

!A

LLOAD ROM2E.L,D1,A\$4000

*** End of Pass 1

LLOAD C0.L,A\$4000

LLOAD C4.L,A\$4000

LLOAD SW.L,A\$4000

LLOAD C8.L,A\$4000

LLOAD CC.L,A\$4000

LLOAD D0.L,D2,A\$4000

LLOAD D4.L,A\$4000

LLOAD D8.L,A\$4000

LLOAD DC.L,A\$4000

LLOAD E0.L,D2,A\$4000

LLOAD E4.L,A\$4000

LLOAD E8.L,A\$4000

LLOAD EA.L,A\$4000

LLOAD EC.L,A\$4000

LLOAD F0.L,D2,A\$4000

LLOAD F4.L,A\$4000

LLOAD F8.L,A\$4000

LLOAD FC.L,A\$4000

LLOAD ROM2E.L,D1,A\$4000

*** End of Pass 2

```
0800      1          ttl "ROM Source Code, ROM2E.L"
0800      2          src "ROM2E.L,D1"
0800      3      ;
0800      4      ;
0800      5      ; ROM2E.L
0800      6      ;
0800      7      ;
0800      8      ; ROM2E Source Code, READ Tape only, with SWEET16, and new
0800      9      ; GARBAG Garbage Collection (Cornelis Bongers concept)
0800     10      ;
0800     11      ; Build Version #14
0800     12      ;
0800     13      ; 2025 April 12
0800     14      ;
0800     15      ;
0800     16      ; DOS 4.5, Build 08
0800     17      ;
0800     18      ; 2025 March 11
0800     19      ;
0800     20      ;
0800     21      ; Start of Source Code: 0x4000
0800     22      ; Start of Symbol List: 0x7000
0800     23      ;
0800     24      ;
0800     25      ; Copyright (c) 2025 March 11 by
0800     26      ; Walland Philip Vrbancic Jr
0800     27      ;
0800     28      ; 6223 East Peabody Street
0800     29      ; Long Beach, California 90808
0800     30      ; Unitied States of America
0800     31      ;
0800     32      ; All Rights Reserved
0800     33      ;
0800     34      ; This software is the confidential and
0800     35      ; proprietary intellectual property of
0800     36      ; Walland Philip Vrbancic Jr
0800     37      ;
0800     38      ;
0000     39      LOC0      epz $00
0001     40      LOC1      epz $01
0002     41      LOC2      epz $02
0800     42      ;
0000     43      R0L      epz $00
0001     44      R0H      epz $01
0018     45      R12L     epz $18
0019     46      R12H     epz $19
001C     47      R14L     epz $1C
001D     48      R14H     epz $1D
001E     49      R15L     epz $1E
001F     50      R15H     epz $1F
0800     51      ;
0000     52      GOWARM    epz $00          ; 00:02, JMP RESTART
0003     53      GOSTROUT epz $03          ; 03:05, JMP STROUT
000A     54      GOUSR     epz $0A          ; 0A:0C, JMP <USER ADDR>
0090     55      JMPADRS   epz $90          ; 90:92, GETS JMP ...
0800     56      ;
000D     57      BYTVALUE  epz $0D          ; for EXP and POWER
000D     58      CHARAC    epz $0D
000E     59      ENDCHR    epz $0E
000F     60      EOLPTR    epz $0F
```

000F	61	NUMDIM	epz	\$0F	
000F	62	TOKNCNTR	epz	\$0F	
0800	63	;			
0010	64	DIMFLG	epz	\$10	
0011	65	VALTYP	epz	\$11	; 11:12
0013	66	DATAFLG	epz	\$13	
0013	67	GARFLG	epz	\$13	
0014	68	SUBFLG	epz	\$14	
0015	69	INPUTFLG	epz	\$15	
0016	70	CPRMASK	epz	\$16	
0016	71	TOGLFLG	epz	\$16	
001A	72	SHAPE	epz	\$1A	; 1A:1B
001C	73	COLBITS	epz	\$1C	
001D	74	COLCOUNT	epz	\$1D	
0800	75	;			
0020	76	WNDLFT	epz	\$20	
0021	77	WNDWDTH	epz	\$21	
0022	78	WNDTOP	epz	\$22	
0023	79	WNDBTM	epz	\$23	
0024	80	CH	epz	\$24	
0025	81	CV	epz	\$25	
0026	82	GBASL	epz	\$26	
0027	83	GBASH	epz	\$27	
0028	84	BASL	epz	\$28	
0029	85	BASH	epz	\$29	
002A	86	BAS2L	epz	\$2A	
002B	87	BAS2H	epz	\$2B	
002C	88	H2	epz	\$2C	
002C	89	LMNEM	epz	\$2C	
002D	90	V2	epz	\$2D	
002D	91	RMNEM	epz	\$2D	
002E	92	CHKSUM	epz	\$2E	
002E	93	FORMAT	epz	\$2E	
002E	94	MASK	epz	\$2E	
002F	95	LENGTH	epz	\$2F	
002F	96	LASTIN	epz	\$2F	
002F	97	SIGN	epz	\$2F	
0800	98	;			
0030	99	COLOR	epz	\$30	
0030	100	HMASK	epz	\$30	
0031	101	MODE	epz	\$31	
0032	102	INVFLG	epz	\$32	
0033	103	PROMPT	epz	\$33	
0034	104	YSAV	epz	\$34	
0035	105	YSAV1	epz	\$35	
0036	106	CSWL	epz	\$36	
0037	107	CSWH	epz	\$37	
0038	108	KSWL	epz	\$38	
0039	109	KSWH	epz	\$39	
003A	110	PCL	epz	\$3A	
003B	111	PCH	epz	\$3B	
003C	112	A1L	epz	\$3C	
003D	113	A1H	epz	\$3D	
003E	114	A2L	epz	\$3E	
003F	115	A2H	epz	\$3F	
0800	116	;			
0040	117	A3L	epz	\$40	
0041	118	A3H	epz	\$41	
0042	119	A4L	epz	\$42	
0043	120	A4H	epz	\$43	
0044	121	A5L	epz	\$44	

```

0044      122  MACSTAT  epz $44
0044      123  OPRND   epz $44
0045      124  T2GUARD epz $45
0045      125  AREG    epz $45
0046      126  XREG    epz $46
0047      127  YREG    epz $47
0048      128  PREG    epz $48
0049      129  SPNT    epz $49
004E      130  RNDL    epz $4E
004F      131  RNDH    epz $4F
0800      132  ;
0050      133  LINNUM  epz $50          ; 50:51
0050      134  ACL     epz $50
0051      135  ACH     epz $51
0052      136  TEMPPT  epz $52
0053      137  LASTPT  epz $53          ; 53:54
0055      138  TEMPST  epz $55          ; 55:5D, str scratch name/len
005E      139  INDEX   epz $5E          ; 5E:5F
0800      140  ;
0060      141  DEST    epz $60          ; 60:61
0061      142  OFFSET  epz $61          ; trick assembler in SHFTFBY1
0062      143  MULMANT epz $62          ; 62:65
0066      144  MULGUARD epz $66
0067      145  PRGTAB  epz $67          ; 67:68
0069      146  VARTAB  epz $69          ; 69:6A
006B      147  ARYTAB  epz $6B          ; 6B:6C
006D      148  STREND  epz $6D          ; 6D:6E
006F      149  FRETOP  epz $6F          ; 6F:70
0800      150  ;
0071      151  FRESPEC epz $71          ; 71:72
0073      152  MEMSIZE epz $73          ; 73:74
0075      153  CURLIN  epz $75          ; 75:76
0077      154  OLDLIN  epz $77          ; 77:78
0079      155  TEXTPTR epz $79          ; 79:7A
007B      156  DATLIN  epz $7B          ; 7B:7C
007D      157  DATPTR  epz $7D          ; 7D:7E
007F      158  SRCPTR  epz $7F          ; 7F:80
0800      159  ;
0081      160  VARNAM  epz $81          ; 81:82
0083      161  VARPNT  epz $83          ; 83:84
0085      162  FORPNT  epz $85          ; 85:86
0087      163  LASTOP  epz $87
0087      164  TXPTRSAV epz $87          ; 87:88
0089      165  CPRTYPE epz $89
008A      166  FUNCNAM epz $8A          ; 8A:8B
008A      167  TEMP3   epz $8A          ; 8A:8E
008C      168  DSCPTR  epz $8C          ; 8C:8D
008F      169  T3GUARD epz $8F
0800      170  ;
0091      171  RTNADR  epz $91
0092      172  ARGGUARD epz $92
0093      173  TEMP1   epz $93          ; 93:97
0094      174  ARYPNT  epz $94          ; 94:95
0094      175  HIGHDS  epz $94          ; 94:95
0094      176  LEN     epz $94
0095      177  PROCESS epz $95
0096      178  HIGHTR  epz $96          ; 96:97
0098      179  TEMP2   epz $98          ; 98:9C
0099      180  COUNTER epz $99          ; FPOUT
009A      181  EXPCOUNT epz $9A          ; FPOUT, GETINT
009B      182  DPFLAG  epz $9B          ; GETINT

```

```

009B      183  LOWTR      epz  $9B      ; 9B:9C
009C      184  EXPSIGN   epz  $9C      ; GETINT
009D      185  DSCTMP    epz  $9D      ; 9D:9F
009D      186  FACEXP    epz  $9D
009E      187  FACMANT   epz  $9E      ; 9E:A1
0800      188  ;
00A0      189  VARPTR    epz  $A0      ; A0:A1
00A2      190  FACSIGN   epz  $A2
00A3      191  COEFNUM   epz  $A3
00A3      192  MINUSLOC  epz  $A3
00A4      193  EXTSIGN   epz  $A4
00A5      194  ARGEXP    epz  $A5
00A6      195  ARGMANT   epz  $A6      ; A6:A9
00AA      196  ARGSIGN   epz  $AA
00AB      197  XORSIGN   epz  $AB
00AC      198  FACGUARD  epz  $AC
00AB      199  STRING1   epz  $AB      ; AB:AC
00AD      200  STRING2   epz  $AD      ; AD:AE
00AD      201  COEFPTR   epz  $AD      ; AD:AE
00AD      202  SAVY      epz  $AD      ; FPOUT
00AF      203  PRGEND    epz  $AF      ; AF:B0
0800      204  ;
00B1      205  CHRGTADR  epz  $B1
00B8      206  TXTPTR    epz  $B8
0800      207  ;
00C9      208  IRAND      epz  $C9      ; C9:CC 4-byte integer
00CD      209  SIGNFLG   epz  $CD      ; for TAN
00CD      210  SPCLFLAG  epz  $CD      ; for GARBAG
0800      211  ;
00D0      212  HRXDELTA  epz  $D0      ; D0:D1, horizontal delta
00D2      213  HRYDELTA  epz  $D2      ; vertical delta
00D3      214  HRFLAG    epz  $D3      ; cursor move uses bits 6 & 7
00D4      215  HRWORK    epz  $D4      ; D4:D5, working delta
0800      216  ;
00D0      217  SHPVAL     epz  $D0
00D1      218  ROTQVAL   epz  $D1
00D2      219  ROTHVAL   epz  $D2
00D3      220  ROTVVAL   epz  $D3
00D4      221  ROTHSUM   epz  $D4
00D5      222  ROTVSUM   epz  $D5
00D7      223  SHPOLD    epz  $D7
0800      224  ;
00D6      225  RUNFLAG   epz  $D6
00D8      226  ERRFLG    epz  $D8
00DA      227  ERRLIN    epz  $DA      ; DA:DB
00DC      228  ERRPOS    epz  $DC      ; DC:DD
00DE      229  ERRNUM    epz  $DE
00DF      230  ERRSTK    epz  $DF
0800      231  ;
00E0      232  HRXCOOR   epz  $E0      ; E0:E1
00E2      233  HRYCOOR   epz  $E2
00E4      234  HRCOLOR   epz  $E4
00E5      235  HRHORZ    epz  $E5
00E6      236  HRPAG     epz  $E6
00E7      237  HRSCALE   epz  $E7
00E8      238  HRSHTBL   epz  $E8      ; E8:E9
00EA      239  HRCOLCNT  epz  $EA
0800      240  ;
00F0      241  FIRST      epz  $F0
00F1      242  SPEEDBYT  epz  $F1
00F2      243  TRACEFLG  epz  $F2

```

```

00F3      244  FLASHBYT epz $F3
00F4      245  TXTPTRSV epz $F4          ; F4:F5
00F6      246  CURLINSV epz $F6          ; F6:F7
00F8      247  REMSTK   epz $F8
00F9      248  HRROT    epz $F9
0800      249  ;
0800      250          enz
0800      251  ;
0001      252  HLINMOD   equ 1            ; HLIN routine modification
0001      253  FPCDMOD   equ 1            ; floating point routines mod
0800      254  ;
0000      255  ZERO      equ $00
0002      256  IVARLEN   equ $02
0003      257  AVARLEN   equ $03
0003      258  MAXPDL    equ $03
0004      259  MANTLEN   equ $04
0005      260  AHDRLN    equ $05
0005      261  FACLEN    equ $05
0006      262  FACSIZE   equ FACLEN+1      ; FACLEN + FACGUARD
0007      263  SVARLEN   equ $07
0007      264  PXLSBYTE  equ $07
0008      265  BYTEBITS  equ $08
000B      266  DFLTDIM   equ $0B
0014      267  FRAMSIZE  equ 2+2+FACSIZE+1+FACSIZE+2+1 ; FOR/NEXT frame
0020      268  MANTBITS  equ MANTLEN*BYTEBITS
0800      269  ;
0003      270  MOVEMASK  equ $03
0004      271  PLOTMASK  equ $04
0007      272  SCMDMASK  equ $07
000F      273  SROTMASK  equ $0F
0800      274  ;
003F      275  INVERSE   equ $3F
007F      276  FLASH     equ $7F
007F      277  INVRSE80  equ $7F
007F      278  MSBCLR    equ $7F
0080      279  MSBSET    equ $80
0080      280  EXPBIAS   equ $80
00DF      281  LWRMASK   equ $DF
00EF      282  MAXINPUT  equ $EF          ; max APPLESOFT input
00FE      283  NEG TWO   equ $FE
00FF      284  NEG ONE   equ $FF
0800      285  ;
0082      286  CTRLB     equ $82
0083      287  CTRLC     equ $83
0085      288  CTRL E    equ $85
0087      289  ASCIBELL  equ $87
0088      290  CTRLH     equ $88
0088      291  LARROW    equ $88
008A      292  DARROW    equ $8A
008A      293  CTRLJ     equ $8A
008B      294  CTRLK     equ $8B
008B      295  UARROW    equ $8B
008D      296  RETURN    equ $8D
0090      297  CTRLP     equ $90
0093      298  CTRLS     equ $93
0095      299  CTRLU     equ $95
0095      300  RARROW    equ $95
0098      301  CTRLX     equ $98
0099      302  CTRL Y    equ $99
009B      303  ESCAPE    equ $9B
00A0      304  SPACE     equ $A0

```

```

0800      305      ;
0006      306      GOODF8      equ  $06                ; 0xF8 ROM version
0800      307      ;
00A5      308      PWRUPBYT    equ  $A5
0800      309      ;
00FE      310      ERROR.1     equ  $FE                ; INPUT error
00FF      311      ERROR.2     equ  $FF                ; ctrl-C input error
0800      312      ;
0800      313      ;
0800      314      ; Tokens values that are referenced directly.
0800      315      ;
00C0      316      TK.TAB      equ  $C0
00C1      317      TK.TO      equ  $C1
00C2      318      TK.FN      equ  $C2
00C3      319      TK.SPC     equ  $C3
00C4      320      TK.THEN    equ  $C4
00C5      321      TK.AT      equ  $C5
00C6      322      TK.NOT     equ  $C6
00C7      323      TK.STEP    equ  $C7
00C8      324      TK.PLUS    equ  $C8
00C9      325      TK.MINUS   equ  $C9
00CF      326      TK.GRTR    equ  $CF
00D0      327      TK.EQUAL   equ  $D0
00D2      328      TK.SGN     equ  $D2
00D7      329      TK.SCRN    equ  $D7
0800      330      ;
0800      331      ;
0800      332      ; Operator Tag Values (OTV) used directly.
0800      333      ;
0046      334      OTV.OR      equ  $46                ; OR is lowest precedence
0050      335      OTV.AND     equ  $50                ; AND is next precedence
0064      336      OTV.REL     equ  $64                ; relational operators
0079      337      OTV.ADD     equ  $79                ; binary + and - operators
007B      338      OTV.MUL     equ  $7B                ; multiply & divide operators
007D      339      OTV.PWR     equ  $7D                ; exponentiation operator
007F      340      OTV.NEQ     equ  $7F                ; unary (-) and comparison (=)
0800      341      ;
0800      342      ;
0100      343      PAGESIZE    equ  $100
0100      344      STACK      equ  $100
0800      345      ;
0200      346      INPUT      equ  $200
0800      347      ;
03ED      348      XFERADR     equ  $3ED                ; XFER destination address
0800      349      ;
03EF      350      AUTOBRK     equ  $3EF                ; 3EF:3F1
03F2      351      AUTORSET    equ  $3F2                ; 3F2:3F3
03F4      352      PWRSTATE    equ  $3F4
03F5      353      USRAHAND    equ  $3F5                ; 3F5:3F7
03F8      354      USRYHAND    equ  $3F8                ; 3F8:3FA
03FB      355      NMASKIRQ     equ  $3FB                ; 3FB:3FD
03FE      356      MASKIRQ     equ  $3FE                ; 3FE:3FF
0800      357      ;
0400      358      TEXTPG1     equ  $400
05B0      359      PG1TXLOC    equ  $5B0
0800      360      ;
047B      361      OLDCH      equ  $47B                ; last CH used by video F/W
04FB      362      XMODE      equ  $4FB                ; video firmware mode
0800      363      ;
057B      364      OURCH      equ  $57B                ; 80 column CH
05FB      365      OURCV      equ  $5FB                ; 80 column CV

```

```

0800      366      ;
067B      367      XCHAR      equ  $67B      ; character to be printed/read
06FB      368      XCOORD      equ  $6FB      ; GOTOXY X-coord (Pascal only)
0800      369      ;
077B      370      XTEMP1      equ  $77B      ; temp
077B      371      OLDBASL      equ  $77B      ; last BASL (Pascal only)
07FB      372      XTEMP2      equ  $7FB      ; temp2
07FB      373      OLDBASH      equ  $7FB      ; last BASH (Pascal only)
07F8      374      MSLOT      equ  $7F8
0800      375      ;
0800      376      PAGE08      equ  $0800
0C00      377      PAGE0C      equ  $0C00
1000      378      PAGE10      equ  $1000
2000      379      PAGE20      equ  $2000
4000      380      PAGE40      equ  $4000
BD00      381      PAGEBD      equ  $BD00
BF00      382      PAGEBF      equ  $BF00
C000      383      PAGEC0      equ  $C000
C100      384      PAGEC1      equ  $C100
C800      385      PAGEC8      equ  $C800
D000      386      PAGEDO      equ  $D000
F000      387      PAGEF0      equ  $F000
FE00      388      PAGEFE      equ  $FE00
0800      389      ;
C000      390      KEY      equ  $C000
C000      391      STR80OFF      equ  $C000
C001      392      STR80ON      equ  $C001
C002      393      RAMRDOFF      equ  $C002
C003      394      RAMRDON      equ  $C003
C004      395      RAMWROFF      equ  $C004
C005      396      RAMWRON      equ  $C005
C006      397      CXROMOFF      equ  $C006
C007      398      CXROMON      equ  $C007
C008      399      AUXZPOFF      equ  $C008
C009      400      AUXZPON      equ  $C009
C00A      401      C3ROMOFF      equ  $C00A
C00B      402      C3ROMON      equ  $C00B
C00C      403      VID80OFF      equ  $C00C
C00D      404      VID80ON      equ  $C00D
C00E      405      ALTCHOFF      equ  $C00E
C00F      406      ALTCHON      equ  $C00F
0800      407      ;
C010      408      CLRKEY      equ  $C010
C011      409      RDBANK2      equ  $C011
C012      410      RDLGRAM      equ  $C012
C013      411      RDRAMRD      equ  $C013
C014      412      RDRAMWR      equ  $C014
C015      413      RDCXROM      equ  $C015
C016      414      RDAUXZP      equ  $C016
C017      415      RDC3ROM      equ  $C017
C018      416      RDSTR80      equ  $C018
C019      417      RDVRTBLK      equ  $C019
C01A      418      RDTEXT      equ  $C01A
C01B      419      RDMIXED      equ  $C01B
C01C      420      RDPAGE2      equ  $C01C
C01D      421      RDHIRES      equ  $C01D
C01E      422      RDALTCH      equ  $C01E
C01F      423      RDVID80      equ  $C01F
0800      424      ;
C020      425      TAPEOUT      equ  $C020
C030      426      SPKRTOGL      equ  $C030

```



```

0800      427 ;
C050      428 TEXTOFF equ $C050
C051      429 TEXTON equ $C051
C052      430 MIXEDOFF equ $C052
C053      431 MIXEDON equ $C053
C054      432 PAGE1ON equ $C054
C055      433 PAGE2ON equ $C055
C056      434 HIRESOFF equ $C056
C057      435 HIRESON equ $C057
C058      436 ANN1OFF equ $C058
C05A      437 ANN2OFF equ $C05A
C05D      438 ANN3ON equ $C05D
C05F      439 ANN4ON equ $C05F
0800      440 ;
C060      441 TAPEIN equ $C060
C061      442 PB1IN equ $C061
C062      443 PB2IN equ $C062
C064      444 GC1IN equ $C064
0800      445 ;
C070      446 GCTOGL equ $C070
0800      447 ;
C080      448 RAM2WP equ $C080
C081      449 ROM2WE equ $C081
C082      450 ROM2WP equ $C082
C083      451 RAM2WE equ $C083
C088      452 RAM1WP equ $C088
C08B      453 RAM1WE equ $C08B
0800      454 ;
0800      455 ;
0800      456 ; BASIC mode bits
0800      457 ;
0800      458 ; 0..... BASIC active
0800      459 ; 1..... Pascal active
0800      460 ; .0.....
0800      461 ; .1.....
0800      462 ; ..0..... print control characters
0800      463 ; ..1..... do not print ctrl chars
0800      464 ; ...0....
0800      465 ; ...1....
0800      466 ; ....0... print next ctrl char
0800      467 ; ....1... do not print next ctrl char
0800      468 ; .....0..
0800      469 ; .....1..
0800      470 ; .....0.
0800      471 ; .....1.
0800      472 ; .....0 mouse text inactive
0800      473 ; .....1 mouse text active
0800      474 ;
0040      475 M.6 equ %01000000
0020      476 M.CTL2 equ %00100000 ; do not print controls
0010      477 M.4 equ %00010000
0008      478 M.CTL equ %00001000 ; temp ctrl disable
0004      479 M.2 equ %00000100
0002      480 M.1 equ %00000010
0001      481 M.MOUSE equ %00000001
0800      482 ;
0800      483 ;
0800      484 ; Pascal mode bits
0800      485 ;
0800      486 ; 0..... BASIC active
0800      487 ; 1..... Pascal active

```

```
0800      488 ; .0.....
0800      489 ; .1.....
0800      490 ; ..0.....
0800      491 ; ..1.....
0800      492 ; ...0.... cursor always on
0800      493 ; ...1.... cursor always off
0800      494 ; ....0... GOTOXY n/a
0800      495 ; ....1... GOTOXY in progress
0800      496 ; .....0.. normal video
0800      497 ; .....1.. inverse video
0800      498 ; .....0. Pascal 1.1 firmware active
0800      499 ; .....1. Pascal 1.0 interface
0800      500 ; .....0 mouse text inactive
0800      501 ; .....1 mouse text active
0800      502 ;
0080      503 M.PASCAL equ %10000000 ; Pascal active
0010      504 M.CURSOR equ %00010000 ; do not print cursor
0008      505 M.GOXY equ %00001000 ; GOTOXY in progress
0004      506 M.VMODE equ %00000100 ; Pascal video mode
0002      507 M.PAS1.0 equ %00000010 ; Pascal 1.0 mode
0800      508 ;
0800      509 ;
0800      510 ; Definitions
0800      511 ;
0800      512 ; ACA - At Correct Address. Same address as unaltered ROM.
0800      513 ;
0800      514 ;
0800      515      icl "C0.L"
```

```
LLOAD C0.L,A$4000
```

```

0800      1          ttl "ROM Source Code, C0.L"
0800      2      ;
0800      3      ;
0800      4      ; C0.L
0800      5      ;
0800      6      ;
C000      7          org PAGEC0
C000      8          obj PAGE10
C000      9          usr
C000     10      ;
C000     11      ;
C000     12      IOSPACE:
C000     13      ;
C000     14      ;
C000     15          dfs PAGESIZE,ZERO
C100     16      ;
C100     17      ;
C100     18      ; DOCXCMD is called by the patched 0xF8 ROM. It provides
C100     19      ; an extension to the 0xF8 routines that do not work in 80
C100     20      ; columns.
C100     21      ;
C100     22      ; Before jumping here the 0xF8 ROM disables slot I/O and
C100     23      ; enables ROM I/O. This makes the entire space from 0xC100
C100     24      ; to 0xCFFF available except for the 0xC300 page.
C100     25      ;
C100     26      ; On exit, slot I/O is restored if necessary.
C100     27      ;
C100     28      ; The STACK has the PHP for status of the internal 0xCN00
C100     29      ; ROM.
C100     30      ;
C100     31      ; If VID80 is on and the XMODE byte is valid, this call
C100     32      ; will be handled by the 80 column routine. Otherwise, it
C100     33      ; will be handled by the 40 column routine. Return to the
C100     34      ; Autostart ROM is done by CXEXIT.
C100     35      ;
C100     36      ;
C100 4C 11 C2     37      DOCXCMD    jmp XDOCMD
C103     38      ;
C103     39      ;
C103     40      ; CLREOP support.
C103     41      ;
C103 A4 24     42      XCLREOP    ld  CH
C105 A5 25     43                  lda  CV
C107     44      ;
C107 48     45      ^1        pha
C108     46      ;
C108 20 00 CE   47                  jsr  XVTAB3
C10B 20 F3 C1   48                  jsr  XCLREOL1
C10E     49      ;
C10E A0 00     50                  ld  #ZERO
C110     51      ;
C110 68     52                  pla
C111 69 00     53                  adc  #ZERO
C113     54      ;
C113 C5 23     55                  cmp  WNDBTM
C115 90 F0     56                  bcc  <1
C117     57      ;
C117 B0 34     58                  bcs  >5          ; always taken
C119     59      ;
C119     60      ;

```

```

C119      61 ; HOME support.
C119      62 ;
C119 A5 22 63 XHOME      lda WNDTOP
C11B 85 25 64           sta CV
C11D      65 ;
C11D A0 00 66           ldy #ZERO
C11F 84 24 67           sty CH
C121      68 ;
C121 F0 E4 69           beq <1                ; always taken
C123      70 ;
C123      71 ;
C123      72 ; SCROLL support.
C123      73 ;
C123 A5 22 74 XSCROLL    lda WNDTOP
C125 48    75           pha
C126      76 ;
C126 20 00 CE 77           jsr XVTAB3
C129      78 ;
C129 A5 28 79 ^2        lda BASL
C12B 85 2A 80           sta BAS2L
C12D      81 ;
C12D A5 29 82           lda BASH
C12F 85 2B 83           sta BAS2H
C131      84 ;
C131 A4 21 85           ldy WNDWDTH
C133 88    86           dey
C134      87 ;
C134 68    88           pla
C135 69 01 89           adc #1
C137      90 ;
C137 C5 23 91           cmp WNDBTM
C139 B0 0D 92           bcs >4
C13B      93 ;
C13B 48    94           pha
C13C      95 ;
C13C 20 00 CE 96           jsr XVTAB3
C13F      97 ;
C13F B1 28 98 ^3        lda (BASL),Y
C141 91 2A 99           sta (BAS2L),Y
C143     100 ;
C143 88    101           dey
C144 10 F9 102           bpl <3
C146     103 ;
C146 30 E1 104           bmi <2                ; always taken
C148     105 ;
C148 A0 00 106 ^4        ldy #ZERO
C14A     107 ;
C14A 20 F3 C1 108           jsr XCLREOL1
C14D     109 ;
C14D A5 25 110 ^5        lda CV
C14F     111 ;
C14F 4C 00 CE 112 ^6        jmp XVTAB3
C152     113 ;
C152     114 ;
C152     115 ; SETWND support.
C152     116 ;
C152 A9 28 117 XSETWND    lda #40
C154 85 21 118           sta WNDWDTH
C156     119 ;
C156 A9 18 120           lda #24
C158 85 23 121           sta WNDBTM

```

```

C15A          122 ;
C15A A9 17    123         lda #23
C15C 85 25    124         sta CV
C15E D0 EF    125         bne <6                ; always taken
C160          126 ;
C160          127 ;
C160          128 ; CLEOLZ support.
C160          129 ;
C160 A4 2A    130 XCLREOL2 ldy BAS2L
C162          131 ;
C162 4C F3 C1 132         jmp XCLREOL1
C165          133 ;
C165          134 ;
C165          135 ; 80 column routines begin here.
C165          136 ;
C165          137 ; 80 column SCROLL support.
C165          138 ;
C165 4C EB CB 139 YSCROLL jmp SCROLLUP
C168          140 ;
C168          141 ;
C168          142 ; 80 column CLREOL support.
C168          143 ;
C168 4C 97 CC 144 YCLREOL jmp XGS
C16B          145 ;
C16B          146 ;
C16B          147 ; 80 column CLEOLZ support.
C16B          148 ;
C16B A4 2A    149 YCLREOL2 ldy BAS2L
C16D          150 ;
C16D 4C 9A CC 151         jmp XGSEOLZ
C170          152 ;
C170          153 ;
C170          154 ; 80 column CLREOP support.
C170          155 ;
C170 4C 77 CC 156 YCLREOP jmp XVT
C173          157 ;
C173          158 ;
C173          159 ; 80 column SETWND support.
C173          160 ;
C173 4C A5 C2 161 YSETWND jmp XSETWNDX
C176          162 ;
C176          163 ;
C176          164 ; RESET support.
C176          165 ;
C176          166 XRESET:
C176 4C B5 C2 167 YRESET jmp XRESETX
C179          168 ;
C179          169 ;
C179          170 ; 80 column RDKEY support.
C179          171 ;
C179 4C 11 CE 172 YRDKEY jmp XRDKEY2
C17C          173 ;
C17C          174 ;
C17C          175 ; 80 column HOME support.
C17C          176 ;
C17C 20 74 CC 177 YHOME jsr XFF
C17F          178 ;
C17F AD 7B 05 179         lda OURCH
C182 85 24    180         sta CH
C184 8D 7B 04 181         sta OLDCH
C187          182 ;

```

```

C187 4C FB CD      183          jmp XVTAB2
C18A              184      ;
C18A              185      ;
C18A              186      ; 80 column IOPORT support.
C18A              187      ;
C18A B4 00        188 YIOPORT ldy LOC0,X
C18C F0 0F        189          beq >2
C18E              190      ;
C18E C0 1B        191          cpy #KEYIN
C190 F0 0E        192          beq ISO
C192              193      ;
C192 20 7D CD     194          jsr QUIT
C195              195      ;
C195              196      ;
C195              197      ; IOPORT support.
C195              198      ;
C195 B4 00        199 XIOPORT ldy LOC0,X
C197 F0 04        200          beq >2
C199              201      ;
C199 A9 FD        202 ^1      lda /KEYIN
C19B 95 01        203          sta LOC1,X
C19D              204      ;
C19D B5 01        205 ^2      lda LOC1,X
C19F              206      ;
C19F 60           207          rts
C1A0              208      ;
C1A0              209      ;
C1A0 A5 37        210 ISO      lda CSWH
C1A2 C9 C3        211          cmp /C3SPACE
C1A4 D0 F3        212          bne <1
C1A6              213      ;
C1A6 4C 32 C8     214          jmp C3IN
C1A9              215      ;
C1A9              216      ;
C1A9              217      ; RDKEY support.
C1A9              218      ;
C1A9 A4 24        219 XRDKEY ldy CH
C1AB              220      ;
C1AB B1 28        221          lda (BASL),Y
C1AD 48           222          pha
C1AE              223      ;
C1AE 29 3F        224          and #INVERSE          ; set screen to inverse
C1B0 09 40        225          ora #$40
C1B2 91 28        226          sta (BASL),Y
C1B4              227      ;
C1B4 68           228          pla
C1B5              229      ;
C1B5 60           230          rts
C1B6              231      ;
C1B6              232      ;
C1B6              233      ; Implemented BASCALC support that restores Y-reg.
C1B6              234      ;
C1B6 A4 28        235 XBASCLC ldy BASL
C1B8              236      ;
C1B8 20 BA CA     237          jsr XBASCALC
C1BB 90 4B        238          bcc CXEXIT          ; always taken
C1BD              239      ;
C1BD              240      ;
C1BD              241      ; RDESC support.
C1BD              242      ;
C1BD              243      ; Map ^H to J, ^U to K, ^J to M, and ^K to I.

```

```

C1BD      244 ;
C1BD 20 01 FD 245 XRDESC      jsr UPRCASE
C1C0      246 ;
C1C0 A0 03   247             ldy #KBDOUT-KBDTBL+1
C1C2      248 ;
C1C2 D9 EF C2 249 ^1        cmp KBDTBL,Y
C1C5 D0 03   250             bne >2
C1C7      251 ;
C1C7 B9 F3 C2 252             lda KBDOUT,Y
C1CA      253 ;
C1CA 88      254 ^2        dey
C1CB 10 F5   255             bpl <1
C1CD      256 ;
C1CD 30 39   257             bmi CXEXIT          ; always taken
C1CF      258 ;
C1CF      259 ;
C1CF      260 ; NEWVW VIDWAIT support.
C1CF      261 ;
C1CF 20 70 C8 262 XNEWVW      jsr CXNEWVW
C1D2      263 ;
C1D2 4C 08 C2 264             jmp CXEXIT
C1D5      265 ;
C1D5      266 ;
C1D5      267 ; GETFMT support.
C1D5      268 ;
C1D5      269 XGETFMT:
C1D5 8A      270 YGETFMT      txa
C1D6 85 2E   271             sta FORMAT
C1D8      272 ;
C1D8 29 03   273             and #3
C1DA 85 2F   274             sta LENGTH
C1DC      275 ;
C1DC A5 2A   276             lda BAS2L          ; recall opcode index
C1DE      277 ;
C1DE 4C D5 C5 278             jmp XGETFMT2
C1E1      279 ;
C1E1      280 ;
C1E1      281 ; MINIASM support.
C1E1      282 ;
C1E1      283 XGOMINI:
C1E1 20 F0 FC 284 YGOMINI      jsr NXTLINE
C1E4      285 ;
C1E4 8A      286             txa
C1E5 85 34   287             sta YSAV
C1E7      288 ;
C1E7 60      289             rts
C1E8      290 ;
C1E8      291 ;
C1E8      292 ; PICKFIX support.
C1E8      293 ;
C1E8      294 XPICKFIX:
C1E8 AC 7B 05 295 YPICKFIX      ldy OURCH
C1EB 20 44 CE 296             jsr PICK
C1EE      297 ;
C1EE 09 80   298             ora #MSBSET
C1F0      299 ;
C1F0 60      300             rts
C1F1      301 ;
C1F1      302 ;
C1F1      303 ; CLREOL support.
C1F1      304 ;

```

```

C1F1 A4 24      305 XCLREOL ldy CH
C1F3           306 ;
C1F3 A9 A0      307 XCLREOL1 lda #SPACE
C1F5           308 ;
C1F5 2C 1E C0   309          bit RDALTCH
C1F8 10 06      310          bpl >1
C1FA           311 ;
C1FA 24 32      312          bit INVFLG
C1FC 30 02      313          bmi >1
C1FE           314 ;
C1FE A9 20      315          lda #' '          ; use inverse blank
C200           316 ;
C200 4C A5 CC   317 ^1          jmp CLR40
C203           318 ;
C203           319 ;
C203           320 ; VTAB support. Modified to save Y-reg in BASL. Fall
C203           321 ; into CXEXIT.
C203           322 ;
C203 A4 28      323 XVTAB      ldy BASL
C205 20 00 CE   324          jsr XVTAB3
C208           325 ;
C208           326 ;
C208           327 ; If PLP is +, turn CX ROM off, otherwise leave CX ROM on.
C208           328 ;
C208 28         329 CXEXIT    plp
C209 30 03      330          bmi >1
C20B           331 ;
C20B 4C 3C FA   332          jmp CXOFF
C20E           333 ;
C20E 4C 3F FA   334 ^1          jmp CXRTN
C211           335 ;
C211           336 ;
C211 88         337 XDOCMD    dey
C212 30 BB      338          bmi XNEWVW          ; Y-reg = 0
C214           339 ;
C214 88         340          dey
C215 30 A6      341          bmi XRDESC          ; Y-reg = 1
C217           342 ;
C217 88         343          dey
C218 30 9C      344          bmi XBASCLC          ; Y-reg = 2
C21A           345 ;
C21A 88         346          dey
C21B 30 3E      347          bmi XKEYIN          ; Y-reg = 3
C21D           348 ;
C21D 88         349          dey
C21E 30 E3      350          bmi XVTAB          ; Y-reg = 4
C220           351 ;
C220 A9 C2      352          lda /CXEXIT-1
C222 48         353          pha
C223           354 ;
C223 A9 07      355          lda #CXEXIT-1
C225 48         356          pha
C226           357 ;
C226 AD FB 04   358          lda XMODE
C229 29 D6      359          and #M.PASCAL|M.6|M.4|M.2|M.1
C22B D0 0E      360          bne >1
C22D           361 ;
C22D 98         362          tya
C22E           363 ;
C22E D8         364          cld          ; added to clear DECIMAL
C22F 18         365          clc

```



```

C230          366 ;
C230 69 0C    367      adc #YREGTBLY-YREGTBLX
C232 48      368      pha
C233          369 ;
C233 20 50 C8 370      jsr CSETUP
C236 20 FB CD 371      jsr XVTAB2
C239          372 ;
C239 68      373      pla
C23A A8      374      tay
C23B          375 ;
C23B A9 C1   376 ^1     lda /PAGEC1
C23D 48      377      pha
C23E          378 ;
C23E B9 43 C2 379      lda YREGTBLX,Y
C241 48      380      pha
C242          381 ;
C242 60      382      rts
C243          383 ;
C243          384 ;
C243          385 ; PAGEC1 routines.
C243          386 ;
C243 18      387 YREGTBLX byt XHOME-1      ; Y-reg = 5
C244 22      388      byt XSCROLL-1      ; Y-reg = 6
C245 F0      389      byt XCLREOL-1      ; Y-reg = 7
C246 5F      390      byt XCLREOL2-1     ; Y-reg = 8
C247 75      391      byt XRESET-1      ; Y-reg = 9
C248 02      392      byt XCLREOP-1     ; Y-reg = 10
C249 A8      393      byt XRDKEY-1      ; Y-reg = 11
C24A 51      394      byt XSETWND-1     ; Y-reg = 12
C24B E0      395      byt XGOMINI-1     ; Y-reg = 13
C24C 94      396      byt XIOPORT-1     ; Y-reg = 14
C24D E7      397      byt XPICKFIX-1    ; Y-reg = 15
C24E D4      398      byt XGETFMT-1     ; Y-reg = 16
C24F          399 ;
C24F 7B      400 YREGTBLY byt YHOME-1      ; Y-reg = 5
C250 64      401      byt YSCROLL-1      ; Y-reg = 6
C251 67      402      byt YCLREOL-1      ; Y-reg = 7
C252 6A      403      byt YCLREOL2-1     ; Y-reg = 8
C253 75      404      byt YRESET-1      ; Y-reg = 9
C254 6F      405      byt YCLREOP-1     ; Y-reg = 10
C255 78      406      byt YRDKEY-1      ; Y-reg = 11
C256 72      407      byt YSETWND-1     ; Y-reg = 12
C257 E0      408      byt YGOMINI-1     ; Y-reg = 13
C258 89      409      byt YIOPORT-1     ; Y-reg = 14
C259 E7      410      byt YPICKFIX-1     ; Y-reg = 15
C25A D4      411      byt YGETFMT-1     ; Y-reg = 16
C25B          412 ;
C25B          413 ;
C25B          414 ; KEYIN support.
C25B          415 ;
C25B 2C 1F C0 416 XKEYIN  bit RDVID80
C25E 10 06    417      bpl >2
C260          418 ;
C260 20 74 C8 419      jsr CXNEWVW2
C263          420 ;
C263 4C 08 C2 421 ^1     jmp CXEXIT
C266          422 ;
C266 A8      423 ^2     tay
C267          424 ;
C267 8A      425      txa
C268 48      426      pha

