA
C755- 20 81 BE    JSR EVAL       ;get descriptor pointer of 2nd string
C758- 20 89 BD    JSR CHKSTR     ;in ($61) and check for string type
C75B- 68          PLA      ;move descriptor pointer
C75C- 85 6C      STA *STRNG1    ;of 1st string
C75E- 68          PLA      ;to
C75F- 85 6D      STA *STRNG1+1  ;($6C)
C761- A0 00      LDY #0        ;point to length
C763- B1 6C      LDA (STRNG1),Y ;get length of 1st string
C765- 18          CLC      ;prepare for add
C766- 71 61      ADC (FACMO),Y ;add length of 2nd string
C768- 90 05      BCC SIZEOK    ;if longer than 256 characters
C76A- A2 B0      c76a     LDX #b2bd-ERRTAB ;point to STRING TOO LONG
C76C- 4C CF B3    JMP ERROR     ;and go do error and restart
C76F- 20 9E C5    SIZEOK     JSR STRINI    ;else preserve space for result
C772- 20 8C C7    JSR MOVINS    ;move result to hi memory
C775- A5 4D      LDA *DSCPNT    ;get pointer to descriptor of
C777- A4 4E      LDY *DSCPNT+1  ;2nd string
C779- 20 BC C7    JSR FRETMP    ;throw away 2nd string
C77C- 20 9E C7    JSR MOVD0     ;move 2nd string to hi memory
C77F- A5 6C      LDA *STRNG1    ;get pointer to descriptor
C781- A4 6D      LDY *STRNG1+1  ;of 1st string
C783- 20 BC C7    JSR FRETMP    ;throw away 1st string
C786- 20 F3 C5    JSR PUTNEW    ;add descriptor to desc. stack
C789- 4C B2 BD    JMP TSTOP     ;and evaluate rest of expression
;*****
;*
;* Store string text pointed to *
;* by ($6C) in hi memory. *
;*
;*****
C78C- A0 00      MOVINS     LDY #0        ;point to length
C78E- B1 6C      MOVINS     LDA (STRNG1),Y ;get it
C790- 48          PHA      ;save length on stack
C791- C8          INY      ;point to textpointer lo
C792- B1 6C      LDA (STRNG1),Y ;get it
C794- AA          TAX      ;in X
C795- C8          INY      ;point to textpointer hi
C796- B1 6C      LDA (STRNG1),Y ;get it
C798- A8          TAY      ;in Y
C799- 68          PLA      ;and get length in A
;*****
;*
;* Store string text pointed to *
;* by YX with length in A in hi *
;* memory. *
;*
;*****
C79A- 86 1F      MOVSTR     STX *INDEX    ;set up temporary
C79C- 84 20      MOVSTR     STY *INDEX+1  ;pointer to text in ($1F)
C79E- A8          MOVD0     TAY      ;get length in Y
C79F- F0 0A      BEQ MVDONE   ;if zero, adapt pointer and exit
C7A1- 48          PHA      ;save length on stack
C7A2- 88          MOVLP     DEY      ;adapt index
C7A3- B1 1F      LDA (INDEX),Y ;get character from text

```

```

C7A5- 91 32      STA (FRESPC),Y ;move to hi memory
C7A7- 98          TYA ;get current index
C7A8- 00 F8          BNE MOVLP ;if not all characters done, loop
C7AA- 68          PLA ;else get length back
C7AB- 18          MVDONE CLC ;prepare for add
C7AC- 65 32          ADC *FRESPC ;adapt temporary pointer
C7AE- 85 32          STA *FRESPC ;save it
C7B0- 90 02          BCC MVSTRT ;if page crossed
C7B2- E6 33          INC *FRESPC+1 ;adapt hi also
C7B4- 60          MVSTRT RTS ;and exit
;*****
;*           *
;* Throw away the string with   *
;* ($61) pointing to its des-   *
;* criptor.                    *
;*           *
;*****
C7B5- 20 89 BD  FRESTR JSR CHKSTR ;check if string type
C7B8- A5 61  FREFAC LDA *FACMO ;get pointer to descriptor
C7BA- A4 62  LDY *FACLO ;in YA
;*****
;*           *
;* Throw away the string with YA *
;* pointing to its descriptor.   *
;*           *
;*****
C7BC- 85 1F  FRETMP STA *INDEX ;set pointer to descriptor
C7BE- 84 20  STY *INDEX+1 ;in ($1F)
C7CO- 20 11 C8  JSR FRETMS ;throw away descriptor on descr. stack
C7C3- 00 39  BNE FREO2 ;if descriptor thrown away
C7C5- 20 4E BA  JSR STRADJ ;set up pointer to backpointer
C7C8- 90 34  BCC FREO2 ;if not set up, exit
C7CA- 88  DEY ;point to backpointer hi (Y=$1)
C7CB- A9 FF  LDA #$FF ;get invalid backpointer hi
C7CD- 91 1F  STA (INDEX),Y ;store it
C7CF- 88  DEY ;point to backpointer lo (Y=$0)
C7D0- 8A  TXA ;get length back
C7D1- 91 1F  STA (INDEX),Y ;and complete invalid backpointer
;throws away string with next garbage collect)
C7D3- 48  PHA ;save length on stack
C7D4- 49 FF  EOR #$FF ;invert it
C7D6- 38  SEC ;add one (2's complement number now)
C7D7- 65 1F  ADC *INDEX ;point ($1F) to start of text
C7D9- A4 20  LDY *INDEX+1 ;get hi byte
C7DB- 80 01  BCS RESOO ;if page crossed
C7DD- 88  DEY ;adapt hi
C7DE- 85 1F  RESOO STA *INDEX ;and complete
C7EO- 84 20  STY *INDEX+1 ;pointer to text
C7E2- AA  TAX ;save lo byte in X
C7E3- 68  PLA ;get length back
C7E4- C4 31.  CPY *FRETOP+1 ;if not at start of strings hi
C7E6- 00 39  BNE FRERTS ;exit
C7E8- E4 30  CPX *FRETOP ;if lo not at start of strings
C7EA- 00 35  BNE FRERTS ;exit
C7EC- 48  PHA ;else save length
C7ED- 38  SEC ;add one (for backpointer)
C7EE- 65 30  ADC *FRETOP ;move start of strings lo up
C7FO- 85 30  STA *FRETOP ;(throws away string)
C7F2- 90 02  BCC FREO1 ;if page crossed
C7F4- E6 31  INC *FRETOP+1 ;adapt hi
C7F6- E6 30  FREO1 INC *FRETOP ;correct lo for backpointer
C7F8- 00 02  BNE FREPLA ;if lo was $FF (crossed page)
C7FA- E6 31  INC *FRETOP+1 ;adapt hi again
C7FC- 68  FREPLA PLA ;get length back

```

C7FD- 60		RTS	;and exit with AXY=descriptor
C7FE- A0 00	FRE02	LDY #0	;point to length
C800- B1 1F		LDA (INDEX),Y	;get it
C802- 48		PHA	;save it on stack
C803- C8		INY	;point to text pointer lo
C804- B1 1F		LDA (INDEX),Y	;get it
C806- AA		TAX	;in X
C807- C8		INY	;point to text pointer hi
C808- B1 1F		LDA (INDEX),Y	;get that too
C80A- A8		TAY	;in Y
C80B- 86 1F		STX *INDEX	;and set up pointer to text
C80D- 84 20		STY *INDEX+1	;in (\$1F)
C80F- 68		PLA	;get length back
C810- 60		RTS	;and exit with AXY=descriptor

		;*	
		;* Remove last descriptor stack *	
		;* entry if it matches the des- *	
		;* criptor of the string that is *	
		;* thrown away. *	
		;*	

C811- C4 15	FRETMS	CPY *LASTPT+1	;if hi equals descriptor stackpointer
C813- D0 DC		BNE FRERTS	
C815- C5 14		CMP *LASTPT	;and lo also
C817- D0 08		BNE FRERTS	
C819- 85 13		STA *TEMPPT	;set new stackpointer
C81B- E9 03		SBC #3	;and point (\$14)
C81D- 85 14		STA *LASTPT	;to previous entry
C81F- A0 00		LDY #0	;exit with YA holding
C821- 60	FRERTS	RTS	;new pointer in descr. stack (Z=1)
		.FI "B4C.M09"	

1BFA 338C-4F86 B4C.M09

		;name BASIC4.OC.M09	

		;*	
		;* Perform CHR\$. *	
		;*	

C822- 20 D7 C8	CHRD	JSR CONINT	;input and evaluate a 1 byte expression
C825- 8A		TXA	;move result to A
C826- 48		PHA	;save it
C827- A9 01		LDA #1	;preserve space for a string
C829- 20 A6 C5		JSR STRSPA	;with length 1 and store descr.
C82C- 68		PLA	;get value back
C82D- A0 00		LDY #0	;point to start of text in memory
C82F- 91 5F		STA (DSCTMP+1),Y	;store value
C831- 68		PLA	;remove return address
C832- 68		PLA	;from stack
C833- 4C F3 C5		JMP PUTNEW	;and add descr. to descr. stack

		;*	
		;* Perform LEFT\$. *	
		;*	

C836- 20 97 C8	LEFTD	JSR PREAM	;get 1st two parameters
C839- D1 40		CMP (DSCPNT),Y	;if length <= length of string
C83B- 98		TYA	;get a zero (start at begin of text)
C83C- 90 04	RLEFT	BCC RLEFT1	;and return empty string
C83E- B1 40		LDA (DSCPNT),Y	;else get length from descriptor
C840- AA		TAX	;in X

```

C841- 98          TYA      ;get start value
C842- 48          RLEFT1  ;on stack
C843- 8A          RLEFT2  ;get length or parameter in A
C844- 48          RLEFT3  ;and save it on stack
C845- 20 A6 C5    JSR STRSPA ;reserve space for result
C848- A5 4D        LDA *DSCPNT ;point YA
C84A- A4 4E        LDY *DSCPNT+1 ;to descr. of result of stringexpression
C84C- 20 BC C7    JSR FRETMP ;throw that away
C84F- 68          PLA      ;get length or parameter back
C850- A8          TAY      ;in Y
C851- 68          PLA      ;get position back
C852- 18          CLC      ;prepare for add
C853- 65 1F        ADC *INDEX ;calculate start in stringtext
C855- 85 1F        STA *INDEX ;in ($1F)
C857- 90 02        BCC PULMOR ;if page crossed
C859- E6 20        INC *INDEX+1 ;adapt hi byte
C85B- 98          PULMOR  TYA   ;get length or parameter back
C85C- 20 9E C7    JSR MOVDO  ;move result to hi memory
C85F- 4C F3 C5    JMP PUTNEW ;and leave descr. on descriptor stack
;*****
;*
;* Perform RIGHT$.      *
;*
;*****
C862- 20 97 C8    RIGHTD  JSR PREAM ;get 1st two parameters
C865- 18          CLC      ;subtract one extra
C866- F1 4D        SBC (DSCPNT),Y ;calculate negative position
C868- 49 FF        EOR #$FF ;position in 2's complement now
C86A- 4C 3C C8    JMP RLEFT  ;and perform 'LEFT$'
;*****
;*
;* Perform MID$.      *
;*
;*****
C86D- A9 FF        MIDD    LDA #$FF ;get default length
C86F- 85 62        STA *FACLO ;in $62
C871- 20 76 00    JSR CHRGOT ;get current character
C874- C9 29        CMP #''
C876- F0 06        BEQ MID2  ;use length 255
C878- 20 F5 BE    JSR CHKCOM ;else test for comma
C87B- 20 D4 C8    JSR GETBYT ;read length in X and $62
C87E- 20 97 C8    MID2    JSR PREAM ;get two parameters
C881- F0 4B        BEQ GOFUC ;if position zero, ILLEGAL QUANTITY ERR
C883- CA          DEX      ;else decrement length
C884- 8A          TXA      ;get length in A
C885- 48          PHA      ;save it on stack
C886- 18          CLC      ;subtract one extra
C887- A2 00        LOX #0   ;get empty string length
C889- F1 40        SBC (DSCPNT),Y ;calculate position in stringtext
C88B- B0 B6        BCS RLEFT2 ;if negative, return empty string
C88D- 49 FF        EOR #$FF ;else convert to 2's complement
C88F- C5 62        CMP *FACLO ;if now smaller than 3rd parameter
C891- 90 B1        BCC RLEFT3 ;go do 'LEFT$'
C893- A5 62        LDA *FACLO ;else get maximum length
C895- B0 AD        BCS RLEFT3 ;and return rest of string (branch alwa
;*****
;*
;* Read 1st two parameters of      *
;* LEFT$, RIGHT$ or MID$.      *
;* The 2nd parameter is returned *
;* in YA (length or position),  *
;* the pointer to the descriptor *
;* of the result of the string   *
;* expression is left in ($4D).  *

```

```

        ;* *
;*****  

C897- 20 EF BE  PREAM  JSR CHKCLS      ;current character must be )  

C89A- 68          PLA           ;get return address lo  

C89B- A8          TAY           ;in Y  

C89C- 68          PLA           ;and return address hi  

C89D- 85 52       STA *SIZE      ;in $52  

C89F- 68          PLA           ;remove return address  

C8A0- 68          PLA           ;from JSR $0051 ($C082)  

C8A1- 68          PLA           ;get 2nd parameter (length or position)  

C8A2- AA          TAX           ;in X  

C8A3- 68          PLA           ;get descriptor pointer  

C8A4- 85 40       STA *DSCPNT    ;in  

C8A6- 68          PLA           ;($40)  

C8A7- 85 4E       STA *DSCPNT+1  ;get return address hi  

C8A9- A5 52       LDA *SIZE      ;on stack again  

C8AB- 48          PHA           ;get lo also  

C8AC- 98          TYA           ;and complete return address  

C8AD- 48          PHA           ;and get length or position  

C8AE- A0 00       LDY #0         ;in YA  

C8B0- 8A          TXA           ;then exit  

C8B1- 60          RTS           ;*****  

        ;* *  

        ;* Perform LEN.          *  

        ;* *  

        ;*****  

C8B2- 20 B8 C8  LEN   JSR LEN2        ;get length of argument in Y  

C8B5- 4C CB C4       JMP SNGFLT     ;and convert to FLP  

        ;* *  

        ;* Throw away the string pointed *  

        ;* to by ($61) and set the var. *  

        ;* type to numeric. Return with *  

        ;* Y=length of the string.    *  

        ;* *  

        ;*****  

C8B8- 20 B5 C7  LEN2  JSR FRESTR     ;throw away string  

C8B9- A2 00          LDX #0         ;set variable type  

C8BD- 86 07          STX *VALTYP    ;to numeric  

C8BF- A8          TAY           ;get the length in Y  

C8C0- 60          RTS           ;and exit  

        ;* *  

        ;* Perform ASC.          *  

        ;* *  

        ;*****  

C8C1- 20 B8 C8  ASC   JSR LEN2        ;get length in Y  

C8C4- F0 08          BEQ GOFUC      ;if zero, ILLEGAL QUANTITY ERROR  

C8C6- A0 00          LDY #0         ;point to 1st character of text  

C8C8- B1 1F          LDA (INDEX),Y  ;get it  

C8CA- A8          TAY           ;in Y  

C8CB- 4C CB C4       JMP SNGFLT     ;and convert to FLP  

        ;* *  

        ;* ILLEGAL QUANTITY ERROR, then *  

        ;* restart BASIC.          *  

        ;* *  

        ;*****  

C8CE- 4C 73 C3  GOFUC JMP FGERR      ;do ILLEGAL QUANTITY ERROR  

        ;* *  

        ;* Input 1 integer (<256) in X  *  

        ;* and $62.                  *

```

```

        ;* CHRGET points before the      *
        ;* number (skips leading commas  *
        ;* and so on).                  *
        ;*                                *
        ;*****  

C8D1- 20 70 00 GTBYTC JSR CHRGET      ;get next character
        ;*****  

        ;*                                *
        ;* Input 1 integer (<256) in X  *
        ;* and $62.                      *
        ;*                                *
        ;*****  

C8D4- 20 84 BD GETBYT JSR FRMNUM    ;check if numeric type
C8D7- 20 E3 C2 CONINT JSR POSINT    ;read one 'subscript' into ($61)
C8DA- A6 61 LDX *INDICE           ;if hi byte not zero
C8DC- D0 F0 BNE GOFUC            ;go do ILLEGAL QUANTITY ERROR
C8DE- A6 62 LDX *FACLO             ;else get the lo byte
C8E0- 4C 76 00 JMP CHRGOT          ;and get current character again
        ;*****  

        ;*                                *
        ;* Perform VAL.                 *
        ;*                                *
        ;*****  

C8E3- 20 B8 C8 VAL      JSR LEN2      ;get length of string in Y
C8E6- D0 03             BNE c8eb      ;if zero
C8E8- 4C 20 CA          JMP ZEROFC    ;return zero
C8EB- A6 77             c8eb        LDX *TXTPTR   ;else save CHRGET's
C8ED- A4 78             LDY *TXTPTR+1 ;pointer
C8EF- 86 6E              STX *CURTOL    ;in
C8F1- 84 6F              STY *CURTOL+1 ;($6E)
C8F3- A6 1F              LDX *INDEX1    ;point CHRGET
C8F5- 86 77              STX *TXTPTR    ;to start of string
C8F7- 18                CLC          ;prepare for add
C8F8- 65 1F              ADC *INDEX1    ;and point
C8FA- 85 21              STA *INDEX2    ;($21) to the end
C8FC- A6 20              LDX *INDEX1+1 ;get hi byte
C8FE- 86 78              STX *TXTPTR+1 ;complete CHRGET
C900- 90 01              BCC VAL2      ;if page crossed
C902- E8                INX          ;adapt hi byte
C903- 86 22              VAL2        STX *INDEX2+1 ;and complete pointer to end
C905- A0 00              LDY #0        ;point to backpointer lo
C907- B1 21              LDA (INDEX2),Y ;get it
C909- 48                PHA          ;on stack
C90A- 98                TYA          ;get terminator
C90B- 91 21              STA (INDEX2),Y ;in backpointer lo
C90D- 20 76 00          JSR CHRGOT    ;get current character
C910- 20 29 CE          JSR FIN       ;convert string to FLP# in accu1
C913- 68                PLA          ;get backpointer lo back
C914- A0 00              LDY #0        ;point to backpointer lo
C916- 91 21              STA (INDEX2),Y ;restore backpointer
C918- A6 6E              ST2TXT     LDX *CURTOL    ;get CHRGET's
C91A- A4 6F              LDY *CURTOL+1 ;pointer in YA
C91C- 86 77              STX *TXTPTR    ;restore
C91E- 84 78              STY *TXTPTR+1 ;CHRGET
C920- 60                VALRTS     RTS          ;and exit
        .FI "B4C.M10"

```

1A97 338C-4E23 B4C.M10

```

        ;name BASIC4.OC.M10
        ;*****  

        ;*                                *
        ;* Input and evaluate parameters *

```

```

        ;* for POKE and WAIT.          *
        ;*
        ;*****
C921- 20 84 B0  GETNUM JSR FRMNUM      ;check if numeric type
C924- 20 20 C9  GETADR JSR GETADR      ;read expression for address
C927- 20 F5 BE  COMBYT JSR CHKCOM      ;check if current char. is a comma
C92A- 4C D4 C8  JMP GETBYT      ;and read expression for value
        ;*****
        ;*
        ;* Convert FLP accu1 to a 2 byte *
        ;* integer in ($11) and AY.      *
        ;*
        ;*****
C92D- A5 63  GETADR LDA *FACSGN      ;if number is negative
C92F- 30 90      BMI GOFUC       ;go do ILLEGAL QUANTITY ERROR
C931- A5 5E      LDA *FACEXP      ;else get exponent
C933- C9 91      CMP #$91       ;if number >65535
C935- B0 97      BCS GOFUC       ;do ILLEGAL QUANTITY ERROR
C937- 20 D1 CD  JSR QINT        ;if within limits, convert to integer
C93A- A5 61      LDA *INDICE      ;get hi
C93C- A4 62      LDY *INDICE+1    ;and lo in AY
C93E- 84 11      STY *POKER      ;and in
C940- 85 12      STA *POKER+1    ;($11)
C942- 60          RTS           ;then exit
        ;*****
        ;*
        ;* Perform PEEK.              *
        ;*
        ;*****
C943- A5 12  PEEK   LDA *POKER+1    ;get current value
C945- 48          PHA           ;of ($11)
C946- A5 11      LDA *POKER      ;on the
C948- 48          PHA           ;stack
C949- 20 20 C9  JSR GETADR      ;read address to be PEEKed
C94C- A0 00      LDY #0         ;get offset zero
C94E- B1 11      GETCON  LDA (POKER),Y ;get value for address
C950- A8          TAY           ;in Y
C951- 68          DOSGFL PLA     ;restore
C952- 85 11      STA *POKER      ;($11)
C954- 68          PLA           ;from
C955- 85 12      STA *POKER+1    ;stack
C957- 4C CB C4  JMP SNGFLT      ;and go convert to floating
        ;*****
        ;*
        ;* Perform POKE.              *
        ;*
        ;*****
C95A- 20 21 C9  POKE   JSR GETNUM      ;get address and value
C95D- 8A          TXA           ;get value in A
C95E- A0 00      LDY #0         ;get offset zero
C960- 91 11      STA (POKER),Y ;store value in memory
C962- 60          RTS           ;and return to caller
        ;*****
        ;*
        ;* Perform WAIT.              *
        ;*
        ;*****
C963- 20 21 C9  FNWAIT JSR GETNUM      ;get address and AND value
C966- 86 46      STX *ANDMSK     ;save AND value
C968- A2 00      LDX #0         ;get default EOR value
C96A- 20 76 00  JSR CHRGOT      ;get current character
C96D- F0 03      BEQ STORDO     ;if end line use default EOR
C96F- 20 27 C9  JSR COMBYT      ;else read EOR value in X
C972- 86 47      STORDO STX *EORMSK ;and store in memory

```

```

C974- A0 00      LDY #0          ;get offset zero
C976- B1 11      WAITER LDA (POKER),Y ;get memory contents
C978- 45 47      EOR *EORMSK   ;invert requested bits
C97A- 25 46      AND *ANDMSK   ;remove unwanted bits
C97C- F0 F8      BEQ WAITER    ;if zero, wait
C97E- 60         ZERRTS RTS    ;else exit
;*****
;*                                     *
;* Add .5 to FLP accu 1.           *
;*                                     *
;*****                                     *
C97F- A9 C7      FADDH LDA #L,FHALF ;point YA
C981- A0 D0      LDY #H,FHALF   ;to FLP# .5
C983- 4C 9D C9      JMP FADD     ;and add that to FLP accu1
;*****
;*                                     *
;* Perform subtraction.           *
;* (FLPaccu1=FLPaccu2-FLPaccu1). *
;*                                     *
;*****                                     *
C986- 20 C2 CB    FSUB  JSR CONUPK   ;load FLP accu2 from memory
C989- A5 63      FSUBT LDA *FACSGN  ;get sign of FLP accu1
C98B- 49 FF      EOR #$FF      ;invert it
C98D- 85 63      STA *FACSGN   ;and set FLP number negative
C98F- 45 68      EOR *ARGSGN   ;add sign of FLP accu2
C991- 85 6C      STA *ARISGN   ;and store EOR of signs
C993- A5 5E      LDA *FACEXP   ;get exponent
C995- 4C A0 C9      JMP FADDT    ;and perform addition
;*****
;*                                     *
;* Loop entry in perform addi-   *
;* tion.                         *
;*                                     *
;*****                                     *
C998- 20 CF CA    FADDS JSR SHIFTR   ;shift FLP accu by A bits to the right
C99B- 90 3C      FADDT BCC FADD4    ;branch always, do add
;*****
;*                                     *
;* Perform addition.             *
;* (FLPaccu1=FLPaccu2+FLPaccu1). *
;*                                     *
;*****                                     *
C99D- 20 C2 CB    FADD  JSR CONUPK   ;load FLP accu2 from memory
C9A0- D0 03      FADDT BNE c9a5      ;if FLP accu1 is zero
C9A2- 4C 32 CD    c9a5  JMP MOVFA     ;copy FLP accu2 to FLP accu1
C9A5- A6 60      LDX *FACOV     ;else get rounding byte
C9A7- 86 53      STX *OLDOV     ;in OLDOV
C9A9- A2 66      LDX #ARGEEXP   ;point to FLP accu2
C9AB- A5 66      LDA *ARGEEXP   ;get exponent of FLP accu2
C9AD- A8         FADDC TAY       ;in Y
C9AE- F0 CE      BEQ ZERRTS    ;if FLP accu2 zero, return
C9B0- 38         SEC        ;else prepare for subtraction
C9B1- E5 5E      SBC *FACEXP   ;subtract exponent of FLP accu1
C9B3- F0 24      BEQ FADD4    ;if ex. the same, go add
C9B5- 90 12      BCC FADDA    ;if FLP accu2 higher
C9B7- 84 5E      STY *FACEXP   ;set result's exponent in accu1
C9B9- A4 68      LDY *ARGSGN   ;get sign from accu 2
C9BB- 84 63      STY *FACSGN   ;store it as sign of result
C9BD- 49 FF      EOR #$FF      ;invert sign
C9BF- 69 00      ADC #0        ;add carry (2's complement now)
C9C1- A0 00      LDY #0        ;clear
C9C3- 84 53      STY *OLDOV     ;its rounding byte
C9C5- A2 5E      LDX #FAC      ;point to FLP accu1
C9C7- D0 04      BNE FADD1    ;and shift it into position

```

C9C9- A0 00	FADDA	LDY #0	;accu1 highest, clear
C9CB- 84 60		STY *FACOV	;rounding byte
C9CD- C9 F9	FADD1	CMP #\$F9	;if exponents differ more than -8
C9CF- 30 C7		BMI FADDS	;shift smallest accu into byte position
C9D1- A8		TAY	;else get difference in Y
C9D2- A5 60		LDA *FACOV	;get rounding byte
C9D4- 56 01		LSR *1,X	;get LSBit in carry
C9D6- 20 E6 CA		JSR ROLSHF	;and shift accu into bit position
C9D9- 24 6C	FADD4	BIT *ARISGN	;get EOR of signs
C9DB- 10 57		BPL FADD2	;if the same, go add fractions
C9DD- A0 5E		LDY #FAC	;else point to accu1
C9DF- E0 66		CPX #ARGEEXP	;if accu2 was highest
C9E1- F0 02		BEQ SUBIT	;go do subtract of fractions
C9E3- A0 66		LDY #ARGEEXP	;else point to accu2
C9E5- 38	SUBIT	SEC	;add one extra
C9E6- 49 FF		EOR #\$FF	;invert 2nd guard byte (2's complement
C9E8- 65 53		ADC *OLDMOV	;and add to 1st guard byte
C9EA- 85 60		STA *FACOV	;and store rounding byte

;*			
;* Subtract accu pointed to by X *			
;* from accu pointed to by Y and *			
;* leave result in accu1. *			
;*			

C9EC- B9 04 00		LDA 4,Y	;get LSB mantissa of highest accu
C9EF- F5 04		SBC *4,X	;subtract lowest
C9F1- 85 62		STA *FACLO	;and save in accu1
C9F3- B9 03 00		LDA 3,Y	;do the same
C9F6- F5 03		SBC *3,X	;with lower middle
C9F8- 85 61		STA *FACMO	;mantissa
C9FA- B9 02 00		LDA 2,Y	;and
C9FD- F5 02		SBC *2,X	;upper middle
C9FF- 85 60		STA *FACMOH	;mantissa
CA01- B9 01 00		LDA 1,Y	;also for
CA04- F5 01		SBC *1,X	;MSB
CA06- 85 5F		STA *FACHO	;mantissa
CA08- B0 03	FADFLT	BCS NORMAL	;if no carry, sign has changed
CA0A- 20 70 CA		JSR NEGFAC	;go negate accu1 (result)
CA0D- A0 00	NORMAL	LDY #0	;set up clear rounding byte
CA0F- 98		TYA	;zero bit counter
CA10- 18		CLC	;prepare for add

;*			
;* Shift accu1 (result) to the *			
;* right until MSB mantissa *			
;* nonzero. *			
;*			

CA11- A6 5F	NORM3	LDX *FACHO	;get MSB mantissa
CA13- D0 4A		BNE NORM1	;if not zero, go invert accu1
CA15- A6 60		LDX *FACMOH	;else
CA17- 86 5F		STX *FACHO	;shift
CA19- A6 61		LDX *FACMO	;accu1
CA1B- 86 60		STX *FACMOH	;eight
CA1D- A6 62		LDX *FACLO	;bits
CA1F- 86 61		STX *FACMO	;to the
CA21- A6 60		LDX *FACOV	;right
CA23- 86 62		STX *FACLO	;(<=multiply by 256)
CA25- 84 60		STY *FACOV	;save rounding byte
CA27- 69 08		ADC #8	;add 8 to exponent
CA29- C9 20		CMP #32	;if not 32 bits shifted
CA2B- D0 E4		BNE NORM3	;shift another byte
CA2D- A9 00	ZEROFC	LDA #0	;if 32 bits shifted, set

```

CA2F- 85 5E      ZEROFL STA *FACEXP      ;exponent
CA31- 85 63      ZEROML STA *FACSGN      ;and sign to zero (=result zero)
CA33- 60          RTS                  ;and exit
.FI "B4C.M11"

```

1A39 338C-4DC5 B4C.M11

```

;name BASIC4.OC.M11
;*****
;*
;* Add accu2 to accu1 and leave *
;* result in accu1.           *
;*                               *
;*****
CA34- 65 53      FADD2   ADC *OLDOV      ;add rounding byte
CA36- 85 6D      STA *FACOV      ;save it
CA38- A5 62      LDA *FACLO      ;get LSB mantissa
CA3A- 65 6A      ADC *ARGLO      ;add accu2's LSB mantissa
CA3C- 85 62      STA *FACLO      ;and store result in accu1
CA3E- A5 61      LDA *FACMO      ;do the same for
CA40- 65 69      ADC *ARGMO      ;the lower middle
CA42- 85 61      STA *FACMO      ;mantissas
CA44- A5 60      LDA *FACMOH     ;and for
CA46- 65 68      ADC *ARGMOH     ;the upper middle
CA48- 85 60      STA *FACMOH     ;mantissas
CA4A- A5 5F      LDA *FACHO      ;also
CA4C- 65 67      ADC *ARGHO      ;with
CA4E- 85 5F      STA *FACHO      ;MSB mantissas
CA50- 4C 6C CA    JMP SQUEEZ     ;and adjust for carry from MSB
;*****
;*
;* Normalize fraction of accu1  *
;* until MSbit of MSB mantissa *
;* set. (Fraction is then >=1   *
;* and <2 or <=-1 and >-2).    *
;*
;*****
CA53- 69 01      NORM2   ADC #1        ;add 1 to exponent
CA55- 06 60      ASL *FACOV      ;and
CA57- 26 62      ROL *FACLO      ;rotate
CA59- 26 61      ROL *FACMO      ;fraction
CA5B- 26 60      ROL *FACMOH     ;1 bit
CA5D- 26 5F      ROL *FACHO      ;(=multiply by two)
CA5F- 10 F2      NORM1   BPL NORM2     ;repeat until MSbit set
CA61- 38          SEC            ;then prepare for subtraction
CA62- E5 5E      SBC *FACEXP     ;and set exponent accordingly
CA64- B0 C7      BCS ZEROFC     ;if underflow, zero result
CA66- 49 FF      EOR #$FF       ;else get 2's complement
CA68- 69 01      ADC #1        ;of exponent
CA6A- 85 5E      STA *FACEXP     ;and store it
CA6C- 90 0E      SQUEEZ BCC RNDRTS  ;if overflow in fraction
CA6E- E6 5E      RNDSHF INC *FACEXP ;adapt exponent
CA70- F0 42      BEQ OVERR      ;if exponent overflow, OVERFLOW ERROR
CA72- 66 5F      ROR *FACHO      ;else
CA74- 66 60      ROR *FACMOH     ;rotate
CA76- 66 61      ROR *FACMO      ;fraction
CA78- 66 62      ROR *FACLO      ;1 bit
CA7A- 66 60      ROR *FACOV      ;and save guard bit
CA7C- 60          RNDRTS RTS      ;then exit
;*****
;*
;* Negate accu1 (Get fraction  *
;* with opposite sign).        *

```

```

        ;* *
        ;***** *****
CA7D- A5 63    NEGFACT LDA *FACSGN      ;invert
CA7F- 49 FF     EOR #$FF          ;sign
CA81- 85 63    STA *FACSGN
CA83- A5 5F     NEGFCH LDA *FACHO      ;invert
CA85- 49 FF     EOR #$FF          ;MSB mantissa
CA87- 85 5F     STA *FACHO
CA89- A5 60     LDA *FACMOH      ;invert
CA8B- 49 FF     EOR #$FF          ;upper middle
CA8D- 85 60     STA *FACMOH      ;mantissa
CA8F- A5 61     LDA *FACMO      ;invert
CA91- 49 FF     EOR #$FF          ;lower middle
CA93- 85 61     STA *FACMO      ;mantissa
CA95- A5 62     LDA *FACLO       ;invert
CA97- 49 FF     EOR #$FF          ;LSB mantissa
CA99- 85 62     STA *FACLO
CA9B- A5 60     LDA *FACOV       ;invert
CA9D- 49 FF     EOR #$FF          ;guard bit
CA9F- 85 60     STA *FACOV
CAA1- E6 60     INC *FACOV       ;add one in guard bit
CAA3- D0 0E     BNE INCFR
CAA5- E6 62     INCFACT INC *FACLO      ;add one in LSB mantissa
CAA7- D0 0A     BNE INCFR
CAA9- E6 61     INC *FACMO       ;add one in lower middle mantissa
CAA8- D0 06     BNE INCFR
CAA9- E6 60     INC *FACMOH      ;add one in upper middle mantissa
CAA9- D0 02     BNE INCFR
CAA1- E6 5F     INC *FACHO       ;add one in MSB mantissa
CAB3- 60       INCFR RTS        ;and exit with 2's complement
        ;*****
        ;* *
        ;* OVERFLOW ERROR, then restart *
        ;* BASIC. *
        ;*
        ;*****
CAB4- A2 45     OVERR  LDX #b252-ERRTAB ;point to OVERFLOW
CAB6- 4C CF B3   JMP ERROR        ;go do error
        ;*****
CAB9- A2 22     MULSHF LDX #RESH0-1  ;point to FLP accu3
        ;*****
        ;*
        ;* Shift accu pointed to by X
        ;* A bytes to the right.
        ;* (Initially entered at $CACF).
        ;*
        ;*****
CABB- B4 04     SHFTR2 LDY *4,X      ;move LSB mantissa
CABD- 84 60     STY *FACOV       ;to guard byte
CABF- B4 03     LDY *3,X      ;move lower middle
CAC1- 94 04     STY *4,X      ;to LSB mantissa
CAC3- B4 02     LDY *2,X      ;move upper middle
CAC5- 94 03     STY *3,X      ;to lower middle
CAC7- B4 01     LDY *1,X      ;move MSB mantissa
CAC9- 94 02     STY *2,X      ;to upper middle
CACB- A4 65     LDY *BITS      ;get overflow
CACD- 94 01     STY *1,X      ;in MSB mantissa
CACF- 69 08     SHIFTR ADC. #8   ;add 8 to exponent
CAD1- 30 E8     BMI SHFTR2    ;if still not positive
CAD3- FD E6     BEQ SHFTR2    ;shift again 8 bits
CAD5- E9 08     SBC #8        ;else undo correction
CAD7- A8        TAY           ;save exponent in Y
CAD8- A5 6D     LDA *FACOV       ;get guard bit
CADA- B0 14     BCS SHFTRT    ;if exponent zero now, exit

```

```

;*****
;*
;* Rotate accu pointed to by X *
;* 1 bit to the right. *
;*
;*****
CADC- 16 01      SHFTR3 ASL *1,X           ;get MS bit
CADE- 90 02      BCC SHFTR4           ;if clear, go shift
CAED- F6 01      INC *1,X            ;else add one in MS mantissa
CAE2- 76 01      SHFTR4 ROR *1,X           ;and shift
CAE4- 76 01      ROR *1,X            ;whole fraction
;*****
;*
;* Shift accu pointed to by X *
;* 1 bit to the right. *
;* (Must be preceded with *
;* LSR *$01,X). *
;*
;*****
CAE6- 76 02      ROLSHF ROR *2,X           ;by
CAE8- 76 03      ROR *3,X            ;1 bit
CAEA- 76 04      ROR *4,X
CAEC- 6A          ROR A              ;shift guard bit too
CAED- C8          INY                ;adapt exponent
CAEE- D0 EC      BNE SHFTR3           ;if not zero, shift 1bit
CAF0- 18          SHFRTT CLC             ;clear carry
CAF1- 60          RTS                ;and exit
;*****
;*
;* Floating point constant: *
;* +1.00000000 *
;* (default STEP size in FOR). *
;*
;*****
CAF2- 81          FONE    .BY $81           ;exponent
CAF3- 00          .BY $00           ;mantissa MSB
CAF4- 00          .BY $00           ;mantissa upper middle
CAF5- 00          .BY $00           ;mantissa lower middle
CAF6- 00          .BY $00           ;mantissa LSB
;*****
;*
;* Table of constants for LOG. *
;* Numbers are: *
;* 3 (number of terms-1) *
;* +.4342559419 *
;* +.5765845412 *
;* +.9618007592 *
;* +2.885390073 *
;*
;*****
CAF7- 03          LOGCN2 .BY $03           ;series has four terms
CAF8- 7F          .BY $7F           ;exponent
CAF9- 5E          .BY $5E           ;mantissa MSB
CAFA- 56          .BY $56           ;mantissa upper middle
CAF8- CB          .BY $CB           ;mantissa lower middle
CAF9- 79          .BY $79           ;mantissa LSB
CAF8- 80          .BY $80           ;exponent
CAFE- 13          .BY $13           ;mantissa MSB
CAFF- 9B          .BY $9B           ;mantissa upper middle
CB00- 0B          .BY $0B           ;mantissa lower middle
CB01- 64          .BY $64           ;mantissa LSB
CB02- 80          .BY $80           ;exponent
CB03- 76          .BY $76           ;mantissa MSB
CB04- 38          .BY $38           ;mantissa upper middle

```

CB05- 93	.BY \$93	;mantissa lower middle
CB06- 16	.BY \$16	;mantissa LSB
CB07- 82	.BY \$82	;exponent
CB08- 38	.BY \$38	;mantissa MSB
CB09- AA	.BY \$AA	;mantissa upper middle
CBOA- 3B	.BY \$3B	;mantissa lower middle
CB0B- 20	.BY \$20	;mantissa LSB

	;* *	
	;* Floating point constant: *	
	;* +.7071067691 (.5*SQR(2)) *	
	;* *	

CB0C- 80	SQR05 .BY \$80	;exponent
CB0D- 35	.BY \$35	;mantissa MSB
CBOE- 04	.BY \$04	;mantissa upper middle
CB0F- F3	.BY \$F3	;mantissa lower middle
CB10- 34	.BY \$34	;mantissa LSB

	;* *	
	;* Floating point constant: *	
	;* +1.414213538 (SQR(2)) *	
	;* *	

CB11- 81	SQR20 .BY \$81	;exponent
CB12- 35	.BY \$35	;mantissa MSB
CB13- 04	.BY \$04	;mantissa upper middle
CB14- F3	.BY \$F3	;mantissa lower middle
CB15- 34	.BY \$34	;mantissa LSB
	.FI "B4C.M12"	

1930 338C-4CC9 B4C.M12

	;name BASIC4.OC.M12	

	;* *	
	;* Floating point constant: *	
	;* -.5000000000 *	
	;* *	

CB16- 80	NEGHLF .BY \$80	;exponent
CB17- 80	.BY \$80	;mantissa MSB
CB18- 00	.BY \$00	;mantissa upper middle
CB19- 00	.BY \$00	;mantissa lower middle
CB1A- 00	.BY \$00	;mantissa LSB

	;* *	
	;* Floating point constant: *	
	;* +.6931458097 (LOG(2)) *	
	;* *	

CB1B- 80	LOG2 .BY \$80	;exponent
CB1C- 31	.BY \$31	;mantissa MSB
CB1D- 72	.BY \$72	;mantissa upper middle
CB1E- 17	.BY \$17	;mantissa lower middle
CB1F- F8	.BY \$F8	;mantissa LSB

	;* *	
	;* Perform LOG.	
	;* (Logarithm with base e).	
	;* *	

CB20- 20 61 CD LOG	JSR SIGN	;get sign of FLP accu1

```

CB23- F0 02 BEQ LOGERR ;if accu1 zero, ILLEGAL QUANTITY ERROR
CB25- 10 03 BPL LOG1 ;if positive, go do LOG
CB27- 4C 73 C3 LOGERR JMP FCERR ;else do ILLEGAL QUANTITY ERROR
CB2A- A5 5E LOG1 LDA *FACEXP ;get exponent
CB2C- E9 7F SBC #$7F ;convert to number of shifts
CB2E- 48 PHA ;save on stack
CB2F- A9 80 LDA #$80 ;normalize accu1
CB31- 85 5E STA *FACEXP ;by setting exponent to 'zero'
CB33- A9 0C LDA #L,SQR05 ;point YA to FLP
CB35- A0 CB LDY #H,SQR05 ;constant .5*SQR(2)
CB37- 20 9D C9 JSR FADD ;add that # to FLP accu1
CB3A- A9 11 LDA #L,SQR20 ;point YA to FLP
CB3C- A0 CB LDY #H,SQR20 ;constant SQR(2)
CB3E- 20 45 CC JSR FDIV ;divide accu1 by that number
CB41- A9 F2 LDA #L,FONE ;point YA to FLP
CB43- A0 CA LDY #H,FONE ;constant 1
CB45- 20 86 C9 JSR FSUB ;subtract that from accu1
CB48- A9 F7 LDA #L,LOGCN2 ;point YA to
CB4A- A0 CA LDY #H,LOGCN2 ;polynome constants
CB4C- 20 D7 D1 JSR POLYX ;evaluate polynome
CB4F- A9 16 LDA #L,NEGHLF ;point YA to FLP
CB51- A0 CB LDY #H,NEGHLF ;constant -.5
CB53- 20 9D C9 JSR FADD ;add to FLP accu1
CB56- 68 PLA ;get number of shifts back
CB57- 20 B4 CE JSR FINLOG ;add exponent to accu
CB5A- A9 1B MULLN2 LDA #L,LOG2 ;point YA to FLP
CB5C- A0 CB LDY #H,LOG2 ;constant LOG(2)
;and multiply result with that
*****  

;*  

;* Multiply FLP# pointed to by *  

;* YA with FLP accu1 and leave *  

;* result there. *  

;*  

*****  

CB5E- 20 C2 CB FMULT JSR CONUPK ;load FLP accu2 from memory
*****  

;*  

;* Perform multiplication. *  

;* (FLPaccu1=FLPaccu1*FLPaccu2). *  

;*  

*****  

CB61- D0 03 FMULTT BNE cb66 ;if accu1 zero
CB63- 4C C1 CB JMP MULTRT ;exit (do nothing)
CB66- 20 ED CB cb66 JSR MULDIY ;add exponents
CB69- A9 00 LDA #0 ;zero
CB6B- 85 23 STA *RESHO ;result so far
CB6D- 85 24 STA *RESMOH ;(also called
CB6F- 85 25 STA *RESMO ;FLP accu3)
CB71- 85 26 STA *RESLO
CB73- A5 60 LDA *FACOV ;get rounding byte
CB75- 20 8F CB JSR MLTPLY ;multiply with accu3
CB78- A5 62 LDA *FACLO ;get LSB mantissa
CB7A- 20 8F CB JSR MLTPLY ;multiply with accu3
CB7D- A5 61 LDA *FACMO ;get lower middle mantissa
CB7F- 20 8F CB JSR MLTPLY ;multiply with accu3
CB82- A5 60 LDA *FACMOH ;get upper middle mantissa
CB84- 20 8F CB JSR MLTPLY ;multiply with accu3
CB87- A5 5F LDA *FACHO ;get MSB mantissa
CB89- 20 94 CB JSR MLTPL1 ;multiply with accu3
CB8C- 4C C5 CC JMP MOVFR ;and get result in FLP accu1
*****  

;*  

;* Calculate A*FLP accu3 in *

```

```

;* FLP accu3. *
;*
;*****
CB8F- D0 03    MLTPLY BNE MLTPL1      ;if A is zero
CB91- 4C B9 CA  JMP MULSHF       ;shift accu3 one byte right
CB94- 4A        MLTPL1 LSR A        ;else get bit in carry
CB95- 09 80     ORA #$80        ;set A to zero after 8 shifts
CB97- A8        MLTPL2 TAY        ;save A in Y
CB98- 90 19     BCC MLTPL3       ;if bit was set
CB9A- 18        CLC             ;prepare for add
CB9B- A5 26     LDA *RESLO       ;get LSB mantissa of result so far
CB9D- 65 6A     ADC *ARGLO       ;add MSB mantissa of multiplier
CB9F- 85 26     STA *RESLO       ;and store in result
CBA1- A5 25     LDA *RESMO       ;do the same
CBA3- 65 69     ADC *ARGMO       ;for the lower middle
CBA5- 85 25     STA *RESMO       ;mantissa
CBA7- A5 24     LDA *RESMOH      ;and the
CBA9- 65 68     ADC *ARGMOH      ;upper middle
CBAB- 85 24     STA *RESMOH      ;mantissa
CBAD- A5 23     LDA *RESHO       ;finish
CBAF- 65 67     ADC *ARGHO       ;with MSB
CBB1- 85 23     STA *RESHO       ;mantissa
CBB3- 66 23     MLTPL3 ROR *RESHO   ;set bits
CBB5- 66 24     ROR *RESMOH      ;in position
CBB7- 66 25     ROR *RESMO       ;for next
CBB9- 66 26     ROR *RESLO       ;loop
CBBB- 66 60     ROR *FACOV       ;do rounding byte also
CBBD- 98        TYA             ;get byte back
CBBE- 4A        LSR A          ;if not all bits done
CBBF- D0 D6     BNE MLTPL2       ;repeat
CBC1- 60        MULTRT RTS       ;else exit
;*****
;*
;* Load FLP accu2 with the FLP # *
;* pointed to by YA. *
;*
;*****
CBC2- 85 1F     CONUPK STA *INDEX    ;save pointer to #
CBC4- 84 20     STY *INDEX+1      ;in ($1F)
CBC6- A0 04     LDY #4          ;point to LSB mantissa
CBC8- B1 1F     LDA (INDEX),Y    ;get it
CBCA- 85 6A     STA *ARGLO       ;and store in accu2
CBCC- 88        DEY             ;point to lower middle mantissa
CBCD- B1 1F     LDA (INDEX),Y    ;get it
CBCF- 85 69     STA *ARGMO       ;and store in accu2
CBD1- 88        DEY             ;point to upper middle mantissa
CBD2- B1 1F     LDA (INDEX),Y    ;get it
CBD4- 85 68     STA *RESMOH      ;and store in accu2
CBD6- 88        DEY             ;point to MSB mantissa
CBD7- B1 1F     LDA (INDEX),Y    ;get it
CBD9- 85 68     STA *ARGSGN      ;save it sign of accu2
CBDB- 45 63     EOR *FACSGN      ;add sign of accu1
CBDD- 85 6C     STA *ARISGN      ;and store EOR of signs
CBDF- A5 6B     LDA *ARGSGN      ;get MSB mantissa again
CBE1- 09 80     ORA #$80        ;set the MSbit
CBE3- 85 67     STA *ARGHO       ;and store MSB mantissa
CBE5- 88        DEY             ;point to exponent
CBE6- B1 1F     LDA (INDEX),Y    ;get it
CBE8- 85 66     STA *ARGEXP       ;and store in accu2
CBEA- A5 5E     LDA *FACEXP      ;get exponent of accu1
CBEF- 60        RTS             ;and exit
;*****
;*
;* Add exponents of accu1 and   *
;
```

```

        ;* accu2 in accu1. If one of      *
        ;* the accus is zero, zero is    *
        ;* shown in accu1.              *
        ;*                               *
        ;*****  

CBED- A5 66     MULDIV LDA *ARGEXP      ;get exponent of FLP accu2
CBEF- F0 1F     MLDEXP BEQ ZEREMV    ;if zero, zero accu1 and exit
CBF1- 18         CLC                 ;else prepare for add
CBF2- 65 5E     ADC *FACEXP       ;add to exponent of FLP accu1
CBF4- 90 04     BCC TRYOFF        ;if no carry, set MSbit
CBF6- 30 1D     BMI GOOVER        ;if carry and negative, overflow
CBF8- 18         CLC                 ;prepare for add
CBF9- 2C         .BY $2C            ;else skip next instruction
.CBFA- 10 14     TRYOFF BPL ZEREMV    ;if no carry result is zero, show it
CBFC- 69 80     ADC #$80          ;else add excess 128
CBFE- 85 5E     STA *FACEXP       ;and store exponent
CC00- 00 03     BNE cc05          ;if result zero
CC02- 4C 31 CA   JMP ZEROML        ;set sign to +
CC05- A5 6C     LDA *ARISGN       ;else get EOR of signs
CC07- 85 63     STA *FACSGN       ;in sign of accu1
CC09- 60         RTS                ;and exit
        ;*****  

        ;*                               *
        ;* Set accu1 to zero if its    *
        ;* contents is negative or zero. *
        ;* If accu1 positive, give    *
        ;* OVERFLOW ERROR.           *
        ;*                               *
        ;*****  

CC0A- A5 63     MLDVEX LDA *FACSGN      ;get sign of accu1
CC0C- 49 FF     EOR #$FF          ;invert it
CC0E- 30 05     BMI GOOVER        ;if accu1 is negative
CC10- 68         ZEREMV PLA          ;pull
CC11- 68         PLA               ;return address
CC12- 4C 2D CA   JMP ZEROFC        ;and zero accu1
CC15- 4C B4 CA   GOOVER JMP OVERR    ;else do OVERFLOW ERROR
.FI "B4C.M13"

```

19D1 338C-405D B4C.M13

```

        ;name BASIC4.OC.M13
        ;*****  

        ;*                               *
        ;* Multiply FLP accu1 with 10.  *
        ;*                               *
        ;*****  

CC18- 20 42 CD  MUL10  JSR MOVAF      ;copy FLP accu1 to FLP accu2
CC1B- AA         TAX               ;  

CC1C- F0 10     BEQ MUL10R      ;if accu1 is zero, exit
CC1E- 18         CLC               ;prepare for add
CC1F- 69 02     ADC #2            ;add 2 to exponent (times 4)
CC21- B0 F2     BCS GOOVER       ;if overflow, do OVERFLOW ERROR
CC23- A2 00     FINML6 LDX #0          ;clear
CC25- 86 6C     STX *ARISGN      ;sign comparison
CC27- 20 AD C9   JSR FADDC        ;add FLP accu2 to FLP accu1 (times 5)
CC2A- E6 5E     INC *FACEXP       ;multiply FLP accu1 with 2 (times 10)
CC2C- F0 E7     BEQ GOOVER       ;if exponent zero, OVERFLOW ERROR
CC2E- 60         MUL10R RTS          ;else exit
        ;*****  

        ;*                               *
        ;* Floating point constant:    *
        ;*      +10.00000000             *
        ;*                               *

```

```

;*****
CC2F- 84      TENC   .BY $84          ;exponent
CC30- 20      .BY $20          ;mantissa MSB
CC31- 00      .BY $00          ;mantissa upper middle
CC32- 00      .BY $00          ;mantissa lower middle
CC33- 00      .BY $00          ;mantissa LSB
;*****
;*
;* Divide FLP accu1 by 10. *
;*
;*****
CC34- 20 42 CD DIV10  JSR MOVAF      ;copy FLP accu1 in FLP accu2
CC37- A9 2F      LDA #L,TENC      ;point YA
CC39- A0 CC      LDY #H,TENC      ;to FLP constant 10
CC3B- A2 00      LDX #0          ;clear
CC3D- 86 6C      FDIVF   STX *ARISGN  ;sign comparison
CC3F- 20 D8 CC      JSR MOVFM      ;load FLP accu1 with constant
CC42- 4C 48 CC      JMP FDIVT      ;and perform division
;*****
;*
;* Perform division: *
;* FLPaccu1=FLPaccu2/FLPaccu1. *
;*
;*****
CC45- 20 C2 CB  FDIV   JSR CONUPK    ;load FLP accu2 from memory
CC48- F0 76      FDIVT   BEQ DVOERR   ;if accu1=0, DIVISION BY ZERO ERROR
CC4A- 20 51 CD      JSR ROUND     ;else round FLP accu1
CC4D- A9 00      LDA #0          ;get initial exponent
CC4F- 38          SEC             prepare for subtraction
CC50- E5 5E      SBC *FACEXP   ;get exponent with opposite sign
CC52- 85 5E      STA *FACEXP   ;and store it
CC54- 20 ED CB  JSR MULDIV   ;'add' exponents
CC57- E6 5E      INC *FACEXP   ;add one (2's complement)
CC59- F0 BA      BEQ GOOVER    ;if zero now, OVERFLOW ERROR
CC5B- A2 FC      LDX #$FC        ;get -4 as index (point to $22)
CC5D- A9 01      LDA #1          ;set carry after 8 bits, initial result
CC5F- A4 67      DIVIDE  LDY *ARGHO   ;get mantissa MSB, accu2
CC61- C4 5F      CPY *FACHO    ;if not same as accu1
CC63- D0 10      BNE SAVQUO   ;go adapt result
CC65- A4 68      LDY *ARGMOH   ;else get mantissa upper middle
CC67- C4 60      CPY *FACMOH   ;if not same as accu1
CC69- D0 0A      BNE SAVQUO   ;go adapt result
CC6B- A4 69      LDY *ARGMO   ;else get mantissa lower middle
CC6D- C4 61      CPY *FACMO   ;if not same as accu1
CC6F- D0 04      BNE SAVQUO   ;go adapt result
CC71- A4 6A      LDY *ARGLO   ;else get mantissa LSB
CC73- C4 62      CPY *FACLO   ;compare with accu1
CC75- 08          SAVQUO PHP      ;save the flags
CC76- 2A          ROL A          ;multiply A by two
CC77- 90 09      BCC QSHFT    ;if MSB was clear, loop
CC79- E8          INX             ;else point to next byte
CC7A- 95 26      STA *RESLO,X  ;store A in intermediate result
CC7C- F0 32      BEQ LD100    ;if X=0, do another 2 bits
CC7E- 10 34      BPL DIVNRM   ;if X is positive, calculate rounding b
CC80- A9 01      LDA #1          ;else set new byte counter flag
CC82- 28          QSHFT   PLP      ;get the flags back
CC83- B0 0E      BCS DIVSUB   ;if accu2 highest, subtract divisor
CC85- 06 6A      SHFARG  ASL *ARGLO  ;else
CC87- 26 69      ROL *ARGMO   ;multiply
CC89- 26 68      ROL *ARGMOH  ;FLP accu2
CC8B- 26 67      ROL *ARGHO   ;with two
CC8D- B0 E6      BCS SAVQUO   ;if MSB was set, adapt result
CC8F- 30 CE      BMI DIVIDE   ;if MSB now set, compare again
CC91- 10 E2      BPL SAVQUO   ;else adapt result

```

```

CC93- A8      DIVSUB TAY          ;save result in Y
CC94- A5 6A    LDA *ARGLO        ;and subtract
CC96- E5 62    SBC *FACLO        ;divisor
CC98- 85 6A    STA *ARGLO        ;from
CC9A- A5 69    LDA *ARGMO        ;FLP accu2
CC9C- E5 61    SBC *FACMO        ;and
CC9E- 85 69    STA *ARGMO        ;store
CCA0- A5 68    LDA *ARGMOH       ;new
CCA2- E5 60    SBC *FACMOH       ;result
CCA4- 85 68    STA *ARGMOH       ;in
CCA6- A5 67    LDA *ARGHO        ;FLP accu2
CCA8- E5 5F    SBC *FACHO        ;
CCA9- 85 67    STA *ARGHO        ;
CCAC- 98      TYA              ;get partial result back
CCAD- 4C 85 CC  JMP SHFARG       ;and do next byte
CCB0- A9 40    LD100  LDA #$40     ;flag two extra bits to do
CCB2- D0 CE    ccb2   BNE QSHFT    ;and handle those
CCB4- DA      DIVNRM ASL A      ;multiply
CCB5- DA      ASL A            ;last
CCB6- DA      ASL A            ;2 bits'
CCB7- DA      ASL A            ;result
CCB8- DA      ASL A            ;with
CCB9- DA      ASL A            ;64
CCBA- 85 6D    STA *FACOV       ;and save as guard bits
CCBC- 28      PLP              ;get the flags back (clean stack)
CCBD- 4C C5 CC  JMP MOVFR        ;and move result to FLP accu1
;*****
;*                                     *
;* DIVISION BY ZERO ERROR, then      *
;* restart BASIC.                   *
;*                                     *
;*****                                *
CCCC- A2 85    DVOERR LDX #b292-ERRTAB ;point to DIVISION BY ZERO
CCC2- 4C CF B3  JMP ERROR         ;go do error and restart
;*****
;*                                     *
;* Move intermediate result to      *
;* to FLP accu 1.                   *
;*                                     *
;*****                                *
CCC5- A5 23    MOVFR  LDA *RESHO      ;get MSB mantissa of result
CCC7- 85 5F    STA *FACMO        ;in FLP accu1
CCC9- A5 24    LDA *RESMOH       ;get upper middle mantissa
CCCB- 85 60    STA *FACMOH       ;in FLP accu1
CCCD- A5 25    LDA *RESMO        ;get lower middle mantissa result
CCCF- 85 61    STA *FACMO        ;in FLP accu1
CCD1- A5 26    LDA *RESLO         ;get LSB mantissa
CCD3- 85 62    STA *FACLO         ;and complete result
CCD5- 4C 0D CA  JMP NORMAL        ;then normalize FLP accu1
;*****
;*                                     *
;* Load FLP accu 1 from memory.    *
;* The number to be loaded is      *
;* pointed to by YA.               *
;*                                     *
;*****                                *
CCD8- 85 1F    MOVFM  STA *INDEX       ;set pointer to
CCDA- 84 20    STY *INDEX+1      ;number in ($1F)
CCDC- A0 04    ccdc   LDY #4        ;move 5 bytes
CCDE- B1 1F    LDA (INDEX),Y      ;get mantissa LSB
CCE0- 85 62    STA *FACLO        ;in accu1
CCE2- 88      DEY              ;point to mantissa lower middle
CCE3- B1 1F    LDA (INDEX),Y      ;get it
CCE5- 85 61    STA *FACMO        ;in accu1

```

CCE7- 88	DEY	;point to mantissa upper middle
CCE8- B1 1F	LDA (INDEX),Y	;get it
CCEA- 85 60	STA *FACMOH	;in accu1
CCEC- 88	DEY	;point to mantissa MSB
CCED- B1 1F	LDA (INDEX),Y	;get it
CCEF- 85 63	STA *FACSGN	;in the sign byte
CCF1- 09 80	ORA #\$80	;set the MSB
CCF3- 85 5F	STA *FACHO	;and store as mantissa MSB
CCF5- 88	DEY	;point to the
CCF6- B1 1F	LDA (INDEX),Y	;exponent
CCF8- 85 5E	STA *FACEXP	;and set it in accu1
CCFA- 84 60	STY *FACOV	;then clear the rounding byte
CCFC- 60	RTS	;and exit

	;*	
	;* Store FLP accu 1 in memory. *	
	;* The contents is moved to \$59 *	
	;* and further (save area for *	
	;* FLP accu1 during polynome *	
	;* calculations). *	
	;*	

CCFD- A2 59	MOV2F	LDX #L,TEMPF2 ;point to save area
CCFF- 2C		.BY \$2C ;skip next instruction

	;*	
	;* Store FLP accu 1 in memory. *	
	;* The contents is moved to \$54 *	
	;* and further (save area for *	
	;* FLP accu1 during evaluation *	
	;* of TAN). *	

CD00- A2 54	MOV1F	LDX #L,TEMPF1 ;point to save area
CD02- A0 00		LDY #H,TEMPF1 ;get hi byte also
CD04- F0 04		BEQ MOVMF ;and move to YX

	;*	
	;* Store FLP accu 1 in memory. *	
	;* The contents is moved to *	
	;* (\$46). *	

CD06- A6 46	MOVVF	LDX *FORPNT ;load YX
CD08- A4 47		LDY *FORPNT+1 ;with (\$46) and:
	.FI "B4C.M14"	

1AB0 338C-4E3C B4C.M14

```

;name BASIC4.OC.M14
*****
;*
;* Store FLP accu 1 in memory. *
;* The contents is moved to the *
;* location pointed to by YX. *
;*
*****
CDOA- 20 51 CD  MOVMF JSR ROUND ;round FLP accu1
CD0D- 86 1F      STX *INDEX   ;set pointer to
CD0F- 84 20      STY *INDEX+1 ;destination in ($1F)
CD11- A0 04      LDY #4      ;move five bytes
CD13- A5 62      LDA *FACLO  ;get mantissa LSB
CD15- 91 1F      STA (INDEX),Y ;into memory

```

```

CD17- 88      DEY
CD18- A5 61    LDA *FACMO      ;get mantissa lower middle
CD1A- 91 1F    STA (INDEX),Y   ;into memory
CD1C- 88      DEY
CD1D- A5 60    LDA *FACMOH     ;get mantissa upper middle
CD1F- 91 1F    STA (INDEX),Y   ;into memory
CD21- 88      DEY
CD22- A5 63    LDA *FACSGN     ;get sign
CD24- 09 7F    ORA #$7F        ;set bits 0 through 6
CD26- 25 5F    AND *FACHO      ;add sign to MSB mantissa
CD28- 91 1F    STA (INDEX),Y   ;move to memory
CD2A- 88      DEY
CD2B- A5 5E    LDA *FACEXP      ;then
CD2D- 91 1F    STA (INDEX),Y   ;move the exponent
CD2F- 84 60    STY *FACOV      ;clear the rounding byte
CD31- 60      RTS            ;and exit
;*****
;*
;* Copy FLP accu2 to FLP accu1. *
;*
;*****
CD32- A5 6B    MOVFA  LDA *ARGSGN    ;get the sign of accu2
CD34- 85 63    MOVFA1 STA *FACSGN    ;in accu1
CD36- A2 05    LDX #5          ;move five bytes
CD38- B5 65    MOVFAL LDA *ARGEEXP-1,X ;get byte from accu2
CD3A- 95 50    STA *FAC-1,X     ;in accu1
CD3C- CA       DEX             ;adapt index
CD3D- D0 F9    BNE MOVFAL      ;if not all done, loop
CD3F- 86 60    STX *FACOV      ;else clear rounding byte
CD41- 60      RTS            ;and exit
;*****
;*
;* Round FLP accu1 and copy it  *
;* into FLP accu2.              *
;*
;*****
CD42- 20 51 CD  MOVAF  JSR ROUND     ;round FLP accu1
CD45- A2 06    MOVEF  LDX #6          ;move six bytes
CD47- B5 5D    MOVAFL LDA *FAC-1,X   ;get byte from accu1
CD49- 95 65    STA *ARGEEXP-1,X   ;in accu2
CD4B- CA       DEX             ;adapt index
CD4C- D0 F9    BNE MOVAFL      ;if not all done, loop
CD4E- 86 60    STX *FACOV      ;else clear rounding byte
CD50- 60      MOVRTS RTS          ;and exit
;*****
;*
;* Round FLP accu1.              *
;*
;*****
CD51- A5 5E    ROUND  LDA *FACEXP    ;get exponent
CD53- F0 FB    BEQ MOVRTS      ;if accu1 zero, exit
CD55- 06 60    ASL *FACOV      ;get rounding bit in carry
CD57- 90 F7    BCC MOVRTS      ;if nothing to round, exit
CD59- 20 A5 CA INCRNDO JSR INCFCAC ;else add 1 to mantissa
CD5C- D0 F2    BNE MOVRTS      ;if no overflow in MSB mantissa, exit
CD5E- 4C 6E CA JMP RNDSHF      ;else give OVERFLOW ERROR
;*****
;*
;* Get sign of FLP accu1 in A.   *
;* Zero:   A=0.                  *
;* Positive: A=1.                *
;* Negative: A=FF.               *
;*
;*****

```

CD61- A5 5E	SIGN	LDA *FACEXP	;get exponent
CD63- F0 09		BEQ SIGNRT	;if accu1 zero, exit with A=0
CD65- A5 63	FCSIGN	LDA *FACSGN	;else get sign
CD67- 2A	FCOMPS	ROL A	;in carry
CD68- A9 FF		LDA #\$FF	;get value for negative
CD6A- B0 02		BCS SIGNRT	;if negative, exit with A=FF
CD6C- A9 01		LDA #1	;if positive get value for that
CD6E- 60	SIGNRT	RTS	;and exit

		;*	
		;* Perform SGN.	
		;*	
		;*****	
CD6F- 20 61 CD	SGN	JSR SIGN	;get sign of FLP accu1 in A

		;*	
		;* Store the value of A as	
		;* floating point number in	
		;* FLP accu 1.	
		;*	

CD72- 85 5F	FLOAT	STA *FACHO	;store A lo byte
CD74- A9 00		LDA #0	;clear
CD76- 85 60		STA *FACMOH	;the hi byte
CD78- A2 88		LDX #\$88	;get exponent

		;*	
		;* Convert the integer in \$5F	
		;* (hi) and \$60 (lo) to floating	
		;* with the exponent held in X.	
		;*	

CD7A- A5 5F	FLOATS	LDA *FACHO	;get the lo byte
CD7C- 49 FF		EOR #\$FF	;invert it
CD7E- 2A		ROL A	;get MSbit in carry
CD7F- A9 00	FLOATC	LDA #0	;clear
CD81- 85 62		STA *FACLO	;LSB mantissa
CD83- 85 61		STA *FACMO	;and lower middle mantissa
CD85- 86 5E	FLOATB	STX *FACEXP	;store exponent
CD87- 85 60		STA *FACOV	;clear rounding byte
CD89- 85 63		STA *FACSGN	;and set sign
CD8B- 4C 08 CA		JMP FADFLT	;and normalize FLP accu1

		;*	
		;* Perform ABS.	
		;*	

CD8E- 46 63	ABS	LSR *FACSGN	;clear the MSB of the sign byte
CD90- 60		RTS	;and exit (what a long routine.....)

		;*	
		;* Compare FLP accu1 with the	
		;* FLP number pointed to by YA.	
		;* On exit A holds result:	
		;* A=0 :accu and number equal	
		;* A=1 :accu is higher	
		;* A=FF :number is higher	
		;*	

CD91- 85 21	FCOMP	STA *INDEX2	;set pointer to
CD93- 84 22	FCOMPN	STY *INDEX2+1	;number in (\$21)
CD95- A0 00		LDY #0	;point to exponent
CD97- B1 21		LDA (INDEX2),Y	;get it
CD99- C8		INY	;point to mantissa MSB

```

CD9A- AA      TAX          ;save the exponent in X
CD9B- F0 C4    BEQ SIGN     ;if zero, get sign of accu1 and exit
CD9D- B1 21    LDA (INDEX2),Y ;else get mantissa MSB (and sign)
CD9F- 45 63    EOR *FACSGN  ;add sign to sign of accu
CDA1- 30 C2    BMI FCSIGN   ;if signs opposite, get flag and exit
CDA3- E4 5E    CPX *FACEXP  ;if exponents
CDAS- D0 21    BNE FCOMPC   ;equal
CDA7- B1 21    LDA (INDEX2),Y ;get mantissa MSB again
CDA9- 09 80    ORA #$80     ;set MSbit
CDAB- C5 5F    CMP *FACHO   ;if mantissa's equal
CDAD- D0 19    BNE FCOMPC   ;point to upper middle mantissa
CDAF- C8      INY          ;and get it
CDB0- B1 21    LDA (INDEX2),Y ;if these equal
CDB2- C5 60    CMP *FACMOH  ;point to lower middle mantissa
CDB4- D0 12    BNE FCOMPC   ;get it
CDB6- C8      INY          ;if equal
CDB7- B1 21    LDA (INDEX2),Y ;point to LSB mantissa
CDB9- C5 61    CMP *FACMO   ;get rounding byte
CDBB- D0 0B    BNE FCOMPC   ;in carry
CDBD- C8      INY          ;get LSB mantissa
CDBE- A9 7F    LDA #$7F     ;add rounding byte
CDC0- C5 60    CMP *FACOV   ;if same, return with A=0
CDC2- B1 21    LDA (INDEX2),Y ;else get sign of accu
CDC4- E5 62    SBC *FACLO   ;if accu larger
CDC6- F0 28    BEQ QINTRT  ;invert it
CDC8- A5 63    FCOMPC     BCC FCOMPD   ;and set A accordingly
CDC9- 90 02    FCOMPC     EOR #$FF
CDCC- 49 FF    FCOMPD     JMP FCOMPS
CDCE- 4C 67 CD  FCOMPD     ;*****
;*
;* Convert FLP accu '1 to a four *
;* byte integer. The highest *
;* byte of the result is held in *
;* $5F, the lowest in $62.      *
;*
;*****
CDD1- A5 5E    QINT       LDA *FACEXP  ;get the exponent
CDD3- F0 4A    QINT       BEQ CLRIFAC ;if accu zero, fill it with zeroes
CDD5- 38      SEC         ;else prepare for subtract
CDD6- E9 A0    SEC         SBC #$A0    ;subtract exponent -32
CDD8- 24 63    SEC         BIT *FACSGN ;if number positive
CDDA- 10 09    SEC         BPL QISHFT ;go shift accu
CDDC- AA      TAX         ;else save exponent
CDDD- A9 FF    TAX         LDA #$FF    ;set shift in
CDDF- 85 65    STA *BITS   ;to ones
CDE1- 20 83 CA JSR NEGFCH ;negate fraction
CDE4- 8A      TXA         ;get exponent back
CDE5- A2 5E    QISHFT    LDX #FAC    ;point to accu1
CDE7- C9 F9    QISHFT    CMP #$F9    ;if less than 8 bits to be shifted
CDE9- 10 06    QISHFT    BPL QINT1  ;do bit shift
CDEB- 20 CF CA JSR SHIFTR ;else shift accu into position
CDEE- 84 65    STY *BITS   ;clear the shift in
CDF0- 60      QINTRT    RTS        ;and exit
CDF1- A8      QINT1     TAY         ;get number of bit shifts in Y
CDF2- A5 63    QINT1     LDA *FACSGN ;get sign
CDF4- 29 80    QINT1     AND #$80   ;bit only
CDF6- 46 5F    QINT1     LSR *FACHO ;move MSB mantissa
CDF8- 05 5F    QINT1     ORA *FACHO ;add the sign bit
CDF9- 85 5F    QINT1     STA *FACHO ;and store the mantissa
CDFC- 20 E6 CA JSR ROLSHF ;then shift accu into bit position
CDFE- 84 65    QINT1     STY *BITS  ;clear the shift in
CE01- 60      QINT1     RTS        ;and exit
.FI "B4C.M15"

```

1B50 338C-4EDC B4C.M15

```

;name BASIC4.OC.M15
;*****
;*                               *
;* Perform INT.                  *
;*                               *
;*****                         *
CE02- A5 5E      INT    LDA *FACEXP   ;get exponent of accu1
CE04- C9 A0      CMP #$A0      ;if greater than 32
CE06- B0 20      BCS INTRTS   ;exit (no conversion)
CE08- 20 D1 CD   JSR QINT     ;else convert accu1 to integer
CE0B- 84 6D      STY *FACOV   ;clear the rounding byte
CE0D- A5 63      LDA *FACSGN  ;get the sign
CE0F- 84 63      STY *FACSGN  ;set sign of accu to positive
CE11- 49 80      EOR #$80     ;invert the sign bit
CE13- 2A         ROL A       ;get sign bit in carry
CE14- A9 A0      LDA #$A0     ;get correct exponent
CE16- 85 5E      STA *FACEXP   ;in accu1
CE18- A5 62      LDA *FACLO   ;get lowest byte
CE1A- 85 03      STA *INTEGR  ;in odd/even flag
CE1C- 4C 08 CA   JMP FADFLT  ;and negate accu1 if it was negative
;*****
;*                               *
;* Clear FLP accu 1.           *
;*                               *
;*****                         *
CE1F- 85 5F      CLRFA C STA *FACHO   ;clear MSB mantissa
CE21- 85 60      STA *FACMOH  ;clear upper middle mantissa
CE23- 85 61      STA *FACMO   ;clear lower middle mantissa
CE25- 85 62      STA *FACLO   ;clear LSB mantissa
CE27- A8         TAY          ;and zero Y
CE28- 60         INTRTS     RTS        ;then exit
;*****
;*                               *
;* Convert the string pointed to *
;* by CHRGET into a FLP number  *
;* in FLP accu 1.               *
;*                               *
;*****                         *
CE29- A0 00      FIN     LDY #0       ;clear
CE2B- A2 0A      LDX #10      ;eleven bytes of work area
CE2D- 94 5A      FINZLP   STY *DECCNT,X ;($5A-$64, including accu1)
CE2F- CA         DEX          ;adapt index
CE30- 10 FB      BPL FINZLP  ;and loop until all done
CE32- 90 0F      BCC FINDGQ  ;if character was numeric, go handle it
CE34- C9 2D      CMP #'-'   ;else if character was
CE36- D0 04      BNE QPLUS   ;a minus sign
CE38- 86 64      STX *SGNFLG  ;flag negative number in $64
CE3A- F0 04      BEQ FINC    ;and get next character
CE3C- C9 2B      QPLUS    CMP #'+'  ;if character was not plus
CE3E- D0 05      BNE FIN1    ;check for others
CE40- 20 70 00   FINC    JSR CHRGET  ;else get next character
CE43- 90 58      FINDGQ   BCC FINDIG  ;if numeric, handle it
CE45- C9 2E      FIN1     CMP #'.'  ;else if period
CE47- F0 2E      BEQ FINDP   ;flag period
CE49- C9 45      CMP #'E'   ;if not sign for exponent
CE4B- D0 30      BNE FINE   ;and exit with accu1 holding #
CE4D- 20 70 00   JSR CHRGET  ;if E, get 1st character of exponent
CE50- 90 17      BCC FNEDG1  ;if numeric, read exponent
CE52- C9 AB      CMP #MINUTK  ;if token for -
CE54- F0 0E      BEQ FINEC1  ;handle negative exponent

```

```

CE56- C9 2D      CMP #'-'          ;if minus sign
CE58- F0 0A      BEQ FINEC1       ;do the same
CESA- C9 AA      CMP #PLUSTK     ;if token for +
CESC- F0 08      BEQ FINEC       ;handle positive exponent
CESE- C9 2B      CMP #'+'          ;if plus sign
CE60- F0 04      BEQ FINEC       ;do the same
CE62- D0 07      BNE FINEC2      ;if other character, normalize and exit
CE64- 66 5D      FINEC1 ROR *EXPSSGN ;flag negative exponent in MSB
CE66- 20 70 00    FINEC JSR CHRGET  ;get 1st digit of exponent
CE69- 90 5C      FNEDG1 BCC FINEDG  ;if numeric, handle exponent
CE6B- 24 5D      FINEC2 BIT *EXPSSGN ;if no exponent found yet
CE6D- 10 0E      BPL FINE        ;normalize and exit
CE6F- A9 00      LDA #0          ;else
CE71- 38          SEC             ;prepare for subtract
CE72- E5 5B      SBC *TENEXP     ;invert exponent found so far
CE74- 4C 7F CE    JMP FINE1        ;normalize accu1 and exit
CE77- 66 5C      FINDP ROR *DPTFLG  ;period found, flag in MSB of $5C
CE79- 24 5C      BIT *DPTFLG     ;if 1st period
CE7B- 50 C3      BVC FINC        ;handle it
CE7D- A5 5B      FINE  LDA *TENEXP  ;else get exponent
CE7F- 38          FINE1 SEC        ;prepare for subtract
CE80- E5 5A      SBC *DECCNT     ;subtract # of digits after period
CE82- 85 5B      STA *TENEXP     ;and store as new exponent
CE84- F0 12      BEQ FINQNG      ;if exponent zero, exit
CE86- 10 09      BPL FINMUL      ;if positive, normalize accu1
CE88- 20 34 CC    FINDIV JSR DIV10  ;else divide accu1 by 10
CE8B- E6 5B      INC *TENEXP     ;until
CE8D- D0 F9      BNE FINDIV      ;exponent zero
CE8F- F0 07      BEQ FINQNG      ;then normalize and exit
CE91- 20 18 CC    FINMUL JSR MUL10 ;positive exponent, multiply accu1 by 1
CE94- C6 5B      DEC *TENEXP     ;until
CE96- D0 F9      BNE FINMUL      ;exponent zero
CE98- A5 64      FINQNG LDA *SGNFLG ;then get sign
CE9A- 30 01      BMI NEGXQS      ;if positive
CE9C- 60          RTS             ;exit
CE9D- 4C 4B D1    NEGXQS JMP NEGOP ;else negate accu1 and exit
CEA0- 48          FINDIG PHA        ;handle ordinary digit; save it
CEA1- 24 5C      BIT *DPTFLG     ;if period found
CEA3- 10 02      BPL FINDG1      ;adapt # of digits after it
CEA5- E6 5A      INC *DECCNT     ;multiply accu1 by 10
CEA7- 20 18 CC    FINDG1 JSR MUL10 ;get current digit back
CEAA- 68          PLA             ;prepare for subtract
CEAB- 38          SEC             ;remove ASCII
CEAC- E9 30      SBC #'0'         ;add to accu1
CEAE- 20 B4 CE    JSR FINLOG      ;and loop
CEB1- 4C 40 CE    JMP FINC        ;*****
;*****
;*
;* Add current byte to FLP
;* accu 1.
;*
;*****
CEB4- 48          FINLOG PHA        ;save current value
CEB5- 20 42 CD    JSR MOVAF      ;copy accu1 to accu2
CEB8- 68          PLA             ;get value back
CEB9- 20 72 CD    JSR FLOAT       ;store the value of A as FLP #
CEBC- A5 6B      LDA *ARGSSGN   ;get sign of accu 2
CEBE- 45 63      EOR *FACSSGN   ;add sign of accu 1
CECO- 85 6C      STA *ARISGN     ;and store sign flag
CEC2- A6 5E      LDX *FACEXP     ;get exponent of accu 1
CEC4- 4C A0 C9    JMP FADDT      ;and add accu 2 to accu 1
;*****
;*
;* Check exponent size (number

```

```

;* after the E). *
;*
;*****
CEC7- A5 5B    FINEDG LDA *TENEXP      ;get result so far
CEC9- C9 0A    CMP #10      ;if less than 10
CECB- 90 09    BCC MLEX10      ;add current digit
CECD- A9 64    LDA #100      ;else get maximum
CECF- 24 5D    BIT *EXPSGN      ;if negative exponent
CED1- 30 11    BMI MLEXMI      ;store maximum
CED3- 4C B4 CA  JMP OVERR      ;else do OVERFLOW ERROR
;*****
;*
;* Multiply current value by 10 *
;* and add current digit. Store *
;* result in $58 (exponent after *
;* the E in the number). *
;*
;*****
CED6- 0A        MLEX10 ASL A      ;multiply current
CED7- 0A        ASL A      ;value by two
CED8- 18        CLC      ;prepare for add
CED9- 65 5B    ADC *TENEXP      ;add current value (times 5)
CEDB- 0A        ASL A      ;multiply result by 2 (times 10)
CEDC- 18        CLC      ;prepare for add
CEDD- A0 00    LDY #0      ;point to current digit
CEDF- 71 77    ADC (TXTPTR),Y  ;add it
CEE1- 38        SEC      ;prepare for subtract
CEE2- E9 30    SBC #'0'      ;remove ASCII
CEE4- 85 5B    MLEXMI STA *TENEXP      ;store result
CEE6- 4C 66 CE  JMP FINEC      ;and loop
;*****
;*
;* Floating point constant: *
;* +99,999,999.91 *
;* (Used by string to floating *
;* conversion routine). *
;*
;*****
CEE9- 9B        N0999 .BY $9B      ;exponent
CEEA- 3E        .BY $3E      ;MSB
CEE9- BC        .BY $BC      ;upper middle
CEEC- 1F        .BY $1F      ;lower middle
CEED- FD        .BY $FD      ;LSB
;*****
;*
;* Floating point constant: *
;* +999,999,999.8 *
;* (Used by string to floating *
;* conversion routine). *
;*
;*****
CEEE- 9E        N9999 .BY $9E      ;exponent
CEEF- 6E        .BY $6E      ;MSB
CEFO- 6B        .BY $6B      ;upper middle
CEF1- 27        .BY $27      ;lower middle
CEF2- FD        .BY $FD      ;LSB
;*****
;*
;* Floating point constant: *
;* +1,000,000,000 *
;* (Used by string to floating *
;* conversion routine). *
;*
;*****

```

```

CEF3- 9E      NMIL    .BY $9E          ;exponent
CEF4- 6E      .BY $6E          ;MSB
CEF5- 6B      .BY $6B          ;upper middle
CEF6- 28      .BY $28          ;lower middle
CEF7- 00      .BY $00          ;LSB
;*****
;*
;* Checksum byte (over C-ROM). *
;*
;*****
CEF8- F9      CKSMCO .BY $F9          ;checksum byte
;*****
;*
;* The area $CEF9-$CF77 is fil-
;* led with $AA bytes.
;*
;*****
.BA CKSMCO+$80
.FI "B4C.M16"

```

1C60 338C-4FF9 B4C.M16

```

;name BASIC4.OC.M16
;*****
;*
;* Print IN followed by line#.
;*
;*****
CF78- A9 00    INPRT   LDA #L,INTXT    ;point YA
CF7A- A0 B3    LDY #H,INTXT    ;to IN message
CF7C- 20 90 CF  JSR STROU2     ;print the message (why not STROUT?)
CF7F- A5 37    LDA *CURLIN+1  ;get current line
CF81- A6 36    LDX *CURLIN    ;number in AX
;*****
;*
;* Print AX as decimal number.
;*
;*****
CF83- 85 5F    LINPRT  STA *FACHO    ;in FLP accu 1
CF85- 86 60    STX *FACMOH    ;as integer
CF87- A2 90    LDX #$90      ;get desired exponent
CF89- 38       SEC        ;make number positive
CF8A- 20 7F CD  JSR FLOATC    ;convert to floating
CF8D- 20 95 CF  JSR FOUTC    ;convert result to string
CF90- 4C 1D BB  STROU2     ;and print it
;*****
;*
;* Convert FLP accu 1 to a
;* string starting at $100. On
;* exit YA points to the string.
;*
;*****
CF93- A0 01    FOUT    LDY #1         ;get initial index
CF95- A9 20    FOUTC   LDA #$20      ;get a blank
CF97- 24 63    BIT *FACSGN    ;if accu 1 positive
CF99- 10 02    BPL FOUT1    ;precede number with blank
CF9B- A9 20    LDA #'-'     ;else get minus sign
CF9D- 99 FF 00  FOUT1   STA LOFBUF,Y ;store in result area
CFA0- 85 63    STA *FACSGN    ;set accu positive
CFA2- 84 6E    STY *FBUFPT    ;store index in string
CFA4- C8       INY        ;point to first digit
CFA5- A9 30    LDA #'0'      ;get default 1st digit
CFA7- A6 5E    LDX *FACEXP    ;get exponent of accu1

```

```

CFA9- D0 03      BNE cfae      ;if accu 1 zero
CFAB- 4C BA D0    JMP FOUT19   ;store zero and terminator, then exit
;*****
;*
;* Normalize accu1 between 1E9   *
;* and 1E10. Exponent in base   *
;* ten is left in $5A.          *
;*                                *
;*****



CFAE- A9 00      cfae     LDA #0       ;else get a zero
CFB0- E0 80      CPX #$80    ;if exponent of accu1
CFB2- F0 02      BEQ FOUT37  ;is 'zero', multiply
CFB4- B0 09      BCS FOUT7   ;if lower than $80, divide
CFB6- A9 F3      FOUT37   LDA #L,NMIL  ;point YA
CFB8- A0 CE      LDY #H,NMIL  ;to FLP# 99,999,999.75
CFBA- 20 5E CB    JSR FMULT   ;multiply accu1 with that number
CFBD- A9 F7      LDA #$F7    ;and store
CFBF- 85 5A      FOUT7    STA *DECCNT ;correct exponent after E
CFC1- A9 EE      FOUT4    LDA #L,N9999 ;point YA to
CFC3- A0 CE      LDY #H,N9999 ;FLP# -9,999,999.91
CFC5- 20 91 CD    JSR FCOMP   ;compare accu1 with that number
CFC8- F0 1E      BEQ BIGGES  ;if equal convert accu1 to string
CFC9- 10 12      BPL FOUT9   ;if accu1 lower, multiply with 10 and 10
CFCC- A9 E9      FOUT3    LDA #L,N0999 ;accu lower, point YA
CFCE- A0 CE      LDY #H,N0999 ;to FLP number 1,000,000,000
CFD0- 20 91 CD    JSR FCOMP   ;compare accu1 with that number
CFD3- F0 02      BEQ FOUT38  ;if equal, *10 and convert to string
CFD5- 10 0E      BPL FOUT5   ;if accu1 lower, round and convert
CFD7- 20 18 CC    FOUT38   JSR MUL10  ;if accu1 >=number, multiply with 10
CFDA- C6 5A      DEC *DECCNT ;adapt exponent after E
CFDC- D0 EE      BNE FOUT3   ;if not zero, loop
CFDE- 20 34 CC    FOUT9    JSR DIV10  ;else divide accu1 by 10
CFE1- E6 5A      INC *DECCNT ;adapt exponent after E
CFE3- D0 DC      BNE FOUT4   ;and loop
CFE5- 20 7F C9    FOUT5    JSR FADDH  ;add .5 to accu1 (round it)
;*****
;*
;* Test if number is to be con- *
;* verted to scientific format.  *
;*
;*****



CFE8- 20 D1 CD    BIGGES   JSR QINT    ;convert accu1 to a four byte integer
CFEB- A2 D1      LDX #1      ;get initial position of period
CFED- A5 5A      LDA *DECCNT ;get exponent after E
CFEF- 18         CLC       ;prepare for add
CFF0- 69 0A      ADC #10    ;add ten
CFF2- 30 09      BMI FOUTPI ;if still negative, calculate
CFF4- C9 0B      CMP #11    ;if now >10
CFF6- B0 06      BCS FOUT6   ;do scientific format
CFF8- 69 FF      ADC #$FF    ;else calculate
CFFA- AA         TAX      ;position of period in X
CFFB- A9 02      LDA #2      ;get new exponent after E
CFFD- 38         FOUTPI   SEC      ;prepare for subtract
CFFE- E9 02      FOUT6    SBC #2    ;calculate correct exponent after E
D000- 85 5B      D000     STA *TENEXP ;and store it
D002- 86 5A      STX *DECCNT ;store position of period
D004- 8A         TXA      ;get position of period
D005- F0 02      BEQ FOUT39  ;if zero, or
D007- 10 13      BPL FOUT8   ;if negative
D009- A4 6E      FOUT39   LDY *FBUFPT ;get string index
D00B- A9 2E      LDA #'.'  ;get a period
D00D- C8         INY      ;point to next character in string
D00E- 99 FF 00    STA FBUFFR-1,Y ;store the period
D011- 8A         TXA      ;get position of period again

```

```

0012- F0 06      BEQ FOUT16      ;if not zero,
0014- A9 30      LDA #'0'        ;get a zero digit
0016- C8          INY            ;point to next character of string
0017- 99 FF 00      STA FBUFFR-1,Y ;store the zero
001A- 84 6E      FOUT16 STY *FBUFPT ;save string index
                                ;*****
                                ;*
                                ;* Convert the 4 byte integer in *
                                ;* accu1 to an ASCII string.      *
                                ;*
                                ;*****
001C- A0 00      FOUT8 LDY #0      ;get initial constant offset
001E- A2 80      FOUTIM LDX #$80   ;get initial digit and flag
0020- A5 62      FOUT2 LDA *FACLO  ;get lowest byte
0022- 18          CLC            ;prepare for add
0023- 79 CF D0      ADC FOUTBL+3,Y ;add constant
0026- 85 62      STA *FACLO    ;store lowest byte
0028- A5 61      LDA *FACMO    ;do the
002A- 79 CE D0      ADC FOUTBL+2,Y ;same for the
002D- 85 61      STA *FACMO    ;lower middle byte
002F- A5 60      LDA *FACMOH   ;and
0031- 79 CD D0      ADC FOUTBL+1,Y ;the
0034- 85 60      STA *FACMOH   ;upper middle byte
0036- A5 5F      LDA *FACHO    ;do
0038- 79 CC D0      ADC FOUTBL,Y ;MSB
003B- 85 5F      STA *FACHO    ;also
003D- E8          INX            ;adapt digit
003E- B0 04      BCS FOUT41   ;if overflow, test flag
0040- 10 DE      BPL FOUT2    ;if no overflow and flag clear, loop
0042- 30 02      BMI FOUT40   ;else convert to digit
0044- 30 DA      FOUT41 BMI FOUT2 ;if overflow and flag set, loop
0046- 8A          FOUT40 TXA    ;else get digit in A
0047- 90 04      BCC FOUTYP   ;if overflow
0049- 49 FF      EOR #$FF    ;get negative value
004B- 69 0A      ADC #10    ;calculate ten minus digit
004D- 69 2F      FOUTYP ADC #$2F ;convert to ASCII numeral
004F- C8          INY            ;point
0050- C8          INY            ;to
0051- C8          INY            ;next
0052- C8          INY            ;constant in table
0053- 84 44      STY *FDECPT  ;and save table index
0055- A4 6E      LDY *FBUFPT ;get string index
0057- C8          INY            ;point to next character
0058- AA          TAX            ;get flag in X
0059- 29 7F      AND #$7F    ;remove flag (ASCII digit now)
005B- 99 FF 00      STA LOFBUF,Y ;store in string
005E- C6 5A      DEC *DECCNT  ;adapt period position
0060- D0 06      BNE STXBUF   ;if not zero, skip period
0062- A9 2E      LDA #'.'    ;else get a period
0064- C8          INY            ;point to next string character
0065- 99 FF 00      STA FBUFFR-1,Y ;and add period to string
0068- 84 6E      STXBUF STY *FBUFPT ;save string index
006A- A4 44      LDY *FDECPT ;get table index
006C- 8A          TXA            ;get flag in A
006D- 49 FF      EOR #$FF    ;invert it
006F- 29 80      AND #$80    ;remove 'digit' (start with zero)
0071- AA          TAX            ;get flag in X again
0072- C0 24      CPY #FDCEND-FOUTBL ;if at end of constant table
0074- F0 04      BEQ FOULDY   ;handle exponent and terminator
0076- C0 3C      CPY #TIMEND-FOUTBL ;if not at end of TI$ table
0078- D0 A6      BNE FOUT2    ;loop
                                ;*****

```

```

        ;*
        ;* Remove trailing zeroes and      *
        ;* add exponent if necessary.    *
        ;*                                *
        ;*****  

007A- A4 6E    FOULDY LDY *FBUFPT      ;else get string index
007C- B9 FF 00  FOUT11 LDA LOBUF,Y   ;get last digit
007F- 88        DEY                  ;point to previous digit
0080- C9 30     CMP #'0'             ;if last digit zero
0082- F0 F8     BEQ FOUT11         ;loop (remove trailing zeroes)
0084- C9 2E     CMP #'.'            ;if last is a period
0086- F0 01     BEQ FOUT12         ;handle exponent
0088- C8        INY                  ;else point to next character
0089- A9 2B     FOUT12 LDA #'+'    ;get a plus sign
008B- A6 5B     LDX *TENEXP       ;get exponent after E
008D- F0 2E     BEQ FOUT17         ;if zero, store terminator and exit
008F- 10 08     BPL FOUT14         ;if positive, store plus
0091- A9 00     LDA #0              ;else get a zero
0093- 38        SEC                  ;prepare for subtract
0094- E5 5B     SBC *TENEXP       ;negate exponent after E
0096- AA        TAX                 ;save it in X
0097- A9 20     LDA #'--'          ;get a minus sign
0099- 99 01 01  FOUT14 STA FBUFFR+1,Y ;store in string
009C- A9 45     LDA #'E'           ;get exponent marker
009E- 99 00 01  STA FBUFFR,Y      ;in string
00A1- 8A        TXA                 ;get exponent after E back
00A2- A2 2F     LDX #$2F          ;get initial ASCII value
00A4- 38        SEC                  ;prepare for subtract
00A5- E8        FOUT15 INX          ;adapt ASCII digit
00A6- E9 0A     SBC #10            ;subtract 10 from exponent
00A8- B0 FB     BCS FOUT15         ;until below zero
00AA- 69 3A     ADC #$3A            ;convert value to ASCII digit (units)
00AC- 99 03 01  STA FBUFFR+3,Y   ;move to string
00AF- 8A        TXA                 ;get ASCII digit for tens
00B0- 99 02 01  STA FBUFFR+2,Y   ;move to string
00B3- A9 00     LDA #0              ;get terminator byte
00B5- 99 04 01  STA FBUFFR+4,Y   ;at end of string
00B8- F0 08     BEQ FOUT20         ;and exit
00BA- 99 FF 00  FOUT19 STA FBUFFR-1,Y ;store zero digit
00BD- A9 00     FOUT17 LDA #0          ;get terminator byte
00BF- 99 00 01  STA FBUFFR,Y      ;terminate string
00C2- A9 00     FOUT20 LDA #L,FBUFFR ;point YA
00C4- A0 01     LDY #H,FBUFFR       ;to start of string
00C6- 60        RTS                  ;and exit
.FI "B4C.M17"

```

18EA 3292-4B7C B4C.M17

```

;name BASIC4.OC.M17
;*****  

;*                                         *
;* External labels                      *
;*                                         *
;*****  

INTEGR .DE $0003      ;one byte integer from QINT
CHARAC .DE $0003       ;starting delimiter
ENDCHR .DE $0004       ;ending delimiter
COUNT .DE $0005        ;general counter
DIMFLG .DE $0006        ;flag: subscripts and integers allowed
VALTYP .DE $0007        ;variable type: string or numeric
INTFLG .DE $0008        ;numeric type: floating or integer
GARBFL .DE $0009        ;flag in garbage collect
SUBFLG .DE $000A        ;flag: subscripts allowed

```

DOMASK .DE \$000C	;mask used by relational operators
DSDESC .DE \$000D	;descriptor of DS\$
CHANNL .DE \$0010	;current CMD output file number
POKER .DE \$0011	;address to be PEAKed or POKEd
LINNUM .DE \$0011	;number read by LINGET
TEMPPT .DE \$0013	;descriptor stack pointer
LASTPT .DE \$0014	;last descriptor on dsc. stack
TEMPST .DE \$0016	;top of descriptor stack
INDEX .DE \$001F	;indirect index
INDEX1 .DE \$001F	;same
INDEX2 .DE \$0021	;2nd indirect index
RESHO .DE \$0023	;MSB mantissa of extra FLP accu
RESMOH .DE \$0024	;upper middle mantissa of same
ADDEND .DE \$0025	;stepsize in array size calc.
RESMO .DE \$0025	;lower middle mantissa of extra FLP acc
RESLO .DE \$0026	;LSB mantissa of same
VARTAB .DE \$002A	;pointer: start of variables
ARYTAB .DE \$002C	;pointer: start of arrays
STREND .DE \$002E	;pointer: end of arrays
FRETOP .DE \$0030	;pointer: start of strings
FRESPC .DE \$0032	;pointer: end of strings
MEMSIZ .DE \$0034	;pointer: top of RAM
CURLIN .DE \$0036	;current BASIC line number
VARNAM .DE \$0042	;current variable name
FDECPT .DE \$0044	;index in FLP constants in FOUT
VARPNT .DE \$0044	;pointer to current variable
FORPNT .DE \$0046	;pointer to variable
ANDMSK .DE \$0046	;AND mask in WAIT
EORMSK .DE \$0047	;EOR mask in WAIT
OPMASK .DE \$004A	;operator mask from FRMEVL
GRBPNT .DE \$004B	;pointer in garbage collect
DEFPNT .DE \$004B	;pointer to function definition
DSCPNT .DE \$004D	;pointer to string descriptor
JMPER .DE \$0051	;indirect pointer for function despatch
SIZE .DE \$0052	
OLDOV .DE \$0053	
TEMPF1 .DE \$0054	
ARYPNT .DE \$0055	
HIGHDS .DE \$0055	
HIGHTR .DE \$0057	
TEMPF2 .DE \$0059	
LOWDS .DE \$005A	
DECCTN .DE \$005A	
TENEXP .DE \$005B	
LOWTR .DE \$005C	
DPTFLG .DE \$005C	
GRBTOP .DE \$005C	
EXPSGN .DE \$005D	
FAC .DE \$005E	
FACEXP .DE \$005E	
DSCTMP .DE FAC	
FACHO .DE \$005F	
FACMOH .DE \$0060	
FACMO .DE \$0061	
INDICE .DE \$0061	
FACLO .DE \$0062	
FACSGN .DE \$0063	
SGNFLG .DE \$0064	
BITS .DE \$0065	
ARGEEXP .DE \$0066	
ARGHO .DE \$0067	
ARGMOH .DE \$0068	
ARGMO .DE \$0069	
ARGLO .DE \$006A	

ARGSGN .DE \$006B	;FLP accu2: unpacked sign
ARISGN .DE \$006C	;EOR of accu1 and 2 signs
STRNG1 .DE \$006C	;pointer to string
FACOV .DE \$006D	;old overflow; used for rounding
CURTOL .DE \$006E	;absolute linear index
STRNG2 .DE \$006E	;pointer to string
FBUFPT .DE \$006E	;pointer to string in FBUFFR
CHRGET .DE \$0070	;gets next character
CHRGOT .DE CHRGET+6	;gets current character
TXTPTR .DE CHRGET+7	;pointer to current character
CTIMR .DE \$008B	;jiffy clock
CSTAT .DE \$0096	;STatus byte
TRMPOS .DE \$00C6	;current column number of cursor
PI .DE \$00FF	;character for PI
LOFBUF .DE \$00FF	;initial character in FOUT
FBUFFR .DE \$0100	;string buffer for FOUT
BUF .DE \$0200	;BASIC's input buffer
FUNDSP .DE \$B066	;table of function addresses
ERRTAB .DE \$B200	;message: NEXT WITHOUT FOR
b21d .DE \$B21D	;message: SYNTAX
b223 .DE \$B223	;message: RETURN WITHOUT GOSUB
b237 .DE \$B237	;message: OUT OF DATA
b242 .DE \$B242	;message: ILLEGAL QUANTITY
b252 .DE \$B252	;message: OVERFLOW
b25a .DE \$B25A	;message: OUT OF MEMORY
b267 .DE \$B267	;message: UNDEF'D STATEMENT
b278 .DE \$B278	;message: BAD SUBSCRIPT
b285 .DE \$B285	;message: REDIM'D ARRAY
b292 .DE \$B292	;message: DIVISION BY ZERO
b2a2 .DE \$B2A2	;message: ILLEGAL DIRECT
b2b0 .DE \$B2B0	;message: TYPE MISMATCH
b2bd .DE \$B2BD	;message: STRING TOO LONG
b2cc .DE \$B2CC	;message: FILE DATA
b2d5 .DE \$B2D5	;message: FORMULA TOO COMPLEX
b2e8 .DE \$B2E8	;message: CAN'T CONTINUE
b2f6 .DE \$B2F6	;message: UNDEF'D FUNCTION
INTXT .DE \$B300	;message: IN
BLTU .DE \$B350	;move block in memory
REASON .DE \$B3A0	;test for overlap strings/arrays
OMERR .DE \$B3CD	;OUT OF MEMORY ERROR
ERROR .DE \$B3CF	;do error pointed to by X
DATA .DE \$B883	;perform DATA
STRADJ .DE \$BA4E	;set up pointer to backpointer
STROUT .DE \$BB1D	;print string from YA until zero
FRMNUM .DE \$BD84	;get next non-string value
CHKNUM .DE \$BD87	;test if value is numeric
CHKSTR .DE \$BD89	;test if value is string type
CHKVAL .DE \$BD8A	;test for type indicated by carry
FRMEVL .DE \$BD98	;evaluate any expression
TSTOP .DE \$BD82	;evaluate rest of expression
EVAL .DE \$BE81	;get descriptor of string in FLP accu1
PARCHK .DE \$BEE9	;test for (, then evaluate expression
CHKCLS .DE \$BEEF	;test for)
CHKOPN .DE \$BEF2	;test for (
CHKCOM .DE \$BEF5	;test for comma
SYNCHR .DE \$BEF7	;test for the character in A
SNERR .DE \$BF00	;SYNTAX ERROR exit
ISVRET .DE \$BF8F	;return address in 'get value'
CHKDS .DE \$BFFC	
FHALF .DE \$DOC7	;pointer to FLP# .5
ZERO .DE \$DOC9	;pointer to dummy value
FOUTBL .DE \$DOCC	;pointer to start of decimal conversion
FDCEND .DE \$DOFO	;end of decimal conversion table
TIMEND .DE \$D108	;end of time conversion table

```

NEGOP .DE $014B      ;store digit and terminator in string c
POLYX .DE $01D7      ;evaluate polynome
FNTK .DE $00A5      ;token for FN
PLUSTK .DE $00AA     ;token for +
MINUTK .DE $00AB     ;token for -
EQULTK .DE $00B2     ;token for =
.EN

```

END MAE PASS

LABELFILE

ABS =CD8E	ADDEND =0025	ADDIND =C44B
ANDMSK =0046	ANDOP =C089	ARGEXP =0066
ARGHO =0067	ARGLO =006A	ARGMO =0069
ARGMOH =0068	ARGSGN =006B	ARISGN =006C
ARYDON =C29D	ARYGET =C259	ARYPNT =0055
ARYTAB =002C	ARYVA2 =C21C	ARYVA3 =C220
ARYVGO =C228	ASC =C8C1	AYINT =C2EA
BIGGES =CFE8	BITS =0065	BLTU =B350
BSERR =C370	BSERR7 =C433	BUF =0200
CAT =C74F	CHANNL =0010	CHARAC =0003
CHKCLS =BEEF	CHKCOM =BEF5	CHKDS =BFFC
CHKNUM =BD87	CHKOPN =BEF2	CHKSTR =BD89
CHKVAL =BD8A	CHRD =C822	CHRGET =0070
CHRGOT =0076	CKSMCO =CEF8	CLRFAC =CE1F
COL00 =C68A	COL00A =C69E	COL00B =C693
COL01 =C6A9	COL02 =C6B2	COL02A =C6F0
COL02B =C6D8	COL03 =C703	COMBYT =C927
CONINT =C8D7	CONUPK =CBC2	COUNT =0005
CSTAT =0096	CTIMR =008B	CURLIN =0036
CURTOL =006E	DATA =B883	DECCTN =005A
DECCUR =C3F8	DEF =C4DC	DEFFIN =C578
DEFPNT =004B	DEFSTF =C541	DIM =C121
DIM3 =C11E	DIMFLG =0006	DIMRTS =C476
DIV10 =CC34	DIVIDE =CC5F	DIVNRM =CCB4
DIVSUB =CC93	DOCMP =C112	DOMASK =000C
DOREL =C0B6	D0SGFL =C951	DPTFLG =005C
DSCPNT =004D	DSCTMP =005E	DSDESC =000D
DVOERR =CCCO	DVARTS =C290	EATEM =C150
ENDCHR =0004	ENDGRB =C716	EORMSK =0047
EQULTK =00B2	ERRDIR =C4CF	ERRG02 =C5FB
ERRG03 =C375	ERRGUF =C4D7	ERROR =B3CF
ERRTAB =B200	EVAL =BE81	EXPSGN =005D
FAC =005E	FACEXP =005E	FACH0 =005F
FACLO =0062	FACMO =0061	FACMOH =0060
FACOV =0060	FACSGN =0063	FADD =C99D
FADD1 =C9CD	FADD2 =CA34	FADD4 =C9D9
FADD5 =C998	FADDA =C9C9	FADDc =C9AD
FADDH =C97F	FADDT =C9AO	FADFLT =CA08
FBUFFR =D100	FBUFPt =006E	FCERR =C373
FCOMP =CD91	FCOMPc =CDC8	FCOMPd =CDCE
FCOMPn =CD93	FCOMPs =CD67	FCSIGN =CD65
FDCEND =D0F0	FDECPT =0044	FDIV =CC45
FDIVF =CC3D	FDIVT =CC48	FHALF =D0C7
FIN =CE29	FIN1 =CE45	FINC =CE40
FINDG1 =CEA7	FINDGQ =CE43	FINDIG =CEAO
FINDIV =CE88	FINDP =CE77	FINE =CE7D
FINE1 =CE7F	FINEC =CE66	FINEC1 =CE64
FINEC2 =CE6B	FINEDG =CEC7	FINGO =C076
FINLOG =CEB4	FINML6 =CC23	FINMUL =CE91
FINNOW =C2C3	FINPTR =C2B9	FINQNG =CE98

FINZLP =CE2D	FLOAT =CD72	FLOATB =CD85
FLOATC =CD7F	FLOATS =CD7A	FMAPTR =C2C8
FMULT =CB5E	FMULTT =CB61	FNDOER =C51D
FNEDG1 =CE69	FNTK =00A5	FNWAIT =C963
FONE =CAF2	FORPNT =0046	FOULDY =D07A
FOUT =CF93	FOUT1 =CF9D	FOUT11 =D07C
FOUT12 =D089	FOUT14 =D099	FOUT15 =D0A5
FOUT16 =D01A	FOUT17 =D0BD	FOUT19 =D0BA
FOUT2 =D020	FOUT20 =D0C2	FOUT3 =CFCC
FOUT37 =CFB6	FOUT38 =CFD7	FOUT39 =D009
FOUT4 =CFC1	FOUT40 =D046	FOUT41 =D044
FOUT5 =CFE5	FOUT6 =CFFE	FOUT7 =CFBF
FOUT8 =D01C	FOUT9 =CFDE	FOUTBL =D0CC
FOUTC =CF95	FOUTIM =D01E	FOUTPI =CFFD
FOUTYP =D040	FRE =C4A8	FRE01 =C7F6
FRE02 =C7FE	FREFAC =C7B8	FREPLA =C7FC
FRERTS =C821	FRESPC =0032	FRESTR =C7B5
FRETMP =C7BC	FRETMS =C811	FRETOP =D030
FRMEVL =B098	FRMNUM =B084	FSUB =C986
FSUBT =C989	FUNDSP =B066	GARBA2 =C66A
GARBAG =C65B	GARBFL =0009	GETADR =C92D
GETBYT =C804	GETCMP =C106	GETCON =C94E
GETDEF =C415	GETFNM =C50A	GETNUM =C921
GETRTS =C65A	GETSPA =C61D	GETTIM =C003
GIVAYF =C4BC	GLOOP =C67E	GLOP1 =C6CE
GOBADV =C1DB	GOFLOT =C11B	GOFUC =C8CE
GOGO =C263	GOGO1 =C281	GOMOVF =C040
GOOVER =CC15	GOTARY =C378	GRBEND =C700
GRBPNT =D04B	GRBTOP =D05C	GREASE =C3E4
GTBYTC =C801	HIGHDS =D055	HIGHTR =D057
INCFAC =CAAS	INCFR1 =CAB3	INCRND =C059
INDEX =D01F	INDEX1 =D01F	INDEX2 =D021
INDICE =D061	INDL0P =C306	INLPN1 =C43A
INLPN2 =C439	INLPNM =C41D	INPRT =CF78
INT =CE02	INTEGR =D003	INTERR =C13C
INTFLG =D008	INTIDX =C2DD	INTRTS =CE28
INTXT =B30D	ISARY =C2FC	ISFUN =C047
ISLETC =C1B6	ISLRTS =C1BF	ISSEC =C14F
ISVRET =BF8F	JMPER =D051	JSRGM =C2D4
LASTPT =D0014	LD100 =CCB0	LDZR =C1C6
LEFTD =C836	LEN =C8B2	LEN2 =C8B8
LINNUM =D011	LINPRT =CF83	LOFBUF =D0FF
LOG =CB20	LOG1 =CB2A	LOG2 =CB1B
LOGCN2 =CAF7	LOGERR =CB27	LOPFDA =C347
LOPFDV =C353	LOPFN =C19B	LOPFND =C191
LOPPTA =C3B1	LOWDS =D05A	LOWTR =D05C
MEMSIZ =D034	MID2 =C87E	MIDD =C86D
MINUTK =D0AB	MLDEXP =CBEF	MLDVEX =CCOA
MLEX10 =CED6	MLEXMI =CEE4	MLTPL1 =CB94
MLTPL2 =CB97	MLTPL3 =CBB3	MLTPLY =CB8F
MOV00 =C730	MOV01 =C73F	MOV1F =CD00
MOV2F =CCFD	MOVAF =CD42	MOVAFL =CD47
MOVDO =C79E	MOVEF =CD45	MOVFA =CD32
MOVFA1 =CD34	MOVFAL =CD38	MOVFM =CCD8
MOVFR =CCCS	MOVINS =C78C	MOVLP =C7A2
MOVMF =CDDA	MOVPTN =C726	MOVRTS =CD50
MOVSTR =C79A	MOVTOP =C735	MOVVF =CD06
MUL10 =CC18	MUL10R =CC2E	MULDIV =CBE0
MULLN2 =CB5A	MULSHF =CAB9	MULTRT =CBC1
MVDONE =C7AB	MVSTRT =C7B4	N0999 =CEE9
N32768 =C2D9	N9999 =CEEE	NEGFAC =CA7D
NEGFCN =CA83	NEGHLF =CB16	NEGQP =D14B
NEGXQS =CE9D	NMARY1 =C362	NMIL =CEF3
NOFREF =C4AF	NONONO =C2F7	NORM1 =C45F

NORM2 =CA53	NORM3 =CA11	NORMAL =CA0D
NOSEC =C15A	NOTDIM =C3C1	NOTEVE =C208
NOTEVL =C1CB	NOTFDD =C38C	NOTFL1 =C45C
NOTFLT =C39F	NOTFNS =C1C0	NOTIT =C1AC
NOTSTR =C164	NXTCMP =C0FB	NXTPTR =C1AB
OKNORM =C071	OLDOV =D053	OMERR =B3CD
OMERR1 =C436	OPMASK =D04A	OROP =CD86
OVERR =CAB4	PARCHK =BEE9	PEEK =C943
PI =D0FF	PLUSTK =D0AA	POKE =C95A
POKER =D011	POLYX =D1D7	POS =C4C9
POSINT =C2E3	PREAM =C897	PTRGET =C12B
PTRGT1 =C130	PTRGT2 =C132	PTRGT3 =C13F
PULMOR =C85B	PUTNEW =C5F3	PUTNW1 =C5FE
QCOMP =C101	QDSAV =CD1C	QDSVAR =C1E6
QINT =CDD1	QINT1 =CDF1	QINTG0 =C2F9
QINTRT =CDFO	QISHFT =CDE5	QPLUS =CE3C
QSHFT =CC82	QSTATV =COOF	QSTAVR =C1DE
REASON =B3AD	RES00 =C7DE	RESH0 =D023
RESL0 =D026	RESM0 =D025	RESMOH =D024
RIGHTD =C862	RLEFT =C83C	RLEFT1 =C842
RLEFT2 =C843	RLEFT3 =C844	RNDRTS =CA7C
RNDSHF =CA6E	ROLSHF =CAE6	ROUND =CD51
SAVQUO =CC75	SET00 =C746	SETINX =C744
SGN =CD6F	SGNFLG =D064	SHFARG =CC85
SHFTR2 =CABB	SHFTR3 =CADC	SHFTR4 =CAE2
SHFTRT =CAF0	SHIFTR =CACF	SIGN =CD61
SIGNRT =CD6E	SIZE =D052	SIZEOK =C76F
SKIP2 =C71F	SKIP2A =C724	SNERR =BF00
SNGFLT =C4CB	SQR05 =CB0C	SQR20 =CB11
SQUEEZ =CA6C	ST2TXT =C918	STASGN =C0F6
ST0ML1 =C462	ST0MLT =C3A8	STORD0 =C972
STRADJ =BA4E	STRCMP =COCE	STRCP =C5E8
STRD =C58E	STREND =D02E	STRFI1 =C5D1
STRFI2 =C5D2	STRFIN =C5CD	STRFRE =C644
STRGET =C5C0	STRINI =C59E	STRLIT =C5B0
STRLT2 =C5B6	STRNAM =C17B	STRNG1 =D06C
STRNG2 =D06E	STROU2 =CF90	STROUT =BB1D
STRSPA =C5A6	STRST2 =C5DE	STXBUF =D068
STXFND =C18F	SUBFLG =D0DA	SUBIT =C9E5
SYNCHR =BEF7	TEMPF1 =D054	TEMPF2 =D059
TEMPPT =D013	TEMPST =D016	TENC =CC2F
TENEXP =D05B	TIMEND =D108	TIMSTR =C598
TRMPOS =D0C6	TRYAG2 =C61F	TRYAG3 =C62D
TRYAG4 =C63A	TRYOFF =CBFA	TSTOP =BDB2
TURNON =C174	TXTPTR =D077	UMLCNT =C4A3
UMLRTS =C4A7	UMULT =C477	UMULTC =C48A
UMULTD =C480	VAL =C8E3	VAL2 =C903
VALRTS =C920	VALTYP =D007	VARNAM =D042
VAROK =C1F2	VARPNT =D044	VARTAB =D02A
WAITER =C976	ZEREMV =CC10	ZERITA =C3F3
ZERO =DOC9	ZEROF1 =CA2F	ZEROFC =CA2D
ZEROML =CA31	ZERRTS =C97E	b21d =B21D
b223 =B223	b237 =B237	b242 =B242
b252 =B252	b25a =B25A	b267 =B267
b278 =B278	b285 =B285	b292 =B292
b2a2 =B2A2	b2b0 =B2B0	b2bd =B2BD
b2cc =B2CC	b2d5 =B2D5	b2e8 =B2E8
b2f6 =B2F6	c187 =C187	c3cd =C3CD
c572 =C572	c5f9 =C5F9	c76a =C76A
c8eb =C8EB	c9a5 =C9A5	cb66 =CB66
cc05 =CC05	ccb2 =CCB2	ccdc =CCDC
cfae =CFAE	d000 =D000	

//0000,DOC7,DOC7

]

```

;NAME BASIC4.0D.CTL
;*****
;*          *
;*      D-ROM for BASIC 4.0      *
;*          *
;* Date 07-01-84  Time 16:30   *
;*          *
;* Made by:          *
;*          *
;* Nico de Vries      *
;* Mari Andriessenrade 49      *
;* 2907 MA Capelle a/d Yssel   *
;* The Netherlands          *
;*          *
;* Original CBM ROM-number   *
;*          *
;*           1465-21          *
;*          *
;*****$DOC7
.FI "BASIC4.0D.M01"

```

194D 341F-4D6C BASIC4.0D.M01

		;name BASIC4.0D.M01
		;*****
		;* *
		;* Floating point constant: *
		;* +.5000000000 *
		;* (Used in SQR and for roun- *
		;* ding purposes; partly used as *
		;* dummy value). *
		;* *
		;*****
DOC7- 80	FHALF	.BY \$80 ;exponent
DOC8- 00		.BY \$00 ;mantissa MSB and sign
DOC9- 00	ZERO	.BY \$00 ;mantissa upper middle byte
DOCA- 00		.BY \$00 ;mantissa lower middle byte
DOCB- 00		.BY \$00 ;mantissa LSB
		;*****
		;* *
		;* Table of nine four-byte con- *
		;* stants used by FLP# to string *
		;* conversion routine. *
		;* *
		;*****
DOCC- FA	FOUTBL	.BY \$FA ;1st constant, MSB
DOCD- 0A		.BY \$0A ;value= -100,000,000 (=-1E8)
DOCE- 1F		.BY \$1F
DOCF- 00		.BY \$00
DOO0- 00		.BY \$00 ;2nd constant
DOO1- 98		.BY \$98 ;value= +10,000,000 (+=1E7)
DOO2- 96		.BY \$96
DOO3- 80		.BY \$80
DOO4- FF		.BY \$FF ;3rd constant
DOO5- F0		.BY \$F0 ;value= -1,000,000 (=-1E6)
DOO6- B0		.BY \$B0
DOO7- C0		.BY \$C0
DOO8- 00		.BY \$00 ;4th constant

```

D009- 01      .BY $01          ;value= +100,000 (=+1E5)
D00A- 86      .BY $86
D00B- A0      .BY $A0
D00C- FF      .BY $FF          ;5th constant
D00D- FF      .BY $FF          ;value= -10,000 (=-1E4)
D00E- D8      .BY $D8
D00F- F0      .BY $F0
D00G- 00      .BY $00          ;6th constant
D00H- 00      .BY $00          ;value= +1,000 (=+1E3)
D00I- 03      .BY $03
D00J- E8      .BY $E8
D00K- FF      .BY $FF          ;7th constant, MSB
D00L- FF      .BY $FF          ;value= -100
D00M- FF      .BY $FF
D00N- 9C      .BY $9C
D00O- 00      .BY $00          ;8th constant
D00P- 00      .BY $00          ;value= +10
D00Q- 00      .BY $00
D00R- 0A      .BY $0A
D00S- FF      .BY $FF          ;9th constant
D00T- FF      .BY $FF          ;value= -1
D00U- FF      .BY $FF
D00V- FF      .BY $FF
D00W- FF      .BY $FF
;*****
;*
;* Table of six four-byte con-
;* stants used by FLP# to string *
;* conversion routine, to con-
;* vert the jiffy clock to TI$. *
;*
;*****
DOFO- FF      FDCEND .BY $FF          ;1st constant
DOF1- DF      .BY $0F          ;value= -21600000
DOF2- 0A      .BY $0A          ;(number of 60th seconds
DOF3- 80      .BY $80          ;in ten hours)
DOF4- 00      .BY $00          ;2nd constant
DOF5- 03      .BY $03          ;value= +2160000
DOF6- 48      .BY $48          ;(number of 60th seconds
DOF7- C0      .BY $C0          ;in one hour)
DOF8- FF      .BY $FF          ;3rd constant
DOF9- FF      .BY $FF          ;value= +36000
DOFA- 73      .BY $73          ;(number of 60th seconds
DOFB- 60      .BY $60          ;in ten minutes)
DOFC- 00      .BY $00          ;4th constant
DOFD- 00      .BY $00          ;value= +3600
DOFE- 0E      .BY $0E          ;(number of 60th seconds
DOFF- 10      .BY $10          ;in a minute)
D100- FF      .BY $FF          ;5th constant
D101- FF      .BY $FF          ;value= -600
D102- FD      .BY $FD          ;(number of 60th seconds
D103- A8      .BY $A8          ;in ten seconds)
D104- 00      .BY $00          ;6th constant
D105- 00      .BY $00          ;value= +60
D106- 00      .BY $00          ;(number of 60th seconds
D107- 3C      .BY $3C          ;in a second)
TIMEND
;*****
;*
;* Perform SQR. *
;*
;*****
D108- 20 42 CD SQR   JSR MOVAF    ;copy FLP accu1 to FLP accu2
D10B- A9 C7      LDA #L,FHALF  ;point YA
D10D- A0 DD      LDY #H,FHALF  ;to FLP constant .50000000

```

D10F- 20 D8 CC		JSR MOVFM ;load FLP accu1 with it, and: ;***** ;* * ;* Perform power calculation. * ;* (FLPaccu1=FLPaccu2*FLPaccu1). * ;* (Evaluated as: FLP accu1= * ;* EXP(FLPaccu1*LOG(FLPaccu2))). * ;* * ;*****
D112- F0 70	FPWRT	BEQ EXP ;if FLP accu1 is zero reflect 1
D114- A5 66		LDA *ARGEXP ;else get exponent of accu2
D116- D0 03		BNE FPWRT1 ;if zero
D118- 4C 2F CA		JMP ZEROF1 ;reflect zero
D11B- A2 4B	FPWRT1	LOX #L,TEMPF3 ;else point to
D11D- A0 00		LDY #H,TEMPF3 ;extra accu
D11F- 20 0A CD		JSR MOVFM ;move FLP accu1 there
D122- A5 6B		LDA *ARGSGN ;get sign of accu2
D124- 10 0F		BPL FPWR1 ;if negative
D126- 20 02 CE		JSR INT ;perform INT on accu1
D129- A9 4B		LDA #L,TEMPF3 ;point to
D12B- A0 00		LDY #H,TEMPF3 ;extra accu
D12D- 20 91 CD		JSR FCOMP ;compare FLP accu1 with extra accu
D130- D0 03		BNE FPWR1 ;if equal
D132- 98		TYA ;get index in accu in A
D133- A4 03		LDY *INTEGR ;get odd/even flag
D135- 20 34 CD	FPWR1	JSR MOVFA1 ;copy FLP accu2 to accu1
D138- 98		TYA ;get odd/even flag back
D139- 48		PHA ;save it
D13A- 20 20 CB		JSR LOG ;perform LOG
D13D- A9 4B		LDA #L,TEMPF3 ;point to
D13F- A0 00		LDY #H,TEMPF3 ;extra accu
D141- 20 5E CB		JSR FMULT ;multiply FLP accu1 with extra accu
D144- 20 84 01		JSR EXP ;perform EXP
D147- 68		PLA ;get odd/even flag back
D148- 4A		LSR A ;if exponent was even
D149- 90 0A		BCC NEGRTS ;exit, else: ;***** ;* * ;* Perform unary minus. * ;* (Negate FLP accu 1). * ;* * ;*****
D14B- A5 5E	NEGOP	LDA *FACEXP ;get exponent
D14D- F0 06		BEQ NEGRTS ;if zero, do nothing
D14F- A5 63		LDA *FACSGN ;else get sign
D151- 49 FF		EOR #\$FF ;invert it
D153- 85 63		STA *FACSGN ;and store it
D155- 60	NEGRTS	RTS ;and exit ;***** ;* * ;* Floating point constant: * ;* +1.44269504 * ;* (=1/(LOG(e)/LOG2)). * ;* (Used in evaluation of EXP). * ;* * ;*****
D156- 81	LOGEB2	.BY \$81 ;exponent
D157- 38		.BY \$38 ;mantissa MSB and sign
D158- AA		.BY \$AA ;upper middle mantissa
D159- 3B		.BY \$3B ;lower middle mantissa
D15A- 29		.BY \$29 ;LSB mantissa ;***** ;* * ;* Table of constants for EXP. *

```

;* Numbers are:                                *
;* 7 (number of terms-1)                      *
;* +0.0000214987637                          *
;* +0.000143719813                          *
;* +0.00134226348                          *
;* +0.00961401701                          *
;* +0.0555051269                           *
;* +0.240226385                           *
;* +0.693147186                           *
;* +1.000000000                          *
;*                                         *
;*****                                         *****

D15B- 07      EXPCON .BY $07          ;series has eight terms
D15C- 71      .BY $71          ;exponent
D15D- 34      .BY $34          ;mantissa MSB and sign
D15E- 58      .BY $58          ;upper middle mantissa
D15F- 3E      .BY $3E          ;lower middle mantissa
D160- 56      .BY $56          ;LSB mantissa
D161- 74      .BY $74          ;exponent
D162- 16      .BY $16          ;mantissa MSB and sign
D163- 7E      .BY $7E          ;upper middle mantissa
D164- B3      .BY $B3          ;lower middle mantissa
D165- 1B      .BY $1B          ;LSB mantissa
D166- 77      .BY $77          ;exponent
D167- 2F      .BY $2F          ;mantissa MSB and sign
D168- EE      .BY $EE          ;upper middle mantissa
D169- E3      .BY $E3          ;lower middle mantissa
D16A- 85      .BY $85          ;LSB mantissa
D16B- 7A      .BY $7A          ;exponent
D16C- 1D      .BY $10          ;mantissa MSB and sign
D16D- 84      .BY $84          ;upper middle mantissa
D16E- 1C      .BY $1C          ;lower middle mantissa
D16F- 2A      .BY $2A          ;LSB mantissa
D170- 7C      .BY $7C          ;exponent
D171- 63      .BY $63          ;mantissa MSB and sign
D172- 59      .BY $59          ;upper middle mantissa
D173- 58      .BY $58          ;lower middle mantissa
D174- 0A      .BY $0A          ;LSB mantissa
D175- 7E      .BY $7E          ;exponent
D176- 75      .BY $75          ;mantissa MSB and sign
D177- FD      .BY $FD          ;upper middle mantissa
D178- E7      .BY $E7          ;lower middle mantissa
D179- C6      .BY $C6          ;LSB mantissa
D17A- 80      .BY $80          ;exponent
D17B- 31      .BY $31          ;mantissa MSB and sign
D17C- 72      .BY $72          ;upper middle mantissa
D17D- 18      .BY $18          ;lower middle mantissa
D17E- 10      .BY $10          ;LSB mantissa
D17F- 81      .BY $81          ;exponent
D180- 00      .BY $00          ;mantissa MSB and sign
D181- 00      .BY $00          ;upper middle mantissa
D182- 00      .BY $00          ;lower middle mantissa
D183- 00      .BY $00          ;LSB mantissa
.FI "BASIC4.0D.M02"

```

1870 341F-4F8F BASIC4.0D.M02

```

;name BASIC4.0D.M02
;*****                                         *****
;*                                         *
;* Perform EXP.                            *
;*                                         *
;*****                                         *****

```

```

D184- A9 56      EXP     LDA #L,LOGEB2 ;point YA
D186- A0 D1      LDY #H,LOGEB2 ;to FLP constant
D188- 20 5E CB   JSR FMULT ;multiply FLP accu with it
D18B- A5 60      LDA *FACOV ;get rounding byte
D18D- 69 50      ADC #$50 ;add eighty
D18F- 90 03      BCC STOLD ;if carry now
D191- 20 59 CD   JSR INCRND ;round FLP accu1
D194- 85 53      STOLD    STA *OLDOV ;and store
D196- 20 45 CD   JSR MOVEF ;copy FLP accu1 to accu2 (no rounding)
D199- A5 5E      LDA *FACEXP ;get exponent of accu1
D19B- C9 88      CMP #$88 ;if argument >256 or <-256
D19D- 90 03      BCC EXP1
D19F- 20 0A CC   GOMDLV  JSR MLDVEX ;zero accu1 or give OVERFLOW error
D1A2- 20 02 CE   EXP1    JSR INT ;perform INT
D1A5- A5 03      LDA *INTEGR ;get odd/even flag
D1A7- 18         CLC      ;prepare for add
D1A8- 69 81      ADC #$81 ;add 129
D1AA- F0 F3      BEQ GOMDLV ;if now zero, OVERFLOW ERROR
D1AC- 38         SEC      ;prepare for subtract
D1AD- E9 01      SBC #$01 ;get correct exponent
D1AF- 48         PHA      ;save it
D1B0- A2 05      LDX #5   ;move six bytes
D1B2- B5 66      SWAPLP  LDA *ARGEEXP,X ;get byte from accu2
D1B4- B4 5E      LDY *FACEXP,X ;and accu1
D1B6- 95 5E      STA *FACEXP,X ;and exchange
D1B8- 94 66      STY *ARGEEXP,X ;accu1 and accu2
D1BA- CA         DEX      ;move bytes
D1BB- 10 F5      BPL SWAPLP ;until whole accu done
D1BD- A5 53      LDA *OLDOV ;get saved rounding value
D1BF- 85 60      STA *FACOV ;in rounding byte
D1C1- 20 89 C9   JSR FSUBT ;perform subtraction
D1C4- 20 4B D1   JSR NEGOP ;negate accu1
D1C7- A9 5B      LDA #L,EXPCON ;point YA
D1C9- A0 D1      LDY #H,EXPCON ;to series constants
D1CB- 20 ED D1   JSR POLY  ;evaluate polynome
D1CE- A9 00      LDA #0   ;clear
D1DD- 85 6C      STA *ARISGN ;sign comparison
D1D2- 68         PLA      ;get odd/even flag back
D1D3- 20 EF CB   JSR MLDEXP ;add exponents
D1D6- 60         RTS      ;and exit
;*****
;*
;* Evaluate polynome. *
;*
;*****
D1D7- 85 6E      POLYX   STA *POLYPT ;store pointer
D1D9- 84 6F      STY *POLYPT+1 ;in ($6E)
D1DB- 20 00 CD   JSR MOV1F ;store FLP accu1 in memory
D1DE- A9 54      LDA #L,TEMPF1 ;point YA to $54 (save area for SIN, CO
D1EO- 20 5E CB   JSR FMULT ;multiply accu1 with that number
D1E3- 20 F1 D1   JSR POLY1 ;evaluate polynome
D1E6- A9 54      LDA #L,TEMPF1 ;point to
D1E8- A0 00      LDY #H,TEMPF1 ;result
D1EA- 4C 5E CB   JMP FMULT ;and multiply accu with that number
;*****
;*
;* Evaluate polynome. *
;* On entry, YA must point to *
;* the number of terms of the *
;* polynome, followed by the *
;* required number of constants *
;* in FLP format. *
;*
;*****

```

```

D1ED- 85 6E      POLY    STA *POLYPT      ;store pointer
D1EF- 84 6F      STY *POLYPT+1   ;to constants in ($6E)
D1F1- 20 FD CC  POLY1   JSR MOV2F      ;store FLP accu1 in memory (at $59)
D1F4- B1 6E      LDA (POLYPT),Y   ;get number of terms
D1F6- 85 64      STA *DEGREE     ;in $64
D1F8- A4 6E      LDY *POLYPT     ;get pointer lo
D1FA- C8         INY          ;point to 1st constant
D1FB- 98         TYA          ;get lo in A
D1FC- D0 02      BNE POLY3     ;if page crossed
D1FE- E6 6F      INC *POLYPT+1  ;adapt hi byte
D200- 85 6E      POLY3   STA *POLYPT     ;and store pointer
D202- A4 6F      LDY *POLYPT+1  ;get it in YA
D204- 20 5E CB  POLY2   JSR FMULT     ;multiply accu1 with saved value
D207- A5 6E      LDA *POLYPT     ;get current
D209- A4 6F      LDY *POLYPT+1  ;pointer to constant in YA
D20B- 18         CLC          ;prepare for add
D20C- 69 05      ADC #5        ;point to next constant
D20E- 90 01      BCC POLY4     ;if page crossed
D210- C8         INY          ;adapt hi byte
D211- 85 6E      POLY4   STA *POLYPT     ;store new
D213- 84 6F      STY *POLYPT+1  ;pointer
D215- 20 9D C9  JSR FADD      ;perform addition (add next constant)
D218- A9 59      LDA #L,TEMPF2   ;point YA
D21A- A0 00      LDY #H,TEMPF2   ;to $59
D21C- C6 64      DEC *DEGREE    ;count terms
D21E- D0 E4      BNE POLY2     ;if not all done, loop
D220- 60         RTS          ;else exit
;*****
;*
;* Floating point constant: *
;* +11879546.4   *
;* (Used by RND(1)).   *
;* BUG: LSB mantissa missing,   *
;*       the exponent of the next   *
;*       constant is used for   *
;*       this.   *
;*
;*****
D221- 98         RMULC   .BY $98        ;MSB and sign
D222- 35         .BY $35        ;upper middle
D223- 44         .BY $44        ;lower middle
D224- 7A         .BY $7A        ;LSB
;*****
;*
;* Floating point constant: *
;* +0.0000000392767778   *
;* (Used by RND(1)).   *
;* BUG: LSB mantissa missing,   *
;*       the opcode of the 1st   *
;*       instruction of RND is   *
;*       used for this.   *
;*
;*****
D225- 68         RADDCC  .BY $68        ;MSB and sign (LSB of previous)
D226- 28         .BY $28        ;upper middle
D227- B1         .BY $B1        ;lower middle
D228- 46         .BY $46        ;LSB
;*****
;*
;* Perform RND.   *
;*
;*****
D229- 20 61 CD  RND    JSR SIGN      ;get sign of FLP accu1
D22C- 30 2E      BMI RND1     ;if negative, interchange bytes

```

```

D22E- D0 17      BNE QSETNR ;if positive, use constants
D230- AD 44 E8  LDA CHTIM ;if zero argument, get timer1 lo
D233- 85 5F      STA *FACHO ;as MSB and sign
D235- AD 48 E8  LDA CHTIM+4 ;get timer2 lo
D238- 85 60      STA *FACMOH ;as upper middle
D23A- AD 45 E8  LDA CHTIM+1 ;get timer1 hi
D23D- 85 61      STA *FACMO ;as lower middle
D23F- AD 49 E8  LDA CHTIM+5 ;get timer2 hi
D242- 85 62      STA *FACLO ;as LSB
D244- 4C 6C D2   JMP STRNEX ;and set 0<result<1
D247- A9 88      QSETNR LDA #L,RNDX ;point YA
D249- A0 00      LDY #H,RNDX ;to last seed
D24B- 20 D8 CC   JSR MOVFM ;load FLP accu1 with it
D24E- A9 21      LDA #L,RMULC ;point YA to
D250- AD 02      LDY #H,RMULC ;constant to be multiplied
D252- 20 5E CB   JSR FMULT ;multiply accu1 with it
D255- A9 25      LDA #L,RADDc ;point YA to
D257- AD 02      LDY #H,RADDc ;constant to be added
D259- 20 9D C9   JSR FADD ;add to FLP accu1
D25C- A6 62      RND1  LDX *FACLO ;get LSB
D25E- A5 5F      LDA *FACHO ;and MSB
D260- 85 62      STA *FACLO ;exchange
D262- 86 5F      STX *FACHO ;them
D264- A6 60      LDX *FACMOH ;get upper middle
D266- A5 61      LDA *FACMO ;and lower middle
D268- 85 60      STA *FACMOH ;exchange
D26A- 86 61      STX *FACMO ;them too
D26C- A9 00      STRNEX LDA #$00 ;get sign
D26E- 85 63      STA *FACSGN ;and set accu positive
D270- A5 5E      LDA *FACEXP ;get exponent
D272- 85 60      STA *FACOV ;in rounding byte
D274- A9 80      LDA #$80 ;get new exponent
D276- 85 5E      STA *FACEXP ;(result between 0 and 1)
D278- 20 0D CA   JSR NORMAL ;normalize accu1
D27B- A2 88      LDX #L,RNDX ;point XY
D27D- A0 00      LDY #H,RNDX ;to seed area
D27F- 4C DA CD   GOMYMF JMP MOVFM ;and store FLP accu1 there
;*****
;*
;* Perform COS. *
;*
;*****
D282- A9 FE      COS    LDA #L,PI2 ;point YA
D284- A0 D2      LDY #H,PI2 ;to FLP constant PI/2
D286- 20 9D C9   JSR FADD ;add that number to argument, then:
;*****
;*
;* Perform SIN. *
;*
;*****
D289- 20 42 CD   SIN    JSR MOVAF ;copy accu1 to accu2
D28C- A9 03      LDA #L,TWOPi ;point YA to
D28E- A0 D3      LDY #H,TWOPi ;constant 2*PI
D290- A6 6B      LDX *ARGSGN ;get exponent of accu2
D292- 20 3D CC   JSR FDIVF ;load accu1 with constant and divide
D295- 20 42 CD   JSR MOVAF ;copy accu1 to accu2
D298- 20 02 CE   JSR INT  ;perform INT
D29B- A9 00      LDA #$00 ;clear
D29D- 85 6C      STA *ARISGN ;sign comparison
D29F- 20 89 C9   JSR FSUBT ;perform subtraction
D2A2- A9 08      LDA #L,FR4 ;point YA to
D2A4- A0 D3      LDY #H,FR4 ;FLP constant .25
D2A6- 20 86 C9   JSR FSUB  ;perform subtraction
D2A9- A5 63      LDA *FACSGN ;get sign of accu1

```

```

D2AB- 48          PHA           ;save it
D2AC- 10 00        BPL SIN1      ;if accu1 negative
D2AE- 20 7F C9    JSR FADDH     ;add .5 to it
D2B1- A5 63        LDA *FACSGN   ;get sign again
D2B3- 30 09        BMI SIN2      ;if now negative
D2B5- A5 0C        LDA *TANSGN   ;get 3rd/4th quadrant flag
D2B7- 49 FF        EOR #$FF      ;invert it
D2B9- 85 0C        STA *TANSGN   ;and store it
D2B8- 20 48 D1    SIM1         JSR NEGOP     ;negate accu1
D2BE- A9 08        SIN2         LDA #L,FR4    ;point YA to
D2C0- A0 03        LDY #H,FR4    ;FLP constant .25
D2C2- 20 90 C9    JSR FADD     ;perform addition
D2C5- 68          PLA           ;get original sign back
D2C6- 10 03        BPL SIN3      ;if accu was negative
D2C8- 20 48 D1    JSR NEGOP     ;negate it
D2CB- A9 00        SIN3         LDA #L,SINCON ;point YA to
D2CD- A0 03        LDY #H,SINCON ;series constants
D2CF- 4C 07 D1    JMP POLYX     ;and evaluate polynome, then exit
.FI "BASIC4.00.M03"

```

17A7 341F-4BC6 BASIC4.00.M03

```

;name BASIC4.00.M03
;*****
;*                               *
;* Perform TAN.                 *
;*                               *
;*****                         *
D2D2- 20 00 CD  TAN   JSR MOV1F      ;save FLP accu in memory at $54
D2D5- A9 00          LDA #$00      ;clear
D2D7- 85 0C          STA *TANSGN   ;3rd/4th quadrant flag
D2D9- 20 89 D2      JSR SIN       ;perform SIN
D2DC- A2 48          LDX #L,TEMPF3  ;point XY to
D2DE- A0 00          LDY #H,TEMPF3  ;extra accu
D2E0- 20 7F D2      JSR GOMVMF    ;move FLP accu1 there
D2E3- A9 54          LDA #L,TEMPF1  ;point YA to
D2E5- A0 00          LDY #H,TEMPF1  ;saved FLP accu1
D2E7- 20 D8 CC      JSR MOVFM     ;load FLP accu1 again
D2EA- A9 00          LDA #$00      ;get sign
D2EC- 85 63          STA *FACSGN   ;and set accu1 positive
D2EE- A5 0C          LDA *TANSGN   ;get 3rd/4th quadrant flag
D2F0- 20 FA D2      JSR COSC      ;push it and perform part of SIN
D2F3- A9 48          LDA #L,TEMPF3  ;point YA to
D2F5- A0 00          LDY #H,TEMPF3  ;extra accu
D2F7- 4C 45 CC      JMP FDIV      ;and perform division, then exit
D2FA- 48             COSC          PHA           ;save
D2FB- 4C BB D2      JMP SIN1      ;and perform part of SIN
;*****
;*                               *
;* Floating point constant:   *
;* +1.57079633               *
;* (PI/2, used by SIN).      *
;*                               *
;*****                         *
D2FE- 81             PI2           .BY $81       ;exponent
D2FF- 49             .BY $49       ;MSB and sign
D300- 0F             .BY $0F       ;upper middle
D301- DA             .BY $DA       ;lower middle
D302- A2             .BY $A2       ;LSB
;*****
;*                               *
;* Floating point constant:   *
;* +6.28318531               *
;*                               *

```

```

        ;* (2*PI, used by SIN).          *
        ;*
        ;* (Why not use same constant in *
        ;* polynome table of SIN ? Saves *
        ;* five bytes !).              *
        ;*
        ;*****  

D303- 83      TWOPI   .BY $83           ;exponent
D304- 49      .BY $49           ;MSB and sign
D305- 0F      .BY $0F           ;upper middle
D306- DA      .BY $DA           ;lower middle
D307- A2      .BY $A2           ;LSB
        ;*****  

        ;*
        ;* Floating point constant:    *
        ;* +0.25000000000             *
        ;* (Used by SIN).            *
        ;*
        ;*****  

D308- 7F      FR4     .BY $7F           ;exponent
D309- 00      .BY $00           ;MSB and sign
D30A- 00      .BY $00           ;upper middle
D30B- 00      .BY $00           ;lower middle
D30C- 00      .BY $00           ;LSB
        ;*****  

        ;*
        ;* Table of constants for SIN. *
        ;* Numbers are:                *
        ;* 5 (number of terms-1)       *
        ;* -14.3813907                *
        ;* +42.0077971                *
        ;* -76.7041703                *
        ;* +81.6052237                *
        ;* -41.3417021                *
        ;* +6.28318532 (=2*PI)       *
        ;*
        ;*****  

D30D- 05      SINCON  .BY $05           ;series has six terms
D30E- 84      .BY $84           ;exponent
D30F- E6      .BY $E6           ;MSB and sign
D310- 1A      .BY $1A           ;upper middle
D311- 2D      .BY $2D           ;lower middle
D312- 1B      .BY $1B           ;LSB
D313- 86      .BY $86           ;exponent
D314- 28      .BY $28           ;MSB and sign
D315- 07      .BY $07           ;upper middle
D316- FB      .BY $FB           ;lower middle
D317- F8      .BY $F8           ;LSB
D318- 87      .BY $87           ;exponent
D319- 99      .BY $99           ;MSB and sign
D31A- 68      .BY $68           ;upper middle
D31B- 89      .BY $89           ;lower middle
D31C- 01      .BY $01           ;LSB
D31D- 87      .BY $87           ;exponent
D31E- 23      .BY $23           ;MSB and sign
D31F- 35      .BY $35           ;upper middle
D320- DF      .BY $DF           ;lower middle
D321- E1      .BY $E1           ;LSB
D322- 86      .BY $86           ;exponent
D323- A5      .BY $A5           ;MSB and sign
D324- 5D      .BY $5D           ;upper middle
D325- E7      .BY $E7           ;lower middle
D326- 28      .BY $28           ;LSB
D327- 83      d327    .BY $83           ;exponent

```

```

D328- 49      .BY $49          ;MSB and sign
D329- 0F      .BY $0F          ;upper middle
D32A- DA      .BY $DA          ;lower middle
D32B- A2      .BY $A2          ;LSB
;*****
;*                               *
;* Perform ATN.               *
;*                               *
;*****
D32C- A5 63    ATN      LDA *FACSGN   ;get sign of accu1
D32E- 48        PHA          ;save it
D32F- 10 03    BPL ATN1     ;if negative
D331- 20 4B D1  JSR NEGOP     ;negate accu1
D334- A5 5E    ATN1      LDA *FACEXP   ;get exponent
D336- 48        PHA          ;save it too
D337- C9 81    CMP #$81
D339- 90 07    BCC ATN2
D33B- A9 F2    LDA #L,FONE   ;point YA to
D330- A0 CA    LDY #H,FONE   ;FLP constant 1.0000000
D33F- 20 45 CC  JSR FDIV     ;perform division
D342- A9 5C    ATN2      LDA #L,ATNCON ;point YA to
D344- A0 D3    LDY #H,ATNCON ;series constants
D346- 20 D7 D1  JSR POLYX    ;evaluate polynome
D349- 68        PLA          ;get exponent back
D34A- C9 81    CMP #$81
D34C- 90 07    BCC ATN3
D34E- A9 FE    LDA #L,PI2    ;point YA to
D350- A0 D2    LDY #H,PI2    ;FLP constant PI/2
D352- 20 86 C9  JSR FSUB     ;perform subtraction
D355- 68        ATN3      PLA          ;get sign back
D356- 10 03    BPL ATN4     ;if negative
D358- 4C 4B D1  JMP NEGOP     ;negate FLP accu1
D358- 60        ATN4      RTS          ;else exit
;*****
;*                               *
;* Table of constants for ATN. *
;* Numbers are:                 *
;* 11 (number of terms-1)       *
;* -0.0000855992390           *
;* +0.00485094216             *
;* -0.0161117018              *
;* +0.0342096380              *
;* -0.0542791328              *
;* +0.0724457196              *
;* -0.0898023954              *
;* +0.110932413               *
;* -0.142839808               *
;* +0.199999120               *
;* -0.333333316               *
;* +1.0000000000               *
;*                               *
;*****
D35C- 0B        ATNCON   .BY $0B          ;series has twelve terms
D35D- 76        .BY $76          ;exponent
D35E- B3        .BY $B3          ;MSB and sign
D35F- 83        .BY $83          ;upper middle
D360- B0        .BY $B0          ;lower middle
D361- D3        .BY $D3          ;LSB
D362- 79        .BY $79          ;exponent
D363- 1E        .BY $1E          ;MSB and sign
D364- F4        .BY $F4          ;upper middle
D365- A6        .BY $A6          ;lower middle
D366- F5        .BY $F5          ;LSB
D367- 7B        .BY $7B          ;exponent

```

```

D368- 83      .BY $83      ;MSB and sign
D369- FC      .BY $FC      ;upper middle
D36A- B0      .BY $B0      ;lower middle
D36B- 10      .BY $10      ;LSB
D36C- 7C      .BY $7C      ;exponent
D36D- 0C      .BY $0C      ;MSB and sign
D36E- 1F      .BY $1F      ;upper middle
D36F- 67      .BY $67      ;lower middle
D370- CA      .BY $CA      ;LSB
D371- 7C      .BY $7C      ;exponent
D372- DE      .BY $DE      ;MSB and sign
D373- 53      .BY $53      ;upper middle
D374- CB      .BY $CB      ;lower middle
D375- C1      .BY $C1      ;LSB
D376- 7D      .BY $7D      ;exponent
D377- 14      .BY $14      ;MSB and sign
D378- 64      .BY $64      ;upper middle
D379- 70      .BY $70      ;lower middle
D37A- 4C      .BY $4C      ;LSB
D37B- 7D      .BY $7D      ;exponent
D37C- B7      .BY $B7      ;MSB and sign
D37D- EA      .BY $EA      ;upper middle
D37E- 51      .BY $51      ;lower middle
D37F- 7A      .BY $7A      ;LSB
D380- 7D      .BY $7D      ;exponent
D381- 63      .BY $63      ;MSB and sign
D382- 30      .BY $30      ;upper middle
D383- 88      .BY $88      ;lower middle
D384- 7E      .BY $7E      ;LSB
D385- 7E      .BY $7E      ;exponent
D386- 92      .BY $92      ;MSB and sign
D387- 44      .BY $44      ;upper middle
D388- 99      .BY $99      ;lowe middle
D389- 3A      .BY $3A      ;LSB
D38A- 7E      .BY $7E      ;exponent
D38B- 4C      .BY $4C      ;MSB and sign
D38C- CC      .BY $CC      ;upper middle
D38D- 91      .BY $91      ;lower middle
D38E- C7      .BY $C7      ;LSB
D38F- 7F      .BY $7F      ;exponent
D390- AA      .BY $AA      ;MSB and sign
D391- AA      .BY $AA      ;upper middle
D392- AA      .BY $AA      ;lower middle
D393- 13      .BY $13      ;LSB
D394- 81      .BY $81      ;exponent
D395- 00      .BY $00      ;MSB and sign
D396- 00      .BY $00      ;upper middle
D397- 00      .BY $00      ;lower middle
D398- 00      .BY $00      ;LSB
.FI "BASIC4.00.M04"

```

1987 341F-4DD6 BASIC4.00.M04

```

;name BASIC4.00.M04
;*****
;*                                *
;* BASIC's CHRGET routine.      *
;* The coldstart routine will   *
;* move it to RAM in the zero-  *
;* page, starting at $0070.     *
;* The routine fetches the next  *
;* character from the program,  *
;* or from the input buffer,    *

```

```

;* setting the flags as follows: *
;* Z : set if a colon or a zero *
;*      byte has been found. (In *
;*      general: an end of state- *
;*      ment byte). *
;* C      clear if a numeric cha- *
;*      racter was input, set in *
;*      other cases. *
;* N      set, if the MSB is set in *
;*      the byte that was input. *
;*      (Indicates shifted cha- *
;*      racters). *
;* V      unaffected. *
;*
;* The routine skips spaces. *
;*
;* Two other entry points are *
;* also used: *
;*
;* CHRGOT ($0076): gets the *
;*      current character back *
;*      and sets the flags as *
;*      CHRGET would do. *
;* QNUM   ($007D): clears carry *
;*      if A holds a number. *
;*      Sets carry in other *
;*      cases. *
;*
;*****  

D399- E6 77 INITAT INC *TXTPTTR      ;$70 - increment current pointer to
D39B- D0 02             BNE CHDGOT      ;$72 - if page crossed
D39D- E6 78             INC *TXTPTTR+1    ;$74 - increment hi also
D39F- AD 60 EA CHDGOT LDA $EA60      ;$76 - get character
D3A2- C9 3A             CMP #$3A      ;$79 - if colon or higher
D3A4- B0 0A             BCS CHDRTS      ;$7B - exit with C set (and Z with colo
D3A6- C9 20             qnum    CMP #$20      ;$7D - if space
D3A8- F0 EF             BEQ INITAT      ;$7F - skip it, get next character
D3AA- 38               SEC      ;$81 - else prepare for subtract
D3AB- E9 30             SBC #'0'       ;$82 - subtract value for ASCII zero
D3AD- 38               SEC      ;$84 - prepare for subtract
D3AE- E9 D0             SBC #$D0      ;$85 - clear C if original was 0 - 9 (Z
D3B0- 60               CHDRTS RTS        ;$87 - exit
;*****  

;* Initial RND-number seed. *
;* Value = +0.811635157 *
;* The coldstart routine will *
;* move it to RAM in the zero- *
;* page, starting at $0088, *
;* together with the CHRGET *
;* routine. *
;*
;*****  

D3B1- 80 .BY $80          ;$88 - exponent
D3B2- 4F .BY $4F          ;$89 MSB mantissa and sign
D3B3- C7 .BY $C7          ;$8A - upper middle mantissa
D3B4- 52 .BY $52          ;$8B - lower middle mantissa
D3B5- 58 .BY $58          ;$8C - LSB mantissa
;*****  

;* BASIC coldstart. *
;* Entered from the reset rou- *
;* tine at $FD16 if diagnostic *
;* sense was high during the *

```

```

;* RESET of the machine.      *
;*                           *
;*****                         *
D3B6- A2 FB    INIT   LDX #L,STKEND ;get initial stackpointer
D3B8- 9A        TXS     ;in stackpointer
D3B9- A9 4C    LDA #$4C ;get opcode for JMP
D3B8- 85 51    STA *JMPER ;set up for functions
D3B0- 85 00    STA *USRPOK ;and for USR
D3BF- A9 73    LDA #L,FCERR ;set initial
D3C1- A0 C3    LDY #H,FCERR ;USR routine to
D3C3- 85 01    STA *USRPOK+1 ;?ILLEGAL QUANTITY ERROR
D3C5- 84 02    STY *USRPOK+2
D3C7- A2 1C    LDX #$1C ;move 28 bytes
D3C9- B0 98 D3  MOVCHG LDA INITAT-1,X ;get byte from CHRGET or RND seed
D3CC- 95 6F    STA *CHRGET-1,X ;move it to zero page
D3CE- CA        DEX     ;until all bytes
D3CF- D0 F8    BNE MOVCHG ;moved, and X=0
D3D1- A9 03    LDA #$03 ;set garbage collect stepsize
D3D3- 85 50    STA *FOUR6 ;(!?!?, not needed with BASIC 4.0 !)
D3D5- 8A        TXA     ;get a zero byte
D3D6- 85 65    STA *BITS ;clear shift in for FLP accu1
D3D8- 85 10    STA *CHANNL ;set default CMD output file#
D3DA- 85 15    STA *LASTPT+1 ;set hi byte of pointer in descriptor s
D3DC- 85 00    STA *DSDESC ;set length of DS$ to zero
D3DE- 48        PHA     ;save a zero (why?)
D3DF- E8        INX     ;X=1
D3E0- 8E FD 01  STX BUF-3 ;set up a dummy line link pointer
D3E3- 8E FC 01  STX BUF-4 ;for the input buffer
D3E6- A2 16    LDX #L,TEMPST ;point to descriptor stack, top
D3E8- 86 13    STX *TEMPPT ;and set descriptor stack pointer
D3EA- A0 04    LDY #H,RAMLOC ;point YA to start of program RAM
D3EC- 85 28    STA *TXTTAB ;set initial pointer
D3EE- 84 29    STY *TXTTAB+1 ;to start of program
D3F0- 85 11    STA *LINNUM ;and pointer for
D3F2- 84 12    STY *LINNUM+1 ;RAM test
D3F4- A8        TAY     ;get a zero offset
D3F5- A9 80    LDA #$80 ;skip joke
D3F7- D0 07    BNE LOOPMN ;and do RAM test
;*****                         *
;*                           *
;* A joke ??                *
;* This entrypoint will print *
;* the usual power up message *
;* with 44031 BYTES FREE.    *
;*                           *
;*****                         *
D3F9- A9 00    LDA #L,ROMLOC ;point YA to $8000
D3FB- A0 B0    LDY #H,ROMLOC ;(start of BASIC ROM space)
D3FD- 4C 1B D4  LOOPMN  JMP USEDEF ;print messages and start BASIC
D400- E6 11    INC *LINNUM ;adapt test pointer lo
D402- D0 04    BNE LOOPM1 ;if page crossed
D404- E6 12    INC *LINNUM+1 ;adapt test pointer hi
D406- 30 0F    BMI USEDEC ;if at start of screen, stop
D408- A9 55    L0OPM1 LDA #$55 ;else get 01010101 pattern
D40A- 91 11    STA (LINNUM),Y ;move to RAM
D40C- D1 11    CMP (LINNUM),Y ;and read it back
D40E- D0 07    BNE USEDEC ;if not equal, stop
D410- DA        ASL A ;else get 10101010 pattern
D411- 91 11    STA (LINNUM),Y ;move to RAM
D413- D1 11    CMP (LINNUM),Y ;and read it back
D415- F0 E9    BEQ LOOPMN ;if equal, test next location
D417- A5 11    USEDEC LDA *LINNUM ;else get
D419- A4 12    LDY *LINNUM+1 ;pointer to RAM end
D41B- 85 34    USEDEF STA *MEMSIZ ;in top of RAM

```

```

D410- 84 35 STY *MEMSIZ+1 ;pointer
D41F- 85 30 STA *FRETOP ;and in
D421- 84 31 STY *FRETOP+1 ;start of strings pointer
D423- A0 00 LDY #$00 ;get offset zero
D425- 98 TYA ;and end of line marker
D426- 91 28 STA (TXTTAB),Y ;set 1st end of line marker
D428- E6 28 INC *TXTTAB ;set start of program pointer
D42A- A9 A4 LDA #L,PATCH2 ;point YA to
D42C- A0 DE LDY #H,PATCH2 ;(new) start up message
D42E- 20 1D BB JSR STROUT ;print it on screen
D431- A5 34 LDA *MEMSIZ ;get top of RAM lo
D433- 38 SEC ;prepare for subtract
D434- E5 28 SBC *TXTTAB ;calculate free space lo
D436- AA TAX ;in X
D437- A5 35 LDA *MEMSIZ+1 ;get top of RAM hi
D439- E5 29 SBC *TXTTAB+1 ;and get free space hi in AX
D43B- 20 83 CF JSR LINPRT ;print the size in decimal
D43E- A9 4B LDA #L,WORDS ;point YA to
D440- A0 04 LDY #H,WORDS ;' BYTES FREE' message
D442- 20 1D BB JSR STROUT ;print it
D445- 20 D4 B5 JSR SCRTCH ;perform NEW (set other pointers)
D448- 4C FF B3 JMP READY ;and go do BASIC (warm start)
;*****
;* *
;* Message: BYTES FREE *
;* *
;*****
D44B- 20 42 59 WORDS .BY ' BYTES FREE' $00 $00
D44E- 54 45 53
D451- 20 46 52
D454- 45 45 00
D457- 00
;*****
;* *
;* Message: *
;* *
;*     ### COMMODORE BASIC ### *
;* *
;* (Not used, carried over from .* *
;* BASIC 2.0. Instead the new *
;* power up message at $DEA4 is *
;* used). *
;* *
;*****
D458- 23 23 23 FREMES .BY '### COMMODORE BASIC ###' $00 $00 $00
D45B- 20 43 4F
D45E- 40 40 4F
D461- 44 4F 52
D464- 45 20 42
D467- 41 53 49
D46A- 43 20 23
D46D- 23 23 00
D470- 00 00
LASTWR ;last byte of BASIC system code plus one
.FI "BASIC4.00.MOS"

```

19FA 341F-4E19 BASIC4.00.MOS

```

;name BASIC4.00.MOS
;*****
;* *
;* Machine Language Monitor. *
;* (Shorthand: MLM). *

```

```

        ;* This entry is the call entry, *
        ;* entered on a reset with diag- *
        ;* nistic sense low.           *
        ;* Also meant as standard entry, *
        ;* but rarely used in this    *
        ;* fashion. (SYS4 is easier,    *
        ;* though not correct).      *
        ;*
        ;*****CALLE*****CALLE*****
D472- A9 43      CALLE   LDA #'C'          ;call entry, get a C
D474- 85 B5      STA *TMPC         ;save it
D476- D0 19      BNE B3          ;and go retrieve PC
        ;*****BRKE*****BRKE*****
D478- 20 A6 F2      BRKE    JSR CLRCHN     ;restore normal I/O (prints to screen)
D47B- A9 42      LDA #'B'          ;get a B for the title
D47D- 85 B5      STA *TMPC         ;save it
D47F- D8          CLD             ;clear possible decimal mode
D480- 4A          LSR A           ;clear carry (why not CLC?)
D481- 68          PLA            ;retrieve Y
D482- 80 05 02    STA YR           ;save in input buffer
D485- 68          PLA            ;retrieve X
D486- 80 04 02    STA XR           ;save in input buffer
D489- 68          PLA            ;retrieve accumulator
D48A- 80 03 02    STA ACC          ;save in input buffer
D48D- 68          PLA            ;retrieve statusregister
D48E- 80 02 02    STA FLGS         ;save in input buffer
D491- .68          83              PLA            ;get pc lo
D492- 69 FF      ADC #$FF         ;decrement it
D494- 80 01 02    STA PCL           ;and store in input buffer
D497- 68          PLA            ;get pc hi
D498- 69 FF      ADC #$FF         ;decrement if page crossed
D49A- 80 00 02    STA PCH           ;and store also
D49D- A5 90      LDA *CINV         ;get IRQ vector lo
D49F- 80 08 02    STA INVL          ;and save in input buffer
D4A2- A5 91      LDA *CINV+1       ;get hi also
D4A4- 80 07 02    STA INVH          ;in input buffer
D4A7- BA          TSX             ;get stackpointer
D4A8- 8E 06 02    STX SP           ;and save it
D4AB- 58          CLI             ;enable interrupts again
D4AC- 20 34 05    JSR CRLF          ;start on new line
D4AF- A6 B5      LDX *TMPC         ;get entry character
D4B1- A9 2A      LDA #'*'          ;and an asterisk
D4B3- 20 31 D7    JSR WRTWO        ;print A and X
D4B6- A9 52      LDA #'R'           ;get command (register display)
D4B8- D0 1A      BNE S0           ;and force register display
D4BA- A9 02      LDA #$02          ;set CHRGET
D4BC- 85 77      STA *TXTPTR        ;to direct mode
        ;BUG: must be TXTPTR+1
D4BE- A9 00      LDA #$00          ;clear
D4C0- 85 DE      STA *WRAP          ;wrap-around byte
D4C2- A2 00      LDX #$0D          ;get a carriage return
D4C4- A9 2E      LDA #'.'          ;and a prompt (period)

```

```

D4C6- 20 31 07      JSR WRTWO      ;print a prompt on a new line
D4C9- 20 98 D7      JSR RDOC       ;get a character
D4CC- C9 2E          CMP #'.'       ;if it is a prompt
D4CE- F0 F9          BEQ ST1        ;ignore, get next character
D4D0- C9 20          CMP #$20      ;if space,
D4D2- F0 F5          BEQ ST1        ;get next character
D4D4- A2 07          S0             LDX #7         ;get # of commands
D4D6- DD 44 05      S1             CMP CMDS,X   ;if command same as in table
D4D9- D0 0B          BNE S2        BNE S2        ;skip if not
D4DB- 86 B4          STX *SAVX     ;save # of cmdnd found
D4DD- BD 4C 05      LDA ADRH,X   ;get hi byte of command address
D4E0- 48             PHA            ;on stack
D4E1- BD 54 05      LDA ADRL,X   ;get lo also
D4E4- 48             PHA            ;complete dummy return address
D4E5- 60             RTS            ;and execute command
D4E6- CA             S2             DEX            ;else point to previous command
D4E7- 10 ED          BPL S1        ;and try for next command
D4E9- 6C FA 03      JMP (USRCMD) ;else check for added commands
; (default is ERRORPR)
;*****
;*
;* Subroutine PPUTP.
;*
;* Moves TMPO to PC save area.
;*
;*****
D4EC- A5 FB          PPUTP         LDA *TMPO      ;get lo byte
D4EE- 80 01 02          STA PCL        ;in PC lo
D4F1- A5 FC          LDA *TMPO+1    ;get hi also
D4F3- 80 00 02          STA PCH        ;complete PC
D4F6- 60             RTS            ;and exit
;*****
;*
;* Subroutine DM.
;*
;* On entry, A holds # of bytes
;* to be displayed, TMPO points
;* to the first address to be
;* displayed.
;*
;*****
D4F7- 85 B5          DM             STA *TMPC     ;save # of bytes to display
D4F9- A0 00          LDY #$00      ;get offset zero
D4FB- 20 31 05      DM1            JSR SPACE      ;print a space
D4FE- B1 FB          LDA (TMPO),Y  ;get byte from memory
D500- 20 22 D7          JSR WROB      ;print as hex
D503- 20 39 05          JSR INCTMP    ;increment address
D506- C6 B5          DEC *TMPC     ;decrement byte counter
D508- D0 F1          BNE DM1      ;if not all done, loop
D50A- 60             RTS            ;else exit
;*****
;*
;* Subroutine BYTE.
;*
;* Read a byte from screen or
;* keyboard and store it. If not
;* stored correctly, report an
;* error.
;*
;*****
D50B- 20 63 07      BYTE           JSR RDOB      ;read a hex byte
D50E- 90 00          BY3            BCC BY3       ;if a valid byte read
D510- A2 00          LDX #$00      ;get offset zero
D512- 81 FB          STA (TMPO,X) ;store the byte

```

```

D514- C1 FB      CMP (TMPO,X) ;and verify it
D516- F0 05      BEQ BY3    ;if not equal
D518- 68          PLA     ;pull
D519- 68          PLA     ;return address
D51A- 4C A4 07      JMP ERROPR ;and give error
D51D- 20 39 05  BY3  JSR INCTMP ;else increment address
D520- C6 B5      DEC *TMPC  ;and adapt byte counter
D522- 60          RTS     ;if all done, exit
;*****  

;*  

;* Subroutine SETR.  

;*  

;* Prepare for register display.  

;*  

;*****  

D523- A9 02      SETR    LDA #L,FLGS ;point
D525- 85 FB      STA *TMPO   ;TMPO
D527- A9 02      LDA #H,FLGS ;to the start
D529- 85 FC      STA *TMPO+1 ;of the 1-byte registers
D52B- A9 05      LDA #5    ;set to display 5 registers
D52D- 60          RTS     ;and exit
;*****  

;*  

;* Subroutine SPAC2.  

;*  

;* Print two spaces.  

;*  

;*****  

D52E- 20 31 05  SPAC2  JSR SPACE ;output a space
;*****  

;*  

;* Subroutine SPACE.  

;*  

;* Print a space.  

;*  

;*****  

D531- A9 20      SPACE   LDA #$20  ;get a space
D533- 2C          .BY $2C    ;skip next instruction
;*****  

;*  

;* Subroutine CRLF.  

;*  

;* Print a Carriage Return.  

;* (Although label suggests it,  

;* no Line Feed is printed.  

;* This is to be considered as a  

;* bug).  

;*  

;*****  

D534- A9 00      CRLF   LD A #$0D ;get a Carriage Return
D536- 4C 66 F2      JMP BSOUT ;print the character in A
;*****  

;*  

;* Subroutine INCTMP.  

;*  

;* Increment TMPO and flag if in- *
;* cremented past $FFFF in WRAP. *
;*  

;*****  

D539- E6 FB      INCTMP  INC *TMPO  ;increment lo byte
D53B- D0 06      BNE SETWR ;if page crossed
D53D- E6 FC      INC *TMPO+1 ;increment hi also
D53F- D0 02      BNE SETWR ;if incremented past $FFFF
D541- E6 DE      INC *WRAP  ;flag wrap around

```

D543- 60 SETWR RTS ;and exit
.FI "BASIC4.00.M06"

1954 3411-4D65 BASIC4.00.M06

```
;name BASIC4.00.M06
;*****
;*          *
;* Table of MLM commands.      *
;*          *
;*****  

D544- 3A     CMDS   .BY ':'      ;modify memory contents
D545- 3B       .BY ';'      ;modify saved registers
D546- 52       .BY 'R'      ;display saved registers
D547- 4D       .BY 'M'      ;display memory
D548- 47       .BY 'G'      ;execute machine code
D549- 58       .BY 'X'      ;exit to BASIC
D54A- 4C       .BY 'L'      ;load from external medium
D54B- 53       .BY 'S'      ;save to external medium
;*****
;*          *
;* Table of hi byte addresses  *
;* of commands.                *
;*          *
;* The .BY directive is used  *
;* because MAE cannot handle  *
;* a construction like:       *
;* .SI H,ALTM-1.              *
;*          *
;*****  

D54C- D6     ADRH    .BY $06      ;should be: H,ALTM-1, modify memory con
D54D- D5       .BY $05      ;should be: H,ALTR-1, modify saved regi
D54E- D5       .BY $05      ;should be: H,DSPLYR-1, display saved r
D54F- D5       .BY $05      ;should be: H,DSPLYM-1, display memory
D550- D6       .BY $06      ;should be: H,GO-1, execute machine cod
D551- D6       .BY $06      ;should be: H,EXIT-1, exit to BASIC
D552- D6       .BY $06      ;should be: H,LD-1, load from external
D553- D6       .BY $06      ;should be: H,LD-1, save to external me
;*****
;*          *
;* Table of lo byte addresses  *
;* of commands.                *
;*          *
;* Concerning the .BY directive: *
;* See note above.              *
;*          *
;*****  

D554- 1C     ADRL    .BY $1C      ;should be: L,ALTM-1, modify memory con
D555- FA       .BY $FA      ;should be: L,ALTR-1, modify saved regi
D556- 86       .BY $86      ;should be: L,DSPLYR-1, display saved r
D557- 88       .BY $88      ;should be: L,DSPLYM-1, display memory
D558- 32       .BY $32      ;should be: L,GO-1, execute machine cod
D559- 6A       .BY $6A      ;should be: L,EXIT-1, exit to BASIC
D55A- 74       .BY $74      ;should be: L,LD-1, load from external
D55B- 74       .BY $74      ;should be: L,LD-1, save to external me
;*****
;*          *
;* Title for register display  *
;* command.                   *
;*          *
;*****
```

```

055C- 00 20 20 REGK .BY $00' PC IRQ SR AC XR YR SP'
055F- 20 20 20
0562- 50 43 20
0565- 20 49 52
0568- 51 20 20
056B- 53 52 20
056E- 41 43 20
0571- 58 52 20
0574- 59 52 20
0577- 53 50

;*****
;*
;* Subroutine ALTRIT.
;*
;*
;* Print a prompt and the character in Y on a new line.
;*
;*****
0579- 98 ALTRIT TYA ;save the extra character
057A- 48 PHA ;on stack
057B- 20 34 05 JSR CRLF ;start on a new line
057E- 68 PLA ;get the extra character back
057F- A2 2E LDX #'.' ;get a prompt
0581- 20 31 D7 JSR WRTWO ;print them
0584- 4C 2E 05 JMP SPAC2 ;followed by two spaces
;*****
;*
;* Command R.
;*
;*
;* Display saved registers.
;* Always executed on entering
;* the monitor.
;*
;*****
0587- A2 00 DSPLYR LDX #0 ;start at the beginning
0589- B0 5C 05 D2 LDA REGK,X ;get a title character
058C- 20 66 F2 JSR BSOUT ;print it
058F- E8 INX ;point to next character
0590- E0 10 CPX #29 ;if not all done,
0592- 00 F5 BNE D2 ;loop
0594- A0 3B LDY #';' ;else get command for alter reg's
0596- 20 79 05 JSR ALTRIT ;print it with a prompt
0599- AD 00 02 LDA PCH ;get PC hi
059C- 20 22 07 JSR WROB ;print in hex
059F- AD 01 02 LDA PCL ;get lo also
05A2- 20 22 07 JSR WROB ;print in hex
05A5- 20 31 05 JSR SPACE ;print a space
05A8- AD 07 02 LDA INVH ;get save IRQ hi
05AB- 20 22 07 JSR WROB ;print in hex
05AE- AD 08 02 LDA INVL ;get lo also
05B1- 20 22 07 JSR WROB ;print in hex
05B4- 20 23 05 JSR SETR ;prepare for display of other reg's
05B7- 20 F7 D4 JSR DM ;show them
05BA- F0 39 BEQ BEQS1 ;and restart monitor
;*****
;*
;* Command M.
;*
;*
;* Display memory in hex.
;* Displays eight bytes on a
;* line.
;*
;*****
05BC- 20 98 D7 DSPLYM JSR RDOC ;read 1st character of address

```

```

D5BF- 20 54 D7      JSR RDOA          ;read rest of start address in TMPO
D5C2- 90 34          BCC ERRS1        ;if no address, error
D5C4- 20 44 D7      JSR T2T2          ;save start address in TMP2
D5C7- 20 98 D7      JSR RDOC          ;read 1st character of end address
D5CA- 20 54 D7      JSR RDOA          ;read complete end address in TMPO
D5CD- 90 29          BCC ERRS1        ;if no address, error
D5CF- 20 44 D7      JSR T2T2          ;get SA in TMPO, EA in TMP2
D502- 20 35 F3  DSP1  JSR STOP1        ;check for STOP key
D505- F0 1E          BEQ BEQS1        ;if pressed, restart monitor
D507- A6 DE          LDX *WRAP         ;if address wrap around
D509- D0 1A          BNE BEQS1        ;restart monitor
D50B- 38             SEC              ;prepare for subtract
D50C- A5 FD          LDA *TMP2        ;get EA lo
D50E- E5 FB          SBC *TMP0        ;subtract SA lo
D50F- A5 FE          LDA *TMP2+1     ;get EA hi
D512- E5 FC          SBC *TMP0+1     ;subtract SA hi
D514- 90 0F          BCC BEQS1        ;if SA higher than EA, restart
D516- A0 3A          LDY #'::'       ;else get command for alter memory
D518- 20 79 D5      JSR ALTRIT       ;print it with a prompt
D51B- 20 17 D7      JSR WROA          ;print current address (=TMPO)
D51E- A9 08          LDA #8            ;display eight bytes
D520- 20 F7 D4      JSR DM            ;print them in hex
D523- F0 00          BEQ DSP1         ;and loop
;*****
;*                                         *
;* Intermediate JuMPs to be           *
;* reached via relative bran-       *
;* ches.                            *
;*                                         *
;*****
DSF5- 4C BA D4  BEQS1  JMP STRT          ;restart monitor
DSF8- 4C A4 D7  ERRS1  JMP ERRORP        ;give error and restart
;*****
;*                                         *
;* Command ;.                         *
;*                                         *
;* Alter saved registers.           *
;*                                         *
;*****
D5FB- 20 63 D7  ALTR  JSR RDOB          ;skip leading spaces
D5FE- 20 54 D7          JSR RDOA          ;read PC in TMPO
D601- 90 03          BCC AL2           ;if no valid input, try IRQ
D603- 20 EC D4          JSR PUPP          ;else load PC
D606- 20 15 F2  AL2  JSR BASIN         ;get a character (space)
D609- 20 54 D7          JSR RDOA          ;read IRQ vector
D60C- 90 0A          BCC AL3           ;if no valid input, try other reg's
D60E- A5 FB          LDA *TMP0          ;else get IRQ lo
D610- 80 08 02          STA INVL          ;in save area
D613- A5 FC          LDA *TMP0+1      ;and hi
D615- 80 07 02          STA INVH          ;also
D618- 20 23 D5  AL3  JSR SETR          ;prepare for other reg's
D61B- D0 0A          BNE A4           ;and fill these
;*****
;*                                         *
;* Command :.                         *
;*                                         *
;* Alter displayed memory.          *
;*                                         *
;*****
D61D- 20 63 D7  ALTM  JSR RDOB          ;skip leading spaces
D620- 20 54 D7          JSR RDOA          ;read PC in TMPO
D623- 90 03          BCC ERRS1        ;if no valid input, error
D625- A9 08          LDA #8            ;store eight bytes
D627- 85 B5          A4               STA *TMPC        ;save # of bytes (5 or 8)

```

```
D629- 20 98 D7 A5      JSR RDOC          ;read character (space)
D62C- 20 0B D5      JSR BYTE           ;read and store bytes
D62F- D0 F8      BNE A5             ;until all done
D631- F0 C2      BEQ BEQS1         ;then restart monitor
.FI "BASIC4.00.M07"
```

190B 3411-4D1C BASIC4.00.M07

```
;name BASIC4.00.M07
;*****
;*                               *
;* Command G.                  *
;*                               *
;* Load registers with saved   *
;* values and execute code star-*
;* ting at given or saved PC.   *
;*                               *
;*****
```

D633- 20 15 F2 G0	JSR BASIN ;get a character
D636- C9 00	CMP #\$00 ;if it was return
D638- F0 0C	BEQ G1 ;execute from saved PC
D63A- C9 20	CMP #\$20 ;if character was not space
D63C- D0 BA	BNE ERRS1 ;give error
D63E- 20 54 07	JSR RDOA ;else read address
D641- 90 03	BCC G1 ;if not valid, use saved PC
D643- 20 EC D4	JSR PUTP ;else load PC with given value
D646- AE 06 02 G1	LDX SP ;get saved stackpointer
D649- 9A	TXS ;in stackpointer
D64A- 78	SEI ;disable interrupts
D64B- AD 07 02	LDA INVH ;get saved IRQ hi
D64E- 85 91	STA *CINV+1 ;load IRQ vector hi
D650- AD 08 02	LDA INVL ;get lo
D653- 85 90	STA *CINV ;and complete vector
D655- AD 00 02	LDA PCH ;push
D658- 48	PHA ;start address
D659- AD 01 02	LDA PCL ;on
D65C- 48	PHA ;stack
D65D- AD 02 02	LDA FLGS ;add flags also
D660- 48	PHA ;on stack
D661- AD 03 02	LDA ACC ;load accu
D664- AE 04 02	LDX XR ;and X
D667- AC 05 02	LDY YR ;and Y
D66A- 40	RTI ;load PC and flags, execute

	;* *
	;* Command X. *
	;* *
	;* Exit to warm start of BASIC. *
	;* *

D668- AE 06 02 EXIT	LDX SP ;get saved stackpointer
D66E- 9A	TXS ;in stackpointer
D66F- 4C FF B3	JMP READY ;and warm start BASIC

	;* *
	;* Intermediate JuMP to be rea- *
	;* ched via relative branches. *
	;* *

D672- 4C A4 D7 ERRL	JMP ERROPR ;give error, then restart

	;* *
	;* Commands L and S. *

```

        ;*
        ;* Save or load a block of      *
        ;* memory.                      *
        ;*                                *
        ;*****                         *
D675- A0 01    LD    LDY #1          ;default is device 1 (cassette #1)
D677- 84 04    STY *FA          ;set device number
D679- 88       DEY          ;Y=0
D67A- 84 D1    STY *FNLEN      ;initial filename length is zero
D67C- 84 96    STY *CSTAT      ;clear status byte
D67E- 84 90    STY *VERCK      ;set LOAD/VERIFY to LOAD
D680- A9 02    LDA #H,SAVNAM   ;place for filename
D682- 85 DB    STA *FNADR+1    ;is
D684- A9 09    LDA #L,SAVNAM   ;in
D686- 85 DA    STA *FNADR      ;input buffer
D688- 20 15 F2  L1    JSR BASIN    ;get a character (why not FFCF?)
D68B- C9 20    CMP #$20        ;if it is a space
D68D- F0 F9    BEQ L1          ;get next
D68F- C9 00    CMP #$00        ;if it is return
D691- F0 1A    BEQ L5          ;must be load with defaults
D693- C9 22    CMP #''
D695- D0 DB    L2    BNE ERR1      ;give error
D697- 20 15 F2  L3    JSR BASIN    ;else get character of filename
D69A- C9 22    CMP #''
D69C- F0 24    BEQ L8          ;check for further input
D69E- C9 00    CMP #$00        ;if return
D6A0- F0 0B    BEQ L5          ;must be load with filename
D6A2- 91 DA    STA (FNADR),Y   ;else save filename character
D6A4- E6 D1    INC *FNLEN      ;adapt length of name
D6A6- C8       INY          ;count characters of name
D6A7- C0 10    CPY #16          ;if name is 16 characters long
D6A9- F0 C7    L4    BEQ ERR1      ;give error
;BUG: increments first, then tests, so
;      error is given at 15 characters instead
;      of 16.
;BUG: does not allow for at-sign (for save with
;      replace), drive number and colon for
;      diskdrive.
;CURE:replace CPY #16 with CPY #20
D6AB- D0 EA    L5    BNE L3          ;else get next filename character
D6AD- A5 B4    LDA *SAVX      ;get command number
D6AF- C9 06    CMP #6           ;if command was not L
D6B1- D0 E2    L6    BNE L2          ;give error
D6B3- 20 56 F3  JSR LD15      ;else go LOAD
D6B6- 20 2B F9  JSR TWAIT      ;and abort I/O
D6B9- A5 96    LDA *CSTAT      ;get status byte
D6BB- 29 10    AND #$10        ;if cassette load error
D6BD- D0 F2    L7    BNE L6          ;give error, restart
D6BF- 4C BA D4  JMP STRT      ;else restart monitor
D6C2- 20 15 F2  L8    JSR BASIN    ;filename read, get a character
D6C5- C9 00    CMP #$00        ;if return
D6C7- F0 E4    BEQ L5          ;can be load only
D6C9- C9 2C    CMP #''
D6CB- D0 F0    L9    BNE L7          ;else character mus be comma
D6CD- 20 63 D7  JSR RD0B      ;if not, give error
D6DD- 29 0F    AND #$0F        ;else read device number (2 char's!)
D6D2- F0 D5    L10   BEQ L4          ;remove hi nibble
D6D4- C9 03    CMP #3           ;if L/S from keyboard, error
D6D6- F0 FA    L11   BEQ L10      ;if L/S from screen
D6D8- 85 D4    STA *FA          ;give error
D6DA- 20 15 F2  JSR BASIN    ;else store device number
D6DD- C9 00    CMP #$00        ;get next character
D6DF- F0 CC    BEQ L5          ;if return
D6E1- C9 2C    CMP #''
;can be load only
;if not comma

```

D6E3- D0 E6	L12	BNE L9 JSR RDOA JSR T2T2 JSR BASIN CMP #',' BNE L12 JSR RDOA LDA *TMPO STA *EAL LDA *TMPO+1 STA *EAH JSR T2T2 JSR BASIN CMP #\$20 BEQ L20 CMP #\$0D BNE L13 LDA *SAVX CMP #7 BNE L14 JSR SVS JMP STRT	;give error ;else read start address in TMPO ;move it to TMP2 ;get next character ;must be comma ;if not, give error ;read end address ;get address lo ;store as end address for save ;get hi also ;and complete end address ;get start address into place ;get next character ;if space, get another ;(why allow spaces here, and not in res ;if not return ;give error ;get command number ;if command is not S ;give error ;else go save ;and restart monitor
D6E5- 20 54 D7		;	*****
D6E8- 20 44 D7		;	*
D6EB- 20 15 F2		;	*
D6EE- C9 2C		;	*
D6F0- D0 F1	L13	;	*
D6F2- 20 54 D7		;	*
D6F5- A5 FB		;	*
D6F7- 85 C9		;	*
D6F9- A5 FC		;	*
D6FB- 85 CA		;	*
D6FD- 20 44 D7	L20	;	*
D700- 20 15 F2		;	*
D703- C9 20		;	*
D705- F0 F9		;	*
D707- C9 OD		;	*
D709- D0 E5	L14	;	*
D70B- A5 B4		;	*
D70D- C9 07		;	*
D70F- D0 F8		;	*
D711- 20 E3 F6		;	*
D714- 4C BA D4		;	*
;*****			
;*			
;* Subroutine WROA.			
;*			
;* Prints TMPO as a hex address.			
;*			
;*****			
D717- A2 01	WROA	LDX #1 LDA *TMPO-1,X PHA LDA *TMPO,X JSR WROB PLA	;why is this? ;get lo byte (why indexed?) ;save it on stack ;get hi (why indexed?) ;print as hex number ;get lo again, and:
D719- B5 FA		;	*****
D71B- 48		;	*
D71C- B5 FB		;	*
D71E- 20 22 D7		;	*
D721- 68		;	*
;*****			
;*			
;* Subroutine WROB.			
;*			
;* Prints A as a hex. byte.			
;*			
;*****			
D722- 48	WROB	PHA LSR A LSR A LSR A LSR A JSR ASC TAX PLA AND #\$0F JSR ASC	;save A on stack ;move hi ;nibble ;to ;lo nibble ;convert nibble to ASCII ;save it in X ;get original back ;get lo nibble ;convert it to ASCII, and:
D723- 4A		;	*****
D724- 4A		;	*
D725- 4A		;	*
D726- 4A		;	*
D727- 20 3A D7		;	*
D72A- AA		;	*
D72B- 68		;	*
D72C- 29 OF		;	*
D72E- 20 3A D7		;	*
;*****			
;*			
;* Subroutine WRTWO.			
;*			
;* Prints the ASCII characters			
;* in X and A.			
;*			
;*****			
D731- 48	WRTWO	PHA TXA JSR BSOUT PLA	;save A on stack ;get the character in X ;print it (why not FFD2?) ;get 2nd character back
D732- 8A		;	*
D733- 20 66 F2		;	*
D736- 68		;	*

D737- 4C 66 F2 JMP BSOUT ;print it also
;*****
;* *
;* Subroutine ASC. *
;* *
;* Converts the nibble in A to *
;* its ASCII hex equivalent. *
;* *
;*****
D73A- 18 ASC CLC ;prepare for add
D73B- 69 F6 ADC #\$F6 ;subtract' nine
D73D- 90 02 BCC AS1 ;if no carry
D73F- 69 06 ADC #\$06 ;add offset for letter
D741- 69 3A AS1 ADC #\$3A ;and convert to ASCII
D743- 60 RTS RTS ;then exit
.FI "BASIC4.0D.M08"

13AO 3411-47B1 BASIC4.0D.M08

;name BASIC4.0D.M08
;*****
;* *
;* Subroutine T2T2. *
;* *
;* Exchanges TMPO and TMP2, both *
;* hi and lo bytes. *
;* *
;*****
D744- A2 02 T2T2 LDX #2 ;exchange two pairs
D746- B5 FA TT LDA *TMPO-1,X ;get byte from TMPO
D748- 48 PHA PHA ;save it on stack
D749- B5 FC LDA *TMP2-1,X ;get byte from TMP2
D74B- 95 FA STA *TMPO-1,X ;in TMPO
D74D- 68 PLA PLA ;get byte from TMPO back
D74E- 95 FC STA *TMP2-1,X ;and move to TMP2
D750- CA DEX DEX ;adapt byte counter
D751- D0 F3 BNE TT BNE TT ;if not all done, loop
D753- 60 RTS RTS ;else exit
;*****
;* *
;* Subroutine RDOA. *
;* *
;* Read a hex address in TMPO. *
;* *
;*****
D754- 20 63 07 RDOA JSR RDOB ;read the hi byte
D757- 90 02 BCC R1 BCC R1 ;if no valid input, try the lo byte
D759- 85 FC STA *TMPO+1 STA *TMPO+1 ;else store the hi byte
D75B- 20 63 07 R1 JSR RDOB ;read the lo byte
D75E- 90 02 BCC R2 BCC R2 ;if no valid input, exit
D760- 85 FB STA *TMPO STA *TMPO ;else store the lo byte
D762- 60 R2 RTS ;and exit
;*****
;* *
;* Subroutine RDOB. *
;* *
;* Read a hex byte in A. *
;* *
;*****
D763- A9 00 RDOB LDA #\$00 ;clear
D765- 80 00 01 STA \$0100 STA \$0100 ;result area
D768- 20 98 07 JSR RDOC JSR RDOC ;read a character
D76B- C9 20 CMP #\$20 CMP #\$20 ;if it is not a space

```

D760- 00 09      BNE R3           ;go convert to a number
D76F- 20 98 D7    JSR RDOC        ;read next character
D772- C9 20      CMP #$20        ;if it is not a space
D774- 00 0F      BNE R4           ;go complete byte
D776- 18          CLC             ;else flag error
D777- 60          RTS             ;and exit
D778- 20 80 07   R3      JSR HEXIT     ;convert character to nibble
D77B- 0A          ASL A           ;move nibble
D77C- 0A          ASL A           ;to
D77D- 0A          ASL A           ;the
D77E- 0A          ASL A           ;hi nibble
D77F- 80 00 01    STA $0100       ;leave it in work area
D782- 20 98 D7    JSR RDOC        ;read next character
D785- 20 80 07   R4      JSR HEXIT     ;convert to nibble
D788- 00 00 01    ORA $0100       ;add hi nibble
D78B- 38          SEC             ;flag complete and correct byte
D78C- 60          RTS             ;and exit
;*****
;*
;* Subroutine HEXIT.          *
;*
;* Converts the ASCII character *
;* in A to a nibble.           *
;*
;*****
D78D- C9 3A      HEXIT          CMP #$3A        ;set carry if character is a letter
D78F- 08          PHP             ;save the flags
D790- 29 0F      AND #$0F        ;remove ASCII
D792- 28          PLP             ;get the flags back
D793- 90 02      BCC H1         ;if character was a number, exit
D795- 69 08      ADC #$08        ;add 9 (carry was set)
D797- 60          H1      RTS             ;and exit
;*****
;*
;* Subroutine RDOC.          *
;*
;* Read one character in A.   *
;*
;*****
D798- 20 15 F2   RDOC          JSR BASIN        ;get a character
D798- C9 0D      CMP #$0D        ;if it is not return
D79D- 00 F8      BNE H1         ;exit
D79F- 68          PLA             ;else remove
D7A0- 68          PLA             ;return address
D7A1- 4C BA D4    JMP STRT        ;and restart
;*****
;*
;* Routine ERROPR.          *
;*
;* Universal error exit.     *
;* Default USRCMD points to *
;* this routine.              *
;*
;*****
D7A4- A9 3F      ERROPR         LDA #'?'       ;get a question mark
D7A6- 20 66 F2    JSR BSOUT        ;print it (why not FFD2?)
D7A9- 4C BA D4    JMP STRT        ;and restart monitor
;*****
;*
;* SYNTAX ERROR exit.        *
;*
;*****
D7AC- 4C 00 BF   SYNERR         JMP SNERR       ;give SYNTAX ERROR, then restart
;*****

```

```

        ;*
        ;* Perform RECORD.          *
        ;*
        ;* (Called from Kernel: $FF9C).  *
        ;*
;*****  

D7AF- A9 01    RECORD LDA #1      ;get default byte position
D7B1- 80 3A 03 STA POS       ;in $033A
D7B4- 20 76 00 JSR CHRGOT    ;get current character
D7B7- A9 23    LDA #'#'     ;# must be next
D7B9- 20 F7 BE JSR SYNCHR    ;test if so
D7BC- 20 8A DE JSR GTVL2     ;else read file number in X
D7BF- E0 00    CPX #0        ;if file number is zero
D7C1- F0 3E    BEQ QTYERL    ;give ILLEGAL QUANTITY ERROR
D7C3- 86 D2    STX *LA       ;else store it in current filenumber
D7C5- 20 F5 BE JSR CHKCOM    ;test for comma
D7C8- F0 E2    BEQ SYNERR    ;if end of statement, ?SYNTAX ERROR
D7CA- 90 0F    BCC NUMADR    ;if character not numeric
D7CC- 20 F2 BE JSR CHKOPN    ;test for (
D7CF- 20 98 BD JSR FRMEVL    ;evaluate expression for record number
D7D2- 20 20 C9 JSR GETADR    ;convert FLP accu1 to integer
D7D5- 20 EF BE JSR CHKCLS    ;test for )
D7D8- 4C E1 D7 JMP REXNEX    ;and continue
D7DB- 20 98 BD NUMADR JSR FRMEVL    ;if numeric, evaluate expression (rec.
D7DE- 20 20 C9 GETADR       ;convert FLP accu1 to integer
D7E1- 20 76 00 REXNEX JSR CHRGOT    ;get current character
D7E4- F0 18    BEQ DONER      ;if end of statement, send parameters
D7E6- 20 F5 BE JSR CHKCOM    ;else test for comma
D7E9- F0 C1    BEQ SYNERR    ;if end of statement, SYNTAX ERROR
D7EB- 20 8A DE JSR GTVL2     ;else read byte position in X
D7EE- E0 00    CPX #0        ;if position zero
D7F0- F0 0F    BEQ QTYERL    ;ILLEGAL QUANTITY ERROR
D7F2- E0 FF    CPX #$FF      ;if beyond sector size
D7F4- F0 0B    BEQ QTYERL    ;also ILLEGAL QUANTITY ERROR
D7F6- 8E 3A 03 STX POS       ;else store byte position
D7F9- 20 76 00 JSR CHRGOT    ;get current character
D7FC- D0 AE    BNE SYNERR    ;if not end of statement, SYNTAX ERROR
D7FE- 4C 31 DA  DONER    JMP BOREC    ;else send RECORD parameters to disk
.FI "BASIC4.00.M09"

```

15E2 3411-49F3 BASIC4.00.M09

```

;name BASIC4.00.M09
;*****  

;*           *
;* ILLEGAL QUANTITY ERROR exit.  *
;*           *
;*****  

D801- 4C 27 DE  QTYERL JMP QTYERR      ;give ILLEGAL QUANTITY ERROR, then rest
;*****  

;*           *
;* Check SYNTAX of disk command. *
;*           *
;* (Source) filename must be   *
;* given. Device number and   *
;* source drive# are optional. *
;* Other parameters generate  *
;* SYNTAX ERROR.               *
;*           *
;*****  

D804- 29 E6    CHK1  AND #$E6      ;if at-sign, write, dest. drive,
D806- F0 03    BEQ CHK2      ;or file number or dest. filename given
D808- 4C D0 BF  CHKER1 JMP SNERR    ;give SYNTAX ERROR

```

D80B- AD 3E 03	CHK2	LDA PARCHK ;else get flags AND #\$01 ;get flag for (source) filename CMP #\$01 ;if no (source) filename given BNE CHKER1 ;SYNTAX ERROR LDA PARCHK ;else get flags again RTS ;and exit ;***** ;* ;* Check SYNTAX of disk command. * ;* ;* Only source drive# and one * ;* filename are allowed. * ;* Other parameters give SYNTAX * ;* ERROR. * ;* ;*****
D818- 29 E7	CHK3	AND #\$E7 ;if other than source drive# and filena BNE CHKER1 ;give SYNTAX ERROR RTS ;else exit ;***** ;* ;* Check SYNTAX of disk command. * ;* ;* Two file names must be given, * ;* destination drive#, source * ;* drive# and device number are * ;* optional. * ;* ;*****
D81D- 29 C4	CHK4	AND #\$C4 ;if at-sign, write or device# given, BNE CHKER1 ;do SYNTAX ERROR LDA PARCHK ;else get flags again
D81F- D0 E7		AND #\$03 ;get flags for both filenames
D821- AD 3E 03		CMP #\$03 ;if one omitted
D824- 29 03	CHK5	BNE CHKER1 ;give SYNTAX ERROR LDA PARCHK ;else get flags again
D826- C9 03		RTS ;and exit. ;***** ;* ;* Check SYNTAX of disk command. * ;* ;* A filename and a file number * ;* must be present. Other para- * ;* meters are optional. * ;* ;*****
D828- D0 DE		AND #\$05 ;if no filename or file# CMP #\$05 ;given, BNE CHKER1 ;give SYNTAX ERROR
D82A- AD 3E 03		LDA PARCHK ;else get flags again
D82D- 60		RTS ;and exit ;***** ;* ;* Dummy messages as bodies for * ;* messages to be sent to disk * ;* as commands. * ;* Meaning of the message items: * ;* ASCII letter: command name * ;* \$D0 : disk ID * ;* \$D1 : source drive # * ;* \$D2 : dest. drive # * ;* \$E0 : read/write (W,L)* ;* \$E1 : file type (S,R) * ;* \$F1 : source filename *
D82E- 29 05	CHK6	
D830- C9 05		
D832- D0 D4		
D834- AD 3E 03		
D837- 60		

```

;* $F2          : dest. filename *
;*
;*****  

D838- FF      TABLD .BY $FF           ; ;?????  

D839- 24 D1    d839 .BY '$' $D1       ;DIRECTORY  

D83B- D1 3A F1 d83b .BY $D1 '' $F1   ;DLOAD, DSAVE  

D83E- 2C E1 2C d83e .BY ',' $E1 ',', $EO ;DOPEN (together with DLOAD)  

D841- EO  

D842- D1 3A F1 d842 .BY $D1 '' $F1 ',' ;RECORD  

D845- 2C  

D846- 41      d846 .BY 'A'          ; ;APPEND (together with HEADER)  

D847- 4E D1 3A d847 .BY 'N' $D1 '' $F1 ',', $DO ;HEADER  

D84A- F1 2C D0  

D84D- 56 D1    d84d .BY 'V' $D1       ;COLLECT  

D84F- 44 D2 3D d84f .BY 'D' $D2 '=' $D1 ;BACKUP  

D852- D1  

D853- 43 D2 3A d853 .BY 'C' $D2 '' $F2 '=' $D1 '' $F1 ;COPY  

D856- F2 3D D1  

D859- 3A F1  

D85B- 43 D2 3A d85b .BY 'C' $D2 '' $F2 '=' $D2 '' $F2 ',', $D1 '' $F1  

D85E- F2 3D D2  

D861- 3A F2 2C  

D864- D1 3A F1  

D867- 52 D1 3A d867 ; ;CONCAT  

.DBY 'R' $D1 ':' $F2 '=' $D1 ':' $F1 ;RENAME  

D86A- F2 3D D1  

D86D- 3A F1  

D86F- 53 D1 3A d86f .BY 'S' $D1 ':' $F1 ;SCRATCH  

D872- F1  

;*****  

;* Perform DIRECTORY or CATALOG. *
;* (Called from Kernal: $FFB4). *
;*  

;*****  

D873- A5 D2    CATLOG LDA *LA        ;save current file number  

D875- 85 B3    STA *WSW         ;in temporary location  

D877- 20 68 DC  JSR DOSPAR      ;read parameters  

D87A- 20 18 D8  JSR CHK3        ;test if 1 drive# and 1 filename  

D87D- A0 00    CATALG LDY #d839-d839 ;get offset in table  

D87F- A2 01    LDX #1          ;get initial dummy message length  

D881- AD 3E 03  LDA PARCHK     ;get syntax flags  

D884- 29 10    AND #$10        ;if drive number given  

D886- F0 01    BEQ CATBLD      ;  

D888- E8          INX            ;adapt dummy command length  

D889- 8A    CATBLD TXA        ;move length to A  

D88A- 20 FA DB  JSR SENDP       ;and build command string  

D88D- A5 B0    LDA *DFLTO      ;get current output device number  

D88F- 85 BA    STA *CNTDN      ;save it  

D891- A9 60    LDA #$60        ;get secondary address 0 (=LOAD)  

D893- 85 D3    STA *SA          ;in current secondary address  

D895- A9 0E    LDA #14          ;get dummy filenumber (14)  

D897- 85 D2    STA *LA          ;in current file number  

D899- 20 B9 F1  JSR UNLSN      ;unlisten all IEEE devices  

D89C- 20 65 F5  JSR FOPEN      ;perform OPEN  

D89F- A9 00    LDA #0          ;clear  

D8A1- 85 96    STA *CSTAT      ;status byte  

D8A3- A0 03    LDY #3          ;read three words (start address, link,  

D8A5- 84 D1    WG220 STY *FNLEN ;save counter  

D8A7- A2 0E    LDX #14          ;get file number again  

D8A9- 20 AF F7  JSR CHKIN      ;set input device (why not $FFC6?)  

D8AC- 20 15 F2  JSR BASIN      ;get a byte from disk (why not $FFCF?)  

D8AF- 85 FD    STA *FNADR2    ;store lo byte

```

D8B1- A4 96	LDY *CSTAT	;if ST shows I/O error
D8B3- D0 5D	BNE WG230	;abort and exit
D8B5- 20 15 F2	JSR BASIN	;else get a byte from disk
D8B8- 85 FE	STA *FNADR2+1	;store hi byte
D8BA- A4 96	LDY *CSTAT	;if I/O error
D8BC- D0 54	BNE WG230	;abort I/O and exit
D8BE- A4 D1	LDY *FNLEN	;else get word counter
D8C0- 88	DEY	;if not drive number yet
D8C1- D0 E2	BNE WG220	;read next two bytes
D8C3- 20 A6 F2	JSR CLRCHN	;restore default I/O
D8C6- 20 23 D9	JSR SUBB	;set output device
D8C9- A6 FD	LDX *FNADR2	;get drive # or # of blocks lo
D8CB- A5 FE	LDA *FNADR2+1	;and hi
D8CD- 20 83 CF	JSR LINPRT	;print them in decimal
D8DD- A9 20	LDA #\$20	;get a space
D8D2- 20 66 F2	JSR BSOUT	;print it (why not \$FFD2 or SPACE in ML
D8D5- 20 A6 F2	JSR CLRCHN	;restore default I/O
	.FI "BASIC4.00.M10"	

1AFC 3411-4F00 BASIC4.00.M10

;name BASIC4.00.M10		
;*****		
;*		
;* Last part of perform DIRECTO- *		
;* RY or CATALOG. *		
;*		
;*****		
D8D8- A2 0E	WG250	LDX #14 ;get file number again
D8DA- 20 AF F7		JSR CHKIN ;set input device
D8DD- 20 15 F2		JSR BASIN ;get a byte from diskname/filename
D8E0- 48		PHA ;save the character
D8E1- 20 A6 F2		JSR CLRCHN ;restore default I/O
D8E4- 68		PLA ;reget character
D8E5- A6 96		LDX *CSTAT ;if any I/O error
D8E7- D0 29		BNE WG230 ;abort and exit
D8E9- C9 00		CMP #\$00 ;if end of directory line
D8EB- F0 18		BEQ WG240 ;print return and loop
D8ED- 20 1A D9		JSR SUBA ;else print the character
D8F0- 20 35 F3		JSR STOP1 ;check for STOP key
D8F3- F0 1D		BEQ WG230 ;if pressed, abort and exit
D8F5- 20 05 F2		JSR GETIN ;get a key from keyboard
D8F8- F0 DE		BEQ WG250 ;if none, do next file/diskname char.
D8FA- C9 20		CMP #\$20 ;if space
D8FC- D0 DA		BNE WG250
D8FE- 20 05 F2	WG255	JSR GETIN ;get another character
D901- F0 FB		BEQ WG255 ;if none, wait for one
D903- D0 D3		BNE WG250 ;if a character, do next file/diskname
D905- A9 0D	WG240	LDA #\$0D ;end of line, get a return
D907- 20 1A D9		JSR SUBA ;print it
D90A- 20 B9 F1		JSR UNLSN ;unlisten all IEEE devices
D90D- A0 02		LDY #2 ;now skip two words
D90F- D0 94		BNE WG220 ;and do next line
D911- 68	WG235	PLA ;?? (unused instruction)
D912- 20 A6 F2	WG230	JSR CLRCHN ;restore default I/O
D915- A9 0E		LDA #14 ;get file number again
D917- 4C E2 F2		JMP CLOSS ;close the file and exit
;*****		
;*		
;* Output a character to current *		
;* output device. *		
;*		
;*****		

```

D91A- 20 23 D9 SUBA    JSR SUBB      ;set output device if necessary
D91D- 20 66 F2          JSR BSOUT     ;print the character in A
D920- 4C A6 F2          JMP CLRCHN   ;and restore default I/O
;*****
;*                                     *
;* Set output device if output      *
;* not to screen.                  *
;*                                     *
;*****
D923- A6 BA    SUBB    LDX *CNTDN   ;get saved output device
D925- E0 03          CPX #3        ;if it was screen
D927- F0 05          BEQ SUBBR     ;exit
D929- A6 B3          LDX *WSW       ;else get output device file number
D92B- 20 FE F7          JSR CHKOUT   ;set output device
D92E- 60    SUBBR    RTS       ;and exit
;*****
;*                                     *
;* Find next possible secondary    *
;* address.                      *
;*                                     *
;*****
D92F- A0 61    ENTRY0  LDY #$61      ;get 1st possible sec. address
D931- C8      ENTRY1  INY       ;adapt secondary address
D932- 98          TYA       ;get it in A
D933- A6 AE          LDX *LDTND   ;get number of open files
D935- CA      ENTRY2  DEX       ;if all done,
D936- 30 07          BMI RFOUND  ;exit
D938- DD 65 02          CMP SAT,X  ;else compare with sec. address used
D93B- F0 F4          BEQ ENTRY1  ;if used, try next sec. address
D93D- D0 F6          BNE ENTRY2  ;else try next open file
D93F- 84 D3    RFOUND  STY *SA      ;save generated secondary address
D941- 60          RTS       ;and exit
;*****
;*                                     *
;* Perform DOPEN.                 *
;*                                     *
;* (Called from Kernal: $FF96).  *
;*                                     *
;*****
D942- 20 68 DC    DOPEN   JSR DOSPAR  ;read parameters
D945- 20 2E D8          JSR CHK6      ;check if filename and file# given
D948- 29 22          AND #$22      ;dest. filename or drive# given
D94A- F0 03          BEQ DOPEN2   ;DOPEN2
D94C- 4C 00 BF          JMP SNERR     ;give SYNTAX ERROR
D94F- 20 2F 09    DOPEN2  JSR ENTRY0  ;else get next possible sec. address
D952- A0 02          LDY #d83b-d839  ;get offset in table
D954- A2 03          LDX #d842-d83b-4 ;get length for read access
D956- 2C 3E 03          BIT PARCHK   ;if write parameter given
D959- 50 02          BVC LEAV
D95B- A2 07    RECLKC LDX #d842-d83b  ;get new length
D95D- 2C 3E 03    LEAV    BIT PARCHK   ;if at-sign in filename
D960- 10 0E          BPL LEAV1
D962- A9 40          LDA #'@'      ;precede commandstring
D964- 80 53 03          STA TBUFF     ;with an at-sign
D967- 8A            TXA       ;get length of dummy command in A
D968- A2 01          LDX #1        ;get offset for commandstring
D96A- 20 FC DB          JSR SENDP1   ;build commandstring
D96D- 4C 63 F5          JMP OP94      ;and perform OPEN
D970- 8A    LEAV1  TXA       ;if no at-sign, get length in A
D971- 20 FA DB          JSR SENDP   ;build commandstring
D974- 4C 63 F5          JMP OP94      ;and perform OPEN
;*****
;*                                     *
;* Perform APPEND.                *

```

```

        ;*                                     *
        ;* (Called from Kernal: $FFAB).  *
        ;*                                     *
        ;*****  

D977- 20 68 DC APPEND JSR DOSPAR      ;read parameters
D97A- 20 2E D8 JSR CHK6          ;check if filename and number given
D97D- 29 E2 AND #$E2          ;if at-sign, write, dest. drive# or
D97F- F0 03 BEQ DAPPEN       ;destination filename given,
D981- 4C 00 BF JMP SNERR        ;give SYNTAX ERROR
D984- 20 2F D9 DAPPEN JSR ENTRYD    ;else get next possible sec. address
D987- A0 09 LDY #d842-d839   ;get offset in table
D989- A9 05 LDA #d847-d842   ;and length
D98B- 20 FA DB JSR SENDP       ;build command string
D98E- 4C 63 F5 JMP OP94        ;and perform OPEN
        ;*****  

        ;*                                     *
        ;* Read error channel of disk  *
        ;* and fill DS$.               *
        ;*                                     *
        ;* Check if DS$ exits already. *
        ;*                                     *
        ;*****  

D991- A5 0D ERRCH1  LDA *DSDESC    ;if length of DS$ is not zero
D993- D0 16 ECHKS   BNE ECHKS     ;go read and fill it, else
        ;*****  

        ;*                                     *
        ;* Read error channel of disk  *
        ;* and fill DS$.               *
        ;*                                     *
        ;* DS$ will be set up.          *
        ;*                                     *
        ;* (Called from Kernal: $FFAB).  *
        ;*                                     *
        ;*****  

D995- A9 28 GETDS   LDA #40        ;set initial length of DS$
D997- 85 0D          STA *DSDESC    ;to forty
D999- 20 1D C6          JSR GETSPA    ;allocate space for 40 characters
D99C- 86 0E          STX *DSDESC+1  ;save generated
D99E- 84 0F          STY *DSDESC+2  ;pointer in ($0E)
D9A0- A9 00          LDA #0         ;clear status byte and
D9A2- A0 29          LDY #41        ;move terminator
D9A4- 20 9E DE          JSR de9e      ;at end of string (patch)
D9A7- A9 0D          LDA #$0D      ;move a return
D9A9- 91 0E          STA (DSDESC+1),Y ;to end of string
D9AB- A5 D4          ECHKS   LDA *FA        ;get current device number
D9AD- D0 04          BNE EREAD     ;if zero
D9AF- A9 08          LDA #8         ;get default
D9B1- 85 D4          STA *FA        ;in current device number
D9B3- 20 D2 F0          EREAD   JSR TALK      ;send talk to disk
D9B6- A9 6F          LDA #$6F      ;get secondary address 15
D9B8- 85 D3          STA *SA        ;in current sec. address
D9BA- 20 93 F1          JSR TKSA      ;send sec. address
D9BD- A0 FF          LDY #$FF      ;get initial index
D9BF- C8          LOOP1   INY          ;adapt index
D9C0- 20 C0 F1          JSR ACPTR     ;get byte from disk
D9C3- C9 0D          CMP #$0D      ;if character is return
D9C5- F0 04          BEQ ERREND    ;terminate DS$
D9C7- 91 0E          STA (DSDESC+1),Y ;else add character to DS$
D9C9- D0 F4          BNE LOOP1    ;and loop
D9CB- A9 00          ERREND   LDA #$00      ;get terminator
D9CD- 91 0E          STA (DSDESC+1),Y ;add to DS$
D9CF- 4C AE F1          JMP UNTLK    ;and untalk disk
        ;*****  

        ;*                                     *

```

```

        ;* Perform HEADER.          *
        ;*                         *
        ;* (Called from Kernel: $FF9F).  *
        ;*                         *
        ;*****  

D9D2- 20 68 DC  FORMAT JSR DOSPAR      ;read parameters
D9D5- 20 04 D8      JSR CHK1       ;test if filename given
D9D8- 29 11      AND #$11       ;source drive and diskname
D9DA- C9 11      CMP #$11       ;must be
D9DC- F0 03      BEQ DFORMA    ;given
D9DE- 4C 00 BF  FERO   JMP SNERR     ;else SYNTAX ERROR
D9E1- 20 1B DA  DFORMA JSR DCLALL   ;close all files
D9E4- 20 9E DB      JSR RUSURE    ;print ARE YOU SURE, get input
D9E7- 90 01      BCC FBUILD    ;if not sure
D9E9- 60          RTS           ;exit
D9EA- A0 0E      FBUILD LDY #d847-d839 ;get offset in table
D9EC- A9 04      LDA #d84d-d847-2 ;and length of dummy
D9EE- AE 3F 03      LDX DISKID    ;if no ID given
D9F1- F0 02      BEQ FCNT       ;send no ID
D9F3- A9 06      LDA #d84d-d847 ;else get new dummy string length
D9F5- 20 98 DA  FCNT   JSR TRANS     ;build and send commandstring
D9F8- 20 91 D9  FERRS  JSR ERRCH1   ;read DS$
D9FB- A0 00      LDY #0         ;point to 1st character
D9FD- B1 0E      LDA (DSDESC+1),Y ;get it
D9FF- C9 32      CMP #'2'       ;if error# <20
DA01- B0 01      BCS FERRP     ;exit
DA03- 60          RTS           ;exit
DA04- 4C D7 DB  FERRP  JMP BADDIS    ;else print ? BAD DISK
.FI "BASIC4.00.M11"

```

18EF 3411-4000 BASIC4.00.M11

```

;name BASIC4.00.M11
*****  

        ;*                         *
        ;* Perform DCLOSE.          *
        ;*                         *
        ;* (Called from Kernel: $FF99).  *
        ;*                         *
        ;*****  

DA07- 20 68 DC  DCLOSE  JSR DOSPAR    ;read parameters
DA0A- 29 F3      AND #$F3       ;if filename or
DA0C- F0 03      BEQ DCLSE     ;device number given
DA0E- 4C 00 BF      JMP SNERR     ;give SYNTAX ERROR
DA11- 20 E1 DB  DCLSE   JSR OLDCLR   ;else clear DS$
DA14- A5 D2      LDA *LA        ;get current file number
DA16- F0 03      BEQ DCLALL   ;if not zero
DA18- 4C E2 F2      JMP CLOSS     ;close that file
DA1B- A5 D4      DCLALL LDA *FA        ;else get current device number
DA1D- A6 AE      LDX *LDTND   ;and number of open files
DA1F- CA          DCLLP  DEX        ;if all files done
DA20- 30 0E      BMI DCLBYE   ;exit
DA22- DD 5B 02      CMP FAT,X    ;else if file on current device
DA25- D0 F8      BNE DCLLP     ;and close that file
DA27- B0 51 02      LDA LAT,X    ;get file number
DA2A- 20 E7 F2      JSR CLOS10   ;and close that file
DA2D- B8          CLV           ;why not a JMP?
DA2E- 50 EB      BVC DCLALL   ;and do next file
DA30- 60          DCLBYE RTS       ;exit
*****  

        ;*                         *
        ;* Set up RECORD parameters in  *
        ;* disk commandbuffer at TBUFF.  *

```

```

        ;*
        ;*****
        DA31- A5 D2    BOBREC LDA *LA      ;get current file number
        DA33- 20 C1 F2          JSR JLTLK   ;search file table for it
        DA36- F0 05          BEQ DRECG   ;if not found
        DA38- A0 17          LDY #$17    ;point to FILE NOT OPEN
        DA3A- 4C AF F5          JMP ERMSG   ;give error and abort files
        DA3D- 20 CD F2    DRECG  JSR JZ100  ;else load file parameters
        DA40- 20 E1 DB          JSR OLDCLR  ;clear DS$
        DA43- A9 50    DREBLD LDA #'P'   ;move command letter
        DA45- 80 53 03          STA TBUFF  ;to commandstring
        DA48- A5 D3    LDA *SA     ;move secondary address
        DA4A- 80 54 03          STA TBUFF+1 ;to command string
        DA4D- A5 11    LDA *LINNUM  ;get record number lo
        DA4F- 80 55 03          STA TBUFF+2 ;in commandstring
        DA52- A5 12    LDA *LINNUM+1 ;get hi
        DA54- 80 56 03          STA TBUFF+3 ;also
        DA57- AD 3A 03          LDA POS    ;get byte position
        DA5A- 80 57 03          STA TBUFF+4 ;and complete string
        DA5D- A2 05    LDX #$05    ;get length of string
        DA5F- 20 4C DC          JSR TRANR   ;set pointer to string
        DA62- 4C 9B DA          JMP TRANS1  ;and send string to disk
        ;*****
        ;*          *
        ;* Perform COLLECT.          *
        ;*          *
        ;* (Called from Kernal: $FFA2). *
        ;*          *
        ;*****
        DA65- 20 68 DC    COLECT JSR DOSPAR ;read parameters
        DA68- 20 18 D8          JSR CHK3    ;check if device number only
        DA6B- 20 1B DA    DCOLLE JSR DCLALL ;close all files on current device
        DA6E- A0 14    LDY #d84d-d839 ;get offset in table
        DA70- A2 01    LDX #1      ;get length (no drive#)
        DA72- AD 3E 03          LDA PARCHK ;if
        DA75- 29 10    AND #$10    ;source drive# given
        DA77- F0 01    BEQ DCOLLO
        DA79- E8      INX      ;adapt length
        DA7A- 8A    DCOLLO TXA    ;get it in A
        DA7B- 4C 98 DA          JMP TRANS   ;build and send commandstring
        ;*****
        ;*          *
        ;* Perform BACKUP.          *
        ;*          *
        ;* (Called from Kernal: $FFA5). *
        ;*          *
        ;*****
        DA7E- 20 68 DC    BACKUP JSR DOSPAR ;read parameters
        DA81- 29 30    AND #$30    ;if both source and destination
        DA83- C9 30    CMP #$30    ;drive#'s given
        DA85- F0 03    BEQ BBACK   ;go on
        DA87- 4C 00 BF    BERRO  JMP SNERR  ;else give SYNTAX ERROR
        DA8A- AD 3E 03    BBACK  LDA PARCHK ;get flags
        DA8D- 29 C7    AND #$C7    ;if other parameters as well
        DA8F- D0 F6    BNE BERRO  ;give SYNTAX ERROR
        DA91- 20 1B DA    DBACKU JSR DCLALL ;else clear DS$
        DA94- A0 16    LDY #d84f-d839 ;get offset
        DA96- A9 04    LDA #d853-d84f ;and length, then:
        ;*****
        ;*          *
        ;* Send commandstring to disk.  *
        ;*          *
        ;* On entry, Y holds offset in  *
        ;* dummy command table, and A   *

```

```

        ;* holds length of the dummy      *
        ;* command.                      *
        ;*                                *
        ;*****                           *
DA98- 20 FA DB  TRANS  JSR SENDP      ;build commandstring from input
DA9B- A9 6F      TRANS1 LDA #$6F      ;get secondary address 15
DA9D- 85 D3      STA *SA          ;in current secondary address
DA9F- 20 D5 F0      JSR LISTN      ;send listen
DAA2- A5 D3      LDA *SA          ;get secondary address
DAA4- 4C B4 F4      JMP OPENIB     ;and send string to disk
        ;*****                           *
        ;*                                *
        ;* Perform COPY.                 *
        ;*                                *
        ;* (Called from Kernal: $FFA8). *
        ;*                                *
        ;*****                           *
DAA7- 20 68 DC  COPY   JSR DOSPAR    ;read parameters
DAAA- 29 30      AND #$30      ;if only source and destination
DAAC- C9 30      CMP #$30      ;drive numbers given
DAAE- D0 07      BNE COPY2
DAB0- AD 3E 03      LDA PARCHK    ;get flags again
DAB3- 29 C7      AND #$C7      ;if no filenames given
DAB5- F0 09      BEQ COPY3      ;do copy entire drive
DAB7- AD 3E 03      COPY2        LDA PARCHK    ;else get flags
DABA- 20 10 D8      JSR CHK4      ;test if no filenumber
DABD- AD 3E 03      COPCON       LDA PARCHK    ;get flags again
DAC0- A0 1A      COPY3        LDY #d853-d839 ;get offset
DAC2- A9 08      LDA #d85b-d853 ;and length
DAC4- 4C 98 DA      JMP TRANS     ;build and send commandstring
        ;*****                           *
        ;*                                *
        ;* Perform CONCAT.              *
        ;*                                *
        ;* (Called from Kernal: $FF93). *
        ;*                                *
        ;*****                           *
DAC7- 20 68 DC  CONCAT  JSR DOSPAR    ;read parameters
DACA- 20 10 D8      JSR CHK4      ;test if no at-sign, no write and no fi
DACD- A0 22      LDY #d85b-d839 ;get offset in table
DACE- A9 0C      LDA #d867-d85b ;and length
DAD1- 4C 98 DA      JMP TRANS     ;build and send commandstring
        ;*****                           *
        ;*                                *
        ;* Move filename into command  *
        ;* string.                     *
        ;*                                *
        ;* On entry, X holds offset in *
        ;* commandstring built so far. *
        ;* $D1 should hold length, ($DA) *
        ;* should hold pointer to file- *
        ;* name to be moved.           *
        ;*                                *
        ;*****                           *
DAD4- A5 D1      RSFN   LDA *FNLEN    ;save filename length
DAD6- 8D 3A 03      STA FNLEN2   ;in $033A
DAD9- A5 DA      LDA *FNADR    ;move pointer to
DADB- 85 FD      STA *FNADR2   ;filename
DADD- A5 DB      LDA *FNADR+1  ;to
DADF- 85 FE      STA *FNADR2+1 ;(FNADR2)
DAE1- 98         RDFN   TYA          ;save Y
DAE2- 48         RDFN   PHA          ;on stack
DAE3- AC 3A 03      LDY FNLEN2   ;get length of filename
DAE6- F0 10      BEQ RDRTO     ;if zero, exit

```

```

DAE8- A0 00      LDY #$00      ;else get offset zero
DAEA- B1 FD      RDMOV   LDA (FNADR2),Y ;get character of name
DAEC- 9D 53 03    STA TBUFF,X ;move to commandstring
DAEF- E8          INX      ;point both indexes
DAF0- C8          INY      ;to next character
DAF1- CC 3A 03    CPY FNLEN2 ;if not all characters moved
DAF4- D0 F4      BNE RDMOV ;loop
DAF6- F0 01      BEQ RDRT1 ;else exit
DAF8- CA          RDRTO   DEX      ;if no filename, point to previous char
DAF9- 68          RDRT1   PLA      ;get Y
DAFA- A8          TAY      ;back from stack
DAFB- 38          SEC      ;set carry
DAFC- 60          RTS      ;and exit
;*****
;*
;* Move disk ID to command-
;* string.
;*
;* On entry, X holds offset in
;* commandstring built so far.
;*
;*****
DAFD- AD 3F 03    RID     LDA DISKID ;get 1st ID character
DB00- 9D 53 03    STA TBUFF,X ;move to command string
DB03- E8          INX      ;point to next character
DB04- AD 40 03    LDA DISKID+1 ;get 2nd ID character
DB07- 9D 53 03    STA TBUFF,X ;move to commandstring
DB0A- E8          INX      ;point to next character
DB0B- 8A          TXA      ;get offset in A
DB0C- 60          RTS      ;and exit
.FI "BASIC4.00.M12"

```

15E3 3411-49F4 BASIC4.00.M12

```

;name BASIC4.00.M12
;*****
;*
;* Perform DSAVE.
;*
;* (Called from Kernal: $FFAE).
;*
;*****
DB00- 20 68 DC    DSAVE   JSR DOSPAR ;read parameters
DB10- 20 0B D8    DSAVE   JSR CHK2  ;check if filename given
DB13- 29 66        AND #$66  ;if write, dest. drive#, file# or
DB15- F0 03        BEQ DSAVE2 ;destination filename,
DB17- 4C 00 BF    DSAVE2  JMP SNERR ;give SYNTAX ERROR
DB1A- A0 02        DSAVE2  LDY #d83b-d839 ;get offset in table
DB1C- AD 3E 03    DSAVE2  LDA PARCHK ;get flags
DB1F- 29 80        DSAVE2  AND #$80  ;if at-sign given
DB21- F0 0F        SAVLD1  BEQ SAVLD1
DB23- A9 40        SAVLD0  LDA #'@' ;get at-sign
DB25- 80 53 03    SAVLD0  STA TBUFF ;in file name
DB28- A2 01        LDX #1   ;get offset for at-sign
DB2A- A9 03        LDA #d83e-d83b ;get length of dummy
DB2C- 20 FC DB    JSR SENDP1 ;build commandstring
DB2F- 4C E0 F6    JMP SV3   ;and go SAVE
DB32- A9 03        SAVLD1  LDA #d83e-d83b ;get length of dummy
DB34- 20 FA DB    JSR SENDP1 ;build commandstring
DB37- 4C E0 F6    JMP SV3   ;and go SAVE
;*****
;*
;* Perform DLOAD.
;
```

```

        ;* (Called from Kernal: $FFB1). *
        ;*
        ;*****
DB3A- 20 68 DC DLOAD JSR DOSPAR      ;read parameters
DB3D- 20 0B D8          JSR CHK2       ;check if filename given
DB40- 29 E6          AND #$E6       ;if further only source drive#, device#
DB42- F0 03          BEQ DLOAD2     ;filename given, go on
DB44- 4C 00 BF DLOERR JMP SNERR      ;else give SYNTAX ERROR
DB47- A0 02 DLOAD2 LDY #d83b-d839   ;get position of dummy
DB49- A9 03          LDA #d83e-d83b  ;and length
DB4B- 20 FA DB          JSR SENDP     ;build commandstring
DB4E- A9 00          LDA #$00       ;set LOAD/VERIFY
DB50- 85 90          STA *VERCK    ;to LOAD
DB52- 4C 08 F4          JMP LOADNP    ;and go LOAD
        ;*****
        ;*
        ;* Perform RENAME. *
        ;*
        ;* (Called from Kernal: $FFB7). *
        ;*
        ;*****
DB55- 20 68 DC RENAME JSR DOSPAR      ;read parameters
DB58- 20 24 D8          JSR CHK5       ;check if two file names given
DB5B- 29 E4          AND #$E4       ;further only 1 drive# and device#
DB5D- D0 E5          BNE DLOERR     ;allowed, if more, SYNTAX ERROR
DB5F- A0 2E          LDY #d867-d839  ;else get offset
DB61- A9 08          LDA #d86f-d867  ;and length of dummy command
DB63- 4C 98 DA          JMP TRANS     ;build and send commandstring
        ;*****
        ;*
        ;* Perform SCRATCH. *
        ;*
        ;* (Called from Kernal: $FFBA). *
        ;*
        ;*****
DB66- 20 68 DC scrtch JSR DOSPAR      ;get parameters
        ;NOTE on lower case label: SCRTCH used for NEW command
DB69- 20 04 D8          JSR CHK1       ;check if filename and drive# given
DB6C- 20 9E DB DSCRAT JSR RUSURE     ;do 'ARE YOU SURE?'
DB6F- B0 27          BCS NUMBYE    ;if not sure, exit
DB71- A0 36          LDY #d86f-d839  ;else get offset
DB73- A9 04          LDA #CATLOG-d86f ;and length
DB75- 20 98 DA          JSR TRANS     ;build and send commandstring
DB78- 20 99 DB NUMSCR JSR DDIREC    ;check if direct mode
DB7B- D0 1B          BNE NUMBYE    ;if not, exit
DB7D- 20 91 09          JSR ERRCH1    ;else read DS$
DB80- A9 00          LDA #$0D       ;get a carriage return
DB82- 20 02 E2          JSR PRT      ;print it on screen
DB85- A0 00          LDY #$00       ;get offset zero
DB87- B1 0E          NUMLP LDA (DSDESC+1),Y ;get character of DS$
DB89- C9 00          CMP #$00       ;why is this?
DB8B- F0 06          BEQ NUMPRT    ;if end of DS$, print return, exit
DB8D- 20 02 E2          JSR PRT      ;else print to screen
DB90- C8              INY           ;point to next character
DB91- D0 F4          BNE NUMLP    ;and loop
DB93- A9 00          NUMPRT LDA #$0D    ;get a carriage return
DB95- 20 02 E2          JSR PRT      ;print to screen
DB98- 60          NUMBYE RTS      ;and exit
        ;*****
        ;*
        ;* Test if in direct mode. *
        ;* If in direct mode, Z is set. *
        ;*

```

```

*****+
DB99- A5 78  DDIREC LDA *TXTPTR+1 ;get CHRGET's pointer hi
DB9B- C9 02      CMP #H,BUF ;set Z if it points to input buffer
DB9D- 60  DXCRO RTS ;and exit
*****+
;*
;* Print ARE YOU SURE? and get *
;* input. If input equals Y or *
;* YES, clear carry, else set *
;* it. *
;*
*****+
DB9E- 20 99 DB  RUSURE JSR DDIREC ;test if in direct mode
DBA1- D0 32      BNE ANSYES ;if not, exit with C=0
DBA3- A0 B6  RUSUR1 LDY #MSG30-MSG1 ;point to 'ARE YOU SURE?'
DBA5- 20 85 F1      JSR MSG ;print it to screen
DBA8- 20 A6 F2      JSR CLRCHN ;abort all files
DBAB- 20 15 F2  RUSUR2 JSR BASIN ;get a character
DBAE- C9 59      CMP #'Y' ;if not Y
DBB0- D0 19      BNE ANSNO ;test for return
DBB2- 20 15 F2      JSR BASIN ;if Y, get another character
DBB5- C9 0D      CMP #$0D ;if return (Y return typed in)
DBB7- F0 1C      BEQ ANSYES ;exit with C=0
DBB9- C9 45      CMP #'E' ;if not E
DBBB- D0 0E      BNE ANSNO ;test for return
DBBD- 20 15 F2      JSR BASIN ;if E, get next character
DBC0- C9 53      CMP #'S' ;if not S
DBC2- D0 07      BNE ANSNO ;test for return
DBC4- 20 15 F2      JSR BASIN ;if S, get next character
DBC7- C9 0D      CMP #$0D ;if return (YES return typed in)
DBC9- F0 0A      BEQ ANSYES ;exit with C=0
DBCB- C9 00  ANSNO CMP #$0D ;if return typed
DBCD- 38      SEC ;set carry
DBCE- F0 06      BEQ ANSBYE ;exit with C=1
DBD0- 20 15 F2      JSR BASIN ;else get next character
DBD3- D0 F6      BNE ANSNO. ;and wait for return
DBD5- 18  ANSYES CLC ;clear carry
DBD6- 60  ANSBYE RTS ;and exit
*****+
;*
;* Print ? BAD DISK if in direct *
;* mode. *
;*
*****+
DBD7- 20 99 DB  BADDIS JSR DDIREC ;check if direct mode
DBDA- D0 FA      BNE ANSBYE ;if not, exit
DBDC- A0 C5      LDY #MSG31-MSG1 ;else point to ? BAD DISK
DBDE- 4C 85 F1      JMP MSG ;and print it
*****+
;*
;* Clear DS$. *
;* (Set its backpointer invalid) *
;*
*****+
DBE1- 98  OLDCLR TYA ;save Y
DBE2- 48      PHA ;on stack
DBE3- A5 0D      LDA *DSDESC ;if length of DS$
DBE5- F0 0A      BEQ OLDCCL1 ;is zero, clear ST only
DBE7- A0 28      LDY #40 ;else get length of DS$
DBE9- 98      TYA ;in A
DBEA- 91 0E      STA (DSDESC+1),Y ;set backpointer lo
DBEC- C8      INY ;point to backpointer hi.
DBED- A9 FF      LDA #$FF ;get invalid hi byte
DBEF- 91 0E      STA (DSDESC+1),Y ;and complete backpointer

```

```

DBF1- A9 00      OLDCL1 LDA #$00      ;then clear
DBF3- 85 96      STA *CSTAT      ;the status byte
DBF5- 85 0D      STA *DSDESC      ;and set length of DS$ to zero
DBF7- 68          PLA           ;get Y
DBF8- A8          TAY           ;from stack
DBF9- 60          RTS           ;and exit
.FI "BASIC4.00.M13"

```

1A10 3411-4E21 BASIC4.00.M13

```

;name BASIC4.00.M13
;*****
;*          *
;* Build disk command string      *
;* from parameters stored by the *
;* the routine at $DC68.          *
;* The string starts at TBUFF,    *
;* and $D1 holds its length on   *
;* on exit.                      *
;*          *
;*****


DBFA- A2 00      SENDP  LDX #$00      ;get initial index in disk command
DBFC- 8D 41 03    SENDP1 STA count     ;save length of dummy command
DBFF- 20 E1 DB    JSR OLDCLR      ;clear DS$
DC02- CE 41 03    SENDP2 DEC count     ;count dummy command characters
DC05- 30 45      BMI TRANR      ;if all done, exit
DC07- C8          INY           ;else point to next dummy cmd character
DC08- B9 38 D8    LDA d839-1,Y  ;get dummy command byte
DC0B- 10 39      BPL RPLCE      ;if MSB clear, move ASCII char.
DC0D- C9 F1      RXFN1  CMP #$F1      ;if code for source file name
DC0F- D0 03      BNE RXFN2      ;move the filename
DC11- 20 D4 DA    JSR RSFN       ;if code for destination filename
DC14- C9 F2      RXFN2  CMP #$F2      ;move destination filename
DC16- D0 03      BNE RXREC      ;if code for r/w
DC18- 20 E1 DA    JSR RDFN       ;move write character
DC1B- C9 E0      RXREC  CMP #$E0      ;and move
DC1D- D0 05      BNE RXID       ;if code for disk ID
DC1F- AD 3D 03    LDA LREC1      ;move ID into place
DC22- D0 22      BNE RPLCE      ;if code for file type
DC24- C9 D0      RXID   CMP #$D0      ;get source drive#
DC26- D0 03      BNE RXWRT      ;and move to disk command
DC28- 20 FD DA    JSR RID        ;if not code for destination drive#
DC2B- C9 E1      RXWRT  CMP #$E1      ;process next character
DC2D- D0 05      BNE RXD1       ;else get destination drive #
DC2F- 20 57 DC    JSR RWRT       ;convert to ASCII
DC32- D0 12      BNE RPLCE      ;and add to disk command
DC34- C9 D1      RXD1   CMP #$C1      ;point to next disk cmd character
DC36- D0 05      BNE RXD2       ;and process next dummy
DC38- AD 3B 03    LDA DRIVE1     ;save offset in real command
DC3B- 10 07      BPL RXDD       ;point
DC3D- C9 D2      RXD2   CMP #$D2      ;($DA)
DC3F- D0 C1      BNE SENDP2     ;to
DC41- AD 3C 03    LDA DRIVE2     ;start of command string
DC44- D9 30      RXDD   ORA #$30      ;and exit
DC46- 9D 53 03    RPLCE  STA TBUFF,X
DC49- E8          INX           ;name BASIC4.00.M13
DC4A- D0 B6      BNE SENDP2     ;*****
DC4C- 86 D1      TRANR  STX *FNLEN
DC4E- A9 53      LDA #L,TBUFF
DC50- 85 DA      STA *FNADR
DC52- A9 03      LDA #H,TBUFF
DC54- 85 DB      STA *FNADR+1
DC56- 60          RTS           ;name BASIC4.00.M13

```

```

;*****
;*
;* Move file type to command-
;* string.
;*
;* If W specified, sequential
;* file is assumed, else a re-
;* tive file is set, with an L
;* parameter.
;*
;*****
DC57- AD 30 03 RWRT LDA LREC1      ;get record length or type
DC5A- F0 04          BEQ RWRTL     ;if none, get S
DC5C- A9 4C          LDA #'L'      ;else get L
DC5E- D0 07          BNE RWRTS    ;and exit
DC60- A9 53          RWRTL       LDA #'S'      ;get the S for sequential
DC62- 8D 30 03          STA LREC1    ;as default file type
DC65- A9 57          LDA #'W'      ;and a W for write
DC67- 60          RWRTS       RTS         ;and exit
;name BASIC4.0D.M13
;*****
;*
;* Read parameters of disk
;* command and store their in-
;* formation in various parts
;* of cassette buffer #2.
;*
;*****
DC68- A2 00          DOSPAR      LDX #$00      ;clear:
DC6A- 8E 3E 03          STX PARCHK   ;syntax check mask
DC6D- 86 D2          STX *LA       ;current file number
DC6F- 8E 30 03          STX LREC1    ;record length/file type
DC72- 8E 3B 03          STX DRIVE1   ;source drive is default zero
DC75- 8E 3C 03          STX DRIVE2   ;so is destination drive
DC78- 86 D1          STX *FNLEN   ;clear real command string length
DC7A- 8E 3A 03          STX POS      ;clear length of filename/byte position
DC7D- 8E 3F 03          STX DISKID   ;clear disk ID, 1st character
DC80- A2 08          LDX #8       ;get default device number
DC82- 86 D4          STX *FA      ;in current device number
DC84- 20 76 00          JSR CHRGOT   ;get current character
DC87- F0 7A          BEQ DONE1    ;if end of statement, get mask and exit
DC89- C9 23          PARSEL      CMP #'#'
DC8B- F0 30          BEQ LOGADR   ;if sign for file number
DC8D- C9 57          CMP #'W'
DC8F- F0 47          BEQ RECLEM   ;go handle file number
DC91- C9 4C          CMP #'L'
DC93- F0 43          BEQ RECLEM   ;if sign for write to seq. file
DC95- C9 52          CMP #'R'
DC97- D0 06          BNE PARNXT  ;handle it
DC99- 20 70 00          JSR CHRGET   ;if sign for record length
DC9C- 4C 96 DD          JMP DELIM1   ;handle it
DC9F- C9 44          PARNXT     CMP #'D'
DCA1- F0 72          BEQ DRV1     ;if sign for read seq. file
DCA3- C9 91          CMP #ONTK   ;go read drive number
DCA5- F0 62          BEQ ON1      ;if token for ON
DCA7- C9 55          CMP #'U'
DCA9- F0 64          BEQ UNIT1   ;read device number
DCAB- C9 49          CMP #'I'
DCAD- D0 03          BNE NEXT7  ;if sign for device number
DCAF- 4C 37 DD          JMP IDENT   ;read disk ID and loop
DCB2- C9 22          NEXT7      CMP #'"'
DCB4- F0 50          BEQ NAMEL1   ;if start of filename
DCB6- C9 28          CMP #'('
DCB8- F0 4C          BEQ NAMEL1   ;go read filename
;if open parenthesis
;go read expression

```