```

```

C269          427 ;
C269 98        428      tya
C26A 48        429      pha
C26B          430 ;
C26B 48        431      pha
C26C          432 ;
C26C 68        433 ^3      pla
C26D C9 FF     434      cmp #NEGONE
C26F F0 04     435      beq >4
C271          436 ;
C271 A9 FF     437      lda #NEGONE
C273 D0 02     438      bne >5          ; always taken
C275          439 ;
C275 68        440 ^4      pla
C276 48        441      pha
C277          442 ;
C277 48        443 ^5      pha
C278          444 ;
C278 A4 24     445      ldy CH
C27A          446 ;
C27A 91 28     447      sta (BASL),Y
C27C          448 ;
C27C E6 4E     449 ^6      inc RNDL
C27E D0 0A     450      bne >7
C280          451 ;
C280 A5 4F     452      lda RNDH          ; time to blink?
C282          453 ;
C282 E6 4F     454      inc RNDH
C284          455 ;
C284 45 4F     456      eor RNDH
C286 29 40     457      and #$40
C288 D0 E2     458      bne <3          ; yes, blink it
C28A          459 ;
C28A AD 00 C0  460 ^7      lda KEY
C28D 10 ED     461      bpl <6
C28F          462 ;
C28F 68        463      pla
C290          464 ;
C290 A4 24     465      ldy CH
C292          466 ;
C292 68        467      pla
C293 91 28     468      sta (BASL),Y
C295          469 ;
C295 68        470      pla
C296 AA        471      tax
C297          472 ;
C297 AD 00 C0  473      lda KEY
C29A          474 ;
C29A 2C 10 C0  475      bit CLRKEY
C29D          476 ;
C29D C9 FF     477      cmp #NEGONE
C29F D0 C2     478      bne <1
C2A1          479 ;
C2A1 A9 88     480      lda #LARROW
C2A3 D0 BE     481      bne <1          ; always taken
C2A5          482 ;
C2A5          483 ;
C2A5 20 52 C1  484 XSETWNDX jsr XSETWND
C2A8          485 ;
C2A8 2C 1F C0  486      bit RDVID80
C2AB 10 02     487      bpl >1

```

```

C2AD          488 ;
C2AD 06 21    489      asl WNDWDTH
C2AF          490 ;
C2AF A5 25    491 ^1      lda CV
C2B1 8D FB 05 492      sta OURCV
C2B4          493 ;
C2B4 60       494      rts
C2B5          495 ;
C2B5          496 ;
C2B5          497 ; RESET support.
C2B5          498 ;
C2B5 D8       499 XRESETX cld
C2B6          500 ;
C2B6          501 ;      jsr RSETINIT
C2B6          502 ;
C2B6 20 84 FE 503      jsr SETNORM
C2B9 20 2F FB 504      jsr INIT
C2BC 20 93 FE 505      jsr SETVID
C2BF 20 89 FE 506      jsr SETKBD
C2C2          507 ;
C2C2 A9 FF    508      lda #NEGONE
C2C4 8D FB 04 509      sta XMODE
C2C7          510 ;
C2C7          511 ;
C2C7          512 ; If the solid Apple key is pressed go to DIAGS.
C2C7          513 ; DIAGS have been removed.
C2C7          514 ;
C2C7          515 ;      lda PB2IN
C2C7          516 ;      bpl >1
C2C7          517 ;
C2C7          518 ;      jmp DIAGS
C2C7          519 ;
C2C7          520 ;
C2C7          521 ; If the open Apple key is pressed destroy memory and
C2C7          522 ; coldstart the system.
C2C7          523 ;
C2C7 AD 61 C0 524 ^1      lda PB1IN
C2CA 10 11    525      bpl CXRESET
C2CC          526 ;
C2CC          527 ;
C2CC          528 ; Write two bytes of SPACE on each page from 0xBD to 0x01.
C2CC          529 ; Avoid the DOS 4.5.08 interface pages 0xBF and 0xBE.
C2CC          530 ;
C2CC A9 00    531      lda #PAGEBD
C2CE 85 3C    532      sta A1L
C2D0          533 ;
C2D0 A0 BD    534      ldy /PAGEBD
C2D2          535 ;
C2D2 A9 A0    536      lda #SPACE
C2D4          537 ;
C2D4 84 3D    538 ^2      sty A1H
C2D6          539 ;
C2D6 91 3C    540      sta (A1L),Y
C2D8          541 ;
C2D8 88       542      dey
C2D9          543 ;
C2D9 91 3C    544      sta (A1L),Y
C2DB          545 ;
C2DB D0 F7    546      bne <2
C2DD          547 ;
C2DD          548 ;

```

```

C2DD      549 ; If there is an interface card in slot 3, do not enable
C2DD      550 ; the internal CX3 ROM space.  If there is no card in slot
C2DD      551 ; 3 and there is a RAM interface card in the video slot,
C2DD      552 ; switch in the internal CX3 ROM space.
C2DD      553 ;
C2DD      554 ; The //e powers up with the internal CX3 ROM space
C2DD      555 ; switched in.  TSTROMCD switches it out.  CXRESET may or
C2DD      556 ; may not switch it back in.
C2DD      557 ;
C2DD 8D 0B C0 558 CXRESET  sta C3ROMON
C2E0      559 ;
C2E0 20 89 CA 560          jsr TSTROMCD
C2E3 D0 03    561          bne >1
C2E5      562 ;
C2E5 8D 0A C0 563          sta C3ROMOFF
C2E8      564 ;
C2E8 AD FF CF 565 ^1      lda CLRROM          ; disable extension ROM
C2EB      566 ;
C2EB 2C 10 C0 567          bit CLRKEY
C2EE      568 ;
C2EE 60       569          rts
C2EF      570 ;
C2EF      571 ;
C2EF 88 95 8A 572 KBDTBL   byt CTRLH,CTRLU,CTRLJ,CTRLK
C2F2 8B       573 ;
C2F3 CA CB CD 574 KBDOUT   asc "JKMI"
C2F6 C9       575 ;
C2F7      576 ;
C2F7      577          dfs 9,ZERO          ; 9 bytes
C300      578 ;
C300      579 ;
C300      580 C3SPACE:
C300      581 ;
C300      582 ; This page must not be used by any F8 ROM routines.  When
C300      583 ; enabled it claims the C800 space.  Thus peripheral cards
C300      584 ; cannot use AUXMOVE or XFER from their C800 space.
C300      585 ;
C300 2C 2E CA 586 BASICINT bit BITBYT60
C303 70 12    587          bvs BASICENT          ; always taken
C305      588 ;
C305 38       589 BASICIN   sec
C306      590 ;
C306 90 00    591          bcc *+2
C308      592          dfs !-1
C307      593 ;
C307 18       594 BASICOUT clc
C308      595 ;
C308 B8       596          clv
C309 50 0C    597          bvc BASICENT          ; always taken
C30B      598 ;
C30B      599 ;
C30B 01       600          hex 01          ; generic signature byte
C30C 88       601          hex 88          ; device signature byte
C30D      602 ;
C30D 4A       603          byt JPINIT          ; Pascal INIT
C30E 50       604          byt JPREAD          ; Pascal READ
C30F 56       605          byt JPWRITE          ; Pascal WRITE
C310 5C       606          byt JPSTAT          ; Pascal STATUS
C311      607 ;

```

```

C311      608 ;
C311      609 ; 128K support routines.
C311      610 ;
C311 4C 76 C3 611 AUXMOVE jmp DOMOVE ; memory move across banks
C314      612 ;
C314 4C C3 C3 613 XFER jmp DOXFER ; transfer across banks
C317      614 ;
C317      615 ;
C317 8D 7B 06 616 BASICENT sta XCHAR
C31A      617 ;
C31A 98      618 tya
C31B 48      619 pha
C31C      620 ;
C31C 8A      621 txa
C31D 48      622 pha
C31E      623 ;
C31E 08      624 php
C31F      625 ;
C31F AD FB 04 626 lda XMODE
C322      627 ;
C322 2C F8 07 628 bit MSLOT
C325 30 05    629 bmi >2
C327      630 ;
C327 09 08    631 ora #M.CTL
C329 8D FB 04 632 sta XMODE
C32C      633 ;
C32C 20 6D C3 634 ^2 jsr SETC8
C32F      635 ;
C32F 28      636 plp
C330 70 15    637 bvs JBASINIT
C332      638 ;
C332 90 10    639 bcc JC8
C334      640 ;
C334 AA      641 tax
C335 10 0D    642 bpl JC8
C337      643 ;
C337 20 58 CD 644 jsr SETKEYIN
C33A      645 ;
C33A 68      646 pla
C33B AA      647 tax
C33C      648 ;
C33C 68      649 pla
C33D A8      650 tay
C33E      651 ;
C33E AD 7B 06 652 lda XCHAR
C341      653 ;
C341 6C 38 00 654 jmp (KSWL)
C344      655 ;
C344      656 ;
C344 4C 7C C8 657 JC8 jmp CXVIDCK3
C347      658 ;
C347      659 ;
C347 4C 03 C8 660 JBASINIT jmp BASCINIT
C34A      661 ;
C34A      662 ;
C34A 20 6D C3 663 JPINIT jsr SETC8
C34D      664 ;
C34D 4C B4 C9 665 jmp PPINIT
C350      666 ;
C350      667 ;
C350 20 6D C3 668 JPREAD jsr SETC8

```

```

C353      669 ;
C353 4C D6 C9 670      jmp PPREAD
C356      671 ;
C356      672 ;
C356 20 6D C3 673 JPWRITE jsr SETC8
C359      674 ;
C359 4C F0 C9 675      jmp PPWRITE
C35C      676 ;
C35C      677 ;
C35C AA      678 JPSTAT tax
C35D F0 08    679      beq >1
C35F      680 ;
C35F CA      681      dex
C360 D0 07    682      bne >2
C362      683 ;
C362 2C 00 C0 684      bit KEY
C365 10 04    685      bpl >3
C367      686 ;
C367 38      687 ^1      sec
C368      688 ;
C368 60      689      rts
C369      690 ;
C369 A2 03    691 ^2      ldx #3          ; error flag
C36B      692 ;
C36B 18      693 ^3      clc
C36C      694 ;
C36C 60      695      rts
C36D      696 ;
C36D      697 ;
C36D A2 C3    698 SETC8      ldx /C3SPACE
C36F 8E F8 07 699      stx MSLOT
C372      700 ;
C372 AE FF CF 701      ldx CLRROM
C375      702 ;
C375 60      703      rts
C376      704 ;
C376      705 ;
C376      706 ; AUXMOVE routine; crossbank memory move.
C376      707 ;
C376      708 ; A1L/H = source start address
C376      709 ; A2L/H = source end address
C376      710 ; A4L/H = destination start address
C376      711 ;
C376      712 ; carry set = main -> aux
C376      713 ; carry clear = aux -> main
C376      714 ;
C376 48      715 DOMOVE pha
C377      716 ;
C377 98      717      tya
C378 48      718      pha
C379      719 ;
C379 AD 13 C0 720      lda RDRAMRD
C37C 48      721      pha
C37D      722 ;
C37D AD 14 C0 723      lda RDRAMWR
C380 48      724      pha
C381      725 ;
C381 90 08    726      bcc >1
C383      727 ;
C383 8D 02 C0 728      sta RAMRDOFF
C386 8D 05 C0 729      sta RAMWRON

```

```

C389          730 ;
C389 B0 06    731      bcs >2          ; always taken
C38B          732 ;
C38B 8D 04 C0 733 ^1      sta RAMWROFF
C38E 8D 03 C0 734      sta RAMRDON
C391          735 ;
C391 A0 00    736 ^2      ldy #ZERO
C393          737 ;
C393 B1 3C    738 ^3      lda (A1L),Y
C395 91 42    739      sta (A4L),Y
C397          740 ;
C397 E6 42    741      inc A4L
C399 D0 02    742      bne >4
C39B          743 ;
C39B E6 43    744      inc A4H
C39D          745 ;
C39D A5 3C    746 ^4      lda A1L
C39F C5 3E    747      cmp A2L
C3A1          748 ;
C3A1 A5 3D    749      lda A1H
C3A3 E5 3F    750      sbc A2H
C3A5          751 ;
C3A5 E6 3C    752      inc A1L
C3A7 D0 02    753      bne >5
C3A9          754 ;
C3A9 E6 3D    755      inc A1H
C3AB          756 ;
C3AB 90 E6    757 ^5      bcc <3
C3AD          758 ;
C3AD 8D 04 C0 759      sta RAMWROFF
C3B0          760 ;
C3B0 68       761      pla
C3B1 10 03    762      bpl >6
C3B3          763 ;
C3B3 8D 05 C0 764      sta RAMWRON
C3B6          765 ;
C3B6 8D 02 C0 766 ^6      sta RAMRDOFF
C3B9          767 ;
C3B9 68       768      pla
C3BA 10 03    769      bpl >7
C3BC          770 ;
C3BC 8D 03 C0 771      sta RAMRDON
C3BF          772 ;
C3BF 68       773 ^7      pla
C3C0 A8       774      tay
C3C1          775 ;
C3C1 68       776      pla
C3C2          777 ;
C3C2 60       778      rts
C3C3          779 ;
C3C3          780 ;
C3C3          781 ; XFER routine.
C3C3          782 ;
C3C3          783 ; 0x3ED/0x3EE = destination address
C3C3          784 ;
C3C3          785 ; carry set = transfer to aux
C3C3          786 ; carry clear = transfer to main
C3C3          787 ;
C3C3          788 ; vflag set = use aux ZP/STACK
C3C3          789 ; vflag clear = use main ZP/STACK
C3C3          790 ;

```

```

C3C3 48          791 DOXFER pha
C3C4          792 ;
C3C4 AD ED 03   793          lda XFERADR
C3C7 48          794          pha
C3C8          795 ;
C3C8 AD EE 03   796          lda XFERADR+1
C3CB 48          797          pha
C3CC          798 ;
C3CC 90 08      799          bcc >1
C3CE          800 ;
C3CE 8D 03 C0   801          sta RAMRDON
C3D1 8D 05 C0   802          sta RAMWRON
C3D4          803 ;
C3D4 B0 06      804          bcs >2                ; always taken
C3D6          805 ;
C3D6 8D 02 C0   806 ^1          sta RAMRDOFF
C3D9 8D 04 C0   807          sta RAMWROFF
C3DC          808 ;
C3DC 68          809 ^2          pla
C3DD 8D EE 03   810          sta XFERADR+1
C3E0          811 ;
C3E0 68          812          pla
C3E1 8D ED 03   813          sta XFERADR
C3E4          814 ;
C3E4 68          815          pla
C3E5          816 ;
C3E5 70 05      817          bvs >3
C3E7          818 ;
C3E7 8D 08 C0   819          sta AUXZPOFF
C3EA          820 ;
C3EA 50 03      821          bvc >4                ; always taken
C3EC          822 ;
C3EC 8D 09 C0   823 ^3          sta AUXZPON
C3EF          824 ;
C3EF 6C ED 03   825 ^4          jmp (XFERADR)
C3F2          826 ;
C3F2          827 ;
C3F2          828          dfs 2,ZERO                ; 2 bytes
C3F4          829 ;
C3F4          830 ;
C3F4          831 ; This is where the interrupt routine returns to. The ROM
C3F4          832 ; is not necessarily enabled.
C3F4          833 ;
C3F4 8D 81 C0   834 IRQDONE sta ROM2WE
C3F7          835 ;
C3F7 4C 7A FC   836          jmp IRQDONE2
C3FA          837 ;
C3FA          838 ;
C3FA          839 ; This is the main entry point for the interrupt handler.
C3FA          840 ; It enables the internal ROM and simply falls into the
C3FA          841 ; main part of the interrupt handler at 0xC400.
C3FA          842 ;
C3FA 2C 15 C0   843 IRQRTN bit RDCXROM
C3FD          844 ;
C3FD 8D 07 C0   845          sta CXROMON
C400          846 ;
C400          847 ;
C400          848          icl "C4.L"

```

```
LLOAD C4.L,A$4000
```



```

C400      1          ttl "ROM Source Code, C4.L"
C400      2      ;
C400      3      ;
C400      4      ; C4.L
C400      5      ;
C400      6      ;
C400      7      ; Interrupt Handler
C400      8      ;
C400      9      ; The bits of a system status byte is created and saved
C400     10      ; to 0x44. If a bit is on, that function is turned off.
C400     11      ; LC RAM is turned off.
C400     12      ;
C400     13      ; Bit 7      Bit 6      Bit 5      Bit 4
C400     14      ; -----
C400     15      ; RDAUXZP RDPAGE2 RDRAMRD RDRAMWR
C400     16      ; 0=main  0=off   0=off   0=off
C400     17      ; 1=AUX   1=on    1=on    1=on
C400     18      ;
C400     19      ; Bit 3      Bit 2      Bit 1      Bit 0
C400     20      ; -----
C400     21      ; LCRAM   BANK2    BANK1    RDCXROM
C400     22      ; 0=ROM   0=off   0=off   0=off
C400     23      ; 1=RAM   1=on    1=on    1=on
C400     24      ;
C400     25      ; If Bit 3 off, Bits 1 and 2 are off.
C400     26      ; If Bit 3 on, Bit 1 or Bit 2 is on.
C400     27      ;
C400     28      ;
C400 D8     29          cld
C401     30      ;
C401 38     31          sec                      ; C=1 for internal slot space
C402     32      ;
C402 30 01  33          bmi >1
C404     34      ;
C404 18     35          clc
C405     36      ;
C405 48     37      ^1      pha                      ; save on STACK and not 0x45
C406 48     38          pha
C407 48     39          pha
C408     40      ;
C408 8A     41          txa
C409     42      ;
C409 BA     43          tsx
C40A     44      ;
C40A E8     45          inx                      ; ridiculous
C40B E8     46          inx
C40C E8     47          inx
C40D E8     48          inx
C40E     49      ;
C40E 48     50          pha
C40F     51      ;
C40F 98     52          tya
C410 48     53          pha
C411     54      ;
C411 BD 00 01 55          lda STACK,X          ; get BREAK status
C414 29 10     56          and #$10          ; mask for Z flag
C416 A8     57          tay          ; save for later
C417     58      ;
C417     59      ;
C417     60      ; Determine current machine state.

```

```

C417      61 ;
C417 AD 18 C0      62      lda RDSTR80
C41A 2D 1C C0      63      and RDPAGE2
C41D      64 ;
C41D 29 80      65      and #MSBSET
C41F F0 05      66      beq >2
C421      67 ;
C421 A9 20      68      lda #$20
C423 8D 54 C0      69      sta PAGE1ON
C426      70 ;
C426 2A      71 ^2      rol
C427      72 ;
C427 2C 13 C0      73      bit RDRAMRD
C42A 10 05      74      bpl >3
C42C      75 ;
C42C 8D 02 C0      76      sta RAMRDOFF
C42F      77 ;
C42F 09 20      78      ora #$20
C431      79 ;
C431 2C 14 C0      80 ^3      bit RDRAMWR
C434 10 05      81      bpl >4
C436      82 ;
C436 8D 04 C0      83      sta RAMWROFF
C439      84 ;
C439 09 10      85      ora #$10
C43B      86 ;
C43B 2C 12 C0      87 ^4      bit RDLCRAM
C43E 10 0C      88      bpl >6
C440      89 ;
C440 09 0C      90      ora #$0C
C442      91 ;
C442 2C 11 C0      92      bit RDBANK2
C445 10 02      93      bpl >5
C447      94 ;
C447 49 06      95      eor #6
C449      96 ;
C449 8D 81 C0      97 ^5      sta ROM2WE
C44C      98 ;
C44C 2C 16 C0      99 ^6      bit RDAUXZP
C44F 10 0D      100     bpl >7
C451      101 ;
C451 BA      102     tsx
C452 8E 01 01      103     stx STACK+1      ; save aux STACK pointer
C455      104 ;
C455 AE 00 01      105     ldx STACK      ; restore main STACK pointer
C458 9A      106     txs
C459      107 ;
C459 8D 08 C0      108     sta AUXZPOFF
C45C      109 ;
C45C 09 80      110     ora #MSBSET
C45E      111 ;
C45E 88      112 ^7     dey      ; check for BREAK
C45F 30 0C      113     bmi >8
C461      114 ;
C461 85 44      115     sta MACSTAT      ; save machine state
C463      116 ;
C463 68      117     pla
C464 A8      118     tay
C465      119 ;
C465 68      120     pla
C466 AA      121     tax

```

```

C467          122 ;
C467 68        123      pla
C468 68        124      pla
C469 68        125      pla
C46A          126 ;
C46A 4C 47 FA  127      jmp NEWBREAK
C46D          128 ;
C46D 48        129 ^8    pha                ; save machine state
C46E          130 ;
C46E AD F8 07  131      lda MSLOT
C471 48        132      pha
C472          133 ;
C472 A9 C3     134      lda /IRQDONE        ; save return RTI address
C474 48        135      pha
C475          136 ;
C475 A9 F4     137      lda #IRQDONE
C477 48        138      pha
C478          139 ;
C478 08        140      php
C479          141 ;
C479 4C 74 FC  142      jmp GOTOIRQ        ; enter user's IRQ handler
C47C          143 ;
C47C          144 ;
C47C          145 ; The ROM must be reenabled with a LDA ROM2WE if the
C47C          146 ; language card is write protected, write enabled, or
C47C          147 ; being write enabled. An INC ABS,X is used because
C47C          148 ; some of the switches are Read and some are Write since
C47C          149 ; both the 6502 and 65C02 do 2 reads before the write.
C47C          150 ;
C47C AD 81 C0  151 NEWIRQ  lda ROM2WE
C47F          152 ;
C47F 68        153      pla                ; recall machine state
C480 10 07     154      bpl >1
C482          155 ;
C482 8D 09 C0  156      sta AUXZPON
C485          157 ;
C485 AE 01 01  158      ldx STACK+1        ; recall aux STACK pointer
C488 9A        159      txs
C489          160 ;
C489 A0 06     161 ^1    ldy #SWTBLEN
C48B          162 ;
C48B 10 06     163 ^2    bpl >3
C48D          164 ;
C48D BE C1 C4  165      ldx RAMSWTBL,Y
C490          166 ;
C490 FE 00 C0  167      inc IOSPACE,X
C493          168 ;
C493 88        169 ^3    dey
C494 30 03     170      bmi >4
C496          171 ;
C496 0A        172      asl
C497 D0 F2     173      bne <2
C499          174 ;
C499 0A        175 ^4    asl                ; if internal slot space
C49A 0A        176      asl                ; then C=1
C49B          177 ;
C49B 68        178      pla
C49C A8        179      tay
C49D          180 ;
C49D BA        181      tsx
C49E          182 ;

```

```

C49E A9 00      183      lda #*-*          ; get RTI opcode
C4A0           184      dfs !-1
C49F 40        185      rti
C4A0 48        186      pha
C4A1           187      ;
C4A1 A9 C0     188      lda /CXROMOFF
C4A3 48        189      pha
C4A4           190      ;
C4A4 A9 06     191      lda #CXROMOFF
C4A6 69 00     192      adc #ZERO          ; add 1 if internal slot space
C4A8 48        193      pha
C4A9           194      ;
C4A9 A9 00     195      lda #*-*          ; get STA ABS opcode
C4AB           196      dfs !-1
C4AA 8D 00 00  197      sta *-*
C4AD           198      dfs !-2
C4AB 48        199      pha
C4AC           200      ;
C4AC           201      ;
C4AC           202      ; Make return address point to code just pushed onto the
C4AC           203      ; STACK.
C4AC           204      ;
C4AC 9A        205      txs
C4AD 8A        206      txa
C4AE           207      ;
C4AE 69 03     208      adc #3
C4B0 AA        209      tax
C4B1           210      ;
C4B1 38        211      sec
C4B2           212      ;
C4B2 E9 07     213      sbc #7
C4B4 9D 00 01  214      sta STACK,X
C4B7           215      ;
C4B7 E8        216      inx
C4B8           217      ;
C4B8 A9 01     218      lda /STACK
C4BA 9D 00 01  219      sta STACK,X
C4BD           220      ;
C4BD 68        221      pla
C4BE AA        222      tax
C4BF           223      ;
C4BF 68        224      pla
C4C0           225      ;
C4C0 60        226      rts          ; go to code on STACK
C4C1           227      ;
C4C1           228      ;
C4C1 83        229      RAMSWTBL byt RAM2WE
C4C2 8B        230      byt RAM1WE
C4C3 8B        231      byt RAM1WE
C4C4 05        232      byt RAMWRON
C4C5 03        233      byt RAMRDON
C4C6 55        234      byt PAGE2ON
C4C7           235      ;
0006           236      SWTBLLEN equ *-RAMSWTBL
C4C7           237      ;
C4C7           238      ;
C4C7           239      dfs 1,ZERO          ; 1 byte
C4C8           240      ;
C4C8           241      ;
C4C8           242      ; Continuation of NXTLINE2.
C4C8           243      ;

```

```

C4C8 20 00 C5      244 FORM      jsr GETNSP
C4CB              245 ;
C4CB 84 34         246      sty YSAV
C4CD              247 ;
C4CD DD 30 FA     248      cmp CHAR1,X
C4D0 D0 13        249      bne >1
C4D2              250 ;
C4D2 20 00 C5     251      jsr GETNSP
C4D5              252 ;
C4D5 DD 36 FA     253      cmp CHAR2,X
C4D8 F0 0D        254      beq >3
C4DA              255 ;
C4DA BD 36 FA     256      lda CHAR2,X
C4DD F0 07        257      beq >2
C4DF              258 ;
C4DF C9 A4        259      cmp #"$"
C4E1 F0 03        260      beq >2
C4E3              261 ;
C4E3 A4 34        262      ldy YSAV
C4E5              263 ;
C4E5 18           264 ^1      clc
C4E6              265 ;
C4E6 88           266 ^2      dey
C4E7              267 ;
C4E7 26 44        268 ^3      rol OPRND
C4E9              269 ;
C4E9 E0 03        270      cpx #3
C4EB D0 0D        271      bne >5
C4ED              272 ;
C4ED 20 A7 FF     273      jsr GETNUM
C4F0              274 ;
C4F0 A5 3F        275      lda A2H
C4F2 F0 01        276      beq >4
C4F4              277 ;
C4F4 E8           278      inx
C4F5              279 ;
C4F5 86 35        280 ^4      stx YSAV1
C4F7              281 ;
C4F7 A2 03        282      ldx #3
C4F9              283 ;
C4F9 88           284      dey
C4FA              285 ;
C4FA 86 3D        286 ^5      stx A1H
C4FC              287 ;
C4FC CA          288      dex
C4FD 10 C9        289      bpl FORM
C4FF              290 ;
C4FF 60           291      rts
C500              292 ;
C500              293 ;
C500              294 ; GETNSP gets the next non-blank character for the
C500              295 ; Mini-Assembler.
C500              296 ;
C500 20 FD FC     297 GETNSP  jsr UPRMON
C503              298 ;
C503 C9 A0        299      cmp #SPACE
C505 F0 F9        300      beq GETNSP
C507              301 ;
C507 60           302      rts
C508              303 ;
C508              304 ;

```

```

C508 20 75 FE    305 CXSTEP    jsr A1PC
C50B 20 D0 F8    306          jsr INSTDSP
C50E             307 ;
C50E 68          308          pla
C50F 68          309          pla
C510             310 ;
C510 A2 08       311          ldx #INITBLEN
C512             312 ;
C512 BD CC C5    313 ^1      lda INITBL-1,X
C515 95 3C       314          sta A1L,X
C517             315 ;
C517 CA          316          dex
C518 D0 F8       317          bne <1
C51A             318 ;
C51A AD 00 C0    319          lda KEY
C51D 10 19       320          bpl >4
C51F             321 ;
C51F 2C 10 C0    322          bit CLRKEY
C522             323 ;
C522 C9 A0       324          cmp #" "
C524 D0 0E       325          bne >3
C526             326 ;
C526 AD 00 C0    327 ^2      lda KEY
C529 10 FB       328          bpl <2
C52B             329 ;
C52B 2C 10 C0    330          bit CLRKEY
C52E             331 ;
C52E C9 9B       332          cmp #ESCAPE          ; ESC check
C530 D0 02       333          bne >3
C532             334 ;
C532 E6 34       335          inc YSAV
C534             336 ;
C534 C9 83       337 ^3      cmp #$83          ; ^C break
C536 F0 04       338          beq >5
C538             339 ;
C538 A1 3A       340 ^4      lda (PCL,X)
C53A D0 03       341          bne >6
C53C             342 ;
C53C 38          343 ^5      sec          ; break return
C53D B0 77       344          bcs >4          ; always taken
C53F             345 ;
C53F A4 2F       346 ^6      ldy LENGTH
C541             347 ;
C541 C9 00       348          cmp #*- *
C543             349          dfs !-1
C542 60          350          rts
C543 F0 43       351          beq XRTS
C545             352 ;
C545 C9 00       353          cmp #*- *
C547             354          dfs !-1
C546 20 00 00    355          jsr *- *
C549             356          dfs !-2
C547 F0 4F       357          beq XJSR
C549             358 ;
C549 C9 00       359          cmp #*- *
C54B             360          dfs !-1
C54A 4C 00 00    361          jmp *- *
C54D             362          dfs !-2
C54B F0 56       363          beq XJMP
C54D             364 ;
C54D C9 00       365          cmp #*- *

```

```

C54F      366      dfs !-1
C54E 6C 00 00 367      jmp (*-*)
C551      368      dfs !-2
C54F F0 53    369      beq XJMPAT
C551      370      ;
C551 C9 00    371      cmp #*-*
C553      372      dfs !-1
C552 7C 00 00 373      jmp (*-*,X)
C555      374      dfs !-2
C553 F0 20    375      beq XJMPX
C555      376      ;
C555 C9 00    377      cmp #*-*
C557      378      dfs !-1
C556 40       379      rti
C557 F0 2B    380      beq XRTI
C559      381      ;
C559 C9 00    382      cmp #*-*
C55B      383      dfs !-1
C55A 80 00    384      bra *+2
C55C      385      dfs !-1
C55B D0 02    386      bne >7
C55D      387      ;
C55D A9 00    388      lda #*-*
C55F      389      dfs !-1
C55E 10 00    390      bpl *+2
C560      391      dfs !-1
C55F      392      ;
C55F 29 1F    393      ^7 and #$1F
C561 49 14    394      eor #$14
C563      395      ;
C563 C9 04    396      cmp #4
C565 F0 02    397      beq >9
C567      398      ;
C567 B1 3A    399      ^8 lda (PCL),Y
C569      400      ;
C569 99 3C 00 401      ^9 sta A1L,Y
C56C      402      ;
C56C 88       403      dey
C56D 10 F8    404      bpl <8
C56F      405      ;
C56F 20 3F FF 406      jsr RESTORE
C572      407      ;
C572 4C 3C 00 408      jmp A1L
C575      409      ;
C575      410      ;
C575 B1 3A    411      XJMPX lda (PCL),Y
C577 AA       412      tax
C578      413      ;
C578 88       414      dey
C579      415      ;
C579 18       416      clc
C57A      417      ;
C57A B1 3A    418      lda (PCL),Y
C57C 65 46    419      adc XREG
C57E 90 01    420      bcc >1
C580      421      ;
C580 E8       422      inx
C581      423      ;
C581 38       424      ^1 sec
C582 B0 26    425      bcs >2
C584      426      ;

```

; always taken

```

C584 18          427  XRTI      clc
C585          428  ;
C585 68          429          pla
C586 85 48       430          sta PREG
C588          431  ;
C588 68          432  XRTS      pla
C589 85 3A       433          sta PCL
C58B          434  ;
C58B 68          435          pla
C58C          436  ;
C58C 85 3B       437  PCINC2    sta PCH
C58E          438  ;
C58E A5 2F       439  PCINC3    lda LENGTH
C590 20 56 F9    440          jsr PCADJ3
C593          441  ;
C593 84 3B       442          sty PCH
C595          443  ;
C595 18          444          clc
C596 90 14       445          bcc >3          ; always taken
C598          446  ;
C598 18          447  XJSR      clc
C599          448  ;
C599 20 54 F9    449          jsr PCADJ2
C59C          450  ;
C59C AA          451          tax
C59D          452  ;
C59D 98          453          tya
C59E 48          454          pha
C59F          455  ;
C59F 8A          456          txa
C5A0 48          457          pha
C5A1          458  ;
C5A1 A0 02       459          ldy #2
C5A3          460  ;
C5A3 18          461  XJMP      clc
C5A4          462  ;
C5A4 B1 3A       463  XJMPAT    lda (PCL),Y
C5A6 AA          464          tax
C5A7          465  ;
C5A7 88          466          dey
C5A8          467  ;
C5A8 B1 3A       468          lda (PCL),Y
C5AA          469  ;
C5AA 86 3B       470  ^2      stx PCH
C5AC          471  ;
C5AC 85 3A       472  ^3      sta PCL
C5AE          473  ;
C5AE B0 F3       474          bcs XJMP
C5B0          475  ;
C5B0 20 D7 FA    476          jsr REGDSP
C5B3          477  ;
C5B3 A4 34       478          ldy YSAV
C5B5          479  ;
C5B5 18          480          clc          ; normal return
C5B6          481  ;
C5B6 4C CA FC    482  ^4      jmp STEPRTN
C5B9          483  ;
C5B9          484  ;
C5B9 18          485  BRANCH    clc
C5BA          486  ;
C5BA A0 01       487          ldy #1

```



```

C5BC          488 ;
C5BC B1 3A    489     lda (PCL),Y
C5BE 20 56 F9 490     jsr PCADJ3
C5C1          491 ;
C5C1 85 3A    492     sta PCL
C5C3          493 ;
C5C3 98       494     tya
C5C4          495 ;
C5C4 38       496     sec
C5C5 B0 C5    497     bcs PCINC2           ; always taken
C5C7          498 ;
C5C7          499 ;
C5C7 20 4A FF 500 BRANCH2 jsr SAVE
C5CA          501 ;
C5CA 38       502     sec
C5CB B0 C1    503     bcs PCINC3           ; always taken
C5CD          504 ;
C5CD          505 ;
C5CD EA       506 INITBL  nop
C5CE EA       507     nop
C5CF          508 ;
C5CF 4C C7 C5 509     jmp BRANCH2
C5D2          510 ;
C5D2 4C B9 C5 511     jmp BRANCH
C5D5          512 ;
0008          513 INITBLEN equ *-INITBL
C5D5          514 ;
C5D5          515 ;
C5D5          516 ; Form index into mnemonic table.
C5D5          517 ;
C5D5          518 ; 1) 1XXX1010 => 00101XXX
C5D5          519 ; 2) XXXYYY01 => 00111XXX
C5D5          520 ; 3) XXXYYY10 => 00110XXX
C5D5          521 ; 4) XXXYY100 => 00100XXX
C5D5          522 ; 5) XXXXX000 => 000XXXXX
C5D5          523 ;
C5D5 A2 0B    524 XGETFMT2 ldx #OPTBLL-OPTBLC+1
C5D7          525 ;
C5D7 DD 71 CA 526 ^1     cmp OPTBLC,X
C5DA D0 06    527     bne >2
C5DC          528 ;
C5DC BD 7D CA 529     lda OPTBLL,X
C5DF          530 ;
C5DF A0 00    531     ldy #ZERO
C5E1          532 ;
C5E1 60       533     rts
C5E2          534 ;
C5E2 CA       535 ^2     dex
C5E3 10 F2    536     bpl <1
C5E5          537 ;
C5E5 29 8F    538     and #$8F
C5E7          539 ;
C5E7 AA       540     tax
C5E8          541 ;
C5E8 A5 2A    542     lda BAS2L           ; get OP CODE
C5EA          543 ;
C5EA A0 03    544     ldy #3
C5EC          545 ;
C5EC E0 8A    546     cpx #$8A
C5EE F0 0B    547     beq >5
C5F0          548 ;

```

```
C5F0 4A          549  ^3      lsr
C5F1 90 08       550          bcc >5
C5F3             551  ;
C5F3 4A          552          lsr
C5F4             553  ;
C5F4 4A          554  ^4      lsr
C5F5 09 20       555          ora #$20
C5F7             556  ;
C5F7 88          557          dey
C5F8 D0 FA       558          bne <4
C5FA             559  ;
C5FA C8          560          iny
C5FB             561  ;
C5FB 88          562  ^5      dey
C5FC D0 F2       563          bne <3
C5FE             564  ;
C5FE 60          565          rts
C5FF             566  ;
C5FF             567  ;
C5FF             568          dfs 1,ZERO      ; 1 byte
C600             569  ;
C600             570  ;
C600             571          icl "SW.L"
```

```
LLOAD SW.L,A$4000
```

```

C600          1          ttl "ROM Source Code, SW.L"
C600          2          ;
C600          3          ;
C600          4          ; SW.L
C600          5          ;
C600          6          ;
C600          7          ; Implementation of Cornelis Bongers Garbage Collection
C600          8          ; algorithm.
C600          9          ;
C600 24 95      10  PROCVAR bit PROCESS
C602 30 4A      11          bmi PROCSPCL
C604          12          ;
C604 B1 9B      13          lda (LOWTR),Y
C606 85 5E      14          sta INDEX
C608          15          ;
C608 C5 6F      16          cmp FRETOP
C60A          17          ;
C60A C8          18          iny
C60B          19          ;
C60B B1 9B      20          lda (LOWTR),Y
C60D 85 5F      21          sta INDEX+1
C60F          22          ;
C60F E5 70      23          sbc FRETOP+1
C611 90 3A      24          bcc >4
C613          25          ;
C613 A0 00      26          ldy #ZERO
C615          27          ;
C615 B1 5E      28          lda (INDEX),Y
C617 48          29          pha
C618          30          ;
C618 B1 9B      31          lda (LOWTR),Y
C61A F0 31      32          beq >4
C61C          33          ;
C61C C8          34          iny
C61D          35          ;
C61D C9 02      36          cmp #2
C61F F0 05      37          beq >1
C621          38          ;
C621 B0 12      39          bcs >2
C623          40          ;
C623 A9 FF      41          lda #NEGONE
C625          42          ;
C625 2C 00 00    43          bit *-*
C628          44          dfs !-2
C626          45          ;
C626 B1 5E      46          ^1 lda (INDEX),Y
C628          47          ;
C628 91 9B      48          sta (LOWTR),Y
C62A          49          ;
C62A C8          50          iny
C62B          51          ;
C62B A9 FF      52          lda #NEGONE
C62D 85 CD      53          sta SPCLFLAG
C62F 91 9B      54          sta (LOWTR),Y
C631          55          ;
C631 A0 00      56          ldy #ZERO
C633 F0 15      57          beq >3          ; always taken
C635          58          ;
C635 B1 5E      59          ^2 lda (INDEX),Y
C637          60          ;

```

```

C637 C8          61          iny
C638          62          ;
C638 91 9B      63          sta (LOWTR),Y
C63A          64          ;
C63A B1 5E      65          lda (INDEX),Y
C63C 09 80      66          ora #MSBSET
C63E 91 5E      67          sta (INDEX),Y
C640          68          ;
C640 A0 00      69          ldy #ZERO
C642          70          ;
C642 A5 9B      71          lda LOWTR
C644 91 5E      72          sta (INDEX),Y
C646          73          ;
C646 C8         74          iny
C647          75          ;
C647 8A         76          txa
C648 91 5E      77          sta (INDEX),Y
C64A          78          ;
C64A 68         79          ^3      pla
C64B 91 9B      80          sta (LOWTR),Y
C64D          81          ;
C64D 60         82          ^4      rts
C64E          83          ;
C64E          84          ;
C64E C8         85          PROCSPCL iny
C64F 84 94      86          sty LEN
C651          87          ;
C651 A9 FF      88          lda #NEGONE
C653 D1 9B      89          cmp (LOWTR),Y
C655 D0 F6      90          bne <4
C657          91          ;
C657 88         92          dey
C658          93          ;
C658 D1 9B      94          cmp (LOWTR),Y
C65A D0 02      95          bne >1
C65C          96          ;
C65C C6 94      97          dec LEN
C65E          98          ;
C65E 20 73 E5   99          ^1      jsr COPYVAR
C661          100         ;
C661 A5 94      101         lda LEN
C663 91 9B      102         sta (LOWTR),Y
C665          103         ;
C665 C8         104         iny
C666          105         ;
C666 A5 8A      106         lda FUNCNAM
C668 91 9B      107         sta (LOWTR),Y
C66A          108         ;
C66A C8         109         iny
C66B          110         ;
C66B A5 8B      111         lda FUNCNAM+1
C66D 91 9B      112         sta (LOWTR),Y
C66F          113         ;
C66F 60         114         rts
C670          115         ;
C670          116         ;
C670          117         ; This version of the Sweet 16 Metaprocessor has been
C670          118         ; revised from the original; it has 5 added instructions.
C670          119         ;
C670          120         ; Sweet 16 Registers
C670          121         ;