```

DCBA- 4C 00 BF  SNERL  JMP SNERR      ;else give SYNTAX ERROR and restart
;*****
;*                                         *
;* Handle filenumber.                  *
;* If valid filenumber found,        *
;* it is stored in $02, and flag   *
;* in PARCHK is set.                *
;*                                         *
;*****
DCBD- AD 3E 03  LOGADR LDA PARCHK    ;if file number
DCC0- 29 04      AND #$04          ;already read (2nd filenumber)
DCC2- 00 F6      BNE SNERL       ;give SYNTAX ERROR
DCC4- 20 87 DE  JSR GETVAL       ;else read file number in X
DCC7- EO 00      CPX #0           ;if it is zero
DCC9- FO 69      BEQ QTYER2      ;give ILLEGAL QUANTITY ERROR
DCCB- 86 D2      STX *LA          ;else store it in current file number
DCCD- AD 3E 03  LDA PARCHK       ;get syntax flags
DCD0- 09 04      ORA #$04          ;set bit 2
DCD2- 8D 3E 03  STA PARCHK       ;and save flags again
DCD5- 4C 96 DD  JMP DELIM1       ;then test for comma and loop
;*****
;*                                         *
;* Handle L and W parameters.      *
;* If W, handle next character,   *
;* if L store record length in   *
;* LREC1.                         *
;*                                         *
;*****
DCD8- AA          RECLEM TAX       ;save character in X
DCD9- AD 3E 03  LDA PARCHK       ;get syntax flags
DCDC- 29 40      AND #$40          ;get bit 6
DCDE- 00 DA      BNE SNERL       ;if L or W found already, SYNTAX ERROR
DCEO- EO 57      CPX #'W'         ;if W parameter
DCE2- 00 06      BNE RECOO        ;get next character
DCE4- 20 70 00  JSR CHRGET      ;and add flag in syntax flags
DCE7- 4C F8 DC  JMP RECON        ;if L parameter, read record length in
DCEA- 20 87 DE  RECOO  JSR,GETVAL ;if length is zero
DCED- EO 00      CPX #0           ;give ILLEGAL QUANTITY ERROR
DCEF- FO 43      BEQ QTYER2      ;if length exceeds sector capacity
DCF1- EO FF      CPX #255         ;give ILLEGAL QUANTITY ERROR
DCF3- FO 3F      BEQ QTYER2      ;else store record length
DCF5- 8E 3D 03  STX LREC1        ;get syntax flags
DCF8- AD 3E 03  RECON  LDA PARCHK ;set bit 6
DCF9- 09 40      ORA #$40          ;and store flags
DCFD- 8D 3E 03  STA PARCHK       ;then check for comma
DD00- 4C 96 DD  JMP DELIM1       ;*****
;*                                         *
;* Intermediate JuMP to be        *
;* reached via relative          *
;* branches. It reads the syntax *
;* flags, then exits.            *
;*                                         *
;*****
DD03- 4C 23 DE  DONE1  JMP DONE       ;read flags and exit
;*****
;*                                         *
;* Intermediate JuMP to be        *
;* reached via relative          *
;* branches. It handles a file   *
;* name.                          *
;*                                         *
;*****
DD06- 4C 60 DD  NAME1  JMP NAME1      ;handle filename and loop

```

```

;*****
;*
;* Intermediate JuMP to be      *
;* reached via relative        *
;* branches. It handles a device *
;* number. (ON read).          *
;*
;*****
DD09- 20 2C DE  ON1   JSR ON           ;handle device number
DD0C- 4C 96 DD       JMP DELIM1       ;and loop
.FI "BASIC4.0D.M14"

```

1786 3411-4B97 BASIC4.0D.M14

```

;name BASIC4.0D.M14
;*****
;*
;* Intermediate JuMP to be      *
;* reached via relative        *
;* branches. It handles a device *
;* number. (U read).          *
;*
;*****
DD0F- 20 33 DE  UNIT1  JSR de33         ;handle device number after U
DD12- 4C 96 DD       JMP DELIM1       ;and loop
;*****
;*
;* Handle source drive number.  *
;* The drive number is left in  *
;* $033B and $033C.            *
;*
;*****
DD15- AD 3E 03  DRV1   LDA PARCHK      ;get syntax flags
DD18- 29 10          AND #$10        ;if drive number given
DD1A- D0 9E          BNE SNERL       ;give SYNTAX ERROR
DD1C- 20 87 DE       JSR GETVAL      ;get number in X
DD1F- E0 02          CPX #2         ;if number too high
DD21- B0 11          BCS QTYER2      ;give ILLEGAL QUANTITY ERROR
DD23- 8E 3B 03       STX DRIVE1     ;else store it as drive number (without
DD26- 8E 3C 03       STX DRIVE2     ;and as drive number with @
DD29- AD 3E 03       LDA PARCHK      ;get syntax flags
DD2C- 09 10          ORA #$10        ;set bit 4
DD2E- 8D 3E 03       STA PARCHK      ;and store flags
DD31- 4C 96 DD       JMP DELIM1      ;then check for comma
;*****
;*
;* ILLEGAL QUANTITY ERROR exit. *
;*
;*****
DD34- 4C 27 DE  QTYER2  JMP QTYERR      ;give ILLEGAL QUANTITY ERROR
;*****
;*
;* Handle I parameter.          *
;* The disk ID read is stored in *
;* $033F and $0340.            *
;*
;*****
DD37- AD 30 03  IDENT   LDA DIDCHK      ;if record length or
DD3A- 29 FF          AND #$FF        ;ID given,
DD3C- F0 03          BEQ IDCN        ;IDCON
DD3E- 4C 00 BF       JMP SNERR       ;give SYNTAX ERROR
DD41- A0 00          IDCN          LDY #$00        ;get offset zero
DD43- A2 00          IDCN          LDX #$00        ;and index zero

```

```

DD45- E6 77      NEXT3  INC *TXTPTR      ;increment CHRGET's pointer
DD47- D0 02      BNE NEXT4      ;if page crossed
DD49- E6 78      INC *TXTPTR+1    ;adapt hi byte also
DD4B- B1 77      NEXT4  LDA (TXTPTR),Y  ;get next character
DD4D- 9D 3F 03   STA DISKID,X    ;and move ID character
DD50- E8          INX             ;adapt index
DD51- E0 02      CPX #2         ;if not two characters read
DD53- 90 F0      BCC NEXT3      ;read next
DD55- A9 FF      LDA #$FF       ;get flag ID read
DD57- 8D 3D 03   STA DIDCHK     ;in 'record length'
DD5A- 20 70 00   JSR CHRGET     ;get next character
DD5D- 4C 96 00   JMP DELIM1     ;and test for comma
;***** *
;* Read source filename and move *
;* it to $0342 and further if   *
;* valid.                      *
;***** *
;***** *
DD60- AD 3E 03   NAME1  LDA PARCHK    ;get syntax flags
DD63- 29 01      AND #$01       ;clear Z if a name was read already
DD65- 20 49 DE   JSR NEWNAM     ;read name and check length
DD68- 85 D1      STA *FNLEN      ;save length of name
DD6A- 8D 41 03   STA count       ;in counter
DD6D- A9 42      LDA #$42       ;get pointer
DD6F- 85 DA      STA *FNADR     ;to destination
DD71- A9 03      LDA #$03       ;of filename
DD73- 85 DB      STA *FNADR+1   ;in ($DA)
DD75- A0 00      LDY #$00       ;get offset zero
DD77- B1 1F      LOOP6  LDA (INDEX),Y  ;get character of name
DD79- C0 00      CPY #$00       ;if 1st character
DD7B- D0 08      BNE NAMCON    ;is at-sign (save with replace)
DD7D- C9 40      CMP #'@'       ;adapt pointer to name
DD7F- D0 04      BNE NAMCON    ;and set correct length
DD81- E6 DA      INC *FNADR     ;move character into place
DD83- C6 D1      DEC *FNLEN
DD85- 99 42 03   NAMCON STA TBUF2,Y
DD88- C8          INY             ;point to next
DD89- CC 41 03   CPY count      ;if not all done
DD8C- 90 E9      BCC LOOP6     ;loop
DD8E- AD 3E 03   LDA PARCHK    ;else get syntax flags
DD91- 09 01      ORA #$01       ;set bit zero
DD93- 8D 3E 03   STA PARCHK    ;and store flags again, then:
;***** *
;* Get current character and   *
;* test for special signs.    *
;* (Loop entry after 1st param-*
;* ter is read and handled).  *
;***** *
;***** *
DD96- 20 76 00   DELIM1 JSR CHRGOT    ;get current character
DD99- D0 03      BNE NXXX       ;if end of statement
DD9B- 4C 03 00   JMP DONE1      ;get flags and exit
DD9E- C9 2C      NXXX  CMP #','
DDA0- D0 06      BNE NEXT6     ;if comma
DDA2- 20 70 00   JSR CHRGET     ;skip it
DDA5- 4C 89 DC   JMP PARSEL     ;and loop
DDA8- C9 91      NEXT6  CMP #ONTK    ;if token for ON
DDAA- D0 03      BNE NEXT6A   ;read U and device number
DDAC- 4C 09 00   JMP ON1        ;if not token for TO
DDAF- C9 A4      NEXT6A CMP #TOTK    ;give SYNTAX ERROR
DDB1- F0 03      BEQ PARSE2
DDB3- 4C 00 BF   SNER8  JMP SNERR

```

```

;*****
;*
;* Token for TO found.
;* Test for special signs following it.
;*
;*****
DDB6- 20 70 00 PARSE2 JSR CHRGET      ;get next character
DDB9- C9 44 CMP #'D'          ;if sign for drive number
DDBB- F0 13 BEQ DRV2        ;handle destination drive number
DDBD- C9 91 CMP #ONTK       ;if token for ON
DDBF- F0 2B BEQ ON2        ;handle it
DDC1- C9 55 CMP #'U'         ;if sign for device number
DDC3- F0 2D BEQ UNIT2       ;handle device number
DDC5- C9 22 CMP #'"'        ;if start of filename
DDC7- F0 2F BEQ NAME2       ;handle filename
DDC9- C9 28 CMP #'('        ;if open parenthesis
DDCB- F0 2B BEQ NAME2       ;handle expression
DDCD- 4C 00 BF SNER2        JMP SNERR        ;else give SYNTAX ERROR
;*****
;*
;* Handle drive number after TO. *
;*
;*****
DD00- AD 3E 03 DRV2        LDA PARCHK      ;if destination drive #
DD03- 29 20 AND #$20        ;read already
DD05- D0 F6 BNE SNER2       ;give SYNTAX ERROR
DD07- 20 87 DE JSR GETVAL     ;else read drive number in X
DDA- E0 02 CPX #2          ;if invalid number
DDC- B0 49 BCS QTYERR       ;give ILLEGAL QUANTITY ERROR
DDDE- 8E 3C 03 STX DRIVE2    ;else store as destination drive number
DDE1- AD 3E 03 LDA PARCHK      ;get syntax flags
DDE4- 09 20 ORA #$20        ;set bit 5
DDE6- 8D 3E 03 STA PARCHK      ;and store flags
DDE9- 4C OF DE JMP DELIM2      ;then
;*****
;*
;* Handle ON after TO.      *
;*
;*****
DDEC- 20 2C DE ON2        JSR ON          ;read U and device number
DDEF- 4C OF DE JMP DELIM2      ;and loop
;*****
;*
;* Handle U after TO.      *
;*
;*****
DDF2- 20 33 DE UNIT2      JSR de33        ;read device number
DDF5- 4C OF DE JMP DELIM2      ;and loop
;*****
;*
;* Handle filename or expression *
;* for it after TO.          *
;*
;*****
DDF8- AD 3E 03 NAME2      LDA PARCHK      ;get syntax flags
DDFB- 29 02 AND #$02        ;clear Z if 2nd name read already
DDFD- 20 49 DE JSR NEWNAM     ;read filename and test length
DE00- 8D 3A 03 STA FNLEN2     ;length of 2nd filename
DE03- 86 FD STX *FNADR2      ;and pointer to
DE05- 84 FE STY *FNADR2+1    ;stringtext in (FNADR2)
DE07- AD 3E 03 LDA PARCHK      ;get syntax flags
DE0A- 09 02 ORA #$02        ;set bit 1
DE0C- 8D 3E 03 STA PARCHK      ;and store flags

```

.FI "BASIC4.0D.M15"

130A '3411-47EB BASIC4.0D.M15

```

        ;name BASIC4.0D.M15
        ;*****
        ;*          *
        ;* Loopentry after a special      *
        ;* character after T0 is dealt   *
        ;* with.                         *
        ;*          *
        ;*****


DE0F- 20 76 00  DELIM2 JSR CHRGOT      ;get current character
DE12- F0 0F      BEQ DONE           ;if end of statement, exit
DE14- C9 2C      CMP #','
DE16- F0 9E      BEQ PARSE2        ;test for more parameters
DE18- C9 91      CMP #ONTK         ;if token for ON
DE1A- F0 D0      BEQ ON2            ;handle it
DE1C- C9 55      CMP #'U'           ;if sign for device number
DE1E- F0 D2      BEQ UNIT2         ;handle it
DE20- 4C 00 BF  SNER3  JMP SNERR       ;else give SYNTAX ERROR
        ;*****
        ;*          *
        ;* Get syntax flag ($033E) in A  *
        ;* and exit routine.             *
        ;*          *
        ;*****


DE23- AD 3E 03  DONE    LDA PARCHK      ;get syntax flags
DE26- 60          RTS              ;and exit
        ;*****
        ;*          *
        ;* ILLEGAL QUANTITY ERROR exit.  *
        ;*          *
        ;*****


DE27- A2 35      QTYERR   LDX #b242-ERRTAB ;point to ILLEGAL QUANTITY
DE29- 4C CF B3      JMP ERROR        ;give error (why not simply JMP, C373?)
        ;*****
        ;*          *
        ;* ON found, read U and device  *
        ;* number. Store device number  *
        ;* in $D4 and flag number found. *
        ;*          *
        ;*****


DE2C- 20 70 00  ON      JSR CHRGET      ;get next character
DE2F- C9 55      CMP #'U'           ;if it is not U
DE31- D0 ED      BNE SNER3        ;give SYNTAX ERROR, else:
        ;*****
        ;*          *
        ;* U found, read device number. *
        ;* Store device number in $D4   *
        ;* and flag number found.       *
        ;*          *
        ;*****


DE33- 20 87 DE  de33  JSR GETVAL       ;read device number in X
DE36- E0 20      CPX #32            ;if maximum number exceeded
DE38- B0 ED      BCS QTYERR        ;give ILLEGAL QUANTITY ERROR
DE3A- E0 03      CPX #3             ;if below 4
DE3C- 90 E9      BCC QTYERR        ;give ILLEGAL QUANTITY ERROR
DE3E- 86 D4      STX *FA            ;else set current device number
DE40- AD 3E 03      LDA PARCHK      ;get syntax flags
DE43- 09 08      ORA #$08          ;set bit 3
DE45- 8D 3E 03      STA PARCHK      ;and store flags
DE48- 60          RTS              ;and exit this subroutine

```

```

;*****
;*
;* Quotes or open parenthesis *
;* found. Read expression for *
;* filename. Exit with AYX hol-
;* ding descriptor of filename. *
;*
;*****

DE49- D0 D5    NEWNAM BNE SNER3      ;if filename given, SYNTAX ERROR
DE4B- 20 98 BD  JSR FRMEVL      ;else read expression
DE4E- 20 B5 C7  JSR FRESTR      ;throw away string
DE51- AA        TAX             ;get its length in X
DE52- C9 00     CMP #$00        ;if length is zero
DE54- F0 D1     BEQ QTYERR      ;give ILLEGAL QUANTITY ERROR
DE56- C9 12     CMP #18        ;if length exceeds 17
DE58- B0 1A     BCS ERRLEN      ;give STRING TOO LONG ERROR
DE5A- A0 00     LDY #$00        ;else get offset zero
DE5C- B1 1F     LDA (INDEX),Y   ;get 1st character of name
DE5E- C9 40     CMP #'@'        ;if at-sign
DE60- D0 0A     BNE LENCHK      ;get syntax flags
DE62- AD 3E 03  LDA PARCHK      ;if no @ read already
DE65- 29 80     AND #$80        ;flag @ read and exit
DE67- F0 10     BEQ NXXTS      ;else give SYNTAX ERROR
DE69- 4C 00 BF  JMP SNERR       ;no at-sign, get length again
DE6C- 8A        LENCHK TXA      ;if more than 16
DE6D- C9 11     CMP #17        ;give STRING TOO LONG
DE6F- B0 03     BCS ERRLEN      ;else exit
DE71- 4C 82 DE  JMP NXXS       ;point to STRING TOO LONG
DE74- A2 B0     ERRLEN LDX #b2bd-ERRTAB ;and give that error
DE76- 4C CF B3  JMP ERROR       ;get syntax flags
DE79- AD 3E 03  NXXTS LDA PARCHK ;set bit 7
DE7C- 09 80     ORA #$80        ;and store flags again
DE7E- 8D 3E 03  STA PARCHK      ;get length in A
DE81- 8A        NXXTS TXA       ;and pointer to name
DE82- A6 1F     NXXS LDX *INDEX  ;in YX
DE84- A4 20     LDY *INDEX+1    ;then exit
DE86- 60        RTS            ;*****
;*
;* Read number or expression for *
;* number in X.                  *
;*
;*****



DE87- 20 70 00  GETVAL JSR CHRGET      ;get next character
DE8A- D0 03     GTVL2 BNE CONT        ;if end of statement,
DE8C- 4C 00 BF  JMP SNERR       ;give SYNTAX ERROR
DE8F- 90 09     CONT  BCC NUMERC      ;else if character not numeric
DE91- 20 F2 BE  JSR CHKOPN      ;test for (
DE94- 20 D4 C8  JSR GETBYT      ;read number in X
DE97- 4C EF BE  JMP CHKCLS       ;and test for ), exit
DE9A- 4C D4 C8  NUMERC JMP GETBYT    ;if numeric, get number in X
;*****
;*
;* Checksum byte (over D-ROM).   *
;*
;*****



DE9D- F4        CKSMDO .BY $F4        ;checksum byte
;*****
;*
;* Patch for get DS$ from disk. *
;* Clear status byte and end DS$ *
;* with a zero byte.              *
;*
;*****
```

```

DE9E- 85 96    de9e STA *CSTAT      ;clear status byte
DEAD- 91 0E    STA (DSDESC+1),Y ;end DS$ with a zero terminator
DEA2- 88        DEY             ;point to last valid character
DEA3- 60        RTS             ;and exit
;*****  

;*  

;* Patch; new power up message:  

;*  

;* *** COMMODORE BASIC 4.0 ***  

;*  

;*****  

DEA4- 2A 2A 2A  PATCH2 .BY '*** COMMODORE BASIC 4.0 ***' $00 $00 $00  

DEA7- 20 43 4F  

DEAA- 40 4D 4F  

DEAD- 44 4F 52  

DEB0- 45 20 42  

DEB3- 41 53 49  

DEB6- 43 20 34  

DEB9- 2E 30 20  

DEBC- 2A 2A 2A  

DEBF- 0D 0D 0D  

;*****  

;*  

;* The rest of the ROM is filled *  

;* with $AA bytes.  

;*  

;* (Patch area 3).  

;*  

;*****  

.FI "BASIC4.00.M16"

```

1756 3411-4B67 BASIC4.00.M16

```

;name BASIC4.00.M16
;*****  

;*  

;* External labels.  

;*  

;*****  

USRPOK .DE $0000      ;USR despatch
INTEGR .DE $0003      ;one byte integer from QINT
TANSGN .DE $000C      ;flag sign of TAN
DSDESC .DE $000D      ;descriptor of DS$
CHANNL .DE $0010      ;current CMD file number
LINNUM .DE $0011      ;number read by LINGET
TEMPPT .DE $0013      ;descriptor stack pointer
LASTPT .DE $0014      ;bottom of descriptor stack
TEMPST .DE $0016      ;top of descriptor stack
INDEX .DE $001F       ;indirect index
TXTTAB .DE $0028      ;start of BASIC program
FRETOP .DE $0030      ;end of arrays
MEMSIZ .DE $0034      ;top of RAM
TEMPF3 .DE $0048      ;temporary FLP accu
FOUR6 .DE $0050       ;(unused) flag in garbage collect
JMPER .DE $0051       ;function despatch JMP
OLDOV .DE $0053       ;old overflow of FLP accu1
TEMPF1 .DE $0054       ;temporary FLP accu
TEMPF2 .DE $0059       ;temporary FLP accu
FACEXP .DE $005E       ;exponent of FLP accu1
FACHO .DE $005F       ;MSB mantissa of FLP accu1
FACMOH .DE $0060       ;upper middle mantissa of FLP accu1
FACMO .DE $0061        ;lower middle mantissa of FLP accu1
FACLO .DE $0062        ;LSB mantissa of FLP accu1

```

FACSGN .DE \$0063	;sign of FLP accu1
DEGREE .DE \$0064	;grade of polynome
BITS .DE \$0065	;bit counter for shift in
ARGEXP .DE \$0066	;exponent of FLP accu2
ARGHO .DE \$0067	;MSB mantissa of FLP accu2
ARGMOH .DE \$0068	;upper middle mantissa of FLP accu2
ARGMO .DE \$0069	;lower middle mantissa of FLP accu2
ARGLO .DE \$006A	;LSB mantissa of FLP accu2
ARGSGN .DE \$006B	;sign of FLP accu2
ARISGN .DE \$006C	;sign comparison of FLP accus
FACOV .DE \$006D	;overflow of FLP accu1
POLYPT .DE \$006E	;pointer to polynome constants
CHRGET .DE \$0070	;get next character from BASIC
CHRGOT .DE \$0076	;get current character from BASIC
TXTPTR .DE \$0077	;pointer to current character
RNDX .DE \$0088	;current RND number seed
CINV .DE \$0090	;current IRQ vector
ONTK .DE \$0091	;token for ON
CSTAT .DE \$0096	;status byte ST
VERCK .DE \$009D	;LOAD/VERIFY flag
TOTK .DE \$00A4	;token for TO
LDTND .DE \$00AE	;number of open files
DFLTO .DE \$00B0	;default output device number
WSW .DE \$00B3	;DOS: temp. save for file#
SAVX .DE \$00B4	;MLM: save for command number
TMPC .DE \$00B5	;MLM: temporary save
CNTDN .DE \$00BA	;DOS: save for device#
EAL .DE \$00C9	;end address lo for SAVE
EAH .DE \$00CA	;end address hi for SAVE
FNLEN .DE \$00D1	;current filename length
LA .DE \$00D2	;current file number
SA .DE \$00D3	;current secondary address
FA .DE \$00D4	;current device number
FNADR .DE \$00DA	;pointer to current filename
WRAP .DE \$00DE	;MLM: flag for address wrap around
TMPO .DE \$00FB	;MLM: start and current pointer
TMP2 .DE \$00FD	;MLM: end address
FNADR2 .DE \$00FD	;DOS: pointer to 2nd filename
STKEND .DE \$01FB	;top of machine stack
BUF .DE \$0200	;BASIC's input buffer
SAVNAM .DE BUF+9	;MLM: place for filename in S and L cmd
SP .DE \$0206	;MLM: save for stackpointer
YR .DE \$0205	;MLM: save for Y register in BRK
XR .DE \$0204	;MLM: save for X register in BRK
ACC .DE \$0203	;MLM: save for A register in BRK
FLGS .DE \$0202	;MLM: save for SR register in BRK
PCL .DE \$0201	;MLM: save for PC lo byte
PCH .DE \$0200	;MLM: save for PC hi byte
IN VH .DE \$0207	;MLM: saved IRQ vector hi byte
IN VL .DE \$0208	;MLM: saved IRQ vector lo byte
LAT .DE \$0251	;table of file numbers of open files
FAT .DE \$0258	;table of sec. addresses of open files
SAT .DE \$0265	;table of device numbers of open files
POS .DE \$033A	;DOS: byte position in RECORD
FNLEN2 .DE POS	;DOS: length of 2nd filename
DRIVE1 .DE \$033B	;DOS: source drive#
DRIVE2 .DE \$033C	;DOS: destination drive#
LREC1 .DE \$033D	;DOS: record.length or file type
DIDCHK .DE LREC1	;DOS: disk given
PARCHK .DE \$033E	;DOS: syntax check byte
DISKID .DE \$033F	;DOS: disk id (2 characters)
count .DE \$0341	;length of (dummy) command
TBUF2 .DE \$0342	;DOS: buffer for 2nd filename
TBUFF .DE \$0353	;DOS: command string buffer

USRCMD .DE \$03FA	;extension vector for MLM
RAMLOC .DE \$0400	;start of free RAM
ROMLOC .DE \$8000	;start of BASIC ROM area
ERRTAB .DE \$B200	;start of error message table
b242 .DE \$B242	;message: ILLEGAL QUANTITY
b2a2 .DE \$B2A2	;message: ILLEGAL QUANTITY
b2bd .DE \$B2BD	;message: STRING TOO LONG
ERROR .DE \$B3CF	;fatal errors
READY .DE \$B3FF	;BASIC warm start
SCRTCH .DE \$B5D4	;perform NEW
STROUT .DE \$BB1D	;print string from YA till zero byte
FRMEVL .DE \$B098	;evaluate any expression
CHKCLS .DE \$BEEF	;check for)
CHKOPN .DE \$BEF2	;check for (
CHKCOM .DE \$BEF5	;check for a comma
SYNCHR .DE \$BEF7	;check for the character in A
SNERR .DE \$BF00	;SYNTAX ERROR, restart
FCERR .DE \$C373	;ILLEGAL QUANTITY ERROR, restart
GETSPA .DE \$C610	;reserve space in memory for string
FRESTR .DE \$C7B5	;throw away string
GETBYT .DE \$C8D4	;read a number <255 in X
GETADR .DE \$C920	;convert FLP accu1 to a 2 byte integer
FADDH .DE \$C97F	;add .5 to FLP accu1
FSUB .DE \$C986	;load FLP accu2 and perform subtraction
FSUBT .DE \$C989	;perform subtraction
FADD .DE \$C99D	;perform addition
NORMAL .DE \$CA0D	;normalize FLP accu1
ZEROF1 .DE \$CA2F	;store exponent and sign in accu1
FONE .DE \$CAF2	;pointer to FLP constant 1.0000000
LOG .DE \$CB20	;perform LOG
FMULT .DE \$CB5E	;multiply FLP accu1 with FLP#
MLDEXP .DE \$CBEF	;add exponents
MLDVEX .DE \$CC0A	;set accu1 to zero if positive
FDIVF .DE \$CC3D	;load constant in FLPaccu1 and divide
FDIV .DE \$CC45	;perform division
MOVFM .DE \$CCD8	;load FLP accu1 from memory
MOV2F .DE \$CCFD	;store FLP accu1 in memory at \$59
MOV1F .DE \$CD00	;store FLP accu1 in memory at \$54
MOVMF .DE \$CD0A	;store FLP accu1 in memory
MOVFA1 .DE \$CD34	;copy FLP accu2 to FLP accu1
MOVAF .DE \$CD42	;round FLP accu1 and copy to FLP accu2
MOVEF .DE \$CD45	;copy FLP accu1 to FLP accu2
INCRND .DE \$CD59	;check for overflow
SIGN .DE \$CD61	;get sign in accu1 in A
FCOMP .DE \$CD91	;compare FLP accu1 with a FLP number
INT .DE \$CE02	;perform INT
LINPRT .DE \$CF83	;print AX in decimal
.FI "BASIC4.00.M17"	

0608 3411-3A19 BASIC4.00.M17

;name BASIC4.00.M17	
;*****	
;*	
;* External labels, part 2. *	
;*	
;*****	
PRT .DE \$E202	;print A on screen
CHTIM .DE \$E844	;VIA: timer1 lo
MSG1 .DE \$F000	;start of system messages table
;MS1 .DE \$F000	;message: TOO MANY FILES
MSG30 .DE \$FOB6	;message: ARE YOU SURE?
MSG31 .DE \$FOC5	;message: ? BAD DISK

TALK	.DE \$F002	;send talk
LISTN	.DE \$F005	;send listen
MSG	.DE \$F185	;print message from table at \$F000
TKSA	.DE \$F193	;send byte to IEEE
UNTLK	.DE \$F1AE	;send untalk
UNLSN	.DE \$F1B9	;send unlisten
ACPTR	.DE \$F1C0	;get a character from IEEE
GETIN	.DE \$F205	;perform GET
BASIN	.DE \$F215	;get a character in A
BSOUT	.DE \$F266	;print the character in A
CLRCHN	.DE \$F2A6	;restore default I/O
JLTLK	.DE \$F2C1	;search table for filenumber
J2100	.DE \$F2CD	;set file data in table
CLOSS	.DE \$F2E2	;perform part of CLOSE
CLOS10	.DE \$F2E7	;perform part of CLOSE
STOP1	.DE \$F335	;check for STOP key
LD15	.DE \$F356	;universal load routine
LOADNP	.DE \$F408	;perform part of LOAD
OPENIB	.DE \$F4B4	;send name to IEEE
OP94	.DE \$F563	;perform part of OPEN
FOPEN	.DE \$F565	;perform part of OPEN
ERMSG	.DE \$F5AF	;abort files
SV3	.DE \$F6E0	;perform part of SAVE
SV5	.DE \$F6E3	;universal SAVE routine
CHKIN	.DE \$F7AF	;set input device (also \$FFC6)
CHKOUT	.DE \$F7FE	;set output device (also \$FFC9)
TWAIT	.DE \$F928	;wait for normal I/O
INCHR	.DE \$FFCF	;get a character in A
OUTCH	.DE \$FFD2	;print the character in A
	.EN	

END MAE PASS

LABELFILE

A4 =D627	A5 =D629	ACC =D203
ACPTR =F1C0	ADRH =D54C	ADRL =D554
AL2 =D606	AL3 =D618	ALTM =D61D
ALTR =D5FB	ALTRIT =D579	ANSBYE =D8D6
ANSNO =DBCB	ANSYES =D8D5	APPEND =D977
ARGEEXP =D066	ARGHO =D067	ARGLO =D06A
ARGMO =D069	ARGMOH =D068	ARGSGN =D06B
ARISGN =D06C	AS1 =D741	ASC =D73A
ATN =D32C	ATN1 =D334	ATN2 =D342
ATN3 =D355	ATN4 =D35B	ATNCON =D35C
B3 =D491	BACKUP =DA7E	BADDIS =D8D7
BASIN =F215	BBACK =DA8A	BEQS1 =D5F5
BERRD =DA87	BITS =D065	BOBREC =DA31
BRKE =D478	BSOUT =F266	BUF =D020
BY3 =D510	BYTE =D50B	CALLE =D472
CATALG =D87D	CATBLD =D889	CATLOG =D873
CHANNL =D010	CHDGOT =D39F	CHDRTS =D380
CHK1 =D804	CHK2 =D80B	CHK3 =D818
CHK4 =D81D	CHK5 =D824	CHK6 =D82E
CHKCLS =BEEF	CHKCOM =BEF5	CHKER1 =D808
CHKIN =F7AF	CHKQPN =BEF2	CHKOUT =F7FE
CHRGET =D070	CHRGOT =D076	CHTIM =E844
CINV =D090	CKSMDO =D89D	CL0S10 =F2E7
CLOSS =F2E2	CLRCHN =F2A6	CMDS =D544
CNTDN =D08A	COLECT =DA65	CONCAT =D4C7
CONT =D88F	COPCON =DABD	COPY =DAA7
COPY2 =DAB7	COPY3 =DAC0	COS =D282

COSC =D2FA	CRLF =D534	CSTAT =0096
D2 =D589	DAPPEN =D984	DBACKU =DA91
DCLALL =DA1B	DCLBYE =DA30	DCLLP =DA1F
DCLOSE =DA07	DCLSE =DA11	DCOLLO =DA7A
DCOLLE =DA6B	DDIREC =DB99	DEGREE =0064
DELIM1 =DD96	DELIM2 =DEOF	DFLTO =0080
DFORMA =D9E1	DIDCHK =033D	DISKIO =033F
DLOAD =DB3A	DLOAD2 =DB47	DLOERR =DB44
DM =D4F7	DM1 =D4FB	DONE =DE23
DONE1 =DD03	DONER =D7FE	DOPEN =D942
DOPEN2 =D94F	DOSPAR =DC68	DREBLD =DA43
DRECG =DA3D	DRIVE1 =033B	DRIVE2 =033C
DRV1 =DD15	DRV2 =DD0D	DSAVE =DB0D
DSAVE2 =DB1A	DSCRAT =DB6C	DSDESC =0000
DSP1 =D5D2	DSPLYM =DSBC	DSPLYR =D587
DXCRO =DB9D	EAH =00CA	EAL =00C9
ECHKS =D9AB	ENTRY0 =D92F	ENTRY1 =D931
ENTRY2 =D935	ERREAD =D9B3	ERMSG =F5AF
ERRCH1 =D991	ERREND =D9CB	ERRL =D672
ERRLEN =DE74	ERROPR =D7A4	ERROR =B3CF
ERRS1 =D5F8	ERRTAB =B20D	EXIT =D66B
EXP =D184	EXP1 =D1A2	EXPCON =D15B
FA =0004	FACEXP =005E	FACHO =005F
FACLO =0062	FACMO =0061	FACMOH =0060
FACOV =006D	FACSGN =0063	FADD =C99D
FADDH =C97F	FAT =D25B	FBUILD =D9EA
FCERR =C373	FCOMP =CD91	FCONT =D9F5
FDCEND =00FO	FDIV =CC45	FDIVF =CC3D
FERRO =D9DE	FERRP =DA04	FERRS =D9F8
FHALF =D0C7	FLGS =D202	FMULT =CB5E
FNADR =00DA	FNADR2 =00FD	FNLEN =00D1
FNLEN2 =033A	FONE =CAF2	FOPEN =F565
FORMAT =D9D2	FOUR6 =D050	FOUTBL =D0CC
FPWR1 =D135	FPWRT =D112	FPWRT1 =D11B
FR4 =D308	FREMES =D458	FRESTR =C7B5
FRETOP =D030	FRMEVL =BD98	FSUB =C986
FSUBT =C989	G1 =D646	GETADR =C92D
GETBYT =C8D4	GETDS =D995	GETIN =F205
GETSPA =C61D	GETVAL =DE87	GO =D633
GOMDLV =D19F	GOMVMF =D27F	GTVL2 =DE8A
H1 =D797	HEXIT =D78D	IDCON =DD41
IDENT =D037	INCHR =FFCF	INCRND =CD59
INCTMP =D539	INDEX =001F	INIT =D3B6
INITAT =D399	INT =CE02	INTEGR =0003
INVH =D0207	INVL =D0208	JLTLK =F2C1
JMPER =D0051	JZ100 =F2CD	L1 =D688
L10 =D6D2	L11 =D6D6	L12 =D6E3
L13 =D6F0	L14 =D709	L2 =D695
L20 =D700	L3 =D697	L4 =D6A9
L5 =D6AD	L6 =D6B1	L7 =D6B0
L8 =D6C2	L9 =D6CB	LA =0002
LASTPT =D014	LASTWR =D472	LAT =D251
LD =D675	LD15 =F356	LDTND =D0AE
LEAV =D95D	LEAV1 =D970	LENCHK =DE6C
LINNUM =D011	LINPRT =CF83	LISTN =F005
LOADNP =F408	LOG =CB20	LOGADR =DCBD
LOGEB2 =D156	LOOP1 =D9BF	LOOP6 =DD77
LOOPM1 =D408	LOOPMN =D400	LREC1 =D33D
MEMSIZ =D034	MLDEXP =CBEF	MLDVEX =CC0A
MOV1F =CD00	MOV2F =CCFD	MOVAF =CD42
MOVCHG =D3C9	MOVEF =CD45	MOVFA1 =CD34
MOVFM =CCD8	MOVMF =C0DA	MSG =F185
MSG1 =F000	MSG30 =F0B6	MSG31 =F0C5
NAMCON =DD85	NAME1 =DD60	NAME2 =DDF8

NAMEL1 =D006	NEGOP =D14B	NEGRTS =D155
NEWNAM =DE49	NEXT3 =D045	NEXT4 =DD4B
NEXT5 =DE81	NEXT6 =DDA8	NEXT6A =DDAF
NEXT7 =DCB2	NORMAL =CA0D	NUMADR =D7DB
NUMBYE =DB98	NUMERC =DE9A	NUMLP =DB87
NUMPRT =DB93	NUMSCR =DB78	NXX5 =DE82
NXT5 =DE79	NXXX =DD9E	OLDCL1 =DBF1
OLDCLR =DBE1	OLDOV =0053	ON =DE2C
ON1 =D009	ON2 =DDEC	ONTK =0091
OP94 =F563	OPENIB =F4B4	OUTCH =FFD2
PARCHK =033E	PARNXT =DC9F	PARSE2 =DDB6
PARSEL =DC89	PATCH2 =DEA4	PCH =0200
PCL =0201	PI2 =D2FE	POLY =D1ED
POLY1 =D1F1	POLY2 =D204	POLY3 =D200
POLY4 =D211	POLYPT =006E	POLYX =D1D7
POS =033A	PRT =E202	PUTP =D4EC
QSETNR =D247	QTYER2 =DD34	QTYERL =D801
QTYERR =DE27	R1 =D75B	R2 =D762
R3 =D778	R4 =D785	RADDc =D225
RAMLOC =D400	RDFN =DAE1	RDMOV =DAEA
RDOA =D754	RDOB =D763	RDOC =D798
RDRTO =DAF8	RDRT1 =DAF9	READY =B3FF
RECLK =D95B	RECLEM =DCD8	RECON =DCF8
RECOO =DCEA	RECORD =D7AF	REGK =D55C
RENAME =DB55	REXNEX =D7E1	RFOUND =D93F
RID =DAFD	RMULC =D221	RND =D229
RND1 =D25C	RNDX =0088	ROMLOC =B000
RPLCE =DC46	RSFN =DAD4	RUSUR1 =DBA3
RUSUR2 =DBAB	RUSURE =DB9E	RWRT =DC57
RWRTL =DC60	RWRTS =DC67	RXD1 =DC34
RXD2 =DC3D	RXDD =DC44	RXFN1 =DC0D
RXFN2 =DC14	RXIO =DC24	RXREC =DC1B
RXWRT =DC2B	S0 =D4D4	S1 =D4D6
S2 =D4E6	SA =00D3	SAT =0265
SAVL0 =DB23	SAVL01 =DB32	SAVNAM =D209
.SAVX =00B4	SCRTCH =B5D4	SENDP =DBFA
SENDP1 =DBFC	SENDP2 =DCD2	SETR =D523
SETWR =D543	SIGN =CD61	SIN =D289
SIN1 =D2B8	SIN2 =D2BE	SIN3 =D2CB
SINCON =D300	SKIPB =D8D2	SNER2 =D0CD
SNER3 =DE20	SNER8 =DDB3	SNERL =DCBA
SNERR =BF00	SP =D206	SPAC2 =D52E
SPACE =D531	SQR =D108	ST1 =D4C9
STKEND =01FB	STOLD =D194	STOP1 =F335
STRNEX =D26C	STROUT =BB1D	STRT =D4BA
SUBA =D91A	SUBB =D923	SUBBR =D92E
SV3 =F6E0	SV5 =F6E3	SWAPLP =D1B2
SYNCHR =BEF7	SYNERR =D7AC	T2T2 =D744
TABL0 =D838	TALK =F0D2	TAN =D2D2
TANSGN =D00C	TBUF2 =0342	TBUFF =D353
TEMPF1 =0054	TEMPF2 =0059	TEMPF3 =D04B
TEMPPT =0013	TEMPST =0016	TIMEND =D108
TKSA =F193	TMPO =00FB	TMP2 =00FD
TMPC =D00B5	TOTK =00A4	TRANR =DC4C
TRANS =DA98	TRANS1 =DA9B	TT =D746
TWAIT =F92B	TWOPI =D303	TXTPTR =D077
TXTTAB =D028	UNIT1 =D00F	UNIT2 =DDF2
UNLSN =F1B9	UNTLK =F1AE	USEDENC =D417
USEDDEF =D41B	USRCMD =03FA	USRPOK =0000
VERCK =0090	WG220 =D8A5	WG230 =D912
WG235 =D911	WG240 =D905	WG250 =D8D8
WG255 =D8FE	WORDS =D44B	WRAP =D0DE
WROA =D717	WR0B =D722	WRTWO =D731
WSW =D00B3	XR =D204	YR =D205

ZERO =00C9	ZEROF1 =CA2F	b242 =B242
b2a2 =B2A2	b2bd =B2B0	count =0341
d327 =0327	d839 =D839	d83b =D83B
d83e =D83E	d842 =D842	d846 =D846
d847 =D847	d84d =D84D	d84f =D84F
d853 =D853	d85b =D85B	d867 =D867
d86f =D86F	de33 =DE33	de9e =DE9E
qnum =D3A6	scrch =DB66	
//0000,DEC2,DEC2		
]		

```

;NAME BASIC4.OE9N.CTL
;*****
;*      E-ROM for BASIC 4.0      *
;*      for 9 inch screen      *
;*          models             *
;*          with                *
;*          N-keyboard          *
;*          *
;* Date 14-11-83  Time 00:35  *
;*          *
;* Made by:                   *
;*          *
;* Nico de Vries              *
;* Mari Andriessenrade 49    *
;* 2907 MA Capelle a/d Yssel *
;* The Netherlands            *
;*          *
;* Original CBM ROM number   *
;*          *
;*          1447-29              *
;*          *
;*****.BA $E000
.FI "BASIC4.OE9N.M01"

```

16E0 33B1-4A91 BASIC4.OE9N.M01

```

;NAME BASIC4.OE9N.M01
;*****
;*      Reset; initialize I/O.  *
;*          *
;*****.E000- A9 7F      CINT      LDA #$7F      ;disable all VIA interrupts
.E002- 80 4E E8          STA IER
.E005- A2 60          LDX #$60
.E007- A9 00          LDA #$00      ;clear
.E009- 95 80      PX1      STA *CTIMR,X    ;area $80 to $FA
.E00B- CA          DEX
.E00C- 10 FB          BPL PX1
.E00E- A9 55          LDA #L,KEY    ;set
.E010- 85 90          STA *CINV     ;IRQ RAM vector
.E012- A9 E4          LDA #H,KEY    ;to
.E014- 85 91          STA *CINV+1   ;keyboard/cassette routine
.E016- A9 03          LDA #3        ;set default output device
.E018- 85 B0          STA *DFLTO    ;to screen
.E01A- A9 0F          LDA #$0F     ;keyboard PIA, A section:
.E01C- 8D 10 E8          STA PIAL     ;bits 0-3 output, rest input
.E01F- 0A          ASL A
.E020- 8D 40 E8          STA PIA      ;cassette#2 motor off
;casette #1 and #2 write = 1
;IEEE ATN=false NRFD=false
.E023- 8D 42 E8          STA P20B     ;bits 7-5,0 output, 4-1 input
.E026- 8E 22 E8          STX IEE0     ;IEEE PIA, B section: all bits output
.E029- 8E 45 E8          STX T1H      ;VIAT1H: maximum timeout
.E02C- A9 30          LDA #$3D     ;keyboard PIA: cassette#1 motor off
.E02E- 8D 13 E8          STA PIAS     ;IRQ on neg edge on CB1
;(end of screen refresh)

```

```

E031- 2C 12 E8      BIT PIAK      ;clear keyboard PIA interrupt flags
E034- A9 3C          LDA #$3C      ;IEEE PIA: NDAC=false, CA1 neg. edge
E036- 8D 21 E8      STA IEEEIS    ;no interrupts
E039- 8D 23 E8      STA IEEOS    ;IEEE PIA: DAV=false, CB1 neg. edge
                           ;no interrupts
E03C- 8D 11 E8      STA PIAL1    ;keyboard PIA: CA2=1,CA1 neg. edge
                           ;no interrupts
E03F- 8E 22 E8      STX IEEEO    ;IEEE PIA: all IEEE data line false
E042- A9 0C          LDA #$0C      ;VIA:CA2=0 (graphics),CA1 neg. edge
E044- 8D 4C E8      STA PCR      ;no interrupts,CB2 input,neg. edge
E047- 85 A8          STA *BLNCT   ;set blink count for cursor
E049- 85 A7          STA *BLNSW   ;set cursor visible
                           ;*****
                           ;*
                           ;* Initialize screen (clear it). *
                           ;*
                           ;*****
E04B- A0 83          CLSR       LDY #$83      ;get hi-byte also
E04D- A2 18          LDX #24      ;do 25 lines
E04F- 94 E0          LPS1       STY *LDTB1,X ;store hi-byte in table
E051- E0 14          CPX #20      ;if on line 20
E053- F0 08          BEQ LPS2    ;adapt pointer
E055- E0 0D          CPX #13      ;if on line 13
E057- F0 04          BEQ LPS2    ;adapt pointer
E059- E0 07          CPX #7       ;if on line 7
E05B- D0 01          BNE LPS3   ;adapt pointer
E05D- 88             LPS2       DEY        ;adapt hi-byte
E05E- CA             LPS3       DEX        ;adapt line counter
E05F- 10 EE          BPL LPS1   ;if all lines done
E061- 84 C5          STY *PNT+1  ;set pointer in ($C4)
E063- E8             INX        ;make X zero
E064- 86 9F          STX *RVS    ;set rvs off
E066- 86 C4          STX *PNT    ;and store lo-byte of pptr
E068- A9 20          LDA #$20    ;put a blank
E06A- 9D 00 80        LPS4       STA $8000,X ;into all bytes
E06D- 9D 00 81        STA $8100,X ;of screen
E070- 9D 00 82        STA $8200,X
E073- 9D 00 83        STA $8300,X
E076- CA             DEX        ;until whole screen done
E077- D0 F1          BNE LPS4   ;*****
                           ;*
                           ;* Home cursor. *
                           ;*
                           ;*****
E079- A0 00          NXTD       LDY #$00
E07B- 84 C6          STY *PNTR   ;set cursor on start of line
E07D- 84 D8          STY *TBLX   ;and on first screen line
                           ;*****
                           ;*
                           ;* Get pointer to current line in *
                           ;* screen RAM. *
                           ;*
                           ;* Expects $08 to contain current *
                           ;* line number; ($04) will be set *
                           ;* to point to current line; $D5 *
                           ;* will be set to max. length of *
                           ;* that line. *
                           ;*
                           ;*****
E07F- A6 D8          STUPT      LDX *TBLX   ;get screen line of cursor
E081- B5 E0          LDA *LDTB1,X ;get hi-byte of that line
E083- 09 80          ORA #$80    ;set bit 7 (flag in table)
E085- 85 C5          STA *PNT+1  ;and store in pptr

```

E087-	B0 5B E6	LDA LDTB2,X	;get lo-byte from table
E08A-	85 C4	STA *PNT	;store in pntr too
E08C-	A9 27	LDA #39	;set length to 39
E08E-	85 D5	STA *LNMX	
E090-	E0 18	CPX #24	;if not bottom line
E092-	F0 08	BEQ STUPZ	
E094-	B5 E1	LDA *LDTB1+1,X	;and bit 7 from table clear
E096-	30 04	BMI STUPZ	
E098-	A9 4F	LDA #79	;then length is 79
E09A-	85 D5	STA *LNMX	
E09C-	A5 C6	STUPZ LDA *PNTR	;get position on line
E09E-	C9 28	CMP #40	
E0A0-	90 04	BCC STUPR	
E0A2-	E9 28	SBC #40	
E0A4-	85 C6	STA *PNTR	;and store modulo 40
E0A6-	60	STUPR RTS	
;*****			
;*			
;* Get character from keyboard *			
;* buffer. *			
;*			
;*****			
E0A7-	AC 6F 02	LP2 LDY KEYD	;get oldest character
E0AA-	A2 00	LOX #\$00	;and
E0AC-	B0 70 02	LP1 LDA KEYD+1,X	;move rest of buffer
E0AF-	90 6F 02	STA KEYD,X	;forward
E0B2-	E8	INX	
E0B3-	E4 9E	CPX *NDX	;until all char's done
E0B5-	D0 F5	BNE LP1	
E0B7-	C6 9E	DEC *NDX	;adapt index in buffer
E0B9-	98	TYA	;get character in accu
E0BA-	58	CLI	;enable interrupts again
E0BB-	60	RTS	
;*****			
;*			
;* Wait for RETURN from keyboard. *			
;*			
;*****			
E0BC-	20 02 E2	LOOP4 JSR PRT	;echo character on screen
E0BF-	A5 9E	LOOP3 LDA *NDX	;get index in keyb. buffer
E0C1-	85 A7	STA *BLNSW	;if empty, set cursor visible
E0C3-	F0 FA	BEQ LOOP3	;else wait for character
E0C5-	78	SEI	;disable interrupts
E0C6-	A5 AA	LDA *BLNON	;if blink fase one
E0C8-	F0 09	BEQ LP21	
E0CA-	A5 A9	LDA *GDBLN	;get true char. at cursor position
E0CC-	A0 00	LDY #\$00	;and
E0CE-	84 AA	STY *BLNON	;set blink fase zero
E0D0-	20 06 E6	JSR DSPP	;and put char to screen
E0D3-	20 A7 E0	LP21 JSR LP2	;get character from buffer
E0D6-	C9 83	CMP #\$83	;if shift-STOP
E0D8-	D0 10	BNE LP22	
E0DA-	78	SEI	;disable interrupts
E0DB-	A2 09	LDX #9	
E0DD-	86 9E	STX *NDX	;set index in buffer to 9
E0DF-	B0 73 E6	LP23 LDA RUNTB-1,X	;move standard string
E0E2-	90 6E 02	STA KEYD-1,X	;to buffer
E0E5-	CA	DEX	
E0E6-	D0 F7	BNE LP23	
E0E8-	F0 D5	BEQ LOOP3	;and get 1st char. of that string
E0EA-	C9 0D	CMP #\$0D	;if is not RETURN
E0EC-	D0 CE	BNE LOOP4	;go echo to screen and get next char
E0EE-	A4 D5	LDY *LNMX	;else get max. length
E0F0-	B4 AC	STY *CRSW	;flag input from screen

EOF2- B1 C4	CLPS	LDA (PNT),Y	;get char. from screen
EOF4- C9 20		CMP #\$20	;if space at end of line
EOF6- D0 03		BNE CLP6	
EOF8- 88		DEY	;get previous character
EOF9- D0 F7		BNE CLPS	;until valid character found
EOFB- C8	CLPS	INY	
EOFC- 84 A1		STY *INDX	;set record length of current line
EOFE- A0 00		LDY #\$00	
E100- 84 C6		STY *PNTR	;and set cursor at start of line
E102- 84 CD		STY *QTSW	;clear quote flag
E104- A5 A3		LDA *LXSP	;get line number on screen
E106- 30 16		BMI LOPS	;if valid
E108- C5 D8		CMP *TBLX	;and equal to original
E10A- D0 12		BNE LOPS	
E10C- A5 A4		LDA *LXSP+1	;set position
E10E- 85 C6		STA *PNTR	;to original value
E110- C5 A1		CMP *INDX	;if not behind end of record
E112- 90 0A		BCC LOPS	;get char from device 3
E114- B0 2B		BCS CLP2	;else return RETURN

		;* Get character from device 0 *	
		;* or device 3. *	
		;*	

E116- 98	LOOPS	TYA	
E117- 48		PHA	;save Y
E118- 8A		TXA	;and
E119- 48		PHA	;X
E11A- A5 AC		LDA *CRSW	;if input from keyboard
E11C- F0 A1		BEQ LOOP3	;go do keyboard until RETURN
		.FI "BASIC4.OE9N.M02"	

173A 33B1-4AEB BASIC4.OE9N.M02

	;NAME BASIC4.OE9N.M02		

	;*		
	;* Get character from current *		
	;* screen line. *		
	;*		

E11E- A4 C6	LOPS	LDY *PNTR	;get current position
E120- B1 C4		LDA (PNT),Y	;get screen character
E122- 85 D9		STA *DATA	;save it
E124- 29 3F	LOP51	AND #\$3F	;remove bits 6 and 7
E126- 04 D9		ASL *DATA	
E128- 24 D9		BIT *DATA	;if original bit 6 set
E12A- 10 02		BPL LOP54	
E12C- 09 80		ORA #\$80	;set bit 7 (shift bit)
E12E- 90 04	LOP54	BCC LOP52	;if original bit 7 set
E130- A6 CD		LDX *QTSW	;and if in quote mode
E132- D0 04		BNE LOP53	;don't change bit 6
E134- 70 02	LOP52	BVS LOP53	;else if bit 5 set in original
E136- 09 40		ORA #\$40	;set bit 6
E138- E6 C6	LOP53	INC *PNTR	;adapt position
E13A- 20 67 E1		JSR QTSWC	;adapt quote flag
E13D- C4 A1		CPY *INDX	;if max position reached
E13F- D0 17		BNE CLP1	
E141- A9 00	CLP2	LDA #\$00	
E143- 85 AC		STA *CRSW	;flag input from keyboard
E145- A9 00		LDA #\$00	;and echo RETURN
E147- A6 AF		LDX *DFLTN	;if input device

```

E149- E0 03      CPX #3          ;is screen
E14B- F0 06      BEQ CLP2A       ;go echo on screen
E14D- A6 B0      LDX *DFLTO      ;else if output device
E14F- E0 03      CPX #3          ;is screen
E151- F0 03      BEQ CLP21       ;store RETURN in last input
E153- 20 02 E2    CLP2A         JSR PRT        ;echo on screen
E156- A9 0D      CLP21         LDA #$0D        ;and store a RETURN
E158- 85 D9      CLP1          STA *DATA       ;in last key input
E15A- 68          PLA           PLA
E15B- AA          TAX           ;restore X
E15C- 68          PLA           ;and
E15D- A8          TAY           ;Y
E15E- A5 D9      LDA *DATA       ;get character again
E160- C9 DE      CMP #$0E        ;if code for PI
E162- D0 02      BNE CLP7       CLP7
E164- A9 FF      LDA #PI         ;get real code for PI
E166- 60          CLP7         RTS
;*****
;*
;* Toggle quote flag if " found. *
;*
;*****
E167- C9 22      QTSWC        CMP #$22        ;if character is "
E169- D0 08      BNE QTSWL      QTSWL
E16B- A5 CD      LDA *QTSW       ;get quote flag
E16D- 49 01      EOR #$01        ;invert bit 0
E16F- 85 CD      STA *QTSW       QTSWL
E171- A9 22      LDA #$22        ;and restore character
E173- 60          QTSWL        RTS
;*****
;*
;* Fill screen at current   *
;* position.                 *
;*
;*****
E174- 09 40      NXT33        ORA #$40        ;map shifted characters
E176- A6 9F      NXT3          LDX *RVS        ;if reverse switch on
E178- F0 02      BEQ NVS         NVS
E17A- 09 80      NC3          ORA #$80        ;set screencode for reverse
E17C- A6 DC      NVS          LDX *INSRT      ;if in insert mode
E17E- F0 02      BEQ NVSL        NVSL
E180- C6 DC      DEC *INSRT      ;decrease insert count
E182- 20 06 E6    NVSL         JSR DSPP        ;put char at current position in RAM
E185- E6 C6      INC *PNTR        PNTR
E187- A4 D5      LDY *LNMX        LNMX
E189- C4 C6      CPY *PNTR        PNTR
E18B- B0 19      BCS LOOP2       LOOP2
E18D- A6 D8      LDX *TBLX        TBLX
E18F- C0 4F      CPY #79         ;get line number in X
E191- D0 DC      BNE JSTS1       JSTS1
E193- 20 B3 E1    JSR JSTSX      JSTSX
E196- 20 43 E3    JSR NXLN       NXLN
E199- A9 00      LDA #0          ;go to start of next line
E19B- 85 C6      STA *PNTR        PNTR
E19D- F0 07      BEQ LOOP2       LOOP2
E19F- E0 18      JSTS1         CPX #24        ;if beyond length, but <79
E1A1- D0 18      BNE JTS2         JTS2
E1A3- 20 C4 E1    JSR SCRL        SCRL
;scroll screen up (leave cursor on same
;*****
;*
;* Exit from output to screen. *
;*
;*****
E1A6- 68          LOOP2        PLA

```

```

E1A7- A8          TAY           ;restore Y
E1A8- A5 DC       LDA *INSRT    ;if in insert mode
E1AA- F0 02       BEQ LOP2
E1AC- 46 CD       LSR *QTSW     ;set quote mode off
E1AE- 68          PLA
E1AF- AA          TAX          ;restore X
E1B0- 68          PLA          ;and A
E1B1- 58          CLI          ;allow interrupts again
E1B2- 60          RTS
;*****
;*
;* Set next line to *
;* 'not-extended'.   *
;*
;*****
E1B3- E0 17        JSTSX        CPX #23      ;if line number <=23
E1B5- B0 06        BCS JSXB
E1B7- B5 E2        LDA *LDTB1+2,X ;set next line
E1B9- 09 80        ORA #$80     ;to 'not-extended'
E1BB- 95 E2        STA *LDTB1+2,X
E1B0- 60          RTS
;*****
;*
;* Insert blank line if beyond *
;* current length, but not at end *
;* of line.                  *
;*
;*****
E1BE- 20 CD E1    JTS2         JSR JTS2     ;if beyond length but not at pos. 79
;insert new line
E1C1- 4C A6 E1    JMP LOOP2    ;and exit
;*****
;*
;* Get or insert new line.      *
;*
;*****
E1C4- 20 69 E3    SCRL         JSR SCROL    ;scroll lines on screen
E1C7- C6 A3        DEC *LXSP     ;decrement copy of line number
E1C9- C6 D8        DEC *TBLX     ;and line# itself (cursor on same line)
E1CB- A6 D8        LDX *TBLX     ;get line number
E1CD- 16 E1        ASL *LDTB1+1,X ;get flag of line in carry
E1CF- 56 E1        LSR *LDTB1+1,X ;set line to 'extended'
E1D1- 20 B3 E1    JSR JSTSX    ;and next line to 'not-extended'
E1D4- A5 C6        LDA *PNTR     ;save position
E1D6- 48          PHA
E1D7- 20 7F EO    JSR STUPT    ;and set pointer
E1DA- 68          PLA
E1DB- 85 C6        STA *PNTR     ;and restore position
E1DD- 60          RTS
;*****
;*
;* Backward over line boundary. *
;*
;*****
E1DE- A0 27        BKLN         LDY #39
E1E0- A6 D8        LDX *TBLX     ;if current line
E1E2- D0 06        BNE BKLN1    ;is top line
E1E4- 86 C6        STX *PNTR     ;set cursor home
E1E6- 68          PLA
E1E7- 68          PLA          ;remove return address
E1E8- D0 BC        BNE LOOP2    ;and exit
E1EA- B5 DF        BKLN1       LDA *LDTB1-1,X ;if not on top line
E1EC- 30 05        BMI NTCN2    ;and line is 'extended'
E1EE- CA          DEX

```

E1EF- B5 DF		LDA *LDTB1-1,X	;get indicator of next line
E1F1- A0 4F		LDY #79	;and set position to 79
E1F3- CA	NTCN2	DEX	;adapt line number
E1F4- 86 D8		STX *TBLX	;and store it
E1F6- 85 C5		STA *PNT+1	;and store new pointer
E1F8- BD 5B E6		LDA LDTB2,X	;store lo-byte
E1FB- 85 C4		STA *PNT	;as well
E1FD- 84 C6		STY *PNTR	;set cursor at end of line
E1FF- 84 D5		STY *LNMX	;and set length to 79
E201- 60		RTS	
		;*****	
		;*	
		;* Put character to screen. *	
		;*	
		;*****	
E202- 48	PRT	PHA	;save A
E203- 85 D9		STA *DATA	;also in temporary field
E205- 8A		TXA	
E206- 48		PHA	;save X
E207- 98		TYA	
E208- 48		PHA	;and Y
E209- A9 00		LDA #0	
E20B- 85 AC		STA *CRSW	;set copy of position to zero
E20D- A4 C6		LDY *PNTR	;get position
E20F- A5 D9		LDA *DATA	;get character again
E211- 10 03		BPL e216	;if shifted
E213- 4C A4 E2		JMP NXTX	;go do shifted character
E216- C9 0D	e216	CMP #\$0D	;if RETURN
E218- D0 03		BNE NJTL	
E21A- 4C 59 E3		JMP NXTL	;go do RETURN
E21D- C9 20	NJTL	CMP #\$20	;if printable
E21F- 90 08		BCC NTCN	
E221- 29 3F		AND #\$3F	;convert to screen code
E223- 20 67 E1		JSR QTSWC	;check for quote mode
E226- 4C 76 E1		JMP NXT3	;and put to screen
E229- A6 DC	NTCN	LDX *INSRT	;if inserts outstanding
E22B- F0 03		BEQ CNC3X	
E22D- 4C 7A E1		JMP NC3	;print as control character
E230- C9 14	CNC3X	CMP #\$14	;if DELETE
E232- D0 1C		BNE NTCN1	
E234- 88		DEY	;decrement character position
E235- 84 C6		STY *PNTR	;and store it
E237- 10 06		BPL BK1	;if gone to previous line
E239- 20 DE E1		JSR BKLN	;go do appropriate action
E23C- 4C 4A E2		JMP BK2	;and insert blank at end
		.FI "BASIC4.OE9N.M03"	

12BB 33B1-466C BASIC4.OE9N.M03

;NAME BASIC4.OE9N.M03			
E23F- C8	BK1	INY	;else
E240- B1 C4		LDA (PNT),Y	;compress
E242- 88		DEY	;line
E243- 91 C4		STA (PNT),Y	;upto
E245- C8		INY	;end of
E246- C4 C5		CPY *PNT+1	;line
E248- D0 F5		BNE BK1	
E24A- A9 20	BK2	LDA #\$20	;and insert blank
E24C- 91 C4		STA (PNT),Y	;at end
E24E- D0 3C		BNE JPL3	;and exit
E250- A6 CD	NTCN1	LDX *QTSW	;if in quote mode
E252- F0 03		BEQ NC3W	
E254- 4C 7A E1	CNC3	JMP NC3	;print as control character

E257- C9 12	NC3W	CMP #\$12	;if RVS-ON
E259- D0 02		BNE NC1	
E25B- 85 9F		STA *RVS	;set reverse flag
E25D- C9 13	NC1	CMP #\$13	;if HOME
E25F- D0 03		BNE NC2	
E261- 20 79 E0		JSR NXTD	;set cursor home
E264- C9 1D	NC2	CMP #\$1D	;if cursor-RIGHT
E266- D0 12		BNE NCX2	
E268- C8		INY	
E269- 84 C6		STY *PNT	;advance cursor
E26B- 88		DEY	
E26C- C4 C5		CPY *PNT+1	;if at line end
E26E- 90 07		BCC NCZ2	
E270- 20 43 E3		JSR NXLN	;go to next line
E273- A0 00		LDY #\$00	;set cursor at beginning
E275- 84 C6	JPL4	STY *PNT	;store position
E277- 4C A6 E1	NCZ2	JMP LOOP2	;and exit
E27A- C9 11	NCX2	CMP #\$11	;if not cursor DOWN
E27C- D0 DE		BNE JPL3	;exit (ignore key)
E27E- 18		CLC	;else prepare for add
E27F- 98		TYA	;get position in A
E280- 69 28		ADC #40	;add 40
E282- A8		TAY	;put in Y again
E283- C5 D5		CMP *LNMX	;if not beyond length
E285- 90 EE		BCC JPL4	;go store position
E287- F0 EC		BEQ JPL4	;if at end also
E289- 20 43 E3		JSR NXLN	;else goto next line
E28C- 4C A6 E1	JPL3	JMP LOOP2	;and exit

;*			
;* This code is not used, left *			
;* over from previous generations *			
;* of CBM BASIC (not used then) *			
;* too). *			
;*			

E28F- E8	PRT1	INX	
E290- 85 D8		STA *TBLX	
E292- 98		TYA	;get position in A
E293- E9 28		SBC #40	;set to previous line
E295- 85 C6		STA *PNT	;store it
E297- E6 D8		INC *TBLX	;adapt line number
E299- AD 5B E6		LDA LDTB2	;get pointer for topline
E29C- 85 C4		STA *PNT	;store lo
E29E- A5 E0		LDA *LDTB1	;get hi also
E2A0- 85 C5		STA *PNT+1	;store it
E2A2- D0 E8		BNE JPL3	;and exit

;*			
;* Put shifted characters to *			
;* screen. *			
;*			

E2A4- 29 7F	NXTX	AND #\$7F	;remove shift bit
E2A6- C9 7F		CMP #PI-\$80	;if code for PI
E2A8- D0 02		BNE NXTX1	
E2AA- A9 5E		LDA #\$5E	;get real code for PI
E2AC- C9 20	NXTX1	CMP #\$20	;if printable
E2AE- 90 03		BCC e2b3	
E2B0- 4C 74 E1		JMP NXT33	;go fill screen
E2B3- C9 00	e2b3	CMP #\$00	;if shift-RETURN
E2B5- D0 03		BNE UPS	
E2B7- 4C 59 E3		JMP NXTL	;go do RETURN
E2BA- A6 CD	UPS	LDX *QTSW	;if quote mode

E2BC-	D0 30	BNE UP6	;print as control character
E2BE-	C9 14	CMP #\$14	;no quote mode, if INSERT
E2CO-	D0 28	BNE UP9	
E2C2-	A4 D5	LDY *LNMX	;get line length
E2C4-	B1 C4	LDA (PNT),Y	;get last char. of line
E2C6-	C9 20	CMP #\$20	;if blank
E2C8-	D0 04	BNE INS3	
E2CA-	C4 C6	CPY *PNTR	;and not on last position
E2CC-	D0 07	BNE INS1	;then enough space on line
E2CE-	CO 4F	INS3 CPY #79	;if last char. not blank
E2D0-	F0 BA	BEQ JPL3	;and length is 79, ignore
E2D2-	20 E2 E3	JSR NEWLN	;else scroll rest of screen down
E2D5-	A4 D5	INS1 LDY *LNMX	;get new length
E2D7-	88	INS2 DEY	;move
E2D8-	B1 C4	LDA (PNT),Y	;characters
E2DA-	C8	INY	;one position
E2DB-	91 C4	STA (PNT),Y	;further
E2DD-	88	DEY	;until
E2DE-	C4 C6	CPY *PNTR	;all done
E2EO-	D0 F5	BNE INS2	
E2E2-	A9 20	LDA #\$20	;and put a blank
E2E4-	91 C4	STA (PNT),Y	;in space opened up
E2E6-	E6 DC	INC *INSRT	;adapt insert counter
E2E8-	D0 56	BNE JLP2	;and exit
E2EA-	A6 DC	.UP9 LDX *INSRT	;if inserts outstanding
E2EC-	F0 05	BEQ UP2	
E2EE-	09 40	UP6 ORA #\$40	;convert to screen code
E2F0-	4C 7A E1	JMP NC3	;and print as control char.
E2F3-	C9 11	UP2 CMP #\$11	;if cursor-UP
E2F5-	D0 2B	BNE NXT2	
E2F7-	A5 C6	LDA *PNTR	
E2F9-	C9 28	CMP #40	;and current position >40
E2FB-	90 06	BCC UP1	
E2FD-	E9 28	SBC #40	;move cursor up
E2FF-	85 C6	STA *PNTR	
E301-	B0 30	BCS JLP2	
E303-	A6 08	UP1 LDX *TBLX	;get line number
E305-	F0 39	BEQ JLP2	;if top line, exit
E307-	B5 DF	LDA *LDTB1-1,X	;if previous line 'not-extended'
E309-	10 07	BPL UP3	
E30B-	C6 D8	DEC *TBLX	;decrement line number
E30D-	20 7F EO	JSR STUPT	;adapt pointer
E310-	90 2E	BCC JLP2	;and exit
E312-	CA	UP3 DEX	;if previous line 'extended'
E313-	CA	DEX	;decrement line number twice
E314-	86 D8	STX *TBLX	;and store it
E316-	20 7F EO	JSR STUPT	;get pointer
E319-	A5 C6	LDA *PNTR	;get position
E31B-	18	CLC	
E31C-	69 28	ADC #40	;and set cursor on second
E31E-	85 C6	STA *PNTR	;part of that line
E320-	D0 1E	BNE JLP2	;and exit
E322-	C9 12	NXT2 CMP #\$12	;if RVS-off
E324-	D0 04	BNE NXT6	
E326-	A9 00	LDA #\$00	
E328-	85 9F	STA *RVS	;reset reverse flag
E32A-	C9 10	NXT6 CMP #\$1D	;if cursor-LEFT
E32C-	D0 0B	BNE e339	
E32E-	88	DEY	
E32F-	84 C6	STY *PNTR	;decrement position
E331-	10 0D	BPL JLP2	;exit, if still on same line
E333-	20 DE E1	JSR BKLN	;else do backwards over boundary
E336-	4C A6 E1	e336 JMP LOOP2	;and exit
E339-	C9 13	e339 CMP #\$13	;if not CLR-screen

E33B- D0 03		BNE JLP2 ;exit (ignore key)
E33D- 20 4B E0	e33d	JSR CLSR ;else clear the screen
E340- 4C A6 E1	JLP2	JMP LOOP2 ;and exit
;*****		
;* * * * *		
;* Set next line number. *		
;* * * * *		
;*****		
E343- 38	NXLN	SEC ;set copy of line number invalid
E344- 46 A3		LSR *LXSP ;BUG: wrong type of shift
E346- A6 D8		LDX *TBLX ;get line number
E348- E8	NXLN2	INX ;increment it
E349- E0 19		CPX #25 ;if at maximum line number
E34B- D0 03		BNE NXLN1
E34D- 20 69 E3		JSR SCROL ;scroll screen up
E350- B5 E0	NXLN1	LDA *LDTB1,X ;if line is 'extended'
E352- 10 F4		BPL NXLN2 ;scroll another time
E354- 86 D8		STX *TBLX ;store line number
E356- 4C 7F E0		JMP STUPT ;and get pointer of current line
;*****		
;* * * * *		
;* Action for RETURN. *		
;* * * * *		
;*****		
E359- A9 00	NXTL	LDA #0
E35B- 85 DC		STA *INSRT ;cancel inserts outstanding
E35D- 85 9F		STA *RVS ;reset reverse flag
E35F- 85 CD		STA *QTSW ;cancel quote mode
E361- 85 C6		STA *PNTR ;set cursor at beginning of line
E363- 20 43 E3		JSR NXLN ;set next line number
E366- 4C A6 E1		JMP LOOP2 ;and exit
.FI "BASIC4.OE9N.M04"		

19E4 33B1-4D95 BASIC4.OE9N.M04

;NAME BASIC4.OE9N.M04.		
;*****		
;* * * * *		
;* Scroll screen up. *		
;* * * * *		
E369- A0 00	SCROL	LDY #0 ;get lo-byte of first line
E36B- 84 C4		STY *PNT ;store it
E36D- A9 80		LDA #\$80 ;get hi-byte also
E36F- 85 C8		STA *SAH ;store in source pointer
E371- 85 C5		STA *PNT+1 ;and in destination pointer
E373- A9 28		LDA #40 ;get lo of source pointer
E375- 24 E1		BIT *LDTB1+1 ;if 1st line is 'extended'
E377- 30 02		BMI SCRL1
E379- A9 50		LDA #80 ;get new lo byte
E37B- 85 C7	SCRL1	STA *SAL ;and complete pointers
E37D- A9 34		LDA #\$34 ;set screen invisible
E37F- 8D 11 E8		STA PIAL1 ;BUG: hardware to do this missing
E382- E1 C7	MLP1	LDA (SAL),Y ;get char of next line
E384- 91 C4		STA (PNT),Y ;move it to current line
E386- C8		INY ;until
E387- D0 F9		BNE MLP1 ;whole page done
E389- E6 C8		INC *SAH ;adapt source pointer hi
E38B- E6 C5		INC *PNT+1 ;and destination pointer hi
E38D- A9 84		LDA #\$84 ;if last line
E38F- C5 C8		CMP *SAH ;not reached yet
E391- D0 EF		BNE MLP1 ;loop
E393- A9 E8		LDA #\$E8 ;else adapt pointer

E395- 85 C4	STA *PNT	;to point	
E397- C6 C5	DEC *PNT+1	;beyond last screen line	
E399- A9 20	LDA #\$20	;get a blank	
E39B- C6 C4	MLP2	DEC *PNT	;adapt pointer
E39D- C6 C7		DEC *SAL	;use scrolled portion as counter
E39F- 91 C4		STA (PNT),Y	;fill bottom line with spaces
E3A1- DD F8		BNE MLP2	
E3A3- A2 19		LDX #25	;get line number
E3A5- 86 D8		STX *TBLX	;store it
E3A7- A2 00	SCRL4	LDX #0	;start at first line
E3A9- C6 D8		DEC *TBLX	;adapt line number
E3AB- B5 E0	SCRL5	LDA *LDTB1,X	;get extension flag of 1st line
E3AD- 29 7F		AND #\$7F	;remove flag bit
E3AF- B4 E1		LDY *LDTB1+1,X	;get flag of next line
E3B1- 10 02		BPL SCRL3	;if not extended
E3B3- 09 80		ORA #\$80	;add flag to previous line
E3B5- 95 E0	SCRL3	STA *LDTB1,X	;and store
E3B7- E8		INX	;adapt line number
E3B8- E0 19		CPX #25	;if not all done
E3BA- D0 EF		BNE SCRL5	;loop
E3BC- A9 83		LDA #\$83	;set last line
E3BE- 85 F8		STA *LDTB1+24	;to 'not extended'
E3C0- A5 E0		LDA *LDTB1	;if top line was extended
E3C2- 10 E3		BPL SCRL4	;move all flags 1 extra line
E3C4- A9 3C		LDA #\$3C	;set screen visible again
E3C6- 8D 11 E8		STA PIAL1	;BUG: hardware missing
		;BUG: sends EOI on IEEE	
E3C9- A9 FE		LDA #\$FE	;get key image of RVS key
E3CB- CD 12 E8		CMP PIAK	;if RVS pressed
E3CE- D0 0F		BNE MLP42	
E3D0- A0 08		LDY #8	;get counter
E3D2- 8D 45 E8	MLP4	STA T1H	;load timer hi (and start it)
E3D5- 2C 4D E8	MLP4:1	BIT IFR	;and wait until
E3D8- 50 FB		BVC MLP41	;timed out
E3DA- 88		DEY	;adapt counter
E3DB- D0 F5		BNE MLP4	;if not all done, loop
E3DD- 84 9E		STY *NDX	;then cancel keyboard buffer
E3DF- A5 D8	MLP42	LDA *TBLX	;get screenline number
E3E1- 60		RTS	;and exit

		/*	*
		/* Scroll rest of screen down	*
		/* and insert a blank line on	*
		/* cursor's line.	*
		/*	*

E3E2- A6 D8	NEWLN	LDX *TBLX	;get current line number
E3E4- E8		INX	
E3E5- A9 34		LDA #\$34	;set screen invisible
E3E7- 8D 11 E8		STA PIAL1	;BUG: hardware to do this missing
E3EA- E0 18		CPX #24	;if on last line but one
E3EC- F0 33		BEQ BLKLN	;blank this line
E3EE- 90 03		BCC NEWLX	;if on last line
E3FO- 4C C4 E1		JMP SCRL	;insert a line
E3F3- A2 17	NEWLX	LDX #23	;start with bottom line
E3F5- B5 E1	NEXL1	LDA *LDTB1+1,X	;get flag of next line
E3F7- 09 80		ORA #\$80	;add MSB
E3F9- 85 C8		STA *SAH	;and set up pointer
E3FB- B4 E0		LDY *LDTB1,X	;get previous linemarker
E3FD- 30 02		BMI NEWLA	;if 'extended'
E3FF- 29 7F		AND #\$7F	;remove MSB
E401- 95 E1	NEWLA	STA *LDTB1+1,X	;and put in table
E403- 98		TYA	;move marker of previous line
E404- 09 80		ORA #\$80	;add MSB

```

E406- 85 C5      STA *PNT+1    ;and set up pointer
E408- A0 27      LDY #39       ;40 characters to be moved
E40A- BD 5C E6    LDA LDTB2+1,X ;get lo-byte of lower line
E40D- 85 C7      STA *SAL      ;complete pointer
E40F- BD 5B E6    LDA LDTB2,X  ;get lo-byte of upper line
E412- 85 C4      STA *PNT      ;complete pointer
E414- B1 C4      NELL        LDA (PNT),Y   ;get character of upper line
E416- 91 C7      STA (SAL),Y  ;store in lower line
E418- 88          DEY          ;
E419- 10 F9      BPL NELL     ;repeat until all char.s moved
E41B- CA          DEX          ;if not all lines done
E41C- E4 D8      CPX *TBLX    ;
E41E- D0 D5      BNE NEXL1    ;repeat for other lines
E420- E8          INX          ;
E421- B5 EO      BLKLN        LDA *LDTB1,X ;get marker of top line
E423- 09 80      ORA #$80     ;add MSB
E425- 85 C5      STA *PNT+1    ;set up pointer
E427- 29 7F      AND #$7F     ;set line 'extended'
E429- 95 EO      STA *LDTB1,X ;into table
E42B- BD 5B E6    LDA LDTB2,X ;get lo-byte
E42E- 85 C4      STA *PNT      ;complete pointer
E430- A0 27      LDY #39       ;
E432- A9 20      LDA #$20     ;get blank
E434- 91 C4      BLKL        STA (PNT),Y   ;fill top line with blanks
E436- 88          DEY          ;
E437- 10 FB      BPL BLKL     ;until whole line filled
E439- A9 3C      LDA #$3C     ;set screen visible again
E43B- 8D 11 E8    STA PIAL1    ;BUG: sends EOI on IEEE
                           ;BUG: hardware to do this missing
E43E- 58          CLI          ;enable interrupts again
E43F- 4C 7F EO    JMP STUPT    ;and get pointer to current line
                           ;*****
                           ;*
                           ;* Default interrupt routine, *
                           ;* entered 60 times per second. *
                           ;*
                           ;* (Pointed to by ROM). *
                           ;*
                           ;*****  

E442- 48          PULS        PHA          ;save A
E443- 8A          TXA         PHA          ;save X
E444- 48          PHA         TYA          ;and Y
E445- 98          TYA         TSX          ;get stackpointer
E446- 48          PHA         LDA $0104,X ;get saved status reg.
E447- BA          TSX         AND #$10    ;test for BRK-instruction
E448- BD 04 01    LDA $0104,X ;if BRK
E44B- 29 10      AND #$10    BEQ PULS1    ;go do BRK-routine
E44D- F0 03      BEQ PULS1    JMP (CBINV) ;else do IRQ-routine
E44F- 6C 92 00    JMP (CBINV) ;*****
E452- 6C 90 00    PULS1      JMP (CINV)   ;*****
                           ;*
                           ;* Rest of interrupt routine; *
                           ;* do cursor blink. *
                           ;*
                           ;* (Default pointed to $90). *
                           ;*
                           ;*****  

E455- 20 68 F7    KEY         JSR UDTIM    ;to clocktick (why not use $FFEA?) e
E458- A5 A7      LDA *BLNSW    ;if cursor invisible
E45A- D0 18      BNE KEY4    ;skip blink handling
E45C- C6 A8      DEC *BLNCT    ;decrement countdown
E45E- D0 14      BNE KEY4    ;if countdown zero
E460- A9 14      LDA #20     ;get countdown

```

E462- 85 A8	STA *BLNCT	;and store it
E464- A4 C6	LDY *PNTR	;get column position
E466- 46 AA	LSR *BLNON	;determine blink 'phase
	;and set phase zero	
E468- B1 C4	LDA (PNT),Y	;get character at current position
E46A- 80 04	BCS KEY5	;if in phase zero
E46C- E6 AA	INC *BLNON	;set blink phase one
E46E- 85 A9	STA *GDBLN	;and store true character
E470- 49 80	EOR #\$80	;reverse screen character
E472- 91 C4	KEY5 STA (PNT),Y	;and put to screen
	;*****	
	;*	*
	;* Preparation for keyboard scan. *	*
	;*	*
	;*****	
E474- A2 FF	KEY4 LDX #\$FF	;this value received
E476- 86 A6	STX *SFDX	;on no key at all
E478- E8	INX	;set X to zero
E479- 86 98	STX *SHFLAG	;and clear shift indicator
E47B- A2 50	LDX #80	;this number of keys to be scanned
E47D- AD 10 E8	LDA PIAL	
E480- 29 F0	AND #\$F0	;set zero in
E482- 8D 10 E8	STA PIAL	;keyboard scan line number
	;*****	
	;*	*
	;* Cassette motor control. *	*
	;*	*
	;*****	
E485- A0 00	LDY #0	;get flag value
E487- AD 10 E8	LDA PIAL	;get cassette sense lines
E48A- 0A	ASL A	;in bits
E48B- 0A	ASL A	;6 and 7
E48C- 0A	ASL A	;of accu
E48D- 10 06	BPL KEY3	;if no key down on cassette 1
E48F- 84 F9	STY *CAS1	;flag cassette 1 off
E491- A9 30	LDA #\$30	;get value for cassette 1 off
E493- D0 06	BNE KL24	;switch off and test cass#2
E495- A5 F9	LDA *CAS1	;key down, if #1 flag on
E497- D0 05	BNE KL2	;test cassette #2
E499- A9 35	LDA #\$35	;else get value for motor#1 on
E49B- 8D 13 E8	STA PIAS	;and set cassette #1
E49E- 90 09	BCC KL23	;if key of #2 up
E4A0- 84 FA	STY *CAS2	;flag cassette #2 off
E4A2- AD 40 E8	LDA PIA	;get byte
E4A5- 09 10	ORA #\$10	;and switch cass#2 motor off
E4A7- D0 09	BNE KL25	;and store the byte
E4A9- A5 FA	KL23 LDA *CAS2	;key down, if #2 flag on
E4AB- D0 08	BNE KL22	;finish cassette control
E4AD- AD 40 E8	LDA PIA	;else
E4B0- 29 EF	AND #\$EF	;switch #2 motor on
E4B2- 8D 40 E8	KL25 STA PIA	
	.FI "BASIC4.OE9N.MOS"	

19A1 33B1-4052 BASIC4.OE9N.MOS

	;NAME BASIC4.OE9N.MOS	
	;*****	
	;*	*
	;* Scan keyboard.	*
	;*	*
	;*****	
E4B5- A0 08	KL22 LDY #8	;this number of bits in a byte
E4B7- AD 12 E8	LDA PIAK	;get key image

```

E4BA- CD 12 E8      CMP PIAK          ;wait until
E4BD- DD F6          BNE KL22          ;steady
E4BF- 4A             LSR A            ;get bit in carry
E4C0- B0 1C          BCS CKIT          ;if no key count and loop
E4C2- 48             PHA              ;if key, save image
E4C3- BD 0A E6      LDA CHAR-1,X   ;get key value
E4C6- DD 06          BNE CKIS1        ;if shift
E4C8- A9 01          LDA #1            ;set indicator
E4CA- 85 98          STA *SHFLAG      ;to .1
E4CC- DD 0F          BNE CKUT          ;and scan further
E4CE- C9 FF          CMP #$FF          ;if no key
E4D0- F0 0B          BEQ CKUT          ;if <
E4D2- C9 3C          CMP #'<'         ;if <
E4D4- DD 05          BNE SPCK          ;and cassette input
E4D6- 2C 11 E8      BIT PIAL1        ;repeat that key
E4D9- 30 02          BMI CKUT          ;key found, save current index
E4DB- 86 A6          SPCK             ;key found, save current index
E4DD- 68             CKUT             ;reget image
E4DE- CA             CKIT             ;adapt key counter
E4DF- F0 08          BEQ CKIT1        ;if all done, exit
E4E1- 88             DEY              ;adapt bit counter
E4E2- DD DB          BNE KL1           ;if not zero, test next bit
E4E4- EE 10 E8      INC PIAL          ;else do next scan line
E4E7- DD CC          BNE KL22          ;load bit counter and loop
E4E9- A5 A6          CKIT1            ;get index found
E4EB- C5 97          CMP *LSTX          ;if not same as last time
E4ED- F0 1C          BEQ PREEND        ;store index
E4EF- 85 97          STA *LSTX          ;if index negative (no key)
E4F1- AA             TAX               ;exit
E4F2- 30 17          BMI PREEND        ;else get keyvalue
E4F4- BD 0A E6      LDA CHAR-1,X   ;get shift status
E4F7- 46 98          LSR *SHFLAG      ;get shift status
E4F9- 90 02          BCC KN1            ;if pressed,
E4FB- 09 80          ORA #$80          ;add shift bit
E4FD- A6 9E          KN1               ;get buffer index
E4FF- 90 6F 02      STA KEYD,X       ;add key to buffer
E502- E8             INX               ;adapt index
E503- E0 0A          CPX #10            ;if 10 keys in buffer
E505- DD 02          BNE KEYF          ;cancel buffer
E507- A2 00          LDX #0            ;adapt index
E509- 86 9E          KEYF             ;adapt index
E50B- 4C 00 E6      PREEND            ;and return from interrupt
;*****
;*
;* The area $E50E-$E5FF is filled *
;* with $AA bytes.                 *
;*
;*****
.BA $E600
;*****
;*
;* Return from interrupt.          *
;*
;*****
E600- 68             PREND            PLA
E601- A8             TAY               ;restore Y
E602- 68             PLA
E603- AA             TAX              ;and X
E604- 68             PLA              ;and A
E605- 40             RTI               ;and return from interrupt
;*****
;*
;* Store the character in A on    *
;* screen.                      *

```

```

        ;*
        ;*****
E606- A4 C6      DSPP    LDY *PNTR          ;get column position on line
E608- 91 C4      DSPP    STA (PNT),Y       ;store the character there
E60A- 60          RTS
        ;*****
        ;*
        ;* Keyboard table, for 80 keys. *
        ;*
        ;*****
E60B- 3D 2E FF    CHAR    .BY '=' $FF $03 '<' 'C' $12 ;scan line nine
E60E- 03 3C 20
E611- 5B 12
E613- 20 30 00    .BY '-0' $00 '>' $FF 'J0' $00 ;scan line eight
E616- 3E FF 50
E619- 40 00
E61B- 2B 32 FF    .BY '+2' $FF '?,NVX' ;scan line seven
E61E- 3F 2C 4E
E621- 56 58
E623- 55 31 0D    .BY '31' $00 ';'MBCZ' ;scan line six
E626- 3B 40 42
E629- 43 5A
E62B- 2A 35 FF    .BY '*5' $FF ':KHFS' ;scan line five
E62E- 3A 4B 48
E631- 46 53
E633- 36 34 FF    .BY '64' $FF 'LJGDA' ;scan line four
E636- 4C 4A 47
E639- 44 41
E63B- 2F 38 FF    .BY '/8' $FF 'PIYRW' ;scan line three
E63E- 50 49 59
E641- 52 57
E643- 39 37 5E    .BY '97^OUTEQ' ;scan line two
E646- 4F 55 54
E649- 45 51
E64B- 14 11 FF    .BY $14 $11 $FF ')\' $27 '$' ;scan line one
E64E- 29 5C 27
E651- 24 22
E653- 1D 13 5F    .BY $1D $13 '_(&%#!' ;scan line zero
E656- 28 26 25
E659- 23 21
        ;*****
        ;*
        ;* Table of lo-bytes of screen *
        ;* line addresses.           *
        ;*
        ;*****
E65B- 00 28 50    LDTB2   .BY 0 40 80 120 160 200 240
E65E- 78 A0 C8
E661- F0
E662- 18 40 68    .BY 24 64 104 144 184 224
E665- 90 B8 E0
E668- 08 30 58    .BY 8 48 88 128 168 208 248
E66B- 80 A8 D0
E66E- F8
E66F- 20 48 70    .BY 32 72 112 152 192
E672- 98 C0
        ;*****
        ;*
        ;* Message: D shift-L /* RETURN *
        ;* RUN RETURN                   *
        ;*
        ;*****
E674- 44 CC 22    RUNTB   .BY 'D' $CC /* $00 'RUN' $00
E677- 2A 00 52

```

E67A- 55 4E 0D

```

;*****
;*                                     *
;* Checksum byte (over E-ROM).      *
;*                                     *
;*****                                *
E67D- 0E    CKSUME .BY $0E           ;checksum byte
;*****
;*                                     *
;* The rest of the ROM is filled   *
;* with $AA bytes.                 *
;*                                     *
;*****                                *
;
;*****
;*                                     *
;* External Labels.                  *
;*                                     *
;*****                                *
CTIMR  .DE $008D           ;the jiffy clock
CINV   .DE $0090           ;IRQ ram vector
CBINV  .DE $0092           ;BRK RAM vector
LSTX   .DE $0097           ;last scan index
SHFLAG .DE $0098           ;shiftkey indicator
NDX    .DE $009E           ;number of keys in keyboard buffer
RVS    .DE $009F           ;flag for reverse field printing
INDX   .DE $00A1           ;record length of current screen line
LXSP   .DE $00A3           ;cursor row/column, used in INPUT and G
SFDX   .DE $00A6           ;current scan index
BLNSW  .DE $00A7           ;flag: cursor on
BLNCT  .DE $00A8           ;countdown till next cursor blink
GDBLN  .DE $00A9           ;real character under cursor
BLNON  .DE $00AA           ;cursor's blink phase
CRSW   .DE $00AC           ;flag: input from screen or keyboard
DFLTN  .DE $00AF           ;current input device number
DFLTO  .DE $00B0           ;current output device number
PNT    .DE $00C4           ;screen line address
PNTR   .DE $00C6           ;current column position of cursor
SAL    .DE $00C7           ;pointer in screen scroll
SAH    .DE $00C8           ;pointer in screen scroll, hi
QTSW   .DE $00CD           ;flag: nonzero if in quote mode
LNMX   .DE $00D5           ;length of current screen line
TBLX   .DE $00D8           ;current line number of cursor
DATA   .DE $00D9           ;last key input
INSRT  .DE $00DC           ;number of inserts outstanding
LDTB1  .DE $00E0           ;screen line wrap around table
CAS1   .DE $00F9           ;flag: cassette #1 motor on
CAS2   .DE $00FA           ;flag: cassette #2 motor on
PI     .DE $00FF           ;value for PI
KEYD   .DE $026F           ;keyboard buffer
UOTIM  .DE $F768           ;update jiffy clock
PIAL   .DE $E810           ;keyboard PIA: I/O port A and DDR
PIAL1  .DE $E811           ;keyboard PIA: control register A
PIAK   .DE $E812           ;keyboard PIA: I/O port B and DDR
PIAS   .DE $E813           ;keyboard PIA: control register B
IEEEI  .DE $E820           ;IEEE PIA: I/O port A and DDR
IEEEIS .DE $E821           ;IEEE PIA: control register A
IEEO   .DE $E822           ;IEEE PIA: I/O port B and DDR
IEEEOS .DE $E823           ;IEEE PIA: control register B
PIA    .DE $E840           ;VIA: I/O port B
SYNC   .DE $E841           ;VIA: I/O port A with handshake
P2DB   .DE $E842           ;VIA: port B data direction register
P2DA   .DE $E843           ;VIA: port A data direction register
T1L    .DE $E844           ;VIA: timer 1 lo

```

T1H	.DE \$E845	;VIA: timer 1 hi
T1LL	.DE \$E846	;VIA: timer 1 lo, latch
T1LH	.DE \$E847	;VIA: timer 1 hi, latch
T2L	.DE \$E848	;VIA: timer 2 lo
T2H	.DE \$E849	;VIA: timer 2 hi
SR	.DE \$E84A	;VIA: shift register
ACR	.DE \$E84B	;VIA: auxiliary control register
PCR	.DE \$E84C	;VIA: peripheral control register
IFR	.DE \$E84D	;VIA: interrupt flag register
IER	.DE \$E84E	;VIA: interrupt enable register
SYNC1	.DE \$E84F	;VIA: port A without handshake
	.EN	

END MAE PASS

LABELFILE

ACR =E84B	BK1 =E23F	BK2 =E24A
BKLN =E10E	BKLN1 =E1EA	BLKL =E434
BLKLN =E421	BLNCT =00A8	BLNON =00AA
BLNSW =00A7	CAS1 =00F9	CAS2 =00FA
CBINV =0092	CHAR =E60B	CINT =E000
CINV =0090	CKIS1 =E4CE	CKIT =E4DE
CKIT1 =E4E9	CKSUME =E67D	CKUT =E4DD
CLP1 =E158	CLP2 =E141	CLP21 =E156
CLP2A =E153	CLP5 =EOF2	CLP6 =EDFB
CLP7 =E166	CLSR =E04B	CNC3 =E254
CNC3X =E230	CRSW =00AC	CTIMR =008D
DATA =00D9	DFLTN =00AF	DFLTO =0080
DSPP =E606	GDBLN =00A9	IEEI =E820
IEEIS =E821	IEE0 =E822	IEEOS =E823
IER =E84E	IFR =E84D	INDX =00A1
INS1 =E2D5	INS2 =E2D7	INS3 =E2CE
INSRT =00DC	JLP2 =E340	JPL3 =E28C
JPL4 =E275	JSTS1 =E19F	JSTS2 =E1CD
JSTSX =E1B3	JSXB =E1BD	JTS2 =E1BE
KEY =E455	KEY3 =E495	KEY4 =E474
KEY5 =E470	KEYD =026F	KEYF =E509
KL1 =E4BF	KL2 =E49E	KL22 =E4B5
KL23 =E4A9	KL24 =E49B	KL25 =E4B2
KN1 =E4FD	LDTB1 =00E0	LDTB2 =E65B
LNMX =00D5	LOOP2 =E1A6	LOOP3 =E0BF
LOOP4 =E0BC	LOOP5 =E116	LOP2 =E1AE
LOPS =E11E	LOPS1 =E124	LOPS2 =E134
LOPS3 =E138	LOPS4 =E12E	LP1 =EOAC
LP2 =EOA7	LP21 =E0D3	LP22 =EOEA
LP23 =E0DF	LPS1 =E04F	LPS2 =E05D
LPS3 =E05E	LPS4 =E06A	LSTX =0097
LXSP =00A3	MLP1 =E382	MLP2 =E39B
MLP4 =E3D2	MLP41 =E3D5	MLP42 =E3DF
NC1 =E25D	NC2 =E264	NC3 =E17A
NC3W =E257	NCX2 =E27A	NCZ2 =E277
NDX =009E	NELL =E414	NEWLA =E401
NEWLN =E3E2	NEWLX =E3F3	NEXL1 =E3F5
NJTL =E21D	NTCN =E229	NTCN1 =E250
NTCN2 =E1F3	NVS =E17C	NVSL =E182
NXLN =E343	NXLN1 =E350	NXLN2 =E348
NXT2 =E322	NXT3 =E176	NXT33 =E174
NXT6 =E32A	NXTD =E079	NXTL =E359
NXTX =E2A4	NXTX1 =E2AC	P2DA =E843
P2DB =E842	PCR =E84C	PI =00FF
PIA =E840	PIAK =E812	PIAL =E810

PIAL1 =E811	PIAS =E813	PNT =00C4
PNTR =00C6	PREND =E600	PRENDO =E50B
PRT =E202	PRT1 =E28F	PULS =E442
PULS1 =E452	PX1 =E0D9	QTSW =00CD
QTSWC =E167	QTSWL =E173	RUNTB =E674
RVS =009F	SAH =00C8	SAL =00C7
SCRL =E1C4	SCRL1 =E37B	SCRL3 =E3B5
SCRL4 =E3A7	SCRL5 =E3AB	SCROL =E369
SFDX =00A6	SHFLAG =0098	SPCK =E4DB
SR =E84A	STUPR =E0A6	STUPT =E07F
STUPZ =E09C	SYNC =E841	SYNC1 =E84F
T1H =E845	T1L =E844	T1LH =E847
T1LL =E846	T2H =E849	T2L =E848
TBLX =00D8	UDTIM =F768	UP1 =E303
UP2 =E2F3	UP3 =E312	UP5 =E2BA
UP6 =E2EE	UP9 =E2EA	e216 =E216
e2b3 =E2B3	e336 =E336	e339 =E339
e33d =E33D		
//0000,E67E,E67E		
]		

```

;NAME BASIC4.OE9B.CTL
;*****
;*          E-ROM for BASIC 4.0
;*
;*          for 9 inch screen
;*          models
;*          with
;*          B-keyboard
;*
;* Date 13-11-83  Time 19:45
;*
;* Made by:
;*
;* Nico de Vries
;* Mari Andriessenrade 49
;* 2907 MA Capelle a/d Yssel
;* The Netherlands
;*
;* Original CBM ROM number
;*
;*          1474-02
;*
;*****
.BA $E000
.FI "BASIC4.OE9B.M01"

```

16E9 33C9-4AB2 BASIC4.OE9B.M01

```

;NAME BASIC4.OE9B.M01
;*****
;*          Reset; initialize I/O.
;*
;*****
E000- A9 7F      CINT    LDA #$7F      ;disable all VIA interrupts
E002- 20 C2 E5      JSR e5c2      ;PATCH: clear tabstop table
E005- A2 6D      LDX #$6D
E007- A9 00      LDA #$00      ;clear
E009- 95 80      PX1      STA *CTIMR,X   ;area $80 to $FA
E00B- CA      DEX
E00C- 10 FB      BPL PX1
E00E- A9 55      LDA #L,KEY   ;set
E010- 85 90      STA *CINV    ;IRQ RAM vector
E012- A9 E4      LDA #H,KEY   ;to
E014- 85 91      STA *CINV+1  ;keyboard/cassette routine
E016- A9 03      LDA #3       ;set default output device
E018- 85 B0      STA *DFLTO   ;to screen
E01A- A9 0F      LDA #0F      ;keyboard PIA, A section:
E01C- 80 10 E8      STA PIAL    ;bits 0-3 output, rest input
E01F- 0A      ASL A
E020- 80 40 E8      STA PIA     ;cassette#2 motor off
; cassette #1 and #2 write = 1
; IEEE ATN=false NRFD=false
E023- 80 42 E8      STA P2DB    ;bits 7-5,0 output, 4-1 input
E026- 8E 22 E8      STX IEEE0   ;IEEE PIA, B section: all bits output
E029- 8E 45 E8      STX T1H     ;VIAT1H: maximum timeout
E02C- A9 3D      LDA #$3D
E02E- 80 13 E8      STA PIAS    ;keyboard PIA: cassette#1 motor off
; (end of screen refresh)

```

```

E031- 2C 12 E8      BIT PIAK      ;clear keyboard PIA interrupt flags
E034- A9 3C          LDA #$3C      ;IEEE PIA: NDAC=false, CA1 neg. edge
E036- 8D 21 E8      STA IEEEIS    ;no interrupts
E039- 8D 23 E8      STA IEEOS    ;IEEE PIA: DAV=false, CB1 neg. edge
;no interrupts
E03C- 8D 11 E8      STA PIAL1    ;keyboard PIA: CA2=1,CA1 neg. edge
;no interrupts
E03F- 8E 22 E8      STX IEEEO     ;IEEE PIA: all IEEE data line false
E042- A9 0E          LDA #$0E      ;VIA:CA2=1 (lower/upper case),CA1 neg.
E044- 8D 4C E8      STA PCR       ;no interrupts,CB2 input,neg. edge
E047- 85 A8          STA *BLNCT.   ;set blink count for cursor
E049- 85 A7          STA *BLNSW   ;set cursor visible
;*****
;*
;* Initialize screen (clear it). *
;*
;*****
E04B- A0 83          CLSR        LDY #$83      ;get hi-byte also
E04D- A2 18          LDX #24      ;do 25 lines
E04F- 94 E0          LPS1        STY *LDTB1,X ;store hi-byte in table
E051- E0 14          CPX #20      ;if on line 20
E053- F0 08          BEQ LPS2    ;adapt pointer
E055- E0 00          CPX #13      ;if on line 13
E057- F0 04          BEQ LPS2    ;adapt pointer
E059- E0 07          CPX #7       ;if on line 7
E05B- D0 01          BNE LPS3    ;adapt pointer
E05D- 88              LPS2        DEY         ;adapt hi-byte
E05E- CA              LPS3        DEX         ;adapt line counter
E05F- 10 EE          BPL LPS1    ;if all lines done
E061- 84 C5          STY *PNT+1   ;set pointer in ($C4)
E063- E8              INX         ;make X zero
E064- 86 9F          STX *RVS     ;set rvs off
E066- 86 C4          STX *PNT     ;and store lo-byte of pointer
E068- A9 20          LDA #$20     ;put a blank
E06A- 9D 00 80        LPS4        STA $8000,X ;into all bytes
E06D- 9D 00 81          STA $8100,X ;of screen
E070- 9D 00 82          STA $8200,X
E073- 9D 00 83          STA $8300,X
E076- E8              INX         ;until whole screen done
E077- D0 F1          BNE LPS4    ;*****
;*
;* Home cursor. *
;*
;*****
E079- A0 00          NXTD        LDY #$00
E07B- 84 C6          STY *PNTR    ;set cursor on start of line
E07D- 84 D8          STY *TBLX    ;and on first screen line
;*****
;*
;* Get pointer to current line in *
;* screen RAM. *
;*
;* Expects $D8 to contain current *
;* line number; ($C4) will be set *
;* to point to current line; $D5 *
;* will be set to max. length of *
;* that line. *
;*
;*****
E07F- A6 D8          STUPT       LDX *TBLX    ;get screen line of cursor
E081- B5 E0          LDA *LDTB1,X ;get hi-byte of that line
E083- 09 80          ORA #$80     ;set bit 7 (flag in table)
E085- 85 C5          STA *PNT+1   ;and store in pointer

```

E087-	BD 5B E6	LDA LDTB2,X	;get lo-byte from table
E08A-	85 C4	STA *PNT	;store in pointer too
E08C-	A9 27	LDA #39	;set length to 39
E08E-	85 D5	STA *LNMX	
E090-	ED 18	Cpx #24	;if not bottom line
E092-	FD 08	BEQ STUPZ	
E094-	B5 E1	LDA *LDTB1+1,X	;and bit 7 from table clear
E096-	30 04	BMI STUPZ	
E098-	A9 4F	LDA #79	;then length is 79
E09A-	85 D5	STA *LNMX	
E09C-	A5 C6	STUPZ LDA *PNTR	;get position on line
E09E-	C9 28	CMP #40	
EDAO-	90 04	BCC STUPR	
EDA2-	E9 28	SBC #40	
EDA4-	85 C6	STA *PNTR	;and store modulo 40
EDA6-	60	STUPR RTS	
;*****			
;*			
;* Get character from keyboard *			
;* buffer. *			
;*			
;*****			
EOA7-	AC 6F 02	LP2 LDY KEYD	;get oldest character
EOAA-	A2 00	LDX #\$00	
EOAC-	BD 70 02	LP1 LDA KEYD+1,X	;move rest of buffer
EOAF-	9D 6F 02	STA KEYD,X	;forward
E0B2-	E8	INX	
E0B3-	E4 9E	Cpx *NDX	;until all char's done
E0B5-	D0 F5	BNE LP1	
E0B7-	C6 9E	DEC *NDX	;adapt index in buffer
E0B9-	98	TYA	;get character in accu
E0BA-	58	CLI	;enable interrupts again
E0BB-	60	RTS	
;*****			
;*			
;* Wait for RETURN from keyboard. *			
;*			
;*****			
E0BC-	20 02 E2	LOOP4 JSR PRT	;echo character on screen
E0BF-	A5 9E	LOOP3 LDA *NDX	;get index in keyb. buffer
EDC1-	85 A7	STA *BLNSW	;if empty, set cursor visible
EOC3-	FD FA	BEQ LOOP3	;else wait for character
EOC5-	78	SEI	;disable interrupts
EOC6-	A5 AA	LDA *BLNON	;if blink fase one
EOC8-	FD 09	BEQ LP21	
EOCA-	A5 A9	LDA *GDBLN	
EOCC-	A0 00	LDY #\$00	
EOCE-	84 AA	STY *BLNON	
EODO-	20 06 E6	JSR DSPP	
EOD3-	20 A7 ED	LP21 JSR LP2	
EOD6-	C9 83	CMP #\$83	
EOD8-	D0 10	BNE LP22	
EODA-	78	SEI	
EODB-	A2 09	LDX #9	
EODD-	86 9E	STX *NDX	
EODF-	BD 73 E6	LP23 LDA RUNTB~1,X	
EDE2-	9D 6E 02	STA KEYD-1,X	
EDES-	CA	DEX	
EDE6-	D0 F7	BNE LP23	
EDE8-	FD D5	BEQ LOOP3	
EDEA-	C9 OD	CMP #\$00	
EDEC-	D0 CE	BNE LOOP4	
EDEE-	A4 D5	LDY *LNMX	
EDEF-	84 AC	STY *CRSW	

EOF2- B1 C4	CLP5	LDA (PNT),Y	;get char. from screen
EOF4- C9 20		CMP #\$20	;if space at end of line
EOF6- D0 03		BNE CLP6	
EOF8- 88		DEY	;get previous character
EOF9- D0 F7		BNE CLP5	
EOFB- C8	CLP6	INY	
EOFC- 84 A1		STY *INOX	;set record length of current line
EOFE- A0 00		LDY #\$00	
E100- 84 C6		STY *PNTR	;and set cursor at start of line
E102- 84 CD		STY *QTSW	;clear quote flag
E104- A5 A3		LDA *LXSP	;get line number on screen
E106- 30 16		BMI LOPS	;if valid
E108- C5 D8		CMP *TBLX	;and equal to original
E10A- D0 12		BNE LOPS	
E10C- A5 A4		LDA *LXSP+1	;set position
E10E- 85 C6		STA *PNTR	;to original value
E110- C5 A1		CMP *INDX	;if not behind end of record
E112- 90 0A		BCC LOPS	;get char from device 3
E114- B0 28		B.CS CLP2	;else return RETURN

		;*	
		;* Get character from device 0 *	
		;* or device 3. *	
		;*	

E116- 98	LOOPS5	TYA	
E117- 48		PHA	;save Y
E118- 8A		TXA	
E119- 48		PHA	;save X
E11A- A5 AC		LDA *CRSW	;if input from keyboard
E11C- F0 A1		BEQ LOOP3	;go do keyboard until RETURN
		.FI "BASIC4.OE9B.M02"	

1724 33C9-4AED BASIC4.OE9B.M02

		;NAME BASIC4.OE9B.M02	

		;*	
		;* Get character from current *	
		;* screen line. *	
		;*	

E11E- A4 C6	LOP5	LDY *PNTR	;get current position
E120- B1 C4		LDA (PNT),Y	;get screen character
E122- 85 D9		STA *DATA	;save it
E124- 29 3F	LOP51	AND #\$3F	;remove bits 6 and 7
E126- 06 D9		ASL *DATA	
E128- 24 D9		BIT *DATA	;if original bit 6 set
E12A- 10 02		BPL LOP54	
E12C- 09 80		ORA #\$80	;set bit 7 (shift bit)
E12E- 90 04	LOP54	BCC LOP52	;if original bit 7 set
E130- A6 C0		LDX *QTSW	;and if in quote mode
E132- D0 04		BNE LOP53	;don't change bit 6
E134- 70 02	LOP52	BVS LOP53	;else if bit 5 set in original
E136- 09 40		ORA #\$40	;set bit 6
E138- E6 C6	LOP53	INC *PNTR	;adapt position
E13A- 20 67 E1		JSR QTSWC	;adapt quote flag
E13D- C4 A1		CPY *INDX	;if max position reached
E13F- D0 17		BNE CLP1	
E141- A9 00	CLP2	LDA #\$00	
E143- 85 AC		STA *CRSW	;flag input from keyboard
E145- A9 0D		LDA #\$0D	;and echo RETURN
E147- A6 AF		LDX *DFLTN	;if input device

```

E149- E0 03      CPX #3          ;is screen
E14B- F0 06      BEQ CLP2A      ;go echo on screen
E14D- A6 80      LDX *DFLTO    ;else if output device
E14F- E0 03      CPX #3          ;is screen
E151- F0 03      BEQ CLP21      ;store RETURN in last input
E153- 20 02 E2   CLP2A JSR PRT   ;echo on screen
E156- A9 00      CLP21 LDA #$0D   ;and store a RETURN
E158- 85 D9      CLP1 STA *DATA  ;in last key input
E15A- 68          PLA
E15B- AA          TAX           ;restore X
E15C- 68          PLA
E15D- A8          TAY           ;restore Y
E15E- A5 D9      LDA *DATA      ;get character again
E160- C9 DE      CMP #$DE      ;if code for PI
E162- D0 02      BNE CLP7
E164- A9 FF      LDA #PI        ;get real code for PI
E166- 60          CLP7 RTS
;*****
;*
;* Toggle quote flag if " found.
;*
;*****
E167- C9 22      QTSWC CMP #$22  ;if character is "
E169- D0 08      BNE QTSWL
E16B- A5 CD      LDA *QTSW      ;get quote flag
E16D- 49 01      EOR #$01      ;invert bit 0
E16F- 85 CD      STA *QTSW
E171- A9 22      LDA #$22      ;and restore character
E173- 60          QTSWL RTS
;*****
;*
;* Fill screen at current
;* position.
;*
;*****
E174- 09 40      NXT33 ORA #$40  ;map shifted characters
E176- A6 9F      NXT3 LDX *RVS   ;if reverse switch on
E178- F0 02      BEQ NVS
E17A- 09 80      NC3  ORA #$80  ;set screencode for reverse
E17C- A6 DC      NVS  LDX *INSRT ;if in insert mode
E17E- F0 02      BEQ NVSL
E180- C6 DC      DEC *INSRT   ;decrease insert count
E182- 20 06 E6   NVSL JSR DSPP  ;put char at current position in RAM
E185- E6 C6      JSTS INC *PNTR  ;advance character position
E187- A4 D5      LDY *LNMX
E189- C4 C6      CPY *PNTR
E18B- B0 19      BCS LOOP2
E18D- A6 D8      LDX *TBLX      ;get line number in X
E18F- C0 4F      CPY #79      ;and at position 79
E191- D0 0C      BNE JSTS1
E193- 20 B3 E1   JSR JSTSX    ;set current line to extended
E196- 20 43 E3   JSR NXLN    ;go to start of next line
E199- A9 00      LDA #0       ;and set cursor at start
E19B- 85 C6      STA *PNTR    ;of that line
E19D- F0 07      BEQ LOOP2
E19F- E0 18      JSTS1 CPX #24  ;if beyond length, but <79
E1A1- D0 1B      BNE JTS2    ;and on last line
E1A3- 20 C4 E1   JSR SCRL   ;scroll screen up (leave cursor on same
;*****
;*
;* Exit from output to screen.
;*
;*****
E1A6- 68          LOOP2 PLA

```

E1A7- A8		TAY	;restore Y
E1A8- A5 DC		LDA *INSRT	;if in insert mode
E1AA- F0 02		BEQ LOP2	
E1AC- 46 CD		LSR *QTSW	;set quote mode off
E1AE- 68	LOP2	PLA	
E1AF- AA		TAX	;restore X
E1B0- 68		PLA	;and A
E1B1- 58		CLI	;allow interrupts again
E1B2- 60		RTS	

		;*	
		;* Set next line to *	
		;* 'not-extended'. *	
		;* *	

E1B3- E0 17	JSTSX	CPX #23	;if line number <=23
E1B5- B0 06		BCS JSXB	
E1B7- B5 E2		LDA *LDTB1+2,X	;set next line
E1B9- 09 80		ORA #\$80	;to 'not-extended'
E1BB- 95 E2		STA *LDTB1+2,X	
E1BD- 60	JSXB	RTS	

		;*	
		;* Insert blank line if beyond *	
		;* current length, but not at end *	
		;* of line. *	
		;*	

E1BE- 20 CD E1	JTS2	JSR JTS2	;if beyond length but at pos. 79
		;insert new line	
E1C1- 4C A6 E1		JMP LOOP2	;and exit

		;*	
		;* Get or insert new line. *	
		;*	

E1C4- 20 69 E3	SCRL	JSR SCROL	;scroll lines on screen
E1C7- C6 A3		DEC *LXSP	;decrement copy of line number
E1C9- C6 D8		DEC *TBLX	;and line# itself (cursor on same line)
E1CB- A6 D8		LDX *TBLX	;get line number
E1CD- 16 E1	JTS2	ASL *LDTB1+1,X	
E1CF- 56 E1		LSR *LDTB1+1,X	;set line to 'extended'
E1D1- 20 B3 E1		JSR JSTSX	;and next line to 'not-extended'
E1D4- A5 C6		LDA *PNTR	;save position
E1D6- 48		PHA	
E1D7- 20 7F E0		JSR STUPT	;and set pointer
E1DA- 68		PLA	
E1DB- 85 C6		STA *PNTR	;and restore position
E1DD- 60		RTS	

		;*	
		;* Backward over line boundary. *	
		;*	

E1DE- A0 27	BKLN	LDY #39	
E1E0- A6 D8		LDX *TBLX	;if current line
E1E2- D0 06		BNE BKLN1	;is top line
E1E4- 86 C6		STX *PNTR	;set cursor home
E1E6- 68		PLA	
E1E7- 68		PLA	;remove return address
E1E8- D0 BC		BNE LOOP2	;and exit
E1EA- B5 DF	BKLN1	LDA *LDTB1-1,X	;if not on top line
E1EC- 30 05		BMI NTCN2	;and line is 'extended'
E1EE- CA		DEX	

E1EF- B5 DF		LDA *LDTB1-1,X	;get indicator of next line
E1F1- A0 4F		LDY #79	;and set position to 79
E1F3- CA	NTCN2	DEX	;adapt line number
E1F4- 86 D8		STX *TBLX	;and store it
E1F6- 85 C5		STA *POINT	;and store new pointer
E1F8- B0 5B E6		LDA LDTB2,X	;store lo-byte
E1FB- 85 C4		STA *PNT	;as well
E1FD- 84 C6		STY *PNTR	;set cursor at end of line
E1FF- 84 D5		STY *LNMX	;and set length to 79
E201- 60		RTS	
;*****			
;* *			
;* Put character to screen. *			
;* *			
;*****			
E202- 48	PRT	PHA	;save A
E203- 85 D9		STA *DATA	;also in temporary field
E205- 8A		TXA	
E206- 48		PHA	;save X
E207- 98		TYA	
E208- 48		PHA	;and Y
E209- A9 00		LDA #0	
E20B- 85 AC		STA *CRSW	;set copy of position to zero
E20D- A4 C6		LDY *PNTR	;get position
E20F- A5 D9		LDA *DATA	;get character again
E211- 10 03		BPL e216	;if shifted
E213- 4C A4 E2		JMP NXTX	;go do shifted character
E216- C9 0D	e216	CMP #\$00	;if RETURN
E218- D0 03		BNE NJTL	
E21A- 4C 59 E3		JMP NXTL	;go do RETURN
E21D- C9 20	NJTL	CMP #\$20	;if printable
E21F- 90 08		BCC NTCN	
E221- 29 3F		AND #\$3F	;convert to screen code
E223- 20 67 E1		JSR QTSWC	;check for quote mode
E226- 4C 76 E1		JMP NXT3	;and put to screen
E229- A6 DC	NTCN	LDX *INSRT	;if inserts outstanding
E22B- F0 03		BEQ CNC3X	
E22D- 4C 7A E1		JMP NC3	;print as control character
E230- C9 14	CNC3X	CMP #\$14	;if DELETE
E232- D0 1C		BNE NTCN1	
E234- 88		DEY	;decrement character position
E235- 84 C6		STY *PNTR	;and store it
E237- 10 06		BPL BK1	;if gone to previous line
E239- 20 DE E1		JSR BKLN	;go do appropriate action
E23C- 4C 4A E2		JMP BK2	;and insert blank at end
.FI "BASIC4.OE9B.M03"			

1300 33C9-46C9 BASIC4.OE9B.M03

;NAME BASIC4.OE9B.M03			
E23F- C8	BK1	INY	;else
E240- B1 C4		LDA (PNT),Y	;compress
E242- 88		DEY	;line
E243- 91 C4		STA (PNT),Y	;upto
E245- C8		INY	;end of
E246- C4 C5		CPY *POINT	;line
E248- D0 F5		BNE BK1	
E24A- A9 20	BK2	LDA #\$20	;and insert blank
E24C- 91 C4		STA (PNT),Y	;at end
E24E- D0 3C		BNE JPL3	;and exit
E250- A6 CD	NTCN1	LDX *QTSW	;if in quote mode
E252- F0 03		BEQ NC3W	
E254- 4C 7A E1	CNC3	JMP NC3	;print as control character