```

```

C670      122 ; Register Description
C670      123 ; -----
C670      124 ; R0 Accumulator (ACC)
C670      125 ; R1-R11 User registers
C670      126 ; R12 Subroutine return Stack Pointer
C670      127 ; R13 Compare instruction results
C670      128 ; R14 Status Register (PR & carry flag)
C670      129 ; R15 Program Counter (PC)
C670      130 ;
C670      131 ;
C670      132 ; Sweet 16 Non-Register Opcodes
C670      133 ;
C670      134 ; Opcode Mnemonic Description
C670      135 ; -----
C670      136 ; 00 RTN Return to 6502 mode
C670      137 ; 01 BR adr Branch always, PC+ = adr+2
C670      138 ; 02 BNC adr Branch if no carry, PC+ = adr+2
C670      139 ; 03 BC adr Branch if carry, PC+ = adr+2
C670      140 ; 04 BP adr Branch if plus, PC+ = adr+2
C670      141 ; 05 BM adr Branch if minus, PC+ = adr+2
C670      142 ; 06 BZ adr Branch if zero, PC+ = adr+2
C670      143 ; 07 BNZ adr Branch if not zero, PC+ = adr+2
C670      144 ; 08 BM1 adr Branch if minus 1, PC+ = adr+2
C670      145 ; 09 BNM1 adr Branch if not minus 1, PC+ = adr+2
C670      146 ; 0A SOUT chr Send "chr" to COUT
C670      147 ; 0B RS Return from subroutine, PC = (R12--)
C670      148 ; 0C BS adr Branch to subroutine, (R12++) = PC
C670      149 ; 0D RSNS Return from subroutine, PC = R12
C670      150 ; 0E BSNS adr Branch to subroutine, R12 = PC
C670      151 ; 0F SJMP adr Jump to address, PC = adr
C670      152 ;
C670      153 ;
C670      154 ; Sweet 16 Register Opcodes
C670      155 ;
C670      156 ; Opcode Mnemonic Description
C670      157 ; -----
C670      158 ; 1n SET Rn,C Load Rn immediate (with constant)
C670      159 ; 2n LD Rn Load LO ACC from Rn
C670      160 ; 3n ST Rn Move LO ACC into Rn
C670      161 ; 4n LD @Rn Load LO ACC indirectly using Rn, Rn+1
C670      162 ; 5n ST @Rn Move LO ACC indirectly using Rn, Rn+1
C670      163 ; 6n LDD @Rn Load ACC indirectly using Rn, Rn+2
C670      164 ; 7n STD @Rn Move ACC indirectly using Rn, Rn+2
C670      165 ; 8n POP @Rn Rn-1, load LO ACC indirectly using Rn
C670      166 ; 9n STP @Rn Rn-1, move LO ACC indirectly using Rn
C670      167 ; An ADD Rn Add Rn to ACC with carry out
C670      168 ; Bn SUB Rn Subtract Rn from ACC, with carry out
C670      169 ; Cn POPD @Rn Rn-2, HO ACC, Rn-1, LO ACC indirectly
C670      170 ; Dn CPR Rn ACC-Rn->R13, with carry out
C670      171 ; En INR Rn Rn+1->Rn
C670      172 ; Fn DCR Rn Rn-1->Rn
C670      173 ;
C670      174 ;
C670      175 ; The Status Register contains the previous register*2 in
C670      176 ; the LO byte and the carry flag in bit 0 of the HO byte.
C670      177 ;
C670      178 ; Branch opcodes depend on the results of the prior opcode.
C670      179 ; The register*2 of the prior opcode is saved in LO R14.
C670      180 ; It is the value currently in the prior register that is
C670      181 ; tested for the selected branch opcode.
C670      182 ;

```

```

C670      183 ; Before the BS/RS opcodes can be utilized, R12 must be
C670      184 ; initialized with the address of the STACK that contains
C670      185 ; the return from subroutine addresses.
C670      186 ;
C670      187 ;
C670      188 ; Preserve 6502 registers and init SW16 program counter.
C670      189 ;
C670 20 4A FF 190 SW16      jsr SAVE
C673      191 ;
C673 68      192          pla
C674 85 1E    193          sta R15L
C676      194 ;
C676 68      195          pla
C677 85 1F    196          sta R15H
C679      197 ;
C679      198 ;
C679      199 ; Interpret a single SW16 opcode, then fetch the next SW16
C679      200 ; opcode.
C679      201 ;
C679 20 7F C6 202 SW16B      jsr SW16C
C67C      203 ;
C67C 4C 79 C6 204          jmp SW16B
C67F      205 ;
C67F      206 ;
C67F      207 ; Increment the SW16 program counter.
C67F      208 ;
C67F E6 1E    209 SW16C      inc R15L
C681 D0 02    210          bne >1
C683      211 ;
C683 E6 1F    212          inc R15H
C685      213 ;
C685      214 ;
C685      215 ; Push the common HO routine address byte onto the STACK
C685      216 ; and process the SW16 opcode.
C685      217 ;
C685 A9 C7    218 ^1      lda /RTNCMD
C687 48      219          pha
C688      220 ;
C688 A0 00    221          ldy #ZERO
C68A      222 ;
C68A      223 ;
C68A      224 ; Mask the specified SW16 register and double it for
C68A      225 ; non-register address table indexing or prior register.
C68A      226 ;
C68A B1 1E    227          lda (R15L),Y
C68C 29 0F    228          and #$0F
C68E      229 ;
C68E 0A      230          asl
C68F AA      231          tax
C690      232 ;
C690      233 ;
C690      234 ; Extract opcode. If the opcode is zero, process a non-
C690      235 ; register opcode routine.
C690      236 ;
C690 4A      237          lsr
C691      238 ;
C691 51 1E    239          eor (R15L),Y
C693 F0 0D    240          beq TOBR
C695      241 ;
C695      242 ;
C695      243 ; Save register*2 in LO status byte and clear HO status

```

```

C695      244 ; byte. Form opcode*2 for register address table indexing.
C695      245 ;
C695 86 1C  246      stx R14L
C697 84 1D  247      sty R14H
C699      248 ;
C699 4A     249      lsr
C69A 4A     250      lsr
C69B 4A     251      lsr
C69C      252 ;
C69C A8     253      tay
C69D      254 ;
C69D      255 ;
C69D      256 ; Save the LO address onto the STACK to process this SW16
C69D      257 ; register opcode. Also C-flag = 0; N-flag = 0 for SETCMD.
C69D      258 ;
C69D B9 E1 C6 259      lda OPTBL-2,Y
C6A0 48     260      pha
C6A1      261 ;
C6A1 60     262      rts
C6A2      263 ;
C6A2      264 ;
C6A2      265 ; Increment the SW16 program counter for the expected
C6A2      266 ; branch address. Save the LO address onto the STACK to
C6A2      267 ; process this SW16 non-register opcode.
C6A2      268 ;
C6A2 E6 1E  269 TOBR    inc R15L
C6A4 D0 02  270      bne >2
C6A6      271 ;
C6A6 E6 1F  272      inc R15H
C6A8      273 ;
C6A8 BD E2 C6 274 ^2     lda BRTBL,X
C6AB 48     275      pha
C6AC      276 ;
C6AC      277 ;
C6AC      278 ; Recall prior register*2 in LO status byte and set up
C6AC      279 ; carry flag from HO status byte for BC and BNC opcodes.
C6AC      280 ; Also N-flag = 0 and Y-reg = 0.
C6AC      281 ;
C6AC A6 1C  282      ldx R14L
C6AE      283 ;
C6AE A5 1D  284      lda R14H
C6B0 4A     285      lsr
C6B1      286 ;
C6B1 60     287      rts
C6B2      288 ;
C6B2      289 ;
C6B2      290 ; Y-reg is set to 2 by SET opcode index*2. First get HO
C6B2      291 ; byte, then LO byte for designated register.
C6B2      292 ;
C6B2 B1 1E  293 SETCMD2  lda (R15L),Y
C6B4 95 01  294      sta R0H,X
C6B6      295 ;
C6B6 88     296      dey
C6B7      297 ;
C6B7 B1 1E  298      lda (R15L),Y
C6B9 95 00  299      sta R0L,X
C6BB      300 ;
C6BB      301 ;
C6BB      302 ; Increment the SW16 program counter by 2. C-flag clear.
C6BB      303 ;
C6BB A5 1E  304      lda R15L

```

```

C6BD 69 02      305      adc #2
C6BF 85 1E      306      sta R15L
C6C1 90 02      307      bcc >3
C6C3           308      ;
C6C3 E6 1F      309      inc R15H
C6C5           310      ;
C6C5 60         311      ^3      rts
C6C6           312      ;
C6C6           313      ;
C6C6           314      ; R12 must contain the STACK address where to pop the
C6C6           315      ; return SW16 program counter. Pop the HO byte, then
C6C6           316      ; the LO byte. Then R12 - 2 -> R12.
C6C6           317      ;
C6C6           318      ;
C6C6 A2 18      319      RSCMD2    ldx #R12L
C6C8           320      ;
C6C8 20 F7 C7   321      jsr DCRCMD
C6CB           322      ;
C6CB A1 00      323      lda (R0L,X)
C6CD 85 1F      324      sta R15H
C6CF           325      ;
C6CF 20 F7 C7   326      jsr DCRCMD
C6D2           327      ;
C6D2 A1 00      328      lda (R0L,X)
C6D4 85 1E      329      sta R15L
C6D6           330      ;
C6D6 60         331      rts
C6D7           332      ;
C6D7           333      ;
C6D7           334      ; Copy the current SW16 program counter in R15 to R12.
C6D7           335      ;
C6D7 A5 1E      336      BSNSCMD2  lda R15L
C6D9 85 18      337      sta R12L
C6DB           338      ;
C6DB A5 1F      339      lda R15H
C6DD 85 19      340      sta R12H
C6DF           341      ;
C6DF 4C 9C C7   342      jmp BRCMD
C6E2           343      ;
C6E2           344      ;
C6E2           345      ; The byte-pairs of this table have been swapped to remove
C6E2           346      ; the unused Fn+1 entry.
C6E2           347      ;
C6E2 00         348      BRTBL      byt RTNCMD-1      ; 0
C6E3 08         349      OPTBL      byt SETCMD-1      ; 1n
C6E4 9B         350      byt BRCMD-1      ; 1
C6E5 0E         351      byt LDCMD-1      ; 2n
C6E6 9C         352      byt BNCCMD-1     ; 2
C6E7 17         353      byt STCMD-1      ; 3n
C6E8 AD         354      byt BCCMD-1      ; 3
C6E9 20         355      byt LD@CMD-1     ; 4n
C6EA B0         356      byt BPCMD-1      ; 4
C6EB 2A         357      byt ST@CMD-1     ; 5n
C6EC B5         358      byt BMCMD-1      ; 5
C6ED 39         359      byt LDD@CMD-1    ; 6n
C6EE BA         360      byt BZCMD-1      ; 6
C6EF 42         361      byt STD@CMD-1    ; 7n
C6F0 C1         362      byt BNZCMD-1     ; 7
C6F1 52         363      byt POP@CMD-1    ; 8n
C6F2 C8         364      byt BM1CMD-1     ; 8
C6F3 62         365      byt STP@CMD-1    ; 9n

```



```

C6F4 D1      366      byt BNM1CMD-1      ; 9
C6F5 6B      367      byt ADDCMD-1      ; An
C6F6 DA      368      byt SOUTCMD-1     ; A
C6F7 79      369      byt SUBCMD-1      ; Bn
C6F8 0A      370      byt RSCMD-1       ; B
C6F9 4B      371      byt POPD@CMD-1    ; Cn
C6FA 8F      372      byt BSCMD-1       ; C
C6FB 7B      373      byt CPRCMD-1      ; Dn
C6FC DF      374      byt RSNSCMD-1     ; D
C6FD 32      375      byt INRCMD-1      ; En
C6FE 0C      376      byt BSNSCMD-1     ; E
C6FF F6      377      byt DCRCMD-1      ; Fn
C700 E8      378      byt SJMPCMD-1     ; F
C701          379      ;
C701          380      ;
C701          381      ; The following routines must reside on the same page.
C701          382      ;
C701          383      ; RTN opcode returns to 6502 processing mode. Pop the
C701          384      ; SW16C return address, restore the 6502 registers, and
C701          385      ; return to 6502 mode using the SW16 program counter.
C701          386      ;
C701 68      387      RTNCMD    pla
C702 68      388              pla
C703          389      ;
C703 20 3F FF 390              jsr RESTORE
C706          391      ;
C706 4C 78 FA 392              jmp SW16RTN
C709          393      ;
C709          394      ;
C709          395      ; SET opcode gets a 2-byte immediate constant.
C709          396      ;
C709 10 A7    397      SETCMD    bpl SETCMD2      ; always taken
C70B          398      ;
C70B          399      ;
C70B          400      ; RS opcode is return from a BS subroutine call.
C70B          401      ;
C70B 10 B9    402      RSCMD     bpl RSCMD2      ; always taken
C70D          403      ;
C70D          404      ;
C70D          405      ; BSNS opcode is branch to subroutine that does not use a
C70D          406      ; return address STACK.
C70D          407      ;
C70D 10 C8    408      BSNSCMD   bpl BSNSCMD2     ; always taken
C70F          409      ;
C70F          410      ;
C70F          411      ; LD opcode moves the contents of Rn to ACC.
C70F          412      ;
C70F B5 00    413      LDCMD     lda R0L,X
C711 85 00    414              sta R0L
C713          415      ;
C713 B5 01    416              lda R0H,X
C715 85 01    417              sta R0H
C717          418      ;
C717 60      419              rts
C718          420      ;
C718          421      ;
C718          422      ; ST opcode moves the contents of ACC to Rn.
C718          423      ;
C718 A5 00    424      STCMD     lda R0L
C71A 95 00    425              sta R0L,X
C71C          426      ;

```

```

C71C A5 01      427      lda R0H
C71E 95 01      428      sta R0H,X
C720            429      ;
C720 60         430      rts
C721            431      ;
C721            432      ;
C721            433      ; LD@ opcode loads the LO ACC indirectly from memory
C721            434      ; using the address in Rn.  HO ACC is cleared.  Make R0
C721            435      ; the prior register.
C721            436      ;
C721 A1 00      437 LD@CMD  lda (R0L,X)
C723 85 00      438      sta R0L
C725            439      ;
C725 A0 00      440      ldy #ZERO
C727 84 01      441      sty R0H
C729 F0 06      442      beq >4                ; always taken
C72B            443      ;
C72B            444      ;
C72B            445      ; ST@ opcode stores the contents of LO ACC indirectly to
C72B            446      ; memory using the address in Rn.  Make R0 the prior
C72B            447      ; register.  Fall into INRCMD.
C72B            448      ;
C72B A5 00      449 ST@CMD  lda R0L
C72D            450      ;
C72D 81 00      451 ST@CMD2 sta (R0L,X)
C72F            452      ;
C72F A0 00      453      ldy #ZERO
C731            454      ;
C731 84 1C      455 ^4      sty R14L
C733            456      ;
C733            457      ;
C733            458      ; INR opcode increments the register Rn.
C733            459      ;
C733 F6 00      460 INRCMD  inc R0L,X
C735 D0 02      461      bne >5
C737            462      ;
C737 F6 01      463      inc R0H,X
C739            464      ;
C739 60         465 ^5      rts
C73A            466      ;
C73A            467      ;
C73A            468      ; LDD@ opcode loads the ACC indirectly from memory using
C73A            469      ; the address in Rn.  The LO byte is loaded first.  Make
C73A            470      ; R0 the prior register.
C73A            471      ;
C73A 20 21 C7   472 LDD@CMD  jsr LD@CMD
C73D            473      ;
C73D A1 00      474      lda (R0L,X)
C73F 85 01      475      sta R0H
C741            476      ;
C741 90 F0      477      bcc INRCMD                ; always taken
C743            478      ;
C743            479      ;
C743            480      ; STD@ opcode stores the contents of the ACC indirectly to
C743            481      ; memory using the address in Rn.  LO ACC is stored first.
C743            482      ; Make R0 the prior register.
C743            483      ;
C743 20 2B C7   484 STD@CMD  jsr ST@CMD
C746            485      ;
C746 A5 01      486      lda R0H
C748 81 00      487      sta (R0L,X)

```

```

C74A      488 ;
C74A 90 E7      489             bcc INRCMD             ; always taken
C74C      490 ;
C74C      491 ;
C74C      492 ; POPD@ opcode decrements Rn and gets the HO byte
C74C      493 ; indirectly from memory using the address in Rn.
C74C      494 ; Fall into POP@CMD.
C74C      495 ;
C74C 20 F7 C7    496 POPD@CMD jsr DCRCMD
C74F      497 ;
C74F A1 00      498             lda (R0L,X)
C751 A8         499             tay
C752      500 ;
C752 2C 00 00    501             bit *-*
C755      502             dfs !-2
C753      503 ;
C753      504 ;
C753      505 ; POP@ opcode sets the HO byte to zero, then decrements Rn
C753      506 ; and gets the LO byte indirectly from memory using the
C753      507 ; address in Rn. Move the HO and LO bytes into the ACC.
C753      508 ; Make R0 the prior register.
C753      509 ;
C753 A0 00      510 POP@CMD ldy #ZERO
C755      511 ;
C755 20 F7 C7    512             jsr DCRCMD
C758      513 ;
C758 A1 00      514             lda (R0L,X)
C75A 85 00      515             sta R0L
C75C      516 ;
C75C 84 01      517             sty R0H
C75E      518 ;
C75E A0 00      519 ^6 ldy #ZERO
C760 84 1C      520             sty R14L
C762      521 ;
C762 60         522             rts
C763      523 ;
C763      524 ;
C763      525 ; STP@ opcode decrements Rn and stores the LO ACC
C763      526 ; indirectly to memory using the address in Rn. Make R0
C763      527 ; the prior register.
C763      528 ;
C763 20 F7 C7    529 STP@CMD jsr DCRCMD
C766      530 ;
C766 A5 00      531             lda R0L
C768 81 00      532             sta (R0L,X)
C76A      533 ;
C76A 90 F2      534             bcc <6             ; always taken
C76C      535 ;
C76C      536 ;
C76C      537 ; ADD opcode sets Y-reg to 0 so ACC + Rn -> ACC, carry
C76C      538 ; saved to HO status byte. Make R0 the prior register.
C76C      539 ;
C76C A5 00      540 ADDCMD ldy #ZERO
C76E 75 00      541             adc R0L,X
C770 85 00      542             sta R0L
C772      543 ;
C772 A5 01      544             lda R0H
C774 75 01      545             adc R0H,X
C776      546 ;
C776 A0 00      547             ldy #ZERO
C778 F0 0E      548             beq >7             ; always taken

```

```

C77A          549 ;
C77A          550 ;
C77A          551 ; SUB opcode sets Y-reg to 0 so ACC - Rn -> ACC, carry
C77A          552 ; saved to HO status byte. Make R0 the prior register.
C77A          553 ;
C77A A0 00    554 SUBCMD    ldy #ZERO
C77C          555 ;
C77C          556 ;
C77C          557 ; CPR opcode leaves Y-reg set to 13*2 (SET opcode index*2)
C77C          558 ; so ACC - Rn -> R13, carry saved to HO status byte. Make
C77C          559 ; R13 the prior register.
C77C          560 ;
C77C 38       561 CPRCMD    sec
C77D          562 ;
C77D A5 00    563             lda R0L
C77F F5 00    564             sbc R0L,X
C781 99 00 00 565             sta R0L,Y
C784          566 ;
C784 A5 01    567             lda R0H
C786 F5 01    568             sbc R0H,X
C788          569 ;
C788 99 01 00 570 ^7       sta R0H,Y
C78B          571 ;
C78B 84 1C    572             sty R14L
C78D 26 1D    573             rol R14H                ; save carry out bit
C78F          574 ;
C78F 60       575             rts
C790          576 ;
C790          577 ;
C790          578 ; BS opcode is branch to subroutine. Save the current
C790          579 ; SW16 program counter indirectly to memory using the
C790          580 ; address in R12. R15L is saved first, then R15H; finally
C790          581 ; R12 + 2 -> R12. Fall into BRCMD.
C790          582 ;
C790 A2 18    583 BSCMD    ldx #R12L
C792          584 ;
C792 A5 1E    585             lda R15L
C794 20 2D C7 586             jsr ST@CMD2
C797          587 ;
C797 A5 1F    588             lda R15H
C799 20 2D C7 589             jsr ST@CMD2
C79C          590 ;
C79C          591 ;
C79C          592 ; BR opcode is branch always. Fall into BNCCMD.
C79C          593 ;
C79C 18       594 BRCMD    clc
C79D          595 ;
C79D          596 ;
C79D          597 ; BNC opcode is branch if carry is clear from prior opcode.
C79D          598 ;
C79D B0 0E    599 BNCCMD    bcs >9
C79F          600 ;
C79F          601 ;
C79F          602 ; Get displacement byte. If it is negative set Y-reg to
C79F          603 ; minus one for 2's compliment addition. Y-reg = 0.
C79F          604 ;
C79F B1 1E    605             lda (R15L),Y
C7A1 10 01    606             bpl >8
C7A3          607 ;
C7A3 88       608             dey
C7A4          609 ;

```

```

C7A4      610 ;
C7A4      611 ; Add displacement to program counter.
C7A4      612 ;
C7A4 65 1E 613 ^8      adc R15L
C7A6 85 1E 614          sta R15L
C7A8      615 ;
C7A8 98    616          tya
C7A9      617 ;
C7A9 65 1F 618          adc R15H
C7AB 85 1F 619          sta R15H
C7AD      620 ;
C7AD 60    621 ^9      rts
C7AE      622 ;
C7AE      623 ;
C7AE      624 ; BC opcode is branch if carry is set from prior opcode.
C7AE      625 ;
C7AE B0 EC 626 BCCMD    bcs BRCMD
C7B0      627 ;
C7B0 60    628          rts
C7B1      629 ;
C7B1      630 ;
C7B1      631 ; BP opcode is branch if the value of the prior register is
C7B1      632 ; positive.
C7B1      633 ;
C7B1 B5 01 634 BPCMD    lda R0H,X
C7B3 10 E7 635          bpl BRCMD
C7B5      636 ;
C7B5 60    637          rts
C7B6      638 ;
C7B6      639 ;
C7B6      640 ; BM opcode is branch if the value of the prior register is
C7B6      641 ; negative.
C7B6      642 ;
C7B6 B5 01 643 BMCMD    lda R0H,X
C7B8 30 E2 644          bmi BRCMD
C7BA      645 ;
C7BA 60    646          rts
C7BB      647 ;
C7BB      648 ;
C7BB      649 ; BZ opcode is branch if the value of the prior register is
C7BB      650 ; zero.
C7BB      651 ;
C7BB B5 00 652 BZCMD    lda R0L,X
C7BD 15 01 653          ora R0H,X
C7BF F0 DB 654          beq BRCMD
C7C1      655 ;
C7C1 60    656          rts
C7C2      657 ;
C7C2      658 ;
C7C2      659 ; BNZ opcode is branch if the value of the prior register
C7C2      660 ; is not zero.
C7C2      661 ;
C7C2 B5 00 662 BNZCMD    lda R0L,X
C7C4 15 01 663          ora R0H,X
C7C6 D0 D4 664          bne BRCMD
C7C8      665 ;
C7C8 60    666          rts
C7C9      667 ;
C7C9      668 ;
C7C9      669 ; BM1 opcode is branch if the value of the prior register
C7C9      670 ; is negative one.

```

```

C7C9          671 ;
C7C9 B5 00    672 BM1CMD   lda R0L,X
C7CB 35 01    673          and R0H,X
C7CD          674 ;
C7CD 49 FF    675          eor #NEGONE
C7CF F0 CB    676          beq BRCMD
C7D1          677 ;
C7D1 60       678          rts
C7D2          679 ;
C7D2          680 ;
C7D2          681 ; BNM1 opcode is branch if the value of the prior register
C7D2          682 ; is not negative one.
C7D2          683 ;
C7D2 B5 00    684 BNM1CMD   lda R0L,X
C7D4 35 01    685          and R0H,X
C7D6          686 ;
C7D6 49 FF    687          eor #NEGONE
C7D8 D0 C2    688          bne BRCMD
C7DA          689 ;
C7DA 60       690          rts
C7DB          691 ;
C7DB          692 ;
C7DB          693 ; SOUT opcode sends the "chr" value to COUT.
C7DB          694 ;
C7DB B1 1E    695 SOUTCMD   lda (R15L),Y
C7DD          696 ;
C7DD 4C ED FD 697          jmp COUT
C7E0          698 ;
C7E0          699 ;
C7E0          700 ; RSNS opcode is return from a BSNS subroutine call that
C7E0          701 ; does not use a return address STACK. Copy the saved SW16
C7E0          702 ; program counter in R12 to R15.
C7E0          703 ;
C7E0 A5 18    704 RSNSCMD   lda R12L
C7E2 85 1E    705          sta R15L
C7E4          706 ;
C7E4 A5 19    707          lda R12H
C7E6 85 1F    708          sta R15H
C7E8          709 ;
C7E8 60       710          rts
C7E9          711 ;
C7E9          712 ;
C7E9          713 ; SJMP opcode gets a 2-byte immediate address for the SW16
C7E9          714 ; program counter. Get the LO byte, then the HO byte, and
C7E9          715 ; save the address to R15. Make R15 the prior register.
C7E9          716 ; Fall into DCRCMD.
C7E9          717 ;
C7E9 B1 1E    718 SJMPCMD   lda (R15L),Y
C7EB AA       719          tax
C7EC          720 ;
C7EC C8       721          iny
C7ED          722 ;
C7ED B1 1E    723          lda (R15L),Y
C7EF          724 ;
C7EF 86 1E    725          stx R15L
C7F1 85 1F    726          sta R15H
C7F3          727 ;
C7F3 A2 1E    728          ldx #R15L
C7F5 86 1C    729          stx R14L
C7F7          730 ;
C7F7          731 ;

```

```
C7F7          732 ; DCR opcode decrements the register Rn.
C7F7          733 ;
C7F7 B5 00    734 DCRCMD   lda R0L,X
C7F9 D0 02    735         bne >1
C7FB          736 ;
C7FB D6 01    737         dec R0H,X
C7FD          738 ;
C7FD D6 00    739 ^1      dec R0L,X
C7FF          740 ;
C7FF 60       741         rts
C800          742 ;
C800          743 ;
C800          744         icl "C8.L"
```

```
LLOAD C8.L,A$4000
```

```

C800          1          ttl "ROM Source Code, C8.L"
C800          2          ;
C800          3          ;
C800          4          ; C8.L
C800          5          ;
C800          6          ;
C800          7          ; This entry point is only used by Pascal 1.0.
C800          8          ;
C800 4C B0 C9   9  PXINIT    jmp PXINIT2
C803         10          ;
C803         11          ;
C803         12          ; BASIC initialization. This is called by the 0xC3 space
C803         13          ; only after a PR#3 or a JSR $C300.
C803         14          ;
C803         15          ; If the language card is enabled and the ID byte does not
C803         16          ; match, the 0xF8 ROM is copied to the language card. If
C803         17          ; an 80 column card is detected it is enabled, otherwise
C803         18          ; 40 column is enabled. The screen is cleared and the
C803         19          ; character in A-reg is printed.
C803         20          ;
C803 20 F4 CE   21  BASCINIT  jsr COPYROM
C806 20 2A C8   22              jsr C3HOOKS
C809 20 2B CD   23              jsr DO40
C80C         24          ;
C80C A9 01      25              lda #M.MOUSE
C80E 8D FB 04   26              sta XMODE
C811         27          ;
C811 20 90 CA   28              jsr TESTCARD
C814 D0 08      29              bne >1
C816         30          ;
C816 06 21      31              asl WNDWDTH
C818         32          ;
C818 8D 01 C0   33              sta STR80ON
C81B 8D 0D C0   34              sta VID80ON
C81E         35          ;
C81E 8D 0F C0   36          ^1    sta ALTCHON
C821         37          ;
C821 20 74 CC   38              jsr XFF
C824         39          ;
C824 AC 7B 05   40              ldy OURCH
C827         41          ;
C827 4C 7E C8   42              jmp CXVIDCK4
C82A         43          ;
C82A         44          ;
C82A A9 07      45  C3HOOKS  lda #BASICOUT
C82C 85 36      46              sta CSWL
C82E         47          ;
C82E A9 C3      48              lda /BASICOUT
C830 85 37      49              sta CSWH
C832         50          ;
C832         51          ;
C832 A9 05      52  C3IN     lda #BASICIN
C834 85 38      53              sta KSWL
C836         54          ;
C836 A9 C3      55              lda /BASICIN
C838 85 39      56              sta KSWH
C83A         57          ;
C83A 60         58              rts
C83B         59          ;
C83B         60          ;

```



```

C83B      61 ; Moved GETKEY as CXKEYIN after ESCTABL where STAUx was.
C83B      62 ;
C83B      63 ;           dfs 18,ZERO           ; 18 bytes
C84D      64 ;
C84D      65 ;
C84D      66 ; Pascal 1.0 input entry point. Must be at 0xC84D.
C84D      67 ;
C84D 4C 50 C3 68 PXREAD    jmp JPREAD
C850      69 ;
C850      70 ;
C850      71 ; CSETUP compensates for changes to CV, CH, OURCH, and
C850      72 ; WNDWDTH. It updates the video firmware's versions of
C850      73 ; these values.
C850      74 ;
C850 A5 25 75 CSETUP    lda CV
C852 8D FB 05 76         sta OURCV
C855      77 ;
C855 A4 24 78         ldy CH
C857      79 ;
C857 CC 7B 04 80         cpy OLDCH
C85A F0 03 81         beq >1
C85C      82 ;
C85C 8C 7B 05 83         sty OURCH
C85F      84 ;
C85F 18 85 ^1        clc
C860      86 ;
C860 A5 21 87         lda WNDWDTH
C862 ED 7B 05 88         sbc OURCH
C865 B0 05 89         bcs >2
C867      90 ;
C867 A0 00 91         ldy #ZERO
C869 8C 7B 05 92         sty OURCH
C86C      93 ;
C86C AC 7B 05 94 ^2    ldy OURCH
C86F      95 ;
C86F 60 96         rts
C870      97 ;
C870      98 ;
C870      99 ; NEWVW support. CXNEWVW and CXNEWVW2 are used by the
C870     100 ; 0xF8 ROM to input and output characters while VID80 is
C870     101 ; ON. These routines are only called by the 0xC100 to
C870     102 ; 0xC2FF space so as not to cause possible conflict with
C870     103 ; other 0xC800 users.
C870     104 ;
C870 A4 35 105 CXNEWVW   ldy YSAV1
C872     106 ;
C872 18 107         clc
C873     108 ;
C873 B0 00 109         bcs *+2
C875     110         dfs !-1
C874     111 ;
C874 38 112 CXNEWVW2   sec
C875     113 ;
C875 8D 7B 06 114         sta XCHAR
C878     115 ;
C878 98 116         tya
C879 48 117         pha
C87A     118 ;
C87A 8A 119         txa
C87B 48 120         pha
C87C     121 ;

```

```

C87C B0 5E      122 CXVIDCK3 bcs >5
C87E           123 ;
C87E 20 50 C8   124 CXVIDCK4 jsr CSETUP
C881           125 ;
C881 AD 7B 06   126         lda XCHAR
C884           127 ;
C884 C9 8D      128         cmp #RETURN
C886 D0 18      129         bne >2
C888           130 ;
C888 AE 00 C0   131         ldx KEY
C88B 10 13      132         bpl >2
C88D           133 ;
C88D E0 93      134         cpx #CTRLS           ; CTRL-S check
C88F D0 0F      135         bne >2
C891           136 ;
C891 2C 10 C0   137         bit CLRKEY
C894           138 ;
C894 AE 00 C0   139 ^1         ldx KEY
C897 10 FB      140         bpl <1
C899           141 ;
C899 E0 83      142         cpx #CTRLC           ; CTRL-C check
C89B F0 03      143         beq >2
C89D           144 ;
C89D 2C 10 C0   145         bit CLRKEY
C8A0           146 ;
C8A0 29 7F      147 ^2         and #MSBCLR           ; clear MSB
C8A2           148 ;
C8A2 C9 20      149         cmp #$20           ; control character check
C8A4 B0 06      150         bcs >3
C8A6           151 ;
C8A6 20 D2 CA   152         jsr CTRLCHR2
C8A9           153 ;
C8A9 4C BD C8   154         jmp CTRLON
C8AC           155 ;
C8AC AD 7B 06   156 ^3         lda XCHAR
C8AF 20 38 CE   157         jsr STORCHAR
C8B2           158 ;
C8B2 C8         159         iny
C8B3           160 ;
C8B3 8C 7B 05   161         sty OURCH
C8B6           162 ;
C8B6 C4 21      163         cpy WNDWDTH
C8B8 90 03      164         bcc CTRLON
C8BA           165 ;
C8BA 20 51 CB   166         jsr XCR
C8BD           167 ;
C8BD AD FB 04   168 CTRLON    lda XMODE
C8C0 29 F7      169         and #NEGONE-M.CTL
C8C2 8D FB 04   170         sta XMODE
C8C5           171 ;
C8C5 AD 7B 05   172 BIORET    lda OURCH
C8C8           173 ;
C8C8 2C 1F C0   174         bit RDVID80
C8CB 10 02      175         bpl >4
C8CD           176 ;
C8CD A9 00      177         lda #ZERO
C8CF           178 ;
C8CF 85 24      179 ^4         sta CH
C8D1 8D 7B 04   180         sta OLDCH
C8D4           181 ;
C8D4 68         182         pla

```

```

C8D5 AA          183          tax
C8D6          184          ;
C8D6 68          185          pla
C8D7 A8          186          tay
C8D8          187          ;
C8D8 AD 7B 06    188          lda XCHAR
C8DB          189          ;
C8DB 60          190          rts
C8DC          191          ;
C8DC A4 24       192          ^5    ldy CH
C8DE          193          ;
C8DE AD 7B 06    194          lda XCHAR
C8E1 91 28       195          sta (BASL),Y
C8E3          196          ;
C8E3 20 50 C8    197          jsr CSETUP
C8E6          198          ;
C8E6 20 26 CE    199          XINPUT jsr INVERT
C8E9 20 8D C9    200          jsr CXKEYIN
C8EC          201          ;
C8EC 8D 7B 06    202          sta XCHAR
C8EF          203          ;
C8EF 20 26 CE    204          jsr INVERT
C8F2          205          ;
C8F2 A8          206          tay
C8F3          207          ;
C8F3 AD FB 04    208          lda XMODE
C8F6 29 08       209          and #M.CTL
C8F8 F0 CB       210          beq BIORET
C8FA          211          ;
C8FA C0 8D       212          cpy #RETURN
C8FC D0 08       213          bne >6
C8FE          214          ;
C8FE AD FB 04    215          lda XMODE
C901 29 F7       216          and #NEGONE-M.CTL
C903 8D FB 04    217          sta XMODE
C906          218          ;
C906 C0 9B       219          ^6    cpy #ESCAPE          ; ESC check
C908 F0 11       220          beq >7
C90A          221          ;
C90A C0 95       222          cpy #RARROW
C90C D0 B7       223          bne BIORET
C90E          224          ;
C90E AC 7B 05    225          ldy OURCH
C911 20 44 CE    226          jsr PICK
C914          227          ;
C914 09 80       228          ora #MSBSET          ; set MSB
C916 8D 7B 06    229          sta XCHAR
C919 D0 AA       230          bne BIORET          ; always taken
C91B          231          ;
C91B          232          ;
C91B          233          ; Escape sequence start. Only if one of the following
C91B          234          ; characters is pressed is it executed, otherwise it is
C91B          235          ; ignored.
C91B          236          ;
C91B          237          ; @ - home and clear
C91B          238          ; E - clear to end of line
C91B          239          ; F - clear to end of screen
C91B          240          ; I - move cursor up
C91B          241          ; J - move cursor left
C91B          242          ; K - move cursor right
C91B          243          ; M - move cursor down

```

```

C91B      244 ; 4 - enter 40 column mode
C91B      245 ; 8 - enter 80 column mode
C91B      246 ;
C91B      247 ; ^D - disable printing of control characters
C91B      248 ; ^E - enable printing of control characters
C91B      249 ; ^Q - quit, like PR#0 or IN#0
C91B      250 ;
C91B 20 B1 CE 251 ^7 jsr ESCON
C91E 20 8D C9 252 jsr CXKEYIN
C921 20 C4 CE 253 jsr ESCOFF
C924 20 01 FD 254 jsr UPRCASE
C927      255 ;
C927 29 7F      256 and #MSBCLR ; clear MSB
C929      257 ;
C929 A0 10      258 ldy #ESCTABL-ESCCHAR+1
C92B      259 ;
C92B D9 7C C9 260 ^8 cmp ESCTABL,Y
C92E F0 05      261 beq >9
C930      262 ;
C930 88      263 dey
C931 10 F8      264 bpl <8
C933      265 ;
C933 30 0F      266 bmi >1 ; always taken
C935      267 ;
C935 B9 6B C9 268 ^9 lda ESCCHAR,Y
C938 29 7F      269 and #MSBCLR ; clear MSB
C93A      270 ;
C93A 20 D6 CA 271 jsr CTRLCHR
C93D      272 ;
C93D B9 6B C9 273 lda ESCCHAR,Y
C940 30 D9      274 bmi <7
C942      275 ;
C942 10 A2      276 bpl XINPUT
C944      277 ;
C944 A8      278 ^1 tay
C945      279 ;
C945 AD FB 04 280 lda XMODE
C948      281 ;
C948 C0 11      282 cpy #17 ; was it Quit (DC1)?
C94A D0 0B      283 bne >2
C94C      284 ;
C94C 20 4A CD 285 jsr XNAK
C94F      286 ;
C94F A9 98      287 lda #CTRLX ; fake a ^X
C951 8D 7B 06 288 sta XCHAR
C954      289 ;
C954 4C C5 C8 290 jmp BIORET
C957      291 ;
C957 C0 05      292 ^2 cpy #5 ; was it a ^E?
C959 D0 08      293 bne >5
C95B      294 ;
C95B 29 DF      295 and #NEGONE-M.CTL2 ; enable control characters
C95D      296 ;
C95D 8D FB 04 297 ^3 sta XMODE
C960      298 ;
C960 4C E6 C8 299 ^4 jmp XINPUT
C963      300 ;
C963 C0 04      301 ^5 cpy #4 ; was it a ^D?
C965 D0 F9      302 bne <4
C967      303 ;
C967 09 20      304 ora #M.CTL2 ; disable control characters

```

```

C969 D0 F2      305          bne <3          ; always taken
C96B           306          ;
C96B           307          ;
C96B           308          ; These control characters perform the escape functions
C96B           309          ; when they execute.  If the MSB is set, escape mode is
C96B           310          ; not exited after they execute their function.
C96B           311          ;
C96B 0C         312  ESCCHAR  hex 0C          ; @, formfeed
C96C 1C         313          hex 1C          ; A, FS
C96D 08         314          hex 08          ; B, BS
C96E 0A         315          hex 0A          ; C, LF
C96F 1F         316          hex 1F          ; D, US
C970 1D         317          hex 1D          ; E, GS
C971 0B         318          hex 0B          ; F, VT
C972 9F         319          hex 9F          ; I, US (stay ESC)
C973 88         320          hex 88          ; J, BS (stay ESC)
C974 9C         321          hex 9C          ; K, FS (stay ESC)
C975 8A         322          hex 8A          ; M, LF (stay ESC)
C976 11         323          hex 11          ; 4, DC1
C977 12         324          hex 12          ; 8, DC2
C978 88         325          hex 88          ; <-, BS (stay ESC)
C979 8A         326          hex 8A          ; DN, LF (stay ESC)
C97A 9F         327          hex 9F          ; UP, US (stay ESC)
C97B 9C         328          hex 9C          ; ->, FS (stay ESC)
C97C           329          ;
C97C 40         330  ESCTABL  asc '@'          ; handle old escapes
C97D 41         331          asc 'A'
C97E 42         332          asc 'B'
C97F 43         333          asc 'C'
C980 44         334          asc 'D'
C981 45         335          asc 'E'
C982 46         336          asc 'F'
C983 49         337          asc 'I'
C984 4A         338          asc 'J'
C985 4B         339          asc 'K'
C986 4D         340          asc 'M'
C987 34         341          asc '4'
C988 38         342          asc '8'
C989 08         343          byt LARROW^MSBSET ; left arrow
C98A 0A         344          byt DARROW^MSBSET ; down arrow
C98B 0B         345          byt UARROW^MSBSET ; up arrow
C98C 15         346          byt RARROW^MSBSET ; right arrow
C98D           347          ;
C98D           348          ;
C98D E6 4E      349  CXKEYIN  inc RNDL
C98F D0 02      350          bne >1
C991           351          ;
C991 E6 4F      352          inc RNDH
C993           353          ;
C993 AD 00 C0   354  ^1      lda KEY
C996 10 F5      355          bpl CXKEYIN
C998           356          ;
C998 2C 10 C0   357          bit CLRKEY
C99B           358          ;
C99B C9 FF      359          cmp #NEGONE
C99D D0 02      360          bne >2
C99F           361          ;
C99F A9 88      362          lda #LARROW
C9A1           363          ;
C9A1 60         364  ^2      rts
C9A2           365          ;