```

E257- C9 12      NC3W    CMP #$12          ;if RVS-ON
E259- D0 02      NC1     BNE NC1
E25B- 85 9F      STA *RVS           ;set reverse flag
E25D- C9 13      NC1     CMP #$13          ;if HOME
E25F- D0 03      BNE NC2
E261- 20 79 E0    JSR NXTD           ;set cursor home
E264- C9 10      NC2     CMP #$10          ;if cursor-RIGHT
E266- D0 12      BNE NCX2
E268- C8         INY
E269- 84 C6      STY *PNTR          ;advance cursor
E26B- 88         DEY
E26C- C4 C5      CPY *POINT         ;if at line end
E26E- 90 07      BCC NCZ2
E270- 20 43 E3    JSR NXLN           ;go to next line
E273- A0 00      LDY #$00          ;set cursor at beginning
E275- 84 C6      JPL4    STY *PNTR          ;store position
E277- 4C A6 E1    NCZ2   JMP LOOP2          ;and exit
E27A- 4C 49 E5    NCX2   JMP e549           ;PATCH: test other control char's
E27D- EA         NOP
E27E- 18         e27e   CLC
E27F- 98         TYA
E280- 69 28      ADC #40           ;add 40
E282- A8         TAY           ;put in Y again
E283- C5 05      CMP *LNMX          ;if not beyond length
E285- 90 EE      BCC JPL4           ;go store position
E287- F0 EC      BEQ JPL4           ;if at end also
E289- 20 43 E3    JSR NXLN           ;else goto next line
E28C- 4C A6 E1    JPL3   JMP LOOP2          ;and exit
;*****
;*
;* This code is not used, left *
;* over from previous generations *
;* of CBM BASIC (not used then *
;* too). *
;*
;*****
E28F- E8         PRT1   INX
E290- 85 D8      STA *TBLX          ;get position in A
E292- 98         TYA           ;set to previous line
E293- E9 28      SBC #40           ;store it
E295- 85 C6      STA *PNTR          ;adapt line number
E297- E6 D8      PRT2   INC *TBLX          ;get pointer for topline
E299- AD 5B E6    LDA LDTB2          ;store lo
E29C- 85 C4      STA *PNT            ;store hi also
E29E- A5 E0      LDA *LDTB1          ;store it
E2A0- 85 C5      STA *POINT          ;and exit
E2A2- D0 E8      BNE JPL3
;*****
;*
;* Put shifted characters to *
;* screen. *
;*
;*****
E2A4- 29 7F      NXTX   AND #$7F          ;remove shift bit
E2A6- C9 7F      CMP #PI-$80          ;if code for PI
E2A8- D0 02      BNE NXTX1
E2AA- A9 5E      LDA #$5E           ;get real code for PI
E2AC- C9 20      NXTX1  CMP #$20           ;if printable
E2AE- 90 03      BCC e2b3
E2B0- 4C 74 E1    JMP NXT33          ;go fill screen
E2B3- C9 00      e2b3   CMP #$00           ;if shift-RETURN
E2B5- D0 03      BNE UPS
E2B7- 4C 59 E3    JMP NXTL           ;go do RETURN
E2BA- A6 CD      UPS    LDX *QTSW          ;if quote mode

```

E2BC-	D0 30	BNE UP6	;print as control character	
E2BE-	C9 14	CMP #\$14	;no quote mode, if INSERT	
E2CO-	D0 28	BNE UP9		
E2C2-	A4 D5	LDY *LNMX	;get line length	
E2C4-	B1 C4	LDA (PNT),Y	;get last char. of line	
E2C6-	C9 20	CMP #\$20	;if blank	
E2C8-	D0 04	BNE INS3		
E2CA-	C4 C6	CPY *PNTR	;and not on last position	
E2CC-	D0 07	BNE INS1	;then enough space on line	
E2CE-	CO 4F	INS3	CPY #79	;if last char. not blank
E2D0-	F0 BA	BEQ JPL3	;and length is 79, ignore	
E2D2-	20 E2 E3	JSR NEWLN	;else scroll rest of screen down	
E2D5-	A4 D5	INS1	LDY *LNMX	;get new length
E2D7-	88	INS2	DEY	;move
E2D8-	B1 C4	LDA (PNT),Y	;characters	
E2DA-	C8	INY	;one position	
E2DB-	91 C4	STA (PNT),Y	;further	
E2DD-	88	DEY	;until	
E2DE-	C4 C6	CPY *PNTR	;all done	
E2EO-	D0 F5	BNE INS2		
E2E2-	A9 20	LDY #\$20	;and put a blank	
E2E4-	91 C4	STA (PNT),Y	;in space opened up	
E2E6-	E6 DC	INC *INSRT	;adapt insert counter	
E2E8-	D0 56	BNE JLP2	;and exit	
E2EA-	A6 DC	UP9	LDX *INSRT	;if inserts outstanding
E2EC-	F0 05	BEQ UP2		
E2EE-	09 40	UP6	ORA #\$40	;convert to screen code
E2FO-	4C 7A E1	JMP NC3	;and print as control char.	
E2F3-	C9 11	UP2	CMP #\$11	;if cursor-UP
E2F5-	D0 2B	BNE NXT2		
E2F7-	A5 C6	LDA *PNTR		
E2F9-	C9 28	CMP #40	;and current position >40	
E2FB-	90 06	BCC UP1		
E2FD-	E9 28	SBC #40	;move cursor up	
E2FF-	85 C6	STA *PNTR		
E301-	B0 30	BCS JLP2		
E303-	A6 D8	UP1	LDX *TBLX	;and exit
E305-	F0 39	BEQ JLP2	;get line number	
E307-	B5 DF	LDA *LDTB1-1,X	;if top line, exit	
E309-	10 07	BPL UP3	;if previous line 'not-extended'	
E30B-	C6 D8	DEC *TBLX		
E30D-	20 7F EO	JSR STUPT	;decrement line number	
E310-	90 2E	BCC JLP2	;adapt pointer	
E312-	CA	UP3	DEX	;and exit
E313-	CA	DEX	;if previous line 'extended'	
E314-	86 D8	STX *TBLX	;decrement line number twice	
E316-	20 7F EO	JSR STUPT	;and store it	
E319-	A5 C6	LDA *PNTR	;get pointer	
E31B-	18	CLC	;get position	
E31C-	69 28	ADC #40		
E31E-	85 C6	STA *PNTR	;and set cursor on second	
E320-	D0 1E	BNE JLP2	;part of that line	
E322-	C9 12	NXT2	CMP #\$12	;and exit
E324-	D0 04	BNE NXT6	;if RVS-off	
E326-	A9 00	LDA #\$00		
E328-	85 9F	STA *RVS	;reset reverse flag	
E32A-	C9 10	NXT6	CMP #\$10	;if not cursor-LEFT
E32C-	D0 0B	BNE e339	;check rest of char's	
E32E-	88	DEY	;else	
E32F-	84 C6	STY *PNTR	;decrement position	
E331-	10 0D	BPL JLP2	;exit, if still on same line	
E333-	20 DE E1	JSR BKLN	;else do backwards over boundary	
E336-	4C A6 E1	e336	JMP LOOP2	;and exit
E339-	4C 87 E5	e339	JMP e587	;PATCH: check other characters

```

E33C- EA      e33c    NOP          ;needed because of patch
E33D- 20 4B E0 e33d    JSR CLSR     ;clear the screen
E340- 4C A6 E1 JLP2    JMP LOOP2    ;and exit
;***** *
;* Set next line number. *
;* *
;***** *
E343- 38      NXLN    SEC          ;set copy of line number invalid
E344- 46 A3      LSR *LXSP     ;BUG: wrong type of shift
E346- A6 D8      LDX *TBLX     ;get line number
E348- E8      NXLN2   INX          ;increment it
E349- E0 19      CPX #25      ;if at maximum line number
E34B- 00 03      BNE NXLN1   ;scroll screen up
E34D- 20 69 E3      JSR SCROL   ;if line is 'extended'
E350- B5 E0      NXLN1   LDA *LDTB1,X ;scroll another time
E352- 10 F4      BPL NXLN2   ;store line number
E354- 86 D8      STX *TBLX     ;and get pointer of current line
E356- 4C 7F E0      JMP STUPT    ;Action for RETURN.
;***** *
;* *
;***** *
E359- A9 00      NXTL    LDA #0       ;cancel inserts outstanding
E35B- 85 DC      STA *INSRT   ;reset reverse flag
E35D- 85 9F      STA *RVS     ;cancel quote mode
E35F- 85 CD      STA *QTSW    ;set cursor at beginning of line
E361- 85 C6      STA *PNTR    ;set next line number
E363- 20 43 E3      JSR NXLN    ;and exit
E366- 4C A6 E1      JMP LOOP2    .FI "BASIC4.OE9B.M04"

```

1A69 33C9-4E32 BASIC4.OE9B.M04

```

;NAME BASIC4.OE9B.M04
;*****
;* Scroll screen up. *
;* *
;*****
E369- A0 00      SCROL   LDY #0       ;get lo-byte of first line
E36B- 84 C4      SCROL   STY *PNT    ;store it
E36D- A9 80      SCROL   LDA #$80    ;get hi-byte also
E36F- 85 C8      SCROL   STA *SAH    ;store in source pointer
E371- 85 C5      SCROL   STA *POINT  ;and in destination pointer
E373- A9 28      SCROL   LDA #40    ;get lo of source pointer
E375- 24 E1      SCROL   BIT *LDTB1+1 ;if 1st line is 'extended'
E377- 30 02      SCROL   BMI SCRL1  ;get new lo byte
E379- A9 50      SCRL1   LDA #80    ;and complete pointers
E37B- 85 C7      SCRL1   STA *SAL    ;set screen invisible
E37D- A9 34      SCRL1   LDA #34    ;BUG: hardware to do this missing
E37F- 80 11 E8      MLP1    STA PIAL1  ;get char of next line
E382- B1 C7      MLP1    LDA (SAL),Y ;move it to current line
E384- 91 C4      MLP1    STA (PNT),Y ;until
E386- C8          MLP1    INY        ;whole page done
E387- D0 F9      MLP1    BNE MLP1    ;adapt source pointer hi
E389- E6 C8      MLP1    INC *SAH    ;and destination pointer hi
E38B- E6 C5      MLP1    INC *POINT  ;if last line
E38D- A9 84      MLP1    LDA #$84    ;not reached yet
E38F- C5 C8      MLP1    CMP *SAH    ;loop
E391- D0 EF      MLP1    BNE MLP1    ;else adapt pointer
E393- A9 E8      MLP1    LDA #$E8

```

E395- 85 C4	STA *PNT	;to point
E397- C6 C5	DEC *POINT	;beyond last screen line
E399- A9 20	LDA #\$20	;get a blank
E39B- C6 C4	MLP2 DEC *PNT	;adapt pointer
E39D- C6 C7	DEC *SAL	;use scrolled portion as counter
E39F- 91 C4	STA (PNT),Y	;fill bottom line with spaces
E3A1- DD F8	BNE MLP2	
E3A3- A2 19	LDX #25	;get line number
E3A5- 86 D8	STX *TBLX	;store it
E3A7- A2 00	SCRL4 LDX #0	;start at first line
E3A9- C6 D8	DEC *TBLX	;adapt line number
E3AB- B5 E0	SCRL5 LDA *LDTB:1,X	;get extension flag of 1st line
E3AD- 29 7F	AND #\$7F	;remove flag bit
E3AF- B4 E1	LDY *LDTB:1+1,X	;get flag of next line
E3B1- 10 02	BPL SCRL3	;if not extended
E3B3- 09 80	ORA #\$80	;add flag to previous line
E3B5- 95 E0	SCRL3 STA *LDTB:1,X	;and store
E3B7- E8	INX	;adapt line number
E3B8- ED 19	CPX #25	;if not all done
E3B9- DD EF	BNE SCRL5	;loop
E3BC- A9 83	LDA #\$83	;set last line
E3BE- 85 F8	STA *LDTB:1+24	;to 'not extended'
E3C0- A5 E0	LDA *LDTB:1	;if top line was extended
E3C2- 10 E3	BPL SCRL4	;move all flags 1 extra line
E3C4- A9 3C	LDA #\$3C	;set screen visible again
E3C6- 8D 11 E8	STA PIAL1	;BUG: hardware missing
	;BUG: sends EOI on IEEE	
E3C9- A9 FE	LDA #\$FE	;get key image of RVS key
E3CB- CD 12 E8	CMP PIAK	;if RVS pressed
E3CE- DD 0F	BNE MLP42	
E3D0- AD 08	LDY #8	;get counter
E3D2- 8D 45 E8	MLP4 STA T1H	;load timer hi (and start it)
E3D5- 2C 4D E8	MLP41 BIT IFR	;and wait until
E3D8- 50 FB	BVC MLP41	;timed out
E3DA- 88	DEY	;adapt counter
E3DB- DD F5	BNE MLP4	;if not all done, loop
E3DD- 84 9E	STY *NDX	;then cancel keyboard buffer
E3DF- A5 D8	MLP42 LDA *TBLX	;get screenline number
E3E1- 60	RTS	;and exit

	;*	
	;* Scroll rest of screen down *	
	;* and insert a blank line on	
	;* cursor's line.	
	;*	

E3E2- A6 D8	NEWLN LDX *TBLX	;get current line number
E3E4- E8	INX	
E3E5- A9 34	LDA #\$34	;set screen invisible
E3E7- 8D 11 E8	STA PIAL1	;BUG: hardware to do this missing
E3EA- E0 18	CPX #24	;if on last line but one
E3EC- FD 33	BEQ BLKLN	;blank this line
E3EE- 90 03	BCC NEWLX	;if on last line
E3F0- 4C C4 E1	JMP SCRL	;insert a line
E3F3- A2 17	NEWLX LDX #23	;start with bottom line
E3F5- B5 E1	NEXL1 LDA *LDTB:1+1,X	;get flag of next line
E3F7- 09 80	ORA #\$80	;add MSB
E3F9- 85 C8	STA *SAH	;and set up pointer
E3FB- B4 E0	LDY *LDTB:1,X	;get previous linemarker
E3FD- 30 02	BMI NEWLA	;if 'extended'
E3FF- 29 7F	AND #\$7F	;remove MSB
E401- 95 E1	NEWLA STA *LDTB:1+1,X	;and put in table
E403- 98	TYA	;move marker of previous line
E404- 09 80	ORA #\$80	;add MSB

E406- 85 C5		STA *POINT	;and set up pointer
E408- A0 27		LDY #39	;40 characters to be moved
E40A- BD 5C E6		LDA LDTB2+1,X	;get lo-byte of lower line
E40D- 85 C7		STA *SAL	;complete pointer
E40F- BD 5B E6		LDA LDTB2,X	;get lo-byte of upper line
E412- 85 C4		STA *PNT	;complete pointer
E414- B1 C4	NELL	LDA (PNT),Y	;get character of upper line
E416- 91 C7		STA (SAL),Y	;store in lower line
E418- 88		DEY	
E419- 10 F9		BPL NELL	;repeat until all char.s moved
E41B- CA		DEX	;if not all lines done
E41C- E4 D8		CPX *TBLX	
E41E- D0 D5		BNE NEXL1	;repeat for other lines
E420- E8		INX	
E421- B5 E0	BLKLN	LDA *LDTB1,X	;get marker of top line
E423- 09 80		ORA #\$80	;add MSB
E425- 85 C5		STA *POINT	;set up pointer
E427- 29 7F		AND #\$7F	;set line 'extended'
E429- 95 E0		STA *LDTB1,X	;into table
E42B- BD 5B E6		LDA LDTB2,X	;get lo-byte
E42E- 85 C4		STA *PNT	;complete pointer
E430- A0 27		LDY #39	
E432- A9 20		LDA #\$20	;get blank
E434- 91 C4	BLKL	STA (PNT),Y	;fill top line with blanks
E436- 88		DEY	
E437- 10 F8		BPL BLKL	;until whole line filled
E439- A9 3C		LDA #\$3C	;set screen visible again
E43B- 8D 11 E8		STA PIAL1	;BUG: sends EOI on IEEE
			;BUG: hardware to do this missing
E43E- 58		CLI	;enable interrupts again
E43F- 4C 7F E0		JMP STUPT	;and get pointer to current line

			/*
			;* Default interrupt routine,
			;* entered 60 times per second.
			/*
			;* (Pointed to by ROM).
			/*

E442- 48	PULS	PHA	;save A
E443- 8A		TXA	
E444- 48		PHA	;save X
E445- 98		TYA	
E446- 48		PHA	;and Y
E447- BA		TSX	
E448- BD 04 01		LDA \$0104,X	;get saved status reg.
E44B- 29 10		AND #\$10	;test for BRK-instruction
E44D- F0 03		BEQ PULS1	;if BRK
E44F- 6C 92 00		JMP (CBINV)	;go do BRK-routine
E452- 6C 90 00	PULS1	JMP (CINV)	;else do IRQ-routine

			/*
			;* Rest of interrupt routine;
			;* do cursor blink.
			/*
			;* (Default pointed to by \$90).
			/*

E455- 20 68 F7	KEY	JSR UDTIM	;to clocktick (why not use \$FFEA?)
E458- A5 A7		LDA *BLNSW	;if cursor_invisible
E45A- D0 21		BNE KEY4	;skip blink handling
E45C- C6 A8		DEC *BLNCT	;decrement countdown
E45E- D0 1D		BNE KEY4	;if countdown zero
E460- 4E F8 03		LSR \$03F8	;get repeat status

E463- 90 04	BCC e469	;if REPEAT pressed
E465- A9 02	LDA #2	;get short countdown
E467- D0 02	BNE e46b	;and store
E469- A9 14	e469 LDA #20	;else get long countdown
E46B- 85 A8	e46b STA *BLNCT	
E46D- A4 C6	LDY *PNTR	;get column position
E46F- 46 AA	LSR *BLNON	;determine blink phase
	;and set phase zero	
E471- B1 C4	LDA (PNT),Y	;get character at current position
E473- B0 04	BCS KEYS	;if in phase zero
E475- E6 AA	INC *BLNON	;set blink phase one
E477- 85 A9	STA *GDBLN	;and store true character
E479- 49 80	KEY5 EOR #\$80	;reverse screen character
E47B- 91 C4	STA (PNT),Y	;and put to screen
	;*****	
	;*	*
	;* Preparation for keyboard scan. *	
	;*	*
	;*****	
E47D- A2 FF	KEY4 LDX #\$FF	;this value received
E47F- 86 A6	STX *SFDX	;on no key at all
E481- E8	INX	;set X to zero
E482- 86 98	STX *SHFLAG	;and clear shift indicator
E484- A2 50	LDX #80	;this number of keys to be scanned
E486- AD 10 E8	LDA PIAL	
E489- 29 F0	AND #\$FO	;set zero in
E48B- 8D 10 E8	STA PIAL	;keyboard scan line number
	;*****	
	;*	*
	;* Cassette motor control. *	
	;*	*
	;*****	
E48E- AD 00	LDY #0	;get flag value
E490- AD 10 E8	LDA PIAL	;get cassette sense lines
E493- 0A	ASL A	;in bits
E494- 0A	ASL A	;6 and 7
E495- 0A	ASL A	;of accu
E496- 10 06	BPL KEY3	;if no key down on cassette 1
E498- 84 F9	STY *CAS1	;flag cassette 1 off
E49A- A9 30	LDA #\$3D	;get value for cassette 1 off
E49C- D0 06	BNE KL24	;switch off and test cass#2
E49E- A5 F9	KEY3 LDA *CAS1	;key down, if #1 flag on
E4A0- D0 05	BNE KL2	;test cassette #2
E4A2- A9 35	LDA #\$35	;else get value for motor#1 on
E4A4- 8D 13 E8	KL24 STA PIAS	;and set cassette #1
E4A7- 90 09	KL2 BCC KL23	;if key of #2 up
E4A9- 84 FA	STY *CAS2	;flag cassette #2 off
E4AB- AD 40 E8	LDA PIA	;get byte
E4AE- 09 10	ORA #\$10	;and switch cass#2 motor off
E4B0- D0 09	BNE KL25	;and store the byte
E4B2- A5 FA	KL23 LDA *CAS2	;key down, if #2 flag on
E4B4- D0 0D	BNE KL22	;finish cassette control
	;BUG: does not clear repeat flag	
E4B6- AD 40 E8	LDA PIA	;else
E4B9- 29 EF	AND #\$EF	;switch #2 motor on
E4BB- 8D 40 E8	KL25 STA PIA	
	.FI "BASIC4.OE9B.MOS"	

16FA 33C9-4AC3 BASIC4.OE9B.MOS

```
;NAME BASIC4.OE9B.MOS
;*****
;*
```

```

;* Scan keyboard. *
;*
;*****
E4BE- A9 00      e4be    LDA #0          ;clear
E4C0- 8D F8 03    STA $03F8       ;repeat status
E4C3- A0 08      KL22     LDY #8          ;this number of bits in a byte
E4C5- AD 12 E8    LDA PIAK       ;get key image
E4C8- CD 12 E8    CMP PIAK       ;wait until
E4CB- D0 F6      BNE KL22       ;steady
E4CD- 4A         KL1      LSR A          ;get bit in carry
E4CE- B0 1C      BCS CKIT       ;if no key count and loop
E4D0- 48         PHA      ;if key, save image
E4D1- B0 0A E6    LDA CHAR-1,X  ;get key value
E4D4- D0 07      BNE CKIS1      ;if shift
E4D6- A9 01      LDA #1          ;set indicator
E4D8- 85 98      STA *SHFLAG    ;to i
E4DA- 4C EB E4    JMP CKUT       ;and scan further
E4DD- C9 10      CKIS1     CMP #16        ;no shift, if REPEAT pressed

E4DF- D0 08      BNE SPCK      ;flag REPEAT
E4E1- A9 03      LDA #3          ;and scan further
E4E3- 8D F8 03    STA $03F8       ;key found, save current index
E4E6- 4C EB E4    JMP CKUT       ;reget image
E4E9- 86 A6      SPCK      STX *SFDX      ;adapt key counter
E4EB- 68         CKUT      PLA           ;if all done, exit
E4EC- CA         CKIT      DEX           ;adapt bit counter
E4ED- F0 08      BEQ CKIT1      ;if not zero, test next bit
E4EF- 88         DEY      ;else do next scan line
E4F0- D0 DB      BNE KL1        ;load bit counter and loop
E4F2- EE 10 E8    INC PIAL       ;get index found
E4F5- D0 CC      BNE KL22       ;if same as last time
E4F7- A5 A6      LDA *SFDX      ;get repeat status
E4F9- C5 97      CMP *LSTX      ;if no bit rotated out, exit
E4FB- D0 16      BNE e513       ;else adapt countup
E4FD- 4E F8 03    LSR $03F8      ;get count up
E500- 90 44      BCC PRENDO    ;if not 4 interrupts waited,
E502- EE F9 03    INC $03F9      ;exit
E505- AD F9 03    LDA $03F9      ;else store new count up
E508- C9 04      CMP #4          ;get last index
E50A- 30 3A      BMI PRENDO    ;store index
E50C- A9 00      LDA #0          ;if index negative (no key)
E50E- 8D F9 03    STA $03F9      ;exit
E511- A5 97      LDA *LSTX      ;else get keyvalue
E513- 85 97      e513      STA *LSTX      ;save flags
E515- AA         TAX      ;remove shift bit
E516- 30 2E      BMI PRENDO    ;get flags, if value was negative
E518- B0 0A E6    LDA CHAR-1,X  ;it has no shifted value, go add
E51B- 08         PHP      ;get shift status
E51C- 29 7F      AND #$7F      ;if not pressed, go add to buffer
E51E- 28         PLP      ;concerns all ctl. chars and space)
E51F- 30 17      BMI KN1        ;if lower than comma
E521- 46 98      LSR *SHFLAG    ;go add shiftbit
E523- 90 13      BCC KN1        ;or higher than ;
E525- C9 2C      CMP #',.'      ;add shiftbit as well
E527- 90 00      BCC e536       ;(concerns all letters)
E529- C9 3C      CMP #'<'       ;else convert to shift value
E52B- B0 09      BCS e536       ;BUG: wrong results if IRQ entered
E52D- E9 0F      SBC #$0F      ;with decimal flag set.
E52F- C9 20      CMP #$20      ;if now higher than space
E531- B0 05      BCS KN1        ;(concerns ,-. numbers ;)
E533- B0 05      BCS KN1        ;go put in buffer

```

E533- 69 20.		ADC #\$20 ;else convert to shift value ;BUG: wrong results if IRQ entered. ; with decimal flag set.
E535- 2C		.BY \$2C ;skip next instruction
E536- 09 80	e536	ORA #\$80 ;add shift bit
E538- A6 9E	KN1	LDX *NDX ;get buffer index
E53A- 90 6F 02		STA KEYD,X ;add key to buffer
E53D- E8		INX ;adapt index
E53E- E0 0A		CPX #10 ;if 10 keys in buffer
E540- D0 02		BNE KEYF
E542- A2 00		LDX #0 ;cancel buffer
E544- 86 9E	KEYF	STX *NDX ;adapt index
E546- 4C 00 E6	PRENDO	JMP PREND ;and return from interrupt
		;***** ;* ;* Patch: Do rest of unshifted * ;* keys. * ;* ;*****
E549- C9 11	e549	CMP #\$11 ;if cursor UP
E54B- D0 03		BNE e550
E54D- 4C 7E E2		JMP e27e ;go do cursor UP
E550- C9 09	e550	CMP #9 ;if not TAB
E552- D0 30		BNE e584 ;exit (ignore key)
E554- 20 A1 E5		JSR e5a1 ;get bitmask and offset in TAB table
E557- AC 3A 03	e557	LDY \$033A ;get column number
E55A- EE 3A 03		INC \$033A ;adapt it
E55D- C4 D5		CPY *LNMX ;if beyond line length
E55F- 90 09		BCC e56a
E561- A5 D5		LDA *LNMX ;get last column number
E563- 85 C6		STA *PNTR ;and put cursor there
E565- CE 3A 03		DEC \$033A ;adapt column number again
E568- D0 1A		BNE e584 ;and exit
E56A- 0E 3E 03	e56a	ASL \$033E ;else
E56D- D0 0A		BNE e579 ;if byte exceeded
E56F- E8		INX ;adapt index
E570- E0 0A		CPX #10 ;if beyond last index possible
E572- F0 10		BEQ e584 ;do nothing
E574- A9 01		LDA #1 ;else get new bitmask
E576- 80 3E 03		STA \$033E ;store it
E579- B0 EE 03	e579	LDA \$03EE,X ;get tabstop byte
E57C- 20 3E 03		AND \$033E ;apply mask
E57F- F0 D6		BEQ e557 ;if not at tabstop, try next
E581- C8		INY ;else calculate column
E582- 84 C6		STY *PNTR ;and put cursor there
E584- 4C A6 E1	e584	JMP LOOP2 ;and exit
		;***** ;* ;* Patch: Do rest of shifted * ;* characters. * ;* ;*****
E587- C9 13	e587	CMP #\$13 ;if CLR screen
E589- D0 03		BNE e58e
E58B- 4C 30 E3		JMP e33d ;go clear screen
E58E- C9 09	e58e	CMP #9 ;if not shift-TAB
E590- D0 0C		BNE e59e ;exit (ignore key)
E592- 20 A1 E5		JSR e5a1 ;else get mask and index in table
E595- B0 EE 03		LDA \$03EE,X ;get tabstop byte
E598- 40 3E 03		EOR \$033E ;invert the mask bit (set or
E59B- 90 EE 03		STA \$03EE,X ;remove tabstop)
E59E- 4C A6 E1	e59e	JMP LOOP2 ;and exit
		;***** ;* ;*****

```

    ;* Get bitmask for TAB table in    *
    ;* $033E and offset in table in    *
    ;* X.                                *
    ;*                                         *
    ;*****                                         *
E5A1- A5 C6      e5a1    LDA *PNTR           ;get column #
E5A3- 29 F8      AND #$F8            ;remove 3 lowest bits
E5A5- 80 3A 03    STA $033A           ;save it
E5A8- 4A          LSR A              *
E5A9- 4A          LSR A              *
E5AA- 4A          LSR A              ;divide it by 8
E5AB- AA          TAX                ;and move index to X
E5AC- A9 01      LDA #1              ;get initial bitmask
E5AE- 80 3E 03    STA $033E           *
E5B1- A4 C6      LDY *PNTR           ;get current cursor position
E5B3- CC 3A 03    CPY $033A           ;if same as position searched
E5B6- F0 09      BEQ e5c1            ;exit
E5B8- 0E 3E 03    ASL $033E           ;else adapt bitmask
E5BB- EE 3A 03    INC $033A           ;and count bits
E5BE- 4C B3 E5    JMP e5b3            ;and loop
E5C1- 60          RTS                *
    ;*****                                         *
    ;*                                         *
    ;* Patch routine: clear TAB             *
    ;* table.                            *
    ;*                                         *
    ;*****                                         *
E5C2- 8D 4E E8    e5c2    STA IER            ;VIA: inhibit all interrupts
E5C5- A2 0A      LDX #10              ;get counter
E5C7- A9 00      LDA #0              *
E5C9- 9D EE 03    STA $03EE,X        ;clear TABstop byte
E5CC- CA          DEX                ;adapt counter
E5CD- 10 FA      BPL e5c9            ;if not all done, loop
E5CF- 60          RTS                ;else exit
    ;*****                                         *
    ;*                                         *
    ;* The area $E500-$E5FF is filled   *
    ;* with $AA bytes.                  *
    ;*                                         *
    ;*****                                         *
.BA $E600
    ;*****                                         *
    ;*                                         *
    ;* Return from interrupt.           *
    ;*                                         *
    ;*****                                         *
E600- 68          PREND   PLA
E601- A8          TAY                ;restore Y
E602- 68          PLA
E603- AA          TAX                ;and X
E604- 68          PLA                ;and A
E605- 40          RTI                ;and return from interrupt
    ;*****                                         *
    ;*                                         *
    ;* Store the character in A         *
    ;* on screen.                      *
    ;*                                         *
    ;*****                                         *
E606- A4 C6      DSPP    LDY *PNTR           ;get column position on line
E608- 91 C4      STA (PNT),Y       ;store the character there
E60A- 60          RTS                ;and exit
.FI "BASIC4.OE9B.M06"

```

11CB 33C9-4594 BASIC4.OE9B.M06

```

;NAME BASIC4.OE9B.M06
;*****
;* Keyboard table, for 80 keys.
;*
;* Keys with MSB set have no
;* shifted equivalent, e.g.
;* all the numerals.
;*
;*****
E60B- FF FF 3A CHAR .BY $FF $FF ':' $03 '963' $0F ;scan line nine
E60E- 03 39 36
E611- 33 DF
E613- B1 2F FF .BY $B1 '// $FF $13 'M X' $12 ;scan line eight
E616- 13 40 20
E619- 58 12
E61B- B2 10 FF .BY $B2 $10 $FF $B0 ',NVZ' ;scan line seven
E61E- B0 2C 4E
E621- 56 5A
E623- B3 00 FF .BY $B3 $00 $FF $AE '.BC' $00 ;scan line six
E626- AE 2E 42
E629- 43 00
E62B- B4 DB 4F .BY $B4 $DB '0' $11 'UTEQ' ;scan line five
E62E- 11 55 54
E631- 45 51
E633- 14 50 49 .BY $14 'PI' $DC 'YRW' $09 ;scan line four
E636- DC 59 52
E639- 57 09
E63B- B6 C0 4C .BY $B6 $C0 'L' $00 'JGDA' ;scan line three
E63E- 0D 4A 47
E641- 44 41
E643- B5 3B 4B .BY $B5 ';K' $DD 'HFS' $9B ;scan line two
E646- DD 48 46
E649- 53 9B
E64B- B9 FF 0E .BY $B9 $FF $DE $B7 $B0 '741' ;scan line one
E64E- B7 B0 37
E651- 34 31
E653- FF FF 1D .BY $FF $FF $1D $B8 '-852' ;scan line zero
E656- B8 2D 38
E659- 35 32
;*****
;* Table of lo-bytes of screen
;* line addresses.
;*
;*****
E65B- 00 28 50 LDTB2 .BY 0 40 80 120 160 200 240
E65E- 78 A0 C8
E661- F0
E662- 18 40 68 .BY 24 64 104 144 184 224
E665- 90 B8 E0
E668- 08 30 58 .BY 8 48 88 128 168 208 248
E66B- 80 A8 D0
E66E- F8
E66F- 20 48 70 .BY 32 72 112 152 192
E672- 98 C0
;*****
;* Message: D shift-L "* RETURN
;* RUN RETURN
;*
;*****

```

```

E674- 44 CC 22 RUNTB .BY 'D' $CC '**' $00 'RUN' $00
E677- 2A 0D 52
E67A- 55 4E 00
                                ;*****
                                ;*
                                ;* Checksum byte (over E-ROM). *
                                ;*
                                ;*****
E67D- 98 CKSUME .BY $98          ;checksum byte
                                ;*****
                                ;*
                                ;* The rest of the ROM is filled *
                                ;* with $AA bytes.             *
                                ;*
                                ;*****
                                ;
                                ;*****
                                ;*
                                ;* External Labels.           *
                                ;*
                                ;*****
CTIMR  .DE $008D      ;the jiffy clock
CINV   .DE $0090      ;IRQ ram vector
CBINV  .DE $0092      ;BRK RAM vector
LSTX   .DE $0097      ;last scan index
SHFLAG .DE $0098      ;shiftkey indicator
NDX    .DE $009E      ;number of keys in keyboard buffer
RVS    .DE $009F      ;flag for reverse field printing
INDX   .DE $00A1      ;record length of current screen line
LXSP   .DE $00A3      ;cursor row/column, used in INPUT and G
SFDX   .DE $00A6      ;current scan index
BLNSW  .DE $00A7      ;flag: cursor on
BLNCT  .DE $00A8      ;countdown till next cursor blink
GDBLN  .DE $00A9      ;real character under cursor
BLNON  .DE $00AA      ;cursor's blink phase
CRSW   .DE $00AC      ;flag: input from screen or keyboard
DFLTN  .DE $00AF      ;current input device number
DFLTO  .DE $00B0      ;current output device number
PNT    .DE $00C4      ;screen line address, lo
POINT  .DE $00C5      ;screen line address, hi
PNTR   .DE $00C6      ;current column position of cursor
SAL    .DE $00C7      ;pointer in screen scroll
SAH    .DE $00C8      ;pointer in screen scroll, hi
QTSW   .DE $00CD      ;flag: nonzero if in quote mode
LNMX   .DE $00D5      ;length of current screen line
TBLX   .DE $00D8      ;current line number of cursor
DATA   .DE $00D9      ;last key input
INSRT  .DE $00DC      ;number of inserts outstanding
LDTB1  .DE $00E0      ;screen line wrap around table
CAS1   .DE $00F9      ;flag: cassette #1 motor on
CAS2   .DE $00FA      ;flag: cassette #2 motor on
PI     .DE $00FF      ;value for PI
KEYD   .DE $026F      ;keyboard buffer
UDTIM  .DE $F768      ;update jiffy clock
PIAL   .DE $E810      ;keyboard PIA: I/O port A and DDR
PIAL1  .DE $E811      ;keyboard PIA: control register A
PIAK   .DE $E812      ;keyboard PIA: I/O port B and DDR
PIAS   .DE $E813      ;keyboard PIA: control register B
IEEEI  .DE $E820      ;IEEE PIA: I/O port A and DDR
IEEEIS .DE $E821      ;IEEE PIA: control register A
IEEO   .DE $E822      ;IEEE PIA: I/O port B and DDR
IEEEOS .DE $E823      ;IEEE PIA: control register B
PIA    .DE $E840      ;VIA: I/O port B
SYNC   .DE $E841      ;VIA: I/O port A with handshake

```

P2DB	.DE \$E842	;VIA: port B data direction register
P2DA	.DE \$E843	;VIA: port A data direction register
T1L	.DE \$E844	;VIA: timer 1 lo
T1H	.DE \$E845	;VIA: timer 1 hi
T1LL	.DE \$E846	;VIA: timer 1 lo, latch
T1LH	.DE \$E847	;VIA: timer 1 hi, latch
T2L	.DE \$E848	;VIA: timer 2 lo
T2H	.DE \$E849	;VIA: timer 2 hi
SR	.DE \$E84A	;VIA: shift register
ACR	.DE \$E84B	;VIA: auxiliary control register
PCR	.DE \$E84C	;VIA: peripheral control register
IFR	.DE \$E84D	;VIA: interrupt flag register
IER	.DE \$E84E	;VIA: interrupt enable register
SYNC1	.DE \$E84F	;VIA: port A without handshake
	.EN	

END MAE PASS

LABELFILE

ACR =E84B	BK1 =E23F	BK2 =E24A
BKLN =E1DE	BKLN1 =E1EA	BLKL =E434
BLKLN =E421	BLNCT =00A8	BLNON =00AA
BLNSW =00A7	CAS1 =00F9	CAS2 =00FA
CBINV =0092	CHAR =E60B	CINT =E000
CINV =0090	CKIS1 =E4DD	CKIT =E4EC
CKIT1 =E4F7	CKSUME =E67D	CKUT =E4EB
CLP1 =E158	CLP2 =E141	CLP21 =E156
CLP2A =E153	CLPS =EOF2	CLP6 =EOF8
CLP7 =E166	CLSR =EO4B	CNC3 =E254
CNC3X =E230	CRSW =00AC	CTIMR =0080
DATA =00D9	DFLTN =00AF	DFLTO =0080
DSPP =E606	GDBLN =00A9	IEEI =E820
IEEIS =E821	IEE0 =E822	IEEOS =E823
IER =E84E	IFR =E84D	INDX =00A1
INS1 =E2D5	INS2 =E2D7	INS3 =E2CE
INSRT =00DC	JLP2 =E340	JPL3 =E28C
JPL4 =E275	JSTS =E185	JSTS1 =E19F
JSTS2 =E1CD	JSTSX =E1B3	JSXB =E1BD
JTS2 =E1BE	KEY =E455	KEY3 =E49E
KEY4 =E47D	KEY5 =E479	KEYD =026F
KEYF =E544	KL1 =E4CD	KL2 =E4A7
KL22 =E4C3	KL23 =E4B2	KL24 =E4A4
KL25 =E4B8	KN1 =E538	LDTB1 =00E0
LDTB2 =E65B	LNMX =00D5	LOOP2 =E1A6
LOOP3 =E0BF	LOOP4 =E0BC	LOOPS =E116
LOP2 =E1AE	LOPS =E11E	LOPS1 =E124
LOPS2 =E134	LOPS3 =E138	LOPS4 =E12E
LP1 =EOAC	LP2 =EOA7	LP21 =E0D3
LP22 =EOEA	LP23 =EODF	LPS1 =E04F
LPS2 =EO5D	LPS3 =EO5E	LPS4 =E06A
LSTX =0097	LXSP =00A3	MLP1 =E382
MLP2 =E39B	MLP4 =E3D2	MLP41 =E3D5
MLP42 =E3DF	NC1 =E250	NC2 =E264
NC3 =E17A	NC3W =E257	NCX2 =E27A
NCZ2 =E277	NDX =009E	NELL =E414
NEWLA =E401	NEWLN =E3E2	NEWLX =E3F3
NEXL1 =E3F5	NJTL =E21D	NTCN =E229
NTCN1 =E250	NTCN2 =E1F3	NVS =E17C
NVSL =E182	NXLN =E343	NXLN1 =E350
NXLN2 =E348	NXT2 =E322	NXT3 =E176
NXT33 =E174	NXT6 =E32A	NXTD =E079

NXTL =E359	NXTX =E2A4	NXTX1 =E2AC
P2DA =E843	P2DB =E842	PCR =E84C
PI =D0FF	PIA =E840	PIAK =E812
PIAL =E810	PIAL1 =E811	PIAS =E813
PNT =D0C4	PNTR =D0C6	POINT =D0C5
PREND =E600	PRENDO =E546	PRT =E202
PRT1 =E28F	PRT2 =E297	PULS =E442
PULS1 =E452	PX1 =E009	QTSW =D0CD
QTSWC =E167	QTSWL =E173	RUNTB =E674
RVS =D09F	SAH =D0C8	SAL =D0C7
SCRL =E1C4	SCRL1 =E37B	SCRL3 =E385
SCRL4 =E3A7	SCRL5 =E3AB	SCROL =E369
SFDX =D0A6	SHFLAG =D098	SPCK =E4E9
SR =E84A	STUPR =E0A6	STUPT =E07F
STUPZ =E09C	SYNC =E841	SYNC1 =E84F
T1H =E845	T1L =E844	T1LH =E847
T1LL =E846	T2H =E849	T2L =E848
TBLX =D0D8	UDTIM =F768	UP1 =E303
UP2 =E2F3	UP3 =E312	UP5 =E28A
UP6 =E2EE	UP9 =E2EA	e216 =E216
e27e =E27E	e2b3 =E2B3	e336 =E336
e339 =E339	e33c =E33C	e33d =E33D
e469 =E469	e46b =E46B	e4be =E4BE
e513 =E513	e536 =E536	e549 =E549
e550 =E550	e557 =E557	e56a =E56A
e579 =E579	e584 =E584	e587 =E587
e58e =E58E	e59e =E59E	e5a1 =E5A1
e5b3 =E5B3	e5c1 =E5C1	e5c2 =E5C2
e5c9 =E5C9		
//0000,E67E,E67E		
]		

```

;NAME BASIC4.OEF4.CTL
;*****
;*      E-ROM for BASIC 4.0      *
;*                                *
;*      for 12 inch screen      *
;*          40 column models    *
;*              with             *
;*                  N-keyboard   *
;*                                *
;*          ('FAT Forty')       *
;*                                *
;* Date 14-09-83  Time 12:00  *
;*                                *
;* Made by:                   *
;*                                *
;* Nico de Vries              *
;* Mari Andriessenrade 49    *
;* 2907 MA Capelle a/d Yssel *
;* The Netherlands            *
;*                                *
;* Original CBM ROM number   *
;*                                *
;*          1498-01             *
;*                                *
;*****  

.BA $E000  

.FI "BASIC4.OEF4.M01"

```

·1AC6 340B-4ED1 BASIC4.OEF4.M01

			;	NAME BASIC4.OEF4.M01
			;	*****
			;	*
			;	Jump menu, enables standardi-
			;	sation between various machine
			;	types (compatible with 8032).
			;	*
			;	*****
E000- 4C 36 E0 CINT	JMP	CINT1	;	Initialize I/O, cursor home, 4 bells
E003- 4C A7 E0	JMP	LP2	;	Get character from keyboard buffer
E006- 4C 16 E1	JMP	LOOPS	;	Get character from device 0 or 3
E009- 4C 02 E2	JMP	PRT	;	Put character to screen
E00C- 4C 42 E4	JMP	PULS	;	Service interrupt
E00F- 4C 55 E4	JMP	KEY	;	Default interrupt routine
E012- 4C 00 E6 PREEND	JMP	PREND	;	Exit from interrupt
E015- 4C 42 E0	JMP	CLSR	;	Clear screen
E018- 4C 0F E6	JMP	e60f	;	Set to lower/upper case
E01B- 4C 17 E6	JMP	e617	;	Set to upper case/graphics
E01E- 4C 1D E6	JMP	e61d	;	Other CRT controller settings
E021- 4C EA E6	JMP	NEWLN	;	Scroll screen down
E024- 4C D1 E3	JMP	SCROL	;	Scroll screen up
E027- 4C BF E4	JMP	e4bf	;	Find key from decoding table
E02A- 4C 57 E6	JMP	e657	;	Ring bell once
E02D- 4C 98 E0	JMP	STUPR	;	(Store accu in repeat flag)
E030- 4C 98 E0	JMP	STUPR	;	(Set top left of window)
E033- 4C 98 E0	JMP	STUPR	;	(Set right bottom of window)
	;	*****	;	*

```

;* Reset routine.
;*
;*****
E036- 20 83 E6 CINT1 JSR e683      ;Initialize I/O and zero page
E039- 20 17 E6 JSR e617      ;set CRT controller to upper case/graph
E03C- 20 54 E6 JSR e654      ;ring bell twice
E03F- 20 54 E6 JSR e654      ;ring bell twice
;*****
;*
;* Initialize screen (clear it). *
;*
;*****
E042- A2 18    CLSR   LDX #24       ;do 25 lines
E044- A9 C0    CLSR   LDA #$C0      ;get lo-byte of last screen line
E046- A0 83    CLSR   LDY #$83      ;get hi-byte also
E048- 94 E0    LPS1   STY *LDTB1,X ;store hi-byte in table
E04A- 38        SEC
E04B- E9 28        SBC #40      ;calculate new lo-byte
E04D- B0 01        BCS LPS3      ;if page crossed
E04F- 88        DEY
E050- CA        LPS3   DEX
E051- 10 F5        BPL LPS1      ;if all lines done
E053- 84 C5        STY *PNT+1    ;set pointer in ($C4)
E055- E8        INX
E056- 86 9F        STX *RVS      ;set rvs off
E058- 86 C4        STX *PNT      ;and store lo-byte of pointer
E05A- A9 20        LDA #$20      ;put a blank
E05C- 9D 00 80    LPS4   STA $8000,X ;into all bytes
E05F- 9D 00 81    LPS4   STA $8100,X ;of screen
E062- 9D 00 82    LPS4   STA $8200,X
E065- 9D 00 83    LPS4   STA $8300,X
E068- E8        INX
E069- D0 F1        BNE LPS4      ;until whole screen done
E06B- A0 00    NXTD   LDY #$00
E06D- 84 C6        STY *PNTR     ;set cursor on start of line
E06F- 84 D8        STY *TBLX     ;and on first screen line
;*****
;*
;* Get pointer to current line in *
;* screen RAM.                    *
;*
;* Expects $D8 to contain current *
;* line number; ($C4) will be set *
;* to point to current line; $D5   *
;* will be set to max. length of *
;* that line.                      *
;*
;*****
E071- A6 D8    STUPT  LDX *TBLX     ;get screen line of cursor
E073- B5 E0    STUPT  LDA *LDTB1,X ;get hi-byte of that line
E075- 09 80    STUPT  ORA #$80      ;set bit 7 (flag in table)
E077- 85 C5    STUPT  STA *PNT+1    ;and store in pointer
E079- BD 98 E7    STUPT  LDA LDTB2,X ;get lo-byte from table
E07C- 85 C4    STUPT  STA *PNT      ;store in pointer too
E07E- A9 27    STUPT  LDA #39      ;set length to 39
E080- 85 D5    STUPT  STA *LNMX     ;if not bottom line
E082- E0 18    STUPT  CPX #24
E084- F0 08    STUPT  BEQ STUPZ
E086- B5 E1    STUPT  LDA *LDTB1+1,X ;and bit 7 from table clear
E088- 30 04    STUPT  BMI STUPZ
E08A- A9 4F    STUPT  LDA #79      ;then length is 79
E08C- 85 D5    STUPT  STA *LNMX
E08E- A5 C6    STUPT  LDA *PNTR     ;get position on line
E090- C9 28    STUPT  CMP #40

```

```

E092- 90 04      BCC STUPR
E094- E9 28      SBC #40
E096- 85 C6      STA *PNTR      ;and store modulo 40
E098- 60          STUPR       RTS
E099- AA          TAX
E09A- AA          TAX
E09B- AA          TAX
E09C- AA          TAX
E09D- AA          TAX
E09E- AA          TAX
E09F- AA          TAX
E0A0- AA          TAX
E0A1- AA          TAX
E0A2- AA          TAX
E0A3- AA          TAX
E0A4- AA          TAX
E0A5- AA          TAX
E0A6- AA          TAX
;*****
;*          *
;* Get character from keyboard  *
;* buffer.                      *
;*          *
;*****
EOA7- AC 6F 02    LP2        LDY KEYD      ;get oldest character
EOAA- A2 00
EOAC- BD 70 02    LP1        LDX #$00
EOAF- 9D 6F 02    STA KEYD+1,X ;move rest of buffer
                           STA KEYD,X ;foreward
EOB2- E8          INX
EOB3- E4 9E          CPX *NDX      ;until all char's done
EOB5- D0 F5          BNE LP1
EOB7- C6 9E          DEC *NDX      ;adapt index in buffer
EOB9- 98          TYA          ;get character in accu
EOBA- 58          CLI          ;enable interrupts again
EOBB- 60          RTS
;*****
;*          *
;* Wait for RETURN from keyboard. *
;*          *
;*****
EOBC- 20 3C E6    LOOP4     JSR e63c      ;echo character on screen
EOBF- A5 9E    LOOP3     LDA *NDX      ;get index in keyb. buffer
EOC1- 85 A7      STA *BLNSW    ;if empty, set cursor visible
EOC3- F0 FA      BEQ LOOP3    ;else wait for character
EOC5- 78          SEI          ;disable interrupts
EOC6- A5 AA      LDA *BLNON    ;if blink fase one
EOC8- F0 09      BEQ LP21
EOCA- A5 A9      LDA *GDBLN   ;get true char. at cursor position
EDCC- A0 00      LDY #$00
EOCE- 84 AA      STY *BLNON    ;set blink fase zero
E0D0- 20 06 E6    JSR DSPP    ;and put char to screen
E0D3- 20 A7 E0    LP21       JSR LP2
E0D6- C9 83      CMP #$83    ;get character from buffer
E0D8- D0 10      BNE LP22    ;if shift-STOP
E0DA- 78          SEI          ;disable interrupts
E0DB- A2 09      LDX #9
E0DD- 86 9E      STX *NDX      ;set index in buffer to 9
E0DF- BD 8E E7    LP23       LDA RUNTB-1,X ;move standard string
EOE2- 9D 6E 02    STA KEYD-1,X ;to buffer
EOE5- CA          DEX
EOE6- D0 F7      BNE LP23
EOE8- F0 D5      BEQ LOOP3    ;and get 1st char. of that string
EOEA- C9 DD      CMP #$0D    ;if is not RETURN
EOEC- D0 CE      LP22       BNE LOOP4    ;go echo to screen and get next char

```

E0EE- A4 05		LDY *LNMX	;else get max. length
E0F0- 84 AC		STY *CRSW	;flag input from screen
E0F2- B1 C4	CLP5	LDA (PNT),Y	;get char. from screen
E0F4- C9 20		CMP #\$20	;if space at end of line
E0F6- D0 03		BNE CLP6	
E0F8- 88		DEY	;get previous character
E0F9- D0 F7		BNE CLP5	
E0FB- C8	CLP6	INY	
E0FC- 84 A1		STY *INDX	;set record length of current line
E0FE- A0 00		LDY #\$00	
E100- 84 C6		STY *PNTR	;and set cursor at start of line
E102- 84 CD		STY *QTSW	;clear quote flag
E104- A5 A3		LDA *LXSP	;get line number on screen
E106- 3D 16		BMI LOP5	;if valid
E108- C5 D8		CMP *TBLX	;and equal to original
E10A- D0 12		BNE LOP5	
E10C- A5 A4		LDA *LXSP+1	;set position
E10E- 85 C6		STA *PNTR	;to original value
E110- C5 A1		CMP *INDX	;if not behind end of record
E112- 90 0A		BCC LOP5	;get char from device 3
E114- B0 2B		BCS CLP2	;else return RETURN

;*			
;* Get character from device 0			
;* or device 3.			
;*			

E116- 98	LOOP5	TYA	
E117- 48		PHA	;save Y
E118- 8A		TXA	
E119- 48		PHA	;save X
E11A- A5 AC		LDA *CRSW	;if input from keyboard
E11C- F0 A1		BEQ LOOP3	;go do keyboard until RETURN

;*			
;* Get character from current			
;* screen line.			
;*			

E11E- A4 C6	LOP5	LDY *PNTR	;get current position
E120- B1 C4		LDA (PNT),Y	;get screen character
E122- 85 D9		STA *DATA	;save it
E124- 29 3F	LOP51	AND #\$3F	;remove bits 6 and 7
E126- 06 D9		ASL *DATA	
E128- 24 D9		BIT *DATA	;if original bit 6 set
E12A- 10 02		BPL LOP54	
E12C- 09 80		ORA #\$80	;set bit 7 (shift bit)
E12E- 90 04	LOP54	BCC LOP52	;if original bit 7 set
E130- A6 CD		LDX *QTSW	;and if in quote mode
E132- D0 04		BNE LOP53	;don't change bit 6
E134- 70 02	LOP52	BVS LOP53	;else if bit 5 set in original
E136- 09 40		ORA #\$40	;set bit 6
E138- E6 C6	LOP53	INC *PNTR	;adapt position
E13A- 20 67 E1		JSR QTSWC	;adapt quote flag
E13D- C4 A1		CPY *INDX	;if max position reached
E13F- D0 17		BNE CLP1	
E141- A9 00	CLP2	LDA #\$00	
E143- 85 AC		STA *CRSW	;flag input from keyboard
E145- A9 00		LDA #\$00	;and echo RETURN
E147- A6 AF		LDX *DFLTN	;if input device
E149- E0 03		CPX #3	;is screen
E14B- F0 06		BEQ CLP2A	;go echo on screen
E14D- A6 B0		LDX *DFLTO	;else if output device
E14F- E0 03		CPX #3	;is screen

```

E151- F0 03      BEQ CLP21      ;store RETURN in last input
E153- 20 02 E2    CLP2A       JSR PRT      ;echo on screen
E156- A9 00      CLP21       LDA #$0D      ;and store a RETURN
E158- 85 D9      CLP1        STA *DATA     ;in last key input
E15A- 68          PLA         PLA
E15B- AA          TAX         ;restore X
E15C- 68          PLA         PLA
E15D- A8          TAY         ;restore Y
E15E- A5 D9      CLP1        LDA *DATA     ;get character again
E160- C9 DE      CLP7        CMP #$DE      ;if code for PI
E162- D0 02      CLP7        BNE CLP7
E164- A9 FF      CLP7        LDA #PI      ;get real code for PI
E166- 60          CLP7        RTS
.FI "BASIC4.OEF4.M02"

```

191C 340B-4027 BASIC4.OEF4.M02

```

;NAME BASIC4.OEF4.M02
;*****
;*
;* Toggle quote flag if " found. *
;*
;*****
E167- C9 22      QTSWC      CMP #'"'      ;if character is "
E169- D0 08      QTSWL      BNE QTSWL
E16B- A5 CD      QTSWL      LDA *QTSW      ;get quote flag
E16D- 49 01      QTSWL      EOR #$01      ;invert bit 0
E16F- 85 CD      QTSWL      STA *QTSW
E171- A9 22      QTSWL      LDA #'"'      ;and restore character
E173- 60          QTSWL      RTS
;*****
;*
;* Fill screen at current   *
;* position.                 *
;*
;*****
E174- 09 40      NXT33      ORA #$40      ;map shifted characters
E176- A6 9F      NXT3      LDX *RVS      ;if reverse switch on
E178- F0 02      NVT      BEQ NVS
E17A- 09 80      NC3       ORA #$80      ;set screencode for reverse
E17C- A6 DC      NVT      LDX *INSRT    ;if in insert mode
E17E- F0 02      NVT      BEQ NVSL
E180- C6 DC      NVT      DEC *INSRT    ;decrease insert count
E182- 20 06 E6    NVSL      JSR DSPP    ;put char at current position in RAM
E185- E6 C6      JSTS      INC *PNTR    ;advance character position
E187- A4 D5      JSTS      LDY *LNMX    ;if beyond maximum length
E189- C4 C6      JSTS      CPY *PNTR
E18B- B0 19      JSTS      BCS LOOP2
E18D- A6 D8      JSTS      LDX *TBLX    ;get line number in X
E18F- C0 4F      JSTS      CPY #79      ;and at position 79
E191- D0 DC      JSTS      BNE JSTS1
E193- 20 B3 E1      JSTS      JSR JSTSX    ;set current line to extended
E196- 20 A9 E3      JSTS      JSR NXLN    ;go to start of next line
E199- A9 00      JSTS      LDA #0      ;and set cursor at start
E19B- 85 C6      JSTS      STA *PNTR    ;of that line
E19D- F0 07      JSTS      BEQ LOOP2
E19F- E0 18      JSTS1     CPX #24      ;if beyond length, but <79
E1A1- D0 18      JSTS1     BNE JTS2    ;and on last line
E1A3- 20 C4 E1      JSTS1     JSR SCRL    ;scroll screen up and leave cursor on s
;*****
;*
;* Exit from output to screen. *
;*

```

```

;*****
E1A6- 68      LOOP2 PLA
E1A7- A8          TAY ;restore Y
E1A8- A5 DC      LDA *INSRT ;if in insert mode
E1AA- F0 02          BEQ LOP2
E1AC- 46 CD      LSR *QTSW ;set quote mode off
E1AE- 68      LOP2 PLA
E1AF- AA          TAX ;restore X
E1B0- 68      PLA ;and A
E1B1- 58      CLI ;allow interrupts again
E1B2- 60      RTS
;*****
;*
;* Set next line to not-extended.
;*
;*****
E1B3- E0 17      JSTSX CPX #23 ;if line number <=23
E1B5- B0 06      BCS JSXB
E1B7- B5 E2          LDA *LDTB1+2,X ;set next line
E1B9- 09 80      ORA #$80 ;to 'not-extended'
E1BB- 95 E2          STA *LDTB1+2,X
E1BD- 60      RTS
;*****
;*
;* Insert blank line if beyond
;* current length, but not at end
;* of line.
;*
;*****
E1BE- 20 CD E1      JTS2 JSR JSTS2 ;if beyond length but not at pos. 79
;insert new line
E1C1- 4C A6 E1      JMP LOOP2 ;and exit
;*****
;*
;* Get or insert new line.
;*
;*****
E1C4- 20 D1 E3      SCRL JSR SCROL ;scroll lines on screen
E1C7- C6 A3          DEC *LXSP ;decrement copy of line number
E1C9- C6 D8          DEC *TBLX ;and line# itself (cursor on same line)
E1CB- A6 D8          LDX *TBLX ;get line number
E1CD- 16 E1      JTS2 ASL *LDTB1+1,X ;set line to 'extended'
E1CF- 56 E1          LSR *LDTB1+1,X ;set line to 'not-extended'
E1D1- 20 B3 E1      JSR JSTSX ;and next line to 'not-extended'
E1D4- A5 C6          LDA *PNTR ;save position
E1D6- 48          PHA
E1D7- 20 71 ED      JSR STUPT ;and set pointer
E1DA- 68          PLA
E1DB- 85 C6          STA *PNTR ;and restore position
E1DD- 60          RTS
;*****
;*
;* Backward over line boundary.
;*
;*****
E1DE- A0 27      BKLN LDY #39
E1E0- A6 D8          LDX *TBLX ;if current line
E1E2- D0 06          BNE BKLN1 ;is top line
E1E4- 86 C6          STX *PNTR ;set cursor home
E1E6- 68          PLA
E1E7- 68          PLA ;remove return address
E1E8- D0 BC          BNE LOOP2 ;and exit
E1EA- B5 DF      BKLN1 LDA *LDTB1-1,X ;if not on top line
E1EC- 30 05          BMI NTCN2 ;and line is 'extended'

```

E1EE-	CA		DEX	
E1EF-	B5 DF		LDA *LDTB1-1,X	;get indicator of next line
E1F1-	A0 4F		LDY #79	;and set position to 79
E1F3-	CA	NTCN2	DEX	;adapt line number
E1F4-	86 D8		STX *TBLX	;and store it
E1F6-	85 C5		STA *PNT+1	;and store new pointer
E1F8-	B0 98 E7		LDA LDTB2,X	;store lo-byte
E1FB-	85 C4		STA *PNT	;as well
E1FD-	84 C6		STY *PNTR	;set cursor at end of line
E1FF-	84 D5		STY *LNMX	;and set length to 79
E201-	60		RTS	

			/*	*
			/* Put character to screen.	*
			/*	*

E202-	48	PRT	PHA	;save A
E203-	85 D9		STA *DATA	;also in temporary field
E205-	8A		TXA	
E206-	48		PHA	;save X
E207-	98		TYA	
E208-	48		PHA	;and Y
E209-	A9 00		LDA #0	
E20B-	85 AC		STA *CRSW	;set copy of position to zero
E20D-	A4 C6		LDY *PNTR	;get position
E20F-	A5 D9		LDA *DATA	;get character again
E211-	29 7F		AND #\$7F	;remove shift bit
E213-	C9 1B		CMP #\$1B	;if ESC or shift-ESC
E215-	00 03		BNE e21a	;not on keyboard and not in keyboard ta
E217-	4C C6 E3		JMP NXTL	;cancel quote mode
E21A-	A5 D9	e21a	LDA *DATA	;get character again
E21C-	10 03		BPL e221	;if shifted
E21E-	4C D5 E2		JMP NXTX	;go do shifted character
E221-	C9 0D	e221	CMP #\$0D	;if RETURN
E223-	00 03		BNE NJTL	
E225-	4C BF E3		JMP e3bf	;go do RETURN
E228-	C9 20	NJTL	CMP #\$20	;if printable
E22A-	90 08		BCC NTCN	
E22C-	29 3F		AND #\$3F	;convert to screen code
E22E-	20 67 E1		JSR QTSWC	;check for quote mode
E231-	4C 76 E1		JMP NXT3	;and put to screen
E234-	A6 DC	NTCN	LDX *INSRT	;if inserts outstanding
E236-	F0 03		BEQ CNC3X	
E238-	4C 7A E1		JMP NC3	;print as control character
E23B-	C9 14	CNC3X	CMP #\$14	;if DELETE
E230-	00 1C		BNE NTCN1	
E23F-	88		DEY	;decrement character position
E240-	84 C6		STY *PNTR	;and store it
E242-	10 06		BPL BK1	;if gone to previous line
E244-	20 DE E1		JSR BKLN	;go do appropriate action
E247-	4C 55 E2		JMP BK2	;and insert blank at end
E24A-	C8	BK1	INY	;else
E24B-	B1 C4		LDA (PNT),Y	;compress
E24D-	88		DEY	;line
E24E-	91 C4		STA (PNT),Y	;upto
E250-	C8		INY	;end of
E251-	C4 C5		CPY *PNT+1	;line
E253-	00 F5		BNE BK1	
E255-	A9 20	BK2	LDA #\$20	;and insert blank
E257-	91 C4		STA (PNT),Y	;at end
E259-	00 29		BNE NCZ2	
E25B-	A6 CD	NTCN1	LDX *QTSW	;if in quote mode
E25D-	F0 03		BEQ NC3W	
E25F-	4C 7A E1	CNC3	JMP NC3	;print as control character

```

E262- C9 12      NC3W   CMP #$12          ;if RVS-ON
E264- D0 04      BNE NC1
E266- 85 9F      STA *RVS           ;set reverse flag
E268- F0 1A      BEQ NCZ2          ;why not branch directly?
E26A- C9 13      CMP #$13          ;if HOME
E26C- D0 03      BNE NC2
E26E- 20 68 E0      JSR NXTD          ;set cursor home
E271- C9 1D      NC2    CMP #$1D          ;if cursor-RIGHT
E273- D0 12      BNE NCX2          ;set cursor position
E275- C8          INY
E276- 84 C6      STY *PNT          ;advance cursor
E278- 88          DEY
E279- C4 C5      CPY *PNT+1        ;if at line end
E27B- 90 07      BCC NCZ2          ;store position
E27D- 20 A9 E3      JSR NXLN          ;go to next line
E280- A0 00      LDY #$00          ;set cursor at beginning
E282- 84 C6      JPL4    STY *PNT          ;store position
E284- 4C A6 E1      NCZ2    JMP LOOP2        ;and exit
E287- C9 11      NCX2    CMP #$11          ;if cursor-DOWN
E289- D0 11      BNE e29c          ;set cursor position
E28B- 18          CLC
E28C- 98          TYA
E28D- 69 28      ADC #40          ;add 40 to position
E28F- A8          TAY
E290- C5 D5      CMP *LNMX          ;if not beyond length
E292- 90 EE      BCC JPL4          ;go store position
E294- F0 EC      BEQ JPL4          ;if at end also
E296- 20 A9 E3      JSR NXLN          ;else goto next line
E299- 4C A6 E1      JPL3    JMP LOOP2        ;and exit
E29C- C9 16      e29c    CMP #$16          ;if ERASE TO END OF LINE
E29E- D0 0C      BNE e2ac          ;set cursor position
E2A0- A9 20      LDA #$20          ;get blank
E2A2- 88          DEY
E2A3- C8          e2a3    INY
E2A4- 91 C4      STA (PNT),Y        ;fill line with blanks
E2A6- C4 D5      CPY *LNMX          ;until the end
E2A8- 90 F9      BCC e2a3          ;set cursor position
E2AA- B0 ED      BCS JPL3          ;if at end, exit
E2AC- C9 0E      e2ac    CMP #$0E          ;if TEXT
E2AE- D0 05      BNE e2b5          ;set cursor position
E2B0- 20 0F E6      JSR e60f          ;set to text mode
E2B3- 30 E4      BMI JPL3          ;and exit
E2B5- C9 07      e2b5    CMP #$07          ;if BELL
E2B7- D0 05      BNE e2be          ;set cursor position
E2B9- 20 57 E6      JSR e657          ;ring bell once
E2BC- F0 DB      BEQ JPL3          ;and exit
E2BE- C9 09      e2be    CMP #$09          ;if not TAB
E2C0- D0 D7      BNE JPL3          ;ignore
E2C2- C4 D5      e2c2    CPY *LNMX          ;if TAB and at end of line
E2C4- 90 07      BCC e2cd          ;set cursor position
E2C6- A4 D5      LDY *LNMX          ;get length
E2C8- 84 C6      e2c8    STY *PNT          ;and set cursor there
E2CA- 4C A6 E1      JMP LOOP2        ;and exit
E2CD- C8          e2cd    INY          ;check next position
E2CE- 20 88 E5      JSR e588          ;get next tabstop
E2D1- F0 EF      BEQ e2c2          ;if no tabstop, retry for next column
E2D3- D0 F3      BNE e2c8          ;if tabstop found set cursor there
.FI "BASIC4.OEF4.M03"

```

```

;*
;* Put shifted characters to      *
;* screen.                      *
;*                                *
;*****                         *
E205- 29 7F      NXTX    AND #$7F          ;remove shift bit
E207- C9 7F      CMP #PI-$80        ;if code for PI
E209- D0 02      BNE NXTX1
E20B- A9 5E      LDA #$5E          ;get real code for PI
E20D- C9 20      NXTX1   CMP #$20          ;if printable
E20F- 90 03      BCC e2e4
E2E1- 4C 74 E1   JMP NXT33        ;go fill screen
E2E4- C9 00      e2e4    CMP #$0D          ;if shift-RETURN
E2E6- D0 03      BNE UPS
E2E8- 4C BF E3   JMP e3bf          ;go do RETURN
E2EB- A6 CD      UPS     LOX *QTSW
E2ED- D0 30      BNE UP6           ;print as control character
E2EF- C9 14      CMP #$14          ;no quote mode, if INSERT
E2F1- D0 28      BNE UP9
E2F3- A4 D5      LDY *LNMX
E2F5- B1 C4      LDA (PNT),Y    ;get last char. of line
E2F7- C9 20      CMP #$20          ;if blank
E2F9- D0 04      BNE INSS
E2FB- C4 C6      CPY *PNTR
E2FD- D0 07      BNE INS1
E2FF- C0 4F      INS3    CPY #79
E301- F0 81      BEQ NCZ2
E303- 20 EA E6   JSR NEWLN
E306- A4 D5      INS1    LOY *LNMX
E308- 88         INS2    DEY
E309- B1 C4      LDA (PNT),Y    ;characters
E30B- C8         INY
E30C- 91 C4      STA (PNT),Y    ;one position
E30E- 88         DEY
E30F- C4 C6      CPY *PNTR
E311- D0 F5      BNE INSS
E313- A9 20      LDA #$20          ;and put a blank
E315- 91 C4      STA (PNT),Y    ;in space opened up
E317- E6 DC      INC *INSRT
E319- D0 55      BNE JLP2          ;adapt insert counter
E31B- A6 DC      UP9    LOX *INSRT
E31D- F0 05      BEQ UP2
E31F- 09 40      UP6    ORA #$40
E321- 4C 7A E1   JMP NC3
E324- C9 11      UP2    CMP #$11
E326- D0 2B      BNE NXT2
E328- A5 C6      LOA *PNTR
E32A- C9 28      CMP #40
E32C- 90 06      BCC UP1
E32E- E9 28      SBC #40
E330- 85 C6      STA *PNTR
E332- B0 3C      BCS JLP2
E334- A6 D8      LOX *TBLX
E336- F0 38      BEQ JLP2
E338- B5 0F      LOA *LDTB1-1,X
E33A- 10 07      BPL UP3
E33C- C6 D8      DEC *TBLX
E33E- 20 71 EO   JSR STUPT
E341- 90 2D      BCC JLP2
E343- CA         UP3    DEX
E344- CA         DEX
E345- 86 D8      STX *TBLX
E347- 20 71 EO   JSR STUPT
E34A- A5 C6      LOA *PNTR
;
```

```

E34C- 18           CLC
E34D- 69 28        ADC #40      ;and set cursor on second
E34F- 85 C6        STA *PNTR   ;part of that line
E351- D0 10        BNE JLP2    ;and exit
E353- C9 12        NXT2      CMP #$12   ;if RVS-off
E355- D0 06        BNE NXT6
E357- A9 00        LDA #$00
E359- 85 9F        STA *RVS   ;reset reverse flag
E35B- F0 13        BEQ JLP2    ;and exit
E35D- C9 10        NXT6      CMP #$1D   ;if cursor-LEFT
E35F- D0 0B        BNE e36c
E361- 88           DEY
E362- 84 C6        STY *PNTR   ;decrement position
E364- 10 DA        BPL JLP2    ;exit, if still on same line
E366- 20 DE E1     JSR BKLN    ;else do backwards over boundary
E369- 4C A6 E1     JMP LOOP2   ;and-exit
E36C- C9 13        e36c      CMP #$13   ;if CLR-screen
E36E- D0 06        BNE e376
E370- 20 42 E0     JLP2      JSR CLSR   ;clear the screen
E373- 4C A6 E1     JMP LOOP2   ;and exit
E376- C9 16        e376      CMP #$16   ;if ERASE FROM START
E378- D0 00        BNE e387
E37A- A9 20        LDA #$20
E37C- A0 00        LDY #$00
E37E- C4 C6        e37e      CPY *PNTR   ;if current position reached
E380- B0 EE        BCS JLP2    ;exit
E382- 91 C4        STA (PNT),Y ;store blank
E384- C8           INY
E385- D0 F7        BNE e37e
E387- C9 0E        e387      CMP #$0E   ;if GRAPHICS
E389- D0 05        BNE e390
E38B- 20 17 E6     JSR e617    ;set to graphics
E38E- 30 E0        BMI JLP2    ;and exit
E390- C9 07        e390      CMP #$07   ;if shift-BELL
E392- D0 05        BNE e399
E394- 20 54 E6     JSR e654    ;ring bell twice
E397- F0 D7        BEQ JLP2    ;and exit
E399- C9 09        e399      CMP #$09   ;if not shift-TAB
E39B- D0 D3        BNE JLP2    ;ignore the character
E39D- 20 88 E5     JSR e588    ;TAB, get tabstop
E3A0- 40 EF 03     EOR $03EF  ;add or clear tabstopbit
E3A3- 9D F0 03     STA $03F0,X ;and store again
E3A6- 4C A6 E1     JMP LOOP2   ;and exit .
;*****
;*
;* Set next line number. *
;*
;*****
E3A9- 38           NXLN     SEC      ;set copy of line number invalid
E3AA- 46 A3        LSR *LXSP   ;BUG: wrong type of shift
E3AC- A6 D8        LDX *TBLX   ;get line number
E3AE- E8           NXLN2    INX      ;increment it
E3AF- E0 19        CPX #25    ;if at maximum line number
E3B1- D0 03        BNE NXLN1
E3B3- 20 D1 E3     JSR SCROL  ;scroll screen up
E3B6- B5 E0        NXLN1    LDA *LDTB1,X ;if line is 'extended'
E3B8- 10 F4        BPL NXLN2   ;scroll another time
E3BA- 86 D8        STX *TBLX   ;store line number
E3BC- 4C 71 E0     JMP STUPT  ;and get pointer of current line
;*****
;*
;* Action for RETURN. *
;*
;*****

```

E3BF- 20 A9 E3	e3bf	JSR NXLN	;set next line number
E3C2- A9 00	e3c2	LDA #0	
E3C4- 85 C6		STA *PNTR	;set cursor at beginning of line
E3C6- A9 00	NXTL	LDA #0	;(entry used by ESC)
E3C8- 85 DC		STA *INSRT	;cancel inserts outstanding
E3CA- 85 9F		STA *RVS	;reset reverse flag
E3CC- 85 CD		STA *QTSW	;cancel quote mode
E3CE- 4C A6 E1		JMP LOOP2	;and exit
		;*****	
		;*	
		;* Scroll screen up.	
		;*	
		;*****	
E3D1- A2 19	SCROL	LDX #25	;set up line number
E3D3- 86 D8		STX *TBLX	;store it
E3D5- A2 FF	e3d5	LDX #\$FF	
E3D7- E8	e3d7	INX	
E3D8- B0 98 E7		LDA LDTB2,X	;get lo-byte of first line
E3D9- 85 C4		STA *PNT	;store it
E3D0- B5 E0		LDA *LDTB1,X	;get hi-byte also
E3D9- 09 80		ORA #\$80	;set MSB
E3E1- 85 C5		STA *PNT+1	;and store as well
E3E3- E0 18		CPX #24	;if all lines done
E3E5- B0 1D		BCS e404	;go fill bottom line with spaces
E3E7- B4 E1		LDY *LDTB1+1,X	;if next line 'extended'
E3E9- 30 02		BMI SCRL3	
E3EB- 29 7F		AND #\$7F	;set current line to 'extended'
E3ED- 95 E0	SCRL3	STA *LDTB1,X	
E3EF- 98		TYA	;get hi-byte of next line
E3F0- 09 80		ORA #\$80	;set MSB
E3F2- 85 C8	SCRL1	STA *SAH	
E3F4- B0 99 E7		LDA LDTB2+1,X	;get lo-byte also
E3F7- 85 C7		STA *SAL	
E3F9- A0 27		LDY #39	;move 40 characters
E3FB- B1 C7	MLP1	LDA (SAL),Y	;get char of next line
E3FD- 91 C4		STA (PNT),Y	;move it to current line
E3FF- 88		DEY	;until
E400- 10 F9		BPL MLP1	;whole line done
E402- 30 D3		BMI e3d7	;then do next line
E404- 95 E0	e404	STA *LDTB1,X	;store last hi-byte
E406- A0 27		LDY #39	
E408- A9 20		LDA #\$20	;get blank
E40A- 91 C4	MLP2	STA (PNT),Y	;fill bottom line with spaces
E40C- 88		DEY	
E40D- 10 FB		BPL MLP2	
E40F- C6 D8		DEC *TBLX	;adapt line number
E411- A5 E0		LDA *LDTB1	;if first line
E413- 10 C0		BPL e3d5	;is 'extended', repeat
E415- AD 12 E8		LDA PIAK	;get key image (of last row)
E418- C9 FE		CMP #\$FE	;if RVS pressed
E41A- D0 0B		BNE MLP42	
E41C- A0 00		LDY #\$00	
E41E- EA	MLP41	NOP	;wait
E41F- CA		DEX	
E420- D0 FC		BNE MLP41	;a
E422- 88		DEY	
E423- D0 F9		BNE MLP41	;while
E425- 84 9E		STY *NDX	;cancel keyboard buffer
E427- A6 D8	MLP42	LDX *TBLX	;and get line number
E429- 60		RTS	
E42A- AA		TAX	
E42B- AA		TAX	
E42C- AA		TAX	
E42D- AA		TAX	

```

*****  

;*  

;* Clock correction.  

;* Add an extra clock tick in *  

;* every five, because screen is *  

;* refreshed with a 50 Hz rate *  

;* instead of the usual 60 Hz. *  

;*  

;*****  

E42E- 20 EA FF e42e JSR $FFEA      ;adapt clock  

E431- EE ED 03          INC $03ED      ;adapt correction counter  

E434- AD ED 03          LDA $03ED  

E437- C9 06              CMP #6        ;if not five ticks done  

E439- D0 1D              BNE e458      ;do rest of interrupt  

E43B- A9 00              LDA #0        ;if five ticks done  

E43D- 80 ED 03          STA $03ED      ;reset correction count  

E440- F0 EC              BEQ e42e      ;and do an extra tick  

*****  

;*  

;* Default interrupt routine,  

;* entered 50 times per second.  

;*  

;* (Pointed to by ROM).  

;*  

;*****  

E442- 48          PULS PHA           ;save A  

E443- 8A          TXA  

E444- 48          PHA           ;save X  

E445- 98          TYA  

E446- 48          PHA           ;and Y  

E447- BA          TSX  

E448- BD 04 01    LOA $0104,X   ;get saved status reg.  

E44B- 29 10          AND #$10      ;test for BRK-instruction  

E44D- F0 03          BEQ PULS1     ;if BRK  

E44F- 6C 92 00    JMP (CBINV)   ;go do BRK-routine  

E452- 6C 90 00    JMP (CINV)     ;else do IRQ-routine  

E455- 4C 2E E4    KEY JMP e42e      ;IRQ-entry (pointed to by ($90))  

.FI "BASIC4.OEF4.MO4"

```

142C 340B-4837 BASIC4.OEF4.MO4

```

;name BASIC4.OEF4.MO4
*****  

;*  

;* Rest of interrupt routine;  

;* do cursor blink.  

;*  

;*****  

E458- A5 A7          e458 LDA *BLNSW     ;if cursor invisible  

E45A- D0 1F          BNE KEY4      ;skip blink handling  

E45C- C6 A8          DEC *BLNCT     ;decrement countdown  

E45E- D0 1B          BNE KEY4      ;if countdown  

E460- A9 14          LDA #20       ;set new countdown  

E462- 2C EE 03    BIT RPTFLG    ;if repeat  

E465- 10 02          BPL e469      ;  

E467- A9 02          LDA #2        ;set shorter countdown  

E469- 85 A8          STA *BLNCT     ;  

E46B- A4 C6          LDY *PNTR      ;get column position  

E46D- 46 AA          LSR *BLNON     ;determine blink phase  

;and set phase zero  

E46F- B1 C4          LDA (PNT),Y   ;get character at current position  

E471- B0 04          BCS KEYS      ;if in phase zero  

E473- E6 AA          INC *BLNON     ;set blink phase one

```

E475- 85 A9		STA *GDBLN	;and store true character
E477- 49 80	KEY5	EOR #\$80	;reverse screen character
E479- 91 C4		STA (PNT),Y	;and put to screen
		*****	*****
		/*	*
		/* Cassette motor control.	*
		/*	*
		*****	*****
E47B- A0 00	KEY4	LDY #\$00	;prepare for keyboard scan
E47D- AD 10 E8		LDA PIAL	
E480- 29 F0		AND #\$FO	;set zero in
E482- 8D 10 E8		STA PIAL	;keyboard scan line number
E485- AD 10 E8		LDA PIAL	;get cassette sense lines
E488- 0A		ASL A	;in bit 7
E489- 0A		ASL A	;and carry
E48A- 0A		ASL A	
E48B- 10 09		BPL KEY3	;if no key down on cassette 1
E48D- 84 F9		STY *CAS1	;flag cassette 1 off
E48F- AD 13 E8		LDA PIAS	
E492- 09 08		ORA #\$08	;and switch motor off
E494- D0 09		BNE KL24	;and test cass#2
E496- A5 F9	KEY3	LDA *CAS1	;key down,if #1 flag on
E498- D0 08		BNE KL2	;test cass#2
E49A- AD 13 E8		LDA PIAS	;else
E49D- 29 F7		AND #\$F7	;start cass#1 motor
E49F- 8D 13 E8	KL24	STA PIAS	
E4A2- 90 09	KL2	BCC KL23	;if no key down on cass.#2
E4A4- 84 FA		STY *CAS2	;flag cass.#2 off
E4A6- AD 40 E8		LDA PIA	
E4A9- 09 10		ORA #\$10	;and switch cass#2 motor off
E4AB- D0 09		BNE KL25	
E4AD- A5 FA	KL23	LDA *CAS2	;key down,if #2 flag on
E4AF- D0 08		BNE e4bf	;finish cassette control
E4B1- AD 40 E8		LDA PIA	;else
E4B4- 29 EF		AND #\$EF	;switch #2 motor on
E4B6- 8D 40 E8	KL25	STA PIA	
E4B9- 20 BF E4	e4bf	JSR e4bf	;handle keyboard
E4BC- 4C 00 E6		JMP PREND	;and return from interrupt
		*****	*****
		/*	*
		/* Keyboard handling.	*
		/*	*
		*****	*****
E4BF- A0 FF	e4bf	LDY #\$FF	;this index received
E4C1- 84 A6		STY *SFDX	;on no key at all
E4C3- C8		INY	
E4C4- 84 98		STY *SHFLAG	;clear shift key indicator
E4C6- AD EE 03		LDA RPTFLG	;get repeat flag
E4C9- 29 7F		AND #\$7F	;zero MSB
E4CB- 8D EE 03		STA RPTFLG	
E4CE- A2 50		LDX #80	;this number of keys to be done
E4D0- A0 08	KL22	LDY #8	;number of rows
E4D2- AD 12 E8		LDA PIAK	;get key image
E4D5- CD 12 E8		CMP PIAK	;wait
E4D8- D0 F6		BNE KL22	;until steady
E4DA- C9 FF		CMP #\$FF	;if no key
E4DC- D0 09		BNE KL1	
E4DE- 8A		TXA	;move index
E4DF- 38		SEC	
E4E0- E9 08		SBC #\$08	;subtract eight
		;BUG: wrong result if IRQ entered	
		;with decimal flag set.	
E4E2- AA		TAX	;move it back
E4E3- D0.2C		BNE e511	;and try next scan line

E4E5- F0 2F		BEQ CKIT1	;if all keys tried, handle repeat
E4E7- 4A	KL1	LSR A	;key pressed, get bit
E4E8- B0 21		BCS CKIT	;if key found (bit=0)
E4EA- 48		PHA	;save A
E4EB- B0 3E E7		LDA CHAR-1,X	;get char. from table
E4EE- D0 06		BNE CKIS2	;if zero
E4FD- A9 01		LDA #1	;indicate
E4F2- 85 98		STA *SHFLAG	;shift-key found
E4F4- D0 14		BNE CKUT	;and check next bits
E4F6- C9 10	CKIS2	CMP #\$10	;if REPEAT-key pressed
E4F8- D0 DA		BNE CKIS1	;left-over from business version)
E4FA- AD EE 03		LDA RPTFLG	;get flag byte
E4FD- D9 80		ORA #\$80	;set repeat on
E4FF- 8D EE 03		STA RPTFLG	;store it
E502- 30 06		BMI CKUT	;and go repeat
E504- C9 FF	CKIS1	CMP #\$FF	;if legal key
E506- F0 02		BEQ CKUT	
E508- 85 A6		STA *SF0X	;store it
E50A- 68	CKUT	PLA	;restore key image
E50B- CA	CKIT	DEX	;try next key
E50C- F0 08		BEQ CKIT1	;if all done, handle repeat
E50E- 88		DEY	;adapt bit counter
E50F- D0 06		BNE KL1	;if not all done, loop
E511- EE 10 E8	e511	INC PIAL	;check next scan line
E514- D0 BA		BNE KL22	
E516- A5 A6	CKIT1	LDA *SF0X	;get key found
E518- C5 97		CMP *LSTX	;if same as last time
E51A- F0 07		BEQ e523	;go do repeat
E51C- A2 10		LDX #\$10	;else store countdown
E51E- 8E E9 03		STX DELAY	
E521- D0 33		BNE e556	;and exit
E523- 2C EE 03	e523	BIT RPTFLG	;get repeat flag
E526- 30 20		BMI e548	;if REPEAT pressed, go repeat
E528- 70 50		BVS e587	;if repeat off, exit
E52A- C9 FF		CMP #\$FF	;if no key
E52C- F0 59		BEQ e587	;exit
E52E- C9 14		CMP #\$14	;if INST/DEL
E530- F0 DC		BEQ e53e	;do repeat
E532- C9 20		CMP #\$20	;if SPACE
E534- F0 08		BEQ e53e	;do repeat
E536- C9 10		CMP #\$10	;if LEFT/RIGHT
E538- F0 04		BEQ e53e	;do repeat
E53A- C9 11		CMP #\$11	;if not UP/DOWN
E53C- D0 49		BNE e587	;exit
E53E- AE E9 03	e53e	LDX DELAY	;get delay
E541- F0 05		BEQ e548	;if expired, load countdown
E543- CE E9 03		DEC DELAY	;else delay
E546- D0 3F		BNE e587	;and exit
E548- CE EA 03	e548	DEC KOUNT	;decrement countdown
E54B- D0 3A		BNE e587	;if not expired, exit
E54D- A2 04		LDX #4	;else
E54F- 8E EA 03		STX KOUNT	;reload countdown
E552- A6 9E		LDX *NDX	;get # of keys in buffer
E554- D0 31		BNE e587	;if not empty, exit
E556- 85 97	e556	STA *LSTX	;else store new value
E558- C9 FF		CMP #\$FF	;if no key
E55A- F0 2B		BEQ e587	;exit
E55C- EA		NOP	
E55D- EA		NOP	
E55E- EA		NOP	
E55F- EA		NOP	
E560- EA		NOP	
E561- EA		NOP	
E562- EA		NOP	

```

E563- 46 98      LSR *SHFLAG      ;get shift status
E565- 90 13      BCC KN1         ;if pressed
E567- EA          NOP
E568- EA          NOP
E569- EA          NOP
E56A- EA          NOP
E56B- EA          NOP
E56C- EA          NOP
E56D- EA          NOP
E56E- EA          NOP
E56F- EA          NOP
E570- EA          NOP
E571- EA          NOP
E572- EA          NOP
E573- EA          NOP
E574- EA          NOP
E575- EA          NOP
E576- EA          NOP
E577- EA          NOP
E578- 09 80      ORA #$80        ;add shift bit
E57A- A6 9E      LDX *NDX        ;get # of keys in buffer
E57C- EC EB 03      CPX XMAX        ;if more than max. buffer length
E57F- B0 06      BCS e587        ;exit (ignore key)
;BUG: exits on equality, not if more,
;      so default 9, accepts 9 keys maximum,
;      not ten.
E581- 9D 6F 02      STA KEYD,X    ;else store in buffer
E584- E8          INX
E585- 86 9E      STX *NDX        ;and adapt buffer counter
E587- 60          RTS
e587
;*****
;*
;* Set zero flag if not at
;* TAB-stop.
;*
;*****
E588- 98          e588          TYA          ;get position in A
E589- 29 07      AND #$07        ;get lowest 3 bits
E58B- AA          TAX           ;move to index
E58C- BD DC E7      LDA e7dc,X    ;get bit mask from table
E58F- 8D EF 03      STA $03EF        ;store it
E592- 98          TYA          ;get position again
E593- 4A          LSR A         ;divide by eight
E594- 4A          LSR A         ;move to index
E595- 4A          LSR A         ;get byte with tabstops
E596- AA          TAX           ;set zero flag if not at tabstop
E597- BD F0 03      LDA $03F0,X
E59A- 2C EF 03      BIT $03EF
E59D- 60          RTS
.FI "BASIC4.OEF4.MOS"

```

1A49 340B-4E54 BASIC4.OEF4.MOS

```

;name BASIC4.OEF4.MOS
;*****
;*
;* The area $E59E-$E5FF is filled *
;* with $AA bytes.                 *
;*
;*****
.BA $E60'0
;*****
;*

```

```

        ;* Exit from interrupt.      *
        ;*
        ;*****
E600- 68    PREND PLA
E601- A8      TAY          ;restore Y
E602- 68      PLA
E603- AA      TAX          ;and X
E604- 68      PLA          ;and A
E605- 40      RTI          ;and return from interrupt
        ;*****
        ;*
        ;* Store the character in A      *
        ;* on screen.                  *
        ;*
        ;*****
E606- A4 C6    DSPP LDY *PNTR      ;get cursor position on line
E608- 91 C4      STA (PNT),Y   ;store the character
E60A- A9 02      LDA #2        ;and set up new countdown
E60C- 85 A8      STA *BLNCT    ;for cursor flash
E60E- 60      RTS
        ;*****
        ;*
        ;* Set CRT-controller to text      *
        ;* mode.                      *
        ;*
        ;*****
E60F- A9 B1    e60f LDA #L,e7b1
E611- A2 E7      LDX #H,e7b1   ;load pointer to constants
E613- A0 0E      LDY #$0E      ;load VIA reg. value for text
E615- D0 06      BNE e61d    ;and move constants
        ;*****
        ;*
        ;* Set CRT-controller to graphics *
        ;* mode.                      *
        ;*
        ;*****
E617- A9 C3    e617 LDA #L,e7c3
E619- A2 E7      LDX #H,e7c3   ;load pointer to constants
E61B- A0 DC      LDY #$0C      ;and VIA reg. value for graphics
E61D- 85 C7    e61d STA *SAL
E61F- 86 C8      STX *SAH     ;store pointer
E621- AD 4C E8      LDA PCR     ;get VIA register
E624- 29 F0      AND #$FO     ;clear lower nibble
E626- 85 D1      STA *FNLEN   ;save it
E628- 98      TYA          ;get nibble for charactergenerator
E629- 05 D1      ORA *FNLEN   ;add hi-nibble
E62B- 8D 4C E8      STA PCR     ;and set up charactergenerator
E62E- A0 11      LDY #17      ;18 constants to be moved
E630- B1 C7    e630 LDA (SAL),Y ;get constant
E632- 8C 80 E8      STY CRO     ;set register number
E635- 8D 81 E8      STA CR1     ;and set up that register
E638- 88      DEY          ;move character to screen
E639- 10 F5      BPL e630    ;and do all registers
E63B- 60      RTS
        ;*****
        ;*
        ;* Ring bell if right margin      *
        ;* passed.                     *
        ;*
        ;*****
E63C- 20 02 E2    e63c JSR PRT    ;move character to screen
E63F- AA      TAX          ;save it in X
E640- A5 D5      LDA *LNMX    ;get length of line
E642- 38      SEC

```

```

E643- E5 C6      SBC *PNTR      ;subtract current position
E645- C9 05      CMP #5        ;if 5 from margin
E647- D0 39      BNE e682
E649- 8A          TXA          ;get character again
E64A- C9 10      CMP #$10      ;if cursor-RIGHT
E64C- F0 06      BEQ e654      ;ring bell twice
E64E- 29 7F      AND #$7F      ;remove shift-bit
E650- C9 20      CMP #$20      ;if not printable
E652- 90 2E      BCC e682      ;exit, else ring bell twice
;*****  

;*  

;* Ring bell twice.  

;*  

;*  

;*****  

E654- 20 57 E6  e654      JSR e657      ;ring bell once
;*****  

;*  

;* Ring bell once.  

;*  

;*  

;*****  

E657- AC EC 03  e657      LDY $03EC    ;get bell timing
E65A- F0 26      BEQ e682      ;if zero, don't ring bell
E65C- A9 10      LDA #$10      ;set CB2 to free run
E65E- 8D 48 E8  STA ACR
E661- A9 0F      LDA #$0F      ;and to lowest frequency
E663- 8D 4A E8  STA SR
E666- A2 07      LDX #7       ;7 tones to play
E668- BD D4 E7  e668      LDA e7d5-1,X ;get tone constant
E66B- 8D 48 E8  STA T2L      ;put to VIA
E66E- AD EC 03  LDA $03EC    ;get timing
E671- 88          DEY          e671
E672- D0 F0      BNE e671      ;wait, inner loop
E674- 38          SEC
E675- E9 01      SBC #1
E677- D0 F8      BNE e671      ;wait, outer loop
E679- CA          DEX
E67A- D0 EC      BNE e668      ;and do next tone
E67C- 8E 4A E8  STX SR
E67F- 8E 4B E8  STX ACR      ;then silence the VIA
E682- 60          RTS          e682 .   ;and enable cassette operations
;*****  

;*  

;* Reset; initialize I/O.  

;*  

;*****  

E683- A9 7F      e683      LDA #$7F
E685- 8D 4E E8  STA IER      ;disable all VIA interrupts
E688- A2 60      LDX #$60
E68A- A9 00      LDA #$00      ;clear
E68C- 95 80      STA *CTIMR,X ;area $80 to $FA
E68E- CA          DEX
E68F- 10 FB      BPL PX1
E691- A2 0A      LDX #10
E693- 90 F0 03  e693      STA $03F0,X ;clear TABstop table
E696- CA          DEX
E697- 10 FA      BPL e693
E699- 8D EE 03  STA RPTFLG   ;enable repeat of cursor keys
E69C- A9 55      LDA #L,KEY  ;set
E69E- 85 90      STA *CINV   ;IRQ RAM vector
E6A0- A9 E4      LDA #H,KEY  ;to
E6A2- 85 91      STA *CINV+1 ;keyboard/cassette routine
E6A4- A9 03      LDA #3       ;set default output device
E6A6- 85 B0      STA *DFLTO  ;to screen
E6A8- A9 0F      LDA #$0F     ;keyboard PIA, A section:

```

E6AA- 8D 10 E8	STA PIAL	;bits 0-3 output, rest inp
E6AD- 0A	ASL A	
E6AE- 8D 40 E8	STA PIA	;cassette#2 motor off
	;cassette #1 and #2 write = 1	
	;IEEE ATN=false	NRF0=false
E6B1- 8D 42 E8	STA P2DB	;VIA A: bits 7-5,0 output, 4-1 input
E6B4- 8E 22 E8	STX IEE0	;IEEE PIA, B section: all bits output
E6B7- 8E 45 E8	STX T1H	;VIAT1H: maximum timeout
E6BA- A9 3D	LDA #\$3D	;keyboard PIA: cassette#1 motor off
E6BC- 8D 13 E8	STA PIAS	;IRQ on neg edge on CB1
	; (end of screen refresh)	
E6BF- 2C 12 E8	BIT PIAK	;clear keyboard PIA interrupt flags
E6C2- A9 3C	LDA #\$3C	;IEEE PIA: NDAC=false, CA1 neg. edge
E6C4- 8D 21 E8	STA IEEEIS	;no interrupts
E6C7- 8D 23 E8	STA IEEEOS	;IEEE PIA: DAV=false, CB1 neg. edge
	;no interrupts	
E6CA- 8D 11 E8	STA PIAL1	;keyboard PIA: CA2=1,CA1 neg. edge
	;no interrupts	
E6CD- 8E 22 E8	STX IEE0	;IEEE PIA: all data lines false
E6D0- A9 0C	LDA #\$0C	;VIA:CA2=0 (graphics), CA1neg. edge
E6D2- 8D 4C E8	STA PCR	;no interrupts,CB2 input,neg. edge
E6D5- 85 A8	STA *BLNCT	;set blink count for cursor
E6D7- 85 A7	STA *BLNSW	;set cursor visible
E6D9- A9 09	LDA #9	;default keyboard buffer length
	;BUG: must be 10 for bufferlength of 10	
E6DB- 8D EB 03	STA XMAX	
E6DE- A9 10	LDA #\$10	
E6E0- 8D EC 03	STA \$03EC	;set up bell timing,
E6E3- 8D E9 03	STA DELAY	;repeat delay
E6E6- 8D EA 03	STA KOUNT	;and repeat rate
E6E9- 60	RTS	;then exit

	/*	*
	/* Scroll rest of screen down	*
	/* and insert a blank line on	*
	/* cursor's line.	*
	/*	*

E6EA- A6 D8	NEWLN	LDX *TBLX ;get current line number
E6EC- E8		INX
E6ED- E0 18		CPX #24 ;if on last line but one
E6EF- F0 33		BEQ BLKLN ;blank this line
E6F1- 90 03		BCC NEWLX ;if on last line
E6F3- 4C C4 E1		JMP SCRL ;insert a line
E6F6- A2 17	NEWLX	LDX #23 ;else start with bottom line
E6F8- B5 E1	NEXL1	LDA *LDTB1+1,X ;add MSB
E6FA- 09 80		ORA #\$80 ;and set up pointer
E6FC- 85 C8		STA *SAH ;get previous linemarker
E6FE- B4 E0		LDY *LDTB1,X ;if 'extended'
E700- 30 02		BMI NEWLA ;remove MSB
E702- 29 7F		AND #\$7F ;and put in table
E704- 95 E1	NEWLA	STA *LDTB1+1,X ;move marker of previous line
E706- 98		TYA ;add MSB
E707- 09 80		ORA #\$80 ;and set up pointer
E709- 85 C5		STA *PNT+1 ;40 characters to be moved
E70B- A0 27		LDY #39 ;get lo-byte of lower line
E70D- BD 99 E7		LDA LDTB2+1,X ;complete pointer
E710- 85 C7		STA *SAL ;get lo-byte of upper line
E712- BD 98 E7		LDA LDTB2,X ;complete pointer
E715- 85 C4		STA *PNT ;get character of upper line
E717- B1 C4	NELL	LDA (PNT),Y ;store in lower line
E719- 91 C7		STA (SAL),Y
E71B- 88		DEY
E71C- 10 F9		BPL NELL ;repeat until all char.s moved

```

.E71E- CA          DEX          ;if not all lines done
E71F- E4 08        CPX *TBLX
E721- D0 D5        BNE NEXL1   ;repeat for other lines
E723- E8          INX
E724- B5 E0        BLKLN       LDA *LDTB1,X ;get marker of top line
E726- 09 80        ORA #$80    ;add MSB
E728- 85 C5        STA *PNT+1  ;set up pointer
E72A- 29 7F        AND #$7F    ;set line 'extended'
E72C- 95 EO        STA *LDTB1,X ;into table
E72E- B0 98 E7        LDA LDTB2,X ;get lo-byte
E731- 85 C4        STA *PNT    ;complete pointer
E733- A0 27        LDY #39
E735- A9 20        LDA #$20    ;get blank
E737- 91 C4        BLKL        STA (PNT),Y ;fill top line with blanks
E739- 88          DEY
E73A- 10 FB        BPL BLKL    ;until whole line filled
E73C- 4C 71 EO        JMP STUPT   ;and get pointer to current line
.FI "BASIC4.OEF4.M06"

```

·1980 340B-4D98· BASIC4.OEF4.M06

```

;name BASIC4.OEF4.M06
;*****
;*
;* Keyboard table, for 80 keys.
;*
;*****
E73F- 30 2E 10  CHAR .BY '=' $10 $03 '<' ' ' $12 ;scan line nine
E742- 03 3C 20
E745- 5B 12
E747- 20 30 00 .BY '-0' $00 '>' $FF '30' $00 ;scan line eight
E74A- 3E FF 50
E74D- 40 00
E74F- 2B 32 FF .BY '+2' $FF '?,NVX' ;scan line seven
E752- 3F 2C 4E
E755- 56 58
E757- 33 31 00 .BY '31' $0D ';'MBCZ' ;scan line six
E75A- 3B 40 42
E75D- 43 5A
E75F- 2A 35 FF .BY '*5' $FF ':KHFS' ;scan line five
E762- 3A 4B 48
E765- 46 53
E767- 36 34 FF .BY '64' $FF 'LJGDA' ;scan line four
E76A- 4C 4A 47
E76D- 44 41
E76F- 2F 38 FF .BY '/8' $FF 'PIYRW' ;scan line three
E772- 50 49 59
E775- 52 57
E777- 39 37 5E .BY '97^OUTEQ' ;scan line two
E77A- 4F 55 54
E77D- 45 51
E77F- 14 11 09 .BY $14 $11 $09 ')`' $27 '$' ;scan line one
E782- 29 5C 27
E785- 24 22
E787- 1D 13 5F .BY $1D $13 '_(8%#!' ;scan line zero
E78A- 28 26 25
E78D- 23 21
;*****
;*
;* Message: D shift-L "* RETURN
;* RUN RETURN
;*
;*****

```

```

E78F- 44 CC 22 RUNTB .BY 'D' $CC '"*' $0D 'RUN' $0D
E792- 2A 0D 52
E795- 55 4E 0D
;*****
;*
;* Table of lo-bytes of screen *
;* line addresses. *
;*
;*****
E798- 00 28 50 LDTB2 .BY 0 40 80 120 160 200 240
E79B- 78 A0 C8
E79E- F0
E79F- 18 40 68 .BY 24 64 104 144 184 224
E7A2- 90 B8 E0
E7A5- 08 30 58 .BY 8 48 88 128 168 208 248
E7A8- 80 A8 D0
E7AB- F8
E7AC- 20 48 70 .BY 32 72 112 152 192
E7AF- 98 C0
;*****
;*
;* Two tables of 18 constants *
;* each for the CRT-controller. *
;* The first table is for text *
;* mode, the second for graphics *
;* mode. *
;*
;*****
;text mode table
E7B1- 31 e7b1 .BY 49 ;horizontal total
E7B2- 28 .BY 40 ;horizontal displayed
E7B3- 29 .BY 41 ;horizontal sync position
E7B4- 0F .BY 15 ;horizontal sync width
E7B5- 27 .BY 39 ;vertical total
E7B6- 00 .BY 0 ;vertical total adjust
E7B7- 19 .BY 25 ;vertical displayed
E7B8- 20 .BY 32 ;vertical sync position
E7B9- 00 .BY 0 ;interlace mode (off)
E7BA- 09 .BY 9 ;maximum scan line address
E7BB- 00 .BY 0 ;cursor start
E7BC- 00 .BY 0 ;cursor end
E7BD- 10 .BY 16 ;start address (hi) (inverts video)
E7BE- 00 .BY 0 ;start address (lo)
E7BF- 00 .BY 0 ;cursor (hi)
E7C0- 00 .BY 0 ;cursor (lo)
E7C1- 00 .BY 0 ;light pen (hi)
E7C2- 00 .BY 0 ;light pen (lo)
;graphics mode table
E7C3- 31 e7c3 .BY 49 ;horizontal total
E7C4- 28 .BY 40 ;horizontal displayed
E7C5- 29 .BY 41 ;horizontal sync position
E7C6- 0F .BY 15 ;horizontal sync width
E7C7- 31 .BY 49 ;vertical total
E7C8- 00 .BY 0 ;vertical total adjust
E7C9- 19 .BY 25 ;vertical displayed
E7CA- 25 .BY 37 ;vertical sync position
E7CB- 00 .BY 0 ;interlace mode (off)
E7CC- 07 .BY 7 ;maximum scan line address
E7CD- 00 .BY 0 ;cursor start
E7CE- 00 .BY 0 ;cursor end
E7CF- 10 .BY 16 ;start address (hi) (inverts video)
E7D0- 00 .BY 0 ;start address (lo)
E7D1- 00 .BY 0 ;cursor (hi)
E7D2- 00 .BY 0 ;cursor (lo)

```

```

E7D3- 00          .BY 0           ;light pen (hi)
E7D4- 00          .BY 0           ;light pen (lo)
;*****
;*                                     *
;* Constants for bell.               *
;*                                     *
;*****                                     /
E7D5- 0E 1E 3E    e7d5        .BY $0E $1E $3E $7E $3E $1E $0E
E7D8- 7E 3E 1E
E7DB- 0E

;*****
;*                                     *
;* Table of masks for TABs.         *
;*                                     *
;*****                                     /
E7DC- 80 40 20    e7dc        .BY $80 $40 $20 $10 $08 $04 $02 $01
E7DF- 10 08 04
E7E2- 02 01

;*****
;*                                     *
;* Checksum byte (over E-ROM).     *
;*                                     *
;*****                                     /
E7E4- 29          CKSUME .BY $29      ;checksum byte
;*****
;*                                     *
;* The rest of the ROM is filled   *
;* with $AA bytes.                 *
;*                                     *
;*****                                     /
;
;*****
;*                                     *
;* External Labels.                 *
;*                                     *
;*****                                     /
CTIMR  .DE $0080      ;the jiffy clock
CINV   .DE $0090      ;IRQ ram vector
CBINV  .DE $0092      ;BRK RAM vector
LSTX   .DE $0097      ;last scan index
SHFLAG .DE $0098      ;shiftkey indicator
NDX    .DE $009E      ;number of keys in keyboard buffer
RVS    .DE $009F      ;flag for reverse field printing
INDX   .DE $00A1      ;record length of current screen line
LXSP   .DE $00A3      ;cursor row/column, used in INPUT and G
SFDX   .DE $00A6      ;current scan index
BLNSW  .DE $00A7      ;flag: cursor on
BLNCT  .DE $00A8      ;countdown till next cursor blink
-      GDBLN .DE $00A9      ;real character under cursor
BLNON  .DE $00AA      ;cursor's blink phase
CRSW   .DE $00AC      ;flag: input from screen or keyboard
DFLTN  .DE $00AF      ;current input device number
DFLTO  .DE $00B0      ;current output device number
PNT    .DE $00C4      ;screen line address, lo
POINT  .DE $00C5      ;screen line address, hi
PNTR   .DE $00C6      ;current column position of cursor
SAL    .DE $00C7      ;pointer in screen scroll
SAH    .DE $00C8      ;pointer in screen scroll, hi
QTSW   .DE $00CD      ;flag: nonzero if in quote mode
FNLEN  .DE $00D1      ;temp. workarea for CRTC setup
LNMX   .DE $0005      ;length of current screen line
TBLX   .DE $0008      ;current line number of cursor
DATA   .DE $00D9      ;last key input
INSRT  .DE $00DC      ;number of inserts outstanding

```

LDTB1	.DE \$00E0	;screen line wrap around table
CAS1	.DE \$00F9	;flag: cassette #1 motor on
CAS2	.DE \$00FA	;flag: cassette #2 motor on
PI	.DE \$00FF	;value for PI
KEYD	.DE \$026F	;keyboard buffer
DELAY	.DE \$03E9	;delay for repeat key
KOUNT	.DE \$03EA	;repeat rate for repeat key
XMAX	.DE \$03EB	;maximum keyboard buffer length
RPTFLG	.DE \$03EE	;flag: REPEAT pressed
UDTIM	.DE \$FFEA	;update jiffy clock
PIAL	.DE \$E810	;keyboard PIA: I/O port A and DDR
PIAL1	.DE \$E811	;keyboard PIA: control register A
PIAK	.DE \$E812	;keyboard PIA: I/O port B and DDR
PIAS	.DE \$E813	;keyboard PIA: control register B
IEEEI	.DE \$E820	;IEEE PIA: I/O port A and DDR
IEEIS	.DE \$E821	;IEEE PIA: control register A
IEEO	.DE \$E822	;IEEE PIA: I/O port B and DDR
IEEOS	.DE \$E823	;IEEE PIA: control register B
PIA	.DE \$E840	;VIA: I/O port B
SYNC	.DE \$E841	;VIA: I/O port A with handshake
P2DB	.DE \$E842	;VIA: port B data direction register
P2DA	.DE \$E843	;VIA: port A data direction register
T1L	.DE \$E844	;VIA: timer 1 lo
T1H	.DE \$E845	;VIA: timer 1 hi
T1LL	.DE \$E846	;VIA: timer 1 lo, latch
T1LH	.DE \$E847	;VIA: timer 1 hi, latch
T2L	.DE \$E848	;VIA: timer 2 lo
T2H	.DE \$E849	;VIA: timer 2 hi
SR	.DE \$E84A	;VIA: shift register
ACR	.DE \$E84B	;VIA: auxiliary control register
PCR	.DE \$E84C	;VIA: peripheral control register
IFR	.DE \$E84D	;VIA: interrupt flag register
IER	.DE \$E84E	;VIA: interrupt enable register
SYNC1	.DE \$E84F	;VIA: port A without handshake
CRO	.DE \$E880	;CRTC: address register
CR1	.DE \$E881	;CRTC: register selected through CRO
	.EN	

END MAE PASS

LABELFILE

ACR =E84B	BK1 =E24A	BK2 =E255
BKLN =E1DE	BKLN1 =E1EA	BLKL =E737
BLKLN =E724	BLNCT =00A8	BLNON =00AA
BLNSW =00A7	CAS1 =00F9	CAS2 =00FA
CBINV =0092	CHAR =E73F	CINT =E000
CINT1 =E036	CINV =0090	CKIS1 =E504
CKIS2 =E4F6	CKIT =E50B	CKIT1 =E516
CKSUME =E7E4	CKUT =E50A	CLP1 =E158
CLP2 =E141	CLP21 =E156	CLP2A =E153
CLP5 =EOF2	CLP6 =EOFB	CLP7 =E166
CLSR =E042	CNC3 =E25F	CNC3X =E23B
CRO =E880	CR1 =E881	CRSW =00AC
CTIMR =008D	DATA =0009	DELAY =03E9
DFLTN =00AF	DFLTO =00B0	DSPP =E606
FNLEN =00D1	GDBLN =00A9	IEEEI =E820
IEEIS =E821	IEEO =E822	IEEOS =E823
IER =E84E	IFR =E84D	INDX =00A1
INS1 =E306	INS2 =E308	INS3 =E2FF
INSRT =00DC	JLP2 =E370	JPL3 =E299
JPL4 =E282	JSTS =E185	JSTS1 =E19F

JSTS2 =E1CD	JSTSX =E1B3	JSXB =E1BD
JTS2 =E1BE	KEY =E455	KEY3 =E496
KEY4 =E47B	KEY5 =E477	KEYD =026F
KL1 =E4E7	KL2 =E4A2	KL22 =E4D0
KL23 =E4AD	KL24 =E49F	KL25 =E4B6
KN1 =E57A	KOUNT =03EA	LDTB1 =00ED
LDTB2 =E798	LNMX =00D5	LOOP2 =E1A6
LOOP3 =E0BF	LOOP4 =E0BC	LOOP5 =E116
LOP2 =E1AE	LOP5 =E11E	LOP51 =E124
LOP52 =E134	LOP53 =E138	LOP54 =E12E
LP1 =E0AC	LP2 =E0A7	LP21 =E0D3
LP22 =E0EA	LP23 =E0DF	LPS1 =E048
LPS3 =E050	LPS4 =E05C	LSTX =0097
LXSP =00A3	MLP1 =E3FB	MLP2 =E40A
MLP41 =E41E	MLP42 =E427	NC1 =E26A
NC2 =E271	NC3 =E17A	NC3W =E262
NCX2 =E287	NCZ2 =E284	NDX =009E
NELL =E717	NEWLA =E704	NEWLN =E6EA
NEWLX =E6F6	NEXL1 =E6F8	NJTL =E228
NTCN =E234	NTCN1 =E25B	NTCN2 =E1F3
NVS =E17C	NVSL =E182	NXLN =E3A9
NXLN1 =E3B6	NXLN2 =E3AE	NXT2 =E353
NXT3 =E176	NXT33 =E174	NXT6 =E35D
NXTD =E06B	NXTL =E3C6	NXTX =E2D5
NXTX1 =E2DD	P2DA =E843	P2DB =E842
PCR =E84C	PI =0OFF	PIA =E840
PIAK =E812	PIAL =E810	PIAL1 =E811
PIAS =E813	PNT =00C4	PNTR =00C6
POINT =00C5	PREND =E600	PRENDO =E012
PRT =E202	PULS =E442	PULS1 =E452
PX1 =E68C	QTSW =00CD	QTSWC =E167
QTSWL =E173	RPTFLG =03EE	RUNTB =E78F
RVS =009F	SAH =00C8	SAL =00C7
SCRL =E1C4	SCRL1 =E3F2	SCRL3 =E3ED
SCROL =E3D1	SFDX =00A6	SHFLAG =0098
SR =E84A	STUPR =E098	STUPT =E071
STUPZ =E08E	SYNC =E841	SYNC1 =E84F
T1H =E845	T1L =E844	T1LH =E847
T1LL =E846	T2H =E849	T2L =E848
TBLX =00D8	UDTIM =FFEA	UP1 =E334
UP2 =E324	UP3 =E343	UP5 =E2EB
UP6 =E31F	UP9 =E31B	XMAX =03EB
e21a =E2:1A	e221 =E221	e29c =E29C
e2a3 =E2A3	e2ac =E2AC	e2b5 =E2B5
e2be =E2BE	e2c2 =E2C2	e2c8 =E2C8
e2cd =E2CD	e2e4 =E2E4	e36c =E36C
e376 =E376	e37e =E37E	e387 =E387
e390 =E390	e399 =E399	e3bf =E3BF
e3c2 =E3C2	e3d5 =E3D5	e3d7 =E3D7
e404 =E404	e42e =E42E	e458 =E458
e469 =E469	e4b9 =E4B9	e4bf =E4BF
e511 =E511	e523 =E523	e53e =E53E
e548 =E548	e556 =E556	e587 =E587
e588 =E588	e60f =E60F	e617 =E617
e61d =E61D	e630 =E630	e63c =E63C
e654 =E654	e657 =E657	e668 =E668
e671 =E671	e682 =E682	e683 =E683
e693 =E693	e7b1 =E7B1	e7dc =E7DC
e7d5 =E7D5	//0000,E7E5,E7E5	

```

;NAME BASIC4.OE8B.CTL
;*****
;*          *
;*      E-ROM for BASIC 4.0      *
;*          *
;*      for 80 column 12 inch    *
;*      screen models with      *
;*          B-keyboard           *
;*          *
;* Date 07-10-83  Time 22:21   *
;*          *
;* Made by:                    *
;*          *
;* Nico de Vries               *
;* Mari Andriessenrade 49     *
;* 2907 MA Capelle a/d Yssel  *
;* The Netherlands              *
;*          *
;* Original CBM ROM-number    *
;*          *
;*          1474-04                *
;*          *
;*****.BA $E000
.FI "BASIC4.OE8B.M01"

```

194A 33C0-4D0A BASIC4.OE8B.M01

```

;name BASIC4.OE8B.M01
;*****
;*          *
;* Jump menu, enables standardi- *
;* sation between various machine *
;* types.                         *
;*          *
;*****.E000- 4C 4B E0  CINT    JMP e04b      ;Initialize I/O, cursor home and 4 bell
.E003- 4C A7 E0    JMP LP2       ;Get character from keyboard buffer
.E006- 4C 16 E1    JMP LOOPS    ;Get character from device 0 or 3
.E009- 4C 02 E2    JMP PRT      ;Put character to screen
.E00C- 4C 42 E4    JMP PULS     ;Service interrupt
.E00F- 4C 55 E4    JMP KEY      ;Default interrupt routine
.E012- 4C 00 E6  PRENDO  JMP PREND    ;Exit from interrupt
.E015- 4C 51 E0    JMP CLSR     ;Clear screen within window
.E018- 4C 7A E0    JMP e07a     ;Set to lower/upper case
.E01B- 4C 82 E0    JMP e082     ;Set to upper case/graphics
.E01E- 4C 88 E0    JMP e088     ;Other CRT controller settings
.E021- 4C C8 E3    JMP NEWLN    ;Scroll screen down
.E024- 4C E8 E3    JMP SCROL    ;Scroll screen up
.E027- 4C BE E4    JMP e4be     ;Find key from decoding table
.E02A- 4C A7 E6    JMP e6a7     ;Ring bell once
.E02D- 4C 36 E0    JMP e036     ;Store accu in repeat flag
.E030- 4C E1 E1    JMP e1e1     ;Set top left of window
.E033- 4C DC E1    JMP e1dc     ;Set right bottom of window
;*****
;*          *
;* Store A in repeatflag (not  *
;* used).                      *
;*          *
;*****.FI "BASIC4.OE8B.M01"

```

E036- 85 E4	e036	STA *\$E4	;store A in repeat flag
E038- 60		RTS	
E039- AA		TAX	
E03A- AA		TAX	
E03B- AA		TAX	
E03C- AA		TAX	
E03D- AA		TAX	
E03E- AA		TAX	
E03F- AA		TAX	
E040- AA		TAX	
E041- AA		TAX	
E042- AA		TAX	
E043- AA		TAX	
E044- AA		TAX	
E045- AA		TAX	
E046- AA		TAX	
E047- AA		TAX	
E048- AA		TAX	
E049- AA		TAX	
E04A- AA		TAX	
;*****			
;*			
;* Reset routine.			
;*			
;*****			
E04B- 20 OF E6	e04b	JSR e60f	;Initialize I/O and zero page
E04E- 20 7A EO		JSR e07a	;set CRT controller to lower/upper case
;*****			
;*			
;* Initialize screen; clear			
;* screen within window.			
;*			
;*****			
E051- A6 EO	CLSR	LDX *XREC	;get top line # of window
E053- CA		DEX	
E054- E8	e054	INX	;do next line
E055- 20 6C EO		JSR STUPR	;get pointer to line
E058- 20 C1 E1		JSR e1c1	;erase that line till end of window
E05B- E4 E1		CPX *XWRT	;until bottom of window done
E05D- 90 F5		BCC e054	
;*****			
;*			
;* Home cursor (within window).			
;*			
;*****			
E05F- A6 EO	NXTD	LDX *XREC	;get top line # of window
E061- 86 D8		STX *TBLX	;set cursor on that line
E063- A4 E2	e063	LDY *\$E2	;get left margin of window
E065- 84 C6		STY *PNTR	;set cursor there
E067- A6 D8	STUPT	LDX *TBLX	;get line # again
E069- 4C 6F EO		JMP e06f	;and get pointer to line
E06C- A4 E2	STUPR	LDY *\$E2	;get left margin of window
E06E- 88		DEY	
;*****			
;*			
;* Get pointer to current screen *			
;* line.			
;*			
;*****			
E06F- BD 55 E7	e06f	LDA LDTB2,X	;get lo-byte of pntr
E072- 85 C4		STA *PNT	;store in pointer
E074- BD 6E E7		LDA LDTB1,X	;get hi-byte
E077- 85 C5		STA *PNT+1	;and complete pointer
E079- 60		RTS	