```

```

C9A2      366 ;
C9A2      367 ;           dfs 8,ZERO           ; 8 bytes
C9AA      368 ;
C9AA      369 ;
C9AA      370 ; Pascal 1.0 output entry point.  Must be at 0xC9AA.
C9AA      371 ;
C9AA AD 7B 06 372 PXWRITE   lda XCHAR
C9AD 4C 56 C3 373           jmp JPWRITE
C9B0      374 ;
C9B0      375 ;
C9B0 A9 83 376 PXINIT2   lda #M.PASCAL+M.PAS1.0+M.MOUSE
C9B2 D0 02 377           bne >1           ; always taken
C9B4      378 ;
C9B4      379 ;
C9B4 A9 81 380 PPINIT    lda #M.PASCAL+M.MOUSE
C9B6      381 ;
C9B6 48 382 ^1          pha
C9B7      383 ;
C9B7 20 90 CA 384           jsr TESTCARD
C9BA F0 04 385           beq >2
C9BC      386 ;
C9BC 68 387           pla
C9BD      388 ;
C9BD A2 09 389           ldx #9           ; 'NO DEVICE' error
C9BF      390 ;
C9BF 60 391           rts
C9C0      392 ;
C9C0 68 393 ^2          pla
C9C1 8D FB 04 394           sta XMODE
C9C4      395 ;
C9C4 8D 01 C0 396           sta STR80ON
C9C7 8D 0D C0 397           sta VID80ON
C9CA 8D 0F C0 398           sta ALTCHON
C9CD      399 ;
C9CD 20 D4 CE 400           jsr PSETUP
C9D0 20 74 CC 401           jsr XFF
C9D3      402 ;
C9D3 4C 1F CA 403           jmp DOBASL
C9D6      404 ;
C9D6      405 ;
C9D6      406 ; Character is always returned with its MSB off.
C9D6      407 ;
C9D6 20 D4 CE 408 PPREAD   jsr PSETUP
C9D9 20 8D C9 409           jsr CXKEYIN
C9DC      410 ;
C9DC 29 7F 411           and #MSBCLR           ; clear MSB
C9DE 8D 7B 06 412           sta XCHAR
C9E1      413 ;
C9E1 A2 00 414           ldx #ZERO
C9E3      415 ;
C9E3 AD FB 04 416           lda XMODE
C9E6 29 02 417           and #M.PAS1.0
C9E8 F0 02 418           beq >1
C9EA      419 ;
C9EA A2 C3 420           ldx /C3SPACE
C9EC      421 ;
C9EC AD 7B 06 422 ^1          lda XCHAR
C9EF      423 ;
C9EF 60 424           rts
C9F0      425 ;
C9F0      426 ;

```

```

C9F0 29 7F      427 PPWRITE and #MSBCLR      ; clear MSB
C9F2 AA        428 tax
C9F3          429 ;
C9F3 20 D4 CE   430 jsr PSETUP
C9F6          431 ;
C9F6 A9 08      432 lda #M.GOXY
C9F8          433 ;
C9F8 2C FB 04   434 bit XMODE
C9FB D0 32      435 bne >2
C9FD          436 ;
C9FD 8A        437 txa
C9FE          438 ;
C9FE 2C 2E CA   439 bit BITBYT60
CA01 F0 50      440 beq >3
CA03          441 ;
CA03 AC 7B 05   442 ldy OURCH
CA06          443 ;
CA06 24 32      444 bit INVFLG
CA08 10 02      445 bpl >1
CA0A          446 ;
CA0A 09 80      447 ora #MSBSET      ; set MSB
CA0C          448 ;
CA0C 20 70 CE   449 ^1 jsr STORIT
CA0F          450 ;
CA0F C8        451 iny
CA10 8C 7B 05   452 sty OURCH
CA13          453 ;
CA13 C4 21      454 cpy WNDWDTH
CA15 90 08      455 bcc DOBASL
CA17          456 ;
CA17 A9 00      457 lda #ZERO      ; carriage return
CA19 8D 7B 05   458 sta OURCH
CA1C          459 ;
CA1C 20 D8 CB   460 jsr XLF
CA1F          461 ;
CA1F          462 ;
CA1F A5 28      463 DOBASL lda BASL
CA21 8D 7B 07   464 sta OLDBASL
CA24          465 ;
CA24 A5 29      466 lda BASH
CA26 8D FB 07   467 sta OLDBASH
CA29          468 ;
CA29          469 ;
CA29 20 1F CE   470 PWRITER jsr PASINV
CA2C          471 ;
CA2C A2 00      472 ldx #ZERO      ; return, no error
CA2E          473 ;
CA2E 60        474 BITBYT60 rts
CA2F          475 ;
CA2F          476 ;
CA2F          477 ; Handle GOTOXY stuff.
CA2F          478 ;
CA2F 20 1F CE   479 ^2 jsr PASINV
CA32          480 ;
CA32 8A        481 txa
CA33          482 ;
CA33 38        483 sec
CA34          484 ;
CA34 E9 20      485 sbc #$20      ; make binary
CA36          486 ;
CA36 2C FB 06   487 bit XCOORD

```

```

CA39 30 30      488      bmi >5
CA3B           489      ;
CA3B 8D FB 05   490      sta OURCV
CA3E 85 25      491      sta CV
CA40           492      ;
CA40 20 BA CA   493      jsr XBASCALC
CA43           494      ;
CA43 AD FB 06   495      lda XCOORD
CA46 8D 7B 05   496      sta OURCH
CA49           497      ;
CA49 A9 F7      498      lda #NEGONE-M.GOXY
CA4B 2D FB 04   499      and XMODE
CA4E 8D FB 04   500      sta XMODE
CA51 D0 CC      501      bne DOBASL          ; always taken
CA53           502      ;
CA53 20 1F CE   503      ^3      jsr PASINV
CA56           504      ;
CA56 8A         505      txa
CA57           506      ;
CA57 C9 1E      507      cmp #$1E          ; GOTOXY command?
CA59 F0 06      508      beq >4
CA5B           509      ;
CA5B 20 D6 CA   510      jsr CTRLCHR
CA5E           511      ;
CA5E 4C 1F CA   512      jmp DOBASL
CA61           513      ;
CA61           514      ;
CA61           515      ; Start the GOTOXY sequence.
CA61           516      ;
CA61 A9 08      517      ^4      lda #M.GOXY
CA63 0D FB 04   518      ora XMODE
CA66 8D FB 04   519      sta XMODE
CA69           520      ;
CA69 A9 FF      521      lda #NEGONE
CA6B           522      ;
CA6B 8D FB 06   523      ^5      sta XCOORD
CA6E           524      ;
CA6E 4C 29 CA   525      jmp PWRITER
CA71           526      ;
CA71           527      ;
CA71           528      ; 65C02 opcode remapping to MNEML/MNEMR index.
CA71           529      ;
CA71           530      ; If the opcode is found in OPTBLC, then OPTBLL contains
CA71           531      ; the correct MNEML/MNEMR index.
CA71           532      ;
CA71 1A 3A 89   533      OPTBLC    hex 1A3A8903
CA74 03
CA75 5A 7A 14   534      hex 5A7A141C
CA78 1C
CA79 64 74 9C   535      hex 64749C9E
CA7C 9E
CA7D           536      ;
CA7D 37 36 21   537      OPTBLL    hex 37362140
CA80 40
CA81 41 42 43   538      hex 41424343
CA84 43
CA85 44 44 44   539      hex 44444444
CA88 44
CA89           540      ;
CA89           541      ;
CA89           542      ; Enable slot 3 and test for a ROM card.  If none, test

```



```

CA89          543 ; for an 80 column card.  If none, return with a BNE.
CA89          544 ;
CA89 20 B6 F8 545 TSTROMCD jsr TESTROM
CA8C D0 02    546 bne TESTCARD
CA8E          547 ;
CA8E C8       548 iny
CA8F          549 ;
CA8F 60       550 rts
CA90          551 ;
CA90          552 ;
CA90 AD 1C C0 553 TESTCARD lda RDPAGE2
CA93 0A       554 asl
CA94          555 ;
CA94 A9 88    556 lda #$88 ; test character
CA96          557 ;
CA96 2C 18 C0 558 bit RDSTR80
CA99          559 ;
CA99 8D 01 C0 560 sta STR80ON
CA9C          561 ;
CA9C 08       562 php ; save N and C flags
CA9D          563 ;
CA9D 8D 55 C0 564 sta PAGE2ON
CAA0          565 ;
CAA0 AC 00 04 566 ldy TEXTPG1
CAA3          567 ;
CAA3 8D 00 04 568 sta TEXTPG1
CAA6          569 ;
CAA6 AD 00 04 570 lda TEXTPG1
CAA9          571 ;
CAA9 8C 00 04 572 sty TEXTPG1
CAAC          573 ;
CAAC 28       574 plp ; recall status
CAAD B0 03    575 bcs >1
CAAF          576 ;
CAAF 8D 54 C0 577 sta PAGE1ON
CAB2          578 ;
CAB2 30 03    579 ^1 bmi >2
CAB4          580 ;
CAB4 8D 00 C0 581 sta STR80OFF
CAB7          582 ;
CAB7 C9 88    583 ^2 cmp #$88 ; same character?
CAB9          584 ;
CAB9 60       585 rts
CABA          586 ;
CABA          587 ;
CABA          588 ; XBASCALC is the same as the original BASCALC at 0xFBC1.
CABA          589 ;
CABA          590 ; 0 <= line number <= 23. A-reg = 000abcde. Generate
CABA          591 ; BASL = eabab000, BASH = 000001cd.
CABA          592 ;
CABA 48       593 XBASCALC pha
CABB          594 ;
CABB 4A       595 lsr
CABC 29 03    596 and #3
CABE          597 ;
CABE 09 04    598 ora #4
CAC0 85 29    599 sta BASH
CAC2          600 ;
CAC2 68       601 pla
CAC3 29 18    602 and #$18
CAC5          603 ;

```

```

CAC5 90 02      604      bcc >1
CAC7            605      ;
CAC7 69 7F      606      adc #MSBSET-1      ; carry adds 1
CAC9            607      ;
CAC9 85 28      608      ^1      sta BASL
CACB            609      ;
CACB 0A         610      asl
CACC 0A         611      asl
CACD            612      ;
CACD 05 28      613      ora BASL
CACF 85 28      614      sta BASL
CAD1            615      ;
CAD1 60         616      rts      ; carry must be clear
CAD2            617      ;
CAD2            618      ;
CAD2 2C 2E CA   619      CTRLCHR2 bit BITBYT60      ; used to set V-flag
CAD5            620      ;
CAD5 50 00      621      bvc *+2
CAD7            622      dfs !-1
CAD6            623      ;
CAD6 B8         624      CTRLCHR clv      ; clear V-flag, ignore M.CTL
CAD7            625      ;
CAD7 8D 7B 07   626      sta XTEMP1
CADA 48         627      pha
CADB            628      ;
CADB 98         629      tya
CADC 48         630      pha
CADD            631      ;
CADD AC 7B 07   632      ldy XTEMP1
CAE0            633      ;
CAE0 C0 05      634      cpy #5      ; is it NUL..EOT?
CAE2 90 13      635      bcc >1
CAE4            636      ;
CAE4 B9 B4 CB   637      lda CTRLADRH-5,Y
CAE7 F0 0E      638      beq >1      ; CTRL not implemented
CAE9            639      ;
CAE9 50 12      640      bvc >3      ; CTRLCHR always executes
CAEB 30 10      641      bmi >3      ; CR, BEL, LF, BS always done
CAED            642      ;
CAED 8D 7B 07   643      sta XTEMP1
CAF0            644      ;
CAF0 AD FB 04   645      lda XMODE
CAF3 29 28      646      and #M.CTL+M.CTL2
CAF5 F0 03      647      beq >2
CAF7            648      ;
CAF7 38         649      ^1      sec
CAF8 B0 09      650      bcs >4      ; always taken
CAFA            651      ;
CAFA AD 7B 07   652      ^2      lda XTEMP1
CAFD            653      ;
CAFD 09 80      654      ^3      ora #MSBSET      ; set MSB
CAFF            655      ;
CAFF 20 07 CB   656      jsr CTRLXFER
CB02            657      ;
CB02 18         658      clc
CB03            659      ;
CB03 68         660      ^4      pla
CB04 A8         661      tay
CB05            662      ;
CB05 68         663      pla
CB06            664      ;

```

```

CB06 60          665          rts
CB07          666          ;
CB07          667          ;
CB07          668          ; Now execute the subroutine.
CB07          669          ;
CB07 48          670 CTRLXFER pha
CB08          671          ;
CB08 B9 99 CB    672          lda CTRLADRL-5,Y
CB0B 48          673          pha
CB0C          674          ;
CB0C 60          675          rts
CB0D          676          ;
CB0D          677          ;
CB0D          678          ; Turn cursor on for Pascal only.
CB0D          679          ;
CB0D AD FB 04    680 PCURON   lda XMODE
CB10 10 05       681          bpl >2
CB12          682          ;
CB12 29 EF       683          and #$EF
CB14          684          ;
CB14 8D FB 04    685 ^1      sta XMODE
CB17          686          ;
CB17 60          687 ^2      rts
CB18          688          ;
CB18 AD FB 04    689 PCUROFF  lda XMODE
CB1B 10 FA       690          bpl <2
CB1D          691          ;
CB1D 09 10       692          ora #$10
CB1F D0 F3       693          bne <1          ; always taken
CB21          694          ;
CB21          695          ;
CB21          696          ; Removed XBELL and CXWAIT. The same code is at 0xFBDD.
CB21          697          ;
CB21          698          dfs 31,ZERO          ; 31 bytes
CB40          699          ;
CB40          700          ;
CB40          701          ; Execute backspace.
CB40          702          ;
CB40 CE 7B 05    703 XBS     dec OURCH
CB43 10 0B       704          bpl >1
CB45          705          ;
CB45 A5 21       706          lda WNDWDTH
CB47 8D 7B 05    707          sta OURCH
CB4A          708          ;
CB4A CE 7B 05    709          dec OURCH
CB4D          710          ;
CB4D 20 79 CB    711          jsr XUS
CB50          712          ;
CB50 60          713 ^1      rts
CB51          714          ;
CB51          715          ;
CB51          716          ; Execute carriage return.
CB51          717          ;
CB51 A9 00       718 XCR     lda #ZERO
CB53 8D 7B 05    719          sta OURCH
CB56          720          ;
CB56 AD FB 04    721          lda XMODE
CB59 30 03       722          bmi >2          ; Pascal, avoid auto LF
CB5B          723          ;
CB5B 20 D8 CB    724          jsr XLF
CB5E          725          ;

```

```

CB5E 60          726 ^2      rts
CB5F            727 ;
CB5F            728 ;
CB5F            729 ; Execute HOME.
CB5F            730 ;
CB5F A5 22      731 XEM      lda WNDTOP
CB61 85 25      732          sta CV
CB63            733 ;
CB63 A9 00      734          lda #ZERO
CB65 8D 7B 05   735          sta OURCH
CB68            736 ;
CB68 4C FB CD   737          jmp XVTAB2
CB6B            738 ;
CB6B            739 ;
CB6B            740 ; Execute forward space.
CB6B            741 ;
CB6B EE 7B 05   742 XFS      inc OURCH
CB6E            743 ;
CB6E AD 7B 05   744          lda OURCH
CB71 C5 21      745          cmp WNDWDTH
CB73 90 03      746          bcc >3
CB75            747 ;
CB75 20 51 CB   748          jsr XCR
CB78            749 ;
CB78 60          750 ^3      rts
CB79            751 ;
CB79            752 ;
CB79            753 ; Execute reverse linefeed.
CB79            754 ;
CB79 A5 22      755 XUS      lda WNDTOP
CB7B C5 25      756          cmp CV
CB7D B0 1E      757          bcs >7
CB7F            758 ;
CB7F C6 25      759          dec CV
CB81            760 ;
CB81 4C FB CD   761          jmp XVTAB2
CB84            762 ;
CB84            763 ;
CB84            764 ; Execute NORMAL video.
CB84            765 ;
CB84 AD FB 04   766 XSO      lda XMODE
CB87 10 02      767          bpl >4
CB89            768 ;
CB89 29 FB      769          and #NEGONE-M.VMODE ; set NORMAL
CB8B            770 ;
CB8B A0 FF      771 ^4      ldy #NEGONE
CB8D D0 09      772          bne >6                ; always taken
CB8F            773 ;
CB8F            774 ;
CB8F            775 ; Execute INVERSE video.
CB8F            776 ;
CB8F AD FB 04   777 XSI      lda XMODE
CB92 10 02      778          bpl >5
CB94            779 ;
CB94 09 04      780          ora #M.VMODE                ; set INVERSE
CB96            781 ;
CB96 A0 7F      782 ^5      ldy #INVRSE80
CB98            783 ;
CB98 8D FB 04   784 ^6      sta XMODE
CB9B            785 ;
CB9B 84 32      786          sty INVFLG

```

```

CB9D          787 ;
CB9D 60       788 ^7      rts
CB9E          789 ;
CB9E          790 ;
CB9E 0C       791 CTRLADRL byt PCURON-1      ; ENQ
CB9F 17       792          byt PCUROFF-1     ; ACK
CBA0 DC       793          byt RINGBELL-1 ; BEL
CBA1 3F       794          byt XBS-1      ; BS
CBA2 00       795          byt ZERO      ; HT, not imlemented
CBA3 D7       796          byt XLF-1     ; LF
CBA4 76       797          byt XVT-1     ; VT
CBA5 73       798          byt XFF-1     ; FF
CBA6 50       799          byt XCR-1     ; CR
CBA7 83       800          byt XSO-1     ; SO
CBA8 8E       801          byt XSI-1     ; SI
CBA9 00       802          byt ZERO      ; DLE, not implemented
CBAA E6       803          byt XDC1-1    ; DC1
CBAB F8       804          byt XDC2-1    ; DC2
CBAC 00       805          byt ZERO      ; DC3, not implemented
CBAD 00       806          byt ZERO      ; DC4, not implemented
CBAE 49       807          byt XNAK-1    ; NAK
CBAF D3       808          byt SCROLLDN-1 ; SYN
CBB0 EA       809          byt SCROLLUP-1 ; ETB
CBB1 39       810          byt MOUSEOFF-1
CBB2 5E       811          byt XEM-1     ; EM
CBB3 92       812          byt XSUB-1    ; SUB
CBB4 40       813          byt MOUSEON-1
CBB5 6A       814          byt XFS-1     ; FS
CBB6 96       815          byt XGS-1     ; GS
CBB7 00       816          byt ZERO      ; RS, not implemented
CBB8 78       817          byt XUS-1     ; US
CBB9          818 ;
CBB9 4B       819 CTRLADRH hby PCURON-$8001 ; ENQ
CBBA 4B       820          hby PCUROFF-$8001 ; ACK
CBBB FB       821          hby RINGBELL-1  ; BEL
CBBC CB       822          hby XBS-1      ; BS
CBBD 00       823          byt ZERO      ; HT, not imlemented
CBBE CB       824          hby XLF-1     ; LF
CBBF 4C       825          hby XVT-$8001 ; VT
CBC0 4C       826          hby XFF-$8001 ; FF
CBC1 CB       827          hby XCR-1     ; CR
CBC2 4B       828          hby XSO-$8001 ; SO
CBC3 4B       829          hby XSI-$8001 ; SI
CBC4 00       830          byt ZERO      ; DLE, not implemented
CBC5 4C       831          hby XDC1-$8001 ; DC1
CBC6 4C       832          hby XDC2-$8001 ; DC2
CBC7 00       833          byt ZERO      ; DC3, not implemented
CBC8 00       834          byt ZERO      ; DC4, not implemented
CBC9 4D       835          hby XNAK-$8001 ; NAK
CBCA 4B       836          hby SCROLLDN-$8001 ; SYN
CBCB 4B       837          hby SCROLLUP-$8001 ; ETB
CBCC 4D       838          hby MOUSEOFF-$8001
CBCD 4B       839          hby XEM-$8001 ; EM
CBCE 4C       840          hby XSUB-$8001 ; SUB
CBCF 4D       841          hby MOUSEON-$8001
CBD0 4B       842          hby XFS-$8001 ; FS
CBD1 4C       843          hby XGS-$8001 ; GS
CBD2 00       844          byt ZERO      ; RS, not implemented
CBD3 4B       845          hby XUS-$8001 ; US
CBD4          846 ;
CBD4          847 ;

```

CBD4 848 icl "CC.L"

LLOAD CC.L,A\$4000

```

CBD4      1          ttl "ROM Source Code, CC.L"
CBD4      2      ;
CBD4      3      ;
CBD4      4      ; CC.L
CBD4      5      ;
CBD4      6      ;
CBD4      7      ; SCROLLDN and SCROLLUP scroll the screen down and up,
CBD4      8      ; respectively, depending on the value of Y-reg. It
CBD4      9      ; scrolls within windows having even or odd edges for both
CBD4     10      ; 40 and 80 columns down to 1 characters wide.
CBD4     11      ;
CBD4 A0 00     12 SCROLLDN ldy #ZERO          ; scroll down
CBD6 F0 15     13          beq >2          ; always taken
CBD8     14      ;
CBD8     15      ;
CBD8     16      ; Execute linefeed.
CBD8     17      ;
CBD8 E6 25     18 XLF          inc CV
CBDA     19      ;
CBDA A5 25     20          lda CV
CBDC 8D FB 05  21          sta OURCV
CBDF     22      ;
CBDF C5 23     23          cmp WNDBTM
CBE1 B0 03     24          bcs >1
CBE3     25      ;
CBE3 4C 00 CE  26          jmp XVTAB3
CBE6     27      ;
CBE6 CE FB 05  28 ^1          dec OURCV
CBE9     29      ;
CBE9 C6 25     30          dec CV
CBEB     31      ;
CBEB     32      ;
CBEB A0 01     33 SCROLLUP ldy #1          ; scroll up
CBED     34      ;
CBED 8A        35 ^2          txa
CBEE 48        36          pha
CBEF     37      ;
CBEF 8C 7B 07  38          sty XTEMP1
CBF2     39      ;
CBF2 A5 21     40          lda WNDWDTH
CBF4 48        41          pha
CBF5     42      ;
CBF5 2C 1F C0  43          bit RDVID80
CBF8 10 1C     44          bpl >5
CBFA     45      ;
CBFA 8D 01 C0  46          sta STR80ON
CBFD 4A        47          lsr
CBFE AA        48          tax
CBFF     49      ;
CBFF A5 20     50          lda WNDLFT
CC01 4A        51          lsr
CC02     52      ;
CC02 B8        53          clv          ; left edge even
CC03     54      ;
CC03 90 03     55          bcc >3
CC05     56      ;
CC05 2C 2E CA  57          bit BITBYT60      ; left edge odd
CC08     58      ;
CC08 2A        59 ^3          rol
CC09 45 21     60          eor WNDWDTH

```

```

CC0B 4A          61      lsr
CC0C 70 03       62      bvs >4
CC0E             63      ;
CC0E B0 01       64      bcs >4
CC10             65      ;
CC10 CA          66      dex
CC11             67      ;
CC11 86 21       68      ^4      stx WNDWDTH
CC13             69      ;
CC13 AD 1F C0    70      lda RDVID80
CC16             71      ;
CC16 08          72      ^5      php                ; save N, Z, V flags
CC17             73      ;
CC17 A6 22       74      ldx WNDTOP
CC19             75      ;
CC19 98          76      tya
CC1A D0 03       77      bne >6                ; direction
CC1C             78      ;
CC1C A6 23       79      ldx WNDBTM
CC1E CA          80      dex
CC1F             81      ;
CC1F 8A          82      ^6      txa
CC20             83      ;
CC20 20 00 CE    84      jsr XVTAB3
CC23             85      ;
CC23 A5 28       86      ^7      lda BASL
CC25 85 2A       87      sta BAS2L
CC27             88      ;
CC27 A5 29       89      lda BASH
CC29 85 2B       90      sta BAS2H
CC2B             91      ;
CC2B AD 7B 07    92      lda XTEMP1
CC2E F0 32       93      beq >5
CC30             94      ;
CC30 E8          95      inx
CC31             96      ;
CC31 E4 23       97      cpx WNDBTM
CC33 B0 32       98      bcs >6
CC35             99      ;
CC35 8A          100     ^8      txa
CC36             101     ;
CC36 20 00 CE    102     jsr XVTAB3
CC39             103     ;
CC39 A4 21       104     ldy WNDWDTH
CC3B             105     ;
CC3B 28          106     plp                ; recall status
CC3C 08          107     php
CC3D 10 1E       108     bpl >4                ; only do 40 columns
CC3F             109     ;
CC3F AD 55 C0    110     lda PAGE2ON
CC42             111     ;
CC42 98          112     tya
CC43 F0 07       113     beq >1
CC45             114     ;
CC45 B1 28       115     ^9      lda (BASL),Y        ; scroll even bytes
CC47 91 2A       116     sta (BAS2L),Y
CC49             117     ;
CC49 88          118     dey
CC4A D0 F9       119     bne <9
CC4C             120     ;
CC4C 70 04       121     ^1      bvs >2                ; if odd, skip

```



```

CC4E      122 ;
CC4E B1 28 123      lda (BASL),Y
CC50 91 2A 124      sta (BAS2L),Y
CC52      125 ;
CC52 AD 54 C0 126 ^2      lda PAGE1ON
CC55      127 ;
CC55 A4 21 128      ldy WNDWDTH
CC57      129 ;
CC57 B0 04 130      bcs >4
CC59      131 ;
CC59 B1 28 132 ^3      lda (BASL),Y      ; scroll odd bytes
CC5B 91 2A 133      sta (BAS2L),Y
CC5D      134 ;
CC5D 88 135 ^4      dey
CC5E 10 F9 136      bpl <3
CC60      137 ;
CC60 30 C1 138      bmi <7      ; always taken
CC62      139 ;
CC62 CA 140 ^5      dex
CC63      141 ;
CC63 E4 22 142      cpx WNDTOP
CC65 10 CE 143      bpl <8
CC67      144 ;
CC67 28 145 ^6      plp      ; recall status
CC68      146 ;
CC68 68 147      pla
CC69 85 21 148      sta WNDWDTH
CC6B      149 ;
CC6B 20 93 CC 150      jsr XSUB
CC6E 20 FB CD 151      jsr XVTAB2
CC71      152 ;
CC71 68 153      pla
CC72 AA 154      tax
CC73      155 ;
CC73 60 156      rts
CC74      157 ;
CC74      158 ;
CC74      159 ; Execute clear (moved here).
CC74      160 ;
CC74 20 5F CB 161 XFF      jsr XEM
CC77      162 ;
CC77      163 ;      jmp XVT
CC77      164 ;
CC77      165 ;
CC77      166 ; Execute CLR to EOS.
CC77      167 ;
CC77 20 97 CC 168 XVT      jsr XGS
CC7A      169 ;
CC7A A5 25 170      lda CV
CC7C 48 171      pha
CC7D 10 06 172      bpl >2
CC7F      173 ;
CC7F 20 00 CE 174 ^1      jsr XVTAB3
CC82 20 93 CC 175      jsr XSUB
CC85      176 ;
CC85 E6 25 177 ^2      inc CV
CC87      178 ;
CC87 A5 25 179      lda CV
CC89 C5 23 180      cmp WNDBTM
CC8B 90 F2 181      bcc <1
CC8D      182 ;

```

```

CC8D 68          183          pla
CC8E 85 25       184          sta CV
CC90             185          ;
CC90 4C FB CD    186          jmp XVTAB2
CC93             187          ;
CC93             188          ;
CC93             189          ; Execute clear line.
CC93             190          ;
CC93 A0 00       191 XSUB      ldy #ZERO
CC95 F0 03       192          beq XGSEOLZ          ; always taken
CC97             193          ;
CC97             194          ;
CC97             195          ; Execute clear to EOL.
CC97             196          ;
CC97 AC 7B 05    197 XGS       ldy OURCH
CC9A             198          ;
CC9A             199          ;
CC9A A5 32       200 XGSEOLZ   lda INVFLG
CC9C 29 80       201          and #MSBSET
CC9E 09 20       202          ora #' '          ; make it a blank
CCA0             203          ;
CCA0 2C 1F C0    204          bit RDVID80
CCA3 30 15       205          bmi CLR80
CCA5             206          ;
CCA5             207          ;
CCA5             208          ; Clear to end of line for 40 colums.
CCA5             209          ;
CCA5 91 28       210 CLR40     sta (BASL),Y
CCA7             211          ;
CCA7 C8          212          iny
CCA8             213          ;
CCA8 C4 21       214          cpy WNDWDTH
CCAA 90 F9       215          bcc CLR40
CCAC             216          ;
CCAC 60          217          rts
CCAD             218          ;
CCAD             219          ;
CCAD 86 2A       220 CLRHALF   stx BAS2L
CCAF             221          ;
CCAF A2 D8       222          ldx #!-40
CCB1 A0 14       223          ldy #20
CCB3             224          ;
CCB3 A5 32       225          lda INVFLG
CCB5 29 A0       226          and #" "
CCB7             227          ;
CCB7 4C D2 CC    228          jmp CLR2
CCBA             229          ;
CCBA             230          ;
CCBA             231          ; Clear to end of line for 80 columns.
CCBA             232          ;
CCBA 86 2A       233 CLR80     stx BAS2L
CCBC             234          ;
CCBC 48          235          pha
CCBD             236          ;
CCBD 98          237          tya
CCBE 48          238          pha
CCBF             239          ;
CCBF 38          240          sec
CCC0             241          ;
CCC0 E5 21       242          sbc WNDWDTH          ; count = WNDWDTH - Y-reg - 1
CCC2 AA          243          tax

```

```

CCC3      244 ;
CCC3 98    245 tya
CCC4 4A    246 lsr
CCC5 A8    247 tay
CCC6      248 ;
CCC6 68    249 pla
CCC7 45 20 250 eor WNDLFT ; get starting page
CCC9 6A    251 ror
CCCA B0 03 252 bcs >4
CCCC      253 ;
CCCC 10 01 254 bpl >4
CCCE      255 ;
CCCE C8    256 iny
CCCF      257 ;
CCCF 68    258 ^4 pla
CCD0      259 ;
CCD0 B0 0B 260 bcs >5
CCD2      261 ;
CCD2      262 ;
CCD2 2C 55 C0 263 CLR2 bit PAGE2ON
CCD5      264 ;
CCD5 91 28 265 sta (BASL),Y
CCD7      266 ;
CCD7 2C 54 C0 267 bit PAGE1ON
CCDA      268 ;
CCDA E8    269 inx
CCDB F0 06 270 beq >6
CCDD      271 ;
CCDD 91 28 272 ^5 sta (BASL),Y
CCDF      273 ;
CCDF C8    274 iny
CCE0      275 ;
CCE0 E8    276 inx
CCE1 D0 EF 277 bne CLR2
CCE3      278 ;
CCE3 A6 2A 279 ^6 ldx BAS2L
CCE5      280 ;
CCE5 38    281 sec
CCE6      282 ;
CCE6 60    283 rts
CCE7      284 ;
CCE7      285 ;
CCE7      286 ; Execute 40 column mode.
CCE7      287 ;
CCE7 AD FB 04 288 XDC1 lda XMODE
CCEA 30 4D 289 bmi >4
CCEC      290 ;
CCEC 20 2E CD 291 XDC1.2 jsr SETTOP
CCEF      292 ;
CCEF 2C 1F C0 293 bit RDVID80
CCF2 10 12 294 bpl >1
CCF4      295 ;
CCF4 20 8E CD 296 jsr SCR84
CCF7 90 0D 297 bcc >1
CCF9      298 ;
CCF9      299 ;
CCF9      300 ; Execute 80 column mode.
CCF9      301 ;
CCF9 20 90 CA 302 XDC2 jsr TESTCARD
CCFC D0 3B 303 bne >4
CCFE      304 ;

```

```

CCFE 2C 1F C0 305 bit RDVID80
CD01 30 03 306 bmi >1
CD03 307 ;
CD03 20 C1 CD 308 jsr SCR48 ; make it 80 column mode
CD06 309 ;
CD06 18 310 ^1 clc
CD07 311 ;
CD07 AD 7B 05 312 lda OURCH
CD0A 65 20 313 adc WNDLFT
CD0C 314 ;
CD0C 2C 1F C0 315 bit RDVID80
CD0F 30 06 316 bmi >2
CD11 317 ;
CD11 C9 28 318 cmp #40 ; in 40 column mode
CD13 90 02 319 bcc >2
CD15 320 ;
CD15 A9 27 321 lda #39
CD17 322 ;
CD17 8D 7B 05 323 ^2 sta OURCH
CD1A 85 24 324 sta CH
CD1C 325 ;
CD1C A5 25 326 lda CV
CD1E 20 BA CA 327 jsr XBASCALC
CD21 328 ;
CD21 2C 1F C0 329 bit RDVID80
CD24 10 05 330 bpl DO40
CD26 331 ;
CD26 20 6E CD 332 jsr FULL80
CD29 F0 03 333 beq SETTOP ; always taken
CD2B 334 ;
CD2B 335 ;
CD2B 20 6A CD 336 DO40 jsr FULL40
CD2E 337 ;
CD2E 338 ;
CD2E A9 00 339 SETTOP lda #ZERO
CD30 340 ;
CD30 2C 1A C0 341 bit RDTEXT ; mixed?
CD33 30 02 342 bmi >3
CD35 343 ;
CD35 A9 14 344 lda #20
CD37 345 ;
CD37 85 22 346 ^3 sta WNDTOP
CD39 347 ;
CD39 60 348 ^4 rts
CD3A 349 ;
CD3A 350 ;
CD3A 351 ; Execute mouse text OFF and ON.
CD3A 352 ;
CD3A AD FB 04 353 MOUSEOFF lda XMODE
CD3D 09 01 354 ora #M.MOUSE ; set mouse bit
CD3F D0 05 355 bne >5 ; always taken
CD41 356 ;
CD41 357 ;
CD41 AD FB 04 358 MOUSEON lda XMODE
CD44 29 FE 359 and #NEGONE-M.MOUSE ; clear mouse bit
CD46 360 ;
CD46 8D FB 04 361 ^5 sta XMODE
CD49 362 ;
CD49 60 363 rts
CD4A 364 ;
CD4A 365 ;

```

```

CD4A      366 ; Execute Quit.
CD4A      367 ;
CD4A AD FB 04 368 XNAK      lda XMODE                ; only valid in BASIC
CD4D 30 1A      369                bmi >6                ; ignore if Pascal
CD4F      370 ;
CD4F 20 2B CD 371                jsr DO40
CD52 20 7D CD 372                jsr QUIT
CD55 20 61 CD 373                jsr SETCOUT2
CD58      374 ;
CD58      375 ;
CD58 A9 FD      376 SETKEYIN  lda /KEYIN
CD5A 85 39      377                sta KSWH
CD5C      378 ;
CD5C A9 1B      379                lda #KEYIN
CD5E 85 38      380                sta KSWL
CD60      381 ;
CD60 60      382                rts
CD61      383 ;
CD61      384 ;
CD61 A9 FD      385 SETCOUT2  lda /COUT2
CD63 85 37      386                sta CSWH
CD65      387 ;
CD65 A9 F0      388                lda #COUT2
CD67 85 36      389                sta CSWL
CD69      390 ;
CD69 60      391 ^6          rts
CD6A      392 ;
CD6A      393 ;
CD6A      394 ; Set full 40 column window.
CD6A      395 ;
CD6A A9 28      396 FULL40    lda #40
CD6C D0 02      397                bne >7                ; always taken
CD6E      398 ;
CD6E      399 ;
CD6E      400 ; Set full 80 column window.
CD6E      401 ;
CD6E A9 50      402 FULL80    lda #80
CD70      403 ;
CD70 85 21      404 ^7          sta WNDWDTH
CD72      405 ;
CD72 A9 18      406                lda #24
CD74 85 23      407                sta WNDBTM
CD76      408 ;
CD76 A9 00      409                lda #ZERO
CD78 85 22      410                sta WNDDTOP
CD7A 85 20      411                sta WNDLFT
CD7C      412 ;
CD7C 60      413                rts
CD7D      414 ;
CD7D      415 ;
CD7D      416 ; QUIT as used by PR#0 to turn off everything.
CD7D      417 ;
CD7D 2C 1F C0 418 QUIT      bit RDVID80
CD80 10 03      419                bpl >8
CD82      420 ;
CD82 20 EC CC 421                jsr XDC1.2
CD85      422 ;
CD85 8D 0E C0 423 ^8          sta ALTCHOFF
CD88      424 ;
CD88 A9 FF      425                lda #NEGONE                ; destroy mode
CD8A 8D FB 04 426                sta XMODE

```

```

CD8D          427 ;
CD8D 60        428           rts
CD8E          429 ;
CD8E          430 ;
CD8E          431 ; SCRN84 and SCRN48 convert screens between 40 and 80
CD8E          432 ; columns. WNDTOP must be set up to indicate the last
CD8E          433 ; line to process.
CD8E          434 ;
CD8E 8A        435 SCRN84   txa
CD8F 48        436           pha
CD90          437 ;
CD90 A2 17     438           ldx #23           ; start at bottom
CD92 8D 01 C0  439           sta STR80ON
CD95          440 ;
CD95 8A        441 ^1       txa
CD96          442 ;
CD96 20 BA CA  443           jsr XBASCALC
CD99          444 ;
CD99 A0 27     445           ldy #39           ; start at far right
CD9B          446 ;
CD9B 84 2A     447 ^2       sty BAS2L
CD9D          448 ;
CD9D 98        449           tya
CD9E          450 ;
CD9E 4A        451           lsr
CD9F B0 03     452           bcs >3
CDA1          453 ;
CDA1 2C 55 C0  454           bit PAGE2ON
CDA4          455 ;
CDA4 A8        456 ^3       tay           ; 80 column index
CDA5          457 ;
CDA5 B1 28     458           lda (BASL),Y
CDA7          459 ;
CDA7 2C 54 C0  460           bit PAGE1ON
CDAA          461 ;
CDAA A4 2A     462           ldy BAS2L           ; 40 column index
CDAC          463 ;
CDAC 91 28     464           sta (BASL),Y
CDAE          465 ;
CDAE 88        466           dey
CDAF 10 EA     467           bpl <2
CDB1          468 ;
CDB1 CA        469           dex
CDB2 30 04     470           bmi >4
CDB4          471 ;
CDB4 E4 22     472           cpx WNDTOP
CDB6 B0 DD     473           bcs <1
CDB8          474 ;
CDB8 8D 00 C0  475 ^4       sta STR80OFF       ; for 40 columns
CDBB 8D 0C C0  476           sta VID80OFF       ; for 40 columns
CDBE          477 ;
CDBE 4C F5 CD  478           jmp SCRNRET
CDC1          479 ;
CDC1          480 ;
CDC1 8A        481 SCRN48   txa
CDC2 48        482           pha
CDC3          483 ;
CDC3 A2 17     484           ldx #23           ; start at bottom
CDC5          485 ;
CDC5 8A        486 ^1       txa
CDC6          487 ;

```

```

CDC6 20 BA CA 488      jsr XBASCALC
CDC9          489      ;
CDC9 A0 00     490      ldy #ZERO          ; start at far left
CDCB          491      ;
CDCB 8D 01 C0  492      sta STR80ON
CDCE          493      ;
CDCE B1 28     494      ^2      lda (BASL),Y
CDD0          495      ;
CDD0 84 2A     496      sty BAS2L          ; save 40 column index
CDD2          497      ;
CDD2 48        498      pha
CDD3          499      ;
CDD3 98        500      tya
CDD4          501      ;
CDD4 4A        502      lsr
CDD5 B0 03     503      bcs >3
CDD7          504      ;
CDD7 8D 55 C0  505      sta PAGE2ON
CDDA          506      ;
CDDA A8        507      ^3      tay          ; get 80 column index
Cddb          508      ;
Cddb 68        509      pla
CDDC 91 28     510      sta (BASL),Y
CDDE          511      ;
CDDE 8D 54 C0  512      sta PAGE1ON
CDE1          513      ;
CDE1 A4 2A     514      ldy BAS2L
CDE3          515      ;
CDE3 C8        516      iny
CDE4          517      ;
CDE4 C0 28     518      cpy #40
CDE6 90 E6     519      bcc <2
CDE8          520      ;
CDE8 20 AD CC  521      jsr CLRHALF
CDEB          522      ;
CDEB CA        523      dex
CDEC 30 04     524      bmi >4
CDEE          525      ;
CDEE E4 22     526      cpx WNDTOP
CDF0 B0 D3     527      bcs <1
CDF2          528      ;
CDF2 8D 0D C0  529      ^4      sta VID80ON
CDF5          530      ;
CDF5 20 FB CD  531      SCRNET  jsr XVTAB2
CDF8          532      ;
CDF8 68        533      pla
CDF9 AA        534      tax
CDFA          535      ;
CDFA 60        536      rts
CDFB          537      ;
CDFB          538      ;
CDFB A5 25     539      XVTAB2  lda CV
CDFD 8D FB 05  540      sta OURCV
CE00          541      ;
CE00 20 BA CA  542      XVTAB3  jsr XBASCALC
CE03          543      ;
CE03 A5 20     544      lda WNDLFT
CE05          545      ;
CE05 2C 1F C0  546      bit RDVID80
CE08 10 01     547      bpl >1
CE0A          548      ;

```

```

CE0A 4A          549          lsr
CE0B             550          ;
CE0B 18          551          ^1      clc
CE0C             552          ;
CE0C 65 28       553          adc BASL
CE0E 85 28       554          sta BASL
CE10             555          ;
CE10 60          556          rts
CE11             557          ;
CE11             558          ;
CE11 A4 24       559          XRDKEY2  ldy CH
CE13             560          ;
CE13 B1 28       561          lda (BASL),Y
CE15             562          ;
CE15 2C 1F C0    563          bit RDVID80
CE18 10 0C       564          bpl INVERT
CE1A             565          ;
CE1A 60          566          rts
CE1B             567          ;
CE1B             568          ;
CE1B             569          dfs 4,ZERO          ; 4 bytes
CE1F             570          ;
CE1F             571          ;
CE1F             572          ; Must be at 0xCE1F.
CE1F             573          ;
CE1F AD FB 04    574          PASINV  lda XMODE
CE22 29 10       575          and #M.CURSOR
CE24 D0 11       576          bne >1
CE26             577          ;
CE26 48          578          INVERT  pha
CE27             579          ;
CE27 98          580          tya
CE28 48          581          pha
CE29             582          ;
CE29 AC 7B 05    583          ldy OURCH
CE2C             584          ;
CE2C 20 44 CE    585          jsr PICK
CE2F             586          ;
CE2F 49 80       587          eor #MSBSET          ; flip INVERSE/NORMAL
CE31             588          ;
CE31 20 70 CE    589          jsr STORIT
CE34             590          ;
CE34 68          591          pla
CE35 A8          592          tay
CE36             593          ;
CE36 68          594          pla
CE37             595          ;
CE37 60          596          ^1      rts
CE38             597          ;
CE38             598          ;
CE38             599          ; Store a character onto the screen.
CE38             600          ;
CE38 48          601          STORCHAR pha
CE39             602          ;
CE39 24 32       603          bit INVFLG
CE3B 30 02       604          bmi >1
CE3D             605          ;
CE3D 29 7F       606          and #MSBCLR          ; clear MSB
CE3F             607          ;
CE3F 20 70 CE    608          ^1      jsr STORIT
CE42             609          ;

```



```

CE42 68          610          pla
CE43          611          ;
CE43 60          612          rts
CE44          613          ;
CE44          614          ;
CE44          615          ; Get a character from the screen.
CE44          616          ;
CE44 B1 28       617 PICK      lda (BASL),Y
CE46          618          ;
CE46 2C 1F C0    619          bit RDVID80
CE49 10 19       620          bpl >2
CE4B          621          ;
CE4B 8D 01 C0    622          sta STR80ON
CE4E          623          ;
CE4E 84 2A       624          sty BAS2L
CE50 98          625          tya
CE51 45 20       626          eor WNDLFT          ; get starting page
CE53          627          ;
CE53 6A          628          ror
CE54 B0 04       629          bcs >1
CE56          630          ;
CE56 AD 55 C0    631          lda PAGE2ON
CE59          632          ;
CE59 C8          633          iny
CE5A          634          ;
CE5A 98          635 ^1      tya
CE5B 4A          636          lsr
CE5C A8          637          tay
CE5D          638          ;
CE5D B1 28       639          lda (BASL),Y
CE5F          640          ;
CE5F 2C 54 C0    641          bit PAGE1ON
CE62          642          ;
CE62 A4 2A       643          ldy BAS2L
CE64          644          ;
CE64          645          ;
CE64          646          ; Only allow mouse text if alternate character set enabled.
CE64          647          ;
CE64 2C 1E C0    648 ^2      bit RDALTCH
CE67 10 06       649          bpl >3
CE69          650          ;
CE69 C9 20       651          cmp #$20
CE6B B0 02       652          bcs >3
CE6D          653          ;
CE6D 09 40       654          ora #$40
CE6F          655          ;
CE6F 60          656 ^3      rts
CE70          657          ;
CE70          658          ;
CE70          659          ; Store character.
CE70          660          ;
CE70 48          661 STORIT    pha
CE71 29 FF       662          and #NEGONE
CE73 30 16       663          bmi >1
CE75          664          ;
CE75 AD FB 04    665          lda XMODE
CE78 6A          666          ror
CE79          667          ;
CE79 68          668          pla
CE7A 48          669          pha
CE7B          670          ;

```

```

CE7B 90 0E      671      bcc >1
CE7D            672      ;
CE7D            673      ;
CE7D            674      ; Only process mouse text if alternate character set is
CE7D            675      ; enabled.
CE7D            676      ;
CE7D 2C 1E C0    677      bit RDALTCH
CE80 10 09      678      bpl >1
CE82            679      ;
CE82 49 40      680      eor #$40
CE84            681      ;
CE84 2C 2E CA    682      bit BITBYT60
CE87 F0 02      683      beq >1
CE89            684      ;
CE89 49 40      685      eor #$40
CE8B            686      ;
CE8B 2C 1F C0    687      ^1 bit RDVID80
CE8E 10 1D      688      bpl >3
CE90            689      ;
CE90 8D 01 C0    690      sta STR80ON
CE93 48          691      pha
CE94            692      ;
CE94 84 2A      693      sty BAS2L          ; temp storage
CE96 98          694      tya
CE97 45 20      695      eor WNDLFT        ; get starting page
CE99            696      ;
CE99 4A          697      lsr
CE9A B0 04      698      bcs >2
CE9C            699      ;
CE9C AD 55 C0    700      lda PAGE2ON
CE9F            701      ;
CE9F C8          702      iny
CEA0            703      ;
CEA0 98          704      ^2 tya
CEA1 4A          705      lsr
CEA2 A8          706      tay
CEA3            707      ;
CEA3 68          708      pla
CEA4 91 28      709      sta (BASL),Y
CEA6            710      ;
CEA6 AD 54 C0    711      lda PAGE1ON
CEA9            712      ;
CEA9 A4 2A      713      ldY BAS2L
CEAB            714      ;
CEAB 68          715      pla
CEAC            716      ;
CEAC 60          717      rts
CEAD            718      ;
CEAD            719      ;
CEAD            720      ; Quick 40 column store.
CEAD            721      ;
CEAD 91 28      722      ^3 sta (BASL),Y
CEAF            723      ;
CEAF 68          724      pla
CEB0            725      ;
CEB0 60          726      rts
CEB1            727      ;
CEB1            728      ;
CEB1            729      ; Turn ON Escape cursor.
CEB1            730      ;
CEB1 48          731      ESCON pha

```

```

CEB2          732 ;
CEB2 98       733      tya
CEB3 48       734      pha
CEB4          735 ;
CEB4 AC 7B 05 736      ldy OURCH
CEB7 20 44 CE 737      jsr PICK
CEBA          738 ;
CEBA 8D 7B 06 739      sta XCHAR
CEBD          740 ;
CEBD 29 80    741      and #MSBSET      ; save NORMAL/INVERSE bit
CEBF 49 AB    742      eor #"+"      ; make it inverse
CEC1          743 ;
CEC1 4C CD CE 744      jmp ESCRTN
CEC4          745 ;
CEC4          746 ;
CEC4          747 ; Turn off Escape cursor.
CEC4          748 ;
CEC4 48       749 ESCOFF pha
CEC5          750 ;
CEC5 98       751      tya
CEC6 48       752      pha
CEC7          753 ;
CEC7 AC 7B 05 754      ldy OURCH
CECA AD 7B 06 755      lda XCHAR
CECD          756 ;
CECD          757 ;
CECD          758 ; Return for ESCON and ESCOFF.
CECD          759 ;
CECD 20 70 CE 760 ESCRTN jsr STORIT
CED0          761 ;
CED0 68       762      pla
CED1 A8       763      tay
CED2          764 ;
CED2 68       765      pla
CED3          766 ;
CED3 60       767      rts
CED4          768 ;
CED4          769 ;
CED4          770 ; Set up page zero for Pascal.
CED4          771 ;
CED4 20 6E CD 772 PSETUP jsr FULL80
CED7          773 ;
CED7 A9 FF    774      lda #NEGONE
CED9 85 32    775      sta INVFLG
CEDB          776 ;
CEDB AD FB 04 777      lda XMODE
CEDE 29 04    778      and #M.VMODE
CEE0 F0 02    779      beq >1
CEE2          780 ;
CEE2 46 32    781      lsr INVFLG      ; make it INVERSE
CEE4          782 ;
CEE4 AD 7B 07 783 ^1      lda OLDBASL
CEE7 85 28    784      sta BASL
CEE9          785 ;
CEE9 AD FB 07 786      lda OLDBASH
CEEC 85 29    787      sta BASH
EEEE          788 ;
EEEE AD FB 05 789      lda OURCV
CEF1 85 25    790      sta CV
CEF3          791 ;
CEF3 60       792      rts

```

```

CEF4      793 ;
CEF4      794 ;
CEF4      795 ; COPYROM is called when the video firmware is initialized.
CEF4      796 ; Only if the 0xF8 ROM's signature byte does not match, the
CEF4      797 ; language card is enabled for reading. COPYROM copies the
CEF4      798 ; 0xF8 ROM to the language card and restores the state of
CEF4      799 ; the language card.
CEF4      800 ;
CEF4 2C 12 C0 801 COPYROM bit RDLGRAM
CEF7 10 3D    802 bpl >4
CEF9      803 ;
CEF9 A9 06    804 lda #GOODF8
CEFB CD B3 FB 805 cmp SIGROM ; ROM ID byte
CEFE F0 36    806 beq >4 ; bypass if the same
CF00      807 ;
CF00 A2 03    808 ldx #RAM2WE&$F ; enable bank 2 RAM W/E
CF02      809 ;
CF02 2C 11 C0 810 bit RDBANK2 ; which bank is enabled
CF05 30 02    811 bmi >1 ; bank 2
CF07      812 ;
CF07 A2 0B    813 ldx #RAM1WE&$F ; enable bank 1 RAM W/E
CF09      814 ;
CF09 8D B3 FB 815 ^1 sta SIGROM ; save GOODF8
CF0C      816 ;
CF0C 2C 80 C0 817 bit RAM2WP ; write protect RAM
CF0F      818 ;
CF0F AD B3 FB 819 lda SIGROM
CF12 C9 06    820 cmp #GOODF8
CF14 F0 01    821 beq >2 ; bypass if the same
CF16      822 ;
CF16 E8      823 inx ; make it write protect
CF17      824 ;
CF17 2C 81 C0 825 ^2 bit ROM2WE ; enable ROM
CF1A 2C 81 C0 826 bit ROM2WE ; write enable RAM
CF1D      827 ;
CF1D A0 00    828 ldy #F8SPACE ; address of Monitor
CF1F A9 F8    829 lda /F8SPACE
CF21      830 ;
CF21 85 37    831 sta CSWH
CF23 84 36    832 sty CSWL
CF25      833 ;
CF25 B1 36    834 ^3 lda (CSWL),Y
CF27 91 36    835 sta (CSWL),Y
CF29      836 ;
CF29 C8      837 iny
CF2A D0 F9    838 bne <3
CF2C      839 ;
CF2C E6 37    840 inc CSWH
CF2E D0 F5    841 bne <3
CF30      842 ;
CF30 BD 80 C0 843 lda RAM2WP,X ; write protect RAM
CF33 BD 80 C0 844 lda RAM2WP,X ; unnecessary
CF36      845 ;
CF36 60      846 ^4 rts
CF37      847 ;
CF37      848 ;
CF37      849 dfs 3,ZERO ; 3 bytes
CF3A      850 ;
CF3A      851 ;
CF3A      852 ; Mini-Assembler support routines. Calculate offset byte
CF3A      853 ; for relative addresses.

```

```

CF3A      854 ;
CF3A E9 81 855 REL      sbc #$81
CF3C      856 ;
CF3C 4A    857          lsr
CF3D D0 14 858          bne >3
CF3F      859 ;
CF3F A4 3F 860          ldy A2H
CF41      861 ;
CF41 A6 3E 862          ldx A2L
CF43 D0 01 863          bne >1
CF45      864 ;
CF45 88    865          dey
CF46      866 ;
CF46 CA    867 ^1       dex
CF47 8A    868          txa
CF48      869 ;
CF48 18    870          clc
CF49      871 ;
CF49 E5 3A 872          sbc PCL
CF4B 85 3E 873          sta A2L
CF4D 10 01 874          bpl >2
CF4F      875 ;
CF4F C8    876          iny
CF50      877 ;
CF50 98    878 ^2       tya
CF51      879 ;
CF51 E5 3B 880          sbc PCH
CF53      881 ;
CF53 D0 40 882 ^3       bne >7
CF55      883 ;
CF55      884 ;
CF55      885 ; Move instruction to memory.
CF55      886 ;
CF55 A4 2F 887 ^4       ldy LENGTH
CF57      888 ;
CF57 B9 3D 00 889 ^5       lda A1H,Y
CF5A 91 3A    890          sta (PCL),Y
CF5C      891 ;
CF5C 88      892          dey
CF5D 10 F8    893          bpl <5
CF5F      894 ;
CF5F      895 ;
CF5F      896 ; Display instruction.
CF5F      897 ;
CF5F 20 48 F9 898          jsr PRBLNK
CF62      899 ;
CF62 20 1A FC 900          jsr UP
CF65 20 1A FC 901          jsr UP
CF68      902 ;
CF68 4C E3 FC 903          jmp FINDOP
CF6B      904 ;
CF6B      905 ;
CF6B      906 ; Compare disassembly of all known opcodes with the one
CF6B      907 ; typed in until a match is found.
CF6B      908 ;
CF6B A5 3D    909 TRYNEXT lda A1H
CF6D      910 ;
CF6D 20 8E F8 911          jsr INSDS2
CF70      912 ;
CF70 AA      913          tax
CF71      914 ;

```

```

CF71 BD EB F9      915      lda MNEMR,X
CF74 C5 42         916      cmp A4L
CF76 D0 13         917      bne >6
CF78              918      ;
CF78 BD A6 F9      919      lda MNEML,X
CF7B C5 43         920      cmp A4H
CF7D D0 0C         921      bne >6
CF7F              922      ;
CF7F A5 44         923      lda OPRND
CF81              924      ;
CF81 A4 2E         925      ldy FORMAT
CF83 C0 9D         926      cpy #$9D
CF85 F0 B3         927      beq REL
CF87              928      ;
CF87 C5 2E         929      cmp FORMAT
CF89 F0 CA         930      beq <4
CF8B              931      ;
CF8B C6 3D         932      ^6      dec A1H
CF8D D0 DC         933      bne TRYNEXT
CF8F              934      ;
CF8F E6 44         935      inc OPRND
CF91              936      ;
CF91 C6 35         937      dec YSAV1
CF93 F0 D6         938      beq TRYNEXT
CF95              939      ;
CF95              940      ;
CF95              941      ; Point to the error with a caret, beep, and fall into
CF95              942      ; the Mini-Assembler.
CF95              943      ;
CF95 A4 34         944      ^7      ldy YSAV
CF97              945      ;
CF97 98           946      ^8      tya
CF98 AA           947      tax
CF99              948      ;
CF99 4C D2 FC      949      jmp XERR
CF9C              950      ;
CF9C              951      ;
CF9C              952      ; Read a line of input.  If prefaced with " ", decode
CF9C              953      ; mnemonic.  If "$" do monitor command.  Otherwise, parse
CF9C              954      ; HEX address before decoding mnemonic.
CF9C              955      ;
CF9C 20 C7 FF      956      NXTLINE2 jsr ZMODE
CF9F              957      ;
CF9F AD 00 02      958      lda INPUT
CFA2 C9 A0         959      cmp #SPACE
CFA4 F0 12         960      beq >2
CFA6              961      ;
CFA6 C9 8D         962      cmp #RETURN
CFA8 D0 01         963      bne >9
CFAA              964      ;
CFAA 60           965      rts
CFAB              966      ;
CFAB 20 A7 FF      967      ^9      jsr GETNUM
CFAE              968      ;
CFAE C9 93         969      cmp #$89+$B0^": "
CFB0              970      ;
CFB0 D0 E5         971      ^1      bne <8
CFB2              972      ;
CFB2 8A           973      txa
CFB3 F0 E2         974      beq <8
CFB5              975      ;

```

```

CFB5 20 78 FE    976      jsr A1PCLP
CFB8              977      ;
CFB8 A9 03      978      ^2      lda #3                ; starting opcode
CFBA 85 3D      979      sta A1H
CFBC              980      ;
CFBC 20 00 C5   981      ^3      jsr GETNSP
CFBF              982      ;
CFBF 0A          983      asl
CFC0              984      ;
CFC0 E9 BE      985      sbc #$BE
CFC2              986      ;
CFC2 C9 C2      987      cmp #$C2
CFC4 90 D1      988      bcc <8
CFC6              989      ;
CFC6              990      ;
CFC6              991      ; Form mnemonic for later comparison.
CFC6              992      ;
CFC6 0A          993      asl
CFC7 0A          994      asl
CFC8              995      ;
CFC8 A2 04      996      ldx #4
CFCA              997      ;
CFCA 0A          998      ^4      asl
CFCB              999      ;
CFCB 26 42     1000      rol A4L
CFCD 26 43     1001      rol A4H
CFCF              1002      ;
CFCF CA        1003      dex
CFD0 10 F8     1004      bpl <4
CFD2              1005      ;
CFD2 C6 3D     1006      dec A1H
CFD4 F0 F4     1007      beq <4
CFD6              1008      ;
CFD6 10 E4     1009      bpl <3
CFD8              1010      ;
CFD8 A2 05     1011      ldx #5
CFDA 20 C8 C4  1012      jsr FORM
CFDD              1013      ;
CFDD A5 44     1014      lda OPRND
CFDF              1015      ;
CFDF 0A        1016      asl
CFE0 0A        1017      asl
CFE1              1018      ;
CFE1 05 35     1019      ora YSAV1
CFE3              1020      ;
CFE3 C9 20     1021      cmp #$20
CFE5 B0 06     1022      bcs >5
CFE7              1023      ;
CFE7 A6 35     1024      ldx YSAV1
CFE9 F0 02     1025      beq >5
CFEB              1026      ;
CFEB 09 80     1027      ora #MSBSET
CFED              1028      ;
CFED 85 44     1029      ^5      sta OPRND
CFEF 84 34     1030      sty YSAV
CFF1              1031      ;
CFF1 B9 00 02  1032      lda INPUT,Y
CFF4 C9 BB     1033      cmp #";"
CFF6 F0 04     1034      beq >6
CFF8              1035      ;
CFF8 C9 8D     1036      cmp #RETURN

```

```
CFFA D0 B4      1037      bne <1
CFFC            1038      ;
CFFC 4C 6B CF    1039      ^6      jmp TRYNEXT
CFFF            1040      ;
CFFF            1041      ;
CFFF            1042      CLRROM    dfs 1,ZERO      ; 1 byte
D000            1043      ;
D000            1044      ;
```

```
BSAVE C0ROM,D1,A$1000,B,L$1000
```

```
D000            1045      usr C0ROM,D1
D000            1046      ;
D000            1047      ;
D000            1048      icl "D0.L,D2"
```

```
LLOAD D0.L,D2,A$4000
```



```

D000      1          ttl "ROM Source Code, D0.L"
D000      2      ;
D000      3      ;
D000      4      ; D0.L
D000      5      ;
D000      6      ;
D000      7          obj PAGE10
D000      8          usr
D000      9      ;
D000     10      ;
D000     11      ; BASIC statement addresses.  BASIC # = #ADDR/2 + $80.
D000     12      ;
D000     13  BASADDR:
D000 6F D8     14      adr BEND-1          ; END
D002 65 D7     15  BSFOR  adr BFOR-1          ; FOR
D004 F8 DC     16      adr BNEXT-1          ; NEXT
D006 94 D9     17  BSDATA  adr BDATA-1          ; DATA
D008 B1 DB     18      adr BINPUT-1          ; INPUT
D00A 30 F3     19      adr BDEL-1           ; DEL
D00C D8 DF     20      adr BDIM-1           ; DIM
D00E E1 DB     21      adr BREAD-1          ; READ
D010 8F F3     22      adr BGR-1            ; GR
D012 98 F3     23      adr BTEXT-1          ; TEXT
D014 E4 F1     24      adr BPR-1            ; PR#
D016 DD F1     25      adr BIN-1            ; IN#
D018 D4 F1     26      adr BCALL-1           ; CALL
D01A 24 F2     27      adr BPLOT-1           ; PLOT
D01C 31 F2     28      adr BHLIN-1          ; HLIN
D01E 40 F2     29      adr BVLIN-1          ; VLIN
D020 D7 F3     30      adr BHGR2-1          ; HGR2
D022 E1 F3     31      adr BHGR-1           ; HGR
D024 E8 F6     32      adr BHCOLOR-1        ; HCOLOR=
D026 FD F6     33      adr BHPlot-1         ; HPLOT
D028 68 F7     34      adr BDRAW-1          ; DRAW
D02A 6E F7     35      adr BXDRAW-1         ; XDRAW
D02C E6 F7     36      adr BHTAB-1          ; HTAB
D02E 57 FC     37      adr HOME-1           ; HOME
D030 20 F7     38      adr BROT-1           ; ROT=
D032 26 F7     39      adr BSCALE-1         ; SCALE=
D034         40      ;
D034         41      ;          adr HF775-1          ; SHLOAD, removed
D034 57 FF     42      adr IORTS-1
D036         43      ;
D036 6C F2     44      adr BTRACE-1          ; TRACE
D038 6E F2     45      adr BNOTRACE-1        ; NOTRACE
D03A 72 F2     46      adr BNORMAL-1         ; NORMAL
D03C 76 F2     47      adr BINVERSE-1       ; INVERSE
D03E 7F F2     48      adr BFLASH-1         ; FLASH
D040 4E F2     49      adr BCOLOR-1         ; COLOR=
D042 6A D9     50  BSPOP  adr BPOP-1          ; POP
D044 55 F2     51      adr BVTAB-1          ; VTAB
D046 85 F2     52      adr BHIMEM-1         ; HIMEM:
D048 A5 F2     53      adr BLOMEM-1         ; LOMEM:
D04A CA F2     54      adr BONERR-1         ; ONERR
D04C 17 F3     55      adr BRESUME-1        ; RESUME
D04E         56      ;
D04E         57      ;          adr HF3BC-1          ; RECALL, removed
D04E 57 FF     58      adr IORTS-1
D050         59      ;          adr HF39F-1          ; STORE, removed
D050 57 FF     60      adr IORTS-1

```

```

D052      61      ;
D052 61 F2      62      adr BSPEED-1      ; SPEED=
D054 45 DA      63      adr BLET-1      ; LET
D056 3D D9      64      BSGOTO      adr BGOTO-1      ; GOTO
D058 11 D9      65      adr BRUN-1      ; RUN
D05A C8 D9      66      adr BIF-1      ; IF
D05C 48 D8      67      adr BRESTORE-1      ; RESTORE
D05E F4 03      68      adr USRAHAND-1      ; &
D060 20 D9      69      BSGOSUB      adr BGOSUB-1      ; GOSUB
D062 6A D9      70      adr BRETURN-1      ; RETURN
D064 DB D9      71      BSREM      adr BREM-1      ; REM
D066 6D D8      72      adr BSTOP-1      ; STOP
D068 EB D9      73      adr BON-1      ; ON
D06A 83 E7      74      adr BWAIT-1      ; WAIT
D06C C8 D8      75      adr BLOAD-1      ; LOAD
D06E      76      ;
D06E      77      ;      adr HD8B0-1      ; SAVE, removed
D06E 57 FF      78      adr IORTS-1
D070      79      ;
D070 12 E3      80      adr BDEF-1      ; DEF
D072 7A E7      81      adr BPOKE-1      ; POKE
D074 D4 DA      82      BSPRINT      adr BPRINT-1      ; PRINT
D076 95 D8      83      adr BCONT-1      ; CONT
D078 A4 D6      84      adr BLIST-1      ; LIST
D07A 69 D6      85      adr BCLEAR-1      ; CLEAR
D07C 9F DB      86      adr BGET-1      ; GET
D07E 48 D6      87      adr BNEW-1      ; NEW
D080      88      ;
D080      89      ;
D080      90      ; FUNCTION statement addresses.  BASIC # = #ADDR/2 + $92.
D080      91      ;
D080      92      FN1ADDR:
D080 90 EB      93      FS1SGN      adr FSGN      ; SGN
D082 23 EC      94      adr FINT      ; INT
D084 AF EB      95      adr FABS      ; ABS
D086 0A 00      96      adr GOUSR      ; USR
D088 DE E2      97      adr FFRE      ; FRE
D08A F9 DE      98      FS1SCRN      adr FSCREEN      ; SCRN(
D08C CD DF      99      adr FPDL      ; PDL
D08E FF E2     100      adr FPOS      ; POS
D090 8D EE     101      adr FSQR      ; SQR
D092 AE EF     102      adr FRND      ; RND
D094 3E EF     103      adr FLOG      ; LOG
D096 09 EF     104      adr FEXP      ; EXP
D098 EA EF     105      adr FCOS      ; COS
D09A F1 EF     106      adr FSIN      ; SIN
D09C 3A F0     107      adr FTAN      ; TAN
D09E 9E F0     108      adr FATAN      ; ATN
D0A0 64 E7     109      adr FPEEK      ; PEEK
D0A2 D6 E6     110      adr FLEN      ; LEN
D0A4 C5 E3     111      adr FSTR      ; STR$
D0A6 07 E7     112      adr FVAL      ; VAL
D0A8 E5 E6     113      adr FASC      ; ASC
D0AA 46 E6     114      adr FCHR      ; CHR$
D0AC      115      ;
D0AC 5A E6     116      FS2LEFT      adr FLEFT      ; LEFT$
D0AE 86 E6     117      adr FRIGHT      ; RIGHT$
D0B0 91 E6     118      adr FMID      ; MID$
D0B2      119      ;
D0B2 48 EF     120      FS3PI      adr FPI      ; PI
D0B4 41 E9     121      adr FLN      ; LN

```

```

D0B6      122 ;
D0B6      123 ;
D0B6      124 ; Operator TAG addresses. One-byte precedence followed by
D0B6      125 ; two-byte address.
D0B6      126 ;
D0B6      127 TAGADDR:
D0B6 79    128      byt OTV.ADD          ; +
D0B7 C0 E7 129      adr OPLUS-1
D0B9 79    130      byt OTV.ADD          ; -
D0BA A9 E7 131      adr OMINUS-1
D0BC 7B    132      byt OTV.MUL         ; *
D0BD 81 E9 133      adr OMULT-1
D0BF 7B    134      byt OTV.MUL         ; /
D0C0 68 EA 135      adr ODIVIDE-1
D0C2 7D    136      byt OTV.PWR         ; ^
D0C3 96 EE 137      adr OPOWER-1
D0C5 50    138      byt OTV.AND         ; AND
D0C6 54 DF 139      adr OAND-1
D0C8 46    140      byt OTV.OR          ; OR
D0C9 4E DF 141      adr OOR-1
D0CB 7F    142 TAG.NEG byt OTV.NEQ      ; >
D0CC CF EE 143      adr OGT-1
D0CE 7F    144 TAG.EQU byt OTV.NEQ      ; =
D0CF 97 DE 145      adr OEQUAL-1
D0D1 64    146 TAG.REL byt OTV.REL      ; <
D0D2 64 DF 147      adr OLT-1
D0D4      148 ;
D0D4      149 ;
D0D4      150 ; BASIC statement names in DCI format.
D0D4      151 ;
D0D4      152 BASNAME:
D0D4 45 4E C4 153      dci 'END'          ; 0x80
D0D7 46 4F D2 154      dci 'FOR'
D0DA 4E 45 58 155      dci 'NEXT'
D0DD D4
D0DE 44 41 54 156      dci 'DATA'
D0E1 C1
D0E2 49 4E 50 157      dci 'INPUT'
D0E5 55 D4
D0E7 44 45 CC 158      dci 'DEL'
D0EA 44 49 CD 159      dci 'DIM'
D0ED 52 45 41 160      dci 'READ'
D0F0 C4
D0F1 47 D2    161      dci 'GR'          ; 0x88
D0F3 54 45 58 162      dci 'TEXT'
D0F6 D4
D0F7 50 52 A3 163      dci 'PR#'
D0FA 49 4E A3 164      dci 'IN#'
D0FD 43 41 4C 165      dci 'CALL'
D100 CC
D101 50 4C 4F 166      dci 'PLOT'
D104 D4
D105 48 4C 49 167      dci 'HLIN'
D108 CE
D109 56 4C 49 168      dci 'VLIN'
D10C CE
D10D 48 47 52 169      dci 'HGR2'        ; 0x90
D110 B2
D111 48 47 D2 170      dci 'HGR'
D114 48 43 4F 171      dci 'HCOLOR='
D117 4C 4F 52

```

```

D11A BD
D11B 48 50 4C 172 dci 'HPLOT'
D11E 4F D4
D120 44 52 41 173 dci 'DRAW'
D123 D7
D124 58 44 52 174 dci 'XDRAW'
D127 41 D7
D129 48 54 41 175 dci 'HTAB'
D12C C2
D12D 48 4F 4D 176 dci 'HOME'
D130 C5
D131 52 4F 54 177 dci 'ROT=' ; 0x98
D134 BD
D135 53 43 41 178 dci 'SCALE='
D138 4C 45 BD
D13B 179 ;
D13B 180 ; dci 'SHLOAD'
D13B 47 D2 181 dci 'GR'
D13D 182 ;
D13D 54 52 41 183 dci 'TRACE'
D140 43 C5
D142 4E 4F 54 184 dci 'NOTRACE'
D145 52 41 43
D148 C5
D149 4E 4F 52 185 dci 'NORMAL'
D14C 4D 41 CC
D14F 49 4E 56 186 dci 'INVERSE'
D152 45 52 53
D155 C5
D156 46 4C 41 187 dci 'FLASH'
D159 53 C8
D15B 43 4F 4C 188 dci 'COLOR=' ; 0xA0
D15E 4F 52 BD
D161 50 4F D0 189 dci 'POP'
D164 56 54 41 190 dci 'VTAB'
D167 C2
D168 48 49 4D 191 dci 'HIMEM:'
D16B 45 4D BA
D16E 4C 4F 4D 192 dci 'LOMEM:'
D171 45 4D BA
D174 4F 4E 45 193 dci 'ONERR'
D177 52 D2
D179 52 45 53 194 dci 'RESUME'
D17C 55 4D C5
D17F 195 ;
D17F 196 ; dci 'RECALL'
D17F 47 D2 197 dci 'GR'
D181 198 ; dci 'STORE' ; 0xA8
D181 47 D2 199 dci 'GR'
D183 200 ;
D183 53 50 45 201 dci 'SPEED='
D186 45 44 BD
D189 4C 45 D4 202 dci 'LET'
D18C 47 4F 54 203 dci 'GOTO'
D18F CF
D190 52 55 CE 204 dci 'RUN'
D193 49 C6 205 dci 'IF'
D195 52 45 53 206 dci 'RESTORE'
D198 54 4F 52
D19B C5
D19C A6 207 asc "&"

```

D19D	47	4F	53	208	dc	i	'GOSUB'		; 0xB0
D1A0	55	C2							
D1A2	52	45	54	209	dc	i	'RETURN'		
D1A5	55	52	CE						
D1A8	52	45	CD	210	dc	i	'REM'		
D1AB	53	54	4F	211	dc	i	'STOP'		
D1AE	D0								
D1AF	4F	CE		212	dc	i	'ON'		
D1B1	57	41	49	213	dc	i	'WAIT'		
D1B4	D4								
D1B5	4C	4F	41	214	dc	i	'LOAD'		
D1B8	C4								
D1B9				215					;
D1B9				216					;
D1B9	47	D2		217	dc	i	'SAVE'		
D1BB				218	dc	i	'GR'		
D1BB	44	45	C6	219					;
D1BE	50	4F	4B	220	dc	i	'DEF'		0xB8
D1C1	C5								
D1C2	50	52	49	221	dc	i	'PRINT'		
D1C5	4E	D4							
D1C7	43	4F	4E	222	dc	i	'CONT'		
D1CA	D4								
D1CB	4C	49	53	223	dc	i	'LIST'		
D1CE	D4								
D1CF	43	4C	45	224	dc	i	'CLEAR'		
D1D2	41	D2							
D1D4	47	45	D4	225	dc	i	'GET'		
D1D7	4E	45	D7	226	dc	i	'NEW'		0xBF
D1DA				227					;
D1DA	54	41	42	228	dc	i	'TAB('		0xC0
D1DD	A8								
D1DE	54	CF		229	dc	i	'TO'		
D1E0	46	CE		230	dc	i	'FN'		
D1E2	53	50	43	231	dc	i	'SPC('		
D1E5	A8								
D1E6	54	48	45	232	dc	i	'THEN'		
D1E9	CE								
D1EA	41	D4		233	dc	i	'AT'		
D1EC	4E	4F	D4	234	dc	i	'NOT'		
D1EF	53	54	45	235	dc	i	'STEP'		0xC7
D1F2	D0								
D1F3				236					;
D1F3	AB			237	asc		"+"		0xC8
D1F4	AD			238	asc		" - "		
D1F5	AA			239	asc		" * "		
D1F6	AF			240	asc		" / "		
D1F7	DE			241	asc		" ^ "		
D1F8	41	4E	C4	242	dc	i	'AND'		
D1FB	4F	D2		243	dc	i	'OR'		
D1FD	BE			244	asc		" > "		
D1FE	BD			245	asc		" = "		0xD0
D1FF	BC			246	asc		" < "		
D200				247					;
D200	53	47	CE	248	dc	i	'SGN'		0xD2
D203	49	4E	D4	249	dc	i	'INT'		
D206	41	42	D3	250	dc	i	'ABS'		
D209	55	53	D2	251	dc	i	'USR'		
D20C	46	52	C5	252	dc	i	'FRE'		
D20F	53	43	52	253	dc	i	'SCRN('		
D212	4E	A8							

```

D214 50 44 CC 254 dci 'PDL' ; 0xD8
D217 50 4F D3 255 dci 'POS'
D21A 53 51 D2 256 dci 'SQR'
D21D 52 4E C4 257 dci 'RND'
D220 4C 4F C7 258 dci 'LOG'
D223 45 58 D0 259 dci 'EXP'
D226 43 4F D3 260 dci 'COS'
D229 53 49 CE 261 dci 'SIN'
D22C 54 41 CE 262 dci 'TAN' ; 0xE0
D22F 41 54 CE 263 dci 'ATN'
D232 50 45 45 264 dci 'PEEK'
D235 CB
D236 4C 45 CE 265 dci 'LEN'
D239 53 54 52 266 dci 'STR$'
D23C A4
D23D 56 41 CC 267 dci 'VAL'
D240 41 53 C3 268 dci 'ASC'
D243 43 48 52 269 dci 'CHR$' ; 0xE7
D246 A4
D247 270 ;
D247 4C 45 46 271 dci 'LEFT$' ; 0xE8
D24A 54 A4
D24C 52 49 47 272 dci 'RIGHT$'
D24F 48 54 A4
D252 4D 49 44 273 dci 'MID$' ; 0xEA
D255 A4
D256 274 ;
D256 50 C9 275 dci 'PI' ; 0xEB, added statement
D258 4C CE 276 dci 'LN' ; 0xEC, added statement
D25A 277 ;
D25A 00 278 hex 00
D25B 279 ;
D25B 280 ;
D25B 281 ; Error messages in mixed case, complete without using
D25B 282 ; contractions, and in DCI format.
D25B 283 ;
D25B 284 MSGS:
D25B 4E 45 58 285 MSG01 dci 'NEXT without FOR'
D25E 54 20 77
D261 69 74 68
D264 6F 75 74
D267 20 46 4F
D26A D2
D26B 53 79 6E 286 MSG02 dci 'Syntax'
D26E 74 61 F8
D271 52 45 54 287 MSG03 dci 'RETURN without GOSUB'
D274 55 52 4E
D277 20 77 69
D27A 74 68 6F
D27D 75 74 20
D280 47 4F 53
D283 55 C2
D285 4F 75 74 288 MSG04 dci 'Out of Data'
D288 20 6F 66
D28B 20 44 61
D28E 74 E1
D290 49 6C 6C 289 MSG05 dci 'Illegal Quantity'
D293 65 67 61
D296 6C 20 51
D299 75 61 6E
D29C 74 69 74

```

```

D29F F9
D2A0 4F 76 65    290    MSG06    dci 'Overflow'
D2A3 72 66 6C
D2A6 6F F7
D2A8 4F 75 74    291    MSG07    dci 'Out of Memory'
D2AB 20 6F 66
D2AE 20 4D 65
D2B1 6D 6F 72
D2B4 F9
D2B5 55 6E 64    292    MSG08    dci 'Undefined Statement'
D2B8 65 66 69
D2BB 6E 65 64
D2BE 20 53 74
D2C1 61 74 65
D2C4 6D 65 6E
D2C7 F4
D2C8 42 61 64    293    MSG09    dci 'Bad Subscript'
D2CB 20 53 75
D2CE 62 73 63
D2D1 72 69 70
D2D4 F4
D2D5 52 65 64    294    MSG10    dci 'Redefined Array'
D2D8 65 66 69
D2DB 6E 65 64
D2DE 20 41 72
D2E1 72 61 F9
D2E4 44 69 76    295    MSG11    dci 'Division by Zero'
D2E7 69 73 69
D2EA 6F 6E 20
D2ED 62 79 20
D2F0 5A 65 72
D2F3 EF
D2F4 49 6C 6C    296    MSG12    dci 'Illegal Direct'
D2F7 65 67 61
D2FA 6C 20 44
D2FD 69 72 65
D300 63 F4
D302 54 79 70    297    MSG13    dci 'Type Mismatch'
D305 65 20 4D
D308 69 73 6D
D30B 61 74 63
D30E E8
D30F 53 74 72    298    MSG14    dci 'String too Long'
D312 69 6E 67
D315 20 74 6F
D318 6F 20 4C
D31B 6F 6E E7
D31E 46 6F 72    299    MSG15    dci 'Formula too Complex'
D321 6D 75 6C
D324 61 20 74
D327 6F 6F 20
D32A 43 6F 6D
D32D 70 6C 65
D330 F8
D331 43 61 6E    300    MSG16    dci 'Cannot Continue'
D334 6E 6F 74
D337 20 43 6F
D33A 6E 74 69
D33D 6E 75 E5
D340 55 6E 64    301    MSG17    dci 'Undefined Function'
D343 65 66 69

```

```

D346 6E 65 64
D349 20 46 75
D34C 6E 63 74
D34F 69 6F EE
D352          302 ;
D352 20 45 72 303 MSG18   asc  `Error`
D355 72 6F 72
D358 07 00     304         byt  ASCIBELL&MSBCLR,ZERO
D35A          305 ;
D35A 0D        306 MSG20   byt  RETURN&MSBCLR
D35B 42 72 65 307         asc  `Break`
D35E 61 6B
D360 07 00     308         byt  ASCIBELL&MSBCLR,ZERO
D362          309 ;
D362          310 ;
D362          311 ; Called by NEXT and FOR to scan through the STACK for a
D362          312 ; frame with the same variable.  FORPNT contains address
D362          313 ; of FOR or NEXT variable or 0xFFxx if called from RETURN.
D362          314 ; Returns NE if variable not found and X-reg has STACK
D362          315 ; pointer after skipping all frames or EQ if variable found
D362          316 ; and X-reg has STACK pointer of frame found.  Pushed this
D362          317 ; code back three bytes to accelerate this routine.
D362          318 ;
D362 BA        319 GTFORPNT tsx
D363          320 ;
D363 E8        321         inx
D364 E8        322         inx
D365 E8        323         inx
D366 E8        324         inx
D367          325 ;
D367 BD 01 01  326 ^1      lda  STACK+1,X
D36A C9 81     327         cmp  #$80+BSFOR/2
D36C D0 24     328         bne  >4
D36E          329 ;
D36E A5 86     330         lda  FORPNT+1
D370 D0 0D     331         bne  >2
D372          332 ;
D372 BD 02 01  333         lda  STACK+2,X
D375 85 85     334         sta  FORPNT
D377          335 ;
D377 BD 03 01  336         lda  STACK+3,X
D37A 85 86     337         sta  FORPNT+1
D37C          338 ;
D37C C5 86     339         cmp  FORPNT+1           ; accelerate code; force Z=1
D37E          340 ;
D37E 60        341         rts
D37F          342 ;
D37F DD 03 01  343 ^2      cmp  STACK+3,X
D382 D0 07     344         bne  >3
D384          345 ;
D384 A5 85     346         lda  FORPNT
D386 DD 02 01  347         cmp  STACK+2,X
D389 F0 07     348         beq  >4
D38B          349 ;
D38B 18        350 ^3      clc
D38C          351 ;
D38C 8A        352         txa
D38D 69 14     353         adc  #FRAMSIZE           ; size of FOR/NEXT frame
D38F          354 ;
D38F AA        355         tax
D390 D0 D5     356         bne  <1           ; always taken, STACK overflow

```



```

D392          357 ;
D392 60       358 ^4      rts
D393          359 ;
D393          360 ;
D393          361 ; Block transfer utility (ACA). On entry A/Y equals HIGHDS
D393          362 ; or destination END+1 address, LOWTR equals lowest source
D393          363 ; address, and HIGHTR equals highest source address+1.
D393          364 ;
D393 20 E3 D3 365 BLTU      jsr CKSTRSIZ
D396          366 ;
D396 85 6D     367          sta STREND
D398 84 6E     368          sty STREND+1
D39A          369 ;
D39A 38        370          sec
D39B          371 ;
D39B A5 96     372          lda HIGHTR
D39D E5 9B     373          sbc LOWTR
D39F 85 5E     374          sta INDEX
D3A1          375 ;
D3A1 A8        376          tay
D3A2          377 ;
D3A2 A5 97     378          lda HIGHTR+1
D3A4 E5 9C     379          sbc LOWTR+1
D3A6 AA        380          tax
D3A7          381 ;
D3A7 E8        382          inx
D3A8          383 ;
D3A8 98        384          tya
D3A9 F0 23     385          beq >4
D3AB          386 ;
D3AB 38        387          sec
D3AC          388 ;
D3AC A5 96     389          lda HIGHTR
D3AE E5 5E     390          sbc INDEX
D3B0 85 96     391          sta HIGHTR
D3B2 B0 03     392          bcs >1
D3B4          393 ;
D3B4 C6 97     394          dec HIGHTR+1
D3B6          395 ;
D3B6 38        396          sec
D3B7          397 ;
D3B7 A5 94     398 ^1      lda HIGHDS
D3B9 E5 5E     399          sbc INDEX
D3BB 85 94     400          sta HIGHDS
D3BD B0 08     401          bcs >3
D3BF          402 ;
D3BF C6 95     403          dec HIGHDS+1
D3C1 90 04     404          bcc >3                      ; always taken
D3C3          405 ;
D3C3 B1 96     406 ^2      lda (HIGHTR),Y
D3C5 91 94     407          sta (HIGHDS),Y
D3C7          408 ;
D3C7 88        409 ^3      dey
D3C8 D0 F9     410          bne <2
D3CA          411 ;
D3CA B1 96     412          lda (HIGHTR),Y
D3CC 91 94     413          sta (HIGHDS),Y
D3CE          414 ;
D3CE C6 97     415 ^4      dec HIGHTR+1
D3D0 C6 95     416          dec HIGHDS+1
D3D2          417 ;