```

;*****
;*
;* Set CRT controller to text *
;* mode. *
;*
;*****
E07A- A9 2A    e07a   LDA #L,e72a      ;get lo-byte of pointer
E07C- A2 E7    LDX #H,e72a      ;and hi-byte also
E07E- A0 0E    LDY #$0E        ;and get lo-nibble for VIA
E080- D0 06    BNE e088        ;and set up CRT controller
;*****
;*
;* Set CRT controller to graphics *
;* mode. *
;*
;*****
E082- A9 3C    e082   LDA #L,e73c      ;get lo-byte of pointer
E084- A2 E7    LDX #H,e73c      ;and hi-byte also
E086- A0 0C    LDY #$0C        ;and get lo-nibble for VIA
E088- 85 C7    e088   STA *SAL        ;store pointer
E08A- 86 C8    STX *SAH        ;get VIA register
E08C- AD 4C E8  LDA PCR         ;clear lo-nibble
E08F- 29 F0    AND #$F0        ;and save it
E091- 85 D1    STA *FNLEN       ;add to register value
E093- 98        TYA             ;move 18 constants
E094- 05 D1    ORA *FNLEN       ;select CRTC register
E096- 8D 4C E8  STA PCR         ;adapt index in buffer
E099- A0 11    LDY #17        ;enable interrupts again
E09B- B1 C7    e09b   LDA (SAL),Y   ;until all char's done
E09D- 8C 80 E8  STY CRO         ;get character in accu
E0A0- 8D 81 E8  STA CRI         ;echo character on screen
E0A3- 88        DEY             ;if empty, set cursor visible
E0A4- 10 F5    BPL e09b        ;else wait for character
E0A6- 60        RTS             ;if blink fase one
;*****
;*
;* Get character from keyboard *
;* buffer. *
;*
;*****
E0A7- AC 6F 02  LP2   LDY KEYD       ;get oldest character
E0AA- A2 00    LDX #$00        ;move rest of buffer
E0AC- BD 70 02  LP1   LDA KEYD+1,X  ;foreward
E0AF- 9D 6F 02  STA KEYD,X     ;adapt index in buffer
E0B2- E8        INX             ;enable interrupts again
E0B3- E4 9E    CPX *NDX        ;until all char's done
E0B5- D0 F5    BNE LP1         ;get character in accu
E0B7- C6 9E    DEC *NDX        ;echo character on screen
E0B9- 98        TYA             ;if empty, set cursor visible
E0BA- 58        CLI             ;else wait for character
E0BB- 60        RTS             ;if blink fase one
;*****
;*
;* Wait for RETURN from keyboard. *
;*
;*****
E0BC- 20 8C E6  LOOP4  JSR e68c       ;get true char. at cursor position
E0BF- A5 9E    LOOP3  LDA *NDX        ;get index in keyb. buffer
E0C1- 85 A7    STA *BLNSW       ;if empty, set cursor visible
E0C3- F0 FA    BEQ LOOP3        ;else wait for character
E0C5- 78        SEI             ;disable interrupts
E0C6- A5 AA    LDA *BLNON       ;if blink fase one
E0C8- F0 09    BEQ LP21         ;get character in accu
E0CA- A5 A9    LDA *GDBLN       ;echo character on screen

```

E0CC- A0 00		LDY #\$00	
EOCE- 84 AA		STY *BLNON	;set blink fase zero
E0D0- 20 06 E6		JSR DSPP	;and put char to screen
E0D3- 20 A7 EO	LP21	JSR LP2	;get character from buffer
E0D6- C9 83		CMP #\$83	;if shift-STOP
E0D8- D0 10		BNE LP22	
E0DA- 78		SEI	;disable interrupts
E0DB- A2 09		LDX #9	;move ten characters
E0DD- 86 9E		STX *NDX	;set index in buffer to 9
E0DF- BD 20 E7	LP23	LDA RUNTB-1,X	;move standard string
E0E2- 9D 6E 02		STA KEYD-1,X	;to buffer
E0E5- CA		DEX	
E0E6- D0 F7		BNE LP23	
E0E8- F0 D5		BEQ LOOP3	;and get 1st char. of tha
E0EA- C9 0D	LP22	CMP #\$0D	;if is not RETURN
E0EC- D0 CE		BNE LOOP4	;go echo to screen and get next char
E0EE- A4 D5		LDY *LNMX	;else get max. length
E0F0- 84 AC		STY *CRSW	;flag input from screen
E0F2- B1 C4	CLP5	LDA (PNT),Y	;get char. from screen
E0F4- C9 20		CMP #\$20	;if space at end of line
E0F6- D0 03		BNE CLP6	
E0F8- 88		DEY	;get previous character
E0F9- D0 F7		BNE CLP5	
E0FB- C8	CLP6	INY	
E0FC- 84 A1		STY *INDX	;set record length of current line
E0FE- 20 CB E1		JSR e1cb	;set cursor on left margin
E101- EA		NOP	;(returns with Y zero)
E102- 84 CD		STY *QTSW	;clear quote flag
E104- A5 A3		LDA *LXSP	;get line number on screen
E106- 30 19		BMI LOPS	;if valid
E108- C5 D8		CMP *TBLX	;and equal to original
E10A- D0 15		BNE LOPS	
E10C- A5 A4		LDA *LXSP+1	;set position
E10E- 85 C6		STA *PNTR	;to original value
E110- C5 A1		CMP *INDX	;if not behind end of record
E112- 90 00		BCC LOPS	;get char from device 3
E114- B0 2E		BCS CLP2	;else return RETURN

;* *			
;* Get character from device 0 or *			
;* device 3. *			
;* *			

E116- 98	LOOP5	TYA	
E117- 48		PHA	;save Y
E118- 8A		TXA	
E119- 48		PHA	;save X
E11A- 6C E9 00		JMP (\$00E9)	;and get input (default \$E11D)
E11D- A5 AC	e11d	LDA *CRSW	;if input from keyboard
E11F- F0 9E		BEQ LOOP3	;go do keyboard until RETURN
.FI "BASIC4.OE8B.M02"			

1368 33C0-4728 BASIC4.OE8B.M02

;name BASIC4.OE8B.M02			

;* *			
;* Get character from current *			
;* screen line. *			
;* *			

E121- A4 C6	LOPS	LDY *PNTR	;get position
E123- B1 C4		LDA (PNT),Y	;get screen character

E125- 85 D9		STA *DATA	;save it
E127- 29 3F	LOP51	AND #\$3F	;remove bits 6 and 7
E129- 06 D9		ASL *DATA	
E12B- 24 D9		BIT *DATA	;if original bit 6 set
E12D- 10 02		BPL LOP54	
E12F- 09 80		ORA #\$80	;set bit 7 (shift bit)
E131- 90 04	LOP54	BCC LOP52	;if original bit 7 set
E133- A6 C0		LDX *QTSW	;and if in quote mode
E135- D0 04		BNE LOP53	;don't change bit 6
E137- 70 02	LOP52	BVS LOP53	;else if bit 5 set in original
E139- 09 40		ORA #\$40	;set bit 6
E13B- E6 C6	LOP53	INC *PNTR	;adapt position
E13D- 20 6A E1		JSR QTSWC	;adapt quote flag
E140- C4 A1		CPY *INDX	;if max position reached
E142- D0 17		BNE CLP1	
E144- A9 00	CLP2	LDA #\$00	
E146- 85 AC		STA *CRSW	;flag input from keyboard
E148- A9 00		LDA #\$00	;and echo RETURN
E14A- A6 AF		LDX *DFLTN	;if input device
E14C- E0 03		CPX #3	;is screen
E14E- F0 06		BEQ CLP2A	;go echo on screen
E150- A6 B0		LDX *DFLTO	;else if output device
E152- E0 03		CPX #3	;is screen
E154- F0 03		BEQ CLP21	;store RETURN in last input
E156- 20 02 E2	CLP2A	JSR PRT	;echo on screen
E159- A9 00	CLP21	LDA #\$00	;and store a RETURN
E15B- 85 D9	CLP1	STA *DATA	;in last key input.
E15D- 68		PLA	
E15E- AA		TAX	;restore X
E15F- 68		PLA	
E160- A8		TAY	;restore Y
E161- A5 D9		LDA *DATA	;get character again
E163- C9 DE		CMP #\$DE	;if code for PI
E165- D0 02		BNE CLP7	
E167- A9 FF		LDA #PI	;get real code for PI
E169- 60	CLP7	RTS	
		;	*****
		;	*
		;	* Toggle quote flag if " found. *
		;	*
		;	*****
E16A- C9 22	QTSWC	CMP #'"	;if character is "
E16C- D0 08		BNE QTSWL	
E16E- A5 C0		LDA *QTSW	;get quote flag
E170- 49 01		EOR #\$01	;invert bit 0
E172- 85 C0		STA *QTSW	
E174- A9 22		LDA #'"	;and restore character
E176- 60	QTSWL	RTS	
		;	*****
		;	*
		;	* Fill screen at current *
		;	* position. *
		;	*
		;	*****
E177- 09 40	NXT33	ORA #\$40	;map shifted characters
E179- A6 9F	NXT3	LDX *RVS	;if reverse switch on
E17B- F0 02		BEQ NVS	
E17D- 09 80	NC3	ORA #\$80	;set screencode for reverse
E17F- A6 DC	NVS	LDX *INSRT	;if in insert mode
E181- F0 02		BEQ NVSL	
E183- C6 DC		DEC *INSRT	;decrease insert count
E185- 20 06 E6	NVSL	JSR DSPP	;put char at current position in RAM
E188- E6 C6	JSTS	INC *PNTR	;advance character position
E18A- A4 05		LDY *LNMX	;if beyond maximum length

```

E18C- C4 C6      CPY *PNTR
E18E- B0 09      BCS LOOP2
E190- A6 D8      LDX *TBLX      ;get line number in X
E192- 20 A3 E3  JSTS1  JSR NXLN      ;scroll window if necessary
E195- A4 E2      LDY *$E2      ;get left margin
E197- 84 C6      STY *PNTR      ;put cursor there
;*****
;*          *
;* Exit from output to screen.   *
;*          *
;*****
E199- A9 00      LDA #0
E19B- 85 E8      STA *$E8      ;clear HOME counter
E19D      PLA
E19E- A8          TAY      ;restore Y
E19F- A5 DC      LDA *INSRT      ;if inserts outstanding
E1A1- F0 02      BEQ LOP2
E1A3- 46 CD      LSR *QTSW      ;cancel quote mode
E1A5- 68          PLA
E1A6- AA          TAX      ;restore X
E1A7- 68          PLA      ;and A
E1A8- 58          CLI      ;allow interrupts again
E1A9- 60          RTS
;*****
;*          *
;* Backward over line boundary. *
;*          *
;*****
E1AA- A4 D5      BKLN     LDY *LNMX      ;get right margin
E1AC- A6 E0      LDX *XREC      ;and top line number of window
E1AE- E4 D8      CPX *TBLX      ;if on top line
E1B0- 90 08      BCC e1ba      ;set cursor home in window
E1B2- A4 E2      LDY *$E2      ;get left margin
E1B4- 84 C6      STY *PNTR      ;and put cursor home
E1B6- 68          PLA
E1B7- 68          PLA      ;remove return address
E1B8- D0 DF      BNE LOOP2      ;and exit
E1B9- C6 D8      e1ba      DEC *TBLX      ;move cursor to previous line
E1Bc- 84 C6      STY *PNTR      ;and put it on right margin
E1Bd- 4C 67 E0      JMP STUPT      ;and get pointer of that line
;*****
;*          *
;* Erase line until right margin. *
;*          *
;*****
E1C1- A9 20      e1c1      LDA #$20      ;get blank
E1C3- C8      e1c3      INY
E1C4- 91 C4      STA (PNT),Y      ;store on line
E1C6- C4 D5      CPY *LNMX      ;until right margin reached
E1C8- 90 F9      BCC e1c3
E1CA- 60          RTS
;*****
;*          *
;* Set cursor on left margin.   *
;*          *
;*****
E1CB- A4 E2      e1cb      LDY *$E2      ;get left margin
E1CD- 84 C6      STY *PNTR      ;put cursor there
E1CF- A0 00      LDY #0      ;prepare for cancel quote mode
E1D1- 60          RTS
;*****
;*          *
;* Set window to full screen.   *
;*          *

```

```

E1D2- A9 00      e1d2    ;*****
                           LDA #0          ;get top line #
                           TAX           ;and left margin
                           JSR e1e1        ;set them
                           LDA #24          ;get bottom line #
                           LDX #79          ;and right margin
                           ;*****
                           ;*
                           ;* Set bottom right of window. *
                           ;*
                           ;*****
E1DC- 85 E1      e1dc    STA *XWRT       ;store bottom line #
E1DE- 86 D5      RTS     STX *LNMX       ;and right margin
E1EO- 60          RTS     ;*****
                           ;*
                           ;* Set top left of window. *
                           ;*
                           ;*****
E1E1- 85 EO      e1e1    STA **XREC      ;set top line #
E1E3- 86 E2      STA **$E2       ;and left margin
E1E5- 60          RTS
E1E6- AA          TAX
E1E7- AA          TAX
E1E8- AA          TAX
E1E9- AA          TAX
E1EA- AA          TAX
E1EB- AA          TAX
E1EC- AA          TAX
E1ED- AA          TAX
E1EE- AA          TAX
E1EF- AA          TAX
E1FD- AA          TAX
E1F1- AA          TAX
E1F2- AA          TAX
E1F3- AA          TAX
E1F4- AA          TAX
E1F5- AA          TAX
E1F6- AA          TAX
E1F7- AA          TAX
E1F8- AA          TAX
E1F9- AA          TAX
E1FA- AA          TAX
E1FB- AA          - TAX
E1FC- AA          TAX
E1FD- AA          TAX
E1FE- AA          TAX
E1FF- AA          TAX
E200- AA          TAX
E201- AA          TAX
.FI "BASIC4.OE8B.M03"

```

17BF 33C0-4B7F BASIC4.OE8B.M03

```

;name BASIC4.OE8B.M03
;*****
;*
;* Put character to screen.
;*
;*****
E202- 48          PRT    PHA             ;save character
E203- 85 D9      STA *DATA        ;also in temporary field
E205- 8A          TXA

```

E206-	48	PHA	;save X
E207-	98	TYA	
E208-	48	PHA	;and Y
E209-	6C EB 00	JMP (\$00EB)	;and go output (default e20c)
E20C-	A9 00	e20c LDA #0	
E20E-	85 AC	STA *CRSW	;set copy of position to zero
E210-	A4 C6	LDY *PNTR	;get position
E212-	A5 D9	LDA *DATA	;get character again
E214-	29 7F	AND #\$7F	;remove shift bit
E216-	C9 1B	CMP #\$1B	;if ESC or shift-ESC
E218-	D0 03	BNE e21d	
E21A-	4C B0 E3	JMP NXTL	;cancel quote mode and exit
E21D-	A5 D9	e21d LDA *DATA	;get character again
E21F-	10 03	BPL e224	;if shifted
E221-	4C F4 E2	JMP NXTX	;go do shifted character
E224-	C9 0D	e224 CMP #\$0D	;if RETURN
E226-	D0 03	BNE NJTL	
E228-	4C B6 E3	JMP e3b6	;go do RETURN
E22B-	C9 20	NJTL CMP #\$20	;if printable
E22D-	90 08	BCC NTCN	
E22F-	29 3F	AND #\$3F	;convert to screen code
E231-	20 6A E1	JSR QTSWC	;check for quote mode
E234-	4C 79 E1	JMP NXT3	;and put to screen
E237-	A6 DC	NTCN LDX *INSRT	;if inserts outstanding
E239-	F0 03	BEQ CNC3X	
E23B-	4C 7D E1	JMP NC3	;print as control character
E23E-	C9 14	CNC3X CMP #\$14	;if DELETE
E240-	D0 20	BNE NTCN1	
E242-	A4 E2	LDY #\$E2	;get left margin
E244-	C4 C6	CPY *PNTR	;if cursor after that
E246-	90 05	BCC e24d	;go move cursor left
E248-	20 AA E1	JSR BKLN	;else go backward over boundary
E24B-	D0 0F	BNE BK2	;and add a blank at the end
E24D-	C6 C6	e24d DEC *PNTR	;move cursor left
E24F-	A4 C6	LDY *PNTR	
E251-	C8	BK1 INY	;else
E252-	B1 C4	LDA (PNT),Y	;compress
E254-	88	DEY	;line
E255-	91 C4	STA (PNT),Y	;upto
E257-	C8	INY	;end of
E258-	C4 D5	CPY *LNMX	;right margin
E25A-	D0 F5	BNE BK1	
E25C-	A9 20	BK2 LDA #\$20	;and insert blank
E25E-	91 C4	STA (PNT),Y	;at end
E260-	D0 37	BNE JPL3	
E262-	A6 CD	NTCN1 LDX *QTSW	;if in quote mode
E264-	F0 03	BEQ NC3W	
E266-	4C 7D E1	CNC3 JMP NC3	;print as control character
E269-	C9 12	NC3W CMP #\$12	;if RVS-ON
E26B-	D0 02	BNE NC1	
E26D-	85 9F	STA *RVS	;set reverse flag
E26F-	C9 13	NC1 CMP #\$13	;if HOME
E271-	D0 10	BNE NC2	
E273-	A5 E8	LDA *\$E8	;get HOME counter
E275-	10 04	BPL e27b	;if second home
E277-	20 D2 E1	JSR e1d2	;set window to full screen
E27A-	18	CLC	
E27B-	66 E8	e27b ROR *\$E8	;adjust counter
E27D-	20 5F EO	JSR NXTD	;and set cursor to top left
E280-	4C 9D E1	JMP e19d	;and exit
E283-	C9 1D	NC2 CMP #\$1D	;if cursor-RIGHT
E285-	D0 0B	BNE NCX2	
E287-	C8	INY	
E288-	84 C6	STY *PNTR	;advance cursor

```

E28A- 88 DEY
E28B- C4 D5 CPY *LNMX ;if at right margin
E28D- 90 0A BCC JPL3
E28F- 4C 92 E1 JMP JSTS1 ;go to next line and exit
E292- C9 11 NCX2 CMP #$11 ;if cursor-DOWN
E294- D0 06 BNE e29c
E296- 20 A3 E3 JSR NXLN ;move cursor down and scroll if necessary
E299- 4C 99 E1 JPL3 JMP LOOP2 ;and exit
E29C- C9 09 e29c CMP #$09 ;if TAB
E29E- D0 30 BNE e2d0
E2A0- 20 70 E5 JSR e570 ;get bitmask and offset
E2A3- AC 3A 03 e2a3 LDY $033A ;get position of next tab
E2A6- EE 3A 03 INC $033A
E2A9- C4 D5 CPY *LNMX ;if after right margin
E2AB- 90 09 BCC e2b6
E2AD- A5 D5 LDA *LNMX ;get right margin
E2AF- 85 C6 STA *PNTR ;put cursor there
E2B1- CE 3A 03 DEC $033A
E2B4- D0 E3 BNE JPL3 ;and exit
E2B6- 0E 3E 03 e2b6 ASL $033E ;else adapt bitmask
E2B9- D0 0A BNE e2c5 ;if byte exceeded
E2BB- E8 INX ;adapt index
E2BC- E0 0A CPX #10 ;if at last possible index
E2BE- F0 D9 BEQ JPL3 ;exit
E2C0- A9 01 LDA #1 ;else set up
E2C2- 8D 3E 03 STA $033E ;new bitmask
E2C5- BD EE 03 e2c5 LDA $03EE,X ;get tabstopbyte
E2C8- 2D 3E 03 AND $033E ;test.mask, if no tabstop
E2CB- F0 D6 BEQ e2a3 ;check next column
E2CD- C8 INY ;else
E2CE- 84 C6 STY *PNTR ;put cursor there
E2D0- C9 16 e2d0 CMP #$16 ;if ERASE TO END OF LINE
E2D2- D0 0C BNE e2e0
E2D4- A9 20 LDA #$20
E2D6- 88 DEY ;get blank
E2D7- C8 e2d7 INY ;erase cursor's position also
E2D8- 91 C4 STA (PNT),Y ;why not use $E1C1?
E2DA- C4 D5 CPY *LNMX ;fill line with blanks
E2DC- 90 F9 BCC e2d7 ;until right margin reached
E2DE- B0 B9 BCS JPL3 ;if at end, exit
E2E0- C9 15 e2e0 CMP #$15 ;if not DELETE LINE
E2E2- F0 03 BEQ e2e7
E2E4- 4C 91 E5 JMP e591 ;do rest of control characters
E2E7- A5 E0 e2e7 LDA *XREC ;else get top line #
E2E9- 48 PHA ;save it
E2EA- A5 D8 LDA *TBLX ;get current line #
E2EC- 85 E0 STA *XREC ;store in top line #
E2EE- 20 E8 E3 JSR SCROL ;scroll rest of window up
E2F1- 4C CA E5 JMP e5ca ;restore top line # and exit
;*****
;*          *
;* Put shifted characters to      *
;* screen.                      *
;*          *
;*****



E2F4- 29 7F NXTX AND #$7F ;remove shift bit
E2F6- C9 7F CMP #PI-$80 ;if code for PI
E2F8- D0 02 BNE NXTX1
E2FA- A9 5E LDA #$5E ;get real code for PI
E2FC- C9 20 NXTX1 CMP #$20 ;if printable
E2FE- 90 03 BCC e303
E300- 4C 77 E1 JMP NXT33 ;go fill screen
E303- C9 00 e303 CMP #$0D ;if shift-RETURN
E305- D0 03 BNE UPS

```

E307- 4C B6 E3		JMP e3b6	;go do RETURN
E30A- A6 CD	UPS	LDX *QTSW	;if quote mode
E30C- D0 34		BNE UP6	;print as control character
E30E- C9 14		CMP #\$14	;no quote mode, if INSERT
E310- D0 2C		BNE UP9	
E312- A4 D5		LDY *LNMX	;get right margin
E314- B1 C4		LDA (PNT),Y	;get last char. of line
E316- C9 20		CMP #\$20	;if blank
E318- D0 72		BNE JLP2	
E31A- C4 C6		CPY *PNTR	;and not on last position
E31C- 90 6E		BCC JLP2	;then enough space on line
E31E- F0 6C		BEQ JLP2	
E320- A4 D5	INS1	LDY *LNMX	;get right margin
E322- 88	INS2	DEY	;move
E323- B1 C4		LDA (PNT),Y	;characters
E325- C8		INY	;one position
E326- 91 C4		STA (PNT),Y	;further
E328- 88		DEY	;until
E329- C4 C6		CPY *PNTR	;all done
E32B- D0 F5		BNE INS2	
E32D- A9 20		LDA #\$20	;and put a blank
E32F- 91 C4		STA (PNT),Y	;in space opened up
E331- A5 D5		LDA *LNMX	;get right margin
E333- 38		SEC	
E334- E5 C6		SBC *PNTR	;subtract position
E336- E5 DC		SBC *INSRT	;and number of inserts
E338- 30 S2		BMI JLP2	;if not positive, exit
E33A- E6 DC		INC *INSRT	;else add an insert
E33C- D0 4E		BNE JLP2	;and exit
E33E- A6 DC	UP9	LDX *INSRT	;if inserts outstanding
E340- F0 05		BEQ UP2	
E342- 09 40	UP6	ORA #\$40	;convert to screen code
E344- 4C 70 E1		JMP NC3	;and print as control char.
E347- C9 11	UP2	CMP #\$11	;if cursor-UP
E349- D0 00		BNE NXT2	
E34B- A6 E0		LDX *XREC	;get top line #
E34D- E4 D8		CPX *TBLX	;if on top line or above
E34F- B0 3B		BCS JLP2	;exit
E351- C6 D8		DEC *TBLX	;else move cursor up
E353- 20 67 EO		JSR STUPT	;get pointer to new line
E356- D0 34		BNE JLP2	;and exit
E358- C9 12	NXT2	CMP #\$12	;if RVS-off
E35A- D0 04		BNE NXT6	
E35C- A9 00		LDA #\$00	
E35E- 85 9F		STA *RVS	;reset reverse flag
E360- C9 10	NXT6	CMP #\$10	;if cursor-LEFT
E362- D0 0F		BNE e373	
E364- A4 E2		LDY *\$E2	;get left margin
E366- C4 C6		CPY *PNTR	;if left to that
E368- 90 05		BCC e36f	
E36A- 20 AA E1		JSR BKLN	;go backward over boundary
E36D- D0 10		BNE JLP2	;and exit
E36F- C6 C6	e36f	DEC *PNTR	;else move cursor left
E371- 10 19		BPL JLP2	;and exit
E373- C9 13	e373	CMP #\$13	;if CLR(-window)
E375- D0 05		BNE e37c	
E377- 20 51 EO		JSR CLSR	;clear the window
E37A- D0 10		BNE JLP2	;and exit
E37C- C9 09	e37c	CMP #\$09	;if shift-TAB
E37E- D0 0F		BNE e38f	
E380- 20 70 E5		JSR e570	;get bitmask and index
E383- B0 EE 03		LDA \$03EE,X	;get tabstopbyte
E386- 4D 3E 03		EOR \$03EE	;add or remove bit
E389- 9D EE 03		STA \$03EE,X	;store again

```

E38C- 4C 99 E1    JLP2      JMP LOOP2           ;and exit
E38F- C9 16        e38f      CMP #$16            ;if not ERASE FROM START
E391- F0 03        BEQ       e396              ;go do rest of characters
E393- 4C BC E5      e396      JMP e5bc            ;else get blank
E396- A9 20        LDA       #$20              ;start at left margin
E398- A4 E2        LDY       *$E2              ;if current position reached
E39A- C4 C6        e39a      CPY *PNTR           ;exit
E39C- B0 EE        BCS       JLP2              ;else store blank
E39E- 91 C4        STA       (PNT),Y          INY
E3A0- C8          INY
E3A1- D0 F7        BNE       e39a              ;and repeat
;*****
;*
;* Action for cursor down;
;* scroll window up if at bottom.
;*
;*****
E3A3- 46 A3        NXLN      LSR *LXSP           ;set copy of line # invalid
E3A5- A6 D8        LDX       *TBLX             ;get line #
E3A7- E4 E1        CPX       *XWRT             ;if at bottom of window
E3A9- 90 06        BCC       e3b1              JSR SCROL           ;scroll window up
E3AB- 20 E8 E3      JMP       STUPT            ;and get pointer to line
E3AE- 4C 67 EO      INC       *TBLX             ;else adapt line #
E3B1- E6 D8        e3b1      JMP STUPT           ;and get pointer
E3B3- 4C 67 EO      .FI "BASIC4.OE8B.M04"

```

1615 33C0-49D5 BASIC4.OE8B.M04

```

;name BASIC4.OE8B.M04
;*****
;*
;* Process RETURN.
;*
;*****
E3B6- A4 E2        e3b6      LDY *$E2            ;get left margin
E3B8- 84 C6        STY       *PNTR             ;put cursor there
E3BA- 20 B0 E3      JSR       NXTL              ;perform cursor down
E3BD- A9 00        NXTL      LDA #0              JSR STUPR           ;cancel inserts outstanding
E3BF- 85 DC        STA       *INSRT            ;cancel reverse field
E3C1- 85 9F        STA       *RVS              STA *QTSW             ;cancel quote mode
E3C3- 85 CD        STA       *QTSW
E3C5- 4C 99 E1      JMP       LOOP2             ;and exit
;*****
;*
;* Scroll window down.
;*
;*****
E3C8- A6 E1        NEWLN    LDX *XWRT           ;get bottom line #
E3CA- E8          INX
E3CB- CA          DEX
E3CC- 20 6C EO      NXTL1    JSR STUPR           ;get pointer and left margin
E3CF- E4 EO        CPX       *XREC             ;if top line of window reached
E3D1- F0 35        BEQ       MLP4              ;check for paused scroll
E3D3- B0 54 E7      LDA       LDTB2-1,X          STA *SAL             ;get lo-byte of previous line
E3D6- 85 C7        STA       *SAL              LDA LDTB1-1,X          ;get hi-byte also
E3D8- B0 6D E7      STA       *SAH              STA *SAH             ;store the pointer
E3D9- C8          NELL     INY
E3D9- B1 C7        LDA       (SAL),Y          STA (PNT),Y          ;get a character of prev. line
E3E0- 91 C4        STA       (PNT),Y          STA (PNT),Y          ;move to current line
E3E2- C4 D5        CPY       *LNMX             ;until right margin reached
E3E4- 90 F7        BCC       NELL

```

E3E6- B0 E3		BCS NXTL1 ;and do previous line ;***** ;* ;* Scroll window up. ;* ;*****
E3E8- A6 E0	SCROL	LOX *XREC ;get top line # DEX INX
E3EA- CA		
E3EB- E8	SCRLS	INX
E3EC- 20 6C E0		JSR STUPR ;get pointer and left margin
E3EF- E4 E1		CPX *XWRT ;if bottom reached
E3F1- B0 15		BCS MLP4 ;check for paused scroll
E3F3- B0 56 E7		LDA LDTB2+1,X ;get pointer to next line
E3F6- 85 C7		STA *SAL ;store lo-byte
E3F8- B0 6F E7		LDA LDTB1+1,X ;do hi-byte also
E3FB- 85 C8		STA *SAH
E3FD- C8	MLP1	INY ;do next character
E3FE- B1 C7		LDA (SAL),Y ;move from next line
E400- 91 C4		STA (PNT),Y ;to current line
E402- C4 D5		CPY *LNMX ;until right margin reached
E404- 90 F7		BCC MLP1
E406- B0 E3		BCS SCRLS ;then do next line ;***** ;* ;* Insert blank line and pause * ;* scroll if appropriate key is * ;* pressed. * ;* * ;*****
E408- 20 C1 E1	MLP4	JSR e1c1 ;insert blank line
E40B- AD 12 E8		LDA PIAK ;get key image
E40E- C9 FE		CMP #\$FE ;if left arrow pressed
E410- D0 0E		BNE e420
E412- A0 00		LDY #0
E414- EA	MLP41	NOP ;wait
E415- CA		DEX ;a
E416- D0 FC		BNE MLP41 ;while
E418- 88		DEY ;with a
E419- D0 F9		BNE MLP41 ;double loop
E41B- A0 00	MLP42	LDY #0
E41D- 84 9E		STY *NDX ;then cancel buffer
E41F- 60	e41f	RTS
E420- C9 DF	e420	CMP #\$DF ;if colon pressed
E422- D0 FB		BNE e41f
E424- AD 12 E8	e424	LDA PIAK ;get key image again
E427- C9 DF		CMP #\$DF ;wait until
E429- F0 F9		BEQ e424 ;colon released
E42B- C9 FF		CMP #\$FF ;then wait for
E42D- F0 F5		BEQ e424 ;other key pressed
E42F- D0 EA		BNE MLP42 ;then cancel buffer and exit ;***** ;* ;* Do clock ticks; * ;* add an extra tick in every * ;* five (because of 50 Hz screen * ;* refresh instead of the usual * ;* 60 Hz). * ;* * ;*****
E431- 20 EA FF	e431	JSR UDTIM ;do clock tick
E434- E6 F8		INC *\$F8 ;increment tick counter
E436- A5 F8		LDA *\$F8 ;if 5 ticks done
E438- C9 06		CMP #6
E43A- D0 1C		BNE e458 ;if not five, do keyboard

```

E43C- A9 00          LDA #0
E43E- 85 F8          STA *$F8      ;clear counter
E440- F0 EF          BEQ e431      ;and do extra tick
;*****
;*                                     *
;* Interrupt routine;                 *
;* entered 50 times per second.    *
;*                                     *
;* (Pointed to by ROM).           *
;*                                     *
;*****
E442- 48             PULS          PHA           ;save A
E443- 8A             TXA            TXA           ;save X
E444- 48             PHA           PHA           ;save Y
E445- 98             TYA            TYA           ;save Z
E446- 48             PULS          PHA           ;save A
E447- BA             TSX            TSX           ;save B
E448- BD 04 01        LDA $0104,X   ;get saved statusregister
E448- 29 10          AND #$10      ;filter BRK-bit out
E44D- F0 03          BEQ PULS1     ;if BRK
E44F- 6C 92 00        JMP (CBINV)   ;go do BRK routine
E452- 6C 90 00        JMP (CINV)     ;else do IRQ routine (default KEY)
;*****
;*                                     *
;* Default interrupt routine.       *
;*                                     *
;* (Pointed to by ($90)).         *
;*                                     *
;*****
E455- 4C 31 E4        KEY           JMP e431      ;do clock tick(s)
;*****
;*                                     *
;* Rest of interrupt routine;      *
;* do cursor blink                *
;*                                     *
;*****
E458- A5 A7          e458          LDA *BLNSW    ;if cursor invisible
E45A- D0 1E          BNE e47a      ;skip blink handling
E45C- C6 A8          DEC *BLNCT    ;decrement countdown
E45E- D0 1A          BNE e47a      ;if countdown zero
E460- A9 14          LDA #20      ;set new countdown
E462- 24 E4          BIT *RPTFLG   ;get repeat status
E464- 10 02          BPL e468      ;if REPEAT pressed
E466- A9 02          LDA #2      ;get faster countdown
E468- 85 A8          e468          STA *BLNCT
E46A- A4 C6          LDY *PNTR      ;get column position
E46C- 46 AA          LSR *BLNON    ;determine blink phase
;and set phase zero
E46E- B1 C4          LDA (PNT),Y   ;get character at current position
E470- B0 04          BCS KEYS      ;if in phase zero
E472- E6 AA          INC *BLNON    ;set blink phase one
E474- 85 A9          STA *GDBLN    ;and store true character
E476- 49 80          KEYS          EOR #$80      ;reverse screen character
E478- 91 C4          STA (PNT),Y   ;and put to screen
;*****
;*                                     *
;* Cassette motor control.        *
;*                                     *
;*****
E47A- AD 00          e47a          LDY #$00
E47C- AD 10 E8        LDA PIAL
E47F- 29 F0          AND #$F0      ;set zero in
E481- 80 10 E8        STA PIAL      ;keyboard scan line number
E484- AD 10 E8        LDA PIAL      ;get cassette sense lines

```

```

E487- 0A          ASL A
E488- 0A          ASL A
E489- 0A          ASL A
E48A- 10 09        BPL KEY3      ;if no key down on cassette 1
E48C- 84 F9        STY *CAS1     ;flag cassette 1 off
E48E- AD 13 E8     LDA PIAS
E491- 09 08        ORA #$08      ;and switch motor off
E493- 00 09        BNE KL24
E495- A5 F9        KEY3        LDA *CAS1    ;key down,if #1 flag on
E497- D0 08        BNE KL2
E499- AD 13 E8     LDA PIAS
E49C- 29 F7        AND #$F7      ;else
E49E- 8D 13 E8     STA PIAS    ;start cass#1 motor
E4A1- 90 09        KL2         BCC KL23   ;if no key down on cass.#2
E4A3- 84 FA        STY *CAS2     ;flag cass.#2 off
E4A5- AD 40 E8     LDA PIA
E4A8- 09 10        ORA #$10      ;and switch cass#2 motor off
E4AA- D0 09        BNE KL25
E4AC- A5 FA        KL23        LDA *CAS2    ;key down,if #2 flag on
E4AE- D0 08        BNE KL22
E4BD- AD 40 E8     LDA PIA
E4B3- 29 EF        AND #$EF      ;else
E4B5- 8D 40 E8     STA PIA
E4B8- 20 BE E4     JSR e4be    ;handle keyboard
E4BB- 4C D0 E6     JMP PREND   ;and return from interrupt
.FI "BASIC4.OE8B.M05"

```

OC49 33CD-4009 BASIC4.OE8B.M05

```

;name BASIC4.OE8B.M05
;*****
;*
;* Scan keyboard.
;*
;*****
E4BE- A0 FF        e4be       LDY #$FF      ;this index received
E4C0- 84 A6        STY *SFDX     ;on no key at all
E4C2- C8
E4C3- 84 98        INY
E4C5- A5 E4        STY *SHFLAG   ;clear shift key indicator
E4C7- 29 7F        LDA *RPTFLG  ;get repeat status
E4C9- 85 E4        AND #$7F      ;remove REPEAT key bit
E4CB- A2 50        STA *RPTFLG
E4CD- A0 08        e4cd       LDX #80      ;this number of keys to be scanned
E4CF- AD 12 E8     LDY #8       ;number of bits
E4D2- CD 12 E8     LDA PIAK    ;get key image
E4D5- D0 F6        CMP PIAK    ;wait
E4D7- 4A           BNE e4cd   ;until steady
E4D8- B0 1F        KL1        LSR A      ;get bit in carry
E4DA- 48           BCS CKIT
E4DB- B0 D0 E6     PHA
E4DE- D0 06        LDA CHAR-1,X ;save key image
E4E0- A9 01        BNE e4e6   ;get value for that key
E4E1- 01
E4E2- 85 98        LDA #1
E4E4- D0 12        BNE CKUT    ;if shift key
E4E6- C9 10        e4e6       CMP #16      ;set flag for shift key
E4E8- D0 08        BNE CKIS1
E4EA- A5 E4        LDA *RPTFLG ;if REPEAT key
E4EC- 09 80        ORA #$80    ;get repeat status
E4EE- 85 E4        STA *RPTFLG ;add REPEAT key bit
E4F0- 30 06        BMI CKUT    ;and check for further keys
E4F2- C9 FF        CKIS1     CMP #$FF    ;if no key
E4F4- F0 02        BEQ CKUT    ;check for next key

```

E4F6- 85 A6	SPCK	STA *SFDX	;else store key value
E4F8- 68	CKUT	PLA	;reget key image
E4F9- CA	CKIT	DEX	;decrement key counter
E4FA- F0 08		BEQ CKIT1	;if not at end
E4FC- 88		DEY	;decrement bit counter
E4FD- D0 D8		BNE KL1	;and check next bits
E4FF- EE 10 E8		INC PIAL	;do next scan line
E502- D0 C9		BNE e4cd	
E504- A5 A6	CKIT1	LDA *SFDX	;get key found
E506- C5 97		CMP *LSTX	;if same as last time
E508- F0 06		BEQ e510	;do repeat
E50A- A2 10		LDX #16	;get countdown for repeat
E50C- 86 E6		STX *DELAY	
E50E- D0 2F		BNE e53f	;and process key
E510- 24 E4	e510	BIT *RPTFLG	;get repeat status
E512- 30 1E		BMI e532	;if repeat pressed, skip test
E514- 70 59		BVS e56f	;if repeat off, exit
E516- C9 FF		CMP #\$FF	;if no key
E518- F0 55		BEQ e56f	;exit
E51A- C9 14		CMP #\$14	;else if INST/DEL
E51C- F0 0C		BEQ e52a	;do repeat
E51E- C9 20		CMP #\$20	;if SPACE
E520- F0 08		BEQ e52a	;do repeat
E522- C9 1D		CMP #\$1D	;if LEFT/RIGHT
E524- F0 04		BEQ e52a	;do repeat
E526- C9 11		CMP #\$11	;if not UP/DOWN
E528- D0 45		BNE e56f	;exit
E52A- A6 E6	e52a	LDX *DELAY	;get countdown
E52C- F0 04		BEQ e532	;if expired, do delay
E52E- C6 E6		DEC *DELAY	;else count down
E530- D0 30		BNE e56f	;if not expired, exit
E532- C6 E5	e532	DEC *KOUNT	;else decrement delay
E534- D0 39		BNE e56f	;if not expired, exit
E536- A2 04		LDX #4	;else
E538- 86 E5		STX *KOUNT	;store new flash countdown
E53A- A6 9E		LDX *NDX	;get # of keys in buffer
E53C- CA		DEX	;if not empty
E53D- 10 30		BPL e56f	;exit
E53F- 85 97	e53f	STA *LSTX	;store new key value
E541- C9 FF		CMP #\$FF	;if no key
E543- F0 2A		BEQ e56f	;exit
E545- AA		TAX	;save key in X
E546- 08		PHP	;save flags
E547- 29 7F		AND #\$7F	;remove shift bit
E549- 28		PLP	;get flags again
E54A- 30 17		BMI KN1	;if msb was set, ignore msb
E54C- 46 98		LSR *SHFLAG	;get and clear shift status
E54E- 90 13		BCC KN1	;if pressed and
E550- C9 2C		CMP #','	;if lower than comma
			;concerns all ctl. chars. and space)
E552- 90 00		BCC e561	;add shiftbit
E554- C9 3C		CMP #'<'	;or higher than ;
			;concerns all letters)
E556- B0 09		BCS e561	;add shiftbit as well
E558- E9 0F		SBC #\$0F	;else convert char. to shifted value
			;BUG: wrong result if IRQ entered
			;with decimal flag set.
			;concerns ,.-./ numbers ::)
E55A- C9 20		CMP #\$20	;if now higher than space
E55C- B0 05		BCS KN1	;put in buffer
- E55E- 69 20		ADC #\$20	;else convert to shift value
			;BUG: wrong result if IRQ entered
			;with decimal flag set.
E560- 2C		.BY \$2C	;skip next instruction

```

E561- 09 80      e561    ORA #$80          ;add shiftbit
E563- A6 9E      KN1     LDX *NDX           ;get # of keys in buffer
E565- E4 E3      CPX *XMAX          ;if more than max. buffer length
E567- B0 06      BCS e56f          ;exit (ignore key)
                                ;BUG: exits on equality AND if more,
                                ;      so default 9, accepts 9 keys maximum,
                                ;      not ten.
E569- 9D 6F 02      STA KEYD,X       ;else store in buffer
E56C- E8          INX
E56D- 86 9E      STX *NDX           ;and adapt buffer counter
E56F- 60          RTS
                                .FI "BASIC4.OE8B.M06"

```

1A86 33CD-4E46 BASIC4.OE8B.M06

```

;name BASIC4.OE8B.M06
;*****
;*
;* Get bitmask for TAB table in *
;* $033E and offset in table in *
;* X. *
;*
;*****
E570- A5 C6      e570    LDA *PNTR          ;get column #
E572- 29 F8      AND #$F8          ;remove 3 lowest bits
E574- 8D 3A 03      STA $033A          ;save it
E577- 4A          LSR A
E578- 4A          LSR A
E579- 4A          LSR A          ;divide it by 8
E57A- AA          TAX             ;and move to X
E57B- A9 01      LDA #1           ;load bit mask
E57D- 8D 3E 03      STA $033E          ;*****
E580- A4 C6      LDY *PNTR          ;get column # again
E582- CC 3A 03      e582    CPY $033A          ;if position found
E585- F0 09      BEQ e590          ;exit
E587- 0E 3E 03      ASL $033E          ;adapt bit mask
E58A- EE 3A 03      INC $033A          ;and count bits
E58D- 4C 82 ES      JMP e582          ;until position found
E590- 60          RTS
                                ;*****
;*
;* Do rest of unshifted *
;* characters. *
;*
;*****
E591- C9 19      e591    CMP #$19          ;if SCROLL UP
E593- D0 06      BNE e59b          ;*****
E595- 20 E8 E3      JSR SCROL          ;scroll window up
E598- 4C D9 E5      JMP e5d9          ;get pointer to line and exit
E59B- C9 0F      e59b    CMP #$0F          ;if SET TOP
E59D- D0 0B      BNE e5aa          ;*****
E59F- A5 D8      LDA *TBLX          ;get current line#
ESA1- 85 EO      STA *XREC          ;store in top line#
ESA3- A5 C6      LDA *PNTR          ;get current column#
ESA5- 85 E2      STA *$E2          ;store in left margin
ESA7- 4C 99 E1      e5a7    JMP LOOP2          ;and exit
ESA8- C9 0E      e5aa    CMP #$0E          ;if TEXT
ESAC- D0 05      BNE e5b3          ;*****
ESA9- 20 7A EO      JSR e07a          ;set CRT controller to text
ESB1- 30 F4      BMI e5a7          ;and exit
ESB3- C9 07      e5b3    CMP #$07          ;if not BELL
ESB5- D0 F0      BNE e5a7          ;exit (ignore character)
ESB7- 20 A7 E6      JSR e6a7          ;if BELL, ring it

```

ESBA- F0 EB		BEQ e5a7 ;and exit ***** ;* ;* Do rest of shifted ;* characters. ;* *****
ESBC- C9 15	e5bc	CMP #\$15 ;if INSERT LINE
ESBE- D0 12		BNE e5d2
ESCO- A5 E0		LDA *XREC ;get top line#
ESC2- 48		PHA ;save it
ESC3- A5 D8		LDA *TBLX ;get current line#
ESC5- 85 E0		STA *XREC ;put in top line#
ESC7- 20 C8 E3		JSR NEWLN ;scroll 'window' down and insert line
ESCA- 68	e5ca	PLA
ESCB- 85 E0		STA *XREC ;restore original top line#
ESCD- 20 63 E0		JSR e063 ;set cursor at start and get
ESD0- D0 18		BNE e5ea ;and exit
ESD2- C9 19	e5d2	CMP #\$19 ;if SCROLL DOWN
ESD4- D0 08		BNE e5de
ESD6- 20 C8 E3		JSR NEWLN ;scroll window down
ESD9- 20 67 E0	e5d9	JSR STUPT ;get pointer to line
ESDC- D0 0C		BNE e5ea ;and exit
ESDE- C9 0F	e5de	CMP #\$0F ;if SET BOTTOM
ESE0- D0 0B		BNE e5ed
ESE2- A5 D8		LDA *TBLX ;get current line#
ESE4- 85 E1		STA *XWRT ;store in bottom line#
ESE6- A5 C6		LDA *PNTR ;get current column#
ESE8- 85 D5		STA *LNMX ;store in right margin
ESEA- 4C 99 E1	e5ea	JMP LOOP2 ;and exit
ESED- C9 0E	e5ed	CMP #\$0E ;if not GRAPHICs
ESEF- D0 C2		BNE e5b3 ;check for BELL
ESF1- 20 82 E0		JSR e082 ;else set to graphics
ESF4- 30 F4		BMI e5ea ;and exit ***** ;* ;* The area \$E5F6-\$E5FF is filled * ;* with \$AA bytes. ;* *****
		.BA \$E600 ***** ;* ;* Exit from interrupt. ;* *****
E600- 68	PREND	PLA
E601- A8		TAY ;restore Y
E602- 68		PLA
E603- AA		TAX ;and X
E604- 68		PLA ;and A
E605- 40		RTI ;and return from interrupt ***** ;* ;* Store the character in A * ;* on screen. * ;* *****
E606- A4 C6	DSPP	LDY *PNTR ;get cursor position on line
E608- 91 C4		STA (PNT),Y ;store the character
E60A- A9 D2		LDA #2 ;and set up new countdown
E60C- 85 A8		STA *BLNCT ;for cursor flash
E60E- 60		RTS ;then exit *****

```

        ;* Reset; initialize I/O.          *
        ;*                                     *
        ;*****                                 *
E60F- A9 7F      e60f    LDA #$7F
E611- 8D 4E E8      STA IER           ;disable all VIA interrupts
E614- A2 6D      LDX #$6D
E616- A9 00      LDA #$00           ;clear
E618- 85 E8      STA *$E8
E61A- 85 E4      STA *RPTFLG        ;enable repeat of cursor keys (also in
E61C- 95 80      e61c    STA *CTIMR,X   ;area $8D to $FA
E61E- CA          DEX
E61F- 10 FB      BPL e61c
E621- A9 55          LDA #L,KEY      ;set
E623- 85 90          STA *CINV        ;IRQ RAM vector
E625- A9 E4          LDA #H,KEY      ;to
E627- 85 91          STA *CINV+1     ;keyboard cassette routine
E629- A9 09          LDA #9
                                ;BUG: must be 10 for bufferlength of 10
                                ;get max. keyboard buffer length
E62B- 85 E3          STA *XMAX        ;store it
E62D- A9 03          LDA #3
                                ;set default output device
E62F- 85 B0          STA *DFLTO       ;to screen
E631- A9 0F          LDA #$0F
                                ;keyboard PIA, A section:
E633- 8D 10 E8      STA PIAL         ;bits 0-3 output, rest input
E636- 0A          ASL A
E637- 8D 40 E8      STA PIA          ;cassette#2 motor off
                                ;cassette #1 and #2 write = 1
                                ;IEEE ATN=false NRFD=false
E63A- 8D 42 E8      STA P2DB         ;bits 7-5,0 output, 4-1 input
E63D- 8E 22 E8      STX IEEE0        ;IEEE PIA, B section: all bits output
E640- 8E 45 E8      STX T1H          ;VIAT1H: maximum timeout
E643- A9 30          LDA #$3D
                                ;keyboard PIA B:cassette#1 motor off
E645- 8D 13 E8      STA PIAS         ;IRQ on neg edge on CB1
                                ;(end of screen refresh)
E648- 2C 12 E8      BIT PIAK        ;clear keyboard PIA interrupt flags
E64B- A9 3C          LDA #$3C
                                ;IEEE PIA: NDAC=false, CA1 neg. edge
E64D- 8D 21 E8      STA IEEIS        ;no interrupts
E650- 8D 23 E8      STA IEEOS        ;IEEE PIA: DAV=false, CB1 neg. edge
                                ;no interrupts
E653- 8D 11 E8      STA PIAL1        ;keyboard PIA: CA2=1,CA1 neg. edge
                                ;no interrupts
E656- 8E 22 E8      STX IEEE0        ;IEEE PIAB:all data lines false
E659- A9 0E          LDA #$0E
E65B- 85 A8          STA *BLNCT        ;VIA:CA2=1 (lower case),CA1neg. edge
E65D- 85 A7          STA *BLNSW        ;set blink count for cursor
E65F- 85 E6          STA *DELAY        ;set cursor visible
E661- 85 E5          STA *KOUNT        ;set countdown
E663- 8D 4E E8      STA IER
E666- 20 D2 E1      JSR e1d2        ;and delay for repeat
E669- A2 0C          LDX #12
E66B- A9 00          LDA #0
E66D- 9D EE 03      e66d    STA $03EE,X   ;VIA: no interrupts,CB2 input,neg. edg
                                ;clear TABstop table
E670- CA          DEX
E671- 10 FA          BPL e66d
E673- A9 10          LDA #L,e11d
E675- A2 E1          LDX #H,e11d
E677- 85 E9          STA *$E9
E679- 86 EA          STX *$EA
E67B- A9 0C          LDA #L,e20c
E67D- A2 E2          LDX #H,e20c
E67F- 85 EB          STA *$EB
E681- 86 EC          STX *$EC
E683- A9 10          LDA #$10
E685- 85 E7          STA *$E7
                                ;set vector for
                                ;input from screen
                                ;set vector for
                                ;output to screen
                                ;set up
                                ;bell timing

```

E687- 20 A4 E6		JSR e6a4
E68A- F0 18		BEQ e6a4 ;and ring bell 4 times
		;*****
		;* *
		;* Ring bell if right margin *
		;* passed. *
		;* *
		;*****
E68C- 20 02 E2	e68c	JSR PRT ;move character to screen
E68F- AA		TAX ;save it in X
E690- A5 D5		LDA *LNMX ;get right margin
E692- 38		SEC
E693- E5 C6		SBC *PNTR ;subtract current position
E695- C9 05		CMP #5 ;if 5 from margin
E697- D0 37		BNE e6d0
E699- 8A		TXA ;get character again
E69A- C9 1D		CMP #\$1D ;if cursor-RIGHT
E69C- F0 06		BEQ e6a4 ;ring bell twice
E69E- 29 7F		AND #\$7F ;remove shift-bit
E6A0- C9 20		CMP #\$20 ;if not printable
E6A2- 90 2C		BCC e6d0 ;exit, else ring bell twice
		;*****
		;* *
		;* Ring bell twice. *
		;* *
		;*****
E6A4- 20 A7 E6	e6a4	JSR e6a7 ;ring bell once
		;*****
		;* *
		;* Ring bell once. *
		;* *
		;*****
E6A7- A4 E7	e6a7	LDY #\$E7 ;get bell timing
E6A9- F0 25		BEQ e6d0 ;if zero, don't ring bell
E6AB- A9 10		LDA #\$10 ;set CB2 to free run
E6AD- 8D 48 E8		STA ACR
E6B0- A9 0F		LDA #\$0F ;and to lowest frequency
E6B2- 8D 4A E8		STA SR
E6B5- A2 07		LDX #7 ;7 tones to play
E6B7- BD 40 E7	e6b7	LDA e74e-1,X ;get tone constant
E6BA- 8D 48 E8		STA T2L ;put to VIA
E6BD- A5 E7		LDA *\$E7 ;get timing
E6BF- 88	e6bf	DEY
E6C0- D0 FD		BNE e6bf ;wait, inner loop
E6C2- 38		SEC
E6C3- E9 01		SBC #1
E6C5- D0 F8		BNE e6bf ;wait, outer loop
E6C7- CA		DEX
E6C8- D0 ED		BNE e6b7 ;and do next tone
E6CA- 8E 4A E8		STX SR ;then silence the VIA
E6CD- 8E 4B E8		STX ACR ;and enable cassette operations
E6D0- 60	e6d0	RTS ;then exit
		.FI "BASIC4.OE8B.M07"

1AF3 33C0-4EB3 BASIC4.OE8B.M07

```
;name BASIC4.OE8B.M07
;*****
;* *
;* Keyboard table, for 80 keys. *
;* *
;* Keys with MSB set have no *
;* shifted equivalent, e.g. all
```

```

;* the numerals. *
;*
;*****  

E6D1- 16 04 3A CHAR .BY $16 $04 ':' '$03 '963' $0F ;scan line nine
E6D4- 03 39 36
E6D7- 33 DF
E6D9- B1 2F 15 .BY $B1 '' $15 $13 'M X' $12 ;scan line eight
E6DC- 13 40 20
E6DF- 58 12
E6E1- B2 10 0F .BY $B2 $10 $0F $80 ',NVZ' ;scan line seven
E6E4- 80 2C 4E
E6E7- 56 5A
E6E9- B3 00 19 .BY $B3 $00 $19 $AE '.BC' $00 ;scan line six
E6EC- AE 2E 42
E6EF- 43 00
E6F1- B4 DB 4F .BY $B4 $DB '0' $11 'UTEQ' ;scan line five
E6F4- 11 55 54
E6F7- 45 51
E6F9- 14 50 49 .BY $14 'PI' $DC 'YRW' $09 ;scan line four
E6FC- DC 59 52
E6FF- 57 09
E701- B6 C0 4C .BY $B6 $C0 'L' $00 'JGDA' ;scan line three
E704- 00 4A 47
E707- 44 41
E709- B5 3B 4B .BY $B5 ';K' $00 'HFS' $9B ;scan line two
E70C- 00 48 46
E70F- 53 9B
E711- B9 06 DE .BY $B9 $06 $DE $B7 $80 '741' ;scan line one
E714- B7 B0 37
E717- 34 31
E719- 05 0E 1D .BY $05 $0E $1D $B8 '-852' ;scan line zero
E71C- B8 2D 38
E71F- 35 32
;*****  

;* Message: D shift-L "* RETURN *
;* RUN RETURN *
;* *
;*****  

E721- 44 CC 22 RUNTB .BY 'D' $CC '"*' $00 'RUN' $00
E724- 2A 0D 52
E727- 55 4E 0D
;*****  

;* Two tables of 18 constants *
;* each for the CRT-controller. *
;* The first table is for text *
;* mode, the second for graphics *
;* mode. *
;* *
;*****  

;text mode table
E72A- 31 e72a .BY 49 ;horizontal total
E72B- 28 .BY 40 ;horizontal displayed
E72C- 29 .BY 41 ;horizontal sync position
E72D- 0F .BY 15 ;horizontal sync width
E72E- 27 .BY 39 ;vertical total
E72F- 00 .BY 0 ;vertical total adjust
E730- 19 .BY 25 ;vertical displayed
E731- 20 .BY 32 ;vertical sync position
E732- 00 .BY 0 ;interlace mode (off)
E733- 09 .BY 9 ;maximum scan line address
E734- 00 .BY 0 ;cursor start
E735- 00 .BY 0 ;cursor end

```

```

E736- 10      .BY 16          ;start address (hi) (inverts video)
E737- 00      .BY 0           ;start address (lo)
E738- 00      .BY 0           ;cursor (hi)
E739- 00      .BY 0           ;cursor (lo)
E73A- 00      .BY 0           ;light pen (hi)
E73B- 00      .BY 0           ;light pen (lo)
                           ;graphics mode table
E73C- 31      e73c         .BY 49          ;horizontal total
E73D- 28      .BY 40          ;horizontal displayed
E73E- 29      .BY 41          ;horizontal sync position
E73F- 0F      .BY 15          ;horizontal sync width
E740- 31      .BY 49          ;vertical total
E741- 00      .BY 0           ;vertical total adjust
E742- 19      .BY 25          ;vertical displayed
E743- 25      .BY 37          ;vertical sync position
E744- 00      .BY 0           ;interlace mode (off)
E745- 07      .BY 7           ;maximum scan line address
E746- 00      .BY 0           ;cursor start
E747- 00      .BY 0           ;cursor end
E748- 10      .BY 16          ;start address (hi) (inverts video)
E749- 00      .BY 0           ;start address (lo)
E74A- 00      .BY 0           ;cursor (hi)
E74B- 00      .BY 0           ;cursor (lo)
E74C- 00      .BY 0           ;light pen (hi)
E74D- 00      .BY 0           ;light pen (lo)
                           ;*****
                           ;*
                           ;* Constants for bell. *
                           ;*
                           ;*****
E74E- 0E 1E 3E  e74e     .BY $0E $1E $3E $7E $3E $1E $0E
E751- 7E 3E 1E
E754- 0E
                           ;*****
                           ;*
                           ;* Table of lo-bytes of screen *
                           ;* line addresses. *
                           ;*
                           ;*****
E755- 00 50 A0  LDTB2   .BY 0 80 160 240 64 144 224
E758- F0 40 90
E75B- E0
E75C- 30 80 D0      .BY 48 128 208 32 112 192
E75F- 20 70 C0
E762- 10 60 B0      .BY 16 96 176 0 80 160 240
E765- 00 50 A0
E768- F0
E769- 40 90 E0      .BY 64 144 224 48 128
E76C- 30 80
                           ;*****
                           ;*
                           ;* Table of hi-bytes of screen *
                           ;* line addresses. *
                           ;*
                           ;*****
E76E- 80 80 80  LDTB1   .BY $80 $80 $80 $80
E771- 80
E772- 81 81 81      .BY $81 $81 $81 $82 $82 $82
E775- 82 82 82
E778- 83 83 83      .BY $83 $83 $83 $84 $84 $84
E77B- 84 84 84
E77E- 85 85 85      .BY $85 $85 $85 $85
E781- 85
E782- 86 86 86      .BY $86 $86 $86 $87 $87

```

E785- 87 87

E787- BF

```

;*****
;*                                     *
;* Checksum byte (over E-ROM).      *
;*                                     *
;*****                                     *
CKSUME .BY $BF           ;checksum byte
;*****
;*                                     *
;* The rest of the ROM is filled   *
;* with $AA bytes.                 *
;*                                     *
;*****                                     *
;
;*****
;*                                     *
;* External Labels.                  *
;*                                     *
;*****                                     *
CTIMR .DE $008D          ;the jiffy clock
CINV .DE $0090           ;IRQ ram vector
CBINV .DE $0092           ;BRK RAM vector
LSTX .DE $0097           ;last scan index
SHFLAG .DE $0098          ;shiftkey indicator
NDX .DE $009E           ;number of keys in keyboard buffer
RVS .DE $009F           ;flag for reverse field printing
INDX .DE $00A1           ;record length of current screen line
LXSP .DE $00A3           ;cursor row/column, used in INPUT and G
SFDX .DE $00A6           ;current scan index
BLNSW .DE $00A7           ;flag: cursor on
BLNCT .DE $00A8           ;countdown till next cursor blink
GDBLN .DE $00A9           ;real character under cursor
BLNON .DE $00AA           ;cursor's blink phase
CRSW .DE $00AC           ;flag: input from screen or keyboard
DFLTN .DE $00AF           ;current input device number
DFLTO .DE $00B0           ;current output device number
PNT .DE $00C4           ;screen line address, lo
POINT .DE $00C5           ;screen line address, hi
PNTR .DE $00C6           ;current column position of cursor
SAL .DE $00C7           ;pointer in screen scroll
SAH .DE $00C8           ;pointer in screen scroll, hi
QTSW .DE $00CD           ;flag: nonzero if in quote mode
FNLEN .DE $00D1           ;temp. work area for CRTC setup
LNMX .DE $0005           ;length of current screen line
TBLX .DE $0008           ;current line number of cursor
DATA .DE $0009           ;last key input
INSRT .DE $000C           ;number of inserts outstanding
XREC .DE $00E0           ;top line# of window
XWRT .DE $00E1           ;bottom line# of window
XMAX .DE $00E3           ;maximum keyboard buffer length
RPTFLG .DE $00E4           ;flag: REPEAT pressed
KOUNT .DE $00E5           ;repeat countdown
DELAY .DE $00E6           ;repeat rate for repeat key
CAS1 .DE $00F9           ;flag: cassette #1 motor on
CAS2 .DE $00FA           ;flag: cassette #2 motor on
PI .DE $00FF           ;value for PI
KEYD .DE $026F           ;keyboard buffer
UDTIM .DE $FFEA           ;update jiffy clock
PIAL .DE $E810           ;keyboard PIA: I/O port A and DDR
PIAL1 .DE $E811           ;keyboard PIA: control register A
PIAK .DE $E812           ;keyboard PIA: I/O port B and DDR
PIAS .DE $E813           ;keyboard PIA: control register B
IEEEI .DE $E820           ;IEEE PIA: I/O port A and DDR
IEEEIS .DE $E821           ;IEEE PIA: control register A

```

IEEO	.DE \$E822	;IEEE PIA: I/O port B and DDR
IEEOS	.DE \$E823	;IEEE PIA: control register B
PIA	.DE \$E840	;VIA: I/O port B
SYNC	.DE \$E841	;VIA: I/O port A with handshake
P2DB	.DE \$E842	;VIA: port B data direction register
P2DA	.DE \$E843	;VIA: port A data direction register
T1L	.DE \$E844	;VIA: timer 1 lo
T1H	.DE \$E845	;VIA: timer 1 hi
T1LL	.DE \$E846	;VIA: timer 1 lo, latch
T1LH	.DE \$E847	;VIA: timer 1 hi, latch
T2L	.DE \$E848	;VIA: timer 2 lo
T2H	.DE \$E849	;VIA: timer 2 hi
SR	.DE \$E84A	;VIA: shift register
ACR	.DE \$E84B	;VIA: auxiliary control register
PCR	.DE \$E84C	;VIA: peripheral control register
IFR	.DE \$E84D	;VIA: interrupt flag register
IER	.DE \$E84E	;VIA: interrupt enable register
SYNC1	.DE \$E84F	;VIA: port A without handshake
CRO	.DE \$E880	;CRTC: address register
CR1	.DE \$E881	;CRTC: register selected through CRO
	.en	
	.EN	

END MAE PASS

LABELFILE

ACR =E84B	BK1 =E25·1	BK2 =E25C
BKLN =E1AA	BLNCT =00A8	BLNON =00AA
BLNSW =00A7	CAS1 =00F9	CAS2 =00FA
CBINV =0092	CHAR =E6D1	CINT =E000
CINV =0090	CKIS1 =E4F2	CKIT =E4F9
CKIT1 =E504	CKSUME =E787	CKUT =E4F8
CLP1 =E15B	CLP2 =E144	CLP21 =E159
CLP2A =E156	CLP5 =EOF2	CLP6 =EOFB
CLP7 =E169	CLSR =E051	CNC3 =E266
CNC3X =E23E	CRO =E880	CR1 =E88·1
CRSW =00AC	CTIMR =008D	DATA =00D9
DELAY =00E6	DFLTN =00AF	DFLTO =0080
DSPP =E606	FNLEN =0001	GDBLN =00A9
IEEI =E820	IEEIS =E82·1	IEEO =E822
IEEOS =E823	IER =E84E	IFR =E84D
INDX =00A1	INS1 =E320	INS2 =E322
INSRT =00DC	JLP2 =E38C	JPL3 =E299
JSTS =E188	JSTS1 =E192	KEY =E455
KEY3 =E495	KEY5 =E476	KEYD =026F
KL1 =E4D7	KL2 =E4A1	KL22 =E4B8
KL23 =E4AC	KL24 =E49E	KL25 =E4B5
KN1 =E563	KOUNT =00E5	LDTB1 =E76E
LDTB2 =E755	LNMX =0005	LOOP2 =E199
LOOP3 =EOBF	LOOP4 =EOBC	LOOP5 =E1·6
LOP2 =E1A5	LOP5 =E121	LOP51 =E127
LOP52 =E137	LOP53 =E13B	LOP54 =E131
LP1 =EOAC	LP2 =EOA7	LP21 =E0D3
LP22 =EOEA	LP23 =EOOF	LSTX =0097
LXSP =00A3	MLP1 =E3FD	MLP4 =E408
MLP41 =E414	MLP42 =E41B	NC1 =E26F
NC2 =E283	NC3 =E17D	NC3W =E269
NCX2 =E292	NDX =009E	NELL =E3DD
NEWLN =E3C8	NJTL =E22B	NTCN =E237
NTCN1 =E262	NVS =E17F	NVSL =E185
NXLN =E3A3	NXT2 =E358	NXT3 =E179

NXT33 =E177	NXT6 =E360	NXTD =E05F
NXTL =E3B0	NXTL1 =E3CB	NXTX =E2F4
NXTX1 =E2FC	P2DA =E843	P2DB =E842
PCR =E84C	PI =0OFF	PIA =E840
PIAK =E812	PIAL =E810	PIAL1 =E811
PIAS =E813	PNT =00C4	PNTR =00C6
POINT =00C5	PREND =E600	PRENDO =E012
PRT =E202	PULS =E442	PULS1 =E452
QTSW =00CD	QTSWC =E16A	QTSWL =E176
RPTFLG =00E4	RUNTB =E721	RVS =009F
SAH =00C8	SAL =00C7	SCRL5 =E3EB
SCROL =E3E8	SFDX =00A6	SHFLAG =0098
SPCK =E4F6	SR =E84A	STUPR =E06C
STUPT =E067	SYNC =E841	SYNC1 =E84F
T1H =E845	T1L =E844	T1LH =E847
T1LL =E846	T2H =E849	T2L =E848
TBLX =0008	UDTIM =FFEA	UP2 =E347
UPS =E30A	UP6 =E342	UP9 =E33E
XMAX =00E3	XREC =00E0	XWRT =00E1
e036 =E036	e04b =E04B	e054 =E054
e063 =E063	e06f =E06F	e07a =E07A
e082 =E082	e088 =E088	e09b =E09B
e11d =E11D	e19d =E19D	e1ba =E1BA
e1c1 =E1C1	e1c3 =E1C3	e1cb =E1CB
e1d2 =E1D2	e1dc =E1DC	e1e1 =E1E1
e20c =E20C	e21d =E21D	e224 =E224
e24d =E24D	e27b =E27B	e29c =E29C
e2a3 =E2A3	e2b6 =E2B6	e2c5 =E2C5
e2d0 =E2D0	e2d7 =E2D7	e2e0 =E2E0
e2e7 =E2E7	e303 =E303	e36f =E36F
e373 =E373	e37c =E37C	e38f =E38F
e396 =E396	e39a =E39A	e3b1 =E3B1
e3b6 =E3B6	e41f =E41F	e420 =E420
e424 =E424	e431 =E431	e458 =E458
e468 =E468	e47a =E47A	e4be =E4BE
e4cd =E4CD	e4e6 =E4E6	e510 =E510
e52a =E52A	e532 =E532	e53f =E53F
e561 =E561	e56f =E56F	e570 =E570
e582 =E582	e590 =E590	e591 =E591
e59b =E59B	e5a7 =E5A7	e5aa =E5AA
e5b3 =E5B3	e5bc =E5BC	e5ca =E5CA
e5d2 =E5D2	e5d9 =E5D9	e5de =E5DE
e5ea =E5EA	e5ed =E5ED	e60f =E60F
e61c =E61C	e66d =E66D	e68c =E68C
e6a4 =E6A4	e6a7 =E6A7	e6b7 =E6B7
e6bf =E6BF	e6d0 =E6D0	e72a =E72A
e73c =E73C	e74e =E74E	

//0000,E788,E788

]

```

.LS
;name BASIC4.OEG8.CTL
;*****
;*          *
;*      E-ROM for BASIC 4.0      *
;*          *
;*      for 80 column 12 inch    *
;*      screen models with      *
;*          N-keyboard          *
;*          *
;* (nicknamed 'Graphics 80')   *
;*          *
;* (FAT 40 converted to 80     *
;*      columns per line)       *
;*          *
;* A proposed standard ROM    *
;*          *
;* Date 27-09-83 Time 01:30   *
;*          *
;* Made by:                   *
;*          *
;* Nico de Vries              *
;* Mari Andriessenrade 49    *
;* 2907 MA Capelle a/d Yssel *
;* The Netherlands            *
;*          *
;* Original CBM ROM-number   *
;*          *
;*          none!               *
;*          *
;*****.BA $E000
.FI "BASIC4.OEG8.M01"

```

1950 34A7-4DF7 BASIC4.OEG8.M01

```

;name BASIC4.OEG8.M01
;*****
;*          *
;* Jump menu, enables standardi- *
;* sation between various machine *
;* types.                         *
;*          *
;*****.E000- 4C 4B E0  CINT    JMP e04b      ;Initialize I/O, cursor home and 4 bell
.E003- 4C A7 E0    JMP LP2       ;Get character from keyboard buffer
.E006- 4C 16 E1    JMP LOOPS    ;Get character from device 0 or 3
.E009- 4C 02 E2    JMP PRT      ;Put character to screen
.E00C- 4C 42 E4    JMP PULS     ;Service interrupt
.E00F- 4C 55 E4    JMP KEY      ;Default interrupt routine
.E012- 4C 00 E6  PREEND  JMP PREND    ;Exit from interrupt
.E015- 4C 51 E0    JMP CLSR     ;Clear screen within window
.E018- 4C 7A E0    JMP e07a     ;Set to lower/upper case
.E01B- 4C 82 E0    JMP e082     ;Set to upper case/graphics
.E01E- 4C 88 E0    JMP e088     ;Other CRT controller settings
.E021- 4C C8 E3    JMP NEWLN    ;Scroll screen down
.E024- 4C E8 E3    JMP SCROL    ;Scroll screen up
.E027- 4C BE E4    JMP e4be     ;Find key from decoding table
.E02A- 4C A7 E6    JMP e6a7     ;Ring bell once
.E02D- 4C 36 E0    JMP e036     ;Store accu in repeat flag
.E030- 4C E1 E1    JMP e1e1     ;Set top left of window

```

```

E033- 4C DC E1      JMP e1dc          ;Set right bottom of window
;*****
;*                                         *
;* Store A in repeatflag (not      *
;* used).                           *
;*                                         *
;*****
E036- 85 E4      e036   STA *RPTFLG    ;store A in repeat flag
E038- 60           RTS
E039- AA           TAX
E03A- AA           TAX
E03B- AA           TAX
E03C- AA           TAX
E03D- AA           TAX
E03E- AA           TAX
E03F- AA           TAX
E040- AA           TAX
E041- AA           TAX
E042- AA           TAX
E043- AA           TAX
E044- AA           TAX
E045- AA           TAX
E046- AA           TAX
E047- AA           TAX
E048- AA           TAX
E049- AA           TAX
E04A- AA           TAX
;*****
;*                                         *
;* Reset routine.                   *
;*                                         *
;*****
E04B- 20 0F E6      e04b   JSR e60f        ;Initialize I/O and zero page
E04E- 20 82 E0      JSR e082        ;set CRT controller to upper case graph
;*****
;*                                         *
;* Initialize screen; clear       *
;* screen within window.         *
;*                                         *
;*****
E051- A6 E0      CLSR    LDX *XREC      ;get top line # of window
E053- CA           DEX
E054- E8      e054   INX             ;do next line
E055- 20 6C E0      JSR STUPR       ;get pointer to line
E058- 20 C1 E1      JSR e1c1        ;erase that line till end of window
E05B- E4 E1           CPX *XWRT      ;until bottom of window done
E05D- 90 F5           BCC e054
;*****
;*                                         *
;* Home cursor (within window).   *
;*                                         *
;*****
E05F- A6 E0      NXTD    LDX *XREC      ;get top line # of window
E061- 86 D8           STX *TBLX      ;set cursor on that line
E063- A4 E2      e063   LDY *$E2        ;get left margin of window
E065- 84 C6           STY *PNTR      ;set cursor there
E067- A6 D8           STUPT        ;get line # again
E069- 4C 6F E0           JMP e06f        ;and get pointer to line
E06C- A4 E2           STUPR        ;LDY *$E2        ;get left margin of window
E06E- 88           DEY
;*****
;*                                         *
;* Get pointer to current screen  *
;* line.                            *
;*****

```

```

        ;*
        ;*****
E06F- BD 55 E7    e06f    LDA LDTB2,X      ;get lo-byte of pntr
E072- 85 C4          STA *PNT      ;store in pointer
E074- B0 6E E7          LDA LDTB1,X      ;get hi-byte
E077- 85 C5          STA *PNT+1     ;and complete pointer
E079- 60            RTS

        ;*****
        ;*
        ;* Set CRT controller to text      *
        ;* mode.                         *
        ;*
        ;*****
E07A- A9 2A    e07a    LDA #L,e72a    ;get lo-byte of pointer
E07C- A2 E7          LDX #H,e72a    ;and hi-byte also
E07E- A0 0E          LDY #$0E      ;and get lo-nibble for VIA
E080- D0 06          BNE e088      ;and set up CRT controller
        ;*****
        ;*
        ;* Set CRT controller to graphics *
        ;* mode.                         *
        ;*
        ;*****
E082- A9 3C    e082    LDA #L,e73c    ;get lo-byte of pointer
E084- A2 E7          LDX #H,e73c    ;and hi-byte also
E086- A0 0C          LDY #$0C      ;and get lo-nibble for VIA
E088- 85 C7    e088    STA *SAL      ;store pointer
E08A- 86 C8          STX *SAH      ;get VIA register
E08C- AD 4C E8          LDA PCR       ;clear lo-nibble
E08F- 29 F0          AND #$FO      ;and save it
E091- 85 D1          STA *FNLEN    ;add to register value
E093- 98            TYA           ;move 18 constants
E094- 05 D1          ORA *FNLEN    ;select CRTC register
E096- 8D 4C E8          STA PCR       ;and move constant
E099- A0 11          LDY #17      ;until all done
E09B- B1 C7    e09b    LDA (SAL),Y   ;get character from keyboard
E09D- 8C 80 E8          STY CRO      ;forward
E0A0- 8D 81 E8          STA CR1      ;enable interrupts again
E0A3- 88            DEY           ;echo character on screen
E0A4- 10 F5          BPL e09b    ;***** Get character from keyboard
E0A6- 60            RTS

        ;*****
        ;*
        ;* buffer.                         *
        ;*
        ;*****
E0A7- AC 6F 02    LP2    LDY KEYD      ;get oldest character
E0AA- A2 00          LDX #$00      ;move rest of buffer
E0AC- B0 70 02    LP1    LDA KEYD+1,X
E0AF- 9D 6F 02          STA KEYD,X      ;adapt index in buffer
E0B2- E8            INX           ;get character in accu
E0B3- E4 9E          CPX *NDX      ;enable interrupts again
E0B5- D0 F5          BNE LP1      ;***** Wait for RETURN from keyboard.
E0B7- C6 9E          DEC *NDX
E0B9- 98            TYA           ;*****
E0BA- 58            CLI           ;*****
E0BB- 60            RTS

        ;*****
        ;*
        ;*****
E0BC- 20 8C E6    LOOP4   JSR e68c

```

E0BF- A5 9E	LOOP3	LDA *NDX	;get index in keyb. buffer
E0C1- 85 A7		STA *BLNSW	;if empty, set cursor visible
E0C3- F0 FA		BEQ LOOP3	;else wait for character
E0C5- 78		SEI	;disable interrupts
E0C6- A5 AA		LDA *BLNON	;if blink fase one
E0C8- F0 09		BEQ LP21	
E0CA- A5 A9		LDA *GDBLN	;get true char. at cursor position
E0CC- A0 00		LDY #\$00	
E0CE- 84 AA		STY *BLNON	;set blink fase zero
E0D0- 20 06 E6		JSR DSPP	;and put char to screen
E0D3- 20 A7 E0	LP21	JSR LP2	;get character from buffer
E0D6- C9 83		CMP #\$83	;if shift-STOP
E0D8- D0 10		BNE LP22	
E0DA- 78		SEI	;disable interrupts
E0DB- A2 09		LDX #9	;move ten characters
E0DD- 86 9E		STX *NDX	;set index in buffer to 9
E0DF- BD 20 E7	LP23	LDA RUNTB-1,X	;move standard string
E0E2- 9D 6E 02		STA KEYD-1,X	;to buffer
E0E5- CA		DEX	
E0E6- D0 F7		BNE LP23	
E0E8- F0 D5		BEQ LOOP3	;and get 1st char. of that
E0EA- C9 00	LP22	CMP #\$00	;if is not RETURN
E0EC- D0 CE		BNE LOOP4	;go echo to screen and get next char
E0EE- A4 D5		LDY *LNMX	;else get max. length
E0F0- 84 AC		STY *CRSW	;flag input from screen
E0F2- B1 C4	CLP5	LDA (PNT),Y	;get char. from screen
E0F4- C9 20		CMP #\$20	;if space at end of line
E0F6- D0 03		BNE CLP6	
E0F8- 88		DEY	;get previous character
E0F9- D0 F7		BNE CLP5	
E0FB- C8	CLP6	INY	
E0FC- 84 A1		STY *INDX	;set record length of current line
E0FE- 20 CB E1		JSR e1cb	;set cursor on left margin
E101- EA		NOP	;returns with Y zero)
E102- 84 CD		STY *QTSW	;clear quote flag
E104- A5 A3		LDA *LXSP	;get line number on screen
E106- 30 19		BMI LOPS	;if valid
E108- C5 D8		CMP *TBLX	;and equal to original
E10A- D0 15		BNE LOPS	
E10C- A5 A4		LDA *LXSP+1	;set position
E10E- 85 C6		STA *PNTR	;to original value
E110- C5 A1		CMP *INDX	;if not behind end of record
E112- 90 00		BCC LOPS	;get char from device 3
E114- B0 2E		BCS CLP2	;else return RETURN

		;*	
		;* Get character from device 0 or *	
		;* device 3. *	
		;*	

E116- 98	LOOP5	TYA	
E117- 48		PHA	;save Y
E118- 8A		TXA	
E119- 48		PHA	;save X
E11A- 6C E9 00		JMP (\$00E9)	;and get input (default \$E110)
E11D- A5 AC	e11d	LDA *CRSW	;if input from keyboard
E11F- F0 9E		BEQ LOOP3	;go do keyboard until RETURN
		.FI "BASIC4.OEG8.M02"	

```

        ;*
        ;* Get character from current      *
        ;* screen line.                  *
        ;*                                *
        ;*****                           *
E121- A4 C6    LOPS   LDY *PNTR          ;get position
E123- B1 C4    LOPS   LDA (PNT),Y       ;get screen character
E125- 85 D9    LOPS   STA *DATA         ;save it
E127- 29 3F    LOPS   AND #$3F          ;remove bits 6 and 7
E129- 06 D9    LOPS   ASL *DATA         ;shift left
E12B- 24 D9    LOPS   BIT *DATA         ;if original bit 6 set
E12D- 10 02    LOPS   BPL LOP54
E12F- 09 80    LOP54  ORA #$80          ;set bit 7 (shift bit)
E131- 90 04    LOP54  BCC LOP52
E133- A6 CD    LOP54  LDX *QTSW
E135- D0 04    LOP54  BNE LOP53
E137- 70 02    LOP52  BVS LOP53
E139- 09 40    LOP52  ORA #$40          ;else if bit 5 set in original
E13B- E6 C6    LOP53  INC *PNTR         ;adapt position
E13D- 20 6A E1  LOP53  JSR QTSWC
E140- C4 A1    LOP53  CPY *INDX         ;adapt quote flag
E142- D0 17    LOP53  BNE CLP1
E144- A9 00    CLP2   LDA #$00          ;flag input from keyboard
E146- 85 AC    CLP2   STA *CRSW
E148- A9 00    CLP2   LDA #$0D          ;and echo RETURN
E14A- A6 AF    CLP2   LDX *DFLTN
E14C- E0 03    CLP2   CPX #3            ;if input device
E14E- F0 06    CLP2   BEQ CLP2A
E150- A6 B0    CLP2   LDX *DFLTO
E152- E0 03    CLP2   CPX #3            ;is screen
E154- F0 03    CLP2   BEQ CLP21
E156- 20 02 E2  CLP2A  JSR PRT
E159- A9 0D    CLP21  LDA #$0D          ;echo on screen
E15B- 85 D9    CLP21  STA *DATA         ;and store a RETURN
E15D- 68      CLP21  PLA
E15E- AA      CLP21  TAX
E15F- 68      CLP21  PLA
E160- A8      CLP21  TAY
E161- A5 D9    CLP21  LDA *DATA         ;restore Y
E163- C9 DE    CLP21  CMP #$DE          ;get character again
E165- D0 02    CLP21  BNE CLP7
E167- A9 FF    CLP21  LDA #PI           ;if code for PI
E169- 60      CLP7   RTS
        ;*****
        ;*                                *
        ;* Toggle quote flag if " found.  *
        ;*                                *
        ;*****                           *
E16A- C9 22    QTSWC  CMP #'"'          ;if character is "
E16C- D0 08    QTSWC  BNE QTSWL
E16E- A5 CD    QTSWC  LDA *QTSW         ;get quote flag
E170- 49 01    QTSWL  EOR #$01          ;invert bit 0
E172- 85 CD    QTSWL  STA *QTSW
E174- A9 22    QTSWL  LDA #'"'          ;and restore character
E176- 60      QTSWL  RTS
        ;*****
        ;*                                *
        ;* Fill screen at current      *
        ;* position.                  *
        ;*                                *
        ;*****                           *
E177- 09 40    NXT33  ORA #$40          ;map shifted characters
E179- A6 9F    NXT33  LDX #$FF          ;if reverse switch on
E17B- F0 02    NXT33  BEQ NVS

```

```

E17D- 09 80      NC3      ORA #$80          ;set screencode for reverse
E17F- A6 DC      NVS      LDX *INSRT        ;if in insert mode
E181- F0 02      BEQ NVSL
E183- C6 DC      DEC *INSRT        ;decrease insert count
E185- 20 06 E6    NVSL     JSR DSPP        ;put char at current position in RAM
E188- E6 C6      JSTS     INC *PNTR        ;advance character position
E18A- A4 D5      LDY *LNMX        ;if beyond maximum length
E18C- C4 C6      CPY *PNTR
E18E- 20 09      BCS LOOP2
E190- A6 D8      LDX *TBLX        ;get line number in X
E192- 20 A3 E3    JSTS1   JSR NXLN        ;scroll window if necessary
E195- A4 E2      LDY *$E2        ;get left margin
E197- 84 C6      STY *PNTR        ;put cursor there
;*****
;*
;* Exit from output to screen. *
;*
;*****
E199- A9 00      LOOP2   LDA #0
E19B- 85 E8      STA *$E8        ;clear HOME counter
E19D- 68         e19d    PLA
E19E- A8         TAY      ;restore Y
E19F- A5 DC      LDA *INSRT        ;if inserts outstanding
E1A1- F0 02      BEQ LOP2
E1A3- 46 CD      LSR *QTSW        ;cancel quote mode
E1A5- 68         LOP2    PLA
E1A6- AA         TAX      ;restore X
E1A7- 68         PLA      ;and A
E1A8- 58         CLI      ;allow interrupts again
E1A9- 60         RTS
;*****
;*
;* Backward over line boundary. *
;*
;*****
E1AA- A4 D5      BKLN    LDY *LNMX        ;get right margin
E1AC- A6 EO      LDX *XREC        ;and top line number of window
E1AE- E4 D8      CPX *TBLX        ;if on top line
E1B0- 90 08      BCC e1ba
E1B2- A4 E2      LDY *$E2        ;get left margin
E1B4- 84 C6      STY *PNTR        ;and put cursor home
E1B6- 68         PLA
E1B7- 68         PLA      ;remove return address
E1B8- D0 DF      BNE LOOP2        ;and exit
E1BA- C6 D8      e1ba    DEC *TBLX        ;move cursor to previous line
E1BC- 84 C6      STY *PNTR        ;and put it on right margin
E1BE- 4C 67 EO    JMP STUPT       ;and get pointer of that line
;*****
;*
;* Erase line until right margin. *
;*
;*****
E1C1- A9 20      LDA #$20        ;get blank
E1C3- C8         INY
E1C4- 91 C4      STA (PNT),Y      ;store on line
E1C6- C4 D5      CPY *LNMX        ;until right margin reached
E1C8- 90 F9      BCC e1c3
E1CA- 60         RTS
;*****
;*
;* Set cursor on left margin. *
;*
;*****
E1CB- A4 E2      e1cb    LDY *$E2        ;get left margin

```

```

E1CD- 84 C6      STY *PNTR          ;put cursor there
E1CF- A0 00      LDY #0            ;prepare for cancel quote mode
E1D1- 60        RTS
;*****
;*                                     *
;* Set window to full screen.      *
;*                                     *
;*****
E1D2- A9 00      e1d2   LDA #0          ;get top line #
E1D4- AA         TAX               ;and left margin
E1D5- 20 E1 E1    JSR e1e1          ;set them
E1D8- A9 18      LDA #24          ;get bottom line #
E1DA- A2 4F      LDX #79          ;and right margin
;*****
;*                                     *
;* Set bottom right of window.    *
;*                                     *
;*****
E1DC- 85 E1      e1dc   STA *XWRT       ;store bottom line #
E1DE- 86 D5      STX *LNMX       ;and right margin
E1E0- 60        RTS
;*****
;*                                     *
;* Set top left of window.       *
;*                                     *
;*****
E1E1- 85 E0      e1e1   STA *XREC       ;set top line #
E1E3- 86 E2      STX *$E2        ;and left margin
E1E5- 60        RTS
E1E6- AA         TAX
E1E7- AA         TAX
E1E8- AA         TAX
E1E9- AA         TAX
E1EA- AA         TAX
E1EB- AA         TAX
E1EC- AA         TAX
E1ED- AA         TAX
E1EE- AA         TAX
E1EF- AA         TAX
E1F0- AA         TAX
E1F1- AA         TAX
E1F2- AA         TAX
E1F3- AA         TAX
E1F4- AA         TAX
E1F5- AA         TAX
E1F6- AA         TAX
E1F7- AA         TAX
E1F8- AA         TAX
E1F9- AA         TAX
E1FA- AA         TAX
E1FB- AA         TAX
E1FC- AA         TAX
E1FD- AA         TAX
E1FE- AA         TAX
E1FF- AA         TAX
E200- AA         TAX
E201- AA         TAX
.FI "BASIC4.OEG8.M03"

```

17EB 34A7-4C92 BASIC4.OEG8.M03

```

;name BASIC4.OEG8.M03
;*****

```

```

*  

;* Put character to screen.  

;*  

;*****  

E202- 48      PRT    PHA          ;save character  

E203- 85 09    STA *DATA   ;also in temporary field  

E205- 8A       TXA  

E206- 48       PHA          ;save X  

E207- 98       TYA  

E208- 48       PHA          ;and Y  

E209- 6C EB 00  JMP ($00EB) ;and go output (default e20c)  

E20C- A9 00    e20c    LDA #0  

E20E- 85 AC    STA *CRSW  ;set copy of position to zero  

E210- A4 C6    LDY *PNTR  ;get position  

E212- EA       NOP          ;test for ESC  

E213- EA       NOP          ;used to be  

E214- EA       NOP          ;here, to cancel  

E215- EA       NOP          ;quote mode  

E216- EA       NOP          ;in this ROM  

E217- EA       NOP          ;this is  

E218- EA       NOP          ;achieved by  

E219- EA       NOP          ;the STOP key  

E21A- EA       NOP          ;in the interrupt  

E21B- EA       NOP          ;routine  

E21C- EA       NOP  

E21D- A5 09    LDA *DATA  ;get character  

E21F- 10 03    BPL e224  ;if shifted  

E221- 4C F4 E2  JMP NXTX  ;go do shifted character  

E224- C9 00    e224    CMP #$0D  ;if RETURN  

E226- D0 03    BNE NJTL  ;go do RETURN  

E228- 4C B6 E3  JMP e3b6  ;if printable  

E22B- C9 20    NJTL    CMP #$20  

E220- 90 08    BCC NTCN  ;convert to screen code  

E22F- 29 3F    AND #$3F  ;check for quote mode  

E231- 20 6A E1  JSR QTSWC ;and put to screen  

E234- 4C 79 E1  JMP NXT3  ;if inserts outstanding  

E237- A6 DC    NTCN    LDX *INSRT  

E239- F0 03    BEQ CNC3X ;print as control character  

E23B- 4C 70 E1  JMP NC3  ;if DELETE  

E23E- C9 14    CNC3X    CMP #$14  

E240- D0 20    BNE NTCN1 ;get left margin  

E242- A4 E2    LDY #$E2  ;if cursor after that  

E244- C4 C6    CPY *PNTR  ;go move cursor left  

E246- 90 05    BCC e24d  ;else go backward over boundary  

E248- 20 AA E1  JSR BKLN  ;and add a blank at the end  

E24B- D0 0F    BNE BK2   ;move cursor left  

E24D- C6 C6    e24d    DEC *PNTR  

E24F- A4 C6    LDY *PNTR  

E251- C8       BK1     INY  

E252- B1 C4    LDA (PNT),Y ;compress  

E254- 88       DEY  

E255- 91 C4    STA (PNT),Y ;line  

E257- C8       INY  

E258- C4 05    CPY *LNMX ;upto  

E25A- D0 F5    BNE BK1  ;end of  

E25C- A9 20    BK2     LDA #$20 ;right margin  

E25E- 91 C4    STA (PNT),Y ;and insert blank  

E260- D0 37    BNE JPL3  ;at end  

E262- A6 CD    NTCN1  LDX *QTSW ;if in quote mode  

E264- F0 03    BEQ NC3W  ;print as control character  

E266- 4C 70 E1  CNC3    JMP NC3  ;if RVS-ON  

E269- C9 12    NC3W    CMP #$12  

E26B- D0 02    BNE NC1  ;set reverse flag  

E26D- 85 9F    STA *RVS

```

E26F-	C9 13	NC1	CMP #\$13	;if HOME
E271-	D0 10		BNE NC2	
E273-	A5 E8		LDA *\$E8	;get HOME counter
E275-	10 04		BPL e27b	;if' second home
E277-	20 D2 E1		JSR e1d2	;set window to full screen
E27A-	18		CLC	
E27B-	66 E8	e27b	ROR *\$E8	;adjust counter
E27D-	20 5F E0		JSR NXTD	;and set cursor to top left
E280-	4C 90 E1		JMP e19d	;and exit
E283-	C9 1D	NC2	CMP #\$1D	;if cursor-RIGHT
E285-	D0 0B		BNE NCX2	
E287-	C8		INY	
E288-	84 C6		STY *PNTR	;advance cursor
E28A-	88		DEY	
E28B-	C4 D5		CPY *LNMX	;if at right margin
E28D-	90 0A		BCC JPL3	
E28F-	4C 92 E1		JMP JSTS1	;go to next line and exit
E292-	C9 11	NCX2	CMP #\$11	;if cursor-DOWN
E294-	D0 06		BNE e29c	
E296-	20 A3 E3		JSR NXLN	;move cursor down and scroll if necessa
E299-	4C 99 E1	JPL3	JMP LOOP2	;and exit
E29C-	C9 09	e29c	CMP #\$09	;if TAB
E29E-	D0 30		BNE e2d0	
E2A0-	20 70 E5		JSR e570	;get bitmask and offset
E2A3-	AC 3A 03	e2a3	LDY \$033A	;get position of next tab
E2A6-	EE 3A 03		INC \$033A	
E2A9-	C4 D5		CPY *LNMX	;if after right margin
E2AB-	90 09		BCC e2b6	
E2AD-	A5 D5		LDA *LNMX	;get right margin
E2AF-	85 C6		STA *PNTR	;put cursor there
E2B1-	CE 3A 03		DEC \$033A	
E2B4-	D0 E3		BNE JPL3	;and exit
E2B6-	0E 3E 03	e2b6	ASL \$033E	;else adapt bitmask
E2B9-	D0 0A		BNE e2c5	;if byte exceeded
E2BB-	E8		INX	;adapt index
E2BC-	E0 0A		CPX #10	;if at last possible index
E2BE-	F0 D9		BEQ JPL3	;exit
E2C0-	A9 01		LDA #1	;else set up
E2C2-	8D 3E 03		STA \$033E	;new bitmask
E2C5-	B0 EE 03	e2c5	LDA \$03EE,X	;get tabstopbyte
E2C8-	2D 3E 03		AND \$033E	;test mask, if no tabstop
E2CB-	F0 D6		BEQ e2a3	;check next column
E2CD-	C8		INY	;else
E2CE-	84 C6		STY *PNTR	;put cursor there
E2D0-	C9 16	e2d0	CMP #\$16	;if ERASE TO END OF LINE
E2D2-	D0 DC		BNE e2e0	
E2D4-	A9 20		LDA #\$20	;get blank
E2D6-	88		DEY	;erase cursor's position also
E2D7-	C8	e2d7	INY	;why not use XWRTC1?
E2D8-	91 C4		STA (PNT),Y	;fill line with blanks
E2DA-	C4 D5		CPY *LNMX	;until right margin reached
E2DC-	90 F9		BCC e2d7	
E2DE-	B0 B9		BCS JPL3	;if at end, exit
E2E0-	C9 15	e2e0	CMP #\$15	;if not DELETE LINE
E2E2-	F0 03		BEQ e2e7	
E2E4-	4C 91 E5		JMP e591	;do rest of control charscters
E2E7-	A5 E0	e2e7	LDA *XREC	;else get top line #
E2E9-	48		PHA	;save it
E2EA-	A5 D8		LDA *TBLX	;get current line #
E2EC-	85 E0		STA *XREC	;store in top line #
E2EE-	20 E8 E3		JSR SCROL	;scroll rest of window up
E2F1-	4C CA E5		JMP e5ca	;restore top line # and exit

			;	*

```

;* Put shifted characters to *
;* screen.
;*
;*****
E2F4- 29 7F      NXTX    AND #$7F          ;remove shift bit
E2F6- C9 7F      CMP #PI-$80        ;if code for PI
E2F8- D0 02      BNE NXTX1
E2FA- A9 5E      LDA #$5E          ;get real code for PI
E2FC- C9 20      CMP #$20          ;if printable
E2FE- 90 03      BCC e303
E300- 4C 77 E1    JMP NXT33        ;go fill screen
E303- C9 00      e303   CMP #$00          ;if shift-RETURN
E305- D0 03      BNE UPS
E307- 4C B6 E3    JMP e3b6          ;go do RETURN
E30A- A6 CD      UPS    LDX *QTSW        ;if quote mode
E30C- D0 34      BNE UP6           ;print as control character
E30E- C9 14      CMP #$14          ;no quote mode, if INSERT
E310- D0 2C      BNE UP9
E312- A4 05      LDY *LNMX        ;get right margin
E314- B1 C4      LDA (PNT),Y    ;get last char. of line
E316- C9 20      CMP #$20          ;if blank
E318- D0 72      BNE JLP2
E31A- C4 C6      CPY *PNTR        ;and not on last position
E31C- 90 6E      BCC JLP2        ;then enough space on line
E31E- F0 6C      BEQ JLP2
E320- A4 05      INS1   LDY *LNMX        ;get right margin
E322- 88         INS2   DEY
E323- B1 C4      LDA (PNT),Y    ;characters
E325- C8         INY
E326- 91 C4      STA (PNT),Y    ;one position
E328- 88         DEY
E329- C4 C6      CPY *PNTR        ;further
E32B- D0 F5      BNE INS2        ;until
E32D- A9 20      LDA #$20          ;all done
E32F- 91 C4      STA (PNT),Y    ;and put a blank
E331- A5 D5      LDA *LNMX        ;in space opened up
E333- 38         SEC
E334- E5 C6      SBC *PNTR        ;get right margin
E336- E5 DC      SBC *INSRT       ;subtract position
E338- 30 52      BMI JLP2        ;and number of inserts
E33A- E6 DC      INC *INSRT       ;if not positive, exit
E33C- D0 4E      BNE JLP2        ;else add an insert
E33E- A6 DC      UP9    LDX *INSRT       ;and exit
E340- F0 05      BEQ UP2          ;if inserts outstanding
E342- D9 40      UP6    ORA #$40          ;convert to screen code
E344- 4C 70 E1    JMP NC3          ;and print as control char.
E347- C9 11      UP2    CMP #$11          ;if cursor-UP
E349- D0 00      BNE NXT2
E34B- A6 E0      LDX *XREC        ;get top line #
E34D- E4 D8      CPX *TBLX        ;if on top line or above
E34F- B0 3B      BCS JLP2        ;exit
E351- C6 D8      DEC *TBLX        ;else move cursor up
E353- 20 67 E0    JSR STUPT        ;get pointer to new line
E356- D0 34      BNE JLP2        ;and exit
E358- C9 12      NXT2   CMP #$12          ;if RVS-off
E35A- D0 04      BNE NXT6
E35C- A9 00      LDA #$00
E35E- 85 9F      STA *RVS         ;reset reverse flag
E360- C9 10      NXT6   CMP #$10          ;if cursor-LEFT
E362- D0 0F      BNE e373
E364- A4 E2      LDY *$E2          ;get left margin
E366- C4 C6      CPY *PNTR        ;if left to that
E368- 90 05      BCC e36f
E36A- 20 AA E1    JSR BKLN        ;go backward over boundary

```

E360- D0 10		BNE JLP2	;and exit
E36F- C6 C6	e36f	DEC *PNTR	;else move cursor left
E371- 10 19		BPL JLP2	;and exit
E373- C9 13	e373	CMP #\$13	;if CLR(-window)
E375- D0 05		BNE e37c	
E377- 20 51 E0		JSR CLSR	;clear the window
E37A- D0 10		BNE JLP2	;and exit
E37C- C9 09	e37c	CMP #\$09	;if shift-TAB
E37E- D0 0F		BNE e38f	
E380- 20 70 E5		JSR e570	;get bitmask and index
E383- BD EE 03		LDA \$03EE,X	;get tabstopbyte
E386- 4D 3E 03		EOR \$033E	;add or remove bit
E389- 9D EE 03		STA \$03EE,X	;store again
E38C- 4C 99 E1	JLP2	JMP LOOP2	;and exit
E38F- C9 16	e38f	CMP #\$16	;if not ERASE FROM START
E391- F0 03		BEQ e396	
E393- 4C BC E5		JMP e5bc	;go do rest of characters
E396- A9 20	e396	LDA #\$20	;else get blank
E398- A4 E2		LDY *\$E2	;start at left margin
E39A- C4 C6	e39a	CPY *PNTR	;if current position reached
E39C- B0 EE		BCS JLP2	;exit
E39E- 91 C4		STA (PNT),Y	;else store blank
E3A0- C8		INY	
E3A1- D0 F7		BNE e39a	;and repeat

		;*	
		;* Action for cursor down; *	
		;* scroll window up if at bottom. *	
		;*	

E3A3- 46 A3	NXLN	LSR *LXSP	;set copy of line # invalid
E3A5- A6 D8		LDX *TBLX	;get line #
E3A7- E4 E1		CPX *XWRT	;if at bottom of window
E3A9- 90 06		BCC e3b1	
E3AB- 20 E8 E3		JSR SCROL	;scroll window up
E3AE- 4C 67 E0		JMP STUPT	;and get pointer to line
E3B1- E6 D8	e3b1	INC *TBLX	;else adapt line #
E3B3- 4C 67 E0		JMP STUPT	;and get pointer
		.FI "BASIC4.OEG8.M04"	

15C5 34A7-4A6C · BASIC4.OEG8.M04

;name BASIC4.OEG8.M04			

		;*	
		;* Process RETURN. *	
		;*	

E3B6- A4 E2	e3b6	LDY *\$E2	;get left margin
E3B8- 84 C6		STY *PNTR	;put cursor there
E3BA- 20 A3 E3		JSR NXLN	;perform cursor down
E3BD- A9 00	NXTL	LDA #0	
E3BF- 85 DC		STA *INSRT	;cancel inserts outstanding
E3C1- 85 9F		STA *RVS	;cancel reverse field
E3C3- 85 CD		STA *QTSW	;cancel quote mode
E3C5- 4C 99 E1		JMP LOOP2	;and exit

		;*	
		;* Scroll window down. *	
		;*	

E3C8- A6 E1	NEWLN	LDX *XWRT	;get bottom line #
E3CA- E8		INX	

E3CB- CA	NXTL1	DEX	
E3CC- 20 6C E0		JSR STUPR	;get pointer and left margin
E3CF- E4 E0		CPX *XREC	;if top line of window reached
E3D1- F0 35		BEQ MLP4	;check for paused scroll
E3D3- BD 54 E7		LDA LDTB2-1,X	
E3D6- 85 C7		STA *SAL	;get lo-byte of previous line
E3D8- BD 60 E7		LDA LDTB1-1,X	;ge hi-byte also.
E3DB- 85 C8		STA *SAH	;store the pointer
E3DD- C8	NELL	INY	
E3DE- B1 C7		LDA (SAL),Y	;get a character of prev. line
E3E0- 91 C4		STA (PNT),Y	;move to current line
E3E2- C4 D5		CPY *LNMX	;until right margin reached
E3E4- 90 F7		BCC NELL	
E3E6- B0 E3		BCS NXTL1	;and do previous line
		;	*****
		;	*
		;	Scroll window up.
		;	*
		;	*
		;	*****
E3E8- A6 E0	SCROL	LDX *XREC	;get top line #
E3EA- CA		DEX	
E3EB- E8	SCRL5	INX	
E3EC- 20 6C E0		JSR STUPR	;get pointer and left margin
E3EF- E4 E1		CPX *XWRT	;if bottom reached
E3F1- B0 15		BCS MLP4	;check for paused scroll
E3F3- BD 56 E7		LDA LDTB2+1,X	;get pointer to next line
E3F6- 85 C7		STA *SAL	;store lo-byte
E3F8- BD 6F E7		LDA LDTB1+1,X	;do hi-byte also
E3FB- 85 C8		STA *SAH	
E3FD- C8	MLP1	INY	;do next character
E3FE- B1 C7		LDA (SAL),Y	;move from next line
E400- 91 C4		STA (PNT),Y	;to current line
E402- C4 D5		CPY *LNMX	;until right margin reached
E404- 90 F7		BCC MLP1	
E406- B0 E3		BCS SCRL5	;then do next line
		;	*****
		;	*
		;	Insert blank line and pause
		;	*
		;	scroll if appropriate key is
		;	*
		;	*
		;	*
		;	*****
E408- 20 C1 E1	MLP4	JSR e1c1	;insert blank line
E40B- A0 00		LDY #0	
E40D- AD 12 E8		LDA PIAK	;get key image
E410- C9 FB		CMP #\$FB	;if space pressed
E412- D0 09		BNE e41d	
E414- EA	MLP41	NOP	;wait
E415- CA		DEX	;a
E416- D0 FC		BNE MLP41	;while
E418- 88		DEY	;with a
E419- D0 F9		BNE MLP41	;double loop
E41B- F0 0B		BEQ MLP42	;then cancel buffer and exit
E41D- C9 FE	e41d	CMP #\$FE	;if not RVS pressed
E41F- D0 09		BNE e42a	;exit
E421- AD 12 E8	e421	LDA PIAK	;else get key image
E424- C9 FB		CMP #\$FB	
E426- D0 F9		BNE e421	;if no space, wait
E428- 84 9E	MLP42	STY *NDX	;if space, cancel buffer
E42A- 60	e42a	RTS	;and exit
E42B- AA		TAX	
E42C- AA		TAX	
E42D- AA		TAX	
E42E- AA		TAX	

```

E42F- AA          TAX
E430- .AA .       TAX
;*****  

;*  

;* Do clock ticks;  

;* add an extra tick in every  

;* five (because of 50 Hz screen  

;* refresh instead of the usual  

;* 60 Hz).  

;*  

;*****  

E431- 20 EA FF  e431 JSR $FFEA      ;do clock tick
E434- E6 F8       INC *$F8        ;increment tick counter
E436- A5 F8       LDA *$F8        ;if 5 ticks done
E438- C9 06       CMP #6
E43A- D0 1C       BNE e458      ;if not five, do keyboard
E43C- A9 00       LDA #0
E43E- 85 F8       STA *$F8        ;clear counter
E440- F0 EF       BEQ e431      ;and do extra tick
;*****  

;*  

;* Interrupt routine;  

;* entered 50 times per second.  

;*  

;* (Pointed to by ROM).  

;*  

;*****  

E442- 48          PULS  PHA      ;save A
E443- 8A          TXA
E444- 48          PHA      ;save X
E445- 98          TYA
E446- 48          PHA      ;save Y
E447- BA          TSX
E448- BD 04 01    LDA $0104,X   ;get saved statusregister
E44B- 29 10        AND #$10      ;filter BRK-bit out
E44D- F0 03        BEQ PULS1     ;if BRK
E44F- 6C 92 00    JMP (CBINV)   ;go do BRK routine
E452- 6C 90 00    PULS1 JMP (CINV)   ;else do IRQ routine (default KEY)
;*****  

;*  

;* Default interrupt routine.  

;*  

;* (Pointed to by ($90)).  

;*  

;*****  

E455- 4C 31 E4    KEY   JMP e431      ;do clock tick(s)
;*****  

;*  

;* Rest of interrupt routine;  

;* do cursor blink.  