```

```

D3D2 CA          418          dex
D3D3 D0 F2       419          bne <3
D3D5             420          ;
D3D5 60          421          rts
D3D6             422          ;
D3D6             423          ;
D3D6             424          ; Check STACK size to process FOR, GOSUB, or expression
D3D6             425          ; evaluation.
D3D6             426          ;
D3D6 0A          427          CKSTKSIZ asl
D3D7             428          ;
D3D7 69 36       429          adc #$36
D3D9 B0 35       430          bcs OM.ERR
D3DB             431          ;
D3DB 85 5E       432          sta INDEX
D3DD             433          ;
D3DD BA          434          tsx
D3DE             435          ;
D3DE E4 5E       436          cpx INDEX
D3E0 90 2E       437          bcc OM.ERR
D3E2             438          ;
D3E2 60          439          rts
D3E3             440          ;
D3E3             441          ;
D3E3             442          ; Check memory size between arrays and strings.  Protect
D3E3             443          ; 0x94 to 0x9C if GARBAG is called.
D3E3             444          ;
D3E3 C4 70       445          CKSTRSIZ cpy FRETOP+1
D3E5 90 28       446          bcc >4
D3E7             447          ;
D3E7 D0 04       448          bne >1
D3E9             449          ;
D3E9 C5 6F       450          cmp FRETOP
D3EB 90 22       451          bcc >4
D3ED             452          ;
D3ED 48          453          ^1 pha
D3EE             454          ;
D3EE A2 09       455          ldx #9
D3F0             456          ;
D3F0 98          457          tya
D3F1             458          ;
D3F1 48          459          ^2 pha
D3F2             460          ;
D3F2 B5 93       461          lda TEMP1,X
D3F4             462          ;
D3F4 CA          463          dex
D3F5 10 FA       464          bpl <2
D3F7             465          ;
D3F7 20 84 E4    466          jsr GARBAG
D3FA             467          ;
D3FA A2 F7       468          ldx #!-9
D3FC             469          ;
D3FC 68          470          ^3 pla
D3FD 95 9D       471          sta TEMP1+$A,X
D3FF             472          ;
D3FF E8          473          inx
D400 30 FA       474          bmi <3
D402             475          ;
D402 68          476          pla
D403 A8          477          tay
D404             478          ;

```

```
D404 68          479          pla
D405          480          ;
D405 C4 70      481          cpy FRETOP+1
D407 90 06      482          bcc >4
D409          483          ;
D409 D0 05      484          bne OM.ERR
D40B          485          ;
D40B C5 6F      486          cmp FRETOP
D40D B0 01      487          bcs OM.ERR
D40F          488          ;
D40F 60          489          ^4      rts
D410          490          ;
D410          491          ;
D410          492          ; Error handler.  Checks if ERRFLG > 127 for ONERR.
D410          493          ;
D410 A2 4D      494          OM.ERR    ldx #MSG07-MESGS    ; Out of Memory error
D412          495          ;
D412          496          ;
D412          497          icl "D4.L"
```

```
LLOAD D4.L,A$4000
```

```

D412      1          ttl "ROM Source Code, D4.L"
D412      2      ;
D412      3      ;
D412      4      ; D4.L
D412      5      ;
D412      6      ;
D412 24 D8      7  PRterr    bit  ERRFLG
D414 10 03      8          bpl >1
D416      9      ;
D416 4C E9 F2   10          jmp  HANDLERR
D419     11      ;
D419 20 50 DB   12      ^1    jsr  PRTCR
D41C 20 56 DB   13          jsr  OUTPROMT
D41F     14      ;
D41F BD 5B D2   15      ^2    lda  MESSAGES,X
D422 48         16          pha
D423     17      ;
D423 20 58 DB   18          jsr  OUTCHR
D426     19      ;
D426 E8         20          inx
D427     21      ;
D427 68         22          pla
D428 10 F5      23          bpl <2
D42A     24      ;
D42A 20 83 D6   25          jsr  STKINIT
D42D     26      ;
D42D A9 52      27          lda  #MSG18          ; ' Error '
D42F A0 D3      28          ldy  /MSG18
D431     29      ;
D431     30      ;
D431 20 3B DB   31  PRLINUM  jsr  STROUT
D434     32      ;
D434 A4 76      33          ldy  CURLIN+1
D436 20 0A ED   34          jsr  PRTMSG19          ; print MSG19 if Y-reg not -1
D439     35      ;
D439 20 50 DB   36          jsr  PRTCR
D43C     37      ;
D43C     38      ;
D43C     39      ; This is the default DOS restart WARMADR entry.
D43C     40      ;
D43C     41  ASROMWRM:
D43C 20 50 DB   42  RESTART  jsr  PRTCR
D43F     43      ;
D43F A2 DD      44          ldx  #"]"
D441 20 2E D5   45          jsr  INLIN2
D444     46      ;
D444 86 B8      47          stx  TXTPTR
D446 84 B9      48          sty  TXTPTR+1
D448     49      ;
D448 46 D8      50          lsr  ERRFLG
D44A     51      ;
D44A 20 B1 00   52          jsr  CHRGET
D44D     53      ;
D44D AA         54          tax
D44E F0 EC      55          beq  RESTART
D450     56      ;
D450 A2 FF      57          ldx  #NEGONE
D452 86 76      58          stx  CURLIN+1
D454     59      ;
D454 90 06      60          bcc >1          ; a number from CHRGET

```

```

D456      61 ;
D456 20 59 D5      62      jsr PARSINPT
D459      63 ;
D459 4C 05 D8      64      jmp DOTRACE
D45C      65 ;
D45C A6 AF      66 ^1      ldx PRGEND
D45E 86 69      67      stx VARTAB
D460      68 ;
D460 A6 B0      69      ldx PRGEND+1
D462 86 6A      70      stx VARTAB+1
D464      71 ;
D464 20 0C DA      72      jsr LINGET
D467 20 59 D5      73      jsr PARSINPT
D46A      74 ;
D46A 84 0F      75      sty EOLPTR
D46C      76 ;
D46C 20 1A D6      77      jsr FNDLIN
D46F 90 44      78      bcc >4
D471      79 ;
D471 A0 01      80      ld y #1
D473      81 ;
D473 B1 9B      82      lda (LOWTR),Y
D475 85 5F      83      sta INDEX+1
D477      84 ;
D477 A5 69      85      lda VARTAB
D479 85 5E      86      sta INDEX
D47B      87 ;
D47B A5 9C      88      lda LOWTR+1
D47D 85 61      89      sta DEST+1
D47F      90 ;
D47F A5 9B      91      lda LOWTR
D481      92 ;
D481 88      93      dey
D482      94 ;
D482 F1 9B      95      sbc (LOWTR),Y
D484      96 ;
D484 18      97      clc
D485      98 ;
D485 65 69      99      adc VARTAB
D487 85 69     100      sta VARTAB
D489 85 60     101      sta DEST
D48B      102 ;
D48B A5 6A     103      lda VARTAB+1
D48D 69 FF     104      adc #NEGONE
D48F 85 6A     105      sta VARTAB+1
D491      106 ;
D491 E5 9C     107      sbc LOWTR+1
D493 AA      108      tax
D494      109 ;
D494 38      110      sec
D495      111 ;
D495 A5 9B     112      lda LOWTR
D497 E5 69     113      sbc VARTAB
D499 A8      114      tay
D49A B0 03     115      bcs >2
D49C      116 ;
D49C E8      117      inx
D49D      118 ;
D49D C6 61     119      dec DEST+1
D49F      120 ;
D49F 18      121 ^2      clc

```

```

D4A0          122 ;
D4A0 65 5E    123      adc INDEX
D4A2 90 03    124      bcc >3
D4A4          125 ;
D4A4 C6 5F    126      dec INDEX+1
D4A6          127 ;
D4A6 18       128      clc
D4A7          129 ;
D4A7          130 ;
D4A7          131 ; Move higher program lines down over deleted line.
D4A7          132 ;
D4A7 B1 5E    133 ^3      lda (INDEX),Y
D4A9 91 60    134      sta (DEST),Y
D4AB          135 ;
D4AB C8       136      iny
D4AC D0 F9    137      bne <3
D4AE          138 ;
D4AE E6 5F    139      inc INDEX+1
D4B0 E6 61    140      inc DEST+1
D4B2          141 ;
D4B2 CA       142      dex
D4B3 D0 F2    143      bne <3
D4B5          144 ;
D4B5          145 ;
D4B5          146 ; Insert a new line.
D4B5          147 ;
D4B5 AD 00 02 148 ^4      lda INPUT
D4B8 F0 38    149      beq ASENTER
D4BA          150 ;
D4BA A5 73    151      lda MEMSIZE
D4BC A4 74    152      ldy MEMSIZE+1
D4BE          153 ;
D4BE 85 6F    154      sta FRETOP
D4C0 84 70    155      sty FRETOP+1
D4C2          156 ;
D4C2 A5 69    157      lda VARTAB
D4C4 A4 6A    158      ldy VARTAB+1
D4C6          159 ;
D4C6 85 96    160      sta HIGHTR
D4C8 84 97    161      sty HIGHTR+1
D4CA          162 ;
D4CA 65 0F    163      adc EOLPTR
D4CC 90 01    164      bcc >5
D4CE          165 ;
D4CE C8       166      iny
D4CF          167 ;
D4CF 85 94    168 ^5      sta HIGHDS
D4D1 84 95    169      sty HIGHDS+1
D4D3          170 ;
D4D3 20 93 D3 171      jsr BLTU
D4D6          172 ;
D4D6 A5 50    173      lda LINNUM
D4D8 A4 51    174      ldy LINNUM+1
D4DA          175 ;
D4DA 8D FE 01 176      sta INPUT-2
D4DD 8C FF 01 177      sty INPUT-1
D4E0          178 ;
D4E0 A5 6D    179      lda STREND
D4E2 A4 6E    180      ldy STREND+1
D4E4          181 ;
D4E4 85 69    182      sta VARTAB

```

```

D4E6 84 6A      183      sty VARTAB+1
D4E8           184      ;
D4E8 A4 0F      185      ldy EOLPTR
D4EA           186      ;
D4EA           187      ;
D4EA           188      ; Copy a line into a program.
D4EA           189      ;
D4EA B9 FB 01   190      ^6      lda INPUT-5,Y
D4ED           191      ;
D4ED 88         192      dey
D4EE           193      ;
D4EE 91 9B      194      sta (LOWTR),Y
D4F0           195      ;
D4F0 D0 F8      196      bne <6
D4F2           197      ;
D4F2           198      ;
D4F2           199      ; Applesoft interpreter. Used by DOS for the LOAD command
D4F2           200      ; as ASROMRST and is the default DOS reset RESETADR.
D4F2           201      ;
D4F2           202      ASROMRST:
D4F2 20 65 D6   203      ASENTER jsr SETPTRS
D4F5           204      ;
D4F5 A5 67      205      lda PRGTAB
D4F7 A4 68      206      ldy PRGTAB+1
D4F9           207      ;
D4F9 85 5E      208      sta INDEX
D4FB 84 5F      209      sty INDEX+1
D4FD           210      ;
D4FD 18         211      clc
D4FE           212      ;
D4FE A0 01      213      ^1      ldy #1
D500           214      ;
D500 B1 5E      215      lda (INDEX),Y
D502 D0 0B      216      bne >2
D504           217      ;
D504 A5 69      218      lda VARTAB
D506 85 AF      219      sta PRGEND
D508           220      ;
D508 A5 6A      221      lda VARTAB+1
D50A 85 B0      222      sta PRGEND+1
D50C           223      ;
D50C 4C 3C D4   224      jmp RESTART
D50F           225      ;
D50F A0 04      226      ^2      ldy #4
D511           227      ;
D511 C8         228      ^3      iny
D512           229      ;
D512 B1 5E      230      lda (INDEX),Y
D514 D0 FB      231      bne <3
D516           232      ;
D516 C8         233      iny
D517           234      ;
D517 98         235      tya
D518 65 5E      236      adc INDEX
D51A AA         237      tax
D51B           238      ;
D51B A0 00      239      ldy #ZERO
D51D           240      ;
D51D 91 5E      241      sta (INDEX),Y
D51F           242      ;
D51F A5 5F      243      lda INDEX+1

```

```

D521 69 00      244      adc #ZERO
D523           245      ;
D523 C8        246      iny
D524           247      ;
D524 91 5E     248      sta (INDEX),Y
D526           249      ;
D526 86 5E     250      stx INDEX
D528 85 5F     251      sta INDEX+1
D52A           252      ;
D52A 90 D2     253      bcc <1                ; always taken
D52C           254      ;
D52C           255      ;
D52C           256      ; Read an INPUT line and clear all MSBs.  Accelerate code.
D52C           257      ;
D52C A2 80     258      INLIN      ldx #$80
D52E           259      ;
D52E 86 33     260      INLIN2    stx PROMPT
D530           261      ;
D530 20 6A FD  262      jsr GETLINE
D533           263      ;
D533 E0 EF     264      cpx #MAXINPUT
D535 90 02     265      bcc >1
D537           266      ;
D537 A2 EF     267      ldx #MAXINPUT
D539           268      ;
D539 A9 00     269      ^1      lda #ZERO
D53B 9D 00 02  270      sta INPUT,X
D53E           271      ;
D53E 8A        272      txa
D53F F0 0C     273      beq >3
D541           274      ;
D541 BD FF 01  275      ^2      lda INPUT-1,X
D544 29 7F     276      and #MSBCLR
D546 9D FF 01  277      sta INPUT-1,X
D549           278      ;
D549 CA        279      dex
D54A D0 F5     280      bne <2
D54C           281      ;
D54C 8A        282      txa                ; make A-reg zero
D54D           283      ;
D54D CA        284      ^3      dex                ; make X-reg -1
D54E           285      ;
D54E A0 01     286      ldy /INPUT-1
D550           287      ;
D550 60        288      rts
D551           289      ;
D551           290      ;
D551           291      ; Removed INCHR routine as unnecessary.
D551           292      ;
D551           293      ;
D551           294      dfs 8,ZERO                ; 8 bytes
D559           295      ;
D559           296      ;
D559           297      ; Parse and tokenize the input line.
D559           298      ;
D559 A6 B8     299      PARSINPT ldx TXTPTR
D55B           300      ;
D55B CA        301      dex
D55C           302      ;
D55C A0 04     303      ldy #4
D55E 84 13     304      sty DATAFLG

```



```

D560          305 ;
D560 24 D6    306      bit RUNFLAG
D562 10 08    307      bpl PARSE
D564          308 ;
D564 68      309      pla
D565 68      310      pla
D566          311 ;
D566 20 65 D6 312      jsr SETPTRS
D569          313 ;
D569 4C D2 D7 314      jmp NEWSTT
D56C          315 ;
D56C          316 ;
D56C E8      317 PARSE    inx
D56D          318 ;
D56D 20 A0 F7 319 PARSE2  jsr PARSIEX2
D570          320 ;
D570 24 13    321      bit DATAFLG
D572 70 04    322      bvs >1
D574          323 ;
D574 C9 20    324      cmp #' '
D576 F0 F4    325      beq PARSE
D578          326 ;
D578 85 0E    327 ^1     sta ENDCHR
D57A          328 ;
D57A C9 22    329      cmp #'" '
D57C F0 74    330      beq PARSE5
D57E          331 ;
D57E 70 4D    332      bvs PARSE4
D580          333 ;
D580 C9 3F    334      cmp #'? '
D582 D0 04    335      bne >2
D584          336 ;
D584 A9 BA    337      lda #$80+BSPRINT/2
D586 D0 45    338      bne PARSE4          ; always taken
D588          339 ;
D588 C9 30    340 ^2     cmp #'0 '
D58A 90 04    341      bcc >3
D58C          342 ;
D58C C9 3C    343      cmp #' ; '+1
D58E 90 3D    344      bcc PARSE4
D590          345 ;
D590 84 AD    346 ^3     sty STRING2
D592          347 ;
D592 A9 D4    348      lda #BASNAME-PAGESIZE
D594 85 9D    349      sta DSCTMP
D596          350 ;
D596 A9 CF    351      lda /BASNAME-PAGESIZE
D598 85 9E    352      sta DSCTMP+1
D59A          353 ;
D59A A0 00    354      ldy #ZERO
D59C 84 0F    355      sty TOKNCNTR
D59E          356 ;
D59E 88      357      dey
D59F          358 ;
D59F 86 B8    359      stx TXTPTR
D5A1          360 ;
D5A1 CA      361      dex
D5A2          362 ;
D5A2 C8      363 ^4     iny
D5A3 D0 02    364      bne >5
D5A5          365 ;

```

```

D5A5 E6 9E      366      inc DSCTMP+1
D5A7           367      ;
D5A7 E8        368      ^5      inx
D5A8           369      ;
D5A8 20 A0 F7   370      PARSE3   jsr PARSIEX2
D5AB           371      ;
D5AB C9 20      372      cmp #' '
D5AD F0 F8      373      beq <5
D5AF           374      ;
D5AF 38         375      sec
D5B0           376      ;
D5B0 F1 9D      377      sbc (DSCTMP),Y
D5B2 F0 EE      378      beq <4
D5B4           379      ;
D5B4 C9 80      380      cmp #$80
D5B6 D0 41      381      bne PARSE6
D5B8           382      ;
D5B8 05 0F      383      ora TOKNCNTR
D5BA C9 C5      384      cmp #TK.AT
D5BC D0 0D      385      bne >6
D5BE           386      ;
D5BE 20 9B F7   387      jsr PARSIEX1
D5C1           388      ;
D5C1 C9 4E      389      cmp #'N'      ; ATN over AT
D5C3 F0 34      390      beq PARSE6
D5C5           391      ;
D5C5 C9 4F      392      cmp #'O'      ; TO over AT
D5C7 F0 30      393      beq PARSE6
D5C9           394      ;
D5C9 A9 C5      395      lda #TK.AT
D5CB           396      ;
D5CB A4 AD      397      ^6      ldy STRING2
D5CD           398      ;
D5CD E8         399      PARSE4   inx
D5CE C8         400      iny
D5CF           401      ;
D5CF 99 FB 01   402      sta INPUT-5,Y
D5D2           403      ;
D5D2 B9 FB 01   404      lda INPUT-5,Y
D5D5 F0 39      405      beq PARSE7
D5D7           406      ;
D5D7 38         407      sec
D5D8           408      ;
D5D8 E9 3A      409      sbc #': '
D5DA F0 04      410      beq >7
D5DC           411      ;
D5DC C9 49      412      cmp #$46+BSDATA/2
D5DE D0 02      413      bne >8
D5E0           414      ;
D5E0 85 13      415      ^7      sta DATAFLG
D5E2           416      ;
D5E2 38         417      ^8      sec
D5E3           418      ;
D5E3 E9 78      419      sbc #$46+BSREM/2
D5E5 D0 86      420      bne PARSE2
D5E7           421      ;
D5E7 85 0E      422      sta ENDCHR
D5E9           423      ;
D5E9 20 A0 F7   424      ^9      jsr PARSIEX2
D5EC F0 DF      425      beq PARSE4
D5EE           426      ;

```

```

D5EE C5 0E      427      cmp ENDCHR
D5F0 F0 DB      428      beq PARSE4
D5F2           429      ;
D5F2 C8         430 PARSE5   iny
D5F3           431      ;
D5F3 99 FB 01   432      sta INPUT-5,Y
D5F6           433      ;
D5F6 E8         434      inx
D5F7 D0 F0      435      bne <9
D5F9           436      ;
D5F9 A6 B8      437 PARSE6   ldx TXTPTR
D5FB           438      ;
D5FB E6 0F      439      inc TOKNCNTR
D5FD           440      ;
D5FD B1 9D      441 ^1     lda (DSCTMP),Y
D5FF           442      ;
D5FF C8         443      iny
D600 D0 02      444      bne >2
D602           445      ;
D602 E6 9E      446      inc DSCTMP+1
D604           447      ;
D604 0A         448 ^2     asl
D605 90 F6      449      bcc <1
D607           450      ;
D607 B1 9D      451      lda (DSCTMP),Y
D609 D0 9D      452      bne PARSE3
D60B           453      ;
D60B 20 AE F7   454      jsr PARSIEX3
D60E 10 BB      455      bpl <6
D610           456      ;
D610 99 FD 01   457 PARSE7   sta INPUT-3,Y
D613           458      ;
D613 C6 B9      459      dec TXTPTR+1
D615           460      ;
D615 A9 FF      461      lda #NEGONE
D617 85 B8      462      sta TXTPTR
D619           463      ;
D619 60         464      rts
D61A           465      ;
D61A           466      ;
D61A           467      ; Search for line number in LINNUM. If not found, C-flag
D61A           468      ; is clear and LOWTR points to next line. If found, C-flag
D61A           469      ; is set and LOWTR points to the line.
D61A           470      ;
D61A A5 67      471 FNDLIN   lda PRGTAB
D61C A6 68      472      ldx PRGTAB+1
D61E           473      ;
D61E A0 01      474 FNDLIN2  ldy #1
D620           475      ;
D620 85 9B      476      sta LOWTR
D622 86 9C      477      stx LOWTR+1
D624           478      ;
D624 B1 9B      479      lda (LOWTR),Y
D626 F0 1F      480      beq >3
D628           481      ;
D628 C8         482      iny
D629 C8         483      iny
D62A           484      ;
D62A A5 51      485      lda LINNUM+1
D62C D1 9B      486      cmp (LOWTR),Y
D62E 90 18      487      bcc RTN.D6.4

```

```

D630          488 ;
D630 F0 03    489      beq >1
D632          490 ;
D632 88       491      dey
D633 D0 09    492      bne >2
D635          493 ;
D635 A5 50    494 ^1    lda LINNUM
D637          495 ;
D637 88       496      dey
D638          497 ;
D638 D1 9B    498      cmp (LOWTR),Y
D63A 90 0C    499      bcc RTN.D6.4
D63C          500 ;
D63C F0 0A    501      beq RTN.D6.4
D63E          502 ;
D63E 88       503 ^2    dey
D63F          504 ;
D63F B1 9B    505      lda (LOWTR),Y
D641 AA       506      tax
D642          507 ;
D642 88       508      dey
D643          509 ;
D643 B1 9B    510      lda (LOWTR),Y
D645          511 ;
D645 B0 D7    512      bcs FNDLIN2
D647          513 ;
D647 18       514 ^3    clc
D648          515 ;
D648 60       516 RTN.D6.4 rts
D649          517 ;
D649          518 ;
D649 D0 FD    519 BNEW    bne RTN.D6.4      ; if more to statement, ignore
D64B          520 ;
D64B          521 ;
D64B A9 00    522 SCRTCH  lda #ZERO
D64D 85 D6    523      sta RUNFLAG
D64F A8       524      tay
D650          525 ;
D650 91 67    526      sta (PRGTAB),Y
D652          527 ;
D652 C8       528      iny
D653          529 ;
D653 91 67    530      sta (PRGTAB),Y
D655          531 ;
D655 A5 67    532      lda PRGTAB
D657 69 02    533      adc #2
D659 85 69    534      sta VARTAB
D65B 85 AF    535      sta PRGEND
D65D          536 ;
D65D A5 68    537      lda PRGTAB+1
D65F 69 00    538      adc #ZERO
D661 85 6A    539      sta VARTAB+1
D663 85 B0    540      sta PRGEND+1
D665          541 ;
D665          542 ;
D665          543 ; Used by DOS for the RUN and CHAIN commands as ASROMCLR.
D665          544 ;
D665          545 ASROMCLR:
D665 20 97 D6 546 SETPTRS  jsr STXTPTR
D668          547 ;
D668 A9 00    548      lda #ZERO

```

```

D66A          549 ;
D66A          550 ;
D66A D0 2A    551 BCLEAR    bne RTN.D6.9          ; if more to statement, ignore
D66C          552 ;
D66C          553 ;
D66C A5 73    554 CLEARC    lda MEMSIZE
D66E A4 74    555           ldy MEMSIZE+1
D670          556 ;
D670 85 6F    557           sta FRETOP
D672 84 70    558           sty FRETOP+1
D674          559 ;
D674 A5 69    560           lda VARTAB
D676 A4 6A    561           ldy VARTAB+1
D678          562 ;
D678 85 6B    563           sta ARYTAB
D67A 84 6C    564           sty ARYTAB+1
D67C          565 ;
D67C 85 6D    566           sta STREND
D67E 84 6E    567           sty STREND+1
D680          568 ;
D680 20 49 D8 569           jsr BRESTORE
D683          570 ;
D683          571 ;
D683 A2 55    572 STKINIT   ldx #TEMPST
D685 86 52    573           stx TEMPPT
D687          574 ;
D687 68       575           pla
D688 A8       576           tay
D689          577 ;
D689 68       578           pla
D68A          579 ;
D68A A2 F8    580           ldx #$F8
D68C 9A       581           txs
D68D          582 ;
D68D 48       583           pha
D68E          584 ;
D68E 98       585           tya
D68F 48       586           pha
D690          587 ;
D690 A9 00    588           lda #ZERO
D692 85 7A    589           sta TEXTPTR+1
D694 85 14    590           sta SUBFLG
D696          591 ;
D696 60       592 RTN.D6.9 rts
D697          593 ;
D697          594 ;
D697          595 ; Set TXTPTR to the beginning of the program.
D697          596 ;
D697 18       597 STXTPTR   clc
D698          598 ;
D698 A5 67    599           lda PRGTAB
D69A 69 FF    600           adc #NEGONE
D69C 85 B8    601           sta TXTPTR
D69E          602 ;
D69E A5 68    603           lda PRGTAB+1
D6A0 69 FF    604           adc #NEGONE
D6A2 85 B9    605           sta TXTPTR+1
D6A4          606 ;
D6A4 60       607 RTN.D6.A rts
D6A5          608 ;
D6A5          609 ;

```

```

D6A5 90 0A      610  BLIST      bcc >1
D6A7 F0 08      611          beq >1
D6A9           612  ;
D6A9 C9 C9      613          cmp #TK.MINUS
D6AB F0 04      614          beq >1
D6AD           615  ;
D6AD C9 2C      616          cmp #', '
D6AF D0 F3      617          bne RTN.D6.A
D6B1           618  ;
D6B1 20 0C DA    619  ^1      jsr LINGET
D6B4 20 1A D6    620          jsr FNDLIN
D6B7           621  ;
D6B7 20 B7 00    622          jsr CHRGOT
D6BA F0 10      623          beq >3
D6BC           624  ;
D6BC C9 C9      625          cmp #TK.MINUS
D6BE F0 04      626          beq >2
D6C0           627  ;
D6C0 C9 2C      628          cmp #', '
D6C2 D0 E0      629          bne RTN.D6.A
D6C4           630  ;
D6C4 20 B1 00    631  ^2      jsr CHRGET
D6C7           632  ;
D6C7 20 0C DA    633          jsr LINGET
D6CA D0 D8      634          bne RTN.D6.A
D6CC           635  ;
D6CC 68         636  ^3      pla
D6CD 68         637          pla
D6CE           638  ;
D6CE A5 50      639          lda LINNUM
D6D0 05 51      640          ora LINNUM+1
D6D2 D0 06      641          bne LIST2
D6D4           642  ;
D6D4 A9 FF      643          lda #NEGONE
D6D6 85 50      644          sta LINNUM
D6D8 85 51      645          sta LINNUM+1
D6DA           646  ;
D6DA A0 01      647  LIST2    ldy #1
D6DC           648  ;
D6DC B1 9B      649          lda (LOWTR),Y
D6DE F0 41      650          beq LIST5
D6E0           651  ;
D6E0 20 58 D8    652          jsr ISCNTLC
D6E3 20 50 DB    653          jsr PRTCR
D6E6           654  ;
D6E6 C8         655          iny
D6E7           656  ;
D6E7 B1 9B      657          lda (LOWTR),Y
D6E9 AA         658          tax
D6EA           659  ;
D6EA C8         660          iny
D6EB           661  ;
D6EB B1 9B      662          lda (LOWTR),Y
D6ED C5 51      663          cmp LINNUM+1
D6EF D0 04      664          bne >4
D6F1           665  ;
D6F1 E4 50      666          cpx LINNUM
D6F3 F0 02      667          beq >5
D6F5           668  ;
D6F5 B0 2A      669  ^4      bcs LIST5
D6F7           670  ;

```

```

D6F7          671 ;
D6F7          672 ; List one line.
D6F7          673 ;
D6F7 84 85    674 ^5      sty FORPNT
D6F9          675 ;
D6F9 20 BE F7 676        jsr LISTEX1
D6FC          677 ;
D6FC A4 85    678 LIST3   ldy FORPNT
D6FE          679 ;
D6FE A9 20    680        lda #' '
D700          681 ;
D700 20 58 DB 682 LIST4   jsr OUTCHR
D703          683 ;
D703 20 C6 F7 684        jsr LISTEX2
D706 90 07    685        bcc >6
D708          686 ;
D708 20 50 DB 687        jsr PRTCR
D70B          688 ;
D70B A9 05    689        lda #5
D70D 85 24    690        sta CH
D70F          691 ;
D70F C8       692 ^6      iny
D710          693 ;
D710 B1 9B    694        lda (LOWTR),Y
D712 D0 13    695        bne LIST6
D714          696 ;
D714 A8       697        tay
D715          698 ;
D715 B1 9B    699        lda (LOWTR),Y
D717 AA       700        tax
D718          701 ;
D718 C8       702        iny
D719          703 ;
D719 B1 9B    704        lda (LOWTR),Y
D71B          705 ;
D71B 86 9B    706        stx LOWTR
D71D 85 9C    707        sta LOWTR+1
D71F          708 ;
D71F D0 B9    709        bne LIST2          ; branch if not end of program
D721          710 ;
D721 20 50 DB 711 LIST5   jsr PRTCR
D724          712 ;
D724 4C D2 D7 713        jmp NEWSTT
D727          714 ;
D727          715 ;
D727 10 D7    716 LIST6   bpl LIST4
D729          717 ;
D729 38       718        sec
D72A          719 ;
D72A E9 7F    720        sbc #$7F
D72C AA       721        tax
D72D          722 ;
D72D 84 85    723        sty FORPNT
D72F          724 ;
D72F A0 D4    725        ldy #BASNAME-PAGESIZE
D731 84 9D    726        sty DSCTMP
D733          727 ;
D733 A0 CF    728        ldy /BASNAME-PAGESIZE
D735 84 9E    729        sty DSCTMP+1
D737          730 ;
D737 A0 FF    731        ldy #NEGONE

```

```

D739          732 ;
D739 CA       733 ^1      dex
D73A F0 07    734          beq >3
D73C          735 ;
D73C 20 58 D7 736 ^2      jsr GETCHR
D73F 10 FB    737          bpl <2
D741          738 ;
D741 30 F6    739          bmi <1
D743          740 ;
D743 A9 20    741 ^3      lda #' '
D745 20 58 DB 742          jsr OUTCHR
D748          743 ;
D748 20 58 D7 744 ^4      jsr GETCHR
D74B 30 05    745          bmi >5
D74D          746 ;
D74D 20 58 DB 747          jsr OUTCHR
D750 D0 F6    748          bne <4
D752          749 ;
D752 20 58 DB 750 ^5      jsr OUTCHR
D755          751 ;
D755 4C FC D6 752          jmp LIST3
D758          753 ;
D758          754 ;
D758 C8       755 GETCHR   iny
D759 D0 02    756          bne >1
D75B          757 ;
D75B E6 9E    758          inc DSCTMP+1
D75D          759 ;
D75D B1 9D    760 ^1      lda (DSCTMP),Y
D75F          761 ;
D75F 60       762          rts
D760          763 ;
D760          764 ;
D760          765          dfs 6,ZERO          ; 6 bytes
D766          766 ;
D766          767 ;
D766          768 ; FOR statement pushes 18 bytes onto the STACK: 2 bytes
D766          769 ; for TXTPTR, 2 bytes for LINE NUMBER, 6 bytes for initial
D766          770 ; or current FOR variable value, 1 byte for STEP sign, 6
D766          771 ; bytes for STEP value, 2 bytes for address of FOR variable
D766          772 ; in VARTAB, and 1 byte for FOR token or 0x81.
D766          773 ;
D766 A9 80    774 BFOR     lda #$80          ; no subscripts
D768 85 14    775          sta SUBFLG
D76A          776 ;
D76A 20 46 DA 777          jsr BLET
D76D          778 ;
D76D 20 62 D3 779          jsr GTFORPNT
D770 D0 05    780          bne >1
D772          781 ;
D772 8A       782          txa
D773 69 0F    783          adc #15
D775 AA       784          tax
D776          785 ;
D776 9A       786          txs
D777          787 ;
D777 68       788 ^1      pla
D778 68       789          pla
D779          790 ;
D779 A9 0A    791          lda #FRAMSIZE/2
D77B 20 D6 D3 792          jsr CKSTKSIZ

```



```

D77E      793 ;
D77E 20 A3 D9 794      jsr DATSCAN
D781      795 ;
D781 18      796      clc
D782      797 ;
D782 98      798      tya
D783 65 B8    799      adc TXTPTR
D785 48      800      pha
D786      801 ;
D786 A5 B9    802      lda TXTPTR+1
D788 69 00    803      adc #ZERO
D78A 48      804      pha
D78B      805 ;
D78B A5 76    806      lda CURLIN+1
D78D 48      807      pha
D78E      808 ;
D78E A5 75    809      lda CURLIN
D790 48      810      pha
D791      811 ;
D791 A9 C1    812      lda #TK.TO
D793 20 C0 DE 813      jsr SYNTAXCHK
D796      814 ;
D796 20 67 DD 815      jsr CHKNUM
D799 20 64 DD 816      jsr FRMNUM
D79C      817 ;
D79C      818 ;
D79C      819 ; Put FAC into the format that is expected by LOADFAC.
D79C      820 ;
D79C A5 A2    821      lda FACSIGN
D79E 09 7F    822      ora #$7F
D7A0 25 9E    823      and FACMANT
D7A2 85 9E    824      sta FACMANT
D7A4      825 ;
D7A4 A9 AF    826      lda #STEP
D7A6 A0 D7    827      ldy /STEP
D7A8      828 ;
D7A8 85 5E    829      sta INDEX
D7AA 84 5F    830      sty INDEX+1
D7AC      831 ;
D7AC 4C 23 DE 832      jmp FRMSTAK3
D7AF      833 ;
D7AF      834 ;
D7AF A9 04    835 STEP   lda #FP1.0
D7B1 A0 EF    836      ldy /FP1.0
D7B3 20 F9 EA 837      jsr LOADFAC
D7B6      838 ;
D7B6 20 B7 00 839      jsr CHRGOT
D7B9 C9 C7    840      cmp #TK.STEP
D7BB D0 06    841      bne >1
D7BD      842 ;
D7BD 20 B1 00 843      jsr CHRGET
D7C0 20 64 DD 844      jsr FRMNUM
D7C3      845 ;
D7C3 20 82 EB 846 ^1     jsr SIGNCHK
D7C6 20 15 DE 847      jsr FRMSTAK2
D7C9      848 ;
D7C9 A5 86    849      lda FORPNT+1
D7CB 48      850      pha
D7CC      851 ;
D7CC A5 85    852      lda FORPNT
D7CE 48      853      pha

```

```

D7CF          854 ;
D7CF A9 81    855     lda #$80+BSFOR/2
D7D1 48       856     pha
D7D2          857 ;
D7D2          858 ;
D7D2          859 ; Used by DOS for the RUN and CHAIN commands as ASROMNEW.
D7D2          860 ; Used by FOR statement for NEXT statement.
D7D2          861 ;
D7D2          862 ASROMNEW:
D7D2 BA       863 NEWSTT     tsx
D7D3 86 F8    864     stx REMSTK
D7D5          865 ;
D7D5 20 58 D8 866     jsr ISCNTLC
D7D8          867 ;
D7D8 A5 B8    868     lda TXTPTR
D7DA A4 B9    869     ldy TXTPTR+1
D7DC          870 ;
D7DC A6 76    871     ldx CURLIN+1           ; MODE?
D7DE E8       872     inx
D7DF F0 04    873     beq >1           ; DIRECT
D7E1          874 ;
D7E1 85 79    875     sta TEXTPTR           ; RUNNING
D7E3 84 7A    876     sty TEXTPTR+1
D7E5          877 ;
D7E5 A0 00    878     ^1 ldy #ZERO
D7E7          879 ;
D7E7 B1 B8    880     lda (TXTPTR),Y
D7E9 D0 57    881     bne DOSTAMT3
D7EB          882 ;
D7EB 18       883     clc
D7EC          884 ;
D7EC A0 02    885     ldy #2
D7EE          886 ;
D7EE B1 B8    887     lda (TXTPTR),Y
D7F0 F0 34    888     beq GOEND3
D7F2          889 ;
D7F2 C8       890     iny
D7F3          891 ;
D7F3 B1 B8    892     lda (TXTPTR),Y
D7F5 85 75    893     sta CURLIN
D7F7          894 ;
D7F7 C8       895     iny
D7F8          896 ;
D7F8 B1 B8    897     lda (TXTPTR),Y
D7FA 85 76    898     sta CURLIN+1
D7FC          899 ;
D7FC 98       900     tya
D7FD          901 ;
D7FD 65 B8    902     adc TXTPTR
D7FF 85 B8    903     sta TXTPTR
D801 90 02    904     bcc DOTRACE
D803          905 ;
D803 E6 B9    906     inc TXTPTR+1
D805          907 ;
D805          908 ;
D805          909     icl "D8.L"

```

LLOAD D8.L,A\$4000

```

D805          1          ttl "ROM Source Code, D8.L"
D805          2          ;
D805          3          ;
D805          4          ; D8.L
D805          5          ;
D805          6          ;
D805 24 F2      7  DOTRACE  bit  TRACEFLG
D807 10 14      8          bpl >1
D809          9          ;
D809 A6 76     10         ldx CURLIN+1
D80B E8        11         inx
D80C F0 0F     12         beq >1
D80E          13         ;
D80E A9 23     14         lda #'#'
D810 20 58 DB  15         jsr OUTCHR
D813          16         ;
D813 A6 75     17         ldx CURLIN
D815 A5 76     18         lda CURLIN+1
D817          19         ;
D817 20 18 ED  20         jsr LINEPRT
D81A 20 53 DB  21         jsr OUTSPC
D81D          22         ;
D81D 20 B1 00  23         ^1  jsr CHRGET
D820 20 28 D8  24         jsr DOSTAMT
D823          25         ;
D823 4C D2 D7  26         jmp NEWSTT
D826          27         ;
D826          28         ;
D826 F0 62     29  GOEND3  beq END3          ; extended branch
D828          30         ;
D828          31         ;
D828          32         ; Separate BASADDR and FN1ADDR.
D828          33         ;
D828 F0 2D     34  DOSTAMT  beq RTN.D8.5      ; end of line?
D82A          35         ;
D82A E9 80     36  DOSTAMT2 sbc #$80          ; remove bias
D82C 90 11     37         bcc >1
D82E          38         ;
D82E C9 40     39         cmp #FN1ADDR/2
D830 B0 14     40         bcs >2
D832          41         ;
D832 0A        42         asl
D833 A8        43         tay
D834          44         ;
D834 B9 01 D0  45         lda BASADDR+1,Y
D837 48        46         pha
D838          47         ;
D838 B9 00 D0  48         lda BASADDR,Y
D83B 48        49         pha
D83C          50         ;
D83C 4C B1 00  51         jmp CHRGET
D83F          52         ;
D83F 4C 46 DA  53         ^1  jmp BLET
D842          54         ;
D842 C9 3A     55  DOSTAMT3 cmp #' ':'      ; separator
D844 F0 BF     56         beq DOTRACE
D846          57         ;
D846 4C C9 DE  58         ^2  jmp SY.ERR
D849          59         ;
D849          60         ;

```

```

D849 38          61  BRESTORE sec
D84A          62  ;
D84A A5 67      63          lda PRGTAB
D84C E9 01      64          sbc #1
D84E          65  ;
D84E A4 68      66          ld y PRGTAB+1
D850          67  ;
D850 B0 01      68          bcs SETDAPTR
D852          69  ;
D852 88          70          dey
D853          71  ;
D853 85 7D      72  SETDAPTR sta DATPTR
D855 84 7E      73          sty DATPTR+1
D857          74  ;
D857 60          75  RTN.D8.5 rts
D858          76  ;
D858          77  ;
D858          78  ; Modified the ISCNTLC routine by moving the contents of
D858          79  ; INCHR to 0xD860 and moving the branch to DOHANDLR.
D858          80  ;
D858 AD 00 C0   81  ISCNTLC  lda KEY
D85B C9 83      82          cmp #CTRLC
D85D F0 01      83          beq >1
D85F          84  ;
D85F 60          85          rts
D860          86  ;
D860 2C 10 C0   87  ^1      bit CLRKEY
D863          88  ;
D863 A2 FF      89  DOCTRL.C ldx #ERROR.2          ; ctrl-C input error
D865          90  ;
D865          91  ;
D865          92  ; This is the DOS default ERRORADR.
D865          93  ;
D865 24 D8      94  ASROMERR bit ERRFLG
D867 30 47      95          bmi DOHANDLR
D869          96  ;
D869 EA          97          nop
D86A          98  ;
D86A 29 7F      99          and #MSBCLR
D86C C9 03      100         cmp #CTRLC&MSBCLR          ; set C=1, Z=1
D86E          101 ;
D86E          102 ;
D86E B0 01      103  BSTOP   bcs END2
D870          104 ;
D870          105 ;
D870 18          106  BEND    clc
D871          107 ;
D871 D0 3C      108  END2    bne RTN.D8.A
D873          109 ;
D873 A5 B8      110         lda TXTPTR
D875 A4 B9      111         ld y TXTPTR+1
D877          112 ;
D877 A6 76      113         ldx CURLIN+1
D879 E8          114         inx
D87A F0 0C      115         beq >1
D87C          116 ;
D87C 85 79      117         sta TEXTPTR
D87E 84 7A      118         sty TEXTPTR+1
D880          119 ;
D880 A5 75      120         lda CURLIN
D882 A4 76      121         ld y CURLIN+1

```

```

D884          122 ;
D884 85 77    123      sta OLDLIN
D886 84 78    124      sty OLDLIN+1
D888          125 ;
D888 68      126 ^1    pla
D889 68      127      pla
D88A          128 ;
D88A 90 07    129 END3   bcc >2
D88C          130 ;
D88C A9 5A    131      lda #MSG20
D88E A0 D3    132      ldy /MSG20
D890          133 ;
D890 4C 31 D4 134      jmp PRLINUM
D893          135 ;
D893 4C 3C D4 136 ^2    jmp RESTART
D896          137 ;
D896          138 ;
D896 D0 17    139 BCONT   bne RTN.D8.A
D898          140 ;
D898 A2 D6    141      ldx #MSG16-MESGS ; Cannot Continue error
D89A          142 ;
D89A A4 7A    143      ldy TEXTPTR+1
D89C D0 03    144      bne >1
D89E          145 ;
D89E 4C 12 D4 146      jmp PRERR
D8A1          147 ;
D8A1 A5 79    148 ^1    lda TEXTPTR
D8A3          149 ;
D8A3 85 B8    150      sta TXTPTR
D8A5 84 B9    151      sty TXTPTR+1
D8A7          152 ;
D8A7 A5 77    153      lda OLDLIN
D8A9 A4 78    154      ldy OLDLIN+1
D8AB          155 ;
D8AB 85 75    156      sta CURLIN
D8AD 84 76    157      sty CURLIN+1
D8AF          158 ;
D8AF 60      159 RTN.D8.A rts
D8B0          160 ;
D8B0          161 ;
D8B0 4C E9 F2 162 DOHANDLR jmp HANDLERR
D8B3          163 ;
D8B3          164 ;
D8B3 68      165 PULL3A   pla
D8B4 68      166      pla
D8B5 68      167      pla
D8B6          168 ;
D8B6 60      169      rts
D8B7          170 ;
D8B7          171 ;
D8B7          172      dfs 4,ZERO ; 4 bytes
D8BB          173 ;
D8BB          174 ;
D8BB          175 ; Used by CXREAD to read audio waveforms.
D8BB          176 ;
D8BB A2 08    177 RDBYTE   ldx #8 ; bit counter
D8BD          178 ;
D8BD 48      179 ^1    pha ; push current data
D8BE          180 ;
D8BE 20 FF D8 181      jsr RD2BIT ; get data bit in C-flag
D8C1          182 ;

```

```

D8C1 68          183          pla          ; recall data
D8C2 2A          184          rol          ; roll in bit, MSB first
D8C3            185          ;
D8C3 A0 3A       186          ldy #$3A      ; waveform change, 696 cycles
D8C5            187          ;
D8C5 CA          188          dex          ; next bit
D8C6 D0 F5       189          bne <1
D8C8            190          ;
D8C8 60          191          rts
D8C9            192          ;
D8C9            193          ;
D8C9            194          ; Applesoft LOAD command for c2t processing. Copy address
D8C9            195          ; of LINNUM to A1 for start address. Copy address of
D8C9            196          ; LINNUM+2 to A2 for end address. Thus, read two data
D8C9            197          ; bytes, the RUNFLAG, and a checksum.
D8C9            198          ;
D8C9 A9 50       199 BLOAD    lda #LINNUM
D8CB A0 00       200          ldy /LINNUM
D8CD            201          ;
D8CD 85 3C       202          sta A1L
D8CF 84 3D       203          sty A1H
D8D1            204          ;
D8D1 A9 52       205          lda #LINNUM+2
D8D3            206          ;
D8D3 85 3E       207          sta A2L
D8D5 84 3F       208          sty A2H
D8D7            209          ;
D8D7 84 D6       210          sty RUNFLAG
D8D9            211          ;
D8D9 20 9C F3    212          jsr CXREAD
D8DC            213          ;
D8DC            214          ;
D8DC            215          ; Copy address in PRGTAB to A1 for start address. Add
D8DC            216          ; address in PRGTAB to LINNUM for VARTAB and A2 for end
D8DC            217          ; address. Thus, read LINNUM bytes and a checksum.
D8DC            218          ;
D8DC 18          219          clc
D8DD            220          ;
D8DD A5 67       221          lda PRGTAB
D8DF 85 3C       222          sta A1L
D8E1            223          ;
D8E1 65 50       224          adc LINNUM
D8E3 85 69       225          sta VARTAB
D8E5 85 3E       226          sta A2L
D8E7            227          ;
D8E7 A5 68       228          lda PRGTAB+1
D8E9 85 3D       229          sta A1H
D8EB            230          ;
D8EB 65 51       231          adc LINNUM+1
D8ED 85 6A       232          sta VARTAB+1
D8EF 85 3F       233          sta A2H
D8F1            234          ;
D8F1 A5 52       235          lda TEMPPT
D8F3 85 D6       236          sta RUNFLAG
D8F5            237          ;
D8F5 20 9C F3    238          jsr CXREAD          ; read Applesoft program
D8F8            239          ;
D8F8 24 D6       240          bit RUNFLAG
D8FA 30 22       241          bmi RUN2
D8FC            242          ;
D8FC 4C F2 D4    243          jmp ASENTER          ; enter Applesoft interpreter

```

```

D8FF      244 ;
D8FF      245 ;
D8FF      246 ; Read audio waveform for two transitions and return bit
D8FF      247 ; value in the C-flag.
D8FF      248 ;
D8FF 20 02 D9 249 RD2BIT    jsr RDBIT
D902      250 ;
D902 88      251 RDBIT     dey
D903      252 ;
D903 AD 60 C0 253         lda TAPEIN          ; read waveform voltage level
D906 45 2F    254         eor LASTIN         ; compare MSB to last read
D908 10 F8    255         bpl RDBIT          ; branch if no change
D90A      256 ;
D90A 45 2F    257         eor LASTIN         ; compare MSB to last saved
D90C 85 2F    258         sta LASTIN         ; save the new MSB
D90E      259 ;
D90E C0 80    260         cpy #$80           ; get bit value into C-flag
D910      261 ;
D910 60      262         rts
D911      263 ;
D911      264 ;
D911      265         dfs 1,ZERO             ; 1 byte
D912      266 ;
D912      267 ;
D912 08      268 BRUN     php
D913      269 ;
D913 C6 76    270         dec CURLIN+1
D915      271 ;
D915 28      272         plp
D916 F0 06    273         beq RUN2
D918      274 ;
D918 20 6C D6 275         jsr CLEARC
D91B      276 ;
D91B 4C 35 D9 277         jmp GOSUB2
D91E      278 ;
D91E 4C 65 D6 279 RUN2    jmp SETPTRS          ; run the Applesoft program
D921      280 ;
D921      281 ;
D921      282 ; GOSUB pushes 7 bytes onto the STACK: 2 bytes for NEWSTT
D921      283 ; return address, 2 bytes for TXTPTR, 2 bytes for LINE
D921      284 ; number, and 1 byte for GOSUB token or 0xB0.
D921      285 ;
D921 A9 03    286 BGOSUB   lda #3
D923 20 D6 D3 287         jsr CKSTKSIZ
D926      288 ;
D926 A5 B9    289         lda TXTPTR+1
D928 48      290         pha
D929      291 ;
D929 A5 B8    292         lda TXTPTR
D92B 48      293         pha
D92C      294 ;
D92C A5 76    295         lda CURLIN+1
D92E 48      296         pha
D92F      297 ;
D92F A5 75    298         lda CURLIN
D931 48      299         pha
D932      300 ;
D932 A9 B0    301         lda #$80+BSGOSUB/2
D934 48      302         pha
D935      303 ;
D935 20 B7 00 304 GOSUB2   jsr CHRGOT

```

```

D938 20 3E D9      305      jsr BGOTO
D93B              306      ;
D93B 4C D2 D7      307      jmp NEWSTT
D93E              308      ;
D93E              309      ;
D93E              310      ; The GOTO statment is used by RUN and GOSUB.
D93E              311      ;
D93E 20 0C DA      312      BGOTO      jsr LINGET
D941 20 A6 D9      313      jsr DATSCAN2
D944              314      ;
D944 A5 76         315      lda CURLIN+1
D946 C5 51         316      cmp LINNUM+1
D948 B0 0B         317      bcs >1
D94A              318      ;
D94A 38            319      sec                      ; plus an additional 1
D94B              320      ;
D94B A6 B9         321      ldx TXTPTR+1
D94D              322      ;
D94D 98            323      tya
D94E 65 B8         324      adc TXTPTR
D950 90 07         325      bcc >2
D952              326      ;
D952 E8            327      inx
D953              328      ;
D953 B0 04         329      bcs >2                      ; always taken
D955              330      ;
D955              331      ;
D955              332      ; Entry point that is used by DOS for the RUN and CHAIN
D955              333      ; commands as ASROMSET to initialize the start of Applesoft
D955              334      ; when LINNUM is specified in these two DOS commands.
D955              335      ;
D955              336      ASROMSET:
D955 A5 67         337      ^1      lda PRGTAB
D957 A6 68         338      ldx PRGTAB+1
D959              339      ;
D959 20 1E D6      340      ^2      jsr FNDLIN2
D95C 90 1E         341      bcc US.ERR
D95E              342      ;
D95E A5 9B         343      lda LOWTR
D960 E9 01         344      sbc #1
D962 85 B8         345      sta TXTPTR
D964              346      ;
D964 A5 9C         347      lda LOWTR+1
D966 E9 00         348      sbc #ZERO
D968 85 B9         349      sta TXTPTR+1
D96A              350      ;
D96A 60            351      RTN.D9.6 rts
D96B              352      ;
D96B              353      ;
D96B              354      ; Entry for POP and RETURN.  Fixed FORPNT bug.
D96B              355      ;
D96B              356      BRETURN:
D96B D0 FD         357      BPOP      bne RTN.D9.6
D96D              358      ;
D96D A9 FF         359      lda #NEGONE
D96F 85 86         360      sta FORPNT+1
D971              361      ;
D971 20 62 D3      362      jsr GTFORPNT
D974              363      ;
D974 9A            364      txs
D975              365      ;

```



```

D975 C9 B0      366      cmp #$80+BSGOSUB/2
D977 F0 0B      367      beq >1
D979           368      ;
D979           369      ;
D979 A2 16      370      ldx #MSG03-MESGS      ; RETURN without GOSUB error
D97B           371      ;
D97B 2C 00 00   372      bit *-*
D97E           373      dfs !-2
D97C           374      ;
D97C A2 5A      375 US.ERR ldx #MSG08-MESGS      ; Undefined Statement error
D97E           376      ;
D97E 4C 12 D4   377      jmp PRterr
D981           378      ;
D981           379      ;
D981 4C B3 D8   380 PULL3  jmp PULL3A
D984           381      ;
D984           382      ;
D984 68         383      ^1      pla
D985 68         384      pla
D986           385      ;
D986 C0 42      386      cpy #2*$80+BSPOP/2
D988 F0 F7      387      beq PULL3
D98A           388      ;
D98A 85 75      389      sta CURLIN
D98C           390      ;
D98C 68         391      pla
D98D 85 76      392      sta CURLIN+1
D98F           393      ;
D98F 68         394      pla
D990 85 B8      395      sta TXTPTR
D992           396      ;
D992 68         397      pla
D993 85 B9      398      sta TXTPTR+1
D995           399      ;
D995           400      ;
D995           401      ; In order to fix the dangerous code at 0xDA1C, SY.ERR2
D995           402      ; and PULL3 were swapped.
D995           403      ;
D995 20 A3 D9   404 BDATA  jsr DATSCAN
D998           405      ;
D998 18         406 DATA2  clc
D999           407      ;
D999 98         408      tya
D99A 65 B8      409      adc TXTPTR
D99C 85 B8      410      sta TXTPTR
D99E 90 02      411      bcc RTN.D9.A
D9A0           412      ;
D9A0 E6 B9      413      inc TXTPTR+1
D9A2           414      ;
D9A2 60         415 RTN.D9.A rts
D9A3           416      ;
D9A3           417      ;
D9A3           418      ; Scan ahead to next ":" or EOL.
D9A3           419      ;
D9A3 A2 3A      420 DATSCAN ldx #'`
D9A5           421      ;
D9A5 2C 00 00   422      bit *-*
D9A8           423      dfs !-2
D9A6           424      ;
D9A6 A2 00      425 DATSCAN2 ldx #ZERO
D9A8           426      ;

```

```

D9A8 86 0D      427      stx CHARAC
D9AA           428      ;
D9AA A0 00      429      ldy #ZERO
D9AC 84 0E      430      sty ENDCHR
D9AE           431      ;
D9AE A5 0E      432      ^1      lda ENDCHR
D9B0 A6 0D      433      ldx CHARAC
D9B2           434      ;
D9B2 85 0D      435      sta CHARAC
D9B4 86 0E      436      stx ENDCHR
D9B6           437      ;
D9B6 B1 B8      438      ^2      lda (TXTPTR),Y
D9B8 F0 E8      439      beq RTN.D9.A
D9BA           440      ;
D9BA C5 0E      441      cmp ENDCHR
D9BC F0 E4      442      beq RTN.D9.A
D9BE           443      ;
D9BE C8         444      iny
D9BF           445      ;
D9BF C9 22      446      cmp #'"'
D9C1 D0 F3      447      bne <2
D9C3           448      ;
D9C3 F0 E9      449      beq <1                ; always taken
D9C5           450      ;
D9C5           451      ;
D9C5 4C C9 DE    452      SY.ERR2      jmp SY.ERR
D9C8           453      ;
D9C8           454      ;
D9C8           455      dfs 1,ZERO                ; 1 byte
D9C9           456      ;
D9C9           457      ;
D9C9 20 7B DD    458      BIF          jsr FRMEVAL
D9CC 20 B7 00    459      jsr CHRGOT
D9CF           460      ;
D9CF C9 AB      461      cmp #$80+BSGOTO/2
D9D1 F0 05      462      beq >1
D9D3           463      ;
D9D3 A9 C4      464      lda #TK.THEN
D9D5 20 C0 DE    465      jsr SYNTAXCHK
D9D8           466      ;
D9D8 A5 9D      467      ^1      lda DSCTMP
D9DA D0 05      468      bne REM2
D9DC           469      ;
D9DC           470      ;
D9DC 20 A6 D9    471      BREM          jsr DATSCAN2
D9DF F0 B7      472      beq DATA2
D9E1           473      ;
D9E1 20 B7 00    474      REM2          jsr CHRGOT
D9E4 B0 03      475      bcs >1
D9E6           476      ;
D9E6 4C 3E D9    477      jmp BGOTO
D9E9           478      ;
D9E9 4C 28 D8    479      ^1      jmp DOSTAMT
D9EC           480      ;
D9EC           481      ;
D9EC 20 F8 E6    482      BON          jsr GETBYT
D9EF           483      ;
D9EF 48         484      pha
D9F0           485      ;
D9F0 C9 B0      486      cmp #$80+BSGOSUB/2
D9F2 F0 04      487      beq >1

```

```

D9F4          488 ;
D9F4 C9 AB    489      cmp #$80+BSGOTO/2
D9F6 D0 CD    490      bne SY.ERR2
D9F8          491 ;
D9F8 C6 A1    492 ^1    dec FACMANT+3
D9FA D0 04    493      bne >2
D9FC          494 ;
D9FC 68       495      pla
D9FD          496 ;
D9FD 4C 2A D8 497      jmp DOSTAMT2
DA00          498 ;
DA00 20 B1 00 499 ^2    jsr CHRGET
DA03 20 0C DA 500      jsr LINGET
DA06          501 ;
DA06 C9 2C    502      cmp #', '
DA08 F0 EE    503      beq <1
DA0A          504 ;
DA0A 68       505      pla
DA0B          506 ;
DA0B 60       507      RTN.DA.0 rts
DA0C          508 ;
DA0C          509 ;
DA0C          510 ; Convert line number.
DA0C          511 ;
DA0C A2 00    512      LINGET    ldx #ZERO
DA0E 86 50    513          stx LINNUM
DA10 86 51    514          stx LINNUM+1
DA12          515 ;
DA12 B0 F7    516      LINGET2   bcs RTN.DA.0
DA14          517 ;
DA14 E9 2F    518          sbc #'0'-1
DA16 85 0D    519          sta CHARAC
DA18          520 ;
DA18 A5 51    521          lda LINNUM+1
DA1A 85 5E    522          sta INDEX
DA1C          523 ;
DA1C C9 19    524          cmp /6400          ; fixed this dangerous code
DA1E B0 A5    525          bcs SY.ERR2      ; moved entry within reach
DA20          526 ;
DA20 A5 50    527          lda LINNUM
DA22          528 ;
DA22 0A       529          asl
DA23 26 5E    530          rol INDEX
DA25          531 ;
DA25 0A       532          asl
DA26 26 5E    533          rol INDEX
DA28          534 ;
DA28 65 50    535          adc LINNUM
DA2A 85 50    536          sta LINNUM
DA2C          537 ;
DA2C A5 5E    538          lda INDEX
DA2E 65 51    539          adc LINNUM+1
DA30 85 51    540          sta LINNUM+1
DA32          541 ;
DA32 06 50    542          asl LINNUM
DA34 26 51    543          rol LINNUM+1
DA36          544 ;
DA36 A5 50    545          lda LINNUM
DA38 65 0D    546          adc CHARAC
DA3A 85 50    547          sta LINNUM
DA3C 90 02    548          bcc >1

```

```

DA3E          549 ;
DA3E E6 51    550      inc LINNUM+1
DA40          551 ;
DA40 20 B1 00 552 ^1    jsr CHRGET
DA43          553 ;
DA43 4C 12 DA 554      jmp LINGET2
DA46          555 ;
DA46          556 ;
DA46 20 E3 DF 557 BLET   jsr PTRGET
DA49          558 ;
DA49 85 85     559      sta FORPNT
DA4B 84 86     560      sty FORPNT+1
DA4D          561 ;
DA4D A9 D0     562      lda #TK.EQUAL
DA4F 20 C0 DE  563      jsr SYNTAXCHK
DA52          564 ;
DA52 A5 12     565      lda VALTYP+1
DA54 48        566      pha
DA55          567 ;
DA55 A5 11     568      lda VALTYP
DA57 48        569      pha
DA58          570 ;
DA58 20 7B DD  571      jsr FRMEVAL
DA5B          572 ;
DA5B 68        573      pla
DA5C 2A        574      rol
DA5D          575 ;
DA5D 20 6A DD  576      jsr CHKVAL
DA60 D0 18     577      bne >2
DA62          578 ;
DA62 68        579      pla
DA63          580 ;
DA63 10 12     581 LET2   bpl >1      ; branch if real variable
DA65          582 ;
DA65 20 70 EB  583      jsr RNDUP      ; integer, so roundup
DA68 20 16 E1  584      jsr AYINT
DA6B          585 ;
DA6B A0 00     586      ldy #ZERO
DA6D          587 ;
DA6D A5 A0     588      lda FACMANT+2
DA6F 91 85     589      sta (FORPNT),Y
DA71          590 ;
DA71 C8        591      iny
DA72          592 ;
DA72 A5 A1     593      lda FACMANT+3
DA74 91 85     594      sta (FORPNT),Y
DA76          595 ;
DA76 60        596      rts
DA77          597 ;
DA77 4C 27 EB  598 ^1    jmp COPYF2FR
DA7A          599 ;
DA7A          600 ;
DA7A          601 ; A string variable.
DA7A          602 ;
DA7A 68        603 ^2    pla
DA7B          604 ;
DA7B          605 ;
DA7B A0 02     606 PUTSTR  ldy #2
DA7D          607 ;
DA7D B1 A0     608      lda (FACMANT+2),Y
DA7F C5 70     609      cmp FRETOP+1

```

```

DA81 90 17      610      bcc >2
DA83           611      ;
DA83 D0 07      612      bne >1
DA85           613      ;
DA85 88         614      dey
DA86           615      ;
DA86 B1 A0      616      lda (FACMANT+2),Y
DA88 C5 6F      617      cmp FRETOP
DA8A 90 0E      618      bcc >2
DA8C           619      ;
DA8C A4 A1      620      ^1      ldy FACMANT+3
DA8E C4 6A      621      cpy VARTAB+1
DA90 90 08      622      bcc >2
DA92           623      ;
DA92 D0 0D      624      bne >3
DA94           625      ;
DA94 A5 A0      626      lda FACMANT+2
DA96 C5 69      627      cmp VARTAB
DA98 B0 07      628      bcs >3
DA9A           629      ;
DA9A           630      ;
DA9A           631      ; The C-flag is always clear at this entry point.
DA9A           632      ;
DA9A A5 A0      633      ^2      lda FACMANT+2
DA9C A4 A1      634      ldy FACMANT+3
DA9E           635      ;
DA9E 4C B7 DA   636      jmp COPYSTR
DAA1           637      ;
DAA1 A0 00      638      ^3      ldy #ZERO
DAA3           639      ;
DAA3 B1 A0      640      lda (FACMANT+2),Y
DAA5 20 D0 E3   641      jsr STRINI
DAA8           642      ;
DAA8 A5 8C      643      lda DSCPTR
DAAA A4 8D      644      ldy DSCPTR+1
DAAC           645      ;
DAAC 85 AB      646      sta STRING1
DAAE 84 AC      647      sty STRING1+1
DAB0           648      ;
DAB0 20 D4 E5   649      jsr MOVINS
DAB3           650      ;
DAB3 A9 9D      651      lda #FACEXP
DAB5 A0 00      652      ldy /FACEXP
DAB7           653      ;
DAB7           654      ;
DAB7 85 8C      655      COPYSTR sta DSCPTR
DAB9 84 8D      656      sty DSCPTR+1
DABB           657      ;
DABB 20 35 E6   658      jsr FRETMS
DABE           659      ;
DABE A0 00      660      ldy #ZERO
DAC0           661      ;
DAC0 B1 8C      662      lda (DSCPTR),Y
DAC2 91 85      663      sta (FORPNT),Y
DAC4           664      ;
DAC4 C8         665      iny
DAC5           666      ;
DAC5 B1 8C      667      lda (DSCPTR),Y
DAC7 91 85      668      sta (FORPNT),Y
DAC9           669      ;
DAC9 C8         670      iny

```

```

DACA          671 ;
DACA B1 8C    672         lda (DSCPTR),Y
DACC 91 85    673         sta (FORPNT),Y
DACE          674 ;
DACE 60       675 RTN.DA.C rts
DACF          676 ;
DACF          677 ;
DACF 20 3E DB 678 PRSTRING jsr STRPRT
DAD2 20 B7 00 679         jsr CHRGOT
DAD5          680 ;
DAD5          681 ;
DAD5 F0 79    682 BPRINT   beq PRTCR
DAD7          683 ;
DAD7 F0 F5    684 PRINT2   beq RTN.DA.C
DAD9          685 ;
DAD9 C9 C0    686         cmp #TK.TAB
DADB F0 33    687         beq >3
DADD          688 ;
DADD C9 C3    689         cmp #TK.SPC
DADF          690 ;
DADF 18       691         clc
DAE0          692 ;
DAE0 F0 2E    693         beq >3
DAE2          694 ;
DAE2 C9 2C    695         cmp #', '
DAE4 F0 14    696         beq >1
DAE6          697 ;
DAE6 C9 3B    698         cmp #'; '
DAE8 F0 3C    699         beq >7
DAEA          700 ;
DAEA 20 7B DD 701         jsr FRMEVAL
DAED          702 ;
DAED 24 11    703         bit VALTYP
DAEF 30 DE    704         bmi PRSTRING
DAF1          705 ;
DAF1 20 34 ED 706         jsr FPOUT
DAF4 20 E2 E3 707         jsr STRLIT
DAF7          708 ;
DAF7 4C CF DA 709         jmp PRSTRING
DAFA          710 ;
DAFA 20 C6 F7 711 ^1      jsr PRTCREX1
DAFD 30 09    712         bmi >2
DAFF          713 ;
DAFF C9 20    714         cmp #32           ; bug, not 24
DB01 90 05    715         bcc >2
DB03          716 ;
DB03 20 50 DB 717         jsr PRTCR
DB06 D0 1E    718         bne >7           ; always taken
DB08          719 ;
DB08 69 10    720 ^2      adc #$10
DB0A 29 F0    721         and #$F0           ; round to 16 or 32
DB0C AA       722         tax
DB0D          723 ;
DB0D 38       724         sec
DB0E B0 0C    725         bcs >4           ; always taken
DB10          726 ;
DB10 08       727 ^3      php
DB11          728 ;
DB11 20 F5 E6 729         jsr GETBYTC
DB14          730 ;
DB14 C9 29    731         cmp #') '

```

```

DB16 D0 69      732      bne SY.ERR3
DB18            733      ;
DB18 28         734      plp
DB19 90 07      735      bcc >5
DB1B           736      ;
DB1B CA        737      dex
DB1C           738      ;
DB1C 20 D5 F7   739      ^4      jsr PRTCRESX2
DB1F 90 05      740      bcc >7
DB21           741      ;
DB21 AA        742      tax
DB22           743      ;
DB22 E8        744      ^5      inx
DB23           745      ;
DB23 CA        746      ^6      dex
DB24 D0 06      747      bne >8
DB26           748      ;
DB26 20 B1 00   749      ^7      jsr CHRGET
DB29           750      ;
DB29 4C D7 DA   751      jmp PRINT2
DB2C           752      ;
DB2C 20 53 DB   753      ^8      jsr OUTSPC
DB2F D0 F2      754      bne <6      ; always taken
DB31           755      ;
DB31           756      ;
DB31           757      dfs 1,ZERO      ; 1 byte
DB32           758      ;
DB32           759      ;
DB32 A8        760      UNARY2      tay
DB33           761      ;
DB33 8A        762      txa
DB34 48        763      pha
DB35           764      ;
DB35 4C 3F DF   765      jmp UNARY3
DB38           766      ;
DB38           767      ;
DB38 20 34 ED   768      LINEOUT      jsr FPOUT
DB3B           769      ;
DB3B 20 E2 E3   770      STROUT      jsr STRLIT
DB3E           771      ;
DB3E           772      ;
DB3E           773      ; Removed unnecessary logic, reorganized, and rewrote.
DB3E           774      ;
DB3E 20 00 E6   775      STRPRT      jsr FREFAC
DB41           776      ;
DB41 AA        777      tax
DB42 F0 0B      778      beq >2
DB44           779      ;
DB44 A0 00      780      ldy #ZERO
DB46           781      ;
DB46 B1 5E      782      ^1      lda (INDEX),Y
DB48 20 58 DB   783      jsr OUTCHR
DB4B           784      ;
DB4B C8        785      iny
DB4C           786      ;
DB4C CA        787      dex
DB4D D0 F7      788      bne <1
DB4F           789      ;
DB4F 60        790      ^2      rts
DB50           791      ;
DB50           792      ;

```

```

DB50 A9 0D      793 PRTCR      lda #RETURN&MSBCLR    ; removed unnecessary NEGATE
DB52           794 ;
DB52 2C 00 00   795           bit *-*
DB55           796           dfs !-2
DB53           797 ;
DB53           798 ;
DB53 A9 20      799 OUTSPC     lda #' '
DB55           800 ;
DB55 2C 00 00   801           bit *-*
DB58           802           dfs !-2
DB56           803 ;
DB56           804 ;
DB56           805 OUTPROMT:
DB56           806 ;           lda #'?'
DB56 A9 3E      807           lda #'>'
DB58           808 ;
DB58           809 ;
DB58 09 80      810 OUTCHR     ora #MSBSET
DB5A           811 ;
DB5A C9 A0      812           cmp #SPACE
DB5C 90 02      813           bcc >1
DB5E           814 ;
DB5E 05 F3      815           ora FLASHBYT
DB60           816 ;
DB60 20 ED FD   817 ^1       jsr COUT
DB63           818 ;
DB63 29 7F      819           and #MSBCLR
DB65 48         820           pha
DB66           821 ;
DB66 A5 F1      822           lda SPEEDBYT        ; modified logic
DB68 F0 03      823           beq >2
DB6A           824 ;
DB6A 20 A8 FC   825           jsr WAIT
DB6D           826 ;
DB6D 68         827 ^2       pla
DB6E           828 ;
DB6E 60         829           rts
DB6F           830 ;
DB6F           831 ;
DB6F           832 ; Illegal character found in numeric field. Must determine
DB6F           833 ; if error in INPUT, READ, or GET.
DB6F           834 ;
DB6F A5 15      835 INPUTERR  lda INPUTFLG
DB71 F0 14      836           beq RESPERR        ; INPUT
DB73           837 ;
DB73 30 04      838           bmi READERR        ; READ
DB75           839 ;
DB75 A0 FF      840           ldy #NEGONE        ; GET
DB77 D0 04      841           bne ERRLINN      ; always taken
DB79           842 ;
DB79 A5 7B      843 READERR   lda DATLIN
DB7B A4 7C      844           ldy DATLIN+1
DB7D           845 ;
DB7D 85 75      846 ERRLINN   sta CURLIN
DB7F 84 76      847           sty CURLIN+1
DB81           848 ;
DB81 A2 10      849 SY.ERR3   ldx #MSG02-MESGS    ; Syntax error
DB83           850 ;
DB83 4C 12 D4   851           jmp PRTER
DB86           852 ;
DB86           853 ;

```



```

DB86 68          854 INPERR   pla
DB87             855 ;
DB87 24 D8       856 RESPERR bit ERRFLG
DB89 10 05        857 bpl >1
DB8B             858 ;
DB8B A2 FE        859         ldx #ERROR.1          ; bad response to INPUT
DB8D             860 ;
DB8D 4C E9 F2     861         jmp HANDLERR
DB90             862 ;
DB90 A9 EF        863 ^1      lda #MSG22          ; 'Reenter'
DB92 A0 DC        864         ldy /MSG22
DB94             865 ;
DB94 20 3B DB     866         jsr STROUT
DB97             867 ;
DB97 A5 79        868         lda TEXTPTR
DB99 A4 7A        869         ldy TEXTPTR+1
DB9B             870 ;
DB9B 85 B8        871         sta TXTPTR
DB9D 84 B9        872         sty TXTPTR+1
DB9F             873 ;
DB9F 60           874         rts
DBA0             875 ;
DBA0             876 ;
DBA0 20 06 E3     877 BGET    jsr CHKIFDIR
DBA3             878 ;
DBA3 A2 01        879         ldx #INPUT+1
DBA5 A0 02        880         ldy /INPUT+1
DBA7             881 ;
DBA7 A9 00        882         lda #ZERO
DBA9 8D 01 02     883         sta INPUT+1
DBAC             884 ;
DBAC A9 40        885         lda #%01000000      ; set INPUTFLG (GET)
DBAE             886 ;
DBAE 4C EB DB     887         jmp INPTLIST          ; accelerate code
DBB1             888 ;
DBB1             889 ;
DBB1             890         dfs 1,ZERO          ; 1 byte
DBB2             891 ;
DBB2             892 ;
DBB2             893 ; STRPRT rewritten so able to branch rather than jump.
DBB2             894 ;
DBB2 C9 22        895 BINPUT  cmp #' '
DBB4 D0 0D        896         bne >1
DBB6             897 ;
DBB6 20 81 DE     898         jsr STRTXT
DBB9             899 ;
DBB9 A9 3B        900         lda #';'
DBBB 20 C0 DE     901         jsr SYNTAXCHK
DBBE             902 ;
DBBE 20 3E DB     903         jsr STRPRT
DBC1 F0 03        904         beq >2          ; always taken
DBC3             905 ;
DBC3 20 56 DB     906 ^1      jsr OUTPROMT
DBC6             907 ;
DBC6 20 06 E3     908 ^2      jsr CHKIFDIR
DBC9             909 ;
DBC9 A9 2C        910         lda #', '
DBCb 8D FF 01     911         sta INPUT-1
DBCE             912 ;
DBCE 20 2C D5     913         jsr INLIN
DBD1             914 ;

```

```
DBD1 AD 00 02    915          lda INPUT
DBD4 C9 03      916          cmp #CTRLC&MSBCLR
DBD6 D0 11      917          bne READ2
DBD8           918          ;
DBD8 4C 63 D8   919          jmp DOCTRL.C
DBDB           920          ;
DBDB           921          ;
DBDB           922          dfs 1,ZERO          ; 1 byte
DBDC           923          ;
DBDC           924          ;
DBDC 20 56 DB   925  HEXTIN    jsr OUTPROMT
DBDF           926          ;
DBDF 4C 2C D5   927          jmp INLIN
DBE2           928          ;
DBE2           929          ;
DBE2 A6 7D     930  BREAD      ldx DATPTR
DBE4 A4 7E     931          ldy DATPTR+1
DBE6           932          ;
DBE6 A9 98     933          lda #%10011000      ; set INPUTFLG (READ)
DBE8           934          ;
DBE8 2C 00 00  935          bit *-*
DBEB           936          dfs !-2
DBE9           937          ;
DBE9 A9 00     938  READ2      lda #%00000000      ; set INPUTFLG (INPUT)
DBEB           939          ;
DBEB           940          ;
DBEB           941          icl "DC.L"
```

```
LLOAD DC.L,A$4000
```

```

DBEB      1          ttl "ROM Source Code, DC.L"
DBEB      2      ;
DBEB      3      ;
DBEB      4      ; DC.L
DBEB      5      ;
DBEB      6      ;
DBEB      7      ; Set address of INPUT data string using X/Y and value for
DBEB      8      ; INPUTFLG in A: 0x00 for INPUT, 0x40 for GET, and 0x98
DBEB      9      ; for READ.
DBEB     10      ;
DBEB 85 15     11  INPTLIST sta INPUTFLG
DBED     12      ;
DBED 86 7F     13          stx SRCPTR
DBEF 84 80     14          sty SRCPTR+1
DBF1     15      ;
DBF1 20 E3 DF  16  INPTITEM jsr PTRGET
DBF4     17      ;
DBF4 85 85     18          sta FORPNT
DBF6 84 86     19          sty FORPNT+1
DBF8     20      ;
DBF8 A5 B8     21          lda TXTPTR
DBFA A4 B9     22          ldy TXTPTR+1
DBFC     23      ;
DBFC 85 87     24          sta TXPTRSAV
DBFE 84 88     25          sty TXPTRSAV+1
DC00     26      ;
DC00 A6 7F     27          ldx SRCPTR
DC02 A4 80     28          ldy SRCPTR+1
DC04     29      ;
DC04 86 B8     30          stx TXTPTR
DC06 84 B9     31          sty TXTPTR+1
DC08     32      ;
DC08 20 B7 00  33          jsr CHRGOT
DC0B D0 1E     34          bne INSTART
DC0D     35      ;
DC0D 24 15     36          bit INPUTFLG
DC0F 50 0E     37          bvc >1                      ; not GET
DC11     38      ;
DC11 20 0C FD  39          jsr RDKEY                      ; RDKEY2 would be better
DC14     40      ;
DC14 29 7F     41          and #MSBCLR
DC16 8D 00 02  42          sta INPUT
DC19     43      ;
DC19 A2 FF     44          ldx #INPUT-1
DC1B A0 01     45          ldy /INPUT-1
DC1D D0 08     46          bne >2                      ; always taken
DC1F     47      ;
DC1F 30 7F     48      ^1      bmi FINDATA
DC21     49      ;
DC21 20 56 DB  50          jsr OUTPROMT
DC24 20 DC DB  51          jsr HEXTIN
DC27     52      ;
DC27 86 B8     53      ^2      stx TXTPTR
DC29 84 B9     54          sty TXTPTR+1
DC2B     55      ;
DC2B     56      ;
DC2B 20 B1 00  57  INSTART  jsr CHRGET
DC2E     58      ;
DC2E 24 11     59          bit VALTYP
DC30 10 31     60          bpl >5                      ; not a STRING

```

```

DC32          61 ;
DC32 24 15    62      bit INPUTFLG
DC34 50 09    63      bvc >1                ; not GET
DC36          64 ;
DC36 E8       65      inx
DC37 86 B8    66      stx TXTPTR
DC39          67 ;
DC39 A9 00    68      lda #ZERO
DC3B 85 0D    69      sta CHARAC
DC3D F0 0C    70      beq >2                ; always taken
DC3F          71 ;
DC3F 85 0D    72      ^1 sta CHARAC
DC41 C9 22    73      cmp #'"'
DC43 F0 07    74      beq >3
DC45          75 ;
DC45 A9 3A    76      lda #': '
DC47 85 0D    77      sta CHARAC
DC49          78 ;
DC49 A9 2C    79      lda #', '
DC4B          80 ;
DC4B 18       81      ^2 clc
DC4C          82 ;
DC4C 85 0E    83      ^3 sta ENDCHR
DC4E          84 ;
DC4E A5 B8    85      lda TXTPTR
DC50 A4 B9    86      ldy TXTPTR+1
DC52          87 ;
DC52 69 00    88      adc #ZERO
DC54 90 01    89      bcc >4
DC56          90 ;
DC56 C8       91      iny
DC57          92 ;
DC57 20 E9 E3 93      ^4 jsr STRLIT2
DC5A 20 3D E7 94      jsr STRCOPY
DC5D 20 7B DA 95      jsr PUTSTR
DC60          96 ;
DC60 4C 72 DC 97      jmp INSTART3
DC63          98 ;
DC63 48       99      ^5 pha
DC64         100 ;
DC64 AD 00 02 101      lda INPUT
DC67 F0 30    102      beq INPTFLG
DC69         103 ;
DC69 68       104 INSTART2 pla
DC6A 20 4A EC 105      jsr GETINT
DC6D         106 ;
DC6D A5 12    107      lda VALTYP+1
DC6F 20 63 DA 108      jsr LET2
DC72         109 ;
DC72 20 B7 00 110 INSTART3 jsr CHRGOT
DC75 F0 07    111      beq >6
DC77         112 ;
DC77 C9 2C    113      cmp #', '
DC79 F0 03    114      beq >6
DC7B         115 ;
DC7B 4C 6F DB 116      jmp INPUTERR
DC7E         117 ;
DC7E A5 B8    118      ^6 lda TXTPTR
DC80 A4 B9    119      ldy TXTPTR+1
DC82         120 ;
DC82 85 7F    121      sta SRCPTR

```

```

DC84 84 80      122      sty SRCPTR+1
DC86           123      ;
DC86 A5 87      124      lda TXPTRSAV
DC88 A4 88      125      ldy TXPTRSAV+1
DC8A           126      ;
DC8A 85 B8      127      sta TXTPTR
DC8C 84 B9      128      sty TXTPTR+1
DC8E           129      ;
DC8E 20 B7 00   130      jsr CHRGOT
DC91 F0 34      131      beq INPTDONE
DC93           132      ;
DC93 20 BE DE   133      jsr CHKCOM
DC96           134      ;
DC96 4C F1 DB   135      jmp INPTITEM
DC99           136      ;
DC99           137      ;
DC99 A5 15      138      INPTFLG lda INPUTFLG
DC9B D0 CC      139      bne INSTART2      ; READ input
DC9D           140      ;
DC9D 4C 86 DB   141      jmp INPERR
DCA0           142      ;
DCA0           143      ;
DCA0 20 A3 D9   144      FINDATA jsr DATSCAN
DCA3           145      ;
DCA3 C8         146      iny
DCA4           147      ;
DCA4 AA         148      tax
DCA5 D0 10      149      bne >1
DCA7           150      ;
DCA7 C8         151      iny
DCA8           152      ;
DCA8 B1 B8      153      lda (TXTPTR),Y
DCAA F0 18      154      beq >2
DCAC           155      ;
DCAC C8         156      iny
DCAD           157      ;
DCAD B1 B8      158      lda (TXTPTR),Y
DCAF 85 7B      159      sta DATLIN
DCB1           160      ;
DCB1 C8         161      iny
DCB2           162      ;
DCB2 B1 B8      163      lda (TXTPTR),Y
DCB4 85 7C      164      sta DATLIN+1
DCB6           165      ;
DCB6 C8         166      iny
DCB7           167      ;
DCB7 B1 B8      168      ^1 lda (TXTPTR),Y
DCB9 AA         169      tax
DCBA           170      ;
DCBA 20 98 D9   171      jsr DATA2
DCBD           172      ;
DCBD E0 83      173      cpx #$80+BSDATA/2
DCBF D0 DF      174      bne FINDATA
DCC1           175      ;
DCC1 4C 2B DC   176      jmp INSTART
DCC4           177      ;
DCC4 4C A1 E1   178      ^2 jmp OD.ERR
DCC7           179      ;
DCC7           180      ;
DCC7 A5 7F      181      INPTDONE lda SRCPTR
DCC9 A4 80      182      ldy SRCPTR+1