;*  

;*****  

E458- A5 A7        e458  LDA *BLNSW    ;if cursor invisible
E45A- D0 1E        BNE e47a      ;skip blink handling
E45C- C6 A8        DEC *BLNCT    ;decrement countdown
E45E- D0 1A        BNE e47a      ;if countdown zero
E460- A9 10        LDA #16      ;set new countdown
E462- EA          NOP
E463- EA          NOP
E464- EA          NOP
E465- EA          NOP
E466- EA          NOP
E467- EA          NOP
E468- 85 A8        e468  STA *BLNCT

```

E46A- A4 C6		LDY *PNTR ;get column position
E46C- 46 AA		LSR *BLNON ;determine blink phase
		;and set phase zero
E46E- B1 C4		LDA (PNT),Y ;get character at current position
E470- B0 04		BCS KEYS ;if in phase zero
E472- E6 AA		INC *BLNON ;set blink phase one
E474- 85 A9		STA *GDBLN ;and store true character
E476- 49 80		EOR #\$80 ;reverse screen character
E478- 91 C4		STA (PNT),Y ;and put to screen
		;*****
		;* * *
		;* Cassette motor control.
		;* * *
		;*****
E47A- A0 00	e47a	LDY #\$00
E47C- AD 10 E8		LDA PIAL
E47F- 29 F0		AND #\$FD ;set zero in
E481- 8D 10 E8		STA PIAL ;keyboard scan line number
E484- AD 10 E8		LDA PIAL ;get cassette sense lines
E487- 0A		ASL A
E488- 0A		ASL A
E489- 0A		ASL A
E48A- 10 09		BPL KEY3 ;if no key down on cassette 1.
E48C- 84 F9		STY *CAS1 ;flag cassette 1 off
E48E- AD 13 E8		LDA PIAS
E491- 09 08		ORA #\$08 ;and switch motor off
E493- D0 09		BNE KL24 ;and test cass#2
E495- A5 F9	KEY3	LDA *CAS1 ;key down,if #1 flag on
E497- D0 08		BNE KL2 ;test cass#2
E499- AD 13 E8		LDA PIAS ;else
E49C- 29 F7		AND #\$F7 ;start cass#1 motor
E49E- 8D 13 E8	KL24	STA PIAS
E4A1- 90 09	KL2	BCC KL23 ;if no key down on cass.#2
E4A3- 84 FA		STY *CAS2 ;flag cass.#2 off
E4A5- AD 40 E8		LDA PIA
E4A8- 09 10		ORA #\$10 ;and switch cass#2 motor off
E4AA- D0 09		BNE KL25
E4AC- A5 FA	KL23	LDA *CAS2 ;key down,if #2 flag on
E4AE- D0 08		BNE KL22 ;finish cassette control
E4B0- AD 40 E8		LDA PIA ;else
E4B3- 29 EF		AND #\$EF ;switch #2 motor on
E4B5- 8D 40 E8	KL25	STA PIA
E4B8- 20 BE E4	KL22	JSR e4be ;handle keyboard
E4BB- 4C 00 E6		JMP PREND ;and return from interrupt
		.FI "BASIC4.OEG8.MOS"

0993 34A7-3E3A BASIC4.OEG8.MOS

		;name BASIC4.OEG8.MOS
		;*****
		;* * *
		;* Scan keyboard. *
		;* * *
		;*****
E4BE- A0 FF	e4be	LDY #\$FF ;this index received
E4C0- 84 A6		STY *SF0X ;on no key at all
E4C2- C8		INY
E4C3- 84 98		STY *SHFLAG ;clear shift key indicator
E4C5- A2 50		LDX #80 ;this number of keys to be done
E4C7- A0 08	e4c7	LDY #8 ;number of bits
E4C9- AD 12 E8		LDA PIAK ;get key image
E4CC- CD 12 E8		CMP PIAK ;wait
E4CF- D0 F6		BNE e4c7 ;until steady

E4D1- 4A	KL1	LSR A	;get bit in carry
E4D2- B0 13		BCS CKIT	;if key pressed
E4D4- 48		PHA	;save key image
E4D5- BD D0 E6		LDA CHAR-1,X	;get value for that key
E4D8- D0 06		BNE CKIS1	;if shift key
E4DA- A9 01		LDA #1	
E4DC- 85 98		STA *SHFLAG	;set flag for shift key
E4DE- D0 06		BNE CKUT	;and loop
E4E0- C9 FF	CKIS1	CMP #\$FF	;if no key
E4E2- F0 02		BEQ CKUT	;check for next key
E4E4- 85 A6		STA *SFDX	;else store key value
E4E6- 68	CKUT	PLA	;reget key image
E4E7- CA	CKIT	DEX	;decrement key counter
E4E8- F0 08		BEQ CKIT1	;if not at end
E4EA- 88		DEY	;decrement bit counter
E4EB- D0 E4		BNE KL1	;and check next bits
E4ED- EE 10 E8		INC PIAL	;do next scan line
E4F0- D0 05		BNE e4c7	;and loop
E4F2- A2 10	CKIT1	LDX #16	;get countdown
E4F4- A5 A6		LDA *SFDX	;get key found
E4F6- C9 FF		CMP #\$FF	;if no key
E4F8- F0 28		BEQ e522	;exit
E4FA- C5 97		CMP *LSTX	;if same as last time
E4FC- D0 24		BNE e522	;do repeat
E4FE- 24 E4		BIT *RPTFLG	;test repeat flag
E500- 70 4A		BVS e54c	;if repeat off, exit
E502- 10 10		BPL e514	;if repeat on, go repeat
E504- C9 14		CMP #\$14	;else if INST/DEL
E506- F0 0C		BEQ e514	;do repeat
E508- C9 20		CMP #\$20	;if SPACE
E50A- F0 08		BEQ e514	;do repeat
E50C- C9 10		CMP #\$10	;if LEFT/RIGHT
E50E- F0 04		BEQ e514	;do repeat
E510- C9 11		CMP #\$11	;if not UP/DOWN
E512- D0 38		BNE e54c	;exit
E514- A4 9E	e514	LDY *NDX	;get # of keys in buffer
E516- 88		DEY	
E517- 10 33		BPL e54c	;if not empty, exit
E519- C6 E5		DEC *KOUNT	;decrement countdown/delay
E51B- D0 2F		BNE e54c	;if not expired, exit
E51D- A2 01		LDX #1	;else
E51F- 86 A8		STX *BLNCT	;store new flash countdown
E521- E8		INX	
E522- 86 E5	e522	STX *KOUNT	;reload delay
E524- 85 97		STA *LSTX	;store new key value
E526- C9 FF		CMP #\$FF	;if no key
E528- F0 22		BEQ e54c	;exit
E52A- 08		PHP	;save flags
E52B- 29 7F		AND #\$7F	;remove shift bit
E52D- 28		PLP	;get flags again
E52E- 30 06		BMI KN1	;if msb was set, ignore msb
E530- 46 98		LSR *SHFLAG	;get and clear shift status
E532- 90 02		BCC KN1	;if pressed
E534- 09 80		ORA #\$80	;add shift bit
E536- A6 9E	KN1	LDX *NDX	;get # of keys in buffer
E538- E4 E3		CPX *XMAX	;if equal or more than max. length
E53A- B0 10		BCS e54c	;exit (ignore key)
E53C- C9 03		CMP #3	;if STOP key
E53E- D0 06		BNE e546	
E540- A0 00		LDY #0	
E542- 84 CD		STY *QTSW	;cancel quote mode
E544- 84 DC		STY *INSRT	;and zero inserts outstanding
E546- 9D 6F 02	e546	STA KEYD,X	;else store in buffer
E549- E8		INX	

ES4A- 86 9E		STX *NDX	;and adapt buffer counter
ES4C- 60	e54c	RTS	
ES4D- AA		TAX	
ES4E- AA		TAX	
ES4F- AA		TAX	
ES50- AA		TAX	
ES51- AA		TAX	
ES52- AA		TAX	
ES53- AA		TAX	
ES54- AA		TAX	
ES55- AA		TAX	
ES56- AA		TAX	
ES57- AA		TAX	
ES58- AA		TAX	
ES59- AA		TAX	
ES5A- AA		TAX	
ES5B- AA		TAX	
ES5C- AA		TAX	
ES5D- AA		TAX	
ES5E- AA		TAX	
ES5F- AA		TAX	
ES60- AA		TAX	
ES61- AA		TAX	
ES62- AA		TAX	
ES63- AA		TAX	
ES64- AA		TAX	
ES65- AA		TAX	
ES66- AA		TAX	
ES67- AA		TAX	
ES68- AA		TAX	
ES69- AA		TAX	
ES6A- AA		TAX	
ES6B- AA		TAX	
ES6C- AA		TAX	
ES6D- AA		TAX	
ES6E- AA		TAX	
ES6F- AA		TAX	

.FI "BASIC4.OEG8.M06"

194F 34A7-40F6 BASIC4.OEG8.M06

```

;name BASIC4.OEG8.M06
;*****
;*
;* Get bitmask for TAB table in    *
;* $033E and offset in table in   *
;*                                *
;*                                *
;*****
```

ES70- A5 C6	e570	LDA *PNTR	;get column #
ES72- 29 F8.		AND #\$F8	;remove 3 lowest bits
ES74- 80 3A 03		STA \$033A	;save it
ES77- 4A		LSR A	
ES78- 4A		LSR A	
ES79- 4A		LSR A	;divide it by 8
ES7A- AA		TAX	;and move to X
ES7B- A9 01		LDA #1	;load bit mask
ES7D- 80 3E 03		STA \$033E	
ES80- A4 C6		LDY *PNTR	;get column # again
ES82- CC 3A 03	e582	CPY \$033A	;if position found
ES85- F0 09		BEQ e590	;exit.
ES87- 0E 3E 03		ASL \$033E	;adapt bit mask
ES8A- EE 3A 03		INC \$033A	;and count bits

E58D- 4C 82 E5		JMP e582	;until position found
E590- 60	e590	RTS	

		;*	
		;* Do rest of unshifted *	
		;* characters. *	
		;*	

E591- C9 19	e591	CMP #\$19	;if SCROLL UP
E593- D0 06		BNE e59b	
E595- 20 E8 E3		JSR SCROL	;scroll window up
E598- 4C 09 E5		JMP e5d9	;get pointer to line and exit
E59B- C9 0F	e59b	CMP #\$0F	;if SET TOP
E59D- D0 0B		BNE e5aa	
E59F- A5 08		LDA *TBLX	;get current line#
E5A1- 85 E0		STA *XREC	;store in top line#
E5A3- A5 C6		LDA *PNTR	;get current column#
E5A5- 85 E2		STA *\$E2	;store in left margin
E5A7- 4C 99 E1	e5a7	JMP LOOP2	;and exit
E5AA- C9 0E	e5aa	CMP #\$0E	;if TEXT
E5AC- D0 05		BNE e5b3	
E5AE- 20 7A E0		JSR e07a	;set CRT controller to text
E5B1- 30 F4		BMI e5a7	;and exit
E5B3- C9 07	e5b3	CMP #\$07	;if not BELL
E5B5- D0 F0		BNE e5a7	;exit (ignore character)
E5B7- 20 A7 E6		JSR e6a7	;if BELL, ring it
E5BA- F0 EB		BEQ e5a7	;and exit

		;*	
		;* Do rest of shifted *	
		;* characters. *	

E5BC- C9 15	e5bc	CMP #\$15	;if INSERT LINE
E5BE- D0 12		BNE e5d2	
E5C0- A5 E0		LDA *XREC	;get top line#
E5C2- 48		PHA	;save it
E5C3- A5 08		LDA *TBLX	;get current line#
E5C5- 85 E0		STA *XREC	;put in top line#
E5C7- 20 C8 E3		JSR NEWLN	;scroll 'window' down and insert line
E5CA- 68	e5ca	PLA	
E5CB- 85 E0		STA *XREC	;restore original top line#
E5CD- 20 63 E0		JSR e063	;set cursor at start and get
E5D0- D0 18		BNE e5ea	;and exit
E5D2- C9 19	e5d2	CMP #\$19	;if SCROLL DOWN
E5D4- D0 08		BNE e5de	
E5D6- 20 C8 E3		JSR NEWLN	;scroll window down
E5D9- 20 67 E0	e5d9	JSR STUPT	;get pointer to line
E5DC- D0 0C		BNE e5ea	;and exit
E5DE- C9 0F	e5de	CMP #\$0F	;if SET BOTTOM
E5E0- D0 0B		BNE e5ed	
E5E2- A5 D8		LDA *TBLX	;get current line#
E5E4- 85 E1		STA *XWRT	;store in bottom line#
E5E6- A5 C6		LDA *PNTR	;get current column#
E5E8- 85 D5		STA *LNMX	;store in right margin
E5EA- 4C 99 E1	e5ea	JMP LOOP2	;and exit
E5ED- C9 0E	e5ed	CMP #\$0E	;if not GRAPHICs
E5EF- D0 C2		BNE e5b3	;check for BELL
E5F1- 20 82 E0		JSR e082	;else set to graphics
E5F4- 30 F4		BMI e5ea	;and exit
E5F6- AA		TAX	
E5F7- AA		TAX	
E5F8- AA		TAX	
E5F9- AA		TAX	

```

ESFA- AA      TAX
ESFB- AA      TAX
ESFC- AA      TAX
ESFD- AA      TAX
ESFE- AA      TAX
ESFF- AA      TAX
;*****
;*
;* Exit from interrupt. *
;*
;*****
E600- 68      PREND PLA
E601- A8      TAY      ;restore Y
E602- 68      PLA
E603- AA      TAX      ;and X
E604- 68      PLA      ;and A
E605- 40      RTI      ;and return from interrupt
;*****
;*
;* Store the character in A   *
;* on screen.               *
;*
;*****
E606- A4 C6      DSPP LDY *PNTR      ;get cursor position on line
E608- 91 C4      STA (PNT),Y    ;store the character
E60A- A9 01      LDA #1       ;set up new countdown
E60C- 85 A8      STA *BLNCT     ;for cursor flash
E60E- 60      RTS       ;then exit
;*****
;*
;* Reset; initialize I/O.   *
;*
;*****
E60F- A9 7F      e60f  LDA #$7F
E611- 8D 4E E8      STA IER      ;disable all VIA interrupts
E614- A2 60      LDX #$60
E616- A9 00      LDA #$00      ;clear
E618- 95 8D      STA *CTIMR,X  ;area $8D to $FA
E61A- CA      DEX
E61B- 10 FB      BPL e618
E61D- A2 0C      LDX #12
E61F- 9D EE 03      e61f STA $03EE,X  ;clear TAB-stop table
E622- CA      DEX
E623- 10 FA      BPL e61f
E625- A9 55      LDA #L,KEY   ;set
E627- 85 90      STA *CINV     ;IRQ RAM vector
E629- A9 E4      LDA #H,KEY   ;to
E62B- 85 91      STA *CINV+1  ;keyboard/cassette routine
E62D- A9 03      LDA #3       ;set default output device
E62F- 85 B0      STA *DFLTO    ;to screen
E631- A9 0A      LDA #10      ;get maximum
E633- 85 E3      STA *XMAX    ;keyboard buffer length
E635- A9 0F      LDA #0F      ;keyboard PIA, A section:
E637- 8D 10 E8      STA PIAL    ;bits 0-3 output, rest input
E63A- 0A      ASL A
E63B- 8D 40 E8      STA PIA     ;VIA: cassette#2 motor off
;cassette #1 and #2 write = 1
;IEEE ATN=false NRFD=false
E63E- 8D 42 E8.      STA P2DB    ;bits 7-5,0 output, 4-1 input
E641- 8E 22 E8      STX IEEE0   ;IEEE PIA, B section: all bits output
E644- 8E 45 E8      STX T1H     ;VIAT1H: maximum timeout
E647- A9 3D      LDA #$3D      ;keyboard PIA: cassette#1 motor off
E649- 8D 13 E8      STA PIAS    ;IRQ on neg edge on CB1
;(end of screen refresh)

```

E64C- 2C 12 E8	BIT PIAK	;clear keyboard PIA interrupt flags
E64F- A9 3C	LDA #\$3C	;IEEE PIA: NDAC=false, CA1 neg. edge
E651- 8D 21 E8	STA IEEIS	;no interrupts
E654- 8D 23 E8	STA IEEOS	;IEEE PIA: DAV=false, CB1 neg. edge
E657- 8D 11 E8	;no interrupts	
E65A- 8E 22 E8	STA PIAL1	;keyboard PIA: CA2=1,CA1 neg. edge
E65D- A9 0C	;no interrupts	
E65F- 8D 4C E8	STX IEEO	;PIA2B:all IEEE data line false
E662- 85 A8	LDA #\$0C	;VIA: CA2=0 (graphics),CA1 neg. edge
E664- 85 A7	STA PCR	;no interrupts, CB2 input, neg. edge
E666- 85 E4	STA *BLNCT	;set blink count for cursor
E668- 85 E5	STA *BLNSW	;set cursor visible
E66A- 20 D2 E1	STA *RPTFLG	;enable repeat key
E66D- A9 10	STA *KOUNT	;and set delay for repeat
E66F- A2 E1	JSR e1d2	;set window to full screen
E671- 85 E9	LDA #L,e1:d	
E673- 86 EA	LDX #H,e1:d	
E675- A9 0C	STA *\$E9	;set vector for
E677- A2 E2	STX *\$EA	;input from screen
E679- 85 EB	LDA #L,e20c	
E67B- 86 EC	LDX #H,e20c	
E67D- A9 10	STA *\$EB	;set vector for
E67F- 85 E7	STX *\$EC	;output to screen
E681- 20 A4 E6	LDA #\$10	;set up
E684- F0 1E	STA *\$E7	;bell timing.
E686- EA	JSR e6a4	
E687- EA	BEQ e6a4	;and ring bell 4 times
E688- EA	NOP	
E689- EA	NOP	
E68A- EA	NOP	
E68B- EA	NOP	
;*****		
;*		
;* Ring bell if right margin		
;* passed.		
;*		
;*****		
E68C- 20 D2 E2 e68c	JSR PRT	;move character to screen
E68F- AA	TAX	;save it in X
E690- A5 D5	LDA *LNMX	;get right margin
E692- 38	SEC	
E693- E5 C6	SBC *PNTR	;subtract current position
E695- C9 05	CMP #5	;if 5 from margin
E697- D0 37	BNE e6d0	
E699- 8A	TXA	;get character again
E69A- C9 10	CMP #\$1D	;if cursor-RIGHT
E69C- F0 06	BEQ e6a4	;ring bell twice
E69E- 29 7F	AND #\$7F	;remove shift-bit
E6A0- C9 20	CMP #\$20	;if not printable
E6A2- 90 2C	BCC e6d0	;exit, else ring bell twice
;*****		
;*		
;* Ring bell twice.		
;*		
;*****		
E6A4- 20 A7 E6 e6a4	JSR e6a7	;ring bell once
;*****		
;*		
;* Ring bell once.		
;*		
;*****		
E6A7- A4 E7 e6a7	LDY #\$E7	;get bell timing

```

E6A9- F0 25           BEQ e6d0      ;if zero, don't ring bell
E6AB- A9 10           LDA #$10      ;set CB2 to free run
E6AD- 8D 48 E8        STA ACR
E6B0- A9 0F           LDA #$0F      ;and to lowest frequency
E6B2- 8D 4A E8        STA SR
E6B5- A2 07           LDX #7       ;7 tones to play
E6B7- BD 40 E7        e6b7       LDA e74e-1,X ;get tone constant
E6BA- 8D 48 E8        STA T2L      ;put to VIA
E6BD- A5 E7           LDA *$E7      ;get timing
E6BF- 88               DEY
E6C0- D0 FD           BNE e6bf      ;wait, inner loop
E6C2- 38               SEC
E6C3- E9 01           SBC #1
E6C5- D0 F8           BNE e6bf      ;wait, outer loop
E6C7- CA               DEX
E6C8- D0 ED           BNE e6b7      ;and do next tone
E6CA- 8E 4A E8        STX SR       ;then silence the VIA
E6CD- 8E 48 E8        STX ACR      ;and enable cassette operations
E6DD- 60               RTS         ;then exit
.eFI "BASIC4.OEG8.M07"

```

1A55 . 34A7-4EFC BASIC4.OEG8.M07

```

;name BASIC4.OEG8.M07
;*****
;*                                     *
;* Keyboard table, for 80 keys.   *
;*                                     *
;*****
E6D1- 3D 2E FF  CHAR  .BY '=' $FF $03 '<' ' ' $12 ;scan line nine
E6D4- 03 3C 20
E6D7- 5B 12
E6D9- 2D 30 00  .BY '-0' $00 '>' $0E 'J@' $00 ;scan line eight
E6DC- 3E 0E 5D
E6DF- 40 00
E6E1- 2B 32 0F  .BY '+2' $0F '?',NVX' ;scan line seven
E6E4- 3F 2C 4E
E6E7- 56 58
E6E9- 33 31 0F  .BY '31' $0F ' ',MBCZ' ;scan line six
E6EC- 3B 40 42
E6EF- 43 5A
E6F1- 2A 35 16  .BY '*5' '$16 ':KHFS' ;scan line five
E6F4- 3A 4B 48
E6F7- 46 53
E6F9- 36 34 15  .BY '64' $15 'LJGDA' ;scan line four
E6FC- 4C 4A 47
E6FF- 44 41
E701- 2F 38 9B  .BY '/8' $9B 'PIYRW' ;scan line three
E704- 50 49 59
E707- 52 57
E709- 39 37 5E  .BY '97^0UTEQ' ;scan line two
E70C- 4F 55 54
E70F- 45 51
E711- 14 11 09  .BY $14 $11 $09 ')V' $27 '$' ;scan line one
E714- 29 5C 27
E717- 24 22
E719- 10 13 5F  .BY $10 $13 ' (&%#!' ;scan line zero
E71C- 28 26 25
E71F- 23 21
;*****
;*                                     *
;* Message: D shift-L "* RETURN    *
;* RUN RETURN                         *
;
```

```

;* *
;*****RUNTB .BY 'D' $CC '**' $0D 'RUN' $0D
E721- 44 CC 22 RUNTB .BY 'D' $CC '**' $0D 'RUN' $0D
E724- 2A 0D 52
E727- 55 4E 0D
;*****
;* Two tables of 18 constants *
;* each for the CRT-controller. *
;* The first table is for text *
;* mode, the second for graphics *
;* mode. *
;*****
;text mode table
E72A- 31 e72a .BY 49 ;horizontal total
E72B- 28 .BY 40 ;horizontal displayed
E72C- 29 .BY 41 ;horizontal sync position
E72D- 0F .BY 15 ;horizontal sync width
E72E- 27 .BY 39 ;vertical total
E72F- 00 .BY 0 ;vertical total adjust
E730- 19 .BY 25 ;vertical displayed
E731- 20 .BY 32 ;vertical sync position
E732- 00 .BY 0 ;interlace mode (off)
E733- 09 .BY 9 ;maximum scan line address
E734- 00 .BY 0 ;cursor start
E735- 00 .BY 0 ;cursor end
E736- 10 .BY 16 ;start address (hi) (inverts video)
E737- 00 .BY 0 ;start address (lo)
E738- 00 .BY 0 ;cursor (hi)
E739- 00 .BY 0 ;cursor (lo)
E73A- 00 .BY 0 ;light pen (hi)
E73B- 00 .BY 0 ;light pen (lo)

;graphics mode table
E73C- 31 e73c .BY 49 ;horizontal total
E73D- 28 .BY 40 ;horizontal displayed
E73E- 29 .BY 41 ;horizontal sync position
E73F- 0F .BY 15 ;horizontal sync width
E740- 31 .BY 49 ;vertical total
E741- 00 .BY 0 ;vertical total adjust
E742- 19 .BY 25 ;vertical displayed
E743- 25 .BY 37 ;vertical sync position
E744- 00 .BY 0 ;interlace mode (off)
E745- 07 .BY 7 ;maximum scan line address
E746- 00 .BY 0 ;cursor start
E747- 00 .BY 0 ;cursor end
E748- 10 .BY 16 ;start address (hi) (inverts video)
E749- 00 .BY 0 ;start address (lo)
E74A- 00 .BY 0 ;cursor (hi)
E74B- 00 .BY 0 ;cursor (lo)
E74C- 00 .BY 0 ;light pen (hi)
E74D- 00 .BY 0 ;light pen (lo)
;*****
;* *
;* Constants for bell. *
;* *
;*****
E74E- 0E 1E 3E e74e .BY $0E $1E $3E $7E $3E $1E $0E
E751- 7E 3E 1E
E754- 0E
;*****
;* *
;* Table of 10-bytes of screen *
;* line addresses. *

```

```

        ;*
        ;*****
E755- 00 50 A0 LDTB2 .BY 0 80 160 240 64 144 224
E758- F0 40 90
E75B- EO
E75C- 30 80 D0 .BY 48 128 208 32 112 192
E75F- 20 70 C0
E762- 10 60 B0 .BY 16 96 176 0 80 160 240
E765- 00 50 A0
E768- F0
E769- 40 90 EO .BY 64 144 224 48 128
E76C- 30 80
        ;*****
        ;*
        ;* Table of hi-bytes of screen *
        ;* line addresses.           *
        ;*
        ;*****
E76E- 80 80 80 LDTB1 .BY $80 $80 $80 $80
E771- 80
E772- 81 81 81 .BY $81 $81 $81 $82 $82 $82
E775- 82 82 82
E778- 83 83 83 .BY $83 $83 $83 $84 $84 $84
E77B- 84 84 84
E77E- 85 85 85 .BY $85 $85 $85 $85
E781- 85
E782- 86 86 86 .BY $86 $86 $86 $87 $87
E785- 87 87
        ;*****
        ;*
        ;* Copyright statement.      *
        ;*
        ;*****
E787- 20
E788- 41 4C 4C .BY ' '
E78B- 20 41 4C
E78E- 54 45 52
E791- 41 54 49
E794- 4F 4E 53
E797- 3A 20 20
E79A- 20 20 20
E79D- 20 20 20
E7A0- 28 43 29 .BY '(C) 1983 NDV FIRMWARE
E7A3- 20 31 39
E7A6- 38 55 20
E7A9- 4E 44 56
E7AC- 20 46 49
E7AF- 52 4D 57
E7B2- 41 52 45
E7B5- 20 20 20
E7B8- 20 20 20
E7BB- 20 20 20
E7BE- 20 20
        ;*****
        ;*
        ;* The rest of ROM is not used, *
        ;* and may contain anything.   *
        ;*
        ;*****
        ;
        ;*****
        ;*
        ;* External Labels.          *
        ;*

```

```

; ****
CTIMR .DE $0080      ;the jiffy clock
CINV  .DE $0090      ;IRQ RAM vector
CBINV .DE $0092      ;BRK RAM vector
LSTX  .DE $0097      ;last scan index
SHFLAG .DE $0098     ;shiftkey indicator
NDX   .DE $009E      ;number of keys in keyboard buffer
RVS   .DE $009F      ;flag for reverse field printing
INDX  .DE $00A1      ;record length of current screen line
LXSP  .DE $00A3      ;cursor row/column, used in INPUT and G
SFDX  .DE $00A6      ;current scan index
BLNSW .DE $00A7      ;flag: cursor on
BLNCT .DE $00A8      ;countdown till next cursor blink
GDBLN .DE $00A9      ;real character under cursor
BLNDN .DE $00AA      ;cursor's blink phase
CRSW  .DE $00AC      ;flag: input from screen or keyboard
DFLTN .DE $00AF      ;current input device number
DFLTO .DE $00B0      ;current output device number
PNT   .DE $00C4      ;screen line address, lo
POINT .DE $00C5      ;screen line address, hi
PNTR  .DE $00C6      ;current column position of cursor
SAL   .DE $00C7      ;pointer in screen scroll
SAH   .DE $00C8      ;pointer in screen scroll, hi
QTSW  .DE $00CD      ;flag: nonzero if in quote mode
FNLEN .DE $0001      ;temp. work area for CRTC setup
LNMX  .DE $0005      ;length of current screen line
TBLX  .DE $0008      ;current line number of cursor
DATA  .DE $0009      ;last key input
INSRT .DE $000C      ;number of inserts outstanding
XREC  .DE $00E0      ;top line# of window
XWRT  .DE $00E1      ;bottom line# of window
XMAX  .DE $00E3      ;maximum keyboard buffer length
RPTFLG .DE $00E4     ;flag: REPEAT pressed
KOUNT .DE $00E5      ;repeat countdown
DELAY .DE $00E6      ;repeat rate for repeat key
CAS1  .DE $00F9      ;flag: cassette #1 motor on
CAS2  .DE $00FA      ;flag: cassette #2 motor on
PI    .DE $00FF      ;value for PI
KEYD  .DE $026F      ;keyboard buffer
UDTIM .DE $FFEA      ;update jiffy clock
PIAL  .DE $E810      ;keyboard PIA: I/O port A and DDR
PIAL1 .DE $E811      ;keyboard PIA: control register A
PIAK  .DE $E812      ;keyboard PIA: I/O port B and DDR
PIAS  .DE $E813      ;keyboard PIA: control register B
IEEI  .DE $E820      ;IEEE PIA: I/O port A and DDR
IEEIS .DE $E821      ;IEEE PIA: control register A
IEEO  .DE $E822      ;IEEE PIA: I/O port B and DDR
IEEDS .DE $E823      ;IEEE PIA: control register B
PIA   .DE $E840      ;VIA: I/O port B
SYNC  .DE $E841      ;VIA: I/O port A with handshake
P2DB  .DE $E842      ;VIA: port B data direction register
P2DA  .DE $E843      ;VIA: port A data direction register
T1L   .DE $E844      ;VIA: timer 1 lo
T1H   .DE $E845      ;VIA: timer 1 hi
T1LL  .DE $E846      ;VIA: timer 1 lo, latch
T1LH  .DE $E847      ;VIA: timer 1 hi, latch
T2L   .DE $E848      ;VIA: timer 2 lo
T2H   .DE $E849      ;VIA: timer 2 hi
SR    .DE $E84A      ;VIA: shift register
ACR   .DE $E84B      ;VIA: auxiliary control register
PCR   .DE $E84C      ;VIA: peripheral control register
IFR   .DE $E84D      ;VIA: interrupt flag register
IER   .DE $E84E      ;VIA: interrupt enable register
SYNC1 .DE $E84F      ;VIA: port A without handshake

```

CRO	.DE \$E880	;CRTC: address register
CR1	.DE \$E881	;CRTC: register selected through CRO
	.EN	

END MAE PASS

LABELFILE

ACR =E84B	BK1 =E251	BK2 =E25C
BKLN =E1AA	BLNCT =00A8	BLNON =00AA
BLNSW =00A7	CAS1 =00F9	CAS2 =00FA
CBINV =0092	CHAR =E6D1	CINT =E0D0
CINV =0090	CKIS1 =E4E0	CKIT =E4E7
CKIT1 =E4F2	CKUT =E4E6	CLP1 =E15B
CLP2 =E144	CLP21 =E159	CLP2A =E156
CLP5 =EOF2	CLP6 =EOF8	CLP7 =E169
CLSR =E051	CNC3 =E266	CNC3X =E23E
CRO =E880	CR1 =E881	CRSW =00AC
CTIMR =0080	DATA =00D9	DELAY =00E6
DFLTN =00AF	DFLTO =0080	DSPP =E606
FNLEN =00D1	GDBLN =00A9	IEEI =E820
IEEIS =E821	IEE0 =E822	IEEOS =E823
IER =E84E	IFR =E84D	INDX =00A1
INS1 =E320	INS2 =E322	INSRT =00DC
JLP2 =E38C	JPL3 =E299	JSTS =E188
JSTS1 =E192	KEY =E455	KEY3 =E495
KEY5 =E476	KEYD =026F	KL1 =E4D1
KL2 =E4A1	KL22 =E4B8	KL23 =E4AC
KL24 =E49E	KL25 =E4B5	KN1 =E536
KOUNT =00E5	LDTB1 =E76E	LDTB2 =E755
LNMX =0005	LOOP2 =E199	LOOP3 =E0BF
LOOP4 =E0BC	LOOP5 =E116	LOP2 =E1A5
LOPS =E121	LOP52 =E137	LOP53 =E13B
LOP54 =E131	LP1 =E0AC	LP2 =E0A7
LP21 =E0D3	LP22 =E0EA	LP23 =E0DF
LSTX =0097	LXSP =00A3	MLP1 =E3FD
MLP4 =E408	MLP41 =E414	MLP42 =E428
NC1 =E26F	NC2 =E283	NC3 =E170
NC3W =E269	NCX2 =E292	NDX =009E
NELL =E3D0	NEWLN =E3C8	NJTL =E22B
NTCN =E237	NTCN1 =E262	NVS =E17F
NVSL =E185	NXLN =E3A3	NXT2 =E358
NXT3 =E179	NXT33 =E177	NXT6 =E360
NXTD =E05F	NXTL =E3BD	NXTL1 =E3CB
NXTX =E2F4	NXTX1 =E2FC	P2DA =E843
P2DB =E842	PCR =E84C	PI =00FF
PIA =E840	PIAK =E812	PIAL =E810
PIAL1 =E811	PIAS =E813	PNT =00C4
PNTR =00C6	POINT =00C5	PREND =E600
PREND0 =E012	PRT =E202	PULS =E442
PULS1 =E452	QTSW =00CD	QTSWC =E16A
QTSWL =E176	RPTFLG =00E4	RUNTB =E721
RVS =009F	SAH =00C8	SAL =00C7
SCRL5 =E3EB	SCROL =E3E8	SFDX =00A6
SHFLAG =0098	SR =E84A	STUPR =E06C
STUPT =E067	SYNC =E841	SYNC1 =E84F
T1H =E845	T1L =E844	T1LH =E847
T1LL =E846	T2H =E849	T2L =E848
TBLX =0008	UDTIM =FFEA	UP2 =E347
UPS =E30A	UP6 =E342	UP9 =E33E
XMAX =00E3	XREC =00E0	XWRT =00E1
e036 =E036	e04b =E04B	e054 =E054

e063 =E063	e06f =E06F	e07a =E07A
e082 =E082	e088 =E088	e09b =E09B
e11d =E11D	e19d =E19D	e1ba =E1BA
e1c1 =E1C1	e1c3 =E1C3	e1cb =E1CB
e1d2 =E1D2	e1dc =E1DC	e1e1 =E1E1
e20c =E20C	e224 =E224	e24d =E24D
e27b =E27B	e29c =E29C	e2a3 =E2A3
e2b6 =E2B6	e2c5 =E2C5	e2d0 =E2D0
e2d7 =E2D7	e2e0 =E2E0	e2e7 =E2E7
e303 =E303	e36f =E36F	e373 =E373
e37c =E37C	e38f =E38F	e396 =E396
e39a =E39A	e3b1 =E3B1	e3b6 =E3B6
e41d =E41D	e421 =E421	e42a =E42A
e431 =E431	e458 =E458	e468 =E468
e47a =E47A	e4be =E4BE	e4c7 =E4C7
e514 =E514	e522 =E522	e546 =E546
e54c =E54C	e570 =E570	e582 =E582
e590 =E590	e591 =E591	e59b =E59B
e5a7 =E5A7	e5aa =E5AA	e5b3 =E5B3
e5bc =E5BC	e5ca =E5CA	e5d2 =E5D2
e5d9 =E5D9	e5de =E5DE	e5ea =E5EA
e5ed =E5ED	e60f =E60F	e618 =E618
e61f =E61F	e68c =E68C	e6a4 =E6A4
e6a7 =E6A7	e6b7 =E6B7	e6bf =E6BF
e6d0 =E6D0	e72a =E72A	e73c =E73C
e74e =E74E		
//0000,E7C0,E7C0		
]		

```
*****
*      Page 232, I/O area.
*
*****
```

In this page, all I/O chips are situated. Each machine has at least three of these chips:

No. type Part# Use

1. PIA1 6520 Keyboard scanning and some IEEE lines.
2. PIA2 6520 IEEE in- and output.
3. VIA 6522 Cassettes, the parallel user port and some IEEE lines.

Machines with a twelve inch picture tube have an additional chip: the CRT controller chip (shorthand: CRTC). This chip 'manufactures' all signals needed to drive the video display. This chip is a 6845 or an equivalent.

All chips are incompletely decoded, so their registers also appear on other addresses than the ones represented here. Nevertheless, all ROM-software addresses the chips at the addresses mentioned below.

The mirror addresses are restricted to page 232 only (area \$E800-\$E8FF).

For each register, the description and the value after power up are given. The description and value are presented bit by bit where necessary. Binary values are preceded with a percent sign. A value without a preceding character is always in hex, and represents an eight bit byte. In comments a hexadecimal value is preceded by a dollar sign, other values are in decimal there.

```
*****
*      PIA1: addresses $E810, $E811, $E812 and $E813.      *
*****
```

Address	Bit(s)	Value	Dir	Use/Comment
E810	Byte	0F		;Data Direction Register for Port A or ;Peripheral register A (depends on contents ;of bit2 of Control register A at \$E811).
E810	0-3		Out	;Keyboard scan line number. Of the 16 numbers ;possible, only ten are used. Connected ;through a decoder to the keyboard matrix ;column lines.
E810	4		In	;Cassette #1 sense line.
E810	5		In	;Cassette #2 sense line.
E810	6		In	;IEEE: EOI input.
E810	7		In	;Diagnostic sense input. With BASIC 4.0 used ;to determine whether to start BASIC or the ;machine language monitor on power up. ;This input is situated on the user port, ;pin 5.
E811	Byte	3C		;Control register A.
E811	0-1	%00	In	;CA1 input: negative edge active, no IRQ. ;The CA1 line is used as cassette read line ;for cassette #1.
E811	2			;DDR or peripheral port access, set to ;access the peripheral port (after setting ;contents of DDR).
E811	3-5		Out	;CA2 control ; the CA2 line assumes the same ;level as bit 3 of the control register. ;The CA2 line is used as IEEE EOI output. ;9 inch screen models drive this line to ;blank the screen during scroll. (Hardware ;to blank the screen is missing; this is ;carried over from the original 4K/8K PET). ;Interrupt flag for CA2. Always clear, as ;CA2 is used as output. (Read only).
E811	6			;Interrupt flag for CA1. Will be set by a ;negative edge on the CA1 input (cassette #1 ;read). Cleared by a read of this register.
E811	7			;Data Direction Register for Port B or ;Peripheral register B (depends on contents ;of bit2 of Control register B at \$E813).
E812	Byte	00		;Keyboard scan inputs; connected to keyboard ;matrix row lines.
E813	Byte	30		;Control register B.
E813	0-1	%01	In	;CB1 input: negative edge active, interrupt ;enabled. ;The CB1 line is used to sense the end of ;the video display refresh period, causing ;an IRQ interrupt to occur at the end of ;the each screen refresh. To achieve this, ;CB1 is connected to a signal called VIDEO ON ;on 9 inch screen machines; on 12 inch screen ;models (with CRTC) it is connected to ;VERTICAL DRIVE.
E813	2			;DDR or peripheral port access, set to ;access the peripheral port (after setting

			;contents of DDR).
E813	3-5	Out	;CB2 control; the CB2 line assumes the same ;level as bit 3 of the control register. ;The CB2 line is used as switch line for the ;motor of cassette #1. (0=motor on, 1=off).
E813	6		;Interrupt flag for CB2. Always clear, as ;CB2 is used as output. (Read only).
E813	7		;Interrupt flag for CB1. Will be set by a ;negative edge on the CB1 input- (end of ;screen refresh). Cleared by a read of this ;register.

```
*****
*          *
*   PIA2: addresses $E820, $E821, $E822 and $E823.          *
*          *
*****
```

Address	Bit(s)	Value	Dir	Use/Comment
E820	Byte	00		;Data Direction Register for Port A or ;Peripheral register A (depends on contents ;of bit2 of Control register A at \$E811).
E820	Byte		In	;IEEE data input lines. Connected through ;two MC 3446 buffers to the eight IEEE DIO- ;lines.
E821	Byte	3C		;Control register A.
E821	0-1	%00	In	;CA1 input: negative edge active, no IRQ. ;The CA1 line is used as IEEE ATN input, ;which is connected through a buffer to ;the IEEE ATN line.
E821	2	%1		;DDR or peripheral port access, set to ;access the peripheral port (after setting ;contents of DDR).
E821	3-5	%111	Out	;CA2 control; the CA2 line assumes the same ;level as bit 3 of the control register. ;The CA2 line is used as IEEE NDAC output. ;It is connected through a buffer to the ;IEEE NDAC line.
E821	6			;Interrupt flag for CA2. Always clear, as ;CA2 is used as output. (Read only).
E821	7			;Interrupt flag for CA1. Will be set by a ;negative edge on the CA1 input (IEEE ATN).
E822	Byte	FF		;Data Direction Register for Port B or ;Peripheral register B (depends on contents ;of bit2 of Control register B at \$E813).
E822	Byte		Out	;IEEE data output lines. Connected through ;two MC 3446 buffers to the eight IEEE DIO- ;lines.
E823	Byte	3C		;Control register B.
E823	0-1	%00	In	;CB1 input: negative edge active, no IRQ. ;The CB1 line is used as IEEE SRQ input. It ;directly connected to the IEEE connector; no ;buffer is used. It is also connected to the ;user port, at pin 3. This is carried over ;from the original PET which had a diagnostic ;routine to test the line through the ;user port.
E823	2	%1		;DDR or peripheral port access, set to ;access the peripheral port (after setting ;contents of DDR).
E823	3-5	%111	Out	;CB2 control; the CB2 line assumes the same ;level as bit 3 of the control register. ;The CB2 line is used as IEEE DAV output. It ;connected to the IEEE bus through a buffer.
E823	6			;Interrupt flag for CB2. Always clear, as ;CB2 is used as output. (Read only).
E823	7			;Interrupt flag for CB1. Will be set by a ;negative edge on the CB1 input (IEEE SRQ).

* VIA: addresses \$E840 through \$E84F.
* ****

Address	Bit(s)	Value	Dir	Use/Comment
E840	0		In	;In/output register B. This line is used ;as IEEE NDAC input. It is connected through ;a buffer to the IEEE bus.
E840	1	%1	Out	;In/output register B. This line is used ;as IEEE NRFD output. It is connected through ;a buffer to the IEEE bus.
E840	2	%1	Out	;In/output register B. This line is used ;as IEEE ATN output. It is connected through ;a buffer to the IEEE bus.
E840	3	%1	Out	;In/output register B. This line is used ;as cassette write line. It is common for ;both cassettes.
E840	4	%1	Out	;In/output register B. This line is used ;as cassette #2 motor control. (0=motor on, ;1=motor off).
E840	5		In	;In/output register B. Connected to CB1 of ;PIA1 to sense end of screen refresh.
E840	6		In	;In/output register B. This line is used ;as IEEE NRFD input. It is connected through ;a buffer to the IEEE bus.
E840	7		In	;In/output register B. This line is used ;as IEEE DAV input. It is connected through ;a buffer to the IEEE bus.
E841	Byte		In	;In/output register A with handshake. These ;eight lines are the main component of the ;parallel user port. They are connected to ;the pins C through L of the port.
E842	Byte	'1E		;Data direction register for port B.
E843	Byte	00		;Data direction register for port A.
E844	Byte			;Timer1 low order latch (write) or counter (read).
E845	Byte	FF		;Timer1 hi order counter. A write to this ;register will load the lo order counter ;with the contents of the lo order latch and ;start the timer, which will count down with ;a 1 MHz rate (the phase two clock). ;Timer1 counter is used by RND.
E846	Byte	00		;Timer1 low order latch. May be read or writ- ;ten; write is as \$E844, read will return ;the latch value, not the counter value. ;Timer1 is used to time out IEEE handshake, ;and for cassette read. ;9 inch screen models also use it to delay ;screen scroll when appropriate keys are ;pressed.
E847	Byte	00		;Timer2 low order latch (write) or counter (read).
E848	Byte	00		;Timer2 hi order latch. ;Timer2 is used in both cassette reading and ;writing. It is also used to determine tone ;pitch when chiming bell (12" only) or ;playing music through CB2.

E849	Byte	00		;Timer2 hi order counter. ;The timer2 counter is used by RND.
E84A	Byte	00		;Shift register, only used in 12 inch screen ;models to chime the bell. Value of this ;register determines the 'waveform'. ;Also used for playing music through CB2.
E84B	0	%0		;Auxiliary control register. No latching on ;port A.
E84B	1	%0		;No latching on port B.
E84B	2-4	%000		;No shiftregister operation.
	2-4	%100		;Also used in free run, shifting out at ;timer2 rate, for generating music and ;chiming the bell on 12 inch screen models.
E84B	5	%0		;Timer2: timed interrupt.
E84B	6-7	%00		;Timer1: timed interrupt, no influence on ;PB7 line.
E84C	Byte	0C/0E		;Peripheral control register.
E84C	0	%0		;CA1: negative edge is active.
E84C	1-3	%110/%111		;CA2: output, low level. CA2 will assume ;the level indicated by bit 1 of this ;register. The CA2 line controls the current ;character generator in use (lower/upper case ;or upper case(graphics)).
E84C	4	%0		;CB1: negative edge is active.
E84C	5-7	%000		;CB2 control; the CB2 line assumes the same ;level as bit 5 of this register. Default is ;low level.
E84D	Byte			;Interrupt flag register. The bits in this ;register signal what was the cause of an ;interrupt that would occur, if enabled. The ;MSbit signals that any interrupt would occur ;or has occurred. (Logical OR of all other ;bits, easy to test with BIT instructions).
E84E	Byte	7F		;Interrupt Enable register. The MSbit is ;clear, indicating inverted polarity. ;All possible sources of interrupts in the ;VIA are disabled.
E84F	Byte	In		;Port A (parallel user port) without ;handshake.

```
*****
*
*   CRT controller: Addresses $E880 and $E881.          *
*           (Only present in 12 inch screen models).      *
*
*****
```

Address	Reg#	Text mode	Graph. mode	Use/Comment
E880				;Select register. The value in this register ;selects one of the 17 registers in the CRTC ;that can be reached through \$E881 after ;being selected.
E881	00	31	31	;Horizontal total. This register determines ;how many characters a displayed line will ;be long. It therefore determines the line ;scan frequency. With 1MHz character clock, ;this frequency is 1/(50*1uS)=20kHz.
E881	01	28	28	;Horizontal displayed. This register ;determines how many characters are shown ;per line (for 80 column: see note below).
E881	02	29	29	;Horizontal sync. position. This register ;determines the position of the horizontal ;sync. pulse. It adjusts the horizontal posi- ;tion of the display on the screen.
E881	03	0F	0F	;Horizontal sync. width. The length of the ;horizontal sync. pulse is selected here.
E881	04	27	31	;Vertical total. This register selects how ;many lines of characters will be in one ;frame. This number is 40 for text mode, and ;50 for graphics mode.
E881	05	00	00	;Vertical total adjust. This register selects ;how many extra scan lines will be written at ;at the end of a frame. It can be used as a ;fine tune for the frame frequency, e.g. to ;match the mains line frequency.
E881	06	19	19	;Vertical displayed. The number of displayed ;lines of characters is held here. In all ;cases this number is 25.
E881	07	20	25	;Vertical sync. position. With this register ;the display can be centered on the picture ;tube, in the vertical direction. Note the ;difference for graphics and text mode, to ;keep the display correctly centered in both ;modes.
E881	08	00	00	;Interlace mode. This register selects an ;interlaced or non-interlaced display. On ;the CBM's a non interlaced display is used. ;The possibilities with this register vary ;from CRTC type to CRTC type; there are ;simple versions and quite complicated ones. ;cursor mode (not used) is also selected ;here.
E881	09	09	07	;Maximum scan line address. This register ;selects the character height in dots or ;horizontal scan lines. For text mode, 10 ;is used. Because the character generator ;provides only 8 dots, the two lines under ;a character are blank in this mode, impro- ;ving readability. In graphics mode, the

				<p>;value 8 is used, so characters are written ;next to each other. ;Together with registers 0, 4 and 5, this ;register determines the frame scan fre- ;quency: ;$F=1/((R0+1)*CCLK*(R4+1)*(R8+1)+RS*CCLK)$;For text mode, the frequency is: ;$F=1/(50*1uS*40*10+0)=50\text{ Hz}$;For graphics mode, it is: ;$F=1/(50*1uS*50*8+0)=50\text{ Hz}$</p>
E881	0A	00	00	<p>;Cursor start. Although the CRTC can generate ;a cursor by itself, either block or ;underline, blinking or steady, this is not ;used; the cursor is made under software ;control (See interrupt routines in E-ROM).</p>
E881	0B	00	00	<p>;Cursor end. Together with the previous ;register, this register determines the shape ;of the CRTC generated cursor. (Not used).</p>
E881	0C	10	10	<p>;Start address hi. This register holds the ;hi byte at which the scanning of the video ;RAM is started. In CBM machines, The highest ;significant video RAM address is \$03FF. ;(80 column: see note below). This means that ;the lines MA10 through MA13 are not used. ;The lines MA12 and MA13 however, are ;connected in a special way. ;MA12 is connected to an EXOR gate in the ;video signal path, and can control normal or ;inverted video is this way. With the default ;value, MA12 is high, thus inverting the ;video signal. ;MA13 is connected to pin 18 of the character ;generator. This means that a 2nd character ;generator can be selected with this line, ;provided a 2532 EPROM used a character ;generator, with the lower half holding the ;normal generator and the upper half the ;extra generator. Switching can be done by ;storing \$30=48 or \$10=16 in this register.</p>
E881	0D	00	00	<p>;Start address lo. This register holds the ;lo byte at which the scanning of the video ;RAM is started. As can be seen, the RAM is ;scanned from the beginning.</p>
E881	0E	00	00	<p>;Cursor position hi. This register holds the ;hi byte address of the CRTC generated cursor ;(not used).</p>
E881	0F	00	00	<p>;Cursor position lo. This register holds the ;lo byte address of the CRTC generated cursor ;(not used).</p>
E881	10	00	00	<p>;Light pen hi. This register holds the hi ;byte address of the character where a light ;pen strobe occurred. (Not used, light pen ;connection is at memory expansion connector ;J4, pin 21).</p>
E881	11	00	00	<p>;Light pen lo. This register holds the low ;byte address of the character where a light ;pen strobe occurred. (Not used).</p>

Note on 80 column machines:

80 column models use exactly the same CRTC set up as the forty column models. The hardware of these machines is a bit different (jumperable), in such a way, that the video RAM is not a block of 2kbyte as seen by the CRTC but only 1kbyte. 80 column models use the two halves of the phase two clock to display two characters per CCLK (character clock). During the first half of the phase two clock, a character held in the even address RAM is displayed; during the second half of this clock the display address generated by the CRTC does NOT change, but the character held in the odd address video RAM is displayed. This simulates a 2MHz character clock, without having to generate a 2MHz signal.

Wiring of the video RAM is carried out in such a way, that the processor sees a normal block of 2kbyte, although the CRTC addresses only 1kbyte. For further information, see a schematic diagram.

```
*****
*          *
* Pages 233 through 239: Unused area.          *
*          *
```

Address Value Comment

E900	;Area not predetermined for a typical use. May be used by ;replacing the E-ROM (a 2716 or 2316E) by a 2532 EPROM, ;with the lower half holding the original E-ROM contents. ;The upper half of the EPROM may then be used from address ;:\$0900 on.
FFFF	
F000	;Start of operating system ROM area.

```

;NAME BASIC4.OF.CTL
;*****
;*          *
;*      F-ROM for BASIC 4.0      *
;*          *
;* Date 22-01-84  Time 17:55   *
;*          *
;* Made by:                   *
;*          *
;* Nico de Vries             *
;* Mari Andriessenrade 49    *
;* 2907 MA Capelle a/d Yssel *
;* The Netherlands            *
;*          *
;* Original CBM ROM-number   *
;*          *
;*           1465-22           *
;*          *
;*****  

.BA $FO00
.FI "BASIC4.OF.M01"

```

1A54 33B0-4E04 BASIC4.OF.M01

```

;name BASIC4.OF.M01
;*****
;*          *
;* Table of system messages.   *
;*          *
;* Each message has the MSbit *
;* set in the last letter to in- *
;* dicate its end.            *
;*          *
;*****  

MSG1
F000- 54 4F 4F  MS1    .BY 'TOO MANY FILE' $D3
F003- 20 40 41
F006- 4E 59 20
F009- 46 49 4C
F00C- 45 D3
F00E- 46 49 4C  MS2    .BY 'FILE OPE' $CE
F011- 45 20 4F
F014- 50 45 CE
F017- 46 49 4C  MS3    .BY 'FILE NOT OPE' $CE
F01A- 45 20 4E
F01D- 4F 54 20
F020- 4F 50 45
F023- CE
F024- 46 49 4C  MS4    .BY 'FILE NOT FOUN' $C4
F027- 45 20 4E
F02A- 4F 54 20
F02D- 46 4F 55
F030- 4E C4
F032- 0D 53 45  MS5    .BY $00 'SEARCHING' $A0
F035- 41 52 43
F038- 48 49 4E
F03B- 47 A0
F03D- 46 4F 52  MS6    .BY 'FOR' $A0
F040- A0

```

F041- 0D 50 52 MS7 .BY \$0D 'PRESS PLAY' \$A0
 F044- 45 53 53
 F047- 20 50 4C
 F04A- 41 59 A0
 F04D- 26 20 52 MS8 .BY '& RECORD' \$A0
 F050- 45 43 4F
 F053- 52 44 A0
 F056- 4F 4E 20 MS9 .BY 'ON TAPE' \$A3
 F059- 54 41 50
 F05C- 45 20 A3
 F05F- 0D MS10 .BY \$0D
 F060- 4C 4F 41 MS22 .BY 'LOA' \$C4
 F063- C4
 F064- 0D 57 52 MS11 .BY \$0D 'WRIT'
 F067- 49 54
 F069- 49 4E 47 f069 .BY 'ING' \$A0
 F06C- A0
 F06D- 0D MS21 .BY \$0D
 F06E- 56 45 52 MS12 .BY 'VERIF' \$09
 F071- 49 46 09
 F074- 44 45 56 MS13 .BY 'DEVICE NOT PRESEN' \$D4
 F077- 49 43 45
 F07A- 20 4E 4F
 F07D- 54 20 50
 F080- 52 45 53
 F083- 45 4E 04
 F086- 4E 4F 54 MS15 .BY 'NOT INPUT FIL' \$C5
 F089- 20 49 4E
 F08C- 50 55 54
 F08F- 20 46 49
 F092- 4C C5
 F094- 4E 4F 54 MS16 .BY 'NOT OUTPUT FIL' \$C5
 F097- 20 4F 55
 F09A- 54 50 55
 F09D- 54 20 46
 FOA0- 49 4C C5
 FOA3- 0D 46 4F MS17 .BY \$0D 'FOUND' \$A0
 FOA6- 55 4E 44
 FOA9- A0
 FOAA- 0D 4F 4B MS18 .BY \$0D 'OK' \$8D
 FOAD- 8D
 FOAE- 0D 52 45 MS19 .BY \$0D 'READY.' \$8D
 F0B1- 41 44 59
 F0B4- 2E 8D
 F0B6- 0D 41 52 MS30 .BY \$0D 'ARE YOU SURE' \$BF
 F0B9- 45 20 59
 F0BC- 4F 55 20
 F0BF- 53 55 52
 F0C2- 45 20 BF
 F0C5- 0D 3F 20 MS31 .BY \$0D '? BAD DISK' \$8D
 F0C8- 42 41 44
 F0CB- 20 44 49
 FOCE- 53 4B 20
 F0D1- 8D
 ;*****
 ;* *
 ;* Send TALK to IEEE bus. *
 ;* *
 ;*****
 F0D2- A9 40 LDA #\$40 ;get code for TALK
 F0D4- 2C .BY \$2C ;skip next instruction
 ;*****
 ;* *
 ;* Send LISTEN to IEEE bus. *

```

;* *
;*****LISTEN*****
F005- A9 20 LISTN LDA #$20      ;get code for LISTEN
F007- 48      LIST1 PHA          ;save it
F008- AD 40 E8 LDA PIA        ;get NRFD out
F00B- 09 02    ORA #$02       ;set false
F00D- 80 40 E8 STA PIA        ;and send it
F00E- A9 3C    LDA #$3C       ;set
F00E- 80 21 E8 STA IEEIS      ;NDAC false
F00E- 24 A0    BIT *C3PO      ;if byte in buffer
F00E- F0 11    BEQ LIST3      ;set
F00E- A9 34    LDA #$34       ;EOI true
F00E- 80 11 E8 STA PIAL1     ;send the buffered byte
F00E- 20 09 F1    JSR ISOUR    ;move the command byte to buffer
F00F- A9 00    LDA #$00       ;and flag
F00F- 85 A0    STA *C3PO      ;buffer empty
F00F- A9 3C    LDA #$3C       ;then set
F00F- 80 11 E8 STA PIAL1     ;EOI false again
F00F- 68      LIST3 PLA        ;get code back
F00F- 05 D4    ORA *FA        ;add device number
F00F- 85 A5    STA *BSOUR     ;move the command byte to buffer
F00F- AD 40 E8 LIST4 LDA PIA      ;get DAV input
F102- 10 FB    BPL LIST4      ;and wait until false
F104- 29 FB    AND #$FB      ;then set ATN
F106- 80 40 E8 STA PIA        ;true, and:
;*****IEEE Bus*****
;* Send the byte in $A5 to the IEEE bus.
;*
;*****IEEOS*****
F109- A9 3C    ISOUR LDA #$3C      ;set
F10B- 80 23 E8 STA IEEOS     ;DAV false
F10E- AD 40 E8 LDA PIA        ;get NRFD and NDAC
F111- 29 41    AND #$41       ;input bits
F113- C9 41    CMP #$41       ;if both false
F115- F0 55    BEQ ERRP7      ;flag error in ST and exit
F117- A5 A5    LDA *BSOUR     ;else get buffered byte
F119- 49 FF    EOR #$FF       ;set to IEEE polarity
F11B- 80 22 E8 STA IEEO       ;and move it to the bus
F11E- 2C 40 E8 ISR1  BIT PIA      ;then wait
F121- 50 FB    BVC ISR1      ;until NRFD true
F123- A9 34    LDA #$34       ;set
F125- 80 23 E8 STA IEEOS     ;DAV true
F128- A9 FF    ISR0  LDA #$FF      ;and start
F12A- 80 45 E8 STA T1H        ;VIA timer1 (T=65.28ms)
F12D- AD 40 E8 ISR2  LDA PIA      ;get code back
F130- 2C 40 E8 BIT IFR        ;if timed out
F133- 70 1C    BVS ERRS3      ;test for defeat and STOP key
F135- 4A      LSR A          ;else wait
F136- 90 F5    BCC ISR2      ;until NDAC false
F138- A9 3C    ISR3  LDA #$3C      ;if false, set
F13A- 80 23 E8 STA IEEOS     ;DAV false
F13D- A9 FF    LDA #$FF       ;and set zero data
F13F- 80 22 E8 STA IEEO       ;on the IEEE bus
F142- 60      RTS           ;then exit
;*****IEEE Bus*****
;* Send the byte in A to the IEEE bus.
;* Set ATN false after the byte is sent.
;*
;*****IEEOS*****

```

```

F143- 85 A5      SECND   STA *BSOUR      ;store the byte
F145- 20 09 F1      JSR ISOUR      ;send it to the bus
;*****
;*                                     *
;* Set ATN false.                  *
;*                                     *
;*****                                     *
F148- AD 40 E8      SCATN   LDA PIA       ;get ATN
F148- 09 04          ORA #$04      ;set it false
F14D- 80 40 E8          STA PIA      ;and move to the bus
F150- 60          RTS        ;then exit
;*****
;*                                     *
;* Timer timed out on write;      *
;* test if time out defeated.    *
;* If so, test for STOP. If STOP *
;* pressed, stop, else loop.     *
;*                                     *
;*****                                     *
F151- AD FC 03      ERRS3   LDA BOB       ;get defeat flag
F154- 10 0F          BPL ERRPO      ;if no defeat, exit with ST=1
F156- 20 43 F3          JSR STOP      ;else test for STOP key
F159- D0 CD          BNE ISRO      ;if not pressed, loop
;*****
;*                                     *
;* Timer timed out on read;      *
;* test if time out defeated.    *
;* If so, test for STOP. If STOP *
;* pressed, stop, else loop.     *
;*                                     *
;*****                                     *
F15B- AD FC 03      ERRS4   LDA BOB       ;get defeat flag
F15E- 10 10          BPL ERRP1      ;if no defeat, exit with ST=2
F160- 20 43 F3          JSR STOP      ;else test for STOP key
F163- D0 68          BNE ACP00     ;if not pressed, loop
;*****
;*                                     *
;* Exit with ST set to time out  *
;* on write.                      *
;*                                     *
;*****                                     *
F165- A9 01          ERRPO   LDA #$01      ;flag time out on write
F167- 20 C4 FB      ERRO1   JSR UDST      ;in ST byte
F16A- D0 CC          BNE ISR3      ;and exit output
;*****
;*                                     *
;* Exit with ST set to device    *
;* not present.                  *
;*                                     *
;*****                                     *
F16C- A9 80          ERRP7   LDA #$80      ;flag device not present
F16E- 30 F7          ERRO1   BMI ERRO1      ;in ST byte and exit
;*****
;*                                     *
;* Exit with ST set to time out  *
;* on read.                      *
;*                                     *
;*****                                     *
F170- A9 02          ERRP1   LDA #$02      ;flag time out on read
F172- 20 C4 FB          JSR UDST      ;in ST byte
;*****
;*                                     *
;* Clear IEEE control lines.    *
;*                                     *

```

```

;*****
F175- AD 40 E8 ER001 LDA PIA
F178- 29 FD AND #$FD ;NRFD true
F17A- 8D 40 E8 STA PIA
F17D- A9 34 LDA #$34 ;set
F17F- 8D 21 E8 STA IEEEIS ;NDAC true
F182- A9 00 LDA #$0D ;get a carriage return
F184- 60 RTS ;and return that
;*****
;* *
;* Print a message from table at *
;* $FOOO on screen. On entry, Y *
;* holds offset in table. *
;* *
;*****
F185- B9 00 F0 MSG LDA MSG1,Y ;get message byte
F188- 08 PHP ;save the flags
F189- 29 7F AND #$7F ;clear MSbit
F18B- 20 02 E2 JSR PRT ;print it on screen
F18E- C8 INY ;point to next character
F18F- 28 PLP ;get the flags back
F190- 10 F3 BPL MSG ;if MSbit was clear, do next
F192- 60 RTS ;else exit
;*****
;* *
;* Send the byte in A to the *
;* IEEE bus, then set NDAC true. *
;* *
;*****
F193- 85 A5 TKSA STA *BSOUR ;save the byte
F195- 20 09 F1 JSR ISOURL ;send it to the bus
F198- 20 75 F1 TKATN JSR ER001 ;set NDAC and NRFD true
F19B- 4C 48 F1 JMP SCATN ;and set ATN false
;BUG: Correct sequence is: ATN false, NRFD true
; NDAC true.
Sequence used is: NRFD true, NDAC true
ATN false.
CBM equipment only works allright with
the INCORRECT sequence; however official
; IEEE-488/IEC-625 equipment may give trouble!
.FI "BASIC4.OF.M02"

```

1B9C 33B0-4F4C BASIC4.OF.M02

```

;name BASIC4.OF.M02
;*****
;*
;* Move the byte in A to the *
;* buffer. If buffer filled al-
;* ready, send buffer first. *
;*
;*****
F19E- 24 A0 CIOUT BIT *C3PO ;if a byte in the buffer
F1A0- 30 04 BMI CI2 ;send it and move A to buffer
F1A2- C6 A0 DEC *C3PO ;else flag a byte in buffer
F1A4- D0 05 BNE CI4 ;and exit
F1A6- 48 CI2 PHA ;save current byte
F1A7- 20 09 F1 JSR ISOURL ;send byte in buffer
F1AA- 68 PLA ;reget current byte
F1AB- 85 A5 CI4 STA *BSOUR ;move to buffer
F1AD- 60 RTS ;and exit
;*****
;* *

```

```

;* Send UNTALK to IEEE bus.      *
;*                                *
;*****  

F1AE- AD 40 E8 UNTLK LDA PIA          ;get ATN out
F1B1- 29 FB   AND #$FE.        ;set it true
F1B3- 8D 40 E8 STA PIA          ;and put on bus
F1B6- A9 5F   LDA #$5F          ;get code for UNTALK
F1B8- 2C     .BY $2C          ;skip next instruction
;*****  

;*                                *
;* Send UNLISTEN to IEEE bus.    *
;*                                *
;*****  

F1B9- A9 3F   UNLSN  LDA #$3F          ;get code for UNLISTEN
F1B8- 20 07 F0   JSR LIST1        ;send as command byte
F1BE- D0 88   BNE SCATN        ;and set ATN false, exit
;*****  

;*                                *
;* Get one character from the   *
;* IEEE bus in A.                *
;*                                *
;*****  

F1C0- A9 34   ACPTR  LDA #$34          ;set NDAC true
F1C2- 8D 21 E8 STA IEEIS
F1C5- AD 40 E8 LDA PIA          ;get NRFD
F1C8- 09 02   ORA #$02          ;set it false
F1CA- 8D 40 E8 STA PIA          ;on the bus
F1CD- A9 FF   ACP00  LDA #$FF          ;get timer value
F1CF- 8D 45 E8 STA T1H           ;in VIA timer1 (T=65.28ms)
F1D2- 2C 40 E8 ACP01  BIT IFR           ;if timed out
F1D5- 70 84   BVS ERRS4        ;test for defeat and STOP
F1D7- 2C 40 E8 BIT PIA           ;else get DAV in
F1DA- 30 F6   BMI ACP01        ;if not true, wait for it
F1DC- AD 40 E8 LDA PIA           ;if true, set NRFD
F1DF- 29 FD   AND #$FD          ;true
F1E1- 8D 40 E8 STA PIA           ;on the bus
F1E4- 2C 10 E8 BIT PIAL         ;get EOI in
F1E7- 70 05   BVS ACP03        ;if true
F1E9- A9 40   LDA #$40          ;flag EOI
F1EB- 20 C4 FB   JSR UDST        ;in ST byte
F1EE- AD 20 E8 ACP03  LDA IEEI        ;get byte from bus
F1F1- 49 FF   EOR #$FF          ;in correct polarity
F1F3- 48     PHA               ;save it
F1F4- A9 3C   LDA #$3C          ;set
F1F6- 8D 21 E8 STA IEEIS        ;NDAC false
F1F9- 2C 40 E8 ACP05  BIT PIA        ;then wait
F1FC- 10 FB   BPL ACP05        ;until DAV-in false
F1FE- A9 34   LDA #$34          ;then set
F200- 8D 21 E8 STA IEEIS        ;NDAC true again
F203- 68     PLA               ;reget the byte
F204- 60     RTS               ;and exit
;*****  

;*                                *
;* Get one byte in A.            *
;*                                *
;* If from keyboard, don't wait *
;* if buffer empty.              *
;*                                *
;* (Called from Kernal: $FFE4).  *
;*                                *
;*****  

F205- A9 00   GETIN  LDA #$00          ;clear
F207- 85 96   STA *CSTAT        ;status byte ST
F209- A5 AF   LDA *DFLTN        ;get input device number

```

```

F20B- D0 17      BNE BN10          ;if not keyboard, check for others
F20D- A5 9E      LDA *NDX          ;get number of keys in buffer
F20F- F0 51      BEQ BN32          ;if none, exit
F211- 78         SEI              ;else disable interrupts
F212- 4C A7 E0   JMP LP2           ;and get a byte from keyboard buffer.
;*****
;*
;* Get one byte in A.          *
;*
;* If no byte in buffer, wait *
;* for one.                   *
;*
;* (Called from Kernal: $FFCF). *
;*
;*****
F215- A5 AF      BASIN            LDA *DFLTN        ;get current input device #
F217- D0 0B      BNE BN10          ;if input from keyboard
;*****
;*
;* Get one byte from keyboard. *
;*
;*****
F219- A5 C6      LDA *PNTR          ;get cursor position
F21B- 85 A4      STA *LXSP+1       ;copy it
F21D- A5 D8      LDA *TBLX          ;get line# of cursor
F21F- 85 A3      STA *LXSP          ;copy also
F221- 4C 16 E1   JMP LOOPS         ;and get input from keyboard
F224- C9 03      CMP #3            ;if input from screen
F226- D0 09      BNE BN20          ;*****
;*
;* Get one byte from screen.   *
;*
;*****
F228- 85 AC      STA *CRSW          ;flag screen input
F22A- A5 D5      LDA *LNMX          ;get current line length (or right marg
F22C- 85 A1      STA *INDX          ;save it
F22E- 4C 16 E1   JMP LOOPS         ;and get input from screen '
F231- B0 29      BCS BN30          ;if input from IEEE, go
F233- 86 AD      STX *XSAV          ;else save X (input from cassette)
F235- 20 49 F2   JSR JTGET         ;get byte from cassette
F238- 48         PHA              ;save it
F239- 20 49 F2   JSR JTGET         ;get another
F23C- D0 05      BNE JTG35         ;if buffer empty (EOI)
F23E- A9 40      LDA #$40          ;flag
F240- 20 C4 FB   JSR UDST          ;end of file in ST
F243- D6 BA      DEC *BUFPNT-1,X ;adapt cassette buffer byte count
F245- A6 AD      LDX *XSAV          ;reget X
F247- 68         PLA              ;and the byte
F248- 60         RTS              ;and exit
;*****
;*
;* Get one byte from cassette. *
;*
;*****
F249- 20 4B F8   JTGET            JSR JTP20         ;increment tape buffer pointer
F24C- D0 0B      BNE JTG10         ;if at end of buffer
F24E- 20 9A F8   JSR RBLK          ;read next block from tape
F251- A6 D4      LDX *FA            ;get current tape#
F253- A9 00      LDA #$00          ;set byte count
F255- 95 BA      STA *BUFPNT-1,X ;of current buffer to zero.
F257- F0 F0      BEQ JTGET         ;and reget character
F259- B1 D6      LDA (TBUF),Y    ;get character from buffer
F25B- 60         RTS              ;and exit

```

```

;*****  

;*  

;* Get one byte from IEEE.  

;*  

;*****  

F25C- A5 96      BN30    LDA *CSTAT      ;get ST byte  

F25E- F0 03      BEQ BN35      ;if error or EOI  

F260- A9 0D      LDA #$00      ;get a carriage return  

F262- 60         BN32    RTS          ;and exit  

F263- 4C C0 F1   BN35    JMP ACPTR     ;else get byte from IEEE  

;*****  

;*  

;* Output the character in A.  *  

;*  

;* (Called from Kernal: $FFD2).  *  

;*  

;*****  

F266- 48         BSOUT   PHA          ;save the character  

F267- A5 B0      LDA *DFLTO     ;get current output device#  

F269- C9 03      CMP #3        ;if screen  

F26B- D0 04      BNE B010     ;  

;*****  

;*  

;* Output A to screen.  *  

;*  

;*****  

F26D- 68         PLA      ;reget character  

F26E- 4C 02 E2   JMP PRT      ;and print on screen  

;*****  

;*  

;* Output A to IEEE.  *  

;*  

;*****  

F271- 30 04      B010    BMI B020      ;if not to cassette.  

F273- 68         PLA      ;reget character  

F274- 4C 9E F1   JMP CIOUT     ;and put to IEEE  

;*****  

;*  

;* Output A to cassette.  *  

;*  

;*****  

F277- 68         B020    PLA          ;reget character  

F278- 85 B4      B021    STA *T1       ;save it  

F27A- C9 0A      CMP #$0A      ;if character is line feed  

F27C- F0 E4      BEQ BN32      ;exit (skip character)  

;BUG: impossible to write the LF character  

;      to cassette. This test is not needed with  

;      with BASIC 4.0, as files with file numbers  

;      below 128 no longer send a LF after CR.  

F27E- 48         PHA      ;else save character  

F27F- 8A         TXA      ;and  

F280- 48         PHA      ;X  

F281- 98         TYA      ;and  

F282- 48         PHA      ;Y  

F283- 20 4B F8   JSR JTP20     ;increment tape buffer pointer  

F286- D0 10      BNE JTP10     ;if at end  

F288- 20 CB F8   JSR WBLK     ;write buffer to cassette  

F28B- A6 D4      LDX *FA       ;get current cassette number  

F28D- A9 01      LDA #1       ;flag one byte in  

F28F- 95 BA      STA *BUFPNT-1,X ;in current cassette buffer  

F291- A0 00      LDY #0       ;get offset zero  

F293- A9 02      LDA #2       ;and file block flag  

F295- 91 D6      STA (TBUF),Y ;move to cassette buffer  

F297- C8         INY      ;adapt offset

```

```

F298- A5 B4      JTP10  LDA *T1          ;reget character
F29A- 91 D6      STA (TBUF),Y    ;move to buffer
F29C- 68         RSTOR  PLA           ;get
F29D- A8         TAY             ;Y back
F29E- 68         PLA           ;and
F29F- AA         TAX            ;X
F2A0- 68         PLA           ;and A
F2A1- 60         RTS            ;then exit
.FI "BASIC4.OF.M03"

```

16F7 33B0-4AA7 BASIC4.OF.M03

```

;name BASIC4.OF.M03
;*****
;*
;* Abort all files and restore   *
;* default I/O.                 *
;*
;*****
F2A2- A9 00      CLALL  LDA #0          ;set current of open files
F2A4- 85 AE      STA *LDTND       ;to zero (aborts all files)
;*****
;*
;* Restore default I/O.          *
;*
;* (Called from Kernal: $FFCC).  *
;*
;*****
CLRCH
F2A6- A5 B0      CLRCHN LDA *DFLTO     ;get current output device#
F2A8- C9 04      CMP #4          ;if an IEEE device
F2AA- 90 03      BCC JX750
F2AC- 20 B9 F1      JSR UNLSN      ;UNLISTEN that device
F2AF- A5 AF      JX750  LDA *DFLTN     ;get current input device#
F2B1- C9 04      CMP #4          ;if an IEEE device
F2B3- 90 03      BCC JX770
F2B5- 20 AE F1      JSR UNTLK      ;UNTALK it
;*****
;*
;* Restore default I/O.          *
;*
;*****
F2B8- A9 03      JX770  LDA #3          ;get default output device#
F2BA- 85 B0      STA *DFLTO     ;as current output device
F2BC- A9 00      LDA #0          ;get default input device#
F2BE- 85 AF      STA *DFLTN     ;as current input device
F2C0- 60         RTS            ;and exit
;*****
;*
;* Search table of open files   *
;* for the file number in A.    *
;* If no such file, exits with *
;* Z clear.                    *
;* If file found, X holds offset *
;* in file entry table and Z is *
;* set on exit.                 *
;*
;*****
F2C1- A6 AE      JLTLK  LDX *LDTND    ;get number of open files
F2C3- CA          JX600  DEX           ;decrement counter
F2C4- 30 16      BMI JZ101      ;if all files searched, exit
F2C6- DD 51 02      CMP LAT,X    ;else compare with file number
F2C9- F0 11      BEQ JZ101      ;if equal, exit

```

F2CB- DO F6	BNE JX600 ;if not, loop ;***** ;* * ;* Set file data up from entry * ;* table. On entry, X must hold * ;* offset in that table. * ;* * ;*****
F2CD- B0 51 02 JZ100	LDA LAT,X ;get file number STA *LA ;in current file number LDA FAT,X ;get device number STA *FA ;in current device number LDA SAT,X ;get secondary address STA *SA ;in current secondary address RTS ;then exit ;***** ;* * ;* Perform CLOSE. * ;* * ;* (Called from Kernal: \$FFC3). * ;* * ;*****
F2D0- 85 D2	
F2D2- B0 5B 02	
F2D5- 85 D4	
F2D7- B0 65 02	
F2DA- 85 D3	
F2DC- 60 JZ101	
F2D0- 20 00 F5 CLOSE	JSR PARS2 ;get parameters LDA *LA ;get current file number FCLOSE
F2E2- 20 C1 F2 CLOSS	JSR JLTLK ;search table of open files
F2E5- D0 40	BNE JX170 ;if not found, exit
F2E7- 20 CD F2 CLOS10	JSR JZ100 ;else set up parameters
F2EA- 8A	TXA ;save index in table
F2EB- 48	PHA ;on stack
F2EC- A5 D4	LDA *FA ;get device number
F2EE- F0 28	BEQ JX150 ;if keyboard, move table entries
F2F0- C9 D3	CMP #3 ;if screen
F2F2- F0 24	BEQ JX150 ;move table entries
F2F4- B0 1F	BCS JX120 ;if IEEE, close (disk) file
F2F6- A5 D3	LDA *SA ;else cassette, get secondary address
F2F8- 29 DF	AND #\$0F ;get address only
F2FA- F0 1C	BEQ JX150 ;if file opened for read, move entries
F2FC- 20 95 F6	JSR f695 ;else get pointer to current cassette b
F2FF- A9 00	LDA #0 ;get closing byte
F301- 20 78 F2	JSR B021 ;move to cassette
F304- 20 CB F8	JSR WBLK ;write last block
F307- A5 D3	LDA *SA ;get secondary address again
F309- C9 62	CMP #\$62 ;if write EOT block
F30B- D0 0B	BNE JX150
F30D- A9 05	LDA #5 ;get EOT block code
F30F- 20 19 F6	JSR TAPEH ;write EOT block
F312- 4C 18 F3	JMP JX150 ;and move table entries
F315- 20 2F F7	JX120 JSR CLSEI ;clear IEEE channel
F318- 68	JX150 PLA ;get index
F319- AA	TAX ;back in X
F31A- C6 AE	DEC *LDTND ;adapt number of open files
F31C- E4 AE	CPX *LDTND ;if last file in table
F31E- F0 14	BEQ JX170 ;exit
F320- A4 AE	LDY *LDTND ;else get table length
F322- B9 51 02	LDA LAT,Y ;get last entry (file#)
F325- 9D 51 02	STA LAT,X ;into closed file entry
F328- B9 5B 02	LDA FAT,Y ;get last entry (dev.#)
F32B- 9D 5B 02	STA FAT,X ;into closed file entry
F32E- B9 65 02	LDA SAT,Y ;get last entry (sec. address)
F331- 9D 65 02	STA SAT,X ;into closed file entry
F334- 60 JX170	RTS ;and exit ;*****

```

;* Test if STOP-key pressed. If    *
;* so, exits with Z set and I/O   *
;* set to default. Else Z is     *
;* clear.                         *
;*                                *
;*****  

F335- A5 9B      STOP1  LDA *STKEY      ;get key image of last scan
F337- C9 EF      CMP #$EF        ;if image for STOP-key
F339- D0 07      BNE STOP2
F33B- 08          PHP             ;save the flags
F33C- 20 A6 F2      JSR CLRCHN    ;restore default I/O
F33F- 85 9E      STA *NDX       ;cancel keyboard buffer
F341- 28          PLP             ;get flags back
F342- 60      STOP2  RTS             ;and exit
;*****  

;*                                *
;* Test if STOP-key pressed. If    *
;* pressed, perform STOP, else    *
;* return to caller.              *
;*                                *
;* (Called from Kernal: $FFE1).   *
;*                                *
;*****  

F343- 20 35 F3      STOP   JSR STOP1      ;test for STOP-key
F346- 4C C6 B7      JMP stop       ;if pressed, perform STOP, or return
;*****  

;*                                *
;* Print message from table at   *
;* $FOOO if in direct mode.      *
;*                                *
;*****  

F349- 20 51 F3      SPMMSG JSR TXTST      ;test if in direct mode.
F34C- D0 F4      BNE STOP2      ;if not, exit
F34E- 4C 85 F1      JMP MSG        ;else print message from table
;*****  

;*                                *
;* Test if in direct mode.        *
;*                                *
;*****  

F351- A5 78      TXTST  LDA *TXTPTR+1  ;get CHRGET's pointer hi
F353- C9 02      CMP #H,BUF      ;if in direct mode, set Z
F355- 60      TXTRT  RTS             ;and exit
;*****  

;*                                *
;* LOAD subroutine: load or      *
;* verify RAM from device indi-  *
;* cated by $D4.                  *
;*                                *
;*****  

F356- A5 D4      LD15   LDA *FA        ;get current device number
F358- D0 03      BNE f35d      ;if LOAD from keyboard
F35A- 4C 00 BF      LD20  JMP SNERR    ;SYNTAX ERROR
F35D- C9 03      f35d   CMP #3        ;if LOAD from screen,
F35F- F0 F9      BEQ LD20      ;also SYNTAX ERROR
F361- 90 71      BCC LD100     ;if LOAD from cassette, do those
                               .FI "BASIC4.OF.M04"  


```

1876 33B0-4C26 BASIC4.OF.M04

```

;name BASIC4.OF.M04
;*****  

;*                                *
;* LOAD from an IEEE device.    *
;*****  


```

```

        ;*          *
;*****  

F363- A9 60      f363    LDA #$60      ;get secondary address for load  

F365- 85 D3      STA *SA       ;in current secondary address  

F367- A4 D1      LDY *FNLEN   ;get length of filename  

F369- D0 D3      BNE f36e    ;if no name entered  

F36B- 4C 00 BF    JMP SNERR   ;SYNTAX ERROR  

F36E- 20 49 F4    f36e    JSR LD300   ;perform OPEN for IEEE  

F371- 20 A5 F4    JSR OPENI   ;print SEARCHING FOR filename  

F374- 20 D2 F0    JSR TALK    ;send TALK  

F377- A5 D3      LDA *SA       ;send  

F379- 20 93 F1    JSR TKSA    ;secondary address  

F37C- 20 C0 F1    JSR ACPTR   ;get one byte from IEEE  

F37F- 85 FB      STA *STAL    ;store start address lo  

F381- A5 96      LDA *CSTAT   ;get status byte  

F383- 4A          LSR A       ;  

F384- 4A          LSR A       ;if timed out on read  

F385- B0 3A      BCS f3c1    ;print FILE NOT FOUND  

F387- 20 C0 F1    JSR ACPTR   ;else get start address hi  

F38A- 85 FC      STA *STAH    ;in load pointer  

F38C- 20 6D F4    JSR LD410   ;print LOADING  

F38F- A9 FD      LD40    LDA #$FD    ;clear bits for time out  

F391- 25 96      AND *CSTAT   ;in status  

F393- 85 96      STA *CSTAT   ;byte  

F395- 20 43 F3    JSR STOP    ;test for STOP-key  

F398- 20 C0 F1    JSR ACPTR   ;get program byte  

F39B- AA          TAX         ;in X  

F39C- A5 96      LDA *CSTAT   ;if status shows  

F39E- 4A          LSR A       ;time out on read  

F39F- 4A          LSR A       ;retry  

F3A0- B0 ED      BCS LD40    ;else get byte back  

F3A2- 8A          TXA         ;get load/verify flag  

F3A3- A4 90      LDY *VERCK   ;if loading, store byte  

F3A5- F0 0C      BEQ LD50    ;else get offset  

F3A7- 88          DEY         ;and compare with memory contents  

F3A8- D1 FB      CMP (STAL),Y ;if equal, increment pointer and loop  

F3AA- F0 09      BEQ LD60    ;else flag read error  

F3AC- A9 10      LDA #$10    ;in  

F3AE- 05 96      ORA *CSTAT   ;ST byte  

F3B0- 85 96      STA *CSTAT   ;skip next instruction  

F3B2- 2C          .BY $2C    ;store byte in memory  

F3B3- 91 FB      LD50    STA (STAL),Y ;increment load pointer lo  

F3B5- E6 FB      LD60    INC *STAL   ;if page crossed  

F3B7- D0 02      BNE LD64    ;adapt hi byte  

F3B9- E6 FC      INC *STAH   ;if EOI found  

F3BB- 24 96      LD64    BIT *CSTAT ;move pointer  

F3BD- 70 07      BVS LD65    ;else get next byte  

F3BF- 50 CE      BVC LD40    ;point to FILE NOT FOUND  

F3C1- A0 24      f3c1    LDY #MS4-MSG1 ;and give that error  

F3C3- 4C AF F5    JMP ERMSSG  ;get last program address+1 lo  

F3C6- A5 FB      LD65    LDA *STAL   ;in end address lo  

F3C8- 85 C9      STA *EAL    ;do the same  

F3CA- A5 FC      LDA *STAH   ;for the hi byte  

F3CC- 85 CA      STA *EAH    ;UNTALK IEEE  

F3CE- 20 AE F1    JSR UNTLK   ;clear IEEE channel  

F3D1- 4C 2F F7    JMP CLSEI   ;  

;*****  

;*          *  

;* LOAD from cassette.      *  

;*          *  

;*****  

F3D4- 20 95 F6      LD100   JSR f695    ;set pointer to current cassette buffer  

F3D7- 20 57 F8      JSR CSTECL  ;do PRESS PLAY...  

F3DA- 20 49 F4      JSR LD300   ;print SEARCHING (FOR filename)

```

```

F300- A5 D1      LD112  LDA *FNLEN      ;get length of filename
F30F- F0 08      BEQ LD150      ;if, name given
F3E1- 20 D3 F4      JSR FAF       ;search tape for named header
F3E4- D0 08      BNE LD170      ;if EOT found
F3E6- 4C AD F5      LD120  JMP OP160      ;print FILE NOT FOUND, exit
F3E9- 20 E5 F5      LD150  JSR FAH       ;no name, get next header
F3EC- F0 F8      BEQ LD120      ;if none, FILE NOT FOUND
F3EE- E0 01      LD170  CPX #1       ;if no program header
F3F0- D0 EB      BNE LD112      ;get next
F3F2- A5 96      LDA *CSTAT      ;else get status byte
F3F4- 29 10      AND #$10      ;get bit for read error
F3F6- D0 74      BNE LD115      ;if error, exit
F3F8- 20 7B F6      JSR LOAD2      ;else get addresses from header
F3FB- 20 60 F4      JSR LD410      ;print LOADING
F3FE- 4C A3 F8      JMP TRD       ;and get prg from tape
;*****
;*          *
;* Perform LOAD.      *
;*          *
;* (Called from Kernal: $FFDS).  *
;*          *
;*****


F401- A9 00      LOAD   LDA #0       ;set load/verify flag
F403- 85 9D      STA *VERCK     ;to load
F405- 20 7D F4      LD10   JSR PARS1     ;get parameters (VERIFY entry)
F408- 20 CC F6      LD10   JSR SV60      ;move PRG pointers to SAVE pointers
F40B- A9 FF      LOADNP LDA #$FF      ;get image for no key (LOAD entry)
F40D- C5 9B      LD11   CMP *STKEY     ;and wait
F40F- D0 FC      BNE LD11       ;until no key pressed
F411- C5 9B      CMP *STKEY     ;with debounce
F413- D0 F8      BNE LD11
F415- 20 56 F3      JSR LD15       ;then perform LOAD or VERIFY
F418- A5 9D      LDA *VERCK     ;get load/verify flag
F41A- D0 50      BNE LD115      ;if verifying, exit here
F41C- 20 28 F9      JSR TWAIT      ;else wait for normal IRQ
F41F- A5 96      LDA *CSTAT      ;get status byte
F421- 29 10      AND #$10      ;get bit for read error
F423- F0 09      BEQ LD210      ;if no error, print READY.
F425- A0 00      LDY #0       ;else cancel
F427- 84 9E      STY *NDX       ;keyboard buffer
F429- A0 60      LD209  LDY #MS22-MSG1    ;point to LOAD
F42B- 4C AF F5      JMP ERMSG      ;and exit with LOAD ERROR
F42E- A0 AE      LD210  LDY #MS17-MSG1    ;point to READY.
F430- 20 49 F3      JSR SPMMSG     ;print it if in direct mode
F433- 20 51 F3      JSR TXTST       ;test for direct mode
F436- D0 0B      BNE LD205      ;if not direct mode, reset CHRGET, exec
F438- A5 CA      LDA *EAH       ;else get new end of program hi
F43A- 85 2B      STA *VARTAB+1    ;in start of variables hi
F43C- A5 C9      LDA *EAL       ;the same for lo
F43E- 85 2A      STA *VARTAB
F440- 4C AD B4      JMP FINI      ;and go wait for new input
F443- 20 22 B6      LD205  JSR STXTPT     ;reset CHRGET (LOAD from program)
F446- 4C 0B B6      JMP FLOAD      ;and execute prg just loaded
;*****
;*          *
;* Print SEARCHING (FOR file-  *
;* name) in if direct mode.    *
;*          *
;*****


F449- 20 51 F3      LD300  JSR TXTST      ;check if direct mode
F44C- D0 1E      BNE LD115      ;if not, exit
F44E- A0 32      LDY #MS5-MSG1    ;else point to SEARCHING
F450- 20 85 F1      JSR MSG        ;print it on screen
F453- A5 D1      LDA *FNLEN      ;if no filename given

```

```

F455- F0 15      BEQ LD115      ;exit
F457- A0 30      LDY #MS6-MSG1 ;else point to FOR
F459- 20 85 F1   JSR MSG       ;print on screen
F45C- A4 D1      LD105       LDY *FNLEN    ;get length of name
F45E- F0 0C      BEQ LD115      ;if no name, exit
F460- A0 00      LDY #0        ;get offset zero
F462- B1 DA      LD110       LDA (FNADR),Y ;get name character
F464- 20 66 F2   JSR BSOUT     ;print it
F467- C8          INY          ;adapt offset
F468- C4 D1      CPY *FNLEN    ;if not whole name done
F46A- D0 F6      BNE LD110    ;loop
F46C- 60          LD115       RTS         ;else exit
;*****
;*                                     *
;* Print LOADING or VERIFYING if *
;* in direct mode.                 *
;*                                     *
;*****                               

F46D- A0 5F      LD410       LDY #MS10-MSG1 ;point to LOAD
F46F- A5 90      LDA *VERCK     ;if loading
F471- F0 02      BEQ LD420      ;print LOADING
F473- A0 60      LDY #MS21-MSG1 ;else point to VERIFY
F475- 20 49 F3   LD420       JSR SPMMSG   ;print message if in direct mode
F478- A0 69      LDY #f069-MSG1 ;point to ING
F47A- 4C 49 F3   JMP SPMMSG    ;and print that too
;*****
;*                                     *
;* Get parameters for LOAD, SAVE *
;* and VERIFY.                   *
;*                                     *
;*****                               

F47D- A2 00      PARS1       LDX #0        ;clear
F47F- 86 96      STX *CSTAT    ;status byte
F481- 86 D1      STX *FNLEN    ;filename length
F483- 86 D3      STX *SA       ;and set default secondary address
F485- E8          INX          ;get
F486- 86 D4      STX *FA       ;default device (cassette #1)
F488- 20 40 F5   JSR PR140    ;test for end of statement
F48B- 20 30 F5   JSR P200     ;if not, read filename
F48E- 20 40 F5   JSR PR140    ;test if end of statement
F491- 20 9F F4   JSR PR070    ;if not, read device number
F494- 86 D4      STX *FA       ;and store it
F496- 20 40 F5   JSR PR140    ;test if end of statement
F499- 20 9F F4   JSR PR070    ;if not, read secondary address
F49C- 86 D3      STX *SA       ;and store it
F49E- 60          PR060       RTS         ;then exit
;*****
;*                                     *
;* Read a comma, and a number      *
;* <256 in X.                      *
;*                                     *
;*****                               

F49F- 20 55 F5   PR070       JSR PR150    ;read a comma and test if end of statem
F4A2- 4C D4 C8   JMP GETBYT   ;then read integer in X
.FI "BASIC4.OF.MOS"

```

1BD9 33B0-4F89 BASIC4.OF.MOS

```

;name BASIC4.OF.MOS
;*****
;*                                     *
;* Send filename to IEEE bus.        *
;*                                     *
;*****                               

```

```

;*****
F4A5- A5 D3      OPENI   LDA *SA          ;get secondary address
F4A7- 30 F5      BMI PRO60        ;if not set, exit
F4A9- A4 D1      LDY *FNLEN       ;get length
F4AB- F0 F1      BEQ PRO60        ;if no name, exit
F4AD- 20 D5 F0    JSR LISTN        ;else send LISTEN
F4B0- A5 D3      LDA *SA          ;get secondary address
F4B2- 09 F0      OPENIB  ORA #$FO        ;set bits 4-7
F4B4- 20 43 F1    JSR SECND        ;and send it, set ATN false
F4B7- A5 96      LDA *CSTAT       ;get ST byte
F4B9- 10 05      BPL OP35        ;if no device present
F4BB- A0 74      OP37   LDY #MS13-MSG1  ;point to DEVICE NOT PRESENT
F4BD- 4C AF F5    JMP ERMSG        ;do DEVICE NOT PRESENT ERROR
F4C0- A5 D1      OP35   LDA *FNLEN       ;else get length again
F4C2- F0 0C      BEQ OP45        ;if none (why?) send UNLISTEN
F4C4- A0 00      LDY #0           ;else get offset zero
F4C6- B1 DA      OP40   LDA (FNADR),Y  ;get name character
F4C8- 20 9E F1    JSR CIOUT        ;send it to IEEE
F4CB- C8          INY              ;adapt counter
F4CC- C4 D1      CPY *FNLEN       ;if not all done
F4CE- D0 F6      BNE OP40        ;loop
F4D0- 4C B9 F1    OP45   JMP UNLSN       ;else send UNLISTEN and exit
;*****
;*
;* Search for a tape block with *
;* a specified name.          *
;*                                *
;*****
F4D3- 20 E5 F5    FAF     JSR FAH          ;read next tape header
F4D6- F0 1D      BEQ FAF40       ;if EOT block found, exit
F4D8- A0 05      LDY #5           ;else get offset in buffer (point to na
F4DA- 84 B5      STY *T2          ;in buffer offset
F4DC- A0 00      LDY #0           ;get offset in specified name
F4DE- 84 B4      STY *T1          ;in name offset
F4E0- C4 D1      FAF20  CPY *FNLEN       ;if end of specified name reached
F4E2- F0 10      BEQ FAF30       ;exit with A=length
F4E4- B1 DA      LDA (FNADR),Y  ;else get filename character
F4E6- A4 B5      LDY *T2          ;and offset in buffer
F4E8- D1 06      CMP (TBUF),Y    ;if not equal to character searched for
F4EA- D0 E7      BNE FAF          ;read next block
F4EC- E6 B4      INC *T1          ;else increment name offset
F4EE- E6 B5      INC *T2          ;and buffer offset
F4F0- A4 B4      LDY *T1          ;get name offset
F4F2- D0 EC      BNE FAF20       ;and loop
F4F4- 98          FAF30  TYA          ;get length of file name
F4F5- 60          FAF40  RTS          ;and exit
;*****
;*
;* Perform VERIFY.                *
;*                                *
;* (Called from Kernal: $FFDB).  *
;*                                *
;*****
F4F6- A9 01      VER     LDA #1           ;set load/verify flag
F4F8- 85 90      STA *VERCK       ;to verify
F4FA- 20 05 F4    JSR LD10          ;perform part of LOAD
F4FD- A5 96      LDA *CSTAT       ;get status byte
F4FF- 29 10      AND #$10          ;if any read error (=verify error)
F501- F0 05      BEQ VER10         ;VERIFY
F503- A0 6E      LDY #MS12-MSG1  ;point to VERIFY
F505- 4C AF F5    JMP ERMSG        ;and print VERIFY ERROR
F508- A0 AA      VER10  LDY #MS18-MSG1  ;else point to OK
F50A- 4C 85 F1    JMP MSG           ;and print that if in direct mode
;*****

```

```

        ;*
        ;* Fetch parameters for OPEN and *
        ;* CLOSE.
        ;*
        ;*****
F50D- A2 00    PARS2  LDX #0          ;clear:
F50F- 86 D3    STX *SA           ;secondary address
F511- 86 96    STX *CSTAT        ;the status byte
F513- 86 D1    STX *FNLEN        ;and name length
F515- E8      INX              ;set default
F516- 86 D4    STX *FA           ;device number to '1 (cassette #1)
F518- 20 58 F5  JSR PR130        ;test if current character not end of s
F51B- 20 D4 C8  JSR GETBYT       ;if not, read a number <256 in X (filen
F51E- 86 02    STX *LA           ;and set filenumber up
F520- 20 40 F5  JSR PR140        ;test if end of statement
F523- 20 9F F4  JSR PR070        ;if not, read devicenumber in X
F526- 86 D4    STX *FA           ;store it
F528- E0 03    CPX #3           ;if IEEE device
F52A- 90 02    BCC PR100        ;default secondary address=$FF
F52C- C6 03    DEC *SA           ;check if end of statement
F52E- 20 40 F5  PR100  JSR PR140 ;if not, read expression for sec. add.
F531- 20 9F F4  JSR PR070        ;and store secondary address
F534- 86 03    STX *SA           ;test if end of statement
F536- 20 40 F5  FR111  JSR PR140 ;if not, test for comma
F539- 20 55 F5  JSR PR150        ;and read expression for filename
F53C- 20 98 BD  P200  JSR FRMEVL ;throw away that string
F53F- 20 B5 C7  JSR FRESTR       ;and store length of filename
F542- 85 D1    STA *FNLEN        ;get pointer to name lo
F544- A5 1F    LDA *INDEX        ;in pointer to filename
F546- 85 DA    STA *FNADR        ;do hi
F548- A5 20    LDA *INDEX+1      ;also
F54A- 85 DB    STA *FNADR+1      ;then exit
F54C- 60      RTS              ;*****
        ;*
        ;* Check if end of statement. *
        ;* If so, pull return address, *
        ;* and exit to previous caller. *
        ;*
        ;*****
F54D- 20 76 00  PR140  JSR CHRGOT   ;get current character
F550- 00 02      BNE PR147       ;if end of statement
F552- 68      PLA             ;pull
F553- 68      PLA             ;return address
F554- 60      PR147  RTS         ;and exit to previous caller
        ;*****
        ;*
        ;* Check if current character is *
        ;* a comma, and next is not an   *
        ;* end of statement byte.       *
        ;*
        ;*****
F555- 20 F5 BE  PR150  JSR CHKCOM    ;check for comma
        ;*****
        ;*
        ;* Check if current character is *
        ;* not an end of statement byte. *
        ;*
        ;*****
F558- 20 76 00  PR130  JSR CHRGOT   ;get current character
F55B- 00 F7      BNE PR147       ;if end of statement
F55D- 4C 00 BF  PR135  JMP SNERR     ;do SYNTAX ERROR
        ;*****
        ;*

```

```

;* Perform OPEN.          *
;*                      *
;* (Called from Kernel: $FFCO).  *
;*                      *
;*****  

F560- 20 0D F5  OPEN  JSR PARS2      ;fetch OPEN parameters
F563- A5 D2  OP94  LDA *LA        ;get file number
F565- F0 F6  FOPEN  BEQ PR135    ;if zero, SYNTAX ERROR
F567- A0 0E  OP98  LDY #MS2-MSG1 ;point to FILE OPEN
F569- 20 C1 F2  JSR JLTLK      ;search table of open files
F56C- F0 41  BEQ ERMSG      ;if file number found, FILE OPEN ERROR
F56E- A6 AE  OP100  LDX *LDTND   ;else get number of open files
F570- A0 00  LDY #MS1-MSG1    ;point to TOO MANY FILES
F572- 84 96  STY *CSTAT      ;clear the status byte
F574- E0 0A  CPX #10        ;if already ten files open
F576- F0 37  BEQ ERMSG      ;do TOO MANY FILES ERROR
F578- E6 AE  INC *LDTND      ;else adapt number of open files
F57A- A5 D2  LDA *LA        ;get file number again
F57C- 90 51 02  STA LAT,X    ;move to table
F57F- A5 D3  LDA *SA        ;get secondary address
F581- 09 60  ORA #$60      ;add sec. add. bits
F583- 85 03  STA *SA        ;and store current sec. address
F585- 9D 65 02  STA SAT,X    ;get device number
F588- A5 '04  LDA *FA        ;and move to table
F58A- 9D 5B 02  STA FAT,X    ;if device is keyboard, exit
F58D- F0 55  BEQ OP175      ;if it is screen
F58F- C9 03  CMP #3         ;exit
F591- F0 51  BEQ OP175      ;if it is an IEEE device
F593- 90 03  BCC OP150      ;send name to IEEE bus
F595- 4C A5 F4  JMP OPENI     ;cassettes, get secondary address
F598- A5 D3  OP150  LDA *SA      ;remove IEEE bits
F59A- 29 0F  AND #$0F      ;if opened for read
F59C- D0 28  BNE OP200      ;test cassette keys
F59E- 20 57 F8  JSR CSTEL      ;print SEARCHING if in direct mode
F5A1- 20 49 F4  JSR LD300      ;get filename length
F5A4- A5 D1  LDA *FNLEN     ;if a name was given
F5A6- F0 1A  BEQ OP170      ;search named header
F5A8- 20 D3 F4  JSR FAF       ;and perform OPEN for cassette
F5AB- D0 24  BNE OP171      ;*****  

;*                      *
;* Exit with FILE NOT FOUND  *
;* ERROR. Files are aborted.  *
;*                      *
;*****  

F5AD- A0 24  OP160  LDY #MS4-MSG1 ;point to FILE NOT FOUND
;*****  

;*                      *
;* Exit with error message poin- *
;* to by Y (offset in table at  *
;* $FO00 and further).           *
;* Files are aborted.            *
;*                      *
;*****  

F5AF- 20. A2 F2  ERMSG  JSR CLALL      ;abort all files
;*****  

;*                      *
;* Exit with error message poin- *
;* to by Y (offset in table at  *
;* $FO00 and further).           *
;*                      *
;*****  

F5B2- A9 0D  f5b2  LDA #$0D      ;get a carriage return
F5B4- 20 66 F2  JSR BSOUT     ;print it to current output device

```

```

F5B7- A9 3F      LDA #'?'      ;get a question mark
F5B9- 20 66 F2    JSR BSOUT     ;print it
F5BC- 20 85 F1    JSR MSG       ;print error message
F5BF- 4C ED B3    JMP TYPERR    ;print ERROR, and restart BASIC
                           .FI "BASIC4.OF.M06"

```

18A6 33B0-4C56 BASIC4.OF.M06

```

;name BASIC4.OF.M06
;*****
;*
;* Perform OPEN to unnamed cas- *
;* sette file (=next file on   *
;* tape).                      *
;*
;*****
F5C2- 20 E5 F5  OP170  JSR FAH      ;load next tape header
F5C5- F0 E6      BEQ OP160     ;if EOT block found, FILE NOT FOUND
F5C7- D0 08      BNE OP171     ;else get parameters and exit
                           ;*****
;*
;* Perform OPEN for write to  *
;* cassette.                  *
;*
;*****
F5C9- 20 8C F8  OP200  JSR CSTE2    ;do PRESS PLAY AND RECORD
F5CC- A9 04      LDA #$04     ;get header byte for file
F5CE- 20 19 F6    JSR TAPEH    ;write header
F5D1- A6 04  OP171  LDX *FA      ;get device number
F5D3- A9 BF      LDA #191     ;and buffer length
F5D5- A4 D3      LDY *SA      ;and secondary address
F5D7- C0 60      CPY #$60     ;if opened for read
F5D9- F0 07      BEQ OP172     ;set buffer size and exit
F5DB- A0 00      LDY #0       ;else get offset zero
F5DD- A9 02      LDA #2       ;get header byte 2
F5DF- 91 D6      STA (TBUF),Y  ;move to buffer
F5E1- 98          TYA         ;get a zero
F5E2- 95 BA  OP172  STA *BUFPNT-1,X ;set current buffer size
F5E4- 60  OP175  RTS        ;and exit
                           ;*****
;*
;* Load next header from tape. *
;* On exit, Z=1 if EOT block   *
;* found.                      *
;*
;*****
F5E5- A5 9D      FAH        LDA *VERCK   ;get load/verify flag
F5E7- 48          PHA        ;save it
F5E8- 20 9A F8  FAH30   JSR RBLK     ;read a block from tape
F5EB- A0 00      LDY #0       ;get offset zero
F5ED- B1 D6      LDA (TBUF),Y  ;get 1st buffer character
F5EF- C9 05      CMP #5       ;if end of tape
F5F1- F0 21      BEQ FAH40    ;restore VERCK and exit with Z=1
F5F3- C9 01      CMP #1       ;if program header
F5F5- F0 04      BEQ FAH50    ;if not file header
F5F7- C9 04      CMP #4       ;read next block
F5F9- D0 ED      BNE FAH30    ;get header character in X
FSFB- AA  FAH50   TAX        JSR TXTST    ;test for direct mode
FSFC- 20 51 F3    BNE FAH45    ;if not, restore VERCK, exit with Z=0
FSFF- D0 11      LDY #MS17-MSG1 ;else point to FOUND
F601- A0 A3      JSR MSG      ;print it
F603- 20 85 F1    LDY #5       ;get offset to name in buffer
F606- A0 05

```

F608- B1 D6	FAH55	LDA (TBUF),Y	;get character of name
F60A- 20 D2 FF		JSR OUTCH	;print it
F60D- C8		INY	;adapt offset
F60E- C0 1F		CPY #31	;if 31 characters printed
F610- D0 F6		BNE FAH55	
F612- A0 01	FAH45	LDY #1	;set up for exit with Z=0
F614- 68	FAH40	PLA	;get load/verify flag back
F615- 85 90		STA *VERCK	
F617- 98		TYA	;set zero flag
F618- 60		RTS	;and exit
		*****	*****
		;	*
		;	/* Write tape header with lead-
		;	ing character in A.
		;	*
		*****	*****
F619- 85 B4	TAPEH	STA *T1	;save leading byte
F61B- 20 95 F6		JSR f695	;set pointer to current cassette buffer
F61E- A5 FC		LDA *STAH	;save SAVE start address
F620- 48		PHA	;on stack
F621- A5 FB		LDA *STAL	
F623- 48		PHA	
F624- A5 CA		LDA *EAH	;save end address
F626- 48		PHA	
F627- A5 C9		LDA *EAL	;also
F629- 48		PHA	
F62A- A0 BF		LDY #191	;get buffer length
F62C- A9 20		LDA #\$20	;get a blank
F62E- 91 D6	BLNK2	STA (TBUF),Y	;and fill buffer with blanks
F630- 88		DEY	;until
F631- D0 FB		BNE BLNK2	;whole buffer done
F633- A5 B4		LDA *T1	;get leading character back
F635- 91 D6		STA (TBUF),Y	;move to buffer
F637- C8		INY	;point to next position
F638- A5 FB		LDA *STAL	;get start address lo
F63A- 91 D6		STA (TBUF),Y	;in buffer
F63C- C8		INY	;point to next position
F63D- A5 FC		LDA *STAH	;get start address .hi
F63F- 91 D6		STA (TBUF),Y	;in buffer
F641- C8		INY	;point to next position
F642- A5 C9		LDA *EAL	;get end address lo
F644- 91 D6		STA (TBUF),Y	;in buffer
F646- C8		INY	;point to next position
F647- A5 CA		LDA *EAH	;get end address hi
F649- 91 D6		STA (TBUF),Y	;in buffer
F64B- C8		INY	;point to next position
F64C- 84 B5		STY *T2	;and save current buffer index
F64E- A0 00		LDY #0	
F650- 84 B4		STY *T1	
F652- A4 B4	TH20	LDY *T1	;get name offset
F654- C4 D1		CPY *FNLEN	;if whole name moved
F656- F0 0C		BEQ TH30	;set addresses
F658- B1 DA		LDA (FNADR),Y	;get name character
F65A- A4 B5		LDY *T2	;and buffer offset
F65C- 91 D6		STA (TBUF),Y	;move character to buffer
F65E- E6 B4		INC *T1	;adapt name offset
F660- E6 B5		INC *T2	;and buffer offset
F662- D0 EE		BNE TH20	;and loop
F664- 20 AB F6	TH30	JSR LDAD1	;set up pointers to buffer
F667- A9 69		LDA #\$69	;get timing constant
F669- 85 C3		STA *SHCNH	
F66B- 20 D5 F8		JSR TWRT2	;and write buffer to tape
F66E- 68		PLA	;then get
F66F- 85 C9		STA *EAL	;end address lo back

```

F671- 68 PLA ;get
F672- 85 CA STA *EAH ;end address hi back
F674- 68 PLA ;get
F675- 85 FB STA *STAL ;start address lo back
F677- .68 PLA ;get
F678- 85 FC STA *STAH ;start address hi back
F67A- 60 RTS ;and exit
;***** *
;* Read start and end address *
;* from tape header and move *
;* them to ($FB) and ($C9). *
;* *
;***** *
F67B- 20 2B F9 LDAD2 JSR TWAIT ;wait for normal IRQ
F67E- A2 00 LDX #0
F680- A0 01 LDY #1 ;get offset in buffer
F682- B1 D6 LDAD3 LDA (TBUF),Y ;get address byte from buffer
F684- 95 C7 STA *SAL,X ;move to save area
F686- E8 IMX ;adapt counter
F687- C8 INY ;and buifer ofiset
F688- EO D4 CPX #4 ;if not 4 bytes moved
F68A- D0 F6 BNE LDAD3 ;loop
F68C- A5 C7 LDA *SAL ;else get start address lo
F68E- 85 FB STA *STAL ;in start pointer
F690- A5 C8 LDA *SAH ;get start address hi
F692- 85 FC STA *STAH ;and complete start pointer
F694- 60 RTS ;and exit
;***** *
;* Set pointer to current cas- *
;* sette buffer in ($D6), ac- *
;* cording to current device *
;* number. *
;* *
;***** *
F695- A9 7A f695 LDA #L,TAPE1 ;get lo byte start address
F697- 85 D6 STA *TBUF ;of cassette buffer 1 in buffer pointer
F699- A9 02 LDA #H,TAPE1 ;get hi also
F69B- 85 D7 STA *TBUF+1 ;complete pointer
F69D- A5 D4 LDA *FA ;get device number
F69F- 4A LSR A ;if device was one
F6A0- B0 D8 BCS ZZ10 ;exit
F6A2- A9 3A LDA #L,TAPE2 ;else get lo byte start address
F6A4- 85 D6 STA *TBUF ;of cassette buffer 2 in buffer pointer
F6A6- A9 03 LDA #H,TAPE2 ;get hi also
F6A8- 85 D7 STA *TBUF+1 ;complete pointer
F6AA- 60 ZZ10 RTS ;and exit
;***** *
;* Set start and end addresses *
;* of current cassette buffer as *
;* start and end addresses for *
;* write (of header block). *
;* *
;***** *
F6AB- 20 2B F9 LDAD1 JSR TWAIT ;wait for normal IRQ
F6AE- 20 95 F6 JSR f695 ;set pointer to cassette buffer
F6B1- A5 D6 LDA *TBUF ;get lo byte
F6B3- 85 FB STA *STAL ;as start address lo
F6B5- 18 CLC ;prepare for add
F6B6- 69 C0 ADC #192 ;add buffer length (=end addres of buff
F6B8- 85 C9 STA *EAL ;set as end address lo
F6BA- A5 D7 LDA *TBUF+1 ;get hi byte of buffer address

```

```

F6BC- 85 FC      STA *STAH          ;as start address hi
F6BE- 69 00      ADC #$00          ;add carry if page crossed
F6C0- 85 CA      STA *EAH           ;and complete end address
F6C2- 60          RTS              ;then exit
;*****
;*
;* Perform SYS.          *
;* (Called from Kernal: $FFDE).  *
;*
;*****
F6C3- 20 84 B0  SYS   JSR FRMNUM     ;read arithmetic expression
F6C6- 20 20 C9      JSR GETADR      ;convert to integer in ($11)
F6C9- 6C 11 00      JMP (POKER)    ;and go there
.FI "BASIC4.OF.M07"

```

195D 3380-4000 BASIC4.0F.M07

```

;name BASIC4.OF.M07
;*****
;* Move pointers to program      *
;* start and end to SAVE start  *
;* and end address.              *
;*                                *
;*****



F6CC- A5 2A    SV60   LDA *VARTAB      ;get start of variables lo
F6CE- 85 C9      STA *EAL        ;in end address lo
F6D0- A5 2B      LDA *VARTAB+1   ;get start of variables hi
F6D2- 85 CA      STA *EAH        ;in end address hi
F6D4- A5 29      LDA *TXTTAB+1   ;get start of program hi
F6D6- 85 FC      STA *STAH       ;in start address hi
F6D8- A5 28      LDA *TXTTAB     ;get start of program lo
F6DA- 85 FB      STA *STAL       ;in start address lo
F6DC- 60          RTS            ;and exit
;*****
;*                                *
;* Perform SAVE.                 *
;*                                *
;* (Called from Kernal: $FFD8).  *
;*                                *
;*****



F6D0- 20 7D F4    SAVE   JSR PARS1      ;fetch LOAD/SAVE/VERIFY parameters
F6E0- 20 CC F6    SV3    JSR SV60       ;set start and end addresses
F6E3- A5 D4    SV5    LDA *FA          ;get device number
F6E5- D0 05    SV6    BNE SV20       ;if SAVE to keyboard
F6E7- A0 74    SV10   LDY #MS13-MSG1  ;point to DEVICE NOT PRESENT
F6E9- 4C AF F5      JMP ERMSG      ;and do that error
F6EC- C9 03    SV20   CMP #3          ;if SAVE to screen
F6EE- F0 F7      BEQ SV10       ;do DEVICE NOT PRESENT ERROR
F6F0- 90 50      BCC SV100      ;if save to cassette, do those, else:
;*****
;*                                *
;* SAVE to an IEEE device.      *
;*                                *
;*****



F6F2- A9 61      LDA ##$61       ;get secondary address for save
F6F4- 85 D3      STA *SA          ;in current secondary address
F6F6- A4 D1      LDY *FNLEN      ;if no filename given
F6F8- D0 03      BNE f6fd       ;do SYNTAX ERROR
F6FA- 4C 00 BF      JMP SNERR      ;else send filename to IEEE
F6FD- 20 A5 F4      JSR OPENI      ;send LISTEN
F700- 20 05 F0      JSR LISTN      ;send LISTEN

```

```

F703- A5 D3      LDA *SA          ;get secondary address
F705- 20 43 F1    JSR SECND       ;send it, then set ATN false
F708- A0 00      LDY #0          ;get offset zero
F70A- 20 BB FB    JSR RD300       ;move start address to ($C7)
F70D- A5 C7      LDA *SAL         ;get start address lo
F70F- 20 9E F1    JSR CIOUT        ;send to IEEE
F712- A5 C8      LDA *SAH         ;get start address hi
F714- 20 9E F1    JSR CIOUT        ;send to IEEE
· F717- 20 0B FD  SV30     JSR WRT62       ;check if end address reached
F71A- F0 10      BEQ SV50         ;if so, UNLISTEN and exit
F71C- B1 C7      LDA (SAL),Y      ;get prg byte
F71E- 20 9E F1    JSR CIOUT        ;send it to IEEE
F721- 20 43 F3    JSR STOP         ;test for STOP key
F724- E6 C7      INC *SAL         ;adapt pointer
F726- D0 EF      BNE SV30         ;if page crossed
F728- E6 C8      INC *SAH         ;adapt hi byte
F72A- D0 EB      BNE SV30         ;and loop
F72C- 20 B9 F1  SV50     JSR UNLSN        ;end address reached, send UNLISTEN
;*****
;*                                *
;* Clear IEEE channel.           *
;*                                *
;*****
F72F- 24 D3      CLSEI          BIT *SA          ;if default secondary address
F731- 30 78      BMI UD65         ;exit
F733- 20 D5 F0    JSR LISTN        ;send LISTEN
F736- A5 D3      LDA *SA          ;else get secondary address
F738- 29 EF      AND #$EF         ;remove bit 4
F73A- 09 E0      ORA #$E0         ;set bits 5-7 (secondary command byte)
F73C- 20 43 F1    JSR SECND       ;send it, then set ATN false
F73F- 4C B9 F1    JMP UNLSN        ;and send UNLISTEN
;*****
;*                                *
;* Save to cassette.            *
;*                                *
;*****
F742- 20 95 F6  SV100    JSR f695         ;set pointer to current cassette buffer
F745- 20 8C F8      JSR CSTE2        ;do PRESS PLAY AND RECORD
F748- 20 51 F3      JSR TXTST        ;test for direct mode
F74B- D0 08      BNE SV105        ;if in direct mode
F74D- A0 64      LDY #MS11-MSG1   ;point to WRITING
F74F- 20 85 F1      JSR MSG          ;print the message
F752- 20 5C F4      JSR LD105        ;print filename, if one
F755- A9 01      SV105     LDA #1          ;get header leading byte
F757- 20 19 F6      JSR TAPEH        ;write program header
F75A- 20 CE F8      JSR TWRT         ;write blocks to tape
F75D- A5 D3      LDA *SA          ;get secondary address
F75F- 29 02      AND #$02         ;if it was 2
;BUG: accepts any address that has bit 2
;      set, e.g. 2, 3, 6, 7, 10, 11, etc.
F761- F0 48      BEQ UD65         ;get EOT header leading byte
F763- A9 05      LDA #5          ;get EOT header leading byte
F765- 4C 19 F6    JMP TAPEH        ;and write EOT block
;*****
;*                                *
;* Update jiffy clock.          *
;*                                *
;* (Called from Kernal: $FFEA). *
;*                                *
;*****
F768- E6 99      UDTIM     INC *CRFAC      ;increment correction clock lo
F76A- A5 99      LDA *CRFAC      ;get it
F76C- D0 02      BNE UD10         ;if page crossed
F76E- E6 9A      INC *CRFAC+1   ;increment hi byte also

```

F770- C9 6F		UD10	CMP #\$6F	;if correction
F772- D0 06			BNE UD20	;clock
F774- A5 9A			LDA *CRFAC+1	;has reached
F776- C9 02			CMP #\$02	;value 623
F778- F0 21			BEQ UD50	;skip jiffy clock increment
F77A- E6 8F		UD20	INC *CTIMR+2	;else increment jiffy clock lo
F77C- D0 06			BNE UD30	;if page crossed
F77E- E6 8E			INC *CTIMR+1	;increment middle byte
F780- D0 02			BNE UD30	;if page crossed
F782- E6 80			INC *CTIMR	;increment hi byte
F784- A2 00		UD30	LDX #0	;point to hi byte
F786- B5 80		UD40	LDA *CTIMR,X	;get jiffy clock byte
F788- DD AC F7			CMP UD70,X	;if not at 24 hour value
F78B- 90 14			BCC UD60	;get keypress and exit
F78D- E8			INX	;else point to next byte
F78E- E0 03			CPX #3	;if not all done
F790- D0 F4			BNE UD40	;loop
F792- A9 00			LDA #0	;if at 24 hours
F794- 95 8C		UD45	STA *CTIMR-1,X	;clear jiffy clock bytes
F796- CA			DEX	;until
F797- D0 FB			BNE UD45	;all done
F799- F0 06			BEQ UD60	;then get keypress and exit
F79B- A9 00		UD50	LDA #0	;if correction=623
F79D- 85 99			STA *CRFAC	;clear
F79F- 85 9A			STA *CRFAC+1	;it
F7A1- AD 12 E8		UD60	LDA PIAK	;get key image of scan line 9
F7A4- CD 12 E8			CMP PIAK	;wait until
F7A7- D0 F8			BNE UD60	;steady
F7A9- 85 9B			STA *STKEY	;and store it
F7AB- 60		UD65	RTS	;then exit
			;	*****
			;	*
			;	/* Constant: Number of jiffies *
			;	/* in 24 hours. *
			;	/* Value=5184001. *
			;	*
			;	*****
F7AC- 4F		UD70	.BY \$4F	
F7AD- 1A			.BY \$1A	
F7AE- 01			.BY \$01	
			;	*****
			;	*
			;	/* Set input device. *
			;	*
			;	/* (Called from Kernal: \$FFC6). *
			;	*
			;	*****
F7AF- 48		CHKIN	PHA	;save A
F7B0- 8A			TXA	;and
F7B1- 48			PHA	;X
F7B2- 98			TYA	;and
F7B3- 48			PHA	;Y on stack
F7B4- A9 00			LDA #0	;clear
F7B6- 85 96			STA *CSTAT	;status byte
F7B8- 8A			TXA	;get filenumber again
F7B9- 20 C1 F2			JSR JLTLK	;search table for it
F7BC- F0 05			BEQ JX310	;if number not found
F7BE- A0 17	JX300		LDY #MS3-MSG1	;point to FILE NOT OPEN
F7CD- 4C AF F5	JX305		JMP ERMSG	;and exit with error
F7C3- 20 CD F2	JX310		JSR JZ100	;else set file data from table
F7C6- A5 D4			LDA *FA	;get device number
F7C8- F0 10			BEQ JX320	;if keyboard, set current dev.# and exi
F7CA- C9 03			CMP #3	;if screen
F7CC- F0 0C			BEQ JX320	;do the same

F7CE- B0 0F	BCS JX330	;if IEEE, do that
F7D0- A6 D3	LDX *SA	;cassette, get secondary address
F7D2- E0 60	Cpx #\$60	;if it is zero
F7D4- F0 04	BEQ JX320	;set current input device, exit
F7D6- A0 86	LDY #MS15-MSG1	;else point to NOT INPUT FILE
F7D8- D0 E6	BNE JX305	;and exit with error
F7DA- 85 AF	JX320 STA *DFLTN	;set current input device
F7DC- 4C 9C F2	JMP RSTOR	;and restore AXY
F7DF- 48	JX330 PHA	;IEEE device, save device number
F7E0- 20 E1 DB	JSR OLDCLR	;clear DS\$
F7E3- 20 D2 F0	JX3301 JSR TALK	;send TALK to IEEE
F7E6- A5 D3	LDA *SA	;get secondary address
F7E8- 10 06	BPL JX340	;if default
F7EA- 20 98 F1	JSR TKATN	;set NDAC and ATN
F7ED- 4C F3 F7	JMP JX350	;and test for device not present
F7F0- 20 93 F1	JX340 JSR TKSA	;send secondary address
F7F3- A5 96	JX350 LDA *CSTAT	;get status byte
F7F5- 10 03	BPL f7fa	;if device not present
F7F7- 4C BB F4	JMP OP37	;give DEVICE NOT PRESENT ERROR
F7FA- 68	f7fa PLA	;get device number back
F7FB- 4C DA F7	JMP JX320	;and set device number, restore AXY
	.FI "BASIC4.OF.M08"	

1884 33B0-4C64 BASIC4.OF.M08

	;name BASIC4.OF.M08	
	;*****	
	;* * * * *	
	;* Set output device. *	
	;* * * * *	
	;* (Called from Kernal: \$FFC9). *	
	;* * * * *	
	;*****	
F7FE- 48	CHKOUT PHA	;save A
F7FF- 8A	TXA	;and
F800- 48	PHA	;X
F801- 98	TYA	;and
F802- 48	PHA	;Y on stack
F803- A9 00	LDA #0	;clear
F805- 85 96	STA *CSTAT	;status byte
F807- 8A	TXA	;get filenumber again
F808- 20 C1 F2	JSR JLTLK	;search table for it
F808- D0 B1	BNE JX300	;if not found, do FILE NOT OPEN ERROR
F800- 20 CD F2	JSR JZ100	;else set file data from table
F810- A5 D4	LDA *FA	;get device number
F812- D0 04	BNE CK10	;if keyboard,
F814- A0 94	LDY #MS16-MSG1	;point to NOT OUTPUT FILE
F816- D0 A8	BNE JX305	;and exit with error
F818- C9 03	CK10 CMP #3	;if screen
F81A- F0 0C	BEQ JX360	;store output device number
F81C- B0 0F	BCS JX370	;if IEEE, do that
F81E- A6 D3	LDX *SA	;cassette, get secondary address
F820- E0 60	Cpx #\$60	;if it is not zero
F822- D0 04	BNE JX360	;set current output device, exit
F824- A0 94	LDY #MS16-MSG1	;else point to NOT OUTPUT FILE
F826- D0 98	BNE JX305	;and exit with error
F828- 85 B0	JX360 STA *DFLTO	;set current output device
F82A- 4C 9C F2	JMP RSTOR	;and restore AXY
F82D- 48	JX370 PHA	;IEEE device, save device number
F82E- 20 E1 DB	JSR OLDCLR	;clear DS\$
F831- 20 D5 F0	JX3701 JSR LISTN	;send LISTEN to IEEE bus
F834- A5 D3	LDA *SA	;get secondary address
F836- 10 05	BPL JX380	;if default

```

F838- 20 48 F1      JSR SCATN      ;set ATN false
F83B- D0 03          BNE JX390      ;and test for device not present
F83D- 20 43 F1      JX380        ;send secondary address
F840- A5 96          JX390        ;get status byte
F842- 10 03          BPL f847      ;if device not present
F844- 4C BB F4      JMP OP37      ;give DEVICE NOT PRESENT ERROR
F847- 68             f847        PLA           ;get device number back
F848- 4C 28 F8      JMP JX360      ;and set device number, restore AXY
;*****=====
;*          *
;* Increment tape buffer coun-  *
;* counter. If buffer full, Z=1. *
;*          *
;*****=====

F84B- 20 95 F6      JTP20        JSR f695      ;get pointer to current tape buffer
F84E- A6 D4          LDX *FA       ;get device number
F850- F6 BA          INC *BUFPNT-1,X ;adapt number of bytes in buffer
F852- B4 BA          LDY *BUFPNT-1,X ;get that number
F854- C0 C0          CPY #192     ;if buffer full
F856- 60             RTS         ;exit with Z=1
;*****=====
;*          *
;* Test cassette keypress.    *
;* If no key pressed, print   *
;* PRESS PLAY ON TAPE #X, and *
;* wait until key pressed.    *
;*          *
;*****=====

F857- 20 7A F8      CSTEI        JSR CS10      ;test if any key down
F85A- F0 2F          BEQ CS25      ;if so, exit without message
F85C- A0 41          LDY #MS7-MSG1 ;else point to PRESS PLAY
F85E- 20 85 F1      CS30         JSR MSG       ;print it
F861- A0 56          LDY #MS9-MSG1 ;then point to ON TAPE #
F863- 20 85 F1      JSR MSG       ;print that to
F866- A5 D4          LDA *FA       ;get device number
F868- 09 30          ORA #$30     ;convert to ASCII
F86A- 20 02 E2      JSR PRT      ;print it on screen.
F86D- 20 35 F9      CS40         JSR TSTOP    ;wait for normal IRQ (write header)
F870- 20 7A F8      JSR CS10      ;test if any key down
F873- D0 F8          BNE CS40      ;if not, loop
F875- A0 AA          LDY #MS18-MSG1 ;else point to OK
F877- 4C 85 F1      JMP MSG       ;and print that
;*****=====
;*          *
;* Test cassette keypress.    *
;* If key pressed, exit with *
;* Z=1.                      *
;*          *
;*****=====

F87A- A9 10          CS10         LDA #$10      ;get bit mask for cassette#1 sense
F87C- A6 D4          LDX *FA       ;get device number
F87E- CA             DEX          ;if cassette #1
F87F- F0 02          BEQ CS20      ;test for cassette#1 sense
F881- A9 20          LDA #$20      ;else get mask for cassette#2 sense
F883- 2C 10 E8      CS20         BIT PIAL     ;test if key pressed
F886- D0 03          BNE CS25      ;if not, exit with Z=0
F888- 2C 10 E8      CS25         BIT PIAL     ;else set Z
F88B- 60             CS25         RTS          ;and exit
;*****=====
;*          *
;* Test cassette keypress.    *
;* If no key pressed, print   *
;* PRESS PLAY AND RECORD ON TAPE *
;* TAPE #X, and wait for key   *
;*          *

```

```

;* pressed.          *
;*          *
;*****  

F88C- 20 7A F8  CSTE2 JSR CS10      ;test if key pressed
F88F- F0 FA          BEQ CS25      ;if so, exit without message
F891- A0 41          LDY #MS7-MSG1 ;else point to PRESS PLAY
F893- 20 85 F1          MSG         ;print it
F896- A0 4D          LDY #MS8-MSG1 ;point to AND RECORD
F898- D0 C4          BNE CS30      ;print that and ON TAPE #X
;*****  

;*          *
;* Read next block from tape.  *
;*          *
;*****  

F89A- A9 00          RBLK        LDA #0          ;clear
F89C- 85 96          STA *CSTAT    ;status byte
F89E- 85 9D          STA *VERCK    ;and set to load (not verify)
F8A0- 20 AB F6          LLOAD1     ;set addresses to current buffer
F8A3- 20 2B F9  TRD   JSR TWAIT    ;wait for normal IRQ
F8A6- 20 57 F8          CSTE1     ;if key released, do PRESS PLAY...
F8A9- 78              SEI          ;disable interrupts
F8AA- A9 00          LDA #0          ;clear:
F8AC- 85 C2          STA *RDFLG    ;scan/load/count flag
F8AE- 85 CE          STA *SNSWL    ;sync flag
F8B0- 85 CB          STA *CMPO     ;speed correction
F8B2- 85 CO          STA *PTR1     ;pass1 error count
F8B4- 85 C1          STA *PTR2     ;pass2 error count
F8B6- 85 B2          STA *DPSW     ;byte received flag
F8B8- A6 D4          LDX *FA       ;get device number
F8BA- CA              DEX          ;if cassette#2
F8BB- F0 07          BEQ TRD2    ;enable IRQ on CB1
F8BD- A9 90          LDA #$90    ;from VIA
F8BF- 80 4E E8          IER         ;if cassette #1
F8C2- D0 03          BNE TRD3    ;enable IRQ on CA1 from PIA1
F8C4- EE 11 E8  TRD2  INC PIAL1    ;use 7th IRQ vector
F8C7- A2 DE  TRD3   LDX #14      ;and read block
F8C9- D0 15          BNE TAPE    ;*****  

;*          *
;* Write a block to tape.  *
;*          *
;*****  

F8CB- 20 AB F6  WBLK   JSR LLOAD1   ;set addresses to current buffer
F8CE- 20 2B F9  TWRT   JSR TWAIT    ;wait for normal IRQ
F8D1- A9 14          LDA #$14      ;get timing constant
F8D3- 85 C3          STA *SHCNH    ;do PRESS PLAY AND RECORD
F8D5- 20 8C F8  TWRT2  JSR CSTE2    ;disable interrupts
F8D8- 78              SEI          ;enable interrupt
F8D9- A9 A0          LDA #$A0      ;from timer 2 from VIA
F8DB- 80 4E E8          IER         ;use 4th IRQ vector
F8DE- A2 08          LDX #8       ;*****  

;*          *
;* Common code for block read or *
;* block write from/to cassette. *
;*          *
;*****  

F8E0- 20 E0 FC  TAPE   JSR BSIV     ;alter IRQ vector
F8E3- A9 02          LDA #2       ;set
F8E5- 85 DE          STA *FSBLK    ;pass counter (header+2 passes)
F8E7- 20 C9 FB          NEWCH     ;prepare for next byte
F8EA- CE 13 E8          PIAS       ;disable screen refresh interrupt
F8ED- A6 D4          LDX *FA       ;get device number
F8EF- CA              DEX          ;if cassette #1

```

```

F8F0- D0 09      BNE TP20
F8F2- A9 34      LDA #$34
F8F4- 8D 13 E8    STA PIAS      ;start motor of cassette #1
F8F7- 85 F9      STA *CAS1     ;flag it in motor byte
F8F9- D0 0A      BNE TP30
F8FB- AD 40 E8    TP20      LDA PIA      ;cassette #2
.F8FE- 86 FA      STX *CAS2     ;flag cassette #2 motor on
F900- 29 EF      AND #$EF     ;set bit 4 off
F902- 8D 40 E8    STA PIA      ;and switch motor on
F905- A2 FF      TP30      LDX #$FF     ;get delay value
F907- A0 FF      TP32      LDY #$FF     ;in XY
F909- 88          TP35      DEY         ;and wait
F90A- D0 FD      BNE TP35     ;until
F90C- CA          DEX         ;motor
F90D- D0 F8      BNE TP32     ;is on speed
F90F- 8D 49 E8    STA T2H      ;and start timer2
F912- 58          CLI         ;enable interrupts
F913- A9 E4      TP40      LDA #H,KEY   ;get normal IRQ hi byte
F915- C5 91      CMP *CINV+1  ;if normal IRQ
F917- F0 11      BEQ TP50     ;exit
F919- 20 35 F9    JSR TSTOP    ;else test for STOP key
F91C- 2C 13 E8    BIT RIAS    ;if not pressed
F91F- 10 F2      BPL TP40     ;wait for IRQ from PIA1
F921- 2C 12 E8    BIT PIAK    ;if retrace IRQ condition
F924- 20 68 F7    JSR UDTIM   ;update clock
F927- 4C 13 F9    JMP TP40     ;and wait for normal IRQ
F92A- 60          TP50      RTS         ;exit
.FI "BASIC4.OF.M09"

```

1A62 33B0-4E12 BASIC4.OF.M09

```

;name BASIC4.OF.M09
;*****
;* Wait for normal IRQ.
;*
;*****
F92B- 20 35 F9    TWAIT    JSR TSTOP    ;test STOP key
F92E- A9 E4      LDA #H,KEY   ;get normal IRQ vector hi byte
F930- C5 91      CMP *CINV+1  ;compare with IRQ vector hi
F932- D0 F7      BNE TWAIT   ;and wait until equal
F934- 60          RTS       ;then exit
;*****
;* Test STOP key during cassette *
;* activity (needed because nor- *
;* mal keyboard scan inhibited). *
;*
;*****
F935- 20 35 F3    TSTOP    JSR STOP1    ;test STOP key
F938- D0 08      BNE STOP3   ;if not pressed, exit
F93A- 20 C0 FC    JSR TNIF     ;else turn off cassette motors
F93D- 20 B8 F2    JSR JX770   ;restore default I/O
F940- 85 10      STA *CHANNL  ;and set default CMD output device
F942- 4C 43 F3    STOP3    JMP STOP     ;then perform STOP or return to caller
;*****
;* Set timer1 timing.
;* Wait until timer2 will change *
;* its hi-byte, then start      *
;* timer1.                      *
;*
;*****

```

```

F945- 86 CC      STT1    STX *CMP1      ;save elapsed time
F947- A5 CB      LDA *CMPO      ;get speed correction
F949- 0A          ASL A       ;multiply
F94A- 0A          ASL A       ;by four
F94B- 18          CLC         ;prepare for add
F94C- 65 CB      ADC *CMPO      ;add to itself (times 5)
F94E- 18          CLC         ;prepare for add
F94F- 65 CC      ADC *CMP1      ;add elapsed time
F951- 85 CC      STA *CMP1      ;and store as elapsed time
F953- A9 00      LDA #0       ;clear result area
F955- 24 CB      BIT *CMPO      ;if speed correction positive
F957- 30 01      BMI STT2      ;and elapsed time now negative
F959- 2A          ROL A       ;set LSbit
F95A- 06 CC      STT2    ASL *CMP1      ;and get highest
F95C- 2A          ROL A       ;two
F95D- 06 CC      ASL *CMP1      ;bits of elapsed time
F95F- 2A          ROL A       ;in A
F960- AA          TAX         ;move to X
F961- AD 48 E8   STT3    LDA T2L       ;get timer 2 lo
F964- C9 15      CMP #$15      ;if not at $15 yet
F966- 90 F9      BCC STT3      ;wait until so
F968- 65 CC      ADC *CMP1      ;then add elapsed time
F96A- 80 44 E8   STA T1L       ;store value in timer1 lo
F96D- 8A          TXA         ;get constant again
F96E- 60 49 E8   ADC T2H       ;add current timer2 hi
F971- 80 45 E8   STA T1H       ;and start timer1 (will time out later)
F974- 58          CLI         ;wait for timer2 interrupt
F975- 60          RTS         ;and exit
;*****
;*          *
;* Read one byte from tape in  *
;* $DF.                      *
;*          *
;* (Entered from interrupt,   *
;* IRQ vector number 7).     *
;*          *
;*****



F976- AE 49 E8   READ    LDX T2H       ;get timer2 hi
F979- A0 FF      LDY #$FF      ;get complement value
F97B- 98          TYA         ;in Y and A
F97C- ED 48 E8   SBC T2L       ;calculate elapsed time
F97F- EC 49 E8   CPX T2H       ;if hi byte changed
F982- D0 F2      BNE READ      ;repeat
F984- 86 CC      RD1    STX *CMP1      ;else set elapsed time
F986- AA          TAX         ;get difference in lo's in X
F987- 8C 48 E8   STY T2L       ;set timer2
F98A- 8C 49 E8   STY T2H       ;to maximum value (wait for timer1)
F98D- 98          TYA         ;get $FF
F98E- E5 CC      SBC *CMP1      ;calculate remaining time
F990- 86 CC      STX *CMP1      ;save difference
F992- 4A          LSR A       ;divide A
F993- 66 CC      ROR *CMP1      ;and difference
F995- 4A          LSR A       ;by
F996- 66 CC      RD2    ROR *CMP1      ;four
F998- A5 CB      RD3    LDA *CMPO      ;get speed correction
F99A- 18          CLC         ;prepare for add
F99B- 69 3C      ADC #$3C      ;add offset (test in middle of cycle)
F99D- 2C 40 E8   BIT PIA       ;clear interrupt flag of cassette#2
F9A0- 2C 10 E8   BIT PIAL      ;and of cassette#1
F9A3- C5 CC      CMP *CMP1      ;if cycle shorter
F9A5- B0 4A      BCS RDBK      ;exit
F9A7- A6 B2      LDX *DPSW      ;if complete byte read
F9A9- F0 03      BEQ RJDJ      ;go move to memory
F9AB- 4C 9C FA   JRADJ   JMP RADJ

```

F9AE- A6 B7	RJDJ	LDX *PCNTR	;else get bit counter
F9B0- 30 1B		BMI JRAD2	;if all bits done, do end of byte
F9B2- A2 00		LDX #0	;else get bit value zero
F9B4- 69 30		ADC #\$30	;get timing
F9B6- 65 CB		ADC *CMPO	;plus speed correction
F9B8- C5 CC		CMP *CMP1	;if cycle shorter
F9BA- B0 1C		BCS RADX2	;go exit with zero bit
F9BC- E8		INX	;else get one bit
F9BD- 69 26		ADC #\$26	;add timing
F9BF- 65 CB		ADC *CMPO	;and speed correction
F9C1- C5 CC		CMP *CMP1	;if cycle shorter
F9C3- B0 17		BCS RAD1	;go exit with one bit
F9C5- 69 2C		ADC #\$2C	;else add timing
F9C7- 65 CB		ADC *CMPO	;and speed correction
F9C9- C5 CC		CMP *CMP1	;if cycle shorter
F9CB- 90 03		BCC SRER	
F9CD- 4C 4C FA	JRAD2	JMP RAD2	;go do end of byte
F9D0- A5 CE	SRER	LDA *SNSWL	;if cycle longer and no sync
F9D2- F0 1D		BEQ RDBK	;do end of byte
F9D4- 85 BE		STA *RER	;else set timing
F9D6- D0 19		BNE RDBK	;and go do end of byte
F9D8- E6 BF	RADX2	INC *REZ	;0 read, adapt 0/1 balance
F9DA- B0 02		BCS RAD5	
F9DC- C6 BF	RAD1	DEC *REZ	;1 read, adapt 0/1 balance
F9DE- 38	RAD5	SEC	;prepare for subtract
F9DF- E9 13		SBC #\$13	;subtract timing for 0/1 discrimination
F9E1- E5 CC		SBC *CMP1	;subtract cycle time
F9E3- 65 9C		ADC *SVXT	;add accumulated speed correction
F9E5- 85 9C		STA *SVXT	;and store as new acc. speed corr.
F9E7- A5 B9		LDA *FIRT	;get cycle counter
F9E9- 49 01		EOR #\$01	;increment it
F9EB- 85 B9		STA *FIRT	;and store it
F9ED- F0 21		BEQ RAD3	;if one cycle done
F9EF- 86 D9		STX *DATA	;save bit value
F9F1- A5 CE	RDBK	LDA *SNSWL	;if no sync yet
F9F3- F0 18		BEQ RADBK	;exit
F9F5- 2C 40 E8		BIT IFR	;else if timer1 not timed out
F9F8- 50 13		BVC RADBK	;exit
F9FA- A9 00		LDA #0	;if timed out
F9FC- 85 B9		STA *FIRT	;clear cycle counter
F9FE- A5 B7		LDA *PCNTR	;get bit counter
FA00- 10 31		BPL RAD4	;if not all done
FA02- 30 C9		BMI JRAD2	;else do parity
FA04- A2 A6	RADP	LDX #\$A6	;get interbyte timing
FA06- 20 45 F9		JSR STT1	;start timer1
FA09- A5 B1		LDA *PRTY	;get parity bit
FA0B- D0 C3		BNE SRER	;if not zero, set parity error
FA0D- 4C 00 E6	RADBK	JMP PREND	;else return from interrupt
FA10- A5 9C	RAD3	LDA *SVXT	;if 2nd byte cycle, get acc. sp. corr.
FA12- F0 08		BEQ ROUT1	;if zero, clear it
FA14- 30 04		BMI ROUT2	;if under time
FA16- C6 CB		DEC *CMPO	;adapt speed correction
FA18- C6 CB		DEC *CMPO	;undo next increment
FA1A- E6 CB	ROUT2	INC *CMPO	;if over time, adapt speed correction
FA1C- A9 00	ROUT1	LDA #0	;clear accumulated
FA1E- 85 9C		STA *SVXT	;speed correction
FA20- E4 D9		CPX *DATA	;if 2nd cycle is compl. of 1st
FA22- D0 0F		BNE RAD4	;go add bit to byte
FA24- 8A		TXA	;else get it in A
FA25- D0 A9		BNE SRER	;if two zero cycles
FA27- A5 BF		LDA *REZ	;and
FA29- 30 C6		BMI RDBK	;already
FA2B- C9 10		CMP #16	;16 of those
FA2D- 90 C2		BCC RDBK	

FA2F- 85 AB	RAD3G	STA *SYNO	;set sync found
FA31- B0 BE		BCS RDBK	;and do end of byte
FA33- 8A	RAD4	TXA	;get received bit in A
FA34- 45 B1		EOR *PRTY	;add to parity
FA36- 85 B1		STA *PRTY	;and save parity
FA38- A5 CE		LDA *SNSWL	;if sync established
FA3A- F0 D1		BEQ RADBK	
FA3C- C6 B7		DEC *PCNTR	;adapt bit counter
FA3E- 30 C4		BMI RADP	;if all bits done, do end of byte
FA40- 46 D9		LSR *DATA	;else add bit in \$D9
FA42- 66 DF		ROR *MYCH	;to byte in MYCH
FA44- A2 DA		LDX #\$DA	;get interbit timing
FA46- 20 45 F9		JSR STT1	;set it
FA49- 4C 00 E6		JMP PREND	;and read next bit
FA4C- A5 AB	RAD2	LDA *SYNO	;if sync found
FA4E- F0 04		BEQ RAD2Y	
FA50- A5 CE		LDA *SNSWL	;and not established
FA52- F0 07		BEQ RAD2X	;go set time and start reading byte
FA54- A5 B7	RAD2Y	LDA *PCNTR	;else get bit counter
FA56- 30 03		BMI RAD2X	;if all done, read next byte
FA58- 4C DC F9		JMP RAD1	;else add bit to byte
FA5B- 46 CC	RAD2X	LSR *CMP1	;divide cycle time by two
FA5D- A9 93		LDA #\$93	;get timing
FA5F- 38		SEC	;prepare for subtract
FA60- E5 CC		SBC *CMP1	;subtract cycle time
FA62- 65 CB		ADC *CMPO	;add speed correction
FA64- 0A		ASL A	;multiply by two
FA65- AA		TAX	;move time to X
FA66- 20 45 F9		JSR STT1	;and set it
FA69- E6 B2		INC *DPSW	;flag byte received
FA6B- A5 CE	RADQ	LDA *SNSWL	;if sync established
FA6D- D0 11		BNE RADQ2	;move byte read to \$D0
FA6F- A5 AB	RADQ1	LDA *SYNO	;if not established, and not found
FA71- F0 26		BEQ RDBK1	;exit (=read next byte)
FA73- 85 BE		STA *RER	;else set timing
FA75- A9 00		LDA #0	;clear
FA77- 85 AB		STA *SYNO	;sync found flag
FA79- A9 C0		LDA #\$C0	;enable
FA7B- 8D 4E E8		STA IER	;timer1 interrupts
FA7E- 85 CE		STA *SNSWL	;flag sync established
FA80- A5 AB	RADQ2	LDA *SYNO	;get found flag
FA82- 85 CF		STA *DIFF	;flag byte valid
FA84- F0 09		BEQ RADK	;if sync found
FA86- A9 00		LDA #0	;clear
FA88- 85 CE		STA *SNSWL	;sync established flag
FA8A- A9 40		LDA #\$40	;disable
FA8C- 8D 4E E8		STA IER	;timer1 interrupts
FA8F- A5 DF		LDA *MYCH	;get byte read
FA91- 85 DD		STA *OCHAR	;in current cassette byte
FA93- A5 BE		LDA *RER	;get timing
FA95- 05 BF	RADR1	ORA *REZ	;and 0/1 balance
FA97- 85 D0		STA *PRP	;and store combined errors
FA99- 4C 00 E6	RDBK1	JMP PREND	;then exit
		.FI "BASIC4.OF.M10"	

1833 33B0-4BE3 BASIC4.OF.M10

```

;name BASIC4.OF.M10
;*****
;*          *
;* Store byte read from cassette   *
;* in RAM and test block size.    *
;*          *

```

```

*****  

FA9C- 20 C9 FB RADJ   JSR NEWCH      ;prepare for next byte  

FA9F- 85 B2           STA *DPSW      ;clear byte received flag  

FAA1- A2 DA           LDX #$DA      ;set interbyte timing  

FAA3- 20 45 F9       JSR STT1      ;and start timer 1  

FAA6- A5 DE           LDA *FSBLK    ;get number of passes to be read  

FAA8- F0 02           BEQ RD15      ;if some  

FAAA- 85 BD           STA *SHCNL    ;store counter  

FAAC- A9 0F           RD15        LDA #$0F      ;get count bits  

FAAE- 24 C2           BIT *RDFLG    ;if counting  

FAB0- 10 17           BPL RD20      ;check for load  

FAB2- A5 CF           LDA *DIFF     ;if EOT and a valid byte  

FAB4- D0 0C           BNE RD12      ;flag scan  

FAB6- A6 DE           LDX *FSBLK    ;else if last pass  

FAB8- CA              DEX          ;  

FAB9- D0 0B           BNE RD10      ;exit  

FABB- A9 08           LDA #$08      ;if other pass, flag  

FABD- 20 C4 FB       JSR UDST      ;short block in ST  

FAC0- D0 .04          BNE RD10      ;and exit  

FAC2- A9 00           RD12        LDA #0        ;flag scan  

FAC4- 85 C2           STA *RDFLG    ;in cassette flag  

FAC6- 4C 00 E6       RD10        JMP PREND    ;and exit  

FAC9- 70 31           RD20        BVS RD60     ;if loading check for sync found  

FACB- D0 18           BNE RD200    ;if no sync, adapt counter  

FACD- A5 CF           LDA *DIFF     ;if valid byte  

FACF- D0 F5           BNE RD10      ;flag end of tape  

FAD1- A5 D0           LDA *PRP      ;else if any errors  

FAD3- D0 F1           BNE RD10      ;exit  

FAD5- A5 BD           LDA *SHCNL    ;if no errors, get cycle count  

FAD7- 4A              LSR A         ;get LSbit in carry  

FAD8- A5 DD           LDA *OCHAR    ;get current cassette byte  

FADA- 30 03           BMI RD22      ;if MSbit set, test for odd cycle  

FADC- 90 18           BCC RD40      ;if even cycle, exit  

FADE- 18              CLC          ;else skip next instruction  

FADF- B0 15           RD22        BCS RD40     ;if odd cycle, exit  

FAE1- 29 0F           AND #$0F      ;else get 4 LSbits  

FAE3- 85 C2           STA *RDFLG    ;in flag and count  

FAE5- C6 C2           RD200       DEC *RDFLG    ;decrease counter  

FAE7- D0 DD           BNE RD10      ;if not all done, exit  

FAE9- A9 40           LDA #$40      ;else flag LOAD  

FAEB- 85 C2           STA *RDFLG    ;in flag  

FAED- 20 BB FB       JSR RD300    ;move start address to ($07)  

FAFO- A9 00           LDA #0        ;set character count  

FAF2- 85 C3           STA *SHCNH    ;before block to zero  

FAF4- F0 00           BEQ RD10      ;and exit  

FAF6- A9 80           RD40        LDA #$80      ;flag  

FAF8- 85 C2           STA *RDFLG    ;end of tape  

FAFA- D0 CA           BNE RD10      ;and exit  

FAFC- A5 CF           RD60        LDA *DIFF     ;if byte not valid  

FAFE- F0 0A           BEQ RD70      ;check for end address  

FB00- A9 04           LDA #$04      ;else flag  

FB02- 20 C4 FB       JSR UDST      ;short block in ST  

FB05- A9 00           LDA #0        ;flag scan  

FB07- 4C 88 FB       JMP RD161    ;and test for long block  

FB0A- 20 0B FD       RD70        JSR WRT62    ;check if end address reached  

FB0D- D0 03           BNE fb12     ;if so  

FB0F- 4C 89 FB       JMP RD160    ;flag end of tape  

FB12- A6 B0           fb12        LDX *SHCNL    ;else get pass counter  

FB14- CA              DEX          ;if last pass  

FB15- F0 20           BEQ RD58      ;correct possible pass1 error  

FB17- A5 90           LDA *VERCK    ;else get load/verify  

FB19- F0 0C           BEQ RD80      ;if verify  

FB1B- A0 00           LDY #0        ;get offset zero  

FB1D- A5 DD           LDA *OCHAR    ;get current cassette byte

```

FB1F- D1 C7		CMP (SAL),Y	;compare with RAM
FB21- F0 04		BEQ RD80	;if not equal
FB23- A9 01		LDA #\$01	;flag
FB25- 85 00		STA *PRP	;bit error
FB27- A5 00	RD80	LDA *PRP	;get error flag
FB29- F0 4C		BEQ RD59	;if no error, adapt pointer and exit
FB2B- A2 3D		LDX #\$3D	;else get max. number of errors
FB2D- E4 C0		CPX *PTR1	;if more
FB2F- 90 3F		BCC RD55	;set unrecoverable read error, exit
FB31- A6 C0	RD56	LDX *PTR1	;else get offset in error log
FB33- A5 C8		LDA *SAH	;get hi address
FB35- 9D 01 01		STA BAD+1,X	;save in log
FB38- A5 C7		LDA *SAL	;get lo also
FB3A- 9D 00 01		STA BAD,X	;in log
FB3D- E8		INX	;point to
FB3E- E8		INX	;next entry in log
FB3F- 86 C0		STX *PTR1	;and set index
FB41- 4C 77 FB		JMP RD59	;then increment pointer, exit
FB44- A6 C1	RD58	LDX *PTR2	;get number of errors, pass2
FB46- E4 C0		CPX *PTR1	;if same as in pass1
FB48- F0 37		BEQ RD90	;adapt pointer and exit
FB4A- A5 C7		LDA *SAL	;else get address lo
FB4C- DD 00 01		CMP BAD,X	;if not same as pass1 address lo
FB4F- D0 30		BNE RD90	;adapt pointer and exit
FB51- A5 C8		LDA *SAH	;else if hi
FB53- DD 01 01		CMP BAD+1,X	;not same
FB56- D0 29		BNE RD90	;adapt pointer and exit
FB58- E6 C1		INC *PTR2	;else adapt pointer
FB5A- E6 C1		INC *PTR2	;in log
FB5C- A5 9D		LDA *VERCK	;get load/verify flag
FB5E- F0 0C		BEQ RD52	;if verifying
FB60- A5 DD		LDA *OCHAR	;get current cassette byte
FB62- A0 00		LDY #0	;and offset zero
FB64- D1 C7		CMP (SAL),Y	;compare with RAM
FB66- F0 19		BEQ RD90	;if same, increment pointer
FB68- A9 01		LDA #1	;else flag error
FB6A- 85 00		STA *PRP	;error
FB6C- A5 DD	RD52	LDA *PRP	;get error
FB6E- F0 07		BEQ RD59	;if none store byte
FB70- A9 10	RD55	LDA #\$10	;else get bit for read error
FB72- 20 C4 FB		JSR UDST	;add it to ST
FB75- D0 0A		BNE RD90	;and adapt pointer
FB77- A5 9D	RD59	LDA *VERCK	;get load/verify flag
FB79- D0 06		BNE RD90	;if loading
FB7B- A5 DD		LDA *OCHAR	;get byte
FB7D- A0 00		LDY #0	;and offset zero
FB7F- 91 C7		STA (SAL),Y	;move byte to RAM
FB81- E6 C7	RD90	INC *SAL	;adapt lo byte of pointer
FB83- D0 33		BNE RD180	;if page crossed
FB85- E6 C8		INC *SAH	;adapt hi also
FB87- D0 2F		BNE RD180	;and and exit
FB89- A9 80	RD160	LDA #\$80	;flag EOT
FB8B- 85 C2	RD161	STA *RDFLG	;in flag byte
FB8D- A6 DE		LDX *FSBLK	;get pass counter
FB8F- CA		DEX	;if it was last pass
FB90- 30 02		BMI RD167	;adapt counter
FB92- 86 DE		STX *FSBLK	;else adapt pass counter
FB94- C6 BD	RD167	DEC *SHCNL	;adapt pass counter
FB96- F0 08		BEQ RD175	;if not last pass
FB98- A5 C0		LDA *PTR1	;get number of pass1 errors
FB9A- D0 1C		BNE RD180	;if any, exit
FB9C- 85 DE		STA *FSBLK	;else set last block read
FB9E- F0 18		BEQ RD180	;and exit
FBA0- 20 C0 FC	RD175	JSR TNIF	;switch cassette motors off

```

FBA3- 20 BB FB      JSR RD300      ;move start address again
FBA6- A0 00      LDY #0          ;clear number of characters
FBA8- 84 C3      STY *SHCNH    ;before block
FBAA- 20 F9 FC      JSR VPRTY    ;compute checksum
FBAD- A5 C3      LDA *SHCNH    ;get it
FBAF- 45 DD      EOR *OCHAR    ;if same as last byte read
FBB1- F0 D5      BEQ RD180    ;exit
FBB3- A9 20      LDA #$20      ;else get bit for checksum error
FBB5- 20 C4 FB      JSR UDST     ;add to ST
FBB8- 4C 00 E6  RD180  JMP PREND   ;and exit from interrupt
;***** *
;* Move start address to ($C7). *
;* *
;***** *
FBBB- A5 FC  RD300  LDA *STAH    ;get start address hi
FBBD- 85 C8      STA *SAH     ;in load pointer hi
FBBF- A5 FB      LDA *STAL    ;get lo also
FBC1- 85 C7      STA *SAL     ;and complete pointer
FBC3- 60          RTS         ;then exit
;***** *
;* Add the bits in A to the *
;* Status byte.             *
;* *
;***** *
FBC4- 05 96  UDST   ORA *CSTAT   ;add the bits from status byte
FBC6- 85 96      STA *CSTAT   ;and store the byte
FBC8- 60          RTS         ;then exit
;***** *
;* Prepare for read of byte from *
;* tape.                         *
;* *
;***** *
FBC9- A9 08  NEWCH  LDA #8       ;set up
FBCB- 85 B7      STA *PCNTR   ;bit counter
FBCD- A9 00      LDA #0       ;clear
FBCF- 85 B9      STA *FIRT    ;byte cycle counter
FB01- 85 BE      STA *RER     ;interbyte phase one flag
FB03- 85 B1      STA *PRTY    ;parity byte (only 1 bit used)
FB05- 85 BF      STA *REZ     ;interbyte phase two flag
FB07- 60          RTS         ;then exit
;***** *
;* Write a tone to tape, accor- *
;* to the current bit value in *
;* $DD.                         *
;* *
;***** *
FBD8- A5 DD  WRITE  LDA *OCHAR   ;get character to be written
FBDA- 4A          LSR A        ;move LSbit to carry
FBDB- A9 60      LDA #$60     ;if bit set
FBDD- 90 02      BCC WRT1    ;write short period to tape
FBDF- A9 B0  WRTW  LDA #$B0     ;else get long period
FBE1- A2 00  WRT1  LDX #$00     ;get hi byte also
FBE3- 80 48 E8  WRTX  STA T2L    ;and load
FBE6- 8E 49 E8  STX T2H    ;timer2
FBE9- AD 40 E8  LDA PIA     ;then invert
FBEC- 49 08      EOR #$08     ;the tape
FBEE- 80 40 E8  STA PIA     ;write line
FBF1- 29 08      AND #$08     ;and reflect its status in Z
FBF3- 60          RTS         ;then exit
.FI "BASIC4.OF.M11"

```

1AE9 3380-4E99 BASIC4.OF.M11

```

;name BASIC4.OF.M11
;*****
;* Write a pass to tape.
;*
;* (Entered at each interrupt,
;* IRQ vector number 5).
;*
;* Initially entered at $FBF9.
;*
;*****
F8F4- 38      WRT13 SEC          ;set carry
F8F5- 66 C8    ROR *SAH        ;and set hi byte invalid
;BUG: contents of addresses above $7FFF
;cannot be written to cassette with
;this method of stopping
F8F7- 30 3C    WRTN  LDA *RER        ;then exit
F8F9- A5 BE    WRTN  LDA #$10       ;get parameters
F8FB- D0 12    WRTN  BNE fc0f       ;for 1st interbyte marker
F8FD- A9 10    WRTN  LDX #$01       ;write tone to tape
F8FF- A2 01    WRTN  JSR WRTX      ;if output now zero
FC01- 20 E3 FB WRTN  BNE WRT3      ;set phase one done
FC04- D0 2F    WRTN  INC *RER       ;get address hi byte
FC06- E6 BE    WRTN  LDA *SAH       ;if valid, exit
FC08- A5 C8    WRTN  BPL WRT3      ;else alter IRQ
FC0A- 10 29    WRTN  JMP WRTN1      ;if phase one done,
FC0C- 4C 86 FC WRTN  LDA *REZ       ;and phase two not
FC0F- A5 BF    fc0f   BNE WRTN2      ;write tone for 2nd interbyte marker
FC11- D0 09    WRTN  JSR WRTW      ;if output now zero
FC13- 20 DF FB WRTN  BNE WRT3      ;set phase two done
FC16- D0 1D    WRTN  INC *REZ       ;and exit
FC18- E6 BF    WRTN  BNE WRT3      ;if interbyte done, get current bit
FC1A- D0 19    WRTN  JSR WRITE      ;if output zero (one period written)
FC1C- 20 D8 FB WRTN2 LDA *FIRT      ;get byte cycle counter
FC1F- D0 14    WRTN  BNE WRT3      ;invert it
FC21- A5 B9    WRTN  LDA *FIRT      ;and set it
FC23- 49 01    WRTN  EOR #$01       ;if two cycles done, adapt bit counter
FC25- 85 89    WRTN  STA *FIRT      ;if cycle one, get current bit
FC27- F0 0F    WRTN  BEQ WRT2      ;invert it
FC29- A5 DD    WRTN  LDA *OCHAR     ;and store current byte
FC2B- 49 01    WRTN  EOR #$01       ;get current bit
FC2D- 85 DD    WRTN  STA *OCHAR     ;add to parity
FC2F- 29 01    WRTN  AND #$01       ;and store new parity
FC31- 45 B1    WRTN  EOR *PRTY      ;then exit
FC33- 85 B1    WRTN  STA *PRTY      ;two byte cycles done, get LSbit
FC35- 4C 00 E6 WRT3  JMP PREND     ;adapt bit counter
FC38- 46 00    WRT2  LSR *OCHAR     ;get bit counter
FC3A- C6 B7    WRT2  DEC *PCNTR     ;if last bit done, write parity
FC3C- A5 B7    WRT2  LDA *PCNTR     ;if bits left, exit (=write next bit)
FC3E- F0 30    WRT2  BEQ WRT4      ;if beyond byte, prepare for next one
FC40- 10 F3    WRT2  BPL WRT3      ;enable interrupts
FC42- 20 C9 FB WRTS  JSR NEWCH     ;get header byte counter
FC45- 58      WRTS  CLI           ;if header not finished
FC46- A5 BA    WRTS  LDA *CNTDN     ;clear
FC48- F0 12    WRTS  BEQ WRT6      ;checksum
FC4A- A2 00    WRTS  LDX #0         ;adapt count
FC4C- 86 D9    WRTS  STX *DATA      ;get current pass#
FC4E- C6 BA    WRTS1 DEC *CNTDN     ;get current pass#
FC50- A6 DE    WRTS1 LDX *FSBLK     ;if header is written
FC52- E0 02    WRTS1 CPX #2

```

```

FC54- 00 02          BNE WRT61
FC56- 09 80          ORA #$80      ;add $80 to count
FC58- 85 DD          STA *OCHAR   ;and send count as data
FC5A- 00 09          BNE WRT3     ;and exit (=write count)
FC5C- 20 0B FD          WRT6      ;compare addresses
FC5F- 90 0A          BCC WRT7     ;if not at end yet, do next byte
FC61- 00 91          BNE WRT13    ;if beyond, set hi byte invalid
                                ;BUG: does not save last byte. With BASIC,
                                ;      this is no problem, because the start
                                ;      of variables pointer points AFTER the
                                ;      program.
                                ;      However, together with the stop
                                ;      criterium used, it is impossible to
                                ;      write the contents of $7FFF to cassette.
FC63- E6 C8          INC *SAH      ;if at end, move pointer beyond end
FC65- A5 D9          LDA *DATA    ;get checksum
FC67- 85 DD          STA *OCHAR   ;as current byte
FC69- B0 CA          BCS WRT3     ;and exit (=send checksum)
FC6B- A0 00          LDY #0       ;get offset zero
FC6D- B1 C7          LDA (SAL),Y  ;get byte from program
FC6F- 85 DD          STA *OCHAR   ;store as byte to write
FC71- 45 D9          EOR *DATA    ;add byte to checksum
FC73- 85 D9          STA *DATA    ;and save checksum
FC75- E6 C7          INC *SAL     ;adapt pointer lo
FC77- D0 BC          BNE WRT3     ;if page crossed
FC79- E6 C8          INC *SAH     ;adapt .hi byte also
FC7B- D0 B8          BNE WRT3     ;and exit (=write current byte)
FC7D- A5 B1          LDA *PRTY    ;get parity bit
FC7F- 49 01          EOR #$01     ;invert it
FC81- 85 00          STA *OCHAR   ;store as byte to write
FC83- 4C 00 E6          WRTBK    ;and exit (=write parity byte)
                                WRNC
FC86- C6 DE          WRTN1      ;adapt pass counter
FC88- D0 03          BNE WREND    ;if pass is zero now
FC8A- 20 EB FC          WREND    ;stop cassettes
FC8D- A9 50          JSR TNOF     ;get number of cycles
FC8F- 85 B0          STA *SHCNL   ;in counter
FC91- A2 08          LDX #8       ;set IRQ vector 4
FC93- 78             SEI         ;disable interrupts
FC94- 20 E0 FC          WRTBK    ;set IRQ to 4th vector
FC97- D0 EA          BNE WRTBK   ;and exit (=do new IRQ)
                                ****
                                ;*          *
                                ;* Write an interpass tone to  *
                                ;* tape.                      *
                                ;*          *
                                ;* (Entered at each interrupt, *
                                ;* IRQ vector number 4).      *
                                ;*          *
                                ****
FC99- A9 78          WRTZ      ;get inter pass timing
FC9B- 20 E1 FB          WRTZ    ;set it
FC9E- D0 E3          BNE WRTBK   ;if output zero (one period written)
FCA0- C6 B0          DEC *SHCNL   ;adapt count
FCA2- D0 DF          BNE WRTBK   ;if all periods done
FCA4- 20 C9 FB          WRTBK   ;prepare for next byte
FCA7- C6 C3          JSR NEWCH    ;decrement number of tone blocks
FCA9- 10 D8          BPL WRTBK   ;if not all done, exit
FCAB- A2 DA          LDX #10     ;else set
FCAD- 20 E0 FC          WRTBK   ;IRQ vector five (=write bytes)
FCB0- 58             CLI         ;enable interrupts
FCB1- E6 C3          INC *SHCNH   ;set # of tone block to zero
FCB3- A5 DE          LDA *FSBLK   ;get pass number
FCB5- F0 24          BEQ STKY     ;if zero, terminate cassette

```

```

FCB7- 20 BB FB      JSR RD300          ;else move address
FCBA- A2 09      LDX #$09           ;set 9 bytes as
FCBC- 86 BA      STX *CNTDN        ;header count
FCBE- D0 82      BNE WRTS         ;and write them
;***** *
;*      *
;* Switch from cassette inter-  *
;* rupts to screen refresh       *
;* interrupts.                  *
;*      *
;***** *

FCC0- 08      TNIF   PHP              ;save the flags
FCC1- 78      SEI               ;disable interrupts
FCC2- 20 EB FC      JSR TNOF          ;switch cassette motors off
FCC5- A9 7F      LDA #$7F          ;disable cassette write
FCC7- 8D 4E E8      STA IER           ;and cassette #2 read interrupts
FCCA- A9 3C      LDA #$3C          ;disable
FCCC- 8D 11 E8      STA PIAL1        ;cassette#1 read interrupt
FCCF- A9 30      LDA #$30          ;enable
FCD1- 8D 13 E8      STA PIAS           ;screen refresh interrupts
FCD4- A2 0C      LDX #12           ;and set
FCD6- 20 EO FC      JSR BSIV          ;IRQ vector 6 (default $E455)
FC09- 28      PLP               ;restore the flags
FCDA- 60      RTS               ;and exit
;***** *
;*      *
;* Terminate cassette I/O and    *
;* switch to screen refresh       *
;* interrupts.                  *
;*      *
;***** *

FCDB- 20 CO FC  STKY   JSR TNIF          ;switch to refresh interrupts
FCDE- F0 A3      BEQ WRTBK         ;and exit
;***** *
;*      *
;* Alter the IRQ vector at $90    *
;* dependent on the value of X    *
;* on entry.                     *
;*      *
;***** *

FCEO- BD 4C FD  BSIV   LDA BSIT,X      ;get lo byte of vector
FCE3- 85 90      STA *CINV          ;move it
FCE5- 8D 40 FD      LDA BSIT+1,X     ;get hi also
FCE8- 85 91      STA *CINV+1        ;and complete vector
FCEA- 60      RTS               ;then exit
;***** *
;*      *
;* Switch both cassette motors   *
;* off.                         *
;*      *
;***** *

FCEB- A9 3C      TNOF   LDA #$3C          ;stop cassette#1
FCED- 8D 13 E8      STA PIAS          ;motor
FCFO- AD 40 E8      LDA PIA           ;and
FCF3- D9 10      ORA #$10           ;cassette#2
FCF5- 8D 40 E8      STA PIA           ;motor
FCF8- 60      RTS               ;then exit
;***** *
;*      *
;* Compute checksum in $C3.        *
;*      *
;***** *

FCF9- B1 C7      VPRTY  LDA (SAL),Y    ;get byte from prg
FCFB- 45 C3      EOR *SHCNH        ;add to checksum

```

```

FCFD- 85 C3      STA *SHCNH    ;and store new checksum
FCFF- E6 C7      INC *SAL     ;adapt address
F001- D0 02      BNE VP10    ;if page crossed
F003- E6 C8      INC *SAH     ;adapt hi byte also
F005- 20 0B FD  VP10  JSR WRT62 ;check if end address reached
F008- D0 EF      BNE VPRTY   ;if not, loop
F00A- 60          RTS        ;else exit
;*****
;*                                     *
;* Check if end address reached. *
;*                                     *
;*****
F00B- A5 C8      WRT62     LDA *SAH     ;get current address hi
F00D- C5 CA      CMP *EAH     ;if same as end address hi
F00F- D0 04      BNE WRT64   ;*****
F011- A5 C7      LDA *SAL     ;get current address lo
F013- C5 C9      CMP *EAL     ;compare to end address lo
F015- 60          WRT64     RTS        ;and exit
.FI "BASIC4.OF.M12"

```

11B7 33B0-4567` BASIC4.OF.M12

```

;name BASIC4.OF.M12
;*****
;*                                     *
;* Entry on hardware reset,          *
;* pointed to by the hardware       *
;* reset vector.                   *
;*                                     *
;*****
FD16- A2 FF      START    LDX #$FF    ;get initial value
FD18- 78          SEI       ;disable possible spontaneous interrupt
FD19- 9A          TXS       ;and load stackpointer
FD1A- D8          CLD       ;clear possible decimal mode
FD1B- 20 00 ED  JSR CINT   ;initialize I/O, clear screen
FD1E- A9 FF      LDA #L,READY ;set up
FD20- 85 94      STA *NMINV  ;NMI soft vector
FD22- A9 B3      LDA #H,READY ;to point to
FD24- 85 95      STA *NMINV+1 ;BASIC restart
FD26- A9 78      LDA #L,BRKE  ;point the soft
FD28- 85 92      STA *CBINV   ;BRK vector
FD2A- A9 D4      LDA #H,BRKE  ;to the Machine
FD2C- 85 93      STA *CBINV+1 ;Language Monitor
FD2E- A9 A4      LDA #L,ERROPR ;set USRCMD of MLM
FD30- 8D FA 03  STA USRCMD ;to
FD33- A9 D7      LDA #H,ERROPR ;print ?
FD35- 8D FB 03  STA USRCMD+1 ;in MLM
·FD38- A9 00      LDA #0      ;set no defeat
FD3A- 8D FC 03  STA BOB     ;on IEEE transfers
FD3D- 58          CLI       ;enable interrupts (start clock)
FD3E- AD 10 E8  LDA PIAL    ;get diagnostic sense input
FD41- 30 03      BMI fd46   ;if held low
FD43- 4C 72 D4  JMP CALLE   ;start MLM at call entry
FD46- 4C B6 D3  fd46    JMP INIT    ;else coldstart BASIC
;*****
;*                                     *
;* Entry on Non Maskable Inter-  *
;* rupt, pointed to by the hard- *
;* ware .NMI vector.            *
;*                                     *
;*****
FD49- 6C 94 00  NMI   JMP (NMINV) ;goto soft vector's address
;*****

```

```

        ;*
        ;* Table of 8 IRQ vectors.      *
        ;* Used by the tape routines, to *
        ;* read or write from/to cas-   *
        ;* sette.                      *
        ;*
        ;*****  

FD4C- 00 00    BSIT    .SE 0000      ;(used to be diagnostic vector)
FD4E- 00 00    .SE 0000      ;(used to be diagnostic vector)
FD50- 00 00    .SE 0000      ;(used to be diagnostic vector)
FD52- 00 00    .SE 0000      ;(used to be diagnostic vector)
FD54- 99 FC    .SI WRTZ      ;write block to cassette
FD56- F9 FB    .SI WRTN      ;write bits to cassette
FD58- 55 E4    .SE KEY       ;adapt clock and scan keyboard (default
FD5A- 76 F9    .SI READ      ;read bits from cassette
;*****  

;*
;* Checksum byte (over F-ROM).  *
;*
;*****  

FDSC- F5      CKSUMF   .BY $F5      ;checksum byte
;*****  

;*
;* Area $FD5D-$FF92 is filled   *
;* with $AA bytes.              *
;*
;*****  

.BA $FF93
;*****  

;*
;* The KERNAL:                 *
;* A set of jumps to the major  *
;* parts of the operating sys-  *
;* tem. All disk commands are   *
;* also reached through this   *
;* table.                      *
;*
;*****  

FF93- 4C C7 DA. concat JMP CONCAT    ;Perform CONCAT
FF96- 4C 42 D9  dopen JMP DOPEN     ;Perform DOPEN
FF99- 4C 07 DA  dclose JMP DCLOSE   ;Perform DCLOSE
FF9C- 4C AF D7  record JMP RECORD   ;Perform RECORD
FF9F- 4C D2 D9  format JMP FORMAT   ;Perform HEADER
FFA2- 4C 65 DA  colect JMP COLECT  ;Perform COLLECT
FFA5- 4C 7E DA  backup JMP BACKUP  ;Perform BACKUP
FFA8- 4C A7 DA  DCOPY JMP COPY     ;Perform COPY
FFAB- 4C 77 D9  append JMP APPEND   ;Perform APPEND
FFAE- 4C 0D DB  dsave JMP DSAVE    ;Perform DSAVE
FFB1- 4C 3A DB  dload JMP DLOAD   ;Perform DLOAD
          DCAT  ;                                ;Perform CATALOG
FFB4- 4C 73 D8  DIRCAT JMP CATLOG  ;Perform DIRECTORY
FFB7- 4C 55 DB  rename JMP RENAME  ;Perform RENAME
FFBA- 4C 66 DB  SCRATC JMP SCRATCH ;Perform SCRATCH
FFBD- 4C 95 D9  READD$ JMP GETDS   ;Get DS$ from disk
FFC0- 4C 60 F5  COPEN JMP OPEN     ;Perform OPEN
FFC3- 4C DD F2  CCLOS JMP CLOSE    ;Perform CLOSE
FFC6- 4C AF F7  COIN  JMP CHKIN   ;Set input device
FFC9- 4C FE F7  COOUT JMP CHKOUT  ;Set output device
          CCCHN  

FFCC- 4C A6 F2  CLSCHN JMP CLRCHN ;Restore default I/O
FFCF- 4C 15 F2  INCHR JMP BASIN   ;Input a byte
FFD2- 4C 66 F2  OUTCH JMP BSOUT   ;Output a byte
FFD5- 4C 01 F4  CLOAD JMP LOAD    ;Perform LOAD
FFD8- 4C DD F6  CSAVE JMP SAVE    ;Perform SAVE

```

```

FFDB- 4C F6 F4 CVERF JMP VER ;Perform VERIFY
FFDE- 4C C3 F6 CSYS JMP SYS ;Perform SYS
FFE1- 4C 42 F3 ISCNTC JMP STOP2 ;Check for STOP key
FFE4- 4C 05 F2 CGETL JMP GETIN ;Get a byte
FFE7- 4C A2 F2 CCALL JMP CLALL ;Abort all I/O
FFEA- 4C 68 F7 ffea JMP UDTIM ;Update clock
;*****
;*                                     *
;* Area $FFED-$FFF9 is filled      *
;* with $AA bytes.                 *
;*                                     *
;*****                                *
.BA $FFFA
;*****
;*                                     *
;* The Hard vectors for the 6502   *
;* microprocessor. They are in     *
;* order:                           *
;* Non Maskable Interrupt          *
;* Reset                            *
;* Interrupt ReQuest              *
;*                                     *
;*****                                *
FFFA- 49 FD fffa .SI NMI ;NMI
FFFC- 16 FD fffc .SI START ;Reset
FFFE- 42 E4 fffe .SE PULS ;IRQ

;-----
.FI "BASIC4.OF.M13"

```

15C5 33B0-4975 · BASIC4.OF.M13

```

;name BASIC4.OF.M13
;*****
;*                                     . *
;* External labels.                  *
;*                                     *
;*****                                *
CHANNEL .DE $0010 ;current CMD file number
POKER .DE $0011 ;integer read by GETADR
INDEX .DE $001F ;indirect index
TXTTAB .DE $0028 ;start of PRG pointer
VARTAB .DE $002A ;start of variables pointer
CHRGOT .DE $0076 ;BASIC's CHRGOT routine
TXTPTR .DE $0077 ;CHRGET's pointer
CTIMR .DE $0080 ;the jiffy clock
CINV .DE $0090 ;IRQ RAM vector
CBINV .DE $0092 ;BRK RAM vector
NMINV .DE $0094 ;NMI RAM vector
CSTAT .DE $0096 ;Status byte ST
CRFAC .DE $0099 ;correction clock for CTIMR
STKEY .DE $009B ;last keyboard scan index
SVXT .DE $009C ;tape timing constant
VERCK .DE $009D ;Load/Verify flag: 0=load, 1=verify
NDX .DE $009E ;number of keys in keyboard buffer
C3PO .DE $00A0 ;IEEE buffer flag
INDX .DE $00A1 ;length of current screen line
LXSP .DE $00A3 ;save for cursor row and column
BSOUR .DE $00A5 ;IEEE byte buffer
SYNO .DE $00AB ;tape sync flag
CRSW .DE $00AC ;flag: input from screen or keyboard
XSAV .DE $00AD ;X save in cassette handling
LDTND .DE $00AE ;number of open files

```

DFLTN	.DE \$00AF	;default input device number
DFLTO	.DE \$00B0	;default output device number
PRTY	.DE \$00B1	;tape parity
DPSW	.DE \$00B2	;flag: tape byte received
T1	.DE \$00B4	;leading character in tape buffer
T2	.DE \$00B5	;counter for tape
PCNTR	.DE \$00B7	;bit counter for tape
FIRT	.DE \$00B9	;cycle counter for tape
CNTDN	.DE \$00BA	;number of bytes before tape block
BUFPNT	.DE \$00BB	;index in cassette buffer
SHCNL	.DE \$00B0	;counter for tape
RER	.DE \$00BE	;tape write byte
REZ	.DE \$00BF	;write start or read error flag
PTR1	.DE \$00C0	;number of pass1 read errors
PTR2	.DE \$00C1	;number of pass2 read errors
RDFLG	.DE \$00C2	;cassette read flag: scan/count/load/en
SHCNH	.DE \$00C3	;timer before tape write or checksum
PNTR	.DE \$00C6	;current column number of cursor
SAL	.DE \$00C7	;start address lo for LOAD/SAVE
SAH	.DE \$00C8	;start address hi for LOAD/SAVE
EAL	.DE \$00C9	;end address lo for LOAD/SAVE
EAH	.DE \$00CA	;end address hi for LOAD/SAVE
CMPO	.DE \$00CB	;speed correction
CMP1	.DE \$00CC	;elapsed time
SNSWL	.DE \$00CE	;tape read timer flag
DIFF	.DE \$00CF	;end of byte flag
PRP	.DE \$00D0	;tape read error flag
FNLEN	.DE \$00D1	;Length of current filename
LA	.DE \$00D2	;current file number
SA	.DE \$00D3	;current secondary address
FA	.DE \$00D4	;current primary address
LNMX	.DE \$00D5	;current line number of cursor
TBUF	.DE \$00D6	;pointer to current cassette buffer
TBLX	.DE \$00D8	;current column number of cursor
DATA	.DE \$00D9	;last key input or buffer checksum
FNADR	.DE \$00DA	;Pointer to current filename
OCHAR	.DE \$00DD	;tape write byte or tape byte read
FSBLK	.DE \$00DE	;tape block counter
MYCH	.DE \$00DF	;serial word buffer
CAS1	.DE \$00F9	;cassette #1 motor flag
CAS2	.DE \$00FA	;cassette #2 motor flag
STAL	.DE \$00FB	;2nd start address lo for LOAD and SAVE
STAH	.DE \$00FC	;2nd start address hi for LOAD and SAVE
BAD	.DE \$0100	;tape error log table
BUF	.DE \$0200	;BASIC's input buffer
LAT	.DE \$0251	;table of numbers of open files
FAT	.DE \$025B	;table of primary addresses of open fil
SAT	.DE \$0265	;table of secondary addresses of open f
TAPE1	.DE \$027A	;buffer for cassette #1
TAPE2	.DE \$033A	;buffer for cassette #2
USRCMD	.DE \$03FA	;MLM extension vector
BOB	.DE \$03FC	;IEEE timeout defeat flag
TYPERR	.DE \$B3ED	;Restart BASIC with ERROR
READY	.DE \$B3FF	;Restart BASIC
FINI	.DE \$B4AD	;Restart BASIC
FLOAD	.DE \$B60B	;Execute BASIC program
STXTPT	.DE \$B622	;Perform CLR and rechain
stop	.DE \$B7C6	;Perform STOP
FRMNUM	.DE \$BD84	;Evaluate arithmetic expression
FRMEVL	.DE \$BD98	;Evaluate any expression
CHKCOM	.DE \$BEF5	;Check for a comma
SNERR	.DE \$BF00	;Exit with SYNTAX ERROR
FRESTR	.DE \$C7B5	;Throw away string
GETADR	.DE \$C92D	;Convert FLP accuri to integer in (\$11)

GETBYT .DE \$C8D4	;Read an integer <256 in X
INIT .DE \$D3B6	;Coldstart BASIC
CALLE .DE \$D472	;Call MLM
BRKE .DE \$D478	;BRK entry into MLM
ERROPR .DE \$D7A4	;Print ? in MLM
RECORD .DE \$D7AF	;Perform RECORD
CATLOG .DE \$D873	;Perform CATALOG/DIRECTORY
DOPEN .DE \$D942	;Perform DOPEN
APPEND .DE \$D977	;Perform APPEND
GETDS .DE \$D995	;Get DS\$ from disk
FORMAT .DE \$D9D2	;Perform HEADER
DCLOSE .DE \$DA07	;Perform DCLOSE
COLECT .DE \$DA65	;Perform COLLECT
BACKUP .DE \$DA7E	;Perform BACKUP
COPY .DE \$DAA7	;Perform COPY
CONCAT .DE \$DAC7	;Perform CONCAT
DSAVE .DE \$DB0D	;Perform DSAVE
DLOAD .DE \$DE3A	;Perform DLOAD
RENAME .DE \$DE55	;Perform RENAME
SCRTCH .DE \$DB66	;Perform SCRATCH
OLDCLR .DE \$DBE1	;Clear DS\$
CINT .DE \$E000	;Initialize I/O, clear screen
LP2 .DE \$EOA7	;Get byte from keyboard buffer
LOOPS .DE \$E116	;Get byte from screen or keyboard
PRT .DE \$E202	;Move A to screen
PULS .DE \$E442	;IRQ ROM entrypoint
KEY .DE \$E455	;Normal IRQ vector
PREND .DE \$E600	;exit from interrupt
PIAL .DE \$E810	;PIA1: keyboard row, port A
PIAL1 .DE \$E811	;PIA1: control register, port A
PIAK .DE \$E812	;PIA1: keyboard column, port B
PIAS .DE \$E813	;PIA1: control register, port B
IEEI .DE \$E820	;PIA2: IEEE input, port A
IEEIS .DE \$E821	;PIA2: control register, port A
IEEO .DE \$E822	;PIA2: IEEE output, port B
IEEOS .DE \$E823	;PIA2: control register, port B
PIA .DE \$E840	;VIA: port B
T1L .DE \$E844	;VIA: timer 1, lo
T1H .DE \$E845	;VIA: timer 1, hi
T2L .DE \$E848	;VIA: timer 2, lo
T2H .DE \$E849	;VIA: timer 2, hi
IFR .DE \$E84D	;VIA: interrupt flags
IER .DE \$E84E	;VIA: interrupt enable
.EN	

END MAE PASS

LABELFILE

ACPO0 =F1CD	ACPO1 =F1D2	ACPO3 =F1EE
ACPOS =F1F9	ACPTR =F1C0	APPEND =D977
BACKUP =DA7E	BAD =0100	BASIN =F215
BLNK2 =F62E	BN10 =F224	BN20 =F231
BN30 =F25C	BN32 =F262	BN35 =F263
BO10 =F271	BO20 =F277	BO21 =F278
BOB =03FC	BRKE =D478	BSIT =FD4C
BSIV =FCEO	BSOUR =00A5	BSOUT =F266
BUF =0200	BUFPNT =D0BB	C3PO =D0A0
CALLE =D472	CAS1 =00F9	CAS2 =D0FA
CATLOG =D873	CBINV =D092	CCALL =FFE7
CCCHN =FFCC	CCLOS =FFC3	CGETL =FFE4
CHANNL =0010	CHKCOM =BEF5	CHKIN =F7AF

CHKOUT =F7FE	CHRGOT =0076	CI2 =F1A6
CI4 =F1AB	CINT =E000	CINV =0090
CIOUT =F19E	CK10 =F818	CKSUMF =FD5C
CLALL =F2A2	CLOAD =FFD5	CLOS10 =F2E7
CLOSS =F2E2	CLOSE =F2DD	CLRCH =F2A6
CLRCHN =F2A6	CLSCHN =FFCC	CLSEI =F72F
CMP0 =00CB	CMP1 =00CC	CNTDN =00BA
COIN =FFC6	COLECT =DA65	CONCAT =DAC7
COOUT =FFC9	COPEN =FFCO	COPY =DAA7
CRFAC =0099	CRSW =00AC	CS10 =F87A
CS20 =F883	CS25 =F88B	CS30 =F85E
CS40 =F860	CSAVE =FFD8	CSTAT =0096
CSTE2 =F88C	CSTEL =F857	CSYS =FFDE
CTIMR =008D	CVERF =FFDB	DATA =00D9
DCAT =FFB4	DCLOSE =DA07	DCOPY =FFA8
DFLTN =00AF	DFLTO =00B0	DIFF =00CF
DIRCAT =FFB4	DLOAD =DB3A	DOPEN =D942
DPSW =00B2	DSAVE =DB0D	EAH =00CA
EAL =00C9	ER001 =F175	ERMSG =F5AF
ERR01 =F167	ERROPR =D7A4	ERRPO =F165
ERRP1 =F170	ERRP7 =F16C	ERRS3 =F151
ERRS4 =F158	FA =00D4	FAF =F4D3
FAF20 =F4E0	FAF30 =F4F4	FAF40 =F4F5
FAH =F5E5	FAH30 =F5E8	FAH40 =F614
FAH45 =F612	FAH50 =F5FB	FAH55 =F608
FAT =025B	FCLOSE =F2E2	FINI =B4AD
FIRT =0089	FLOAD =B60B	FNADR =00DA
FNLEN =0001	FOPEN =F565	FORMAT =D902
FR:111 =F536	FRESTR =C7B5	FRMEVL =BD98
FRMNUM =BD84	FSBLK =00DE	GETADR =C92D
GETBYT =C8D4	GETDS =D995	GETIN =F205
IEEI =E820	IEEIS =E821	IEEO =E822
IEEOS =E823	IER =E84E	IFR =E84D
INCHR =FFCF	INDEX =001F	INDX =00A1
INIT =D3B6	ISCNTC =FFE1	ISOUR =F109
ISRO =F128	ISR1 =F11E	ISR2 =F12D
ISR3 =F138	JLTLK =F2C1	JRAD2 =F9CD
JRADJ =F9AB	JTG10 =F259	JTG35 =F243
JTGET =F249	JTP10 =F298	JTP20 =F84B
JX120 =F3:15	JX:150 =F318	JX:170 =F334
JX300 =F7BE	JX305 =F7C0	JX310 =F7C3
JX320 =F7DA	JX330 =F7DF	JX3301 =F7E3
JX340 =F7F0	JX350 =F7F3	JX360 =F828
JX370 =F82D	JX3701 =F831	JX380 =F83D
JX390 =F840	JX600 =F2C3	JX750 =F2AF
JX770 =F2B8	JZ100 =F2CD	JZ101 =F2DC
KEY =E455	LA =00D2	LAT =0251
LD10 =F405	LD100 =F3D4	LD105 =F45C
LD11 =F40D	LD110 =F462	LD112 =F3DD
LD115 =F46C	LD120 =F3E6	LD15 =F356
LD150 =F3E9	LD170 =F3EE	LD20 =F35A
LD205 =F443	LD209 =F429	LD210 =F42E
LD300 =F449	LD40 =F38F	LD410 =F46D
LD420 =F475	LD50 =F3B3	LD60 =F3B5
LD64 =F3B8	LD65 =F3C6	LDAD1 =F6AB
LDAD2 =F67B	LDAD3 =F682	LDTND =00AE
LIST1 =F007	LIST3 =FOFA	LIST4 =FOFF
LISTN =F005	LNMX =00D5	LOAD =F401
LOADNP =F40B	LOOPS =E116	LP2 =E0A7
LXSP =00A3	MS1 =F000	MS10 =F05F
MS11 =F064	MS12 =F06E	MS13 =F074
MS15 =F086	MS16 =F094	MS17 =FOA3
MS18 =FOAA	MS19 =FOAE	MS2 =FOOE
MS21 =F06D	MS22 =F060	MS3 =F017

MS30 =F0B6	MS31 =F0C5	MS4 =F024
MS5 =F032	MS6 =F03D	MS7 =F041
MS8 =F04D	MS9 =F056	MSG =F185
MSG1 =F000	MYCH =000F	NDX =009E
NEWCH =FBC9	NMI =FD49	NMINV =0094
OCHAR =00DD	OLDCLR =DBE1	OP100 =F56E
OP150 =F598	OP160 =FSAD	OP170 =F5C2
OP171 =F5D1	OP172 =F5E2	OP175 =F5E4
OP200 =F5C9	OP35 =F4C0	OP37 =F4BB
OP40 =F4C6	OP45 =F4D0	OP94 =F563
OP98 =F567	OPEN =F560	OPENI =F4A5
OPENIB =F4B2	OUTCH =FFD2	P200 =F53C
PARS1 =F47D	PARS2 =F50D	PCNTR =00B7
PIA =E840	PIAK =E812	PIAL =E810
PIAL1 =E811	PIAS =E813	PNTR =00C6
POKER =0011	PR100 =F52E	PR130 =F558
PR135 =F55D	PR140 =F54D	PR147 =F554
PR150 =F555	PREND =E600	PR060 =F49E
PR070 =F49F	PRP =0000	PRT =E202
PRTY =00B1	PTR1 =00C0	PTR2 =00C1
PULS =E442	RAD1 =F9DC	RAD2 =FA4C
RAD2X =FA5B	RAD2Y =FA54	RAD3 =FA10
RAD3G =FA2F	RAD4 =FA33	RAD5 =F9DF
RADBK =FA0D	RADJ =FA9C	RADK =FA8F
RADP =FA04	RADQ =FA6B	RADQ1 =FA6F
RADQ2 =FA80	RADR1 =FA95	RADS =F996
RADX2 =F9D8	RBLK =F89A	RD1 =F984
RD10 =FAC6	RD12 =FAC2	RD15 =FAAC
RD160 =FB89	RD161 =FB8B	RD167 =FB94
RD175 =FB80	RD180 =FB88	RD20 =FAC9
RD200 =FAE5	RD22 =FADF	RD3 =F998
RD300 =FB88	RD40 =FAF6	RD52 =FB6C
RD55 =FB70	RD56 =FB31	RD58 =FB44
RD59 =FB77	RD60 =FAFC	RD70 =FB0A
RD80 =FB27	RD90 =FB81	RDBK =F9F1
RDBK1 =FA99	RDFLG =00C2	READ =F976
READD5 =FFBD	READY =B3FF	RECORD =D7AF
RENAME =D855	RER =008E	REZ =00BF
RJDJ =F9AE	ROUT1 =FA1C	ROUT2 =FA1A
RSTOR =F29C	SA =00D3	SAH =00C8
SAL =00C7	SAT =0265	SAVE =F6DD
SCATN =F148	SCRATC =FFBA	SCRTCH =DB66
SECND =F143	SHCNH =00C3	SHCNL =00BD
SNERR =BF00	SNSWL =00CE	SPMSG =F349
SRER =F9D0	STAHC =00FC	STAL =00FB
START =FD16	STKEY =009B	STKY =FCDB
STOP =F343	STOP1 =F335	STOP2 =F342
STOP3 =F942	STT1 =F945	STT2 =F95A
STT3 =F961	STXTPT =B622	SV10 =F6E7
SV100 =F742	SV105 =F755	SV20 =F6EC
SV3 =F6E0	SV30 =F717	SV5 =F6E3
SV50 =F72C	SV60 =F6CC	SVXT =009C
SYNO =00AB	SYS =F6C3	T1 =00B4
T1H =E845	T1L =E844	T2 =00B5
T2H =E849	T2L =E848	TALK =F0D2
TAPE =F8E0	TAPE1 =027A	TAPE2 =033A
TAPEH =F619	TBLX =0008	TBUF =0006
TH20 =F652	TH30 =F664	TKATN =F198
TKSA =F193	TNIF =FCC0	TNOF =FCEB
TP20 =F8FB	TP30 =F905	TP32 =F907
TP35 =F909	TP40 =F913	TP50 =F92A
TRD =F8A3	TRD2 =F8C4	TRD3 =F8C7
TSTOP =F935	TWAIT =F92B	TWRT =F8CE
TWRT2 =F8D5	TXTPTR =0077	TXTRT =F355

TXTST =F351	TXTTAB =0028	TYPERR =B3ED
UD10 =F770	UD20 =F77A	UD30 =F784
UD40 =F786	UD45 =F794	UD50 =F79B
UD60 =F7A1	UD65 =F7AB	UD70 =F7AC
UDST =FBC4	UDTIM =F768	UNLSN =F1B9
UNTLK =F1AE	USRCMD =03FA	VARTAB =002A
VER =F4F6	VER10 =F508	VERCK =009D
VP10 =FD05	VPRTY =FCF9	WBLK =F8CB
WREND =FC8D	WRITE =FB08	WRNC =FC86
WRT1 =FBE1	WRT13 =FBF4	WRT2 =FC38
WRT3 =FC35	WRT4 =FC7D	WRT6 =FC5C
WRT61 =FC58	WRT62 =FD08	WRT64 =FD15
WRT7 =FC6B	WRTBK =FC83	WRTN =FBF9
WRTN1 =FC86	WRTN2 =FC1C	WRTS =FC42
WRTS1 =FC4E	WRTW =FBDF	WRTX =FBE3
WRTZ =FC99	XSAV =00AD	ZZ10 =F6AA
append =FFAB	backup =FFA5	colect =FFA2
concat =FF93	dclose =FF99	dload =FFB1
dopen =FF96	dsave =FFAE	f069 =F069
f35d =F35D	f363 =F363	f36e =F36E
f3c1 =F3C1	f5b2 =FSB2	f695 =F695
f6fd =F6FD	f7fa =F7FA	f847 =F847
fb12 =FB12	fc0f =FC0F	fd46 =FD46
ffea =FFE4	ffffa =FFFFA	ffffc =FFFFC
fffe =FFF4	format =FF9F	record =FF9C
rename =FFB7	stop =B7C6	
//0000,0000,0000		
]		

```
*****
* Cross reference listing.
*
*****
```

On the following pages, the cross reference listing of the BASIC 4.0 ROMs is given. The listing consists of 6 separate parts:

1. Cross reference from the B-, C-, D- and F-ROMs.
2. Cross reference from the E-ROM for 9" 40 column N-keyboard models.
3. Cross reference from the E-ROM for 9" 40 column B-keyboard models.
4. Cross reference from the E-ROM for 12" 40 col. N-keyb. models (FAT40).
5. Cross reference from the E-ROM for 12" 80 col. B-keyb. models (80XX).
6. Cross reference from the E-ROM for 12" 80 col. N-keyb. models (Graphics 80).

In this way, all ROMs are covered. This first cross reference listing was made using the B-ROM 1465-23. The listing for the B-ROM 1465-19 is not given, due to its bugs and for space reasons.

The listing has the following format:

First a label is given, followed by the address of that label. The label name corresponds with the name in the assembler listing. For some locations, multiple labels exist, in which case the address is shown twice or three times, with the addresses that reference that particular address with that particular label name.

If the address has no label assigned to it, or the official CBM label name was not known, no label is stated. In this case, the label in the source listing is the hex address, written in lower case without a preceding dollar sign.

After the label and the reference address the addresses are printed that make use of that reference address. Each address is preceded with an extra token, that indicates the kind of use:

Token Kind of use

- | | |
|------|--|
| none | Jumped to by a relative branch. |
| # | Jumped to via a JMP absolute or a JMP indirect instruction. |
| \$ | Jumped to via a JSR absolute instruction. |
| : | Location is read, and contents are left unchanged. |
| > | Location is written to, or accessed in read/modify/write fashion; previous data is therefore lost. |

```
*****
*
* Cross reference listing one. Area $B000-$FFFF, except the area      *
*                      $E000-$EFFF (E-ROM and I/O).                      *
*
*****
```

This cross reference listing covers the ROMs 1465-23, 1465-20, 1465-21 and 1465-22. The ROM 1465-19 is not covered by this listing, for space reasons.

Label Address Referenced by

USRPOK 0000:	>D3BD
	>D3C3
	>D3C5
CHARAC 0003:	>B896 :B89E >B8A0 >BC5D >BC61 >BC69 >C094 :COAF >C5B2 :C5C5
INTEGR 0003:	>B900 :B922 >CE1A :D133 :D1A5
ENDCHR 0004:	>B511 >B56E :B575 >B89A :B89C >B8A2 :B8A8 >BC6E >C09A :COA6 >C5B4 :C5C9
COUNT 0005:	>B425 :B480 :B4A2 >B52F :B550 >B582 >C08B :C092 :C098 :COA4 :COA8 :COAD :COB1 :C2C8 >C334 :C381 :C3AA >C3D1 >C417 >C44F
DIMFLG 0006:	>C130 :C2FC >C341 :C37A :C3B5 :C410
VALTYP 0007:	:B93F :BAC0 :BC50 :B08A :B0DB >BE01 >BE83 :BF97 >C000 >C141 >C160 :C301 >C33A :C4A8 >C4BE >C613 >C8BD
INTFLG 0008:	:B93C :BC88 :BFD4 >C143 >C16E :C2FE >C33D
DORES 0009:	>B4FF :B517 >B567 >B65F :B68E >B692 :B6B1
GRBFGL 0009:	>C02E :C032 >C034 :C03B >C61D :C65D >C665
SUBFLG 000A:	>B61F >B6E0 :C168 :C17E >C189 >C4E7 >C511
INPFLG 000B:	:BB4C >BC0B :BC2D :BC54 :BCDE
DOMASK 000C:	>BE66 :C115
TANSGN 000C:	:D2B5 >D2B9 >D2D7 :D2EE
DSDESC 000D:	>B5FA :B9C4 :BF11 :BFFC >D3DC :D991 >D997 :DBE3 >DBF5 000E: :B9CA :B970 :BFCC :C029 :C037 >D99C >D9A9 >D9C7 >D9CD 000F: :B9BE :BF16 :BFCE >D99E
CHANNL 0010:	:B3CF >B3D8 >BA9C :BADB :BAE4 :BB3A :BB61 >BB8F :BB9F >BBAF :BBB4 >BBBB :BB08 :BBF5 :BC3F :BCEB :BF2E >D308 >F940
LINNUM 0011:	:B490 :B5BE :B651 >B659 :B676 >B8F8 :B90A :B912 >B914 >B91C :B920 >B924 >D3F0 >D400 >D40A :D40C >D411 :D413 :D417 :DA40
POKER 0011:	>C93E :C946 :C94E >C952 >C960 :C976 #F6C9
LINNUM 0012:	:B492 :B5B3 :B653 >B65B :B672 :B838 >B8FA :B902 :B918 >B91A >B91E >B928 >D3F2 >D404 :D419 :DA52
POKER 0012:	>C940 :C943 >C955
TEMPPT 0013:	>B610 :C5F3 >C61A >C819 >D3E8
LASTPT 0014:	>C615 :C815 >C810 0015: :C811 >D3DA
INDEX 001F:	>B35C :B36B :B376 >B398 :B39B >B4BA :B4C1 :B4C8 :B4CE >B4D3 >B4D9 >B4DB >B539 >B53E :B548 :B584 >B587 :B590 >B6BE >B6C8 :B6CE :B6D5 >B720 >B904 >B90D >B910 :B916 :B9AB >BA17 >BA27 >BA2C >BA30 >BA3E >BA42 :BA50 :BA56 :BA5A >BA74 :BA7B >BA7D :B82A >BDE4 :BDE7 >BE34 >BE3B #BE53 >C228 :C22E :C232 :C236 :C23D :C24A :C251 >C253 :C265 :C26B :C271 :C293 >C295 :C3CF :C449 >C477 >C620 :C635 >C64C >C650 :C6BE >C6D8 :C705 >C748 >C79A :C7A3 >C7BC >C7CD >C7D1 :C7D7 >C7DE :C800 :C804 :C808 >C80B :C853 >C855 :C8C8 >CBC2 :CBC8 :CBCD :CBD2 :CBD7 :CBE6

Label Address Referenced by

INDEX 001F: >CCD8 :CCDE :CCE3 :CCE8 :CCED :CCF6 >CD00 >CD15 >CD1A
 >CD1F >CD28 >CD2D :DD77 :DE5C :DE82 :F544
 INDEX1 001F: >B434 :B45B :B462 :C8F3 :C8F8
 INDEX 0020: >B4BC :B4D6 >B4DD >B535 >B542 >B58B >B6BA >B6CC >B722
 >BA19 >BA34 >BA76 >BA81 >BE37 >BE3F >C22A >C257 :C259
 >C299 >C62F >C79C >C7BE :C7D9 >C7E0 >C80D >C859 >C8C4
 >CCDA >CD0F :DE84 :F548
 INDEX1 0020: >B430 >B45F >B469 :C8FC
 INDEX2 0021: >B444 >B464 >B037 >C8FA :C907 >C90B >C916 >CD91 :CD97
 :&CD9D :CDA7 :CDB0 :CDB7 :CDC2
 *0022: >B438 >B458 >B46B >C903 >CD93
 RESH0 0023: >CB6B :CBAD >CBB1 >CBB3 :CCC5
 RESMOH 0024: >CB6D :CBA7 >CBAB >CBB5 :CCC9
 RESMO 0025: >C462 >C47B :C49A >CB6F :CBA1 >CBA5 >CBB7 :CCCD
 RESLO 0026: >C480 :C49E >CB71 :CB9B >CB9F >CBB9 >CC7A :CCD1
 TXTTAB 0028: :B4B6 :B5A3 >B5D7 >B5DA :B5DC :B623 :B7B8 :B847 >D3EC
 >D426 >D428 :D434 :F6D8
 0029: :B4B8 :B5A5 :B5E3 :B629 :B7BC :B849 >D3EE :D439 :F6D4
 VARTAB 002A: :B432 :B440 >B442 :B452 :B47C >B49E >B5E1 :B5FF :B9EB
 :C18B >F43E :F6CC
 002B: :B446 >B44A :B484 >B4A0 >B5E7 :B601 :B9E3 :C18D >F43A
 :F600
 ARYTAB 002C: >B603 :C197 :C1F2 >C214 :C343
 002D: >B605 :C193 :C1F4 >C216 :C345
 STREND 002E: >B353 :B49A >B607 :C1FA :C224 :C34F >C3E7 :C401 :C485
 :C640
 002F: >B355 :B49C >B609 :C1FC :C220 :C34B >C3E9 :C409 :C4BA
 :C63A
 FRET0P 0030: :B3A6 :B3C8 >B5F4 :B90D :BA68 :C27D :C4B3 :C623 >C656
 :C684 :C6EA >C71A :C7E8 :C7EE >C7F0 >C7F6 >D41F
 0031: :B3A0 :B3C2 >B5F6 :B904 :BA62 :C277 :C4B8 :C628 >C658
 :C67E :C6E4 >C71C :C7E4 >C7F4 >C7FA >D421
 FRESPC 0032: >C644 :C652 >C676 >C6C4 >C70B :C716 >C7A5 :C7AC >C7AE
 0033: >C646 :C654 >C67C >C6C6 >C70D :C718 >C7B2
 MEMSIZ 0034: :B5F0 :C66E >D41B :D431
 0035: :B5F2 :C670 >D41D :D437
 CURLIN 0036: :B706 >B76C :B708 >B803 :B821 >B878 >B85A >B05E :CF81
 0037: :B3F7 >B415 :B703 >B771 :B7CF :B7DA >B805 :B81E :B836
 >B87B >B85C >B063 :C4CF :CF7F
 OLDDLIN 0038: >B70C :B7FF
 0039: >B7DE :B801
 OLDTXT 003A: >B755 >B7D4 :B7F9 :B871
 003B: >B61D >B757 >B7D6 :B7F2 :B873
 DATLIN 003C: :B856 >BCC5
 003D: :B858 >BCCB
 DATPTR 003E: >B7C1 :BC02
 003F: >B7C3 :BC04
 INPPTR 0040: >BC0D :BC20 >BC9D :BCDA :BCE7
 0041: >BC0F :BC22 >BC9F :BCDC
 VARNAM 0042: :BF93 >C132 :C170 >C172 :C19B :C1CB :C29F :C30B >C312
 :C358 :C398 :C457
 0043: >BF95 >C17B :C1A1 :C1CD :C2A4 :C308 >C315 :C35C :C3A0
 :C45C
 VARPNT 0044: >C2C3 >C46C :C474 :C4FB >C536 :C541 :C55E
 VARPNT 0045: >C2C5 >C471 :C4F8 >C53E :C547 :C55B
 FDECPT 0044: >D053 :D06A
 LSTPNT 0046: >B67C :B683 >B6B6

Label Address Referenced by

FORPNT 0046: >B335 :B341 :B744 >B933 >B959 >B95E :BA25 :BA2E >BA48
 >BC14 >BD22 :BD45 :CD06
 FORPNT 0047: :B32E >B33A :B741 >B861 >B935 :B965 :BA2A :BA32 >BC16
 >BD24 :BD47 :CD08
 ANDMSK 0046: >C966 :C97A
 EORMSK 0047: >C972 :C978
 VARTXT 0048: >BC1C :BCA1
 OPPTR 0048: :BDF8 >BE62
 VARTXT 0049: >BC1E :BCA3
 OPMASK 004A: >B0B0 :BDC3 :BDC5 >BDC9 :BDD1 >BEOF :BE25 >C0D2
 DEFPN 004B: >C516 :C523 >C52D :C534 :C53A :C552 :C557 >C565 >C57B
 >C57F >C583 >C587 >C58B
 GRBPNT 004B: >C674 :C698 :C69F :C6B4 :C6B9 :C6CF :C6E0 :C6F5 :C729
 >C730 :C746
 DEFPN 004C: >C518 :C520 >C530 >C568
 GRBPNT 004C: >C67A :C6E2 :C72B >C732
 DSCPNT 004D: :B9FD >BA13 :BA46 >C5A2 :C775 :C839 :C83E :C848 :C866
 :C889 >C8A4
 004E: :B9FF >BA15 >C5A4 :C777 :C84A >C8A7
 FOUR6 0050: >D3D3
 JMPER 0051: \$C080 >D3BB
 0052: >C079 >C89D
 SIZE 0052: :C8A9
 OLDOV 0053: >C07E >C9A7 >C9C3 :C9E8 :CA34 >D194 :D1BD
 TEMPF1 0054: :B3AF
 HIGHDS 0055: :B374 >B378 >B382 >B389 >B482 >C208 :C20F >C218 :C21C
 >C66C :C68A >C6A5 :C6AC >C6B0
 HIGHDS 0056: >B37C >B380 >B48B >C20A :C211 >C21A :C21E
 ARYPNT 0055: :C238 >C23A :C25F >C204 :C3D0 >C3F4 :C46A
 ARYPNT 0056: :C23F >C241 :C25B >C2D6 :C305 >C3D9 >C3F8 >C3FE :C46F
 HIGHTR 0057: :B358 :B368 >B36D :B380 >B387 >B47E >C1FE >C26D :C283
 >C287 :C28A >C28E
 0058: :B35F >B371 >B38B >B486 >C200 >C275
 LOWDS 005A: >BFB7 >C484 >C4A3 >CE2D :CE80 >CEA5 >CFBF >CFDA >CFE1
 :CFED >D002 >D05E
 TENEXP 005B: >BFB0 :CE72 :CE7D >CE82 >CE8B >CE94 :CEC7 :CED9
 >CEE4 >D000 :D08B :D094
 LOWTR 005C: :B35A :B42E :B43A :B43D :B450 >B4A8 >B5A9 :B5AD :B5B5
 :B5C1 :B5C8 :B5CC :B661 :B66C :B670 :B697 :B69C :B6A0
 >B6A2 :B850 >C191 :C19D :C1A4 :C1AD >C1F6 >C2A1 >C2A6
 >C2AB >C2AE >C2B1 >C2B4 >C2B7 :C2B9 :C2CD >C347 :C355
 :C35E :C363 :C366 :C36A :C385 >C39A >C3A2 >C3AF >C3C2
 >C3C6 :C403 >C407 >C40E :C415 :C427 :C42F :C479 :C47E
 GRBTOP 005C: >C672 >C6B6 >C6BB >C6D1 :C738 >C73F
 DPTFLG 005C: >CE77 :CE79 :CEA1
 LOWTR 005D: :B361 :B436 :B44C >B5AB >B6A4 :B856 >C18F >C1F8 :C2BE
 :C2CF >C349 :C36C :C40C
 GRBTOP 005D: >C678 :C73A >C741 >CD3A
 EXPSGN 005D: >CE64 :CE6B :CECF
 FAC 005E: >B3BA :D1B4 >D1B6
 FACEXP 005E: :B8C2 >B974 :BE50 :BE7E :C2EA :C931 :C993 :C9B1 >C9B7
 >CA2F :CA62 >CA6A >CA6E :CB2A >CB31 :CBEA :CBF2 >CBFE
 >CC2A :CC50 >CC52 >CC57 >CCF8 :CD2B :CD51 :CD61 >CD85
 :CDA3 :CD01 :CE02 >CE16 :CEC2 :CFA7 :D14B :D199 :D270
 >D276 :D334
 DSCTMP 005E: >C0D7 :COEA :COF2 >C5AD >C5D2 :C5FE

Label Address Referenced by

FACHO 005F: :B718 >B71A :BE4D >C00C >C0D9 :C108 >C4C0 >C5A9 >C5BA
 :C602 >C82F >CA06 :CA11 >CA17 :CA4A >CA4E >CA5D >CA72
 :CA83 >CA87 >CAB1 :CB87 :CC61 :CCA8 >CCC7 >CCF3 :CD26
 >CD72 :CD7A :CDAB >CDF6 :CDF8 >CDFA >CE1F >CF83 :D036
 >D03B >D233 :D25E >D262
 FACMOH 0060: :B9A2 :BE4A >C0DB >C4C2 >C5AB >C5BC :C606 >C9FF :CA15
 :>CA1B :CA44 >CA48 >CA5B >CA74 :CA89 >CA8D >CAAD :CB82
 :>CC67 :CCA2 >CCC8 >CCEA :CD1D >CD76 :CDB2 >CE21 >CF85
 :>D02F >D034 >D238 :D264 >D268
 FACMO 0061: :B957 :B9BC :B9C8 :B9DB :B9E9 :B9EF :B9F8 :BE47 :BED4
 :>BF8F :BFDA :BFDE :CD40 :CD62 :CD90 :CDAB :C59E >C60C
 :>C752 :C766 :C7B8 >C9F8 :CA19 >CA1F :CA3E >CA42 >CA59
 :>CA76 :CA8F >CA93 >CAA9 :CB7D :CC60 :CC9C >CCC9 >CCES
 :>CD18 >CD83 :CDB9 >CE23 :D028 >D02D >D23D :D266 >D26A
 INDICE 0061: :C322 >C422 :C445 :C80A :C93A
 FACLO 0062: >B8E2 :B95C :B9E1 :B9F1 :BE44 :BECF >BF91 :BF9F :BFES
 :>C042 :C05F :C096 :COA2 :C5A0 >C60E :C74F :C7BA >C86F
 :>C88F :C893 >C9F1 :CA1D >CA23 :CA38 >CA3C >CA57 >CA78
 :>CA95 >CA99 >CAA5 :CB78 :CC73 :CC96 >CCD3 >CCEO :CD13
 :>CD81 :CDC4 :CE18 >CE25 :D020 >D026 >D242 :D25C >D260
 INDICE 0062: :C327 >C425 :C448 :C8DE :C93C
 FACSGN 0063: :B714 >B976 >B043 :BE20 :BE7A >C0F6 :COFF :C2E6 :C92D
 :>C989 >C98D >C9BB >CA31 :CA7D >CA81 :CBDB >CC07 :CC0A
 :>CCEF :CD22 >CD34 :CD65 >CD89 >CD8E :CD9F :CDC8 :CDD8
 :>CDF2 :CEDD >CEO :CEBE :CF97 >CFA0 :D14F >D153 >D26E
 :>D2A9 >D2B1 >D2EC :D32C
 DEGREE 0064: >D1F6 >D21C
 SGNFLG 0064: >CE38 :CE98
 BITS 0065: :CACB :CD38 >CD49 >CDDF >CDEE >CDFF >D3D6
 ARGEXP 0066: >BE69 :C9AB >CBE8 :CBED :D114 :D1B2 >D1B8
 ARGHO 0067: >BE6C :COBF >COC1 :CA4C :CBAF >CBE3 :CC5F >CC8B :CCA6
 :>CAA
 ARGMOH 0068: >BE6F :CA46 :CBA9 >CBD4 :CC65 >CC89 :CCAO >CCA4
 ARGMO 0069: >BE72 :C0D0 >COE4 :C106 :CA40 :CBA3 >CBCF :CC6B >CC87
 :>CC9A >CC9E
 ARGLO 006A: >BE75 :C0DF >COE6 :CA3A :CB9D >CBCA :CC71 >CC85 :CC94
 :>CC98
 ARGSGN 006B: >BE78 :COBB :C98F :C9B9 >CBD9 :CBDF :CD32 :CEBC :D122
 :>D290
 ARISGN 006C: >BE7C >C991 :C9D9 >CB00 :CC05 >CC25 >CC3D >CECD >D100
 :>D290
 STRNG1 006C: >BA01 :BA08 >C5B6 :C5C1 :C5D5 :C5EC >C75C :C763 :C77F
 :>C78E :C792 :C796
 STRNG1 006D: >BA03 :BA0A >C5B8 :C5D9 :C5E0 :C5EE >C610 >C75F :C781
 FACOV 006D: >BF9D :C9A5 >C9CB :C9D2 >C9EA :CA21 >CA25 >CA36 >CA55
 :>CA7A :CA9B >CAA1 >CABD :CAD8 :CB73 >CBBB >CCBA
 :>CCFA >CD2F >CD3F >CD4E >CD55 >CD87 :CDC0 >CEO :D18B
 :>D1BF >D272
 BUFPTR 006E: >B52B :B552
 FBUPPT 006E: >BFB3 >CFA2 :D009 >D01A :D055 >D068 :D07A
 STRNG2 006E: >B978 >B980 :B982 :B992 >C5D7
 CURTOL 006E: >C3A8 >C3CB :C3EF >C41B :C43C >C44D >C492 >C8EF :C918
 POLYPT 006E: >D1D7 >D1ED :D1F4 :D1F8 >D200 :D207 >D211
 CURTOL 006F: >C394 >C3CD >C3ED >C3FA >C41D :C43A >C453 >C494 >C8F1
 :>C91A
 STRNG2 006F: >C50E
 POLYPT 006F: >D1D9 >D1EF >D1FE :D202 :D209 >D213

Label Address Referenced by

CHRGET 0070: \$B40D \$B647 \$B735 #B79F \$B8EA \$B92A \$BB12 \$BB81 \$BC4D
 \$BD7E \$BDCB \$BE85 #BE9D #BEFD \$BF21 \$C04A \$C145 \$C150
 \$C178 \$C2DD \$C8D1 \$CE40 \$CE4D \$CE66 \$DC99 \$DCE4 \$DD5A
 \$DDA2 \$DDB6 \$DE2C \$DE87
 CHRGOT 0076: \$B63E \$B72E \$B7AC \$B827 \$B8B6 \$B8CB \$BAAS \$BC28 \$BC8D
 \$BCA9 \$BD77 \$BD82 #BF2B \$C125 \$C12D \$C134 \$C32D \$C56A
 \$C871 #C8E0 \$C90D \$C96A \$D7B4 \$D7E1 \$D7F9 \$DC84 \$DD96
 \$DE0F \$F54D \$F558
 TXTPTR 0077: >B409 :B4FB >B531 :B580 >B5A0 >B627 :B6FB :B74D :B75B
 :B761 :B76A :B76F :B774 >B776 :B7CB >B7FB :B81B :B83E
 >B854 >B87E :B888 >B88A :B8A4 >B875 :BC18 >BC24 >BC49
 >BC59 :BC70 :BC99 >BCA5 :BCBE :BCC3 :BCC8 :BCCD >BD68
 :BD98 >BD9E :BE05 >BE0B :BE85 :BEF9 :C501 :C54F >C554
 >C573 :C8EB >C8F5 >C91C :CEDF >D399 >D4BC >DD45 :D04B
 0078: >B40B >B59C >B62D :B6FE :B74F >B77A :B7CD >B7FD :B818
 :B840 >B85A >B881 >B88E >B877 :BC1A >BC26 >BC4B :BC72
 :BC9B >BCA7 >BD6D >BD9C >BE09 :BE87 :C4FE :C54C >C559
 >C576 :C8ED >C8FE >C91E >D390 :D899 >DD49 :F351
 QNUM 0070: \$B9AD
 CTIMR 0080: >B9A4 >F782 :F786 >F794
 008E: >F77E >F794
 008F: >F77A >F794
 CINV 0090: :D49D >D653 >FCE3
 0091: :D4A2 >D64E :F915 :F930 >FCE8
 CBINV 0092: >FD28
 0093: >FD2C
 NMINV 0094: >FD20 #FD49
 0095: >FD24
 CSTAT 0096: :BB0C :BF35 :C017 >D67C :D6B9 >D8A1 :D8B1 :D8BA :D8E5
 >DBF3 >DE9E >F207 :F25C :F381 :F391 >F393 :F39C :F3AE
 >F3B0 :F3BB :F3F2 :F41F >F47F :F4B7 :F4FD >F511 >F572
 >F7B6 :F7F3 >F805 :F840 >F89C :FBC4 >FBC6
 CRFAC 0099: >F768 :F76A >F79D
 009A: >F76E :F774 >F79F
 STKEY 009B: :F335 :F40D :F411 >F7A9
 SVXT 009C: :F9E3 >F9E5 :FA10 >FA1E
 VERCK 009D: >D67E >D850 :F3A3 >F403 :F418 :F46F >F4F8 :F5E5 >F615
 >F89E :FB17 :FB5C :FB77
 NDX 009E: :F200 >F33F >F427
 C3PO 00A0: :C088 :F0E5 >F0E3 :F19E >F1A2
 INDX 00A1: >F22C
 LXSP 00A3: >F21F
 LSTP 00A4: >F21B
 BSOUR 00A5: >F0F0 :F117 >F143 >F193 >F1AB
 SYNO 00AB: >FA2F :FA4C :FA6F >FA77 :FA80
 CRSW 00AC: >F228
 XSAV 00AD: >F233 :F245
 LDTND 00AE: :D933 :DA1D >F2A4 :F2C1 >F31A :F31C :F320 :F56E >F578
 DFLTN 00AF: >F209 :F215 :F2AF >F28E >F7DA
 DFLTO 00B0: :D88D :F267 :F2A6 >F2BA >F828
 PRTY 00B1: :FA09 :FA34 >FA36 >FB03 :FC31 >FC33 :FC7D
 DPSW 00B2: >F8B6 :F9A7 >FA69 >FA9F
 WSW 00B3: >D875 :D929
 XSAV 00B4: >D4DB :D6AD :D70B
 T1 00B4: >F278 :F298 >F4DE >F4EC :F4F0 >F619 :F633 >F650 :F652 >F65E
 RCNT 00B5: >D474 >D47D :D4AF >D4F7 >D506 >D520 >D627 >F4DA :F4E6
 >F4EE >F64C :F65A >F660

Label Address Referenced by

T3	0087:	:F9AE :F9FE >FA3C :FA54 >FBCB >FC3A :FC3C
FIRT	0089:	:F9E7 >F9EB >F9FC >FB0F :FC21 >FC25
CNTDN	008A:	>D88F :D923 >F243 >F255 >F28F >F5E2 >F850 :F852 :FC46 >FC4E >FCBC
SHCNL	008B:	>FAAA :FA05 :FB12 >FB94 >FC8F >FC00
RER	008E:	>F9D4 >FA73 :FA93 >FB01 :FBF9 >FC06
REZ	008F:	>F9D8 >F9DC :FA27 :FA95 >FB05 :FC0F >FC18
PTR1	00C0:	>F8B2 :FB2D :FB31 >FB3F :FB46 :FB98
PTR2	00C1:	>F8B4 :FB44 >FB58 >FB5A
RDFLG	00C2:	>F8AC :FAAE >FAC4 >FAE3 >FAE5 >FAEB >FAF8 >FB8B
SHCNH	00C3:	>F669 >F803 >FAF2 >FBA8 :FBAD >FCA7 >FCB1 :FCFB >FCFD
TRMPOS	00C6:	:BAF0 :BB09 :C4C9
PNTR	00C6:	:F219
SAL	00C7:	>F684 :F68C :F70D :F71C >F724 :FB1F :FB38 :FB4A :FB64 >FB7F >FB81 >FBC1 :FC6D >FC75 :FCF9 >FCFF :FD11
SAH	00C8:	:F690 :F712 >F728 :FB33 :FB51 >FB85 >FBBD >FBF5 :FC08 >FC63 >FC79 >FD03 :FD08
EAL	00C9:	>D6F7 >F3C8 :F43C :F627 :F642 >F66F >F6B8 >F6CE :FD13
EAH	00CA:	>D6FB >F3CC :F438 :F624 :F647 >F672 >F6C0 >F6D2 :F00D
CMPO	00CB:	>F8B0 :F947 :F94C :F955 :F998 :F9B6 :F9BF :F9C7 >FA16 >FA18 >FA1A :FA62
CMP1	00CC:	>F945 :F94F >F951 >F95A >F95D :F968 >F984 :F98E >F990 >F993 >F996 :F9A3 :F9B8 :F9C1 :F9C9 :F9E1 >FA5B :FA60
SNSWL	00CE:	>F8AE :F9D0 :F9F1 :FA38 :FA50 :FA6B >FA7E >FA88
DIFF	00CF:	>FA82 :FAB2 :FACD :FAFC
PRP	00D0:	>FA97 :FAD1 >FB25 :FB27 >FB6A :FB6C
FNLEN	00D1:	>D67A >D6A4 >D8A5 :D8BE :DAD4 >DC4C >DC78 >DD68 >DD83 :&F367 :F3D0 :F453 :F45C :F468 >F481 :F4A9 :F4C0 :F4CC :&F4E0 >F513 >F542 :F5A4 :F654 :F6F6
LA	00D2:	>D7C3 :D873 >D897 :DA14 :DA31 >DC6D >DCCB >F200 :F2E0 >F51E :F563 :F57A
SA	00D3:	>D893 >D93F >D9B8 :DA48 >DA9D :DAA2 >F2DA :F2F6 :F307 :&F365 :F377 >F483 >F49C :F4A5 :F4B0 >F50F >F52C >F534 :&F57F >F583 :F598 :F5D5 >F6F4 :F703 :F72F :F736 :F75D :&F700 :F7E6 :F81E :F834
FA	00D4:	>D677 >D6D8 :D9AB >D9B1 :DA1B >DC82 >DE3E :F0FB :F251 :&F28B >F2D5 :F2EC :F356 >F486 >F494 >F516 >F526 :F588 :&F5D1 :F69D :F6E3 :F7C6 :F810 :F84E :F866 :F87C :F888 :&F8ED
LNMX	00D5:	:F22A
TEUF	00D6:	:F259 >F295 >F29A :F4E8 >F50F :F5E0 :F608 >F62E >F635 >F63A >F63F >F644 >F649 >F65C :F682 >F697 >F6A4 :F6B1 00D7: >F69B >F6A8 :F6BA
TBLX	00D8:	:F21D
DATA	00D9:	>F9EF :FA20 >FA40 >FC4C :FC65 :FC71 >FC73
FNADR	00DA:	>D686 >D6A2 :DAD9 >DC50 >DD6F >DD81 :F462 :F4C6 :F4E4 :&F546 :F658
OCHAR	00DB:	>D682 :DADD >DC54 >DD73 >F54A
FSBLK	00DD:	>FA91 :FAD8 :FB1D :FB60 :FB7B :FBAF :FB08 :FC29 >FC2D >FC38 >FC58 >FC67 >FC6F >FC81
FSBLK	00DE:	>D4C0 >D541 :D5D7 >F8E5 :FAA6 :FAB6 :FB8D >FB92 >FB9C :&FC50 >FC86 :FCB3
MYCH	00DF:	>FA42 :FA8F
CAS1	00F9:	>F8F7
CAS2	00FA:	:D719 :D746 >D74B >F8FE
TMPO	00FB:	:D4EC :D4FE >D512 :D514 >D525 >D539 :D5DE :D60E :D6F5 :&D71C >D760

Label Address Referenced by

STAL	00FB:	>F37F :F3A8 >F3B3 >F3B5 :F3C6 :F621 :F638 >F675 >F68E >F6B3 >F6DA :FBBF
TMPO+1	00FC:	:D4F1 >D529 >D53D :D5E2 :D613 :D6F9 :D749 >D74E >D759
STAH	00FC:	>F38A >F3B9 :F3CA :F61E :F63D >F678 >F692 >F6BC >F6D6 :FB8B
TMP2	00FD:	:D5DC >D8AF :D8C9 >DADB :DAEA >DE03
	00FE:	:D5E0 >D8B8 :D8CB >DADF >DE05
LOFBUF	00FF:	>CF90 >D00E >D017 >D05B >D065 :D07C >D0BA
FBUFFR	0100:	>D09E >D0BF >D765 >D77F :D788
BAD	0100:	>FB3A :FB4C 0101: :B327 :C31E >C329 >D099 >FB35 :FB53 0102: :B332 :B343 :C31A >C324 >D0B0 0103: :B337 :B33C >D0AC 0104: >D0B5 0109: :B040 :B056 010F: :B05B 0110: :B060 0111: :B06A 0112: :B065
STKEND	01FB:	>B556 :B559 >B57A
	01FC:	:B4A5 >D3E3
	01FD:	>B599 >D3E0
	01FE:	>B494
	01FF:	>B497 >BB02
BUF	0200:	:B476 >B4EB :B501 :B544 :B570 :B594 >BAD4 :B8E8 >BC34
PCH	0200:	>D49A >D4F3 :D599 :D655 0201: >B897
PCL	0201:	>D494 >D4EE :D59F :D659
FLGS	0202:	>D48E :D65D
ACC	0203:	>D48A :D661
XR	0204:	>D486 :D664
YR	0205:	>D482 :D667
SP	0206:	>D4A8 :D646 :D66B
INVL	0207:	>D4A4 :D5A8 >D615 :D64B
INVH	0208:	>D49F :D5AE >D610 :D650
LAT	0251::	:DA27 :F2C6 :F2CD :F322 >F325 >F57C
FAT	025B:	:DA22 :F2D2 :F328 >F32B >F58A
SAT	0265:	:D938 :F2D7 :F32E >F331 >F585
TAPE2	033A:	>D7B1 >D7F6 :D457 >DAD6 :DAE3 :DAF1 >DC7A >DE00
DRIVE1	033B:	:DC38 >DC72 >D023
DRIVE2	033C:	:DC41 >DC75 >D026 >DDDE
LREC1	033D:	:DC1F :DC57 >DC62 >DC6F >DCF5 :D037 >D057
PARCHK	033E:	:D80B :D814 :D821 :D82A :D834 :D881 :D956 :D950 :DA72 :DABA :DAB0 :DAB7 :DABD :DB1C >DC6A :DCBD :DCCD >DCD2 :DCD9 :DCF8 >DCFD :D015 :D029 >D02E :D060 :D08E >D093 :D0D0 :D0E1 >D0E6 :D0F8 :DE07 >DE0C :DE23 :DE40 >DE45 :DE62 :DE79 >DE7E
DISKID	033F:	:D9EE :DAFD >DC7D >D04D
DISKID	0340:	:D804
COUNT	0341:	>DBFC >DC02 >D06A :D089
TBUF2	0342:	>D085
TBUFF	0353:	>D964 >DA45 >DAEC >D800 >D807 >DB25 >DC46
	0354:	>DA4A
	0355:	>DA4F
	0356:	>DA54
	0357:	>DA5A
ERRORPR	03FA:	#D4E9 >FD30
	03FB:	>FD35

Label Address Referenced by

B0E.	03FC:	:F151 :F15B >FD3A
	AFFE:	:C076
	AFFF:	:C07B
STMDS	B000:	:B79B
	B001:	:B797
OPTAB	B094:	:B0EB :BE13 :BE2F
	B095:	:BE1E
	B096:	:BE1A
ERRTAB	B200:	:B3E0
FNDFOR	B322:	\$B6E5 \$B863 \$BD26
FFLOOP	B327:	B34D
CMPFOR	B33C:	B330
ADDFRS	B348:	B33F
FFRTS	B34F:	B32C B346
BLTU	B350:	\$B480 \$C20C
BLT1	B374:	B36F
BLTLP	B380:	B385
MOREN1	B384:	B37A B37E B390
DECBLT	B38B:	B366
GETSTK	B393:	\$B6F3 \$B815 \$BDA8
REASON	B3A0:	\$B350 \$C38F \$C3E4
TRYMOR	B3AA:	B3A4
REASAV	B3AE:	B3B2
REASTO	B3B9:	B3B0
REARTS	B3CC:	B3A2 B3A8 B3C4
OMERR	B3CD:	B396 B39D B3C6 B3CA #C436
ERROR	B3CF:	#B4F5 #B7F6 #B870 #BB67 #BD95 #BF02 #C375 #C4D9 #C5FB #C76C #CAB6 #CCC2 #DE29 #DE76
ERRCRD	B3DA:	B3D1
GETERR	B3E0:	B3EB
TYPERR	B3ED:	#F58F
ERRFIN	B3F4:	#B7E8
READY	B3FF:	B3FA #B6A8 #B7EB #D448 #D66F
MAIN	B406:	B411 B479 #B4B3
MAIN1	B41F:	B417
QDECT1	B45A:	B455
MLOOP	B462:	B45D B467 B46E
NODEL	B470:	B42A
NODELC	B48B:	B488
STOLOP	B4A5:	B4AB
FINI	B4AD:	#F440
LNKPRG	B4B6:	\$B473 \$B4B0
CHEAD	B4BF:	B4DF
CZLOOP	B4C7:	B4CA
LNKRTS	B4E1:	B4C3
INLIN	B4E2:	\$B406 #BBFF
INLINC	B4E4:	B4F1
FINLN1	B4F8:	B4E9
CRUNCH	B4FB:	\$B419 \$B422
KLOOP	B501:	B50B B56C
CMPSPC	B50D:	B504
KLOOP1.	B523:	B51D
MUSTCR	B52B:	B525
RESER	B53D:	B54A
RESCON	B544:	B53B B540 B592
GETBPT	B552:	B597
STUFFH	B554:	B508 B50F B519 B521 B529 B573 B577
COLIS	B567:	B561

Label Address Referenced by

NODATT	B569:	B565
STR1	B570:	B57E
STRNG	B579:	B515
NTHIS	B580:	B54E
NTHIS1	B584:	B58E
NTHIS2	B580:	B589
CRDONE	B599:	B55C
FNDLIN	B5A3:	\$B427 \$B63B
FNDLNC	B5A7:	B5CE \$B84B
FNDL01	B5BE:	B5B9
AFFRTS	B5C7:	B5BC
FLINRT	B5D0:	B5AF
FLNRTS	B5D1:	B5B7 B5C3 B5C5 B5D2 B645 B64D
SCRTCH	B5D4:	\$D445
RUNC	B5E9:	\$B470 \$B4AD #B80A
CLEARC	B5F0:	\$B80D
FLOAD	B60B:	#F446
STKINI	B60E:	\$B3ED
STKRTS	B621:	B5EE B636
STXTPT	B622:	\$B5E9 \$F443
GOLST	B638:	B630 B632
LSTEND	B64F:	B641
LIST4	B65D:	B655 B6A6
TSTDUN	B67A:	B674
TYPLIN	B67C:	B678
PRIT4	B683:	B6D7
PLOOP	B687:	B6AB B6AF B6B3
PLOOP1	B694:	B68C
GRODY	B6A8:	B663 B67A B695
QPLOP	B6AB:	B699
RESRCH	B6C5:	B6D2
RESCR1	B6C8:	B6D0
RESCR2	B6CE:	B6CA
PRIT3	B6D4:	. B6C6 B6DC
PRIT3B	B6D5:	B6C3
NOTOL	B6EF:	B6E8
ONEON	B73B:	B733
NEWSTT	B74A:	#B782 #B82D #BD6F
DIRCON	B759:	B753
DIRCN1	B769:	B764
GONE	B77C:	#B41C B778 B7A7
GONE3	B785:	\$B77F #B8D3
GONE2	B787:	#B8E7
GONE4	B795:	B78D
GLET	B7A2:	B789
NORSTS	B7A5:	B75D
SNERR1	B7A9:	B791
RESTOR	B7B7:	\$B60B
RESFIN	B7C1:	B7BE #B.CE2
ISCRTS	B7C5:	B785
STOP	B7C6:	#F346 #F942
STOPC	B7C9:	B7C6
STPEND	B7D8:	#BBF2
DIRIS	B7E0:	B7D2
ENDCON	B7E2:	#B766
GORDY	B7EB:	B7E6
	B7F9:	B7F4
CONTRT	B807:	B7C9 B7EE

Label Address Referenced by

	B800:	B808
RUNC2	B827:	#B810
GOTO	B830:	#B7B4 \$B82A #B800
LUK4IT	B847:	B83A
LUKALL	B848:	B842 B845
GORTS	B85C:	B85D
USERR	B86E:	B84E
SNERR2	B873:	B8E0
RETU1	B876:	B869
DATA	B883:	#B8E5 \$C504
ADDON	B886:	B8C9 \$BCD0
REMRTS	B890:	B88C B8A6 B8AA
DATAN	B891:	\$B6F6 \$B883 \$BCB4
REMN	B894:	\$B833 \$B8C6
EXCHQT	B89C:	B8B1
REMER	B8A4:	B8AF
OKGOTO	B8C2:	B8BB
DOCOND	B8CB:	B8C4
DOCO	B8D3:	B8CE
SNERR3	B8DE:	B908
ONGLOP	B8E2:	B8DC B8F2
ONGLP1	B8EA:	B8E4
ONGRTS	B8F5:	B8FC
LINGET	B8F6:	\$B41F \$B638 \$B64A \$B830 \$B8ED
MORLIN	B8FC:	#B92D
NXTLGC	B92A:	B926
LET	B930:	\$B6E2 #B7A2
QINTGR	B940:	\$BC8A
COPFLT	B961:	B94D
COPSTR	B964:	B94A
INPCOM	B965:	\$BC7F
TIMELP	B978:	B997
NOML6	B992:	B98B
TIMEST	B9A2:	B9A7
TIMNUM	B9AB:	\$B97A \$B984
FCERR2	B9B2:	B970
GOTNUM	B9B5:	B980
GETSPT	B9BA:	B969
DSKX1	B9D2:	B9C6 B9CC
DSKX2	B9D4:	B9C0
QVARIA	B9E1:	B9D8
DNTCPY	B9EF:	B9D6 B9DF B9E5
COPY	B9F6:	#B9CF B9E7 B9ED
COPYC	BA13:	#B9F3
COPYOO	BA2E:	BA21
COPYO1	BA44:	BA39
COPYO2	BA46:	BA4B
STRADJ	BA4E:	\$BA1E \$BA36 \$C7C5
ADJ	BA6C:	BA66
ADJXX	BA70:	#BF1A
ADJ02	BA74:	#BF1E
ADJ00	BA83:	BA7F
ADJ01	BA85:	BA53 BA5C BA60 BA64 BA6A BA72
CMD	BA8E:	\$BA88
SAVEIT	BA98:	BA91
STRDON	BAA2:	BAC2
NEWCHR	BAA5:	BADO
PRINT	BAA8:	#BA9F

Label Address Referenced by

PRINTC BAAA: #BB·15
 FININL BAD2: #B4F8
 CRDO BADF: \$B3DA \$B668 BAA8
 CRFIN BAED: BAE6 \$BB34
 PRTRTS BAEF: BAAA BADD BB28
 COMPRT BAFO: BAB7
 MORCO1 BAF3: BAF5
 TABER BAFD: BAAE BAB3
 ASPAC BB0D: BAFB
 XSPAC BB0E: BB06
 XSPAC2 BB0F: BB1B
 NOTABR BB12: BAB8 BB08
 XSPAC1 BB18: BB10
 STROUT BB1D: \$B3F4 \$B403 \$BB6E #BCF3 #CF90 \$D42E \$D442
 STRPRT BB20: \$BAA2 \$BACA \$BBCA
 STRPR2 BB27: BB32 #BB37
 OUTSPC BB3A: \$BACD \$BB18 \$BBFC
 CRTSKP BB41: BB3C
 OUTQST BB44: \$B3D0 \$BBF9 \$BC43
 OUTDO BB46: \$B3E6 \$B687 \$B6D9 \$BAE1 \$BAEA \$BB2C
 TRMOK BB4C: #BC96
 GETDTL BB56: BB50
 STCURL BB5A: BB54
 SNERR4 BB5E: BB03
 TRMNO1 BB61: BB4E
 DOAGIN BB6A: BB63
 GETTTY BB91: BB7F
 IODONE BBB4: #BA8B \$BBE2
 IORELE BBB6: BBA1
 NOTQTI BBCD: \$BBB1 BBC0
 GETAGN BB05: #BF3B
 BUFFUL BBE8: BBDA BBED
 RTHRTI BBF1: #BF32
 QINLIN BBF5: \$BB05 \$BC46
 GINLIN BFF7: BBF7
 INPCON BC09: BBEB #BF3E
 INPC01 BC0B: \$BB9C
 INLOOP BC11: #BCB1
 QDATA BC3D: BC2F
 GETNTN BC46: BC41
 DATBK BC49: BC3B
 DATBK1 BC4D: BC2B #BCD7
 SETQUT BC61: BC56
 RESETC BC6D: BC5F
 NOWGET BC6E: BC65
 NOWGE1 BC79: BC76
 NUMINS BC85: BC52
 STRDN2 BC8D: #BC82
 TRMOK BC99: BC90 BC94
 DATLOP BCE4: BC3D BC05
 NOWLIN BCCD: BCB9
 VAREND BCDA: BCAC
 VARYD BCE5: BCE0
 INPRTS BCF6: BCE9 BCED
 GETFOR BD1F: BD19 \$BD81
 STXFOR BD22: BD1D
 ERRGOS BD2D: BCC0
 HAVFOR BD2F: BD29

Label Address Referenced by

NEWSGO BD6F: BD7C
 LOOPDN BD72: B059
 FRMNUM BD84: \$B711 \$B738 \$C561 \$C8D4 \$C921 \$F6C3
 CHKNUM BD87: \$B70E \$BDF0 \$BE5F #C083 \$C2E3 \$C4EC #C51A \$C529 \$C58E
 CHKSTR BD89: \$C05A \$C758 \$C7B5
 CHKVAL BD8A: \$B947 \$C0B6
 CHKOK BD90: BD91
 DOCSTR BD91: 'B08C
 CHKERR BD93: B08E
 ERRG04 BD95: BD20
 FRMEVL BD98: \$B8B3 \$B942 \$BABD \$BD84 \$BEEC \$C054 \$C2E0 \$D7CF \$D7DB
 \$DE4B \$F53C
 FRMEV1 BD9E: BD9A
 LPOPER BDA3: #BE27
 TSTOP BDB2: #C789
 LOPREL BDB5: #BDCE
 ENDREL BDD1: BDB8 BDBC
 BDE2: BDDD
 QPREC BDEA: BE11
 DOPREC BDF3: BE18
 NEGPRC BDF4: #BF09
 FINREL BE01: BDD3
 FINRE2 BE08: BE07
 QPREC1 BE13: BDFA
 DOPRE1 BE1A: \$B0F4
 SNERR5 BE2A: BDC7
 PUSHF1 BE2D: \$BE22
 PUSHF BE32: \$B73E
 FORPSH BE41: #B724 BE3D
 QOP BE56: BDD5 BDD9
 QOPGO BE59: BDFA
 QCHNUM BE5B: BDEE
 UNPSTK BE62: BE5D
 PULSTK BE64: BDFF BE16
 QOPRTS BE7E: BE59
 EVAL BE81: \$BDAB \$C755
 EVAL0 BE85: BEAF
 EVAL1 BE8A: BEA7
 EVAL2 BE8D: BE88
 QDOT BEA5: BE94
 STRTXT BE85: \$BBC2
 STRTX2 BE8E: BEBB
 EVAL3 BEC4: BEB3
 EVAL4 BEDB: BEC6
 BEE2: BEDD
 PARCHK BEE9: BEE4 \$C071 \$C526
 CHKCLS BEEF: \$C336 \$C4EF \$C897 \$D7D5 #DE97
 CHKOPN BEF2: \$BEE9 \$C051 \$C4E2 \$D7CC \$DE91
 CHKCOM BEF5: \$BCAE \$C057 \$C11E \$C878 \$C927 \$D7C5 \$D7E6 \$F555
 SYNCHR BEF7: \$B70B \$B7B1 \$B8BF \$B939 \$BA95 \$B889 \$B8A9 \$BBC7 \$C4F4
 \$C5DC \$D7B9
 SNERR BF00: #B7A9 #B873 #B85E #BE2A BEFB #BF28 #C13C #C1DB #C56F
 #D7AC #D808 #D94C #D981 #D9DE #DA0E #DA87 #DB17 #DB44
 #DCBA #DD3E #DDB3 #DDCD #DE20 #DE69 #DE8C #F35A #F36B
 #F55D #F6FA
 DOMIN BF05: BEAB
 GONPRC BF07: BECA
 ISVJMP BF0D: BE90

Label Address Referenced by

PATCHG BF10: #BA6C
 PCTHO BF1D: BF13
 PCTH1 BF1E: BF18
 PATCHH BF21: \$B77C
 BF2B: BF26
 PATCHI BF2E: #88ED
 BF35: BF30
 BF3E: BF39
 ISVAR BF8C: #BF0D
 ISVDS BFC1: BFA7 BFAB
 STRRTS BFD3: BFA3 BFC3 BFC7 BFFE
 G000 BFD4: BF99
 G00000 BFE5: BFD6
 CHKDS BFFC: \$BFC9 \$C024
 GETTIM C003: \$BFAD \$BFF3
 QSTATV C00F: BFED
 QDSAV C01C: C011 C015
 GOMOVF C040: BFE9 BFF1 C01E C022
 ISFUN C047: #BEE6
 OKNORM C071: C04F
 FINGO C076: #C06E
 STRCMP COCE: C0B9
 STASGN COF6: COEC COFO
 NXTCMP COFB: C10A
 QCOMP C101: #C0CB
 GETCMP C106: C0FD
 DOCMP C112: C101 C104 .C10E
 GOFLOT C11B: C117
 DIM3 C11E: C128
 PTRGET C12B: \$B930 \$BC11 \$BD1F \$BF8C \$C4E9
 PTRGT1 C130: \$C122
 PTRGT2 C132: \$C513
 INTERR C13C: C16A
 PTRGT3 C13F: C13A
 ISSEC C14F: C148
 EATEM C150: C153 C158
 NOSEC C15A: C14D
 NOTSTR C164: C15C
 TURNON C174: C162
 STRNAM C17B: C166
 C187: C182
 STXFND C18F: C1B4
 LOPFND C191: C1B1
 LOPFN C19B: C195
 NXTPTR C1AB: C1A6
 NOTIT C1AC: C19F
 ISLETC C1B6: \$BE8D \$C137 \$C14A \$C155
 ISLRTS C1BF: C1B8
 NOTFNS C1CO: C199
 LDZR C1C6: C1D5
 NOTEVL C1CB: C1C4
 GOBADV C1DB: C1E4 C1EC C1FO
 QSTAVR C1DE: C1D1 C1D9
 QDSVAR C1E6: C1EO
 VAROK C1F2: C1E8
 NOTEVE C208: C205
 ARYVA2 C21C: C244 C247
 ARYVA3 C220: C261

Label Address Referenced by

ARYVGO C228: C222
 ARYGET C259: C255 C297 C29B
 GOGO C263: C25D
 GOGO1 C281: C27B
 DVARTS C290: C267 C279 C27F
 ARYDON C29D: C226
 FINPTR C2B9: #C1A8
 FINNOW C2C3: C2C0
 FMAPTR C2C8: \$C37E \$C38C
 JSRGM C2D4: C2D1
 INTIDX C2DD: \$C30E
 POSINT C2E3: \$C8D7
 AYINT C2EA: \$B952 \$BECC \$C08D \$C09F
 NONOGO C2F7: C2E8
 QINTGO C2F9: C2EE
 ISARY C2FC: #C184
 INDLOP C306: C332
 LOPFDA C347: C36E
 LOPFDV C353: C34D
 NMARY1 C362: C35A
 BSERR C370: C387 #C433
 FCERR C373: #B9B2 C2F7 #C8CE #CB27
 ERRG03 C375: C37C
 GOTARY C378: C360
 NOTFDO C38C: C351
 NOTFLT C39F: C39C
 STOMLT C3A8: C3A4
 LOPPTA C3B1: C3D3
 NOTDIM C3C1: C3B7
 GREASE C3E4: C3DF
 ZERITA C3F3: C3F6 C3FC
 DECCUR C3F8: C3F1
 GETDEF C415: #C389
 INLPNM C41D: C451
 BSERR7 C433: C42B
 OMERR1 C436: C3D7 C3E2 C490 C4A1
 INLPN2 C439: C429
 INLPN1 C43A: C431
 ADDIND C44B: C43F
 NOTFL1 C45C: C459
 STOML1 C462: C45E
 DIMRTS C476: C412 C4D2
 UMULT C477: \$C3C8 \$C441
 UMULTD C480: \$C466
 UMULTC C48A: C4A5
 UMLCNT C4A3: C496
 NOREF C4AF: C4AA
 GIVAYF C4BC: #BED8 #BFE2 #C0B3 C4CD
 SNGFLT C4CB: #C8B5 #C8CB #C957
 ERREDIR C4CF: \$BB7A \$BBCD \$C4DF
 ERGVVF C4D7: C53C
 GETFNM C50A: \$C4DC \$C51D
 FNDOER C51D: #BEDF
 DEFSTF C541: C545
 C572: C560
 DEFFIN C578: #C507
 TIMSTR C598: #BFBE
 STRINI C59E: \$B9FA \$C5E9 \$C76F

Label Address Referenced by

STRSPA C5A6: \$C829 \$C845
 STRLIT C5B0: \$BAC7 \$BB1D \$BEBE #BFDO C59C
 STRLT2 C5B6: \$BC79
 STRGET C5C0: C5CB
 STRFIN C5C0: C5C7
 STRFI1 C5D1: C5C3
 STRFI2 C5D2: C5CF
 STRST2 C5D8: C5DB
 STRCP C5E8: C5E2
 PUTNEW C5F3: C5E6 \$C786 #C833 #C85F
 ERRG02 C5FB: C65F
 PUTNW1 C5FE: C5F7
 GETSPA C61D: \$C5A6 \$D999
 TRYAG2 C61F: C668
 TRYAG3 C62D: C62A
 TRYAG4 C63A: C637
 STRFRE C644: C63E
 GETRTS C65A: C620
 GARBAG C65B: C63C C642
 GARBA2 C66A: \$B3B4 \$C4AF \$C661
 GLOOP C67E: C6A7 #C713
 COLOO C68A: C682
 COLOOB C693: C68C
 COLOOA C69E: C6F9
 COLO1 C6A9: C69C
 COLO2 C6B2: C6FE
 GLOP1. C6CE: C6D4
 COLO2B C6D8: C5DE
 COLO2A C6F0: C6E8
 GRBEND C700: C680 C686 C688 C6E6 C6EC C6EE
 COLO3 C703: C6AE
 ENDGRB C716: #C700
 SKIP2 C71F: \$C6F0
 SKIP2A C724: \$C693
 MOVPNT C726: \$C6A1 \$C6C9 \$C710
 MOVOO C730: C720
 MOVTOP C735: \$C690 \$C6C1 \$C708 \$C721
 MOVO1 C73F: C73C
 SETINX C744: \$C6A9 \$C6FB
 SETOO C746: C74C
 CAT C74F: #BDDF
 SIZEOK C76F: C768
 MOVINS C78C: \$BA05 \$C772
 MOVSTR C79A: \$C5F0
 MOVD0 C79E: \$C77C \$C85C
 MOVLP C7A2: C7A8
 MVDONE C7AB: C79F
 MVSTRT C7B4: C7B0
 FRESTR C7B5: \$C8B8 \$DE4E \$F53F
 FREFAC C7B8: \$B96B \$BB20 \$C0D4 \$C4AC
 FRETMP C7BC: \$COE1 \$C779 \$C783 \$C84C
 RESOO C7DE: C7DB
 FREO1 C7F6: C7F2
 FREPLA C7FC: C7F8
 FREO2 C7FE: C7C3 C7C8
 FRETMS C811: \$BA0C \$BA1B \$C7C0
 FRERTS C821: C7E6 C7EA C813 C817
 RLEFT C83C: #C86A

Label Address Referenced by

RLEFT1 C842: C83C
 RLEFT2 C843: C88B
 RLEFT3 C844: C891 C895
 PULMOR C85B: C857
 MID2 C87E: C876
 PREAM C897: \$C836 \$C862 \$C87E
 LEN1 C888: \$C8B2 \$C8C1 \$C8E3
 GOFUNC C8CE: C881 C8C4 C8DC C92F C935
 GTBYTC C8D1: \$BAFE
 GETBYT C8D4: \$B8D6 \$BA8E \$BB84 \$BBA4 \$C067 \$C87B #C92A \$DE94 #DE9A
 #F4A2 \$F51B
 CONINT C8D7: \$C822
 C8EB: C8E6
 VAL2 C903: C900
 ST2TXT C918: \$BC7C #BEC1
 GETNUM C921: \$C95A \$C963
 COMBYT C927: \$C96F
 GETADR C92D: \$C924 \$C949 \$D7D2 \$D7DE \$F6C6
 STORDO C972: C96D
 WAITER C976: C97C
 ZERRTS C97E: C9AE
 FADDH C97F: \$CFE5 \$D2AE
 FSUB C986: \$CB45 \$D2A6 \$D352
 FSUBT C989: \$D1C1 \$D29F
 FADDS C998: C9CF
 FADD C99D: \$B049 #C983 \$CB37 \$CB53 \$D215 \$D259 \$D286 \$D2C2
 FADDT C9A0: #C995 #CEC4
 C9A5: C9A0
 FADDC C9AD: \$CC27
 FADDA C9C9: C9B5
 FADD1 C9CD: C9C7
 FADD4 C9D9: C99B C9B3
 SUBIT C9E5: C9E1
 FADFLT CA08: #CD8B #CE1C
 NORMAL CA0D: CA08 #CC05 \$D278
 NORM3 CA11: CA2B
 ZEROFC CA2D: #C8E8 CA64 #CC12
 ZEROF1 CA2F: #D118
 ZEROML CA31: #CC02
 FADD2 CA34: C9DB
 NORM2 CA53: CA5F
 NORM1 CA5F: CA13
 SQUEEZ CA6C: #CA50
 RNDSHF CA6E: #CD5E
 RNDRTS CA7C: CA6C
 NEGFAC CA7D: \$CA0A
 NEGFCH CA83: \$CDE1
 INCFAC CAA5: \$CD59
 INCFR1 CAB3: CAA3 CAA7 CAAB CAAF
 OVERR CAB4: CA70 #CC15 #CED3
 MULSHF CAB9: #CB91
 SHFTR2 CABB: CAD1 CA03
 SHIFTR CACF: \$C998 \$CDEB
 SHFTR3 CADC: CAEE
 SHFTR4 CAE2: CADE
 ROLSHF CAE6: \$C9D6 \$CDFC
 SHFTRT CAFO: CADA
 LOG CB20: \$D13A

Label Address Referenced by

LOGERR CB27: CB23
 LOG1 CB2A: CB25
 FMULT CB5E: \$CFBA \$D141 \$D188 \$D1E0 #D1EA \$D204 \$D252
 CB66: CB61
 MLTPLY CB8F: \$CB75 \$CB7A \$CB7F \$CB84
 MLTPL1 CB94: \$CB89 CB8F
 MLTPL2 CB97: CBF
 MLTPL3 CBB3: CB98
 MULTRT CBC1: #CB63
 CONUPK CBC2: \$C986 \$C99D \$CB5E \$CC45
 MULDIV CBED: \$CB66 \$CC54
 MLDEXP CBEF: \$D1D3
 TRYOFF CBFA: CBF4
 CC05: CC00
 MLDVEX CCOA: \$D19F
 ZEREMV CC10: CBEF CBFA
 GOOVER CC15: CBF6 CCDE CC21 CC2C CC59
 MUL10 CC18: \$B97D \$B999 \$CE91 \$CEA7 \$CFD7
 FINML6 CC23: \$B98F
 MUL10R CC2E: CC1C
 DIV10 CC34: \$CE88 \$CFDE
 FDIVF CC3D: \$D292
 FDIV CC45: \$CB3E #D2F7 \$D33F
 FDIVT CC48: #CC42
 DIVIDE CC5F: CC8F
 SAVQUO CC75: CC63 CC69 CC6F CC8D CC91
 QSHFT CC82: CC77 CCB2
 SHFARG CC85: #CCAD
 DIVSUB CC93: CC83
 LD100 CCB0: CC7C
 DIVNRM CCB4: CC7E
 DVOERR CCC0: CC48
 MOVFR CCC5: #CB8C #CCB0
 MOVFM CCD8: \$B72B \$BD3C \$BE9A \$C008 #C044 \$CC3F \$D10F \$D24B \$D2E7
 MOV2F CCFD: \$D1F1
 MOV1F CD00: \$D1DB \$D2D2
 MOVVF CD06: #B961 \$BD4C
 MOVMF CDOA: \$C549 CD04 \$D11F #D27F
 MOVFA CD32: \$C09C #C9A2
 MOVFA1 CD34: \$D135
 MOVFAL CD38: CD3D
 MOVAF CD42: \$B987 \$CC18 \$CC34 \$CEB5 \$D108 \$D289 \$D295
 MOVEF CD45: \$D196
 MOVAFL CD47: CD4C
 MOVRTS CD50: CD53 CD57 CD5C
 ROUND CD51: \$B94F \$BE41 \$CC4A \$CDDA \$CD42
 INCRND CD59: \$D191
 SIGN CD61: \$B73B \$CB20 \$CD6F CD9B \$D229
 FCSIGN CD65: CDA1
 FCMPMS CD67: #CDCE
 SIGNRT CD6E: CD63 CD6A
 FLOAT CD72: #C019 #C03D #C11B \$CEB9
 FLOATS CD7A: #C4C6
 FLOATC CD7F: \$CF8A
 FLOATB CD85: #BFF9
 FCOMP CD91: \$C0C7 \$C2F4 \$CFCS \$CFDD \$D12D
 FCMPN CD93: \$B951
 FCMPPC CDC8: CDA5 CDAD CDB4 CDBB

Label Address Referenced by

FCOMPO CDCE: CDCA
 QINT CDD1: \$B99C #C2F9 \$C937 \$CE08 \$CFE8
 QISHFT CDE5: CDDA
 QINTRT CDF0: CDC6
 QINT1 CDF1: CDE9
 INT CE02: \$D126 \$D1A2 \$D298
 CLRFAC CE1F: CDD3
 INTRTS CE28: CEO6
 FIN CE29: \$BC85 #BE8A \$C910
 FINZLP CE2D: CE30
 QPLUS CE3C: CE36
 FINC CE40: CE3A CE7B #CEB1
 FINDGQ CE43: CE32
 FIN1 CE45: CE3E
 FINEC1 CE64: CES4 CE58
 FINEC CE66: CES5 CE60 #CEE6
 FNEDG1 CE69: CES0
 FINEC2 CE6B: CE62
 FINDP CE77: CE47
 FINE CE7D: CE4B CE6D
 FINE1 CE7F: #CE74
 FINDIV CE88: CE8D
 FINMUL CE91: CE86 CE96
 FINQNG CE98: CE84 CE8F
 NEGXQS CE9D: CE9A
 FINDIG CEA0: CE43
 FINDG1 CEA7: CEA3
 FINLOG CEB4: #B9B7 \$CB57 \$CEAE
 FINEDG CEC7: CE69
 MLEX10 CE06: CECB
 MLEXMI CEE4: CED1
 INPRT CF78: \$B3FC
 LINPRT CF83: \$B67E \$D43B \$D8CD
 STROU2 CF90: \$CF7C
 FOUT CF93: \$BAC4 \$CF8D
 FOUTC CF95: \$C593
 FOUT1 CF9D: CF99
 CFAE: CFA9
 FOUT37 CFB6: CFB2
 FOUT7 CFBF: CFB4
 FOUT4 CFC1: CFE3
 FOUT3 CFCC: CFDC
 FOUT38 CFD7: CFD3
 FOUT9 CFDE: CFCA
 FOUT5 CFE5: CFDS
 BIGGES CFE8: CFC8
 FOUTPI CFFD: CFF2
 FOUT6 CFFE: CFF6
 FOUT39 D009: D005
 FOUT16 D01A: D012
 FOUT8 D01C: D007
 FOUTIM D01E: \$BF88
 FOUT2 D020: D040 D044 D078
 FOUT41 D044: D03E
 FOUT40 D046: D042
 FOUTYP D04D: D047
 STXBUF D068: D060
 FOULDY D07A: D074

Label Address Referenced by

FOUT11 D07C: D082
 FOUT12 D089: D086
 FOUT14 D099: D08F
 FOUT15 D0A5: D0A8
 FOUT19 DOBA: #CFAB
 FOUT17 DOBD: D08D
 FOUT20 DOC2: D0B8
 FOUTBL DOC3: :D038
 DOC4: :D031
 DOC5: :D02A
 DOC6: :D023
 FPWRT1 D11B: D116
 FPWR1 D135: D124 D130
 NEGOP D14B: #CE9D \$D1C4 \$D2BB \$D2C8 \$D331 #D358
 NEGRTS D155: D149 D14D
 EXP D184: D112 \$D144
 STOLD D194: D18F
 GOMLDV D19F: D1AA
 EXP1 D1A2: D19D
 SWAPLP D1B2: D1B8
 POLYX D1D7: \$CB4C #D2CF \$D346
 POLY D1ED: \$D1CB
 POLY1 D1F1: \$D1E3
 POLY3 D200: D1FC
 POLY2 D204: D21E
 POLY4 D211: D20E
 QSETNR D247: D22E
 RND1 D25C: D22C
 STRNEX D26C: #D244
 GMOVMF D27F: \$D2E0
 SIN D289: \$D2D9
 SIN1 D2BB: D2AC #D2FB
 SIN2 D2BE: D2B3
 SIN3 D2CB: D2C6
 COSC D2FA: \$D2FO
 ATM1 D334: D32F
 ATN2 D342: D339
 ATN3 D355: D34C
 ATN4 D35B: D356
 D398: :D3C9
 INITAT D399: D3A8
 CHDGOT D39F: D39B
 CHDRTS D3B0: D3A4
 INIT D3B6: #FD46
 MOVCHG D3C9: D3CF
 LOOPMN D400: D3F7 D415
 LOOPM1 D408: D402
 USEDEC D417: D406 D40E
 USEDEF D41B: #D3FO
 CALLE D472: #FD43
 B3 D491: D476
 STRT D4BA: #D5F5 #D6BF #D714 #D7A1 #D7A9
 ST1 D4C9: D4CE D4D2
 S0 D4D4: D4B8
 S1 D4D6: D4E7
 S2 D4E6: D4D9
 PUTP D4EC: \$D603 \$D643
 DM D4F7: \$D5B7 \$D5F0

Label Address Referenced by

DM1	D4FB:	0508
BYTE	D50B:	\$062C
BY3	D510:	050E 0516
SETR	D523:	\$05B4 \$0618
SPAC2	D52E:	#0584
SPACE	D531:	\$04FB \$052E \$05A5
CRLF	D534:	\$04AC \$057B
INCTMP	D539:	\$0503 \$0510
SETWR	D543:	053B 053F
CMD5	D544:	:D4D6
ADRH	D54C:	:D4DD
ADRL	D554:	:D4E1
REGK	D55C:	:D589
ALTRIT	D579:	\$0596 \$05E8
D2	D589:	D592
DSP1	D5D2:	D5F3
BEQS1	D5F5:	D5BA D5D5 D5D9 D5E4 D631
ERRS1	D5F8:	D5C2 D5CD D623 D63C
AL2	D606:	D601
AL3	D618:	D60C
A4	D627:	D618
A5	D629:	D62F
G1	D646:	D638 D641
ERRL	D672:	D695 D6A9
L1	D688:	D68D
L2	D695:	D6B1
L3	D697:	D6AB
L4	D6A9:	D6D2
L5	D6AD:	D691 D6A0 D6C7 D6DF
L6	D6B1:	D6B0
L7	D6B0:	D6CB
L8	D6C2:	D69C
L9	D6CB:	D6E3
L10	D6D2:	D6D6
L12	D6E3:	D6F0
L13	D6F0:	D709
L20	D700:	D705
L14	D709:	D70F
WROA	D717:	\$05EB
WROB	D722:	\$0500 \$059C \$05A2 \$05AB \$05B1 \$071E
WRTWO	D731:	\$04B3 \$04C6 \$0581
ASC	D73A:	\$0727 \$072E
AS1	D741:	D73D
T2T2	D744:	\$05C4 \$05CF \$06E8 \$06FD
TT	D746:	D751
RDDA	D754:	\$05BF \$05CA \$05FE \$0609 \$0620 \$063E \$06E5 \$06F2
R1	D75B:	D757
R2	D762:	D75E
RDOB	D763:	\$050B \$05FB \$061D \$06CD \$0754 \$075B
R3	D778:	D76D
R4	D785:	D774
HEXIT	D780:	\$0778 \$0785
H1	D797:	D793 D79D
RDOC	D798:	\$04C9 \$05BC \$05C7 \$0629 \$0768 \$076F \$0782
ERROPR	D7A4:	#051A #05F8 #D672 D7AC: D7C8 D7E9 D7FC D7AF: #FF9C D7DB: D7CA

Label Address Referenced by

REXNEX D7E1: #D7D8
 DONER D7FE: . D7E4
 QTYER1 D801: D7C1 D7F0 D7F4
 CHK1 D804: \$D9D5 \$DB69
 CHKER1 D808: D812 D81A D81F D828 D832
 CHK2 D80B: D806 \$DB10 \$DB3D
 CHK3 D818: \$D87A \$DA68
 CHK4 D81D: \$DABA \$DACA
 CHK5 D824: \$DB58
 CHK6 D82E: \$D945 \$D97A
 TABLD D838: :DC08
 CATALOG D873: #FFB4
 CATBLD D889: D886
 WG220 D8A5: D8C1 D90F
 WG250 D8D8: D8F8 D8FC D903
 WG255 D8FE: D901
 WG240 D905: D8EB
 WG230 D912: D8B3 D8BC D8E7 D8F3
 SUBA D91A: \$D8ED \$D907
 SUBB D923: \$D8C6 \$D91A
 SUBBR D92E: D927
 ENTRY0 D92F: \$D94F \$D984
 ENTRY1 D931: D93B
 ENTRY2 D935: D93D
 RFOUND D93F: D936
 DOPEN D942: #FF96
 DOPEN2 D94F: D94A
 RECLCK D95D: D959
 LEAV1 D970: D960
 APPEND D977: #FFAB
 DAPPEN D984: D97F
 ERRCH1 D991: \$D9F8 \$DB7D
 GETDS D995: #FFBD
 ECHKS D9AB: D993
 EREAD D9B3: D9AD
 LOOP1 D9BF: D9C9
 ERREND D9CB: D9C5
 FORMAT D9D2: #FF9F
 DFORMA D9E1: D9DC
 FBUILD D9EA: D9E7
 FCONT D9F5: D9F1
 FERRP DA04: . DA01
 DCLOSE DA07: #FF99
 DCLSE DA11: DA0C
 DCLALL DA1B: \$D9E1 DA16 DA2E \$DA6B \$DA91
 DCLLP DA1F: DA25
 DCLBYE DA30: DA20
 BOBREC DA31: #D7FE
 DRECG DA3D: DA36
 COLECT DA65: #FFA2
 DCOLLO DA7A: DA77
 BACKUP DA7E: #FFA5
 BERO DA87: DA8F
 BACK DA8A: DA85
 TRANS DA98: \$D9F5 #DA7B #DAC4 #DAD1 #DB63 \$DB75
 TRANS1 DA9B: #DA62
 COPY DAA7: #FFA8
 COPY2 DAB7: DAAE

Label Address Referenced by

COPY3 DACD: DAB5
 CONCAT DAC7: #FF93
 RSFN DAD4: \$DC11
 RDFN DAE1: \$DC18
 RDMOV DAEA: DAF4
 RDRT0 DAF8: DAE6
 RDRT1 DAF9: DAF6
 RID DAFD: \$DC28
 DSAVE DB0D: #FFAE
 DSAVE2 DB1A: DB15
 SAVLD1 DB32: DB21
 DLOAD DB3A: #FFB1
 DLOERR DB44: DB5D
 DLOAD2 DB47: DB42
 RENAME DB55: #FFB7
 SCRTCH DB66: #FFBA
 NUMLP DB87: DB91
 NUMPRT DB93: DB8B
 NUMBYE DB98: DB6F DB7B
 DDIREC DB99: \$DB78 \$DB9E \$DBD7
 RUSURE DB9E: \$D9E4 \$DB6C
 ANSNO DBCB: DB80 DBBB DB-C2 DBD3
 ANSYES DBD5: DBA1 DBB7 DBC9
 ANSBYE DBD6: DBCE DB.DA
 BADDIS DBD7: #DAD4
 OLDCLR DBE1: \$DA11 \$DA40 \$DBFF \$F7E0 \$F82E
 OLDCL-1 DBF1: DBE5
 SENDP DBFA: \$D88A \$D971 \$D98B \$DA98 \$DB34 \$DB4B
 SENDP1 DBFC: \$D96A \$DB2C
 SENDP2 DCD2: DC3F DC4A
 RXFN2 DC14: DCOF
 RXREC DC1B: DC16
 RXID DC24: DC1D
 RXWRT DC2B: DC26
 RXD1 DC34: DC2D
 RXD2 DC3D: DC36
 RXDD DC44: DC3B
 RPLCE DC46: DC0B DC22 DC32
 TRANR DC4C: \$DA5F DC05
 RWRT DC57: \$DC2F
 RWRTL DC60: DC5A
 RWRTS DC67: DC5E
 DOSPAR DC68: \$D877 \$D942 \$D972 \$D9D2 \$DA07 \$DA65 \$DA7E \$DAA7 \$DAC7
 \$DB0D \$DB3A \$DB55 \$DB66
 PARSEL DC89: #DDA5
 PARNXT DC9F: DC97
 NEXT7 DCB2: DCAD
 SNERL DCBA: DCC2 DCDE DD1A
 LOGADR DCBD: DC8B
 RECLEM DCD8: DC8F DC93
 RECOO DCEA: DCE2
 RECON DCF8: #DCE7
 DONE1 DDO3: DC87 #DD9B
 NAMEL1 DDO6: DCB4 DCB8
 ON1 DDO9: DCA5 #DDAC
 UNIT1 DDOF: DCA9
 DRV1 DD15: DCA1
 QTYER2 DD34: DCC9 DCEF DCF3 DD21

Label Address Referenced by

IDENT DD37: #DCAF
 IDCON DD41: D03C
 NEXT3 DD45: D053
 NEXT4 DD4B: D047
 NAME1 DD60: #DD0
 LOOP6 DD77: D08C
 NAMCON DD85: DD7B DD7F
 DELIM1 DD96: #DC9C #DCD5 #DD00 #DD0C #DD12 #DD31 #DD5D
 NXXX DD9E: D099
 NEXT6 DDA8: D0A0
 NEXT6A DDAF: D0AA
 PARSE2 DDB6: D0B1 DE16
 SNER2 DDCD: D0D5
 DRV2 DDD0: D0B8
 ON2 DDEC: DDBF DE1A
 UNIT2 DDF2: DDC3 DE1E
 NAME2 DDF8: DDC7 DDCB
 DELIM2 DEOF: #DDE9 #DDEF #DDFS
 SNER3 DE20: DE31 DE49
 DONE DE23: #DD03 DE12
 QTYERR DE27: #D801 #DD34 DDDC DE38 DE3C DE54
 ON DE2C: \$DD09 \$DDEC
 DE33: \$DD0F \$DDF2
 NEWNAM DE49: \$DD65 \$DDFD
 LENCHK DE6C: DE60
 ERRLEN DE74: DE58 DE6F
 NXXTS DE79: DE67
 NXXS DE82: #DE71
 GETVAL DE87: \$DCC4 \$DCEA \$DD1C \$DD07 \$DE33
 GTVL2 DE8A: \$D7BC \$D7EB
 CONT DE8F: DE8A
 NUMERC DE9A: DE8F
 DE9E: \$D9A4
 CINT E000: \$FD1B
 LP2 E0A7: #F212
 LOOPS E116: #F221 #F22E
 PRT E202: \$DB82 \$DB8D \$DB95 \$F18B #F26E \$F86A
 PREND E600: #FA0D #FA49 #FA99 #FAC6 #FBB8 #FC35 #FC83
 PIAL E810: :F1E4 :F883 :F888 :F9AO :FD3E
 PIAL1 E811: >FOEB >FOF7 >F8C4 >FCCC
 PIAK E812: :F7A1 :F7A4 :F921
 PIAS E813: >F8EA >F8F4 :F91C >FCD1 >FCED
 IEEEI E820: :F1EE
 IEEEIS E821: >FOE2 >F17F >F1C2 >F1F6 >F200
 IEEO E822: >F11B >F13F
 IEEOS E823: >F10B >F125 >F13A
 PIA E840: :F0D8 >FO0D :FOFF >F106 :F10E :F11E :F12D :F148 >F14D
 :F175 >F17A :F1AE >F1B3 :F1C5 >F1CA :F1D7 :F1DC >F1E1
 :F1F9 :F8FB >F902 :F99D :FBE9 >FBEE :FCFO >FCF5
 CHTIM E844: :D230
 T1L E844: >F96A
 CHTIM E845: :D23A
 T1H E845: >F12A >F1CF >F971
 T2L E848: :D235 :F961 :F97C >F987 >FBE3
 T2H E849: :D23F >F90F :F96E :F976 :F97F >F98A >FBE6
 IFR E84D: :F130 :F1D2 :F9F5
 IER E84E: >F8BF >F8DB >FA7B >FA8C >FCC7
 MSG1 F000: :F185

Label Address Referenced by

TALK F002: \$D9B3 \$F374 \$F7E3
 LISTN F005: \$DA9F \$F4AD \$F700 \$F733 \$F831
 LIST1 F007: \$F1BB
 FOFA: FOE7
 LIST4 F0FF: F102
 ISOUR F109: \$F0EE \$F145 \$F195 \$F1A7
 ISR1 F11E: F121
 ISR0 F128: F159
 ISR2 F120: F136
 ISR3 F138: F16A
 SECND F143: \$F4B4 \$F705 \$F73C \$F83D
 SCATN F148: #F198 F1BE \$F838
 ERRS3 F151: F133
 ERRS4 F158: F1D5
 ERROPO F165: F154
 ERRP1 F167: F16E
 ERRP7 F16C: F115
 ERRO1 F170: F15E
 ER001 F175: \$F198
 MSG F185: \$DBA5 #DBDE F190 #F34E \$F450 \$F459 #F50A \$F58C \$F603
 \$F74F \$F85E \$F863 #F877 \$F893
 TKSA F193: \$D9BA \$F379 \$F7F0
 TKATN F198: \$F7EA
 CIOUT F19E: #F274 \$F4C8 \$F70F \$F714 \$F71E
 CI2 F1A6: F1A0
 CI4 F1AB: F1A4
 UNTLK F1AE: #D9CF \$F2B5 \$F3CE
 UNLSN F1B9: \$D899 \$D90A \$F2AC #F4D0 \$F72C #F73F
 ACPTR F1C0: \$D9C0 #F263 \$F37C \$F387 \$F398
 ACP00 F1CD: F163
 ACP01 F1D2: F1DA
 ACP03 F1EE: F1E7
 ACP05 F1F9: F1FC
 GETIN F205: \$D8F5 \$D8FE #FFE4
 BASIN F215: \$D606 \$D633 \$D688 \$D697 \$D6C2 \$D6DA \$D6EB \$D700 \$D798
 \$D8AC \$D8B5 \$D8DD \$DBAB \$DBB2 \$DBBD \$DBC4 \$D8D0 #FFCF
 BN10 F224: F20B F217
 BN20 F231: F226
 JT635 F243: F23C
 JTGET F249: \$F235 \$F239 F257
 JTG10 F259: F24C
 BN30 F25C: F231
 BN32 F262: F20F F27C
 BN35 F263: F25E
 BSOUT F266: #D536 \$D58C \$D733 #D737 \$D7A6 \$D8D2 \$D91D \$F464 \$F5B4
 \$F5B9 #FFD2
 B010 F271: F26B
 B020 F277: F271
 B021 F278: \$F301
 JTP10 F298: F286
 RSTOR F29C: #F7DC #F82A
 CLALL F2A2: \$F5AF #FFE7
 CLRCHN F2A6: \$D478 \$D8C3 \$D8D5 \$D8E1 \$D912 #D920 \$DBA8 \$F33C #FFCC
 JX750 F2AF: F2AA
 JX770 F2B8: F2B3 \$F930
 JLTLK F2C1: \$DA33 \$F2E2 \$F569 \$F7B9 \$F808
 JX600 F2C3: F2CB
 JZ100 F2CD: \$DA30 \$F2E7 \$F7C3 \$F800

Label Address Referenced by

JZ100 F20C: F2C4 F2C9
 CLOSE F20D: #FFC3
 CLOSS F2E2: #D917 #DA18
 CLOS10 F2E7: \$DA2A
 JX120 F315: F2F4
 JX150 F318: F2EE F2F2 F2FA F30B #F312
 JX170 F334: F2E5 F31E
 STOP1 F335: \$D5D2 \$D8F0 \$F343 \$F935
 STOP2 F342: F339 F34C
 STOP F343: \$F156 \$F160 \$F395 \$F721 #FFE1
 SPMMSG F349: \$F430 \$F475 #F47A
 TXTST F351: \$F349 \$F433 \$F449 \$F5FC \$F748
 LD15 F356: \$D6B3 \$F415
 LD20 F35A: F35F
 F350: F358
 F36E: F369
 LD40 F38F: F3A0 F3BF
 LD50 F3B3: F3A5
 LD60 F3B5: F3AA
 LD64 F3BB: F3B7
 LD90 F3C1: F385
 LD65 F3C6: F3B0
 LD100 F3D4: F361
 LD112 F3D0: F3F0
 LD120 F3E6: F3EC
 LD150 F3E9: F3DF
 LD170 F3EE: F3E4
 LOAD F401: #FFD5
 LD10 F405: \$F4FA
 LOADNP F408: #DB52
 LD11 F40D: F40F F413
 LD210 F42E: F423
 LD205 F443: F436
 LD300 F449: \$F36E \$F30A \$F5A1
 LD105 F45C: \$F752
 LD110 F462: F46A
 LD115 F46C: F3F6 F41A F44C F455 F45E
 LD410 F46D: \$F38C \$F3FB
 F475: F471
 PARS1 F47D: \$F405 \$F600
 PRO60 F49E: F4A7 F4AB
 PRO70 F49F: \$F491 \$F499 \$F523 \$F531
 OPENI F4A5: \$F371 #F595 \$F6FD
 OPENIB F4B4: #DAA4
 OP37 F4BB: #F7F7 #F844
 OP35 F4C0: F4B9
 OP40 F4C6: F4CE
 OP45 F4D0: F4C2
 FAF F4D3: \$F3E1 F4EA \$F5A8
 FAF20 F4E0: F4F2
 FAF30 F4F4: F4E2
 FAF40 F4F5: F4D6
 VER F4F6: #FFD8
 VER10 F508: F501
 PARS2 F50D: \$F20D \$F560
 PR100 F52E: F52A
 P200 F53C: \$F48B
 PR140 F54D: \$F488 \$F48E \$F496 \$F520 \$F52E \$F536

Label Address Referenced by

PR147 F554: F550 F55B
 PR150 F555: \$F49F \$F539
 PR130 F558: \$F518
 PR135 F55D: F565
 OPEN F560: #FFCO
 OP94 F563: #096D #0974 #098E
 FOPEN F565: \$089C
 OP150 F598: F593
 OP160 F5AD: #F3E6 F5C5
 ERMSG F5AF: #0A3A #F3C3 #F42B #F4BD #F505 F56C F576 #F6E9 #F7C0
 OP170 F5C2: F5A6
 OP200 F5C9: F59C
 OP171 F5D1: F5AB F5C7
 OP172 F5E2: F5D9
 OP175 F5E4: F58D F591
 FAH F5E5: \$F3E9 \$F4D3 \$F5C2
 FAH30 F5E8: F5F9
 FAH50 F5FB: F5F5
 FAH55 F608: F610
 FAH45 F612: F5FF
 FAH40 F614: F5F1
 TAPEH F619: \$F30F \$F5CE \$F757 #F765
 BLNK2 F62E: F631
 TH20 F652: F662
 TH30 F664: F656
 LDAD2 F67B: \$F3F8
 LDAD3 F682: F68A
 F695: \$F2FC \$F3D4 \$F61B \$F6AE \$F742 \$F84B
 ZZ10 F6AA: F6A0
 LDAD1 F6AB: \$F664 \$F8A0 \$F8CB
 SYS F6C3: #FFDE
 SV60 F6CC: \$F408 \$F6E0
 SAVE F6DD: #FFD8
 SV3 F6E0: #DB2F #DB37
 SV5 F6E3: \$D711
 SV10 F6E7: F6EE
 SV20 F6EC: F6E5
 F6FD: F6F8
 SV30 F717: F726 F72A
 SV50 F72C: F71A
 CLSEI F72F: \$F315 #F301
 SV100 F742: F6F0
 SV105 F755: F74B
 UDTIM F768: \$F924 #FFEA
 UD10 F770: F76C
 UD20 F77A: F772
 UD20 F784: F77C F780
 UD40 F786: F790
 UD45 F794: F797
 UD50 F79B: F778
 UD60 F7A1: F78B F799 F7A7
 UD65 F7AB: F731 F761
 UD70 F7AC: F788
 CHKIN F7AF: \$D8A9 \$D8DA #FFC6
 JX300 F7BE: F80B
 JX305 F7C0: F7D8 F816 F826
 JX310 F7C3: F7BC
 JX320 F7DA: F7C8 F7CC F7D4 #F7FB

Label Address Referenced by

JX330 F70F: F7CE
 JX340 F7F0: F7E8
 JX350 F7F3: #F7ED
 F7FA: F7F5
 CKKOUT F7FE: \$D92B #FFC9
 CK10 F818: F812
 JX360 F828: F81A F822 #F848
 JX370 F820: F81C
 JX380 F830: F836
 JX390 F840: F83B
 F847: F842
 JTP20 F84B: \$F249 \$F283
 CSTEL F857: \$F307 \$F59E \$F8A6
 CS30 F85E: F898
 CS40 F86D: F873
 CS10 F87A: \$F857 \$F870 \$F88C
 CS20 F883: F87F
 CS25 F88B: F85A F886 F88F
 CSTE2 F88C: \$F5C9 \$F745 \$F805
 RDBLK F89A: \$F24E \$F5E8
 TRD F8A3: #F3FE
 TRD2 F8C4: F8BB
 TRD3 F8C7: F8C2
 WBLK F8CB: \$F288 \$F304
 TWRT F8CE: \$F75A
 TWRT2 F8D5: \$F66B
 TAPE F8E0: F8C9
 TP20 F8FB: F8F0
 TP30 F905: F8F9
 TP32 F907: F900
 TP35 F909: F90A
 TP40 F913: F91F #F927
 TP50 F92A: F917
 TWAIT F92B: \$D6B6 \$F41C \$F67B \$F6AB \$F8A3 \$F8CE F932
 TSTOP F935: \$F86D \$F919 \$F92B
 STOP3 F942: F938
 STT1 F945: \$FA06 \$FA46 \$FA66 \$FAA3
 STT2 F95A: F957
 STT3 F961: F966
 READ F976: F982
 RJDJ F9AE: F9A9
 JRAD2 F9CD: F9B0 FA02
 SRER F9D0: F9CB FA0B FA25
 RADX2 F9D8: F9BA
 RAD1 F9DC: F9C3 #FA58
 RAD5 F9DE: F9DA
 RDBK F9F1: F9A5 F9D2 F9D6 FA29 FA2D FA31
 RADP FA04: FA5E
 RADBK FA0D: F9F3 F9F8 FA3A
 RAD3 FA10: F9ED
 ROUT2 FA1A: FA14
 ROUT1 FA1C: FA12
 RAD4 FA33: FA00 FA22
 RAD2 FA4C: #F9CD
 RAD2Y FA54: FA4E
 RAD2X FA5B: FA52 FA56
 RADQ2 FA80: FA6D
 RADK FA8F: FA84

Label Address Referenced by

RDBK2	FA99:	FA71
RADJ	FA9C:	#F9AB
RD15	FAAC:	FAA8
RD12	FAC2:	FAB4
RD10	FAC6:	FAB9 FAC0 FACF FAD3 FAE7 FAF4 FAFA
RD20	FAC9:	FABO
RD22	FADF:	FADA
RD200	FAE5:	FACB
RD40	FAF6:	FADC FADF
RD60	FAFC:	FAC9
RD70	FB0A:	FAFE
	FB12:	FB0D
RD80	FB27:	FB19 FB21
RD58	FB44:	FB15
RD52	FB6C:	FB5E
RD55	FB70:	FB2F
RD59	FB77:	FB29 #FB41 FB6E
RD90	FB81:	FB48 FB4F FB56 FB66 FB75 FB79
RD160	FB89:	#FB0F
RD161	FB8B:	#FB07
RD167	FB94:	FB90
RD175	FBA0:	FB96
RD180	FBB8:	FB83 FB87 FB9A FB9E FBB1
RD300	FBBB:	\$F70A \$FAED \$FBAA \$FCB7
UDST	FBC4:	\$F167 \$F172 \$F1EB \$F240 \$FABD \$FB02 \$FB72 \$FB85
NEWCH	FBC9:	\$F8E7 \$FA9C \$FC42 \$FCA4
WRITE	FB08:	\$FC1C
WRTW	FBDF:	\$FC13
WRT1	FBE1:	FB0D \$FC9B
WRTX	FBE3:	\$FC01
WRT13	FBF4:	FC61
	FC0F:	FBFB
WRTN2	FC1C:	FC11
WRT3	FC35:	FBF7 FC04 FCOA FC16 FC1A FC1F FC40 FC5A FC69
	FC77	FC7B
WRT2	FC38:	FC27
WRTS	FC42:	FCBE
WRT61	FC58:	FC54
WRT6	FC5C:	FC48
WRT7	FC6B:	FC5F
WRT4	FC7D:	FC3E
WRTBK	FC83:	FC97 FC9E FCA2 FCA9 FCDE
WRTN1	FC86:	#FC0C
WREND	FC8D:	FC88
TNIF	FCC0:	\$F93A \$FBAA \$FCDB
STKY	FCDB:	FCB5
BSIV	FCEO:	\$F8E0 \$FC94 \$FCAD \$FCD6
TNOF	FCEB:	\$FC8A \$FCC2
VPRTY	FCF9:	\$FBAA FD08
VP10	FD05:	FD01
WRT62	FD0B:	\$F717 \$FB0A \$FC5C \$FD05
WRT64	FD15:	FD0F
	FD46:	FD41
BSIT	FD4C:	:FCEO
	FD4D:	:FCES
READDS	FFBD:	#C000
COIN	FFC6:	\$BB8C \$BBAC
COOUT	FFC9:	\$BA99

Label Address Referenced by

CLSCHN	FFCC:	\$B3D3	\$BBBB6
INCHR	FFCF:	\$B4E4	
OUTCH	FFD2:	\$BB46	\$F60A
ISCNTC	FFE1:	\$B665	\$B74A
CGETL	FFE4:	\$BC31	
CCALL	FFE7:	\$B5FC	

```
*****
*
* Cross reference listing two. Area $E000-$E7FF, for 9 inch screen *
*           40 column models with N-keyboard (#1447-29).      *
*
*****
```

On the following pages, the cross reference listing of the BASIC 4.0 ROM 1447-29 is given. This ROM is used in all 40 column machines with a graphics- or N-keyboard that have a 9 inch screen.

The listing has the following format:

First a label is given, followed by the address of that label. The label name corresponds with the name in the assembler listing.

If the address has no label assigned to it, or the official CBM label name was not known, no label is stated. In this case, the label in the source listing is the hex address, written in lower case without a preceding dollar sign.

After the label and the reference address the addresses are printed that make use of the reference address. Each address is preceded with an extra token, that indicates the kind of use:

Token Kind of use

- none Jumped to by relative branch.
- # Jumped to via a JMP absolute or a JMP indirect instruction.
- \$ Jumped to via a JSR absolute instruction.
- :
- > Location is read, and contents is left unchanged.
- >> Location is written to, or accessed in read/modify/write fashion; previous data is therefore lost or altered.

Label Address Referenced by

CTIMR	008D:	>E009
CINV	0090:	>E010 #E452
	0091:	>E014
CBINV	0092::	#E44F
LSTX	0097:	:E4EB >E4EF
SHFLAG	0098:	>E479 >E4CA >E4F7
NDX	009E:	:E0B3. >E0B7 :E0BF >E0DD >E30D :E4FD >E509
RVS	009F:	>E064 :E176 >E25B >E328 >E35D
INDX	00A1:	>E0FC :E110 :E130
LXSP	00A3:	:E104 >E1C7 >E344
LSTP	00A4:	:E10C
SFDX	00A6:	>E476 >E4DB :E4E9
BLNSW	00A7:	>E049 >E0C1 :E458
BLNCT	00A8:	>E047 >E45C >E462
GDBLN	00A9:	:E0CA >E46E
BLNON	00AA:	:E0C6 >E0CE >E466 >E46C
CRSW	00AC:	>E0FO :E11A >E143 >E20B
DFLTN	00AF:	:E147
DFLTO	00B0:	>E018 :E140

Label Address Referenced by

PNTR	00C6:	>E07B :E09C >E0A4 >E100 >E10E :E11E >E138 >E185 :E189 >E19B :E1D4 >E1DB >E1E4 >E1FD :E20D >E235 >E259 >E275 >E295 :E2CA :E2DE :E2F7 >E2FF :E319 >E31E >E32F >E361 :E464 :E606
SAL-	00C7:	>E37B :E382 >E39D >E40D >E416
SAH	00C8:	>E36F >E389 :E38F >E3F9
QTSW	00CD:	>E102 :E130 :E16B >E16F >E1AC :E250 :E2BA >E35F
LNMX	00D5:	>E08E >E09A :E0EE :E187 >E1FF :E246 :E26C :E283 :E2C2 :E2D5
TBLX	00D8:	>E07D :E07F :E108 :E180 >E1C9 :E1CB :E1E0 >E1F4 >E290 >E297 :E303 >E30B >E314 :E346 >E354 >E3A5 >E3A9 :E3DF :E3E2 :E41C
DATA	00D9:	>E122 >E126 :E128 >E158 :E15E >E203 :E20F
INSRT	00DC:	:E17C >E180 :E1A8 :E229 >E2E6 :E2EA >E35B 000F: :E1EA :E1EF :E307
LDTB1	00E0:	>E04F :E081 :E29E :E350 :E3AB >E385 :E3C0 :E3FB :E421 >E429 00E1: :E094 >E1CD >E1CF :E375 :E3AF :E3F5 >E401 00E2: :E1B7 >E1BB 00F8: >E3BE
CAS1	00F9:	>E48F :E495
CAS2	00FA:	>E4A0 :E4A9
	0104:	:E448
	026E:	>E0E2
KEYD	026F:	:E0A7 >E0AF >E4FF
	0270:	:E0AC
	8000:	>E06A
	8100:	>E06D
	8200:	>E070
	8300:	>E073
PX1	E009:	E00C
CLSR	E04B:	\$E33D
LPS1	E04F:	E05F
LPS2	E05D:	E053 E057
LPS3	E05E:	E05B
LPS4	E06A:	E077
NXTD	E079:	\$E261
STUPT	E07F:	\$E107 \$E300 \$E316 #E356 #E43F
STUPZ	E09C:	E092 E096
STUPR	E0A6:	E0A0
LP2	E0A7:	\$E003
LP1	E0AC:	E085
LOOP4	E0BC:	E0EC
LOOP3	E0BF:	E0C3 E0E8 E11C
LP21	E0D3:	E0C8
LP23	E0DF:	E0E6
LP22	E0EA:	E0D8
CLP5	E0F2:	E0F9
CLP6	E0FB:	E0F6
LOPS	E11E:	E106 E10A E112
LOPS4	E12E:	E12A
LOPS2	E134:	E12E
LOPS3	E138:	E132 E134
CLP2	E141:	E144
CLP2A	E153:	E14B
CLP21	E156:	E151
CLP1	E158:	E13F
CLP7	E166:	E162

Label Address Referenced by

QTSWC	E167:	\$E13A \$E223
QTSWL	E173:	E169
NXT33	E174:	#E2B0
NXT3	E176:	#E226
NC3	E17A:	#E22D #E254 #E2F0
NVS	E17C:	E178
NVSL	E182:	E17E
JSTS1	E19F:	E191
LOOP2	E1A6:	E18B E19D #E1C1 E1E8 #E277 #E28C #E336 #E340 #E366
LOP2	E1AE:	E1AA
JSTSX	E1B3:	\$E193 \$E101
JSXB	E1BD:	E1B5
JTS2	E1BE:	E1A1
SCRL	E1C4:	\$E1A3 #E3F0
JSTS2	E1CD:	\$E1BE
BKLN	E1DE:	\$E239 \$E333
BKLN1	E1EA:	E1E2
NTCN2	E1F3:	E1EC
PRT	E202:	\$E0BC \$E153
	E216:	E211
NJTL	E21D:	E218
NTCN	E229:	E21F
CNC3X	E230:	E22B
BK1	E23F:	E237 E248
BK2	E24A:	#E23C
NTCN1	E250:	E232
NC3W	E257:	E252
NC1	E25D:	E259
NC2	E264:	E25F
JPL4	E275:	E285 E287
NCZ2	E277:	E26E
NCX2	E27A:	E266
JPL3	E28C:	E24E E27C E2A2 E2D0
NXTX	E2A4:	#E213
NXTX1	E2AC:	E2A8
	E2B3:	E2AE
UPS	E2BA:	E2B5
INS3	E2CE:	E2C8
INS1	E2D5:	E2CC
INS2	E2D7:	E2E0
UP9	E2EA:	E2C0
UP6	E2EE:	E2BC
UP2	E2F3:	E2EC
UP1	E303:	E2FB
UP3	E312:	E309
NXT2	E322:	E2F5
NXT6	E32A:	E324
	E339:	E32C
JLP2	E340:	E2E8 E301 E305 E310 E320 E331 E338
NXLN	E343:	\$E196 \$E270 \$E289 \$E363
NXLN2	E348:	E352
NXLN1	E350:	E348
NXTL	E359:	#E21A #E2B7
SCROL	E369:	\$E1C4 \$E34D
SCRL1	E37B:	E377
MLP1	E382:	E387 E391
MLP2	E39B:	E3A1
SCRL4	E3A7:	E3C2

Label Address Referenced by

SCRL5	E3AB:	E3BA
SCRL3	E3B5:	E3B1
MLP4	E3D2:	E3D8
MLP4·1	E3D5:	E3D8
MLP4·2	E3DF:	E3CE
NEWLN	E3E2:	\$E2D2
NEWLX	E3F3:	E3EE
NEXL1	E3F5:	E41E
NEWLA	E401:	E3FD
NELL	E414:	E419
BLKLN	E421:	E3EC
BLKL	E434:	E437
PULS1	E452:	E44D
KEY5	E470:	E46A
KEY4	E474:	E45A E45E
KEY3	E495:	E48D
KL24	E49B:	E493
KL2	E49E:	E497
KL23	E4A9:	E49E
KL25	E4B2:	E4A7
KL22	E4B5:	E4AB E4B0 E4E7
KL1	E4BF:	E4E2
CKIS1	E4CE:	E4C6
SPCK	E4DB:	E4D4
CKUT	E4DD:	E4CC E4D0 E4D9
CKIT	E4DE:	E4C0
CKIT1	E4E9:	E4DF
KN1	E4FD:	E4F9
KEYF	E509:	E505
PRENDO	E50B:	E4ED E4F2
PREND	E600:	#E50B
DSPP	E606:	\$E0D0 \$E182
CHAR	E60A:	:E4C3 :E4F4
LDTB2	E65B:	:E087 :E1F8 :E299 :E40F :E42B
LDTB2	E65C:	:E40A
RUNTB	E673:	:E0DF
PIAL	E810:	>E01C :E47D >E482 :E487 >E4E4
PIAL1	E811:	>E03C >E37F >E3C6 >E3E7 >E43B :E4D6
PIAK	E812:	:E031 :E3CB :E4B7 :E4BA
PIAS	E813:	>E02E >E49B
IEEEIS	E821:	>E036
IEEEO	E822:	>E026 >E03F
IEEEOS	E823:	>E039
PIA	E840:	>E020 :E4A2 :E4AD >E4B2
P2DB	E842:	>E023
T1H	E845:	>E029 >E3D2
PCR	E84C:	>E044
IFR	E84D:	:E3D5
IER	E84E:	>E002
UDTIM	F768:	\$E455

```
*****
* Cross reference listing three. Area $E000-$E7FF, for 9" screen *
* 40 column models with B-keyboard (#1474-02). *
*****
*****
```

On the following pages, the cross reference listing of the BASIC 4.0 ROM 1474-02 is given. This ROM is used in all 40 column machines with a business- or B-keyboard that have a 9 inch screen.

The listing has the following format:

First a label is given, followed by the address of that label. The label name corresponds with the name in the assembler listing.

If the address has no label assigned to it, or the official CBM label name was not known, no label is stated. In this case, the label in the source listing is the hex address, written in lower case without a preceding dollar sign.

After the label and the reference address the addresses are printed that make use of the reference address. Each address is preceded with an extra token, that indicates the kind of use:

Token Kind of use

- none Jumped to by a relative branch.
- # Jumped to via a JMP absolute or a JMP indirect instruction.
- \$ Jumped to via a JSR absolute instruction.
- :
- > Location is read, and contents is left unchanged.
- > Location is written to, or accessed in read/modify/write fashion; previous data is therefore lost or altered.

Label Address Referenced by

CTIMR	008D:	>E009
CINV	0090:	>E010 #E452
	0091:	>E014
CRINV	0092::	#E44F
LSTX	0097:	:E4F9 :E511 >E513
SHFLAG	0098:	>E482 >E4D8 >E521
NDX	009E:	:E0B3 >E0B7 :E0BF >E0D0 >E300 :E538 >E544
RVS	009F:	>E064 :E176 >E25B >E328 >E350
INDX	00A1:	>E0FC :E110 :E130
LXSP	00A3:	:E104 >E1C7 >E344
LSTP	00A4:	:E10C
SFDX	00A6:	>E47F >E4E9 :E4F7
BLNSW	00A7:	>E049 >E0C1 :E458
BKLCT	00A8:	>E047 >E45C >E46B
GDBLN	00A9:	:E0CA >E477
BLNON	00AA:	:E0C6 >E0CE >E46F >E475
CRSW	00AC:	>E0F0 :E11A >E143 >E20B
DFLTN	00AF:	:E147
DFLTO	00B0:	>E018 :E140
PNT	00C4:	>E066 >E08A :E0F2 :E120 >E1FB :E240 >E243 >E24C >E29C :E2C4 :E2D8 >E2D8 >E2E4 >E36B >E384 >E395 >E39B >E39F

Label Address Referenced by

PNTR	00C6:	>E07B :E09C >E0A4 >E100 >E10E :E11E >E138 >E185 :E189 >E19B :E1D4 >E1DB >E1E4 >E1FD :E20D >E235 >E269 >E275 >E295 :E2CA :E2DE :E2F7 >E2FF :E319 >E31E >E32F >E361 :E46D >E563 >E582 :E5A1 :E5B1 :E606
SAL	00C7:	>E37B :E382 >E39D >E40D >E416
SAH	00C8:	>E36F >E389 :E38F >E3F9
QTSW	00CD:	>E102 :E130 :E16B >E16F >E1AC :E250 :E2BA >E35F
LNMX	00D5:	>E08E >E09A :E0EE :E187 >E1FF :E246 :E26C :E283 :E2C2 :E2D5 :E55D :E561
TBLX	0008:	>E07D :E07F :E108 :E18D >E1C9 :E1CB :E1E0 >E1F4 >E290 >E297 :E303 >E30B >E314 :E346 >E354 >E3A5 >E3A9 :E3DF :E3E2 :E41C
DATA	00D9:	>E122 >E126 :E128 >E158 :E15E >E203 :E20F
INSRT	00DC:	:E17C >E180 :E1A8 :E229 >E2E6 :E2EA >E35B
	00DF:	:E1EA :E1EF :E307
LDTB1	00E0:	>E04F :E081 :E29E :E350 :E3AB >E3B5 :E3C0 :E3FB :E421 >E429 00E1: :E094 >E1CD >E1CF :E375 :E3AF :E3F5 >E401 00E2: :E1B7 >E1BB 00F8: >E3BE
CAS1	00F9:	>E498 :E49E
CAS2	00FA:	>E4A9 :E4B2
	0104:	:E448
	026E:	>E0E2
KEYD	026F:	:E0A7 >E0AF >E53A
	0270:	:E0AC
TAPE2	033A:	:E557 >E55A >E565 >E5A5 :E5B3 >E5BB
	033E:	>E56A >E576 :E57C :E598 >E5AE >E5B8
	03EE:	:E579 :E595 >E59B >E5C9
RPTFLG	03F8:	>E460 >E4C0 >E4E3 >E4FD
KOUNT	03F9:	>E502 :E505 >E50E 8000: >E06A 8100: >E060 8200: >E070 8300: >E073
PX1	E009:	E00C
CLSR	E04B:	\$E33D
LPS1	E04F:	E05F
LPS2	E05D:	E053 E057
LPS3	E05E:	E05B
LPS4	E06A:	E077
NXTD	E079:	\$E261
STUPT	E07F:	\$E1D7 \$E30D \$E316 #E356 #E43F
STUPZ	E09C:	E092 E096
STUPR	E0A6:	E0A0
LP2	E0A7:	\$E0D3
LP1	E0AC:	E0B5
LOOP4	E0BC:	E0EC
LOOP3	E0BF:	E0C3 E0E8 E11C
LP21	E0D3:	E0C8
LP23	E0DF:	E0E6
LP22	E0EA:	E0D8
CLP5	E0F2:	E0F9
CLP6	E0FB:	E0F6
LOPS	E11E:	E106 E10A E112
LOP54	E12E:	E12A
LOP52	E134:	E12E
LOP53	E138:	E132 E134

Label Address Referenced by

CLP2	E:141:	E114
CLP2A	E:153:	E14B
CLP21	E:156:	E151
CLP1	E:158:	E13F
CLP7	E:166:	E162
QTSWC	E:167:	\$E13A \$E223
QTSWL	E:173:	E169
NXT33	E:174:	#E2B0
NXT3	E:176:	#E226
NC3	E:17A:	#E22D #E254 #E2F0
NVS	E:17C:	E178
NVSL	E:182:	E17E
JSTS1	E:19F:	E191
LOOP2	E:1A6:	E18B E19D #E1C1 E1E8 #E277 #E28C #E336 #E340 #E366 #E584 #E59E
LOP2	E:1AE:	E1AA
JSTSX	E:1B3:	\$E193 \$E1D1
JSXB	E:1BD:	E1B5
JTS2	E:1BE:	E1A1
SCRL	E:1C4:	\$E1A3 #E3F0
JSTS2	E:1CD:	\$E1BE
BKLN	E:1DE:	\$E239 \$E333
BKLN1	E:1EA:	E1E2
NTCN2	E:1F3:	E1EC
PRT	E:202:	\$E0BC \$E153 E216: E211
NJTL	E:21D:	E218
NTCN	E:229:	E21F
CNC3X	E:230:	E22B
BK1	E:23F:	E237 E248
BK2	E:24A:	#E23C
NTCN1	E:250:	E232
NC3W	E:257:	E252
NQ1	E:25D:	E259
NC2	E:264:	E25F
JPL4	E:275:	E285 E287
NCZ2	E:277:	E26E
NCX2	E:27A:	E266 E27E: #E540
JPL3	E:28C:	E24E E2A2 E2D0
NXTX	E:2A4:	#E213
NXTX1	E:2AC:	E2A8 E2B3: E2AE
UPS	E:2BA:	E2B5
INS3	E:2CE:	E2C8
INS1	E:2D5:	E2CC
INS2	E:2D7:	E2E0
UP9	E:2EA:	E2C0
UP6	E:2EE:	E2BC
UP2	E:2F3:	E2EC
UP1	E:303:	E2FB
UP3	E:312:	E309
NXT2	E:322:	E2F5
NXT6	E:32A:	E324 E339: E32C E33D: #E58B
JLP2	E:340:	E2E8 E301 E305 E310 E320 E331
NXLN	E:343:	\$E196 \$E270 \$E289 \$E363

Label Address Referenced by

NXLN2	E348:	E352
NXLN1	E350:	. E34B
NXTL	E359:	#E21A #E2B7
SCROL	E369:	\$E1C4 \$E34D
SCRL1	E37B:	E377
MLP1	E382:	E387 E391
MLP2	E39B:	E3A1
SCRL4	E3A7:	E3C2
SCRL5	E3AB:	E3BA
SCRL3	E3B5:	E3B1
MLP4	E3D2:	E3DB
MLP41	E3D5:	E3D8
MLP42	E3DF:	E3CE
NEWLN	E3E2:	\$E2D2
NEWLX	E3F3:	E3EE
NEXL1	E3F5:	E41E
NEWLA	E401:	E3FD
NELL	E414:	E419
BLKLN	E421:	E3EC
BLKL	E434:	E437
PULS1	E452:	E44D
	E469:	E463 .
	E46B:	E467
KEY5	E479:	E473
KEY4	E47D:	E45A E45E
KEY3	E49E:	E496
KL24	E4A4:	E49C
KL2	E4A7:	E4A0
KL23	E4B2:	E4A7
KL25	E4BB:	E4B0
KL22	E4C3:	E4B4 E4CB E4F5
KL1	E4CD:	E4F0
CKIS1	E4DD:	E4D4
SPCK	E4E9:	E4DF
CKUT	E4EB:	#E4DA #E4E6
CKIT	E4EC:	. E4CE
CKIT1	E4F7:	. E4ED
	E513:	E4FB
	E536:	E527 E52B
KN1	E538:	E51F E523 E531
KEYF	E544:	E540
PREND0	E546:	E500 E50A E516
	E549:	#E27A
	E550:	E54B
	E557:	E57F
	E56A:	E55F
	E579:	E56D
	E584:	E552 E568 E572
	E587:	#E339
	E58E:	E589
	E59E:	E590
	E5A1:	\$E554 \$E592
	E5B3:	#E5BE
	E5C1:	E5B6
	E5C2:	\$E002
	E5C9:	E5C0
PREND	E600:	#E546
DSPP	E606:	\$E0D0 \$E182

Label Address Referenced by

CHAR	E60A:	:E4D1 :E518
LDTB2	E65B:	:E087 :E1F8 :E299 :E40F :E42B
	E65C:	:E40A
RUNTB	E673:	:E0DF
PIAL	E810:	>E01C :E486 >E48B :E490 >E4F2
PIAL1	E811:	>E03C >E37F >E3C6 >E3E7 >E43B
PIAK	E812:	:E031 :E3CB :E4C5 :E4C8
PIAS	E813:	>E02E >E4A4
IEEEIS	E821:	>E036
IEEEO	E822:	>E026 >E03F
IEEOS	E823:	>E039
PIA	E840:	>E020 :E4AB :E4B6 >E4BB
P2DB	E842:	>E023
T1H	E845:	>E029 >E3D2
PCR	E84C:	>E044
IFR	E84D:	:E3D5
IER	E84E:	>E5C2
UDTIM	F768:	\$E455

```
*****
*
* Cross reference listing four. Area $E000-$E7FF, for 12" screen      *
*          40 column models with N-keyboard (#1498-01).      *
*
*****
```

On the following pages, the cross reference listing of the BASIC 4.0 ROM 1498-01 is given. This ROM is used in all 40 column machines with a graphics- or N-keyboard that have a 12 inch screen, so-called FAT forties.

The listing has the following format:

First a label is given, followed by the address of that label. The label name corresponds with the name in the assembler listing.

If the address has no label assigned to it, or the official CBM label name was not known, no label is stated. In this case, the label in the source listing is the hex address, written in lower case without a preceding dollar sign.

After the label and the reference address the addresses are printed that make use of the reference address. Each address is preceded with an extra token, that indicates the kind of use:

Token Kind of use

none	Jumped to by a relative branch.
#	Jumped to via a JMP absolute or a JMP indirect instruction.
\$	Jumped to via a JSR absolute instruction.
:	Location is read, and contents is left unchanged.
>	Location is written to, or accessed in read/modify/write fashion; previous data is therefore lost or altered.

Label Address Referenced by

CTIMR	0080:	>E68C
CINV	0090:	#E452 >E69E
	0091:	>E6A2
CBINV	0092:	#E44F
LSTX	0097:	:E518 >E556
SHFLAG	0098:	>E4C4 >E4F2 >E563
NOX-	009E:	:E0B3 >E0B7 :E0BF >E0D0 >E425 :E552 :E57A >E585
RVS	009F:	>E056 :E176 >E266 >E359 >E3CA
INDX	00A1:	>E0FC :E110 :E13D
LXSP	00A3:	:E104 >E1C7 >E3AA
LXTP	00A4:	:E10C
SFDX	00A6:	>E4C1 >E508 :E516
BLNSW	00A7:	>E0C1 :E458 >E6D7
BLNCT	00A8:	>E45C >E469 >E60C >E6D5
GDBLN	00A9:	:E0CA >E475
BLNON	00AA:	:E0C6 >E0CE >E460 >E473
CRSW	00AC:	>E0F0 :E11A >E143 >E20B
DFLTN	00AF:	:E147
DFLTO	00B0:	:E14D >E6A6
PNT	00C4:	>E058 >E07C :E0F2 :E120 >E1FB :E24B >E24E >E257 >E2A4 :E2F5 :E309 >E30C >E315 >E382 >E3DB >E3FD >E40A :E46F >E470 >E4D8 >E745 >E747 >E774 >E777

Label Address Referenced by

PNTR	00C6:	>E06D :E08E >E096 >E100 >E10E :E11E >E138 >E185 :E189 >E19B :E104 >E10B >E1E4 >E1FD :E20D >E240 >E276 >E282 >E2C8 :E2FB :E30F :E328 >E330 :E34A >E34F >E362 :E37E >E3C4 :E46B :E606 :E643
SAL	00C7:	>E3F7 :E3FB >E610 :E630 >E710 >E719
SAH	00C8:	>E3F2 >E61F >E6FC
QTSW	00CD:	>E102 :E130 :E16B >E16F >E1AC :E25B :E2EB >E3CC
FNLEN	00D1:	>E626 :E629
LNMX	00D5:	>E080 >E08C :E0EE :E187 >E1FF :E251 :E279 :E290 :E2A6 :E2C2 :E2C6 :E2F3 :E306 :E640
TBLX	00D8:	>E06F :E071 :E108 :E18D >E1C9 :E1CB :E1E0 >E1F4 :E334 >E33C >E345 :E3AC >E3BA >E3D3 >E40F :E427 :E6EA :E71F
DATA	00D9:	>E122 >E126 :E128 >E158 :E15E >E203 :E20F :E21A
INSRT	00DC:	:E17C >E180 :E1A8 :E234 >E317 :E31B >E3C8
MYCH	00DF:	:E1EA :E1EF :E338
LDTB1	00E0:	>E048 :E073 :E3B6 :E3DD >E3ED >E404 :E411 :E6FE :E724 >E72C 00E1: :E086 >E1CD >E1CF :E3E7 :E6F8 >E704 00E2: :E1B7 >E1BB
CAS1	00F9:	>E48D :E496
CAS2	00FA:	>E4A4 :E4AD
	0104:	:E448
	026E:	>E0E2
KEYD	026F:	:E0A7 >E0AF >E581
	0270:	:E0AC
DELAY	03E9:	>E51E :E53E >E543 >E6E3
KOUNT	03EA:	>E548 >E54F >E6E6
XMAX	03EB:	:E57C >E6DB
	03EC:	:E657 :E66E >E6E0
	03ED:	>E431 :E434 >E43D
RPTFLG	03EE:	:E462 :E4C6 >E4CB :E4FA >E4FF :E523 >E699
	03EF:	:E3A0 >E58F :E59A
	03F0:	>E3A3 :E597 >E693
	8000:	>E05C
	8100:	>E05F
	8200:	>E062
	8300:	>E065
	E036:	#E000
CLSR	E042:	#E015 \$E370
LPS1	E048:	E051
LPS3	E050:	E04D
LPS4	E05C:	E069
NXTD	E06B:	\$E26E
STUPT	E071:	\$E1D7 \$E33E \$E347 #E3BC #E73C
STUPZ	E08E:	E084 E088
STUPR	E098:	#E02D #E030 #E033 E092
LP2	E0A7:	#E003 \$E003
LP1	E0AC:	E0B5
LOOP4	E0BC:	E0EC
LOOP3	E0BF:	E0C3 E0E8 E11C
LP21	E0D3:	E0C8
LP23	E0DF:	E0E6
LP22	E0EA:	E0D8
CLP5	E0F2:	E0F9
CLP6	E0FB:	E0F6
LOOP5	E116:	#E006
LOP5	E11E:	E106 E10A E112
LOP54	E12E:	E12A

Label Address Referenced by

LOP52	E134:	E12E
LOP53	E138:	E132 E134
CLP2	E141:	E114
CLP2A	E153:	E14B
CLP21	E156:	E151
CLP1	E158:	E13F
CLP7	E166:	E162
QTSWC	E167:	\$E13A \$E22E
QTSWL	E173:	E169
NXT33	E174:	#E2E1
NXT3	E176:	#E231
NC3	E17A:	#E238 #E25F #E321
NVS	E17C:	E178
NVSL	E182:	E17E
JSTS1	E19F:	E191
LOOP2	E1A6:	E1B8 E19D #E1C1 E1E8 #E284 #E299 #E2CA #E369 #E373 #E3A6 #E3CE
LOP2	E1AE:	E1AA
JSTSX	E1B3:	\$E193 \$E1D1
JSXB	E1BD:	E1B5
JTS2	E1BE:	E1A1
SCRL	E1C4:	\$E1A3 #E6F3
JSTS2	E1CD:	\$E1BE
BKLN	E1DE:	\$E244 \$E366
BKLN1	E1EA:	E1E2
NTCN2	E1F3:	E1EC
PRT	E202:	#E009 \$E153 \$E63C E21A: E215 E221: E21C
NJTL	E228:	E223
NTCN	E234:	E22A
CNC3X	E23B:	E236
BK1	E24A:	E242 E253
BK2	E255:	#E247
NTCN1	E25B:	E230
NC3W	E262:	E250
NC1	E26A:	E264
NC2	E271:	E26C
JPL4	E282:	E292 E294
NCZ2	E284:	E259 E268 E27B E301
NCX2	E287:	E273
JPL3	E299:	E2AA E2B3 E2BC E2CO E29C: E289 E2A3: E2A8 E2AC: E29E E2B5: E2AE E2BE: E2B7 E2C2: E2D1 E2C8: E2D3 E2CD: E2C4
NXTX	E205:	#E21E
NXTX1	E200:	E2D9 E2E4: E2DF
UP5	E2EB:	E2E6
INS3	E2FF:	E2F9
INS1	E306:	E2F0
INS2	E308:	E311
UP9	E31B:	E2F1

Label Address Referenced by

UP6	E31F:	E2ED
UP2	E324:	E31D
UP1	E334:	E32C
UP3	E343:	E33A
NXT2	E353:	E326
NXT6	E350:	E355
	E36C:	E35F
JLP2	E373:	E319 E332 E336 E341 E351 E35B E364 E380 E38E E397 E39B
	E376:	E36E
	E37E:	E385
	E387:	E378
	E390:	E389
	E399:	E392
NXLN	E3A9:	\$E196 \$E27D \$E296 \$E3BF
NXLN2	E3AE:	E3B8
NXLN1	E3B6:	E3B1
	E3BF:	#E225 #E2E8
NXTL	E3C6:	#E217
SCROL	E3D1:	#E024 \$E1C4 \$E3B3
	E3D5:	E413
	E3D7:	E402
	E3ED:	E3E9
MLP1	E3FB:	E400
	E404:	E3E5
MLP2	E40A:	E40D
	E41E:	E420 E423
MLP42	E427:	E41A
	E42E:	E440 #E455
PULS	E442:	#E00C
PULS1	E452:	E44D
KEY	E455:	#E00F
	E458:	E439
	E469:	E465
KEY5	E477:	E471
KEY4	E47B:	E45A E45E
KEY3	E496:	E48B
KL24	E49F:	E494
KL2	E4A2:	E498
KL23	E4AD:	E4A2
KL25	E4B6:	E4AB
	E4B9:	E4AF
	E4BF:	#E027 \$E4B9
KL22	E4D0:	E4D8 E514
KL1	E4E7:	E4DC E50F
	E4F6:	E4EE
CKIS1	E504:	E4F8
CKUT	E50A:	E4F4 E502 E506
CKIT	E50B:	E4E8
	E511:	E4E3
CKIT1	E516:	E4E5 E50C
	E523:	E51A
	E53E:	E530 E534 E538
	E548:	E526 E541
	E556:	E521
KN1	E57A:	E565
	E587:	E528 E52C E53C E546 E54B E554 E55A E57F
	E588:	\$E2CE \$E39D

Label Address Referenced by

PREND	E600:	#E012 #E4BC
DSPP	E606:	\$E000 \$E182
	E60F:	#E018 \$E2B0
	E617:	#E01B \$E039 \$E38B
	E61D:	#E01E E615
	E630:	E639
	E63C:	\$E0BC
	E654:	\$E03C \$E03F \$E394 E64C
	E657:	#E02A \$E2B9 \$E654
	E668:	E67A
	E671:	E672 E677
	E682:	E647 E652 E65A
	E683:	\$E036
PX1	E68C:	E68F
	E693:	E697
NEWLN	E6EA:	#E021 \$E303
NEWLX	E6F6:	E6F1
NEXL1	E6F8:	E721
NEWLA	E704:	E700
NELL	E717:	E71C
BLKLN	E724:	E6EF
BLKL	E737:	E73A
CHAR	E73E:	:E4EB
RUNTB	E78E:	:E0DF
LDTB2	E798:	:E079 :E1F8 :E3D8 :E712 :E72E
	E799:	:E3F4 :E700
	E7D4:	:E668
	E7DC:	:E58C
PIAL	E810:	:E47D >E482 :E485 >E511 >E6AA
PIAL1	E811:	>E6CA
PIAK	E812:	:E415 :E4D2 :E4D5 :E6BF
PIAS	E813:	:E48F :E49A >E49F >E6BC
IEEEIS	E821:	>E6C4
IEEEO	E822:	>E6B4 >E6CD
IEEEOS	E823:	>E6C7
PIA	E840:	:E4A6 :E4B1 >E4B6 >E6AE
P2DB	E842:	>E6B1
T1H	E845:	>E6B7
T2L	E848:	>E66B
SR	E84A:	>E663 >E67C
ACR	E84B:	>E65E >E67F
PCR	E84C:	:E621 >E62B >E6D2
IER	E84E:	>E685
CRO	E880:	>E632
CR1	E881:	>E635
udtim	FFEA:	\$E42E

```
*****
* Cross reference listing five. Area $E000-$E7FF, for 12" screen      *
*           80 column models with B-keyboard (#1474-04).          *
*                                                               *
*****
```

On the following pages, the cross reference listing of the BASIC 4.0 ROM 1474-04 is given. This ROM is used in all 80 column machines with a business- or B-keyboard that have a 12 inch screen.

The listing has the following format:

First a label is given, followed by the address of that label. The label name corresponds with the name in the assembler listing.

If the address has no label assigned to it, or the official CBM label name was not known, no label is stated. In this case, the label in the source listing is the hex address, written in lower case without a preceding dollar sign.

After the label and the reference address the addresses are printed that make use of the reference address. Each address is preceded with an extra token, that indicates the kind of use:

Token Kind of use

none	Jumped to by a relative branch.
#	Jumped to via a JMP absolute or a JMP indirect instruction.
\$	Jumped to via a JSR absolute instruction.
:	Location is read, and contents is left unchanged.
>	Location is written to, or accessed in read/modify/write fashion; previous data is therefore lost or altered.

Label Address Referenced by

CTIMR	0080:	>E61C
CINV	0090:	#E452 >E623
	0091:	>E627
CBINV	0092:	#E44F
LSTX	0097:	:E506 >E53F
SHFLAG	0098:	>E4C3 >E4E2 >E54C
NDX	009E:	:E0B3 >E0B7 :E0BF >E0D0 >E41D :E53A :E563 >E56D
RVS	009F:	:E179 >E260 >E35E >E3C1
INDX	00A1:	>E0FC :E110 :E140
LXSP	00A3:	:E104 >E3A3
LXTP	00A4:	:E10C
SFDX	00A6:	>E4C0 >E4F6 :E504
BLNSW	00A7:	>E0C1 :E458 >E65D
BLNCT	00A8:	>E45C >E468 >E60C >E65B
GDBLN	00A9:	:E0CA >E474
BLNON	00AA:	:E0C6 >E0CE >E46C >E472
CRSW	00AC:	>E0F0 :E110 >E146 >E20E
DFLTN	00AF:	:E14A
DFLTO	00B0:	:E150 >E62F
PNT	00C4:	>E072 :E0F2 :E123 >E1C4 :E252 >E255 >E25E >E2D8 :E314 :E323 >E326 >E32F >E39E >E3E0 >E400 :E46E >E478 >E608
POINT	00C5:	>E077

Label Address Referenced by

PNTR	00C6:	>E065 >E10E :E121 >E13B >E188 :E18C >E197 >E1B4 >E1BC :>E1CD :E210 :E244 >E24D :E24F >E288 >E2AF >E2CE :E31A :>E329 :E334 :E366 >E36F :E39A >E3B8 :E46A :E570 :E580 :>E5A3 :E5E6 :E606 :E693
SAL	00C7:	>E088 :E09B >E3D6 :E3DE >E3F6 :E3FE
SAH	00C8:	>E08A >E3DB >E3FB
QTSW	00CD:	>E102 :E133 :E16E >E172 >E1A3 :E262 :E30A >E3C3
FNLEN	00D1:	>E091 :E094
LNMX	00D5:	:E0EE :E18A :E1AA :E1C6 >E1DE :E2 8 :E28B :E2A9 :E2AD :>E2DA :E312 :E320 :E331 :E3E2 :E402 >E5E8 :E690
TBLX	00D8:	>E061 :E067 :E108 :E190 :E1AE >E1BA :E2EA :E34D >E351 :>E3A5 >E3B1 :E59F :E5C3 :E5E2
DATA	00D9:	>E125 >E129 :E12B >E15B :E161 >E203 :E212 :E21D
INSRT	00DC:	:E17F >E183 :E19F :E237 :E336 >E33A :E33E >E3BF :00E0: :E051 :E05F :E1AC >E1E1 :E2E7 >E2EC :E34B :E3CF :E3E8 :>E5A1 :E5C0 >E5C5 >E5CB
	00E1:	:E05B >E1DC :E3A7 :E3C8 :E3EF >E5E4
	00E2:	:E063 :E06C :E195 :E1B2 :E1CB >E1E3 :E242 :E364 :E398 :>E3B6 >E5A5
XMAX	00E3:	:E565 >E62B
RPTFLG	00E4:	>E036 :E462 :E4C5 >E4C9 :E4EA >E4EE :E510 >E61A
DELAY	00E5:	>E532 >E538 >E661
KOUNT	00E6:	>E50C :E52A >E52E >E65F :00E7: >E685 :E6A7 :E6BD :00E8: >E19B :E273 >E27B >E618 :00E9: #E11A >E677 :00EA: >E679 :00EB: #E209 >E67F :00EC: >E681
	00F8:	>E434 :E436 >E43E
CAS1	00F9:	>E48C :E495
CAS2	00FA:	>E4A3 :E4AC
	0104:	:E448
	026E:	>E0E2
KEYD	026F:	:E0A7 >E0AF >E569
	0270:	:E0AC
TAPE2	033A:	:E2A3 >E2A6 >E2B1 >E574 :E582 >E58A :033E: >E2B6 >E2C2 :E2C8 :E386 >E570 >E587 :03EE: :E2C5 :E383 >E389 >E66D :E036: #E02D :E04B: #E000
CLSR	E051:	#E015 \$E377
	E054:	E05D
NXTD	E05F:	\$E27D
	E063:	\$E5C0
STUPT	E067:	#E1BE \$E353 #E3AE #E3B3 \$E5D9
	E06C:	\$E055 \$E3CC \$E3EC
	E06F:	#E069
	E07A:	#E018 \$E04E \$E5AE
	E082:	#E01B \$E5F1
	E088:	#E01E E080
	E09B:	E0A4
LP2	E0A7:	#E003 \$E0D3
LP1	E0AC:	E0B5
LOOP4	E0BC:	E0EC
LOOP3	E0BF:	E0C3 E0E8 E11F
LP21	E0D3:	E0C8
LP23	E0DF:	E0E6

Label Address Referenced by

LP22	E0EA:	E008
CLP5	EOF2:	EOF9
CLP6	EOFB:	EOF6
LOOP5	E116:	#E006
LOPS	E121:	E106 E10A E112
LOP54	E131:	E120
LOP52	E137:	E131
LOP53	E13B:	E135 E137
CLP2	E144:	E114
CLP2A	E156:	E14E
CLP21	E159:	E154
CLP1	E15B:	E142
CLP7	E169:	E165
QTSWC	E16A:	\$E13D \$E231
QTSWL	E176:	E16C
NXT33	E177:	#E300
NXT3	E179:	#E234
NC3	E17D:	#E23B #E266 #E344
NVS	E17F:	E17B
NVSL	E185:	E181
	E192:	#E28F
LOOP2	E199:	E18E E188 #E299 #E38C #E3C5 #E5A7 #E5EA
	E190:	#E280
LOP2	E1A5:	E1A1
BKLN	E1AA:	\$E248 \$E36A
	E1BA:	E1B0
	E1C1:	\$E058 \$E408
	E1C3:	E1C8
	E1CB:	\$E0FE
	E1D2:	\$E277 \$E666
	E1DC:	#E033
	E1E1:	#E030 \$E1D5
PRT	E202:	#E009 \$E156 \$E68C
	E21D:	E218
	E224:	E21F
NJTL	E22B:	E226
NTCN	E237:	E220
CNC3X	E23E:	E239
	E24D:	E246
BK1	E251:	E25A
BK2	E25C:	E24B
NTCN1	E262:	E240
NC3W	E269:	E264
NC1	E26F:	E26B
	E27B:	E275
NC2	E283:	E271
NCX2	E292:	E285
JPL3	E299:	E260 E280 E2B4 E2BE E2DE
	E29C:	E294
	E2A3:	E2CB
	E2B6:	E2AB
	E2C5:	E2B9
	E200:	E29E
	E2D7:	E2DC
	E2E0:	E2D2
	E2E7:	E2E2
NXTX	E2F4:	#E221
NXTX1	E2FC:	E2F8

Label Address Referenced by

	E303:	E2FE
UPS	E30A:	E305
INS2	E322:	E32B
UP9	E33E:	E310
UP6	E342:	E30C
UP2	E347:	E340
NXT2	E358:	E349
NXT6	E360:	E35A
	E36F:	E368
	E373:	E362
	E37C:	E375
JLP2	E38C:	E318 E31C E31E E338 E33C E34F E356 E360 E371
		E37A E39C
	E38F:	E37E
	E396:	E391
	E39A:	E3A1
NXLN	E3A3:	\$E192 \$E296 \$E3BA
	E3B1:	E3A9
	E3B6:	#E228 #E307
NXTL	E3BD:	#E21A
NEWLN	E3C8:	#E021 \$E5C7 \$E5D6
	E3CB:	E3E6
	E3D0:	E3E4
SCROL	E3E8:	#E024 \$E2EE \$E3AB \$E595
	E3EB:	E406
MLP1	E3FD:	E404
	E408:	E3D1 E3F1
	E414:	E416 E419
	E41B:	E42F
	E41F:	E422
	E420:	E410
	E424:	E429 E42D
	E431:	E440 #E455
PULS	E442:	#E00C
PULS1	E452:	E44D
KEY	E455:	#E00F
	E458:	E43A
	E468:	E464
KEY5	E476:	E470
KEY4	E47A:	E45A E45E
KEY3	E495:	E48A
KL24	E49E:	E493
KL2	E4A1:	E497
KL23	E4AC:	E4A1
KL25	E4B5:	E4AA
	E4B8:	E4AE
	E4BE:	#E027 \$E4B8
KL22	E4CD:	E4D5 E502
KL1	E4D7:	E4FD
	E4E6:	E4DE
CKIS1	E4F2:	E4E8
CKUT	E4F8:	E4E4 E4F0 E4F4
CKIT	E4F9:	E4D8
CKIT1	E504:	E4FA
	E510:	E508
	E52A:	E51C E520 E524
	E532:	E512 E52C
	E53F:	E50E

Label Address References by

	E561:	E552	E556	
KN1	E563:	E54A	E54E	E55C
	E56F:	E514	E518	E528
	E570:	\$E2A0	\$E380	
	E582:	#E58D		
	E590:	E585		
	E591:	#E2E4		
	E598:	E593		
	E5A7:	E5B1	E5B5	E5BA
	E5AA:	E59D		
	E5B3:	E5AC	E5EF	
	E5BC:	#E393		
	ESCA:	#E2F1		
	E5D2:	E5BE		
	E5D9:	#E598		
	E5DE:	E5D4		
	ESEA:	E5D0	E5DC	E5F4
	E5ED:	E5E0		
PREND	E600:	#E012	#E4B8	
DSPP	E606:	\$E0D0	\$E185	
	E60F:	\$E04B		
PX1	E61C:	E61F		
	E66D:	E671		
	E68C:	\$E0BC		
	E6A4:	\$E687	E68A	E69C
	E6A7:	#E02A	\$E5B7	\$E6A4
	E6B7:	E6C8		
	E6BF:	E6C0	E6C5	
CHAR	E6D0:	:E4DB	E697	E6A2
RUNTB	E720:	:E0DF		
	E74D:	:E6B7		
LDTB2	E754:	:E3D3		
	E755:	:E06F		
	E756:	:E3F3		
LDTB1	E76D:	:E3D8		
	E76E:	:E074		
	E76F:	:E3F8		
PIAL	E810:	:E47C	>E481	:E484
	E811:		>E4FF	>E633
PIAL1	E812:	:E40B	:E424	:E4CF
	PIAK		:E402	:E648
PIAS	E813:	:E48E	:E499	>E49E
IEEEIS	E821:		>E645	
IEEEO	E822:		>E640	
IEEEOS	E823:		>E656	
PIA	E840:	:E4A5	:E4B0	>E4B5
P2DB	E842:		>E637	
T1H	E845:		>E63A	
T2L	E848:		>E640	
SR	E84A:		>E644	
ACR	E84B:		>E682	>E6CA
PCR	E84C:		>E6AD	>E6CD
IER	E84E:		>E68C	>E096
CRO	E880:		>E6B1	>E663
CR1	E881:		>E6A0	
udtim	FFEA:	\$E431		

```
*****
*
* Cross reference listing six. Area $E000-$E7FF, for 12" screen      *
*                           80 column models with N-keyboard.          *
*
*****
```

On the following pages, the cross reference listing of the BASIC 4.0 ROM for the Graphics 80 is given. This ROM has no CBM part number and can be used in a FAT 40 that has been converted to 80 columns.

The listing has the following format:

First a label is given, followed by the address of that label. The label name corresponds with the name in the assembler listing.

If the address has no label assigned to it, or the official CBM label name was not known, no label is stated. In this case, the label in the source listing is the hex address, written in lower case without a preceding dollar sign.

After the label and the reference address the addresses are printed that make use of the reference address. Each address is preceded with an extra token, that indicates the kind of use:

Token Kind of use

none	Jumped to by a relative branch.
#	Jumped to via a JMP absolute or a JMP indirect instruction.
\$	Jumped to via a JSR absolute instruction.
:	Location is read, and contents is left unchanged.
>	Location is written to, or accessed in read/modify/write fashion; previous data is therefore lost or altered.

Label Address Referenced by

CTIMR	0080:	>E618
CINV	0090:	#E452 >E627
	00 1:	>E62B
CBINV	0092:	#E44F
LSTX	0097:	:E4FA >E524
SHFLAG	0098:	>E4C3 >E4DC >E530
NDX	009E:	:E0B3 >E0B7 :E0BF >E0D0 >E428 :E514 :E536 >E54A
RVS	009F:	:E179 >E260 >E35E >E3C1
INDX	00A1:	>E0FC :E110 :E140
LXSP	00A3:	:E104 >E3A3
LXTP	00A4:	:E1DC
SFDX	00A6:	>E4C0 >E4E4 :E4F4
BLNSW	00A7:	>E0C1 :E458 >E664
BLNCT	00A8:	>E45C >E468 >E51F >E662
GDBLN	00A9:	:E0CA >E474
BLNON	00AA:	:E0C6 >E0CE >E46C >E472
CRSW	00AC:	>E0FO :E110 >E146 >E20E
DFLTN	00AF:	:E14A
DFLTO	00B0:	:E150 >E62F
PNT	00C4:	>E072 :EOF2 :E123 >E1C4 :E252 >E255 >E25E >E208 :E314 :E323 >E326 >E32F >E39E >E3E0 >E400 :E46E >E478 >E608
POINT	00C5:	>E077

Label Address Referenced by

PNTR	00C6:	>E065 >E10E :E121 >E13B >E188 :E18C >E197 >E1B4 >E1BC :>E1CD :E210 :E244 >E24D :E24F >E288 >E2AF >E2CE :E31A :>E329 :E334 :E366 >E36F :E39A >E3B8 :E46A :E570 :E580 :>E5A3 :E5E6 :E606 :E693
SAL	00C7:	>E088 :E09B >E306 :E3DE >E3F6 :E3FE
SAH	00C8:	>E08A >E30B >E3FB
QTSW	00CD:	>E102 :E133 :E16E >E172 >E1A3 :E262 :E30A >E3C3 >E542
FNLEN	00D1:	>E091 :E094 .
LNMX	00D5:	:E0EE :E18A :E1AA :E1C6 >E1DE :E258 :E28B :E2A9 :E2AD :>E2DA :E312 :E320 :E331 :E3E2 :E402 >E5E8 :E690
TBLX	00D8:	>E061 :E067 :E108 :E190 :E1AE >E1BA :E2EA :E34D >E351 :>E3A5 >E3B1 :E59F :E5C3 :E5E2
DATA	00D9:	>E125 >E129 :E12B >E15B :E161 >E203 :E21D
INSRT	00DC:	:E17F >E183 :E19F :E237 :E336 >E33A :E33E >E3BF >E544
	00E0:	:E051 :E05F :E1AC >E1E1 :E2E7 >E2EC :E34B :E3CF :E3E8 :>E5A1 :E5C0 >E5C5 >E5CB
	00E1:	:E05B >E1DC :E3A7 :E3C8 :E3EF >E5E4
	00E2:	:E063 :E06C :E195 :E1B2 :E1CB >E1E3 :E242 :E364 :E398 :>E3B6 >E5A5
XMAX	00E3:	:E538 >E633
RPTFLG	00E4:	>E036 :E4FE >E666
DELAY	00E5:	>E519 >E522 >E668
	00E7:	>E684 :E6A7 :E6BD
	00E8:	>E19B :E273 >E27B
	00E9:	#E11A >E676
	00EA:	>E678
	00EB:	#E209 >E67E
	00EC:	>E680
	00F8:	>E434 :E436 >E43E
CAS1	00F9:	>E48C :E495
CAS2	00FA:	>E4A3 :E4AC
	0104:	:E448
	026E:	>E0E2
KEYD	026F:	:E0A7 >E0AF >E546
	0270:	:E0AC
TAPE2	033A:	:E2A3 >E2A6 >E2B1 >E574 :E582 >E58A
	033E:	>E2B6 >E2C2 :E2C8 :E386 >E57D >E587
	03EE:	:E2C5 :E383 >E389 >E61F
	E036:	#E02D
	E04B:	#E000
CLSR	E051:	#E015 \$E377
	E054:	E05D
NXTD	E05F:	\$E27D
	E063:	\$E5C0
STUPT	E067:	#E1BE \$E353 #E3AE #E3B3 \$E5D9
	E06C:	\$E055 \$E3CC \$E3EC
	E06F:	#E069
STUPZ	E07A:	#E018 \$E5AE
	E082:	#E01B \$E04E \$E5F1
	E088:	#E01E E080
	E09B:	E0A4
LP2	E0A7:	#E003 \$E003
LP1	E0AC:	E0B5
LOOP4	E0BC:	E0EC
LOOP3	E0BF:	E0C3 E0E8 E11F
LP21	E0D3:	E0C8
LP23	E0DF:	E0E6
LP22	E0EA:	E0D8

Label Address Referenced by

CLP5	E0F2:	E0F9
CLP6	E0F8:	E0F6
LOOPS	E116:	#E006
LOPS	E121:	E106 E10A E112
LOPS4	E131:	E12D
LOPS2	E137:	E131
LOPS3	E13B:	E135 E137
CLP2	E144:	E114
CLP2A	E156:	E14E
CLP21	E159:	E154
CLP1	E15B:	E142
CLP7	E169:	E165
QTSWC	E16A:	\$E13D \$E231
QTSWL	E176:	E16C
NXT33	E177:	#E300
NXT3	E179:	#E234
NC3	E17D:	#E23B #E266 #E344
NVS	E17F:	E17B
NVSL	E185:	E181
	E192:	#E28F
JSTS1	E199:	E18E E1B8 #E299 #E38C #E3C5 #E5A7 #E5EA
	E19D:	#E280
LOP2	E1A5:	E1A1
BKLN	E1AA:	\$E248 \$E36A
	E1BA:	E1B0
	E1C1:	\$E058 \$E408
LPS4	E1C3:	E1C8
	E1CB:	\$E0FE
	E1D2:	\$E277 \$E66F
	E1DC:	#E033
	E1E1:	#E030 \$E1D5
PRT	E202:	#E009 \$E156 \$E68C
	E224:	E21F
NJTL	E22B:	E226
NTCN	E237:	E22D
CNC3X	E23E:	E239
	E24D:	E246
BK1	E251:	E25A
BK2	E25C:	E248
NTCN1	E262:	E240
NC3W	E269:	E264
NC1	E26F:	E26B
	E27B:	E275
NC2	E283:	E271
NCX2	E292:	E285
JPL3	E299:	E260 E28D E2B4 E2BE E2DE
	E29C:	E294
	E2A3:	E2CB
	E2B6:	E2AB
	E2C5:	E2B9
	E2D0:	E29E
	E2D7:	E2DC
	E2E0:	E2D2
	E2E7:	E2E2
NXTX	E2F4:	#E221
NXTX1	E2FC:	E2F8
	E303:	E2FE
UPS	E30A:	E305

Label Address Referenced by

INS2	E322:	E32B
UP9	E33E:	E310
UP6	E342:	E30C
UP2	E347:	E340
NXT2	E358:	E349
NXT6	E360:	E35A
	E36F:	E368
	E373:	E362
	E37C:	E375
JLP2	E38C:	E318 E31C E31E E338 E33C E34F E356 E360 E371
		E37A E39C
	E38F:	E37E
	E396:	E391
	E39A:	E3A1
NXLN	E3A3:	\$E192 \$E296 \$E3BA
	E3B1:	E3A9
	E3B6:	#E228 #E307
NEWLN	E3C8:	#E021 \$E5C7 \$E5D6
	E3CB:	E3E6
NELL	E3D0:	E3E4
SCROLL	E3E8:	#E024 \$E2EE \$E3AB \$E595
	E3EB:	E406
MLP1	E3FD:	E404
	E408:	E3D1 E3F1
	E414:	E416 E419
	E41D:	E412
	E421:	E426
	E428:	E41B
	E42A:	E41F
	E431:	E440 #E455
PULS	E442:	#E00C
PULS1	E452:	E440
KEY	E455:	#E00F
	E458:	E43A
KEY5	E476:	E470
KEY4	E47A:	E45A E45E
KEY3	E495:	E48A
KL24	E49E:	E493
KL2	E4A1:	E497
KL23	E4AC:	E4A1
KL25	E4B5:	E4AA
	E4B8:	E4AE
	E4BE:	#E027 \$E4B8
KL1	E4C7:	E4CF E4FO
	E4D1:	E4EB
	E4E0:	E4D8
	E4E6:	E4DE E4E2
	E4E7:	E4D2
	E4F2:	E4E8
	E514:	E502 E506 E50A E50E
	E522:	E4F8 E4FC
	E536:	E52E E532
	E546:	E53E
	E54C:	E500 E512 E517 E51B E528 E53A
	E570:	\$E2A0 \$E380
	E582:	#E580
	E590:	E585
	E591:	#E2E4

Label Address Referenced by

	E59B:	E593
	ESA7:	E5B1 E5B5 E5BA
	ESAA:	E59D
	E5B3:	E5AC E5EF
	E5BC:	#E393
	E5CA:	#E2F1
	E5D2:	E5BE
	E5D9:	#E598
	E5DE:	E5D4
	ESEA:	E5D0 E5DC E5F4
	ESED:	E5E0
PREND	E600:	#E012 #E4BB
DSPP	E606:	\$E000 \$E185
	E60F:	\$E04B
PX1	E618:	E61B
	E61F:	E623
	E68C:	\$E0BC
	E6A4:	\$E686 E689 E69C
	E6A7:	#E02A \$E5B7 \$E6A4
	E6B7:	E6C8
	E6BF:	E6C0 E6C5
CHAR	E6D0:	:E4D5 E697 E6A2 E6A9
RUNTB	E720:	:E0DF
	E74D:	:E6B7
LDTB2	E754:	:E3D3
	E755:	:E06F
	E756:	:E3F3
LDTB1	E76D:	:E3D8
	E76E:	:E074
	E76F:	:E3F8
PIAL	E810:	:E47C >E481 :E484 >E4ED >E637
PIAL1	E811:	>E657
PIAK	E812:	:E40D :E421 :E4C9 :E4CC :E64C
PIAS	E813:	:E48E :E499 >E49E >E649
IEEEIS	E821:	- >E651
IEEEO	E822:	. >E641 >E65A
IEEEOS	E823:	>E654
PIA	E840:	:E4A5 :E4B0 >E4B5 >E63B
P2DB	E842:	>E63E
T1H	E845:	>E644
T2L	E848:	>E68A
SR	E84A:	>E6B2 >E6CA
ACR	E84B:	>E6AD >E6CD
PCR	E84C:	:E08C >E096 >E65F
IER	E84E:	>E611 >E66C
CRO	E880:	>E09D
CR1	E881:	>E0A0
udtim	FFEA:	\$E431

Addendum to the CBM BASIC 4.0 listing

Hindsight

The reader/user of the pages presented here before needs to be aware that the document is now more than 40 years old, and times were quite different then. Computers based on microprocessors were barely 5 years in existence, the standard setting IBM PC was just on sale, we did not have internet and email and the ROM contents of a computer was the closely guarded secret of the manufacturer. The main source of info and software to feed the computer were computer clubs or user groups. For the machines manufactured by then market leader Commodore there were several in the Netherlands, the major two being PBE (PET Benelux Exchange, the oldest) and VCGN (Vereniging Commodore Gebruikers Nederland).

How the listing came to be

The author had been very active in the PBE using his first machine, a PET 2001. His main interest was hardware and the inner workings of the machine. A breakthrough occurred when PBE published the listings of one the first machine language monitors: Supermon. The PET was upgraded to BASIC 2.0, its RAM was extended to 32 kbyte and somewhat later made switchable between the original ROM set and BASIC 2.0. After a while a 4040 floppy disk drive appeared, and an Epson MX-80F/T printer.

At the time hobbyists used Commodores 40 column machines; many had an original PET 2001 often expanded to 32 kbyte of RAM, others bought the newer 3000 series machines. By this time the automation in the office began in earnest and Commodore provided the necessary hardware in the form of the 80 column 8032 computer complete with a new BASIC 4.0 that provided faster string garbage collect compared to BASIC 2.0 and allowed decent handling of the disk drive in BASIC. PBE published a document that was eagerly awaited: a detailed ROM listing of both the original PET 2001 and the 3000 BASIC 2.0 series machines.

The author was severely hampered by the 40 column width of his Commodore, and learnt that it was possible to convert a so called FAT40 (40 column 12 inch model with N keyboard) to 80 column mode. He was very accustomed to the N-keyboard layout and found the 8032's B-layout cumbersome. After some time a second hand FAT40 machine was acquired, and the conversion of an 8032 E-ROM to the N-keyboard was undertaken. This was done by disassembling the 8032 ROM and converting it to the format of the assembler which was then in use: Carl Moser's MAE.

The first and oldest part of the listing presented before was now complete. So why not do the rest of the ROM contents? After all, nobody, as far as was known at the time, had made a BASIC 4.0 listing. Some five months later, the project was finished.

Within the PBE, nobody was interested. Its founder, Johan Smilde, saw no opportunities to make a profitable project out of it. In the end the VCGN was interested and in their offices a complete printout was made, which later on was photocopied and sold as serial numbered books. I received one of them as a gift.

Styling

Pages 1 though 393 of this document have been represented in the form the listing was originally published: printed on a matrix printer and then photocopied and bundled. The resulting styling has been preserved for those pages.

Page 1 is wrong !

Again with hindsight, page 1 claims a few facts which are now known to be not (entirely) correct. The first omission, unknown to the author at that time, is of course the 60Hz frame rate variants of the editor ROMs for the CRTC based models. The listed ROM numbers are correct for the 50Hz versions. The 60 Hz variants have the following CBM ROM numbers:

- 40 column: 901499-01
- 80 column: 901447-03

See page 399 onwards for more details.

The second omission is the fact that BASIC 4.0 was also used in later machines, notably the 8096, the SuperPET (also known as SP9000 or Micro Main Frame) and the 8296(D). These machines use the exact same ROM software in exactly the same ROMs, apart from the 8296(D) which had the B-, C-, D- and F-ROM combined into one larger 16 kbyte 23128 ROM with the number 324746-01, due to its newer main board. The SK, or Separate Keyboard models of the 8032 and 8096 use the same main board as the models with integrated keyboard and therefore also the same ROMs.

A slightly modified BASIC 4.0 is also found in the later CBM-II machines (B- and P-series in the US, 500-, 600- and 700-series in Europe). As these are very scarce they are not dealt with in this document.

Internationalization

When the various machines were initially designed they all had a keyboard layout and character generator suitable for use in the United States. Beginning around 1982, variations of models began to appear that had a different (and larger) editor ROM and an adapted character generator ROM, to provide support for locally required characters not available in the US-originated/designed machines. As far it is known to the author, this only involved the replacement of the E-ROMs and the character generator. It is unclear whether these changes were the result of actions by Commodore itself (probably true for the German speaking market because of the production facilities in Braunschweig, Germany), or that local distributors were (also) responsible for it. In surviving repositories, traces of both phenomena can be found.

The ROM listing presented before on pages 1 through 393 only deals with the ROMs which are characteristic for the US-market and the 50Hz variants of them. This represents the state of the hardware sold in Netherlands and as such the form known then by the author.

Model names and numbers

In the early years Commodore used different model numbers in North America and in Europe, although by the time BASIC 4.0 appeared this practice was largely abandoned. On the internet the US convention is mostly used.

Description	US name	Europe name
Original 4/8k PET (built-in cassette)	PET 2001	PET
2 nd generation, 8k	PET 2001-8N	CBM 3008
2 nd generation, 16k, N keyboard	PET 2001-16N	CBM 3016
2 nd generation, 32k, N keyboard	PET 2001-32N	CBM 3032
2 nd generation, 16k, B keyboard	PET 2001-16B	CBM 3016B
2 nd generation, 32k, B keyboard	PET 2001-32B	CBM 3032B
2 nd generation PET (built-in cassette, green screen, BASIC 2.0)	-	PET 2001-8N

Remark: this pertains to the model number stated on the front of the machine, not the info on the type plate on the back. A European CBM 3016 with an N-keyboard states the number 3016 on the front, but the type number label states 2001-16N

This has the net effect in Europe that only machines with a built-in cassette are referred to as a PET, all others are called CBM's, whereas in most other parts of the world all machines with an integrated CRT monitor are usually referred to as a PET.

Something similar developed with the naming of the then three existing BASIC versions:

Version	Sign-on	US name	Europe name
1.0	*** COMMODORE BASIC ***	original ROMs	old ROMs
2.0	### COMMODORE BASIC ###	upgrade ROMs	new ROMs
4.0	*** COMMODORE BASIC 4.0 ***	BASIC 4	BASIC 4

In Europe version 2.0 is often incorrectly called 3.0 probably due to the fact that that version was sold in machines called CBM 3XXX. Version 3.0 was planned according to the official CBM source code now available (it was version 2.0 with fast garbage collect), but never released. An early hint of this fact was the sign on text of both the VIC-20 and the C-64, which state a version number (V2 or V2.0) and have exactly the same keyword set and bugs as the second generation ROMs.

The 8096

This machine is an 8032 expanded with an extra 64 kbyte of bank switched RAM that is added using an extra board mounted over the main board. The required signals are taken from the main board by removing the CPU from its socket and using a flat cable to the extra board. On the extra board a new CPU socket is provided. The extra board was also available separately to retrofit an existing 8032 or 8032 SK.

The extra board provides 4 banks of 16 kbyte of RAM located at addresses \$8000-\$BFFF and \$C000-\$FFFF. Each address area can have one of two possible 16 kbyte banks switched in, replacing screen RAM, ROM or I/O. Provisions are present to leave screen RAM and/or the I/O accessible. Control is through a write only register at \$FFF0 (a read yields the ROM content of that address).

There is no ROM support for the extra board; all software support is external and usually loaded from disk into RAM. The ROMs of an 8096 (SK) are therefore exactly the same as a standard 8032 (SK).

The SuperPET (a.k.a. SP9000, MMF or Micro Main Frame)

This machine is also based on a standard 8032 expanded with, depending on date of production, one or two extra boards. The board(s) hold an additional 64 kbyte of RAM, organised as 16 4 kbyte banks located at addresses \$9000 - \$9FFF. A (then) unique feature of the SuperPET was its second CPU, a Motorola 6809E. This CPU had its own ROM set but uses the same RAM, screen RAM and I/O as in 6502 mode. Switching CPU's was done with a toggle switch on the right side of the cabinet. The SuperPET is the only Commodore computer of its era to feature a 6551 based RS-232 port.

The official model number of the SuperPET was SP9000 and in Europe was also sold under name Micro Main Frame or MMF for short. The machine was developed by the University of Waterloo Canada and came with a set of compilers for BASIC, Fortran, Pascal, Cobol, APL and an 6809 assembler. These packages were loaded from disk and ran in 6809 mode. The SuperPET had an early form of an encryption chip on board: without the chip the Waterloo 6809 packages would not run.

The BASIC 4.0 ROM set for the 6502 CPU was exactly the same as that of the standard 8032.

The 8296(D)

is a further development of the 8096 using a new main board that takes advantage of more modern chips (larger ROMs, FPLA logic, 4164 DRAM chips). The machine has 128 kbyte of RAM in total of which is 96kbyte is actually usable. All 8296's are housed in the same cabinet as the 8032 SK and 8096 SK and feature a separate keyboard. An 8296D is 8296 with a built-in 8250 low profile dual disk drive.

As with the previous two models mentioned, the BASIC 4.0 ROM contents is the same as with the standard 8032, although only two chips are used to store it. For this reason, although the code is the same, the ROMs have different numbers.

The 60 Hz frame rate differences

As stated previously, the original listing only dealt with the 50 Hz variants of the editor ROMs. This part explains the differences between the 50 and 60 Hz frame rate editor ROMs. The other ROMs are the same.

The frame rate calculation formula is as follows (also see page 294, although the formula given there is wrong):

$$F = 1 / (R0 + 1) * CCLK * (R4+1) * (R8+1) + R5 * ((R0 + 1) * CCLK)$$

For 60 Hz text mode the frame frequency is:

$$F = 1 / (50 * 1\text{us} * 33 * 10 + 3 * 50 \text{ us}) = 1 / (16650) = 60.06 \text{ Hz}$$

For 60 Hz graphics mode the frame frequency is:

$$F = 1 / (50 * 1\text{us} * 41 * 8 + 5 * 50 \text{ us}) = 1 / (16650) = 60.06 \text{ Hz}$$

Note that in both modes the frame rate is not exactly 60.00 Hz; the clock is therefore a tad fast. Given the fixed character clock of 1.00 MHz and the desired line frequency of 20 kHz, a frame rate of exactly 60 Hz cannot be made using a 6845 CRTC and the values used are indeed the closest approximation.

In 50 Hz models however the clock is exactly correct in both screen modes.

40 column CRTC models with N-keyboard (FAT 40)

Compare to pages 226 and 250, address \$E42E onward.

The routine to speed up TI by 6/5 is not used, but Commodore apparently forgot to remove it as it still present in the ROM:

```

*****  

;*          *  

;* Clock correction.      *  

;* Add an extra clock tick in   *  

;* every five, because screen is  *  

;* refreshed with a 50 Hz rate    *  

;* instead of the usual 60 Hz.    *  

;*          *  

;* In the 60Hz version of the ROM *  

;* this routine is not called and *  

;* carried over from the 50Hz ROM *  

;* suggesting Commodore first     *  

;* made the 50Hz ROM and then     *  

;* patched the jump to this code  *  

;* out.                          *  

;*          *  

*****  

E42E- 20 EA FF  Le42e JSR udtim      ;do clock tick  

E431- EE ED 03           INC $03ED    ;adapt correction counter  

E434- AD ED 03           LDA $03ED  

E437- C9 06               CMP #6       ;if not five ticks done  

E439- D0 1D               BNE Le458    ;do rest of interrupt  

E43B- A9 00               LDA #0       ;else  

E43D- 8D ED 03           STA $03ED    ;reset correction count  

E440- F0 EC               BEQ Le42e    ;and do an extra tick

```

The assumption in the comment that the 50 Hz version of the ROM was made first is further made plausible by the fact that 60 Hz ROM has a higher and different number (1499-01 versus 1498-01, compare this to the versions for the 8032 which apparently were seen of revisions of the same thing as the ROM numbers are 1447-03 (60 Hz) and 1447-04 (50 Hz)).

Compare to page 226, address \$E455 onwards.

In the interrupt routine the TI clock is directly updated, not through the 6/5 speed up routine:

```
;*****  
;*          *  
;* Default interrupt routine, en- *  
;* tered 50/60 times per second.  *  
;*          *  
;* (Pointed to by $90).          *  
;*          *  
;*****  
KEY    ;IRQ-entry (pointed to by ($90))  
E455- 20 EA FF      JSR udtim      ;60Hz: do clock tick
```

Compare to page 234 and 235, address \$E7B1 onwards.

The two CRTC parameter tables are different, in order to generate the required 60 Hz instead of 50 Hz frame rate:

```

;*****
;*
;* Two tables of 18 constants *
;* each for the CRT-controller. *
;* The first table is for text   *
;* mode, the second for graphics *
;* mode.                         *
;*
;*****
;text mode table, 60 Hz (ROM 1499-01)

E7B1- 31      Le7b1 .BY 49      ;horizontal total
E7B2- 28          .BY 40      ;horizontal displayed
E7B3- 29          .BY 41      ;horizontal sync position
E7B4- 0F          .BY 15      ;horizontal sync width
E7B5- 20          .BY 32      ;vertical total
E7B6- 03          .BY 3       ;vertical total adjust
E7B7- 19          .BY 25      ;vertical displayed
E7B8- 1D          .BY 29      ;vertical sync position
E7B9- 00          .BY 0       ;interlace mode (off)
E7BA- 09          .BY 9       ;maximum scan line address
E7BB- 00          .BY 0       ;cursor start
E7BC- 00          .BY 0       ;cursor end
E7BD- 10          .BY 16      ;start address (hi) (inverts video)
E7BE- 00          .BY 0       ;start address (lo)
E7BF- 00          .BY 0       ;cursor (hi)
E7C0- 00          .BY 0       ;cursor (lo)
E7C1- 00          .BY 0       ;light pen (hi)
E7C2- 00          .BY 0       ;light pen (lo)

;graphics mode table, 60 Hz (ROM 1499-01)

E7C3- 31      Le7c3 .BY 49      ;horizontal total
E7C4- 28          .BY 40      ;horizontal displayed
E7C5- 29          .BY 41      ;horizontal sync position
E7C6- 0F          .BY 15      ;horizontal sync width
E7C7- 28          .BY 40      ;vertical total
E7C8- 05          .BY 5       ;vertical total adjust
E7C9- 19          .BY 25      ;vertical displayed
E7CA- 21          .BY 33      ;vertical sync position
E7CB- 00          .BY 0       ;interlace mode (off)
E7CC- 07          .BY 7       ;maximum scan line address
E7CD- 00          .BY 0       ;cursor start
E7CE- 00          .BY 0       ;cursor end
E7CF- 10          .BY 16      ;start address (hi) (inverts video)
E7D0- 00          .BY 0       ;start address (lo)
E7D1- 00          .BY 0       ;cursor (hi)
E7D2- 00          .BY 0       ;cursor (lo)
E7D3- 00          .BY 0       ;light pen (hi)
E7D4- 00          .BY 0       ;light pen (lo)

```

Compare to page 235, address \$E787:

Because of the changes, the checksum byte is also different:

```
;*****
;*          *
;* Checksum byte (over E-ROM).   *
;*          *
;*****  
CKSUME  
E787- 20      .BY $BB      ;checksum byte (ROM 1499-01)
```

Models 8032, 8096, SuperPET and 8296(D)

Compare to pages 249 and 250, address \$E431 onward.

The routine to speed up TI by 6/5 is of course missing:

E431- AA	TAX	;these fill bytes not present in 1447-04 ROM
E432- AA	TAX	
E433- AA	TAX	
E434- AA	TAX	
E435- AA	TAX	
E436- AA	TAX	
E437- AA	TAX	
E438- AA	TAX	
E439- AA	TAX	
E43A- AA	TAX	
E43B- AA	TAX	
E43C- AA	TAX	
E43D- AA	TAX	
E43E- AA	TAX	

Compare to page 250, address \$E455 onwards.

In the interrupt routine the TI clock is directly updated:

```
.BA $E455      ;fixed address in F-ROM
;*****
;*          *
;* Default interrupt routine.   *
;*          *
;* (Pointed to by ($90)).    *
;*          *
;*****  
E455- 20 EA FF KEY      JSR udtim      ;do clock tick (1447-03 ROM)
```

Compare to page 257 and 258, address \$E72A onwards.

The two CRTC parameter tables are different, in order to generate the required 60 instead 50 Hz frame rate:

```

;*****
;*
;* Two tables of 18 constants *
;* each for the CRT-controller. *
;* The first table is for text   *
;* mode, the second for graphics *
;* mode.                         *
;*
;*****
;text mode table, 60 Hz (ROM 1447-03)

E72A- 31      Le72a .BY 49      ;horizontal total
E72B- 28      .BY 40      ;horizontal displayed
E72C- 29      .BY 41      ;horizontal sync position
E72D- 0F      .BY 15      ;horizontal sync width
E72E- 20      .BY 32      ;vertical total
E72F- 03      .BY 3       ;vertical total adjust
E730- 19      .BY 25      ;vertical displayed
E731- 1D      .BY 29      ;vertical sync position
E732- 00      .BY 0       ;interlace mode (off)
E733- 09      .BY 9       ;maximum scan line address
E734- 00      .BY 0       ;cursor start
E735- 00      .BY 0       ;cursor end
E736- 10      .BY 16      ;start address (hi) (inverts video)
E737- 00      .BY 0       ;start address (lo)
E738- 00      .BY 0       ;cursor (hi)
E739- 00      .BY 0       ;cursor (lo)
E73A- 00      .BY 0       ;light pen (hi)
E73B- 00      .BY 0       ;light pen (lo)

;graphics mode table, 60 Hz (ROM 1447-03)

E73C- 31      Le73c .BY 49      ;horizontal total
E73D- 28      .BY 40      ;horizontal displayed
E73E- 29      .BY 41      ;horizontal sync position
E73F- 0F      .BY 15      ;horizontal sync width
E740- 28      .BY 40      ;vertical total
E741- 05      .BY 5       ;vertical total adjust
E742- 19      .BY 25      ;vertical displayed
E743- 21      .BY 33      ;vertical sync position
E744- 00      .BY 0       ;interlace mode (off)
E745- 07      .BY 7       ;maximum scan line address
E746- 00      .BY 0       ;cursor start
E747- 00      .BY 0       ;cursor end
E748- 10      .BY 16      ;start address (hi) (inverts video)
E749- 00      .BY 0       ;start address (lo)
E74A- 00      .BY 0       ;cursor (hi)
E74B- 00      .BY 0       ;cursor (lo)
E74C- 00      .BY 0       ;light pen (hi)
E74D- 00      .BY 0       ;light pen (lo)

```

Compare to page 259, address \$E787:

Because of the changes, the checksum byte is also different:

```
;*****
;*
;* Checksum byte (over E-ROM). *
;*
;*****
CKSUME
E787- 20 .BY $20 ;checksum byte (ROM 1447-03)
```

Graphics80 (FAT 40 converted to 80 columns)

Compare to pages 273 and 274, address \$E42B onward.

In the 60 Hz version, the 6/5 speed up routine is not assembled. Depending on the way the (EP)ROM is made the bytes between \$E42B and \$E442 are either \$AA (copying the ROM 1447-03) or the empty state (\$FF).

Compare to page 274, address \$E455 onwards.

In the interrupt routine the TI clock is directly updated, not through the 6/5 speed up routine:

```
.BA $E455 ;fixed address in F-ROM
;*****
;*
;* Default interrupt routine. *
;*
;* (Pointed to by ($90)). *
;*
;*****
E455- 20 EA FF KEY JSR udtim ;do clock tick (60 Hz frame rate)
```

Compare to page 282 and 283, address \$E72A onwards.

The two CRTC parameter tables are different, in order to generate the required 60 instead 50 Hz frame rate:

```

;*****
;*
;* Two tables of 18 constants *
;* each for the CRT-controller. *
;* The first table is for text   *
;* mode, the second for graphics *
;* mode.                         *
;*
;*****  

;text mode table, 60 Hz frame rate  

E72A- 31      Le72a .BY 49          ;horizontal total  

E72B- 28      .BY 40          ;horizontal displayed  

E72C- 29      .BY 41          ;horizontal sync position  

E72D- 0F      .BY 15          ;horizontal sync width  

E72E- 20      .BY 32          ;vertical total  

E72F- 03      .BY 3           ;vertical total adjust  

E730- 19      .BY 25          ;vertical displayed  

E731- 1D      .BY 29          ;vertical sync position  

E732- 00      .BY 0           ;interlace mode (off)  

E733- 09      .BY 9           ;maximum scan line address  

E734- 00      .BY 0           ;cursor start  

E735- 00      .BY 0           ;cursor end  

E736- 10      .BY 16          ;start address (hi) (inverts video)  

E737- 00      .BY 0           ;start address (lo)  

E738- 00      .BY 0           ;cursor (hi)  

E739- 00      .BY 0           ;cursor (lo)  

E73A- 00      .BY 0           ;light pen (hi)  

E73B- 00      .BY 0           ;light pen (lo)  

;graphics mode table, 60 Hz frame rate  

E73C- 31      Le73c .BY 49          ;horizontal total  

E73D- 28      .BY 40          ;horizontal displayed  

E73E- 29      .BY 41          ;horizontal sync position  

E73F- 0F      .BY 15          ;horizontal sync width  

E740- 28      .BY 40          ;vertical total  

E741- 05      .BY 5           ;vertical total adjust  

E742- 19      .BY 25          ;vertical displayed  

E743- 21      .BY 33          ;vertical sync position  

E744- 00      .BY 0           ;interlace mode (off)  

E745- 07      .BY 7           ;maximum scan line address  

E746- 00      .BY 0           ;cursor start  

E747- 00      .BY 0           ;cursor end  

E748- 10      .BY 16          ;start address (hi) (inverts video)  

E749- 00      .BY 0           ;start address (lo)  

E74A- 00      .BY 0           ;cursor (hi)  

E74B- 00      .BY 0           ;cursor (lo)  

E74C- 00      .BY 0           ;light pen (hi)  

E74D- 00      .BY 0           ;light pen (lo)
-----
```