```

```

DCCB          183 ;
DCCB A6 15    184     ldx INPUTFLG
DCCD 30 0D    185     bmi >1                ; READ input
DCCF          186 ;
DCCF A0 00    187     ldy #ZERO
DCD1          188 ;
DCD1 B1 7F    189     lda (SRCPTR),Y
DCD3 F0 77    190     beq RTN.DD.4
DCD5          191 ;
DCD5 A9 DF    192     lda #MSG21
DCD7 A0 DC    193     ldy /MSG21
DCD9          194 ;
DCD9 4C 3B DB 195     jmp STROUT
DCDC          196 ;
DCDC 4C 53 D8 197     ^1     jmp SETDAPTR
DCDF          198 ;
DCDF          199 ;
DCDF          200 MSG21:
DCDF          201 ;     asc `?`
DCDF 3E       202     asc `>`
DCE0 45 78 74 203     asc `Extra Ignored`
DCE3 72 61 20
DCE6 49 67 6E
DCE9 6F 72 65
DCEC 64
DCED 0D 00    204     byt RETURN&MSBCLR,ZERO
DCEF          205 ;
DCEF          206 MSG22:
DCEF          207 ;     asc `?`
DCEF 3E       208     asc `>`
DCF0 52 65 65 209     asc `Reenter`
DCF3 6E 74 65
DCF6 72
DCF7 0D 00    210     byt RETURN&MSBCLR,ZERO
DCF9          211 ;
DCF9          212 ;
DCF9          213 ; Modified this routine to accommodate the FACSIZ value.
DCF9          214 ; Also, to utilize the FPCOMP entry point. Accelerated
DCF9          215 ; and shortened this routine.
DCF9          216 ;
DCF9 F0 04    217 BNEXT     beq >1                ; no variable
DCFB          218 ;
DCFB 20 E3 DF 219 NEXT2     jsr PTRGET            ; get the variable
DCFE          220 ;
DCFE 2C 00 00 221     bit *-*
DD01          222     dfs !-2
DCFF          223 ;
DCFF A0 00    224     ^1     ldy #ZERO                ; make FORPNT+1 = 0
DD01          225 ;
DD01 85 85    226     sta FORPNT
DD03 84 86    227     sty FORPNT+1
DD05          228 ;
DD05 20 62 D3 229     jsr GTFORPNT
DD08 D0 6C    230     bne NF.ERR
DD0A          231 ;
DD0A 9A       232     txs                ; C-flag set after GTFORPNT
DD0B          233 ;
DD0B 8A       234     txa
DD0C 69 03    235     adc #4-1                ; bypass inner loop values
DD0E AA       236     tax                ; address of STEP value
DD0F          237 ;

```

```

DD0F 69 07      238      adc #FACSIZE+1
DD11 48         239      pha                      ; address of END value
DD12           240      ;
DD12 8A         241      txa                      ; recall STEP value address
DD13 A0 01      242      ldy /STACK
DD15 20 F9 EA   243      jsr LOADFAC
DD18           244      ;
DD18 BA         245      tsx
DD19           246      ;
DD19 BD 0B 01   247      lda STACK+2+2+FACSIZE+1,X ; include the PHA above
DD1C 85 A2      248      sta FACSIGN
DD1E           249      ;
DD1E A5 85      250      lda FORPNT
DD20 A4 86      251      ldy FORPNT+1
DD22 20 BE E7   252      jsr FADD                      ; add to STEP value
DD25           253      ;
DD25 20 27 EB   254      jsr COPYF2FR
DD28           255      ;
DD28 68         256      pla                      ; recall END value address
DD29 A0 01      257      ldy /STACK
DD2B 20 B5 EB   258      jsr FPCOMP
DD2E           259      ;
DD2E BA         260      tsx
DD2F           261      ;
DD2F 38         262      sec
DD30           263      ;
DD30 FD 0A 01   264      sbc STACK+2+2+FACSIZE,X ; the FACSIGN
DD33 F0 1D      265      beq >3
DD35           266      ;
DD35 BD 11 01   267      lda STACK+FRAMSIZE-3,X
DD38 85 75      268      sta CURLIN
DD3A           269      ;
DD3A BD 12 01   270      lda STACK+FRAMSIZE-2,X
DD3D 85 76      271      sta CURLIN+1
DD3F           272      ;
DD3F BD 14 01   273      lda STACK+FRAMSIZE,X
DD42 85 B8      274      sta TXTPTR
DD44           275      ;
DD44 BD 13 01   276      lda STACK+FRAMSIZE-1,X
DD47 85 B9      277      sta TXTPTR+1
DD49           278      ;
DD49 4C D2 D7   279      ^2      jmp NEWSTT
DD4C           280      ;
DD4C           281      ;
DD4C 60         282      RTN.DD.4 rts
DD4D           283      ;
DD4D           284      ;
DD4D           285      ; Initialize X-reg with T3GUARD for FPCOMP.
DD4D           286      ;
DD4D A6 8F      287      FPCOMPT3 ldx T3GUARD
DD4F           288      ;
DD4F 4C B7 EB   289      jmp FPCOMP2
DD52           290      ;
DD52           291      ;
DD52 8A         292      ^3      txa
DD53 69 13      293      adc #FRAMSIZE-1          ; C-flag is set from above
DD55 AA         294      tax
DD56           295      ;
DD56 9A         296      txs
DD57           297      ;
DD57 20 B7 00   298      jsr CHRGOT

```

```

DD5A          299 ;
DD5A C9 2C    300      cmp #' , '
DD5C D0 EB    301      bne <2
DD5E          302 ;
DD5E 20 B1 00 303      jsr CHRGET
DD61 20 FB DC 304      jsr NEXT2
DD64          305 ;
DD64          306 ;
DD64 20 7B DD 307      FRMNUM    jsr FRMEVAL
DD67          308 ;
DD67 18       309      CHKNUM    clc
DD68          310 ;
DD68 B0 00    311      bcs *+2
DD6A          312      dfs !-1
DD69          313 ;
DD69 38       314      CHKSTR    sec
DD6A          315 ;
DD6A          316 ;
DD6A          317 ; String or $: VALTYP = 0xFF, VALTYP+1 = 0x00, C = 1
DD6A          318 ; Float or #: VALTYP = 0x00, VALTYP+1 = 0x00, C = 0
DD6A          319 ; Integer or %: VALTYP = 0x00, VALTYP+1 = 0x80, C = 0
DD6A          320 ;
DD6A 24 11    321      CHKVAL    bit VALTYP
DD6C 30 03    322      bmi >2                ; a STRING
DD6E          323 ;
DD6E B0 03    324      bcs TM.ERR
DD70          325 ;
DD70 60       326      ^1      rts
DD71          327 ;
DD71 B0 FD    328      ^2      bcs <1
DD73          329 ;
DD73          330 ;
DD73 A2 A7    331      TM.ERR    ldx #MSG13-MESGS ; Type Mismatch error
DD75          332 ;
DD75 2C 00 00 333      bit *-*
DD78          334      dfs !-2
DD76          335 ;
DD76 A2 00    336      NF.ERR    ldx #MSG01-MESGS ; NEXT without FOR error
DD78          337 ;
DD78 4C 12 D4 338      jmp PRterr
DD7B          339 ;
DD7B          340 ;
DD7B A6 B8    341      FRMEVAL    ldx TXTPTR
DD7D D0 02    342      bne >1
DD7F          343 ;
DD7F C6 B9    344      dec TXTPTR+1
DD81          345 ;
DD81 C6 B8    346      ^1      dec TXTPTR
DD83          347 ;
DD83 A2 00    348      ldx #ZERO
DD85          349 ;
DD85 D0 00    350      bne *+2
DD87          351      dfs !-1
DD86          352 ;
DD86 48       353      FRMEVAL2 pha
DD87          354 ;
DD87 8A       355      txa
DD88 48       356      pha
DD89          357 ;
DD89 A9 01    358      lda #1
DD8B 20 D6 D3 359      jsr CKSTKSIZ

```



```

DD8E          360 ;
DD8E 20 60 DE 361      jsr FRMELMNT
DD91          362 ;
DD91 A9 00    363      lda #ZERO
DD93 85 89    364      sta CPRTYPE
DD95          365 ;
DD95 20 B7 00 366 FRMEVAL3 jsr CHRGOT
DD98          367 ;
DD98 38       368 FRMEVAL4 sec
DD99          369 ;
DD99 E9 CF    370      sbc #TK.GRTR
DD9B 90 17    371      bcc >2
DD9D          372 ;
DD9D C9 03    373      cmp #3
DD9F B0 13    374      bcs >2
DDA1          375 ;
DDA1 C9 01    376      cmp #1
DDA3          377 ;
DDA3 2A       378      rol
DDA4 49 01    379      eor #1
DDA6          380 ;
DDA6 45 89    381      eor CPRTYPE
DDA8 C5 89    382      cmp CPRTYPE
DDAA 90 61    383      bcc SY.ERR4
DDAC          384 ;
DDAC 85 89    385      sta CPRTYPE
DDAE          386 ;
DDAE 20 B1 00 387      jsr CHRGET
DDB1          388 ;
DDB1 4C 98 DD 389      jmp FRMEVAL4
DDB4          390 ;
DDB4 A6 89    391 ^2      ldx CPRTYPE
DDB6 D0 2C    392      bne >6
DDB8          393 ;
DDB8 B0 78    394      bcs NOTMATH
DDBA          395 ;
DDBA 69 07    396      adc #TK.GRTR-TK.PLUS
DDBC 90 74    397      bcc NOTMATH
DDBE          398 ;
DDBE 65 11    399      adc VALTYP
DDC0 D0 03    400      bne >3
DDC2          401 ;
DDC2 4C 97 E5 402      jmp CAT2STR
DDC5          403 ;
DDC5 69 FF    404 ^3      adc #NEGONE
DDC7 85 5E    405      sta INDEX
DDC9          406 ;
DDC9 0A       407      asl
DDCA          408 ;
DDCA 65 5E    409      adc INDEX
DDCC A8       410      tay
DDCD          411 ;
DDCD 68       412 ^4      pla
DDCE D9 B6 D0 413      cmp TAGADDR,Y
DDD1 B0 64    414      bcs NOTMATH3
DDD3          415 ;
DDD3 20 67 DD 416      jsr CHKNUM
DDD6          417 ;
DDD6 48       418 ^5      pha
DDD7          419 ;
DDD7 20 FD DD 420 SAVOP jsr FRMRECUR

```

```

DDDA          421 ;
DDDA 68       422      pla
DDDB          423 ;
DDDB A4 87    424      ldy LASTOP
DDDD 10 17    425      bpl >8
DDDF          426 ;
DDDF AA       427      tax
DDE0 F0 53    428      beq NOTMATH2
DDE2          429 ;
DDE2 D0 5C    430      bne NOTMATH4          ; always taken
DDE4          431 ;
DDE4 46 11    432 ^6      lsr VALTYP
DDE6          433 ;
DDE6 8A       434      txa
DDE7 2A       435      rol
DDE8          436 ;
DDE8 A6 B8    437      ldx TXTPTR
DDEA D0 02    438      bne >7
DDEC          439 ;
DDEC C6 B9    440      dec TXTPTR+1
DDEE          441 ;
DDEE C6 B8    442 ^7      dec TXTPTR
DDF0          443 ;
DDF0 A0 1B    444      ldy #TAG.REL-TAGADDR
DDF2          445 ;
DDF2 85 89    446      sta CPRTYPE
DDF4          447 ;
DDF4 D0 D7    448      bne <4
DDF6          449 ;
DDF6 D9 B6 D0 450 ^8      cmp TAGADDR,Y
DDF9 90 DB    451      bcc <5
DDFB          452 ;
DDFB B0 43    453      bcs NOTMATH4          ; always taken
DDFD          454 ;
DDFD          455 ;
DDFD B9 B8 D0 456 FRMRECUR lda TAGADDR+2,Y
DE00 48       457      pha
DE01          458 ;
DE01 B9 B7 D0 459      lda TAGADDR+1,Y
DE04 48       460      pha
DE05          461 ;
DE05 20 10 DE 462      jsr FRMSTAK
DE08          463 ;
DE08 A5 89    464      lda CPRTYPE
DE0A          465 ;
DE0A 4C 86 DD 466      jmp FRMEVAL2          ; frame evaluation recursion
DE0D          467 ;
DE0D          468 ;
DE0D 4C C9 DE 469 SY.ERR4 jmp SY.ERR
DE10          470 ;
DE10          471 ;
DE10 A5 A2    472 FRMSTAK lda FACSIGN
DE12          473 ;
DE12 BE B6 D0 474      ldx TAGADDR,Y
DE15          475 ;
DE15          476 ;
DE15          477 ; Pull return address and increment. Original code assumed
DE15          478 ; not an end of page address.
DE15          479 ;
DE15 A8       480 FRMSTAK2 tay          ; FACSIGN
DE16          481 ;

```

```

DE16 18          482          clc
DE17          483          ;
DE17 68          484          pla
DE18 69 01        485          adc #1
DE1A 85 5E        486          sta INDEX
DE1C          487          ;
DE1C 68          488          pla
DE1D 69 00        489          adc #ZERO
DE1F 85 5F        490          sta INDEX+1
DE21          491          ;
DE21 98          492          tya
DE22 48          493          pha
DE23          494          ;
DE23          495          ;
DE23          496          ; Pushes Applesoft integer line numbers or FN floating-
DE23          497          ; point values onto the STACK. No need to call RNDUP
DE23          498          ; since FACGUARD is also pushed and pulled from the STACK.
DE23          499          ;
DE23 A5 AC        500 FRMSTAK3 lda FACGUARD
DE25 48          501          pha
DE26          502          ;
DE26 A5 A1        503          lda FACMANT+3
DE28 48          504          pha
DE29          505          ;
DE29 A5 A0        506          lda FACMANT+2
DE2B 48          507          pha
DE2C          508          ;
DE2C A5 9F        509          lda FACMANT+1
DE2E 48          510          pha
DE2F          511          ;
DE2F 4C B1 F1     512          jmp FRMSTAK4
DE32          513          ;
DE32          514          ;
DE32 A0 FF        515 NOTMATH ldy #NEGONE
DE34          516          ;
DE34 68          517          pla
DE35          518          ;
DE35 F0 26        519 NOTMATH2 beq >2
DE37          520          ;
DE37 C9 64        521 NOTMATH3 cmp #OTV.REL
DE39 F0 03        522          beq >1
DE3B          523          ;
DE3B 20 67 DD     524          jsr CHKNUM
DE3E          525          ;
DE3E 84 87        526 ^1      sty LASTOP
DE40          527          ;
DE40          528          ;
DE40          529          ; Pulls CPRMASK and integer line numbers or FN floating-
DE40          530          ; point values from the STACK and puts them into ARG.
DE40          531          ;
DE40 68          532 NOTMATH4 pla
DE41 4A          533          lsr
DE42 85 16        534          sta CPRMASK
DE44          535          ;
DE44 68          536          pla
DE45 85 A5        537          sta ARGEXP
DE47          538          ;
DE47 68          539          pla
DE48 85 A6        540          sta ARGMANT
DE4A          541          ;
DE4A 68          542          pla

```

```

DE4B 85 A7      543      sta ARGMANT+1
DE4D           544      ;
DE4D 68         545      pla
DE4E 85 A8      546      sta ARGMANT+2
DE50           547      ;
DE50 68         548      pla
DE51 85 A9      549      sta ARGMANT+3
DE53           550      ;
DE53 68         551      pla
DE54 85 92      552      sta ARGGUARD
DE56           553      ;
DE56 68         554      pla
DE57 85 AA      555      sta ARGSIGN
DE59           556      ;
DE59 45 A2      557      eor FACSIGN
DE5B 85 AB      558      sta XORSIGN
DE5D           559      ;
DE5D A5 9D      560      ^2    lda FACEXP
DE5F           561      ;
DE5F 60         562      rts
DE60           563      ;
DE60           564      ;
DE60 A9 00      565      FRMELMNT lda #ZERO
DE62 85 11      566      sta VALTYP
DE64           567      ;
DE64 20 B1 00   568      ^1    jsr CHRGET
DE67 B0 03      569      bcs >3
DE69           570      ;
DE69 4C 4A EC   571      ^2    jmp GETINT
DE6C           572      ;
DE6C 20 DA E0   573      ^3    jsr CHKASCII
DE6F B0 64      574      bcs GETIVAL
DE71           575      ;
DE71 C9 2E      576      cmp #'.'
DE73 F0 F4      577      beq <2
DE75           578      ;
DE75 C9 C9      579      cmp #TK.MINUS
DE77 F0 55      580      beq MINUFUNC
DE79           581      ;
DE79 C9 C8      582      cmp #TK.PLUS
DE7B F0 E7      583      beq <1
DE7D           584      ;
DE7D C9 22      585      cmp #'"'
DE7F D0 0F      586      bne NOTFUNC
DE81           587      ;
DE81           588      ;
DE81 A5 B8      589      STRTXT  lda TXTPTR
DE83 A4 B9      590      ldy TXTPTR+1
DE85           591      ;
DE85 69 00      592      adc #ZERO
DE87 90 01      593      bcc >1
DE89           594      ;
DE89 C8         595      iny
DE8A           596      ;
DE8A 20 E2 E3   597      ^1    jsr STRLIT
DE8D           598      ;
DE8D 4C 3D E7   599      jmp STRCOPY
DE90           600      ;
DE90           601      ;
DE90 C9 C6      602      NOTFUNC cmp #TK.NOT
DE92 D0 13      603      bne FNFUNC

```

```

DE94          604 ;
DE94 A0 18    605      ldy #TAG.EQU-TAGADDR
DE96 D0 38    606      bne EQULFUNC          ; always taken
DE98          607 ;
DE98          608 ;
DE98          609 ; EQUAL token.
DE98          610 ;
DE98 A5 9D    611 OEQUAL   lda FACEXP
DE9A D0 03    612          bne >1
DE9C          613 ;
DE9C A0 01    614          ldy #1
DE9E          615 ;
DE9E 2C 00 00 616          bit *-*
DEA1          617          dfs !-2
DE9F          618 ;
DE9F A0 00    619 ^1      ldy #ZERO
DEA1          620 ;
DEA1 4C 01 E3 621          jmp SNGFLT
DEA4          622 ;
DEA4          623 ;
DEA4          624          dfs 3,ZERO          ; 3 bytes
DEA7          625 ;
DEA7          626 ;
DEA7          627 ; Evaluate FN token.
DEA7          628 ;
DEA7 C9 C2    629 FNFUNC   cmp #TK.FN
DEA9 D0 03    630          bne SGNFUNC
DEAB          631 ;
DEAB 4C 54 E3 632          jmp CALLFNC
DEAE          633 ;
DEAE          634 ;
DEAE          635 ; Evaluate SGN token.  Fall into PARENCHK.  Modified this
DEAE          636 ; routine.
DEAE          637 ;
DEAE C9 D2    638 SGNFUNC   cmp #TK.SGN
DEB0 B0 57    639          bcs UNARY          ; accelerate code
DEB2          640 ;
DEB2          641 ;
DEB2          642 ; Evaluate "(expression)".
DEB2          643 ;
DEB2 20 BB DE 644 PARENCHK  jsr CHKOPNP
DEB5 20 7B DD 645          jsr FRMEVAL
DEB8          646 ;
DEB8 A9 29    647 CHKCLSP  lda #'`
DEBA          648 ;
DEBA 2C 00 00 649          bit *-*
DEBD          650          dfs !-2
DEBB          651 ;
DEBB A9 28    652 CHKOPNP  lda #'(`
DEBD          653 ;
DEBD 2C 00 00 654          bit *-*
DEC0          655          dfs !-2
DEBE          656 ;
DEBE A9 2C    657 CHKCOM   lda #`,`
DEC0          658 ;
DEC0          659 ;
DEC0 A0 00    660 SYNTAXCHK ldy #ZERO
DEC2          661 ;
DEC2 D1 B8    662          cmp (TXTPTR),Y
DEC4 D0 03    663          bne SY.ERR
DEC6          664 ;

```

```

DEC6 4C B1 00      665          jmp CHRGET
DEC9              666      ;
DEC9              667      ;
DEC9 A2 10        668 SY.ERR   ldx #MSG02-MESGS      ; Syntax error
DECB              669      ;
DECB 4C 12 D4     670          jmp PRTER
DECE              671      ;
DECE              672      ;
DECE              673      ; Minus function.
DECE              674      ;
DECE A0 15        675 MINUFUNC ldy #TAG.NEG-TAGADDR
DED0              676      ;
DED0              677      ;
DED0              678      ; Equal function.
DED0              679      ;
DED0 68           680 EQUFUNC pla
DED1 68           681          pla
DED2              682      ;
DED2 4C D7 DD     683          jmp SAVOP
DED5              684      ;
DED5              685      ;
DED5 20 E3 DF     686 GETIVAL  jsr PTRGET
DED8              687      ;
DED8 85 A0        688          sta VARPTR
DEDA 84 A1        689          sty VARPTR+1
DEDC              690      ;
DEDC A6 11        691          ldx VALTYP
DEDE F0 05        692          beq >1                ; not a STRING
DEE0              693      ;
DEE0 A2 00        694          ldx #ZERO
DEE2 86 AC        695          stx STRING1+1
DEE4              696      ;
DEE4 60           697          rts
DEE5              698      ;
DEE5              699      ;
DEE5              700      ; Get value from VARPTR into (A/Y) to put into FACMANT.
DEE5              701      ;
DEE5 A6 12        702 ^1      ldx VALTYP+1
DEE7 10 0D        703          bpl >2
DEE9              704      ;
DEE9 A0 00        705          ldy #ZERO
DEEB              706      ;
DEEB B1 A0        707          lda (VARPTR),Y
DEED AA          708          tax
DEEE              709      ;
DEEE C8           710          iny
DEEF              711      ;
DEEF B1 A0        712          lda (VARPTR),Y
DEF1 A8           713          tay
DEF2              714      ;
DEF2 8A           715          txa
DEF3              716      ;
DEF3 4C F2 E2     717          jmp GIVAYFP
DEF6              718      ;
DEF6 4C F9 EA     719 ^2      jmp LOADFAC
DEF9              720      ;
DEF9              721      ;
DEF9              722      ; This is the SCRN( function.
DEF9              723      ;
DEF9 20 EC F1     724 FSCREEN  jsr PLOTFS
DEFC              725      ;

```

```

DEFC 8A          726          txa
DEFD A4 F0       727          ldy FIRST
DEFF             728          ;
DEFF 20 71 F8    729          jsr SCRN
DF02             730          ;
DF02 A8          731          tay
DF03 20 01 E3    732          jsr SNGFLT
DF06             733          ;
DF06 4C B8 DE    734          jmp CHKCLSP
DF09             735          ;
DF09             736          ;
DF09             737          ; This code has been rewritten in order to add the PI and
DF09             738          ; LN statements. The token must be pushed onto the STACK
DF09             739          ; in order to handle very complex or FN expressions. For
DF09             740          ; Function statements, BASIC # = 0x92 + #ADDR/2.
DF09             741          ;
DF09 AA          742          UNARY      tax                      ; Function statement BASIC #
DF0A             743          ;
DF0A 0A          744          asl
DF0B 48          745          pha
DF0C             746          ;
DF0C 20 B1 00    747          jsr CHRGET
DF0F             748          ;
DF0F E0 D7       749          cpx #$92+FS1SCRN/2 ; is = SCRN(?)
DF11 F0 2A       750          beq >2
DF13             751          ;
DF13 E0 E8       752          cpx #$92+FS2LEFT/2  ; is < LEFT$?
DF15 90 23       753          bcc >1
DF17             754          ;
DF17 E0 EB       755          cpx #$92+FS3PI/2    ; is = PI?
DF19 F0 22       756          beq >2
DF1B             757          ;
DF1B B0 1D       758          bcs >1              ; for all other F3 statements
DF1D             759          ;
DF1D 20 BB DE    760          jsr CHKOPNP
DF20 20 7B DD    761          jsr FRMEVAL
DF23 20 BE DE    762          jsr CHKCOM
DF26 20 69 DD    763          jsr CHKSTR
DF29             764          ;
DF29 68          765          pla
DF2A AA          766          tax
DF2B             767          ;
DF2B A5 A1       768          lda VARPTR+1
DF2D 48          769          pha
DF2E             770          ;
DF2E A5 A0       771          lda VARPTR
DF30 48          772          pha
DF31             773          ;
DF31 8A          774          txa
DF32 48          775          pha
DF33             776          ;
DF33 20 F8 E6    777          jsr GETBYT
DF36             778          ;
DF36 68          779          pla
DF37             780          ;
DF37 4C 32 DB    781          jmp UNARY2              ; pull Y-reg and push X-reg
DF3A             782          ;
DF3A             783          ;
DF3A 20 B2 DE    784          ^1      jsr PARENCHK
DF3D             785          ;
DF3D 68          786          ^2      pla

```

```

DF3E A8          787          tay
DF3F            788          ;
DF3F            789          ;
DF3F            790          ; Index into (FN1ADDR - ((FS1SGN/2 + 0x92) * 2) & 0xFF),Y
DF3F            791          ; where Y-reg contains the function statement number.
DF3F            792          ;
DF3F B9 DC CF    793 UNARY3    lda FN1ADDR-NEGONE&2*$92+FS1SGN/2,Y
DF42 85 91       794          sta JMPADRS+1
DF44            795          ;
DF44 B9 DD CF    796          lda FN1ADDR+1-NEGONE&2*$92+FS1SGN/2,Y
DF47 85 92       797          sta JMPADRS+2
DF49            798          ;
DF49 20 90 00    799          jsr JMPADRS
DF4C            800          ;
DF4C 4C 67 DD    801          jmp CHKNUM
DF4F            802          ;
DF4F            803          ;
DF4F A5 A5       804 OOR      lda ARGEXP
DF51 05 9D       805          ora FACEXP
DF53 D0 0B       806          bne TRUE
DF55            807          ;
DF55            808          ;
DF55 A5 A5       809 OAND     lda ARGEXP
DF57 F0 04       810          beq FALSE
DF59            811          ;
DF59 A5 9D       812          lda FACEXP
DF5B D0 03       813          bne TRUE
DF5D            814          ;
DF5D            815          ;
DF5D A0 00       816 FALSE    ldy #ZERO
DF5F            817          ;
DF5F 2C 00 00    818          bit *-*
DF62            819          dfs !-2
DF60            820          ;
DF60 A0 01       821 TRUE     ldy #1
DF62            822          ;
DF62 4C 01 E3    823          jmp SNGFLT
DF65            824          ;
DF65            825          ;
DF65            826          ; Perform relational operations.
DF65            827          ;
DF65 20 6A DD    828 OLT      jsr CHKVAL
DF68 B0 13       829          bcs STRCMP
DF6A            830          ;
DF6A A5 AA       831          lda ARGSIGN
DF6C 09 7F       832          ora #MSBCLR
DF6E            833          ;
DF6E 25 A6       834          and ARGMANT
DF70 85 A6       835          sta ARGMANT
DF72            836          ;
DF72 A9 A5       837          lda #ARGEXP
DF74 A0 00       838          ldy /ARGEXP
DF76 20 B2 EB    839          jsr FPCOMP0
DF79            840          ;
DF79 AA         841          tax
DF7A            842          ;
DF7A 4C B0 DF    843          jmp NUMCMP
DF7D            844          ;
DF7D            845          ;
DF7D            846          ; String comparison function.
DF7D            847          ;

```



```

DF7D A9 00      848  STRCMP    lda #ZERO
DF7F 85 11      849          sta VALTYP
DF81           850  ;
DF81 C6 89      851          dec CPRTYPE
DF83           852  ;
DF83 20 00 E6    853          jsr FREFAC
DF86           854  ;
DF86 85 9D      855          sta DSCTMP
DF88 86 9E      856          stx DSCTMP+1
DF8A 84 9F      857          sty DSCTMP+2
DF8C           858  ;
DF8C A5 A8      859          lda ARGMANT+2
DF8E A4 A9      860          ldy ARGMANT+3
DF90 20 04 E6    861          jsr FRETMP
DF93           862  ;
DF93 86 A8      863          stx ARGMANT+2
DF95 84 A9      864          sty ARGMANT+3
DF97           865  ;
DF97 AA         866          tax
DF98           867  ;
DF98 38         868          sec
DF99           869  ;
DF99 E5 9D      870          sbc FACEXP
DF9B F0 08      871          beq >1
DF9D           872  ;
DF9D A9 01      873          lda #1
DF9F           874  ;
DF9F 90 04      875          bcc >1
DFA1           876  ;
DFA1 A6 9D      877          ldx FACEXP
DFA3           878  ;
DFA3 A9 FF      879          lda #NEGONE
DFA5           880  ;
DFA5 85 A2      881  ^1      sta FACSIGN
DFA7           882  ;
DFA7 A0 FF      883          ldy #NEGONE
DFA9           884  ;
DFA9 E8         885          inx
DFAA           886  ;
DFAA C8         887  ^2      iny
DFAB           888  ;
DFAB CA         889          dex
DFAC D0 07      890          bne >3
DFAE           891  ;
DFAE A6 A2      892          ldx FACSIGN
DFB0           893  ;
DFB0 30 0F      894  NUMCMP    bmi >4
DFB2           895  ;
DFB2 18         896          clc
DFB3 90 0C      897          bcc >4          ; always taken
DFB5           898  ;
DFB5 B1 A8      899  ^3      lda (ARGMANT+2),Y
DFB7 D1 9E      900          cmp (FACMANT),Y
DFB9 F0 EF      901          beq <2
DFBB           902  ;
DFBB A2 FF      903          ldx #NEGONE
DFBD           904  ;
DFBD B0 02      905          bcs >4
DFBF           906  ;
DFBF A2 01      907          ldx #1
DFC1           908  ;

```

```

DFC1 E8          909 ^4      inx
DFC2             910 ;
DFC2 8A          911      txa
DFC3 2A          912      rol
DFC4             913 ;
DFC4 25 16       914      and CPRMASK
DFC6 F0 02       915      beq >5
DFC8             916 ;
DFC8 A9 01       917      lda #1
DFCA             918 ;
DFCA 4C 93 EB    919 ^5      jmp FLOAT
DFCD             920 ;
DFCD             921 ;
DFCD 20 FB E6    922 FPDL    jsr CONVINT
DFD0 20 08 E9    923      jsr PDL2          ; check argument range
DFD3             924 ;
DFD3 4C 01 E3    925      jmp SNGFLT
DFD6             926 ;
DFD6             927 ;
DFD6 20 BE DE    928 ^1      jsr CHKCOM          ; next variable
DFD9             929 ;
DFD9 AA          930 BDIM    tax
DFDA             931 ;
DFDA 20 E8 DF    932      jsr PTRGET2
DFDD             933 ;
DFDD 20 B7 00    934      jsr CHRGOT
DFE0 D0 F4       935      bne <1
DFE2             936 ;
DFE2 60          937      rts
DFE3             938 ;
DFE3             939 ;
DFE3             940 ; PTRGET is controlled by DIMFLG and SUBFLG.
DFE3             941 ;   DIMFLG - nonzero is call from "DIM", else zero.
DFE3             942 ;   SUBFLG - 0x00 initial value
DFE3             943 ;           0x40 if called from GETARYPT (removed)
DFE3             944 ;           0x80 if called from DEF FN
DFE3             945 ;           0xC1:DA if called from FN
DFE3             946 ;
DFE3 A2 00       947 PTRGET   ldx #ZERO
DFE5             948 ;
DFE5 20 B7 00    949      jsr CHRGOT
DFE8             950 ;
DFE8 86 10       951 PTRGET2  stx DIMFLG
DFEA             952 ;
DFEA 85 81       953 PTRGET3  sta VARNAM
DFEC             954 ;
DFEC 20 B7 00    955      jsr CHRGOT
DFEF             956 ;
DFEF 20 DA E0    957      jsr CHKASCI
DFF2 B0 03       958      bcs >1          ; not digit
DFF4             959 ;
DFF4             960 ;
DFF4 4C C9 DE    961 SY.ERR5  jmp SY.ERR
DFF7             962 ;
DFF7             963 ;
DFF7 A2 00       964 ^1      ldx #ZERO
DFF9 86 11       965      stx VALTYP
DFFB 86 12       966      stx VALTYP+1
DFFD             967 ;
DFFD 4C 06 E0    968      jmp PTRGET4          ; jump past 0xE000 vectors
E000             969 ;

```

E000 970 ;

BSAVE D0ROM,D1,A\$1000,B,L\$1000

E000 971 usr D0ROM,D1

E000 972 ;

E000 973 ;

E000 974 icl "E0.L,D2"

LLOAD E0.L,D2,A\$4000

```

E000          1          ttl "ROM Source Code, E0.L"
E000          2          ;
E000          3          ;
E000          4          ; E0.L
E000          5          ;
E000          6          ;
E000          7          obj PAGE10
E000          8          usr
E000          9          ;
E000         10          ;
E000         11          ; This is the DOS default COLDADR.
E000         12          ;
E000 4C 25 F1 13 BASIC      jmp COLDSTRT
E003         14          ;
E003 4C 3C D4 15 BASIC2    jmp RESTART
E006         16          ;
E006         17          ;
E006 20 B1 00 18 PTRGET4   jsr CHRGET
E009 90 05    19          bcc >1
E00B         20          ;
E00B 20 DA E0 21          jsr CHKASCI
E00E 90 0B    22          bcc >3
E010         23          ;
E010 AA      24          ^1    tax
E011         25          ;
E011 20 B1 00 26          ^2    jsr CHRGET
E014 90 FB    27          bcc <2
E016         28          ;
E016 20 DA E0 29          jsr CHKASCI
E019 B0 F6    30          bcs <2
E01B         31          ;
E01B C9 24    32          ^3    cmp #'$'
E01D D0 06    33          bne >4
E01F         34          ;
E01F A9 FF    35          lda #NEGONE
E021 85 11    36          sta VALTYP
E023 D0 10    37          bne >5          ; always taken
E025         38          ;
E025 C9 25    39          ^4    cmp #'%'
E027 D0 13    40          bne >6
E029         41          ;
E029 A5 14    42          lda SUBFLG
E02B 30 C7    43          bmi SY.ERR5
E02D         44          ;
E02D A9 80    45          lda #$80          ; integer value flag
E02F 85 12    46          sta VALTYP+1
E031         47          ;
E031 05 81    48          ora VARNAM
E033 85 81    49          sta VARNAM
E035         50          ;
E035 8A      51          ^5    txa
E036 09 80    52          ora #$80
E038 AA      53          tax
E039         54          ;
E039 20 B1 00 55          jsr CHRGET
E03C         56          ;
E03C 86 82    57          ^6    stx VARNAM+1
E03E         58          ;
E03E 38      59          sec
E03F         60          ;

```

```

E03F 05 14      61      ora SUBFLG      ; 0x00 or 0x80
E041            62      ;
E041 E9 28      63      sbc #'('
E043            64      ;      bne >8
E043 D0 03      65      bne >1
E045            66      ;
E045            67      ; ^7      jmp ARRAY
E045 4C 28 E1    68      jmp ARRAY
E048            69      ;
E048            70      ;
E048            71      ; The GETARYPT routine removed so SUBFLG never set to 0x40.
E048            72      ; There is no need to test SUBFLG here.
E048            73      ;
E048            74      ; ^8      bit SUBFLG
E048            75      ;      bmi >1
E048            76      ;
E048            77      ;      bvs <7
E048            78      ;
E048 A9 00      79      ^1      lda #ZERO
E04A 85 14      80      sta SUBFLG
E04C            81      ;
E04C A5 69      82      lda VARTAB
E04E A6 6A      83      ldx VARTAB+1
E050            84      ;
E050 A0 00      85      ldy #ZERO
E052            86      ;
E052 86 9C      87      ^2      stx LOWTR+1
E054            88      ;
E054 85 9B      89      ^3      sta LOWTR
E056            90      ;
E056 E4 6C      91      cpx ARYTAB+1
E058 D0 04      92      bne >4
E05A            93      ;
E05A C5 6B      94      cmp ARYTAB
E05C F0 18      95      beq PTRGET5
E05E            96      ;
E05E A5 81      97      ^4      lda VARNAM
E060 D1 9B      98      cmp (LOWTR),Y
E062 D0 08      99      bne >5
E064           100      ;
E064 C8         101      iny
E065           102      ;
E065 A5 82      103      lda VARNAM+1
E067 D1 9B      104      cmp (LOWTR),Y
E069 F0 60      105      beq PTRGET7
E06B           106      ;
E06B 88         107      dey
E06C           108      ;
E06C 18         109      ^5      clc
E06D           110      ;
E06D A5 9B      111      lda LOWTR
E06F 69 07      112      adc #SVARLEN      ; next descriptor
E071 90 E1      113      bcc <3
E073           114      ;
E073 E8         115      inx
E074 D0 DC      116      bne <2      ; always taken
E076           117      ;
E076           118      ;
E076           119      ; Variable not found.
E076           120      ;
E076 68         121      PTRGET5 pla

```

```

E077 48          122      pha
E078          123      ;
E078 C9 D7      124      cmp #GETIVAL+2
E07A D0 0D      125      bne PTRGET6
E07C          126      ;
E07C BA        127      tsx
E07D          128      ;
E07D BD 02 01   129      lda STACK+2,X
E080 C9 DE      130      cmp /GETIVAL+2
E082 D0 05      131      bne PTRGET6
E084          132      ;
E084 A9 05      133      lda #IVALZERO
E086 A0 E1      134      ldy /IVALZERO
E088          135      ;
E088 60         136      rts
E089          137      ;
E089          138      ;
E089          139      ; Insert a simple variable.
E089          140      ;
E089 A5 6B      141      PTRGET6 lda ARYTAB
E08B A4 6C      142      ldy ARYTAB+1
E08D          143      ;
E08D 85 9B      144      sta LOWTR
E08F 84 9C      145      sty LOWTR+1
E091          146      ;
E091 A5 6D      147      lda STREND
E093 A4 6E      148      ldy STREND+1
E095          149      ;
E095 85 96      150      sta HIGHTR
E097 84 97      151      sty HIGHTR+1
E099          152      ;
E099 18         153      clc
E09A          154      ;
E09A 69 07      155      adc #SVARLEN
E09C 90 01      156      bcc >1
E09E          157      ;
E09E C8         158      iny
E09F          159      ;
E09F          160      ;
E09F          161      ; ARYPNT and HIGHDS are the same pointers.
E09F          162      ;
E09F 85 94      163      ^1      sta ARYPNT
E0A1 84 95      164      sty ARYPNT+1
E0A3          165      ;
E0A3 20 93 D3   166      jsr BLTU
E0A6          167      ;
E0A6 A5 94      168      lda ARYPNT
E0A8 A4 95      169      ldy ARYPNT+1
E0AA          170      ;
E0AA C8         171      iny
E0AB          172      ;
E0AB 85 6B      173      sta ARYTAB
E0AD 84 6C      174      sty ARYTAB+1
E0AF          175      ;
E0AF          176      ;
E0AF          177      ; Name the 7-byte descriptor and clear the last 5 bytes.
E0AF          178      ;
E0AF A0 00      179      ldy #ZERO
E0B1          180      ;
E0B1 A5 81      181      lda VARNAM
E0B3 91 9B      182      sta (LOWTR),Y

```

```

E0B5      183 ;
E0B5 C8    184      iny
E0B6      185 ;
E0B6 A5 82  186      lda VARNAM+1
E0B8 91 9B  187      sta (LOWTR),Y
E0BA      188 ;
E0BA A9 00  189      lda #ZERO
E0BC      190 ;
E0BC C8    191      iny
E0BD      192 ;
E0BD 91 9B  193      sta (LOWTR),Y
E0BF      194 ;
E0BF C8    195      iny
E0C0      196 ;
E0C0 91 9B  197      sta (LOWTR),Y
E0C2      198 ;
E0C2 C8    199      iny
E0C3      200 ;
E0C3 91 9B  201      sta (LOWTR),Y
E0C5      202 ;
E0C5 C8    203      iny
E0C6      204 ;
E0C6 91 9B  205      sta (LOWTR),Y
E0C8      206 ;
E0C8 C8    207      iny
E0C9      208 ;
E0C9 91 9B  209      sta (LOWTR),Y
E0CB      210 ;
E0CB      211 ;
E0CB      212 ; Insert the address of the variable's value.
E0CB      213 ;
E0CB A4 9C  214 PTRGET7  ldy LOWTR+1
E0CD      215 ;
E0CD 18     216      clc
E0CE      217 ;
E0CE A5 9B  218      lda LOWTR
E0D0 69 02  219      adc #IVARLEN
E0D2 90 01  220      bcc >3
E0D4      221 ;
E0D4 C8    222      iny
E0D5      223 ;
E0D5 85 83  224 ^3      sta VARPNT
E0D7 84 84  225      sty VARPNT+1
E0D9      226 ;
E0D9 60     227      rts
E0DA      228 ;
E0DA      229 ;
E0DA      230 ; If A-reg contains an ASCII letter A-Z, set C-flag,
E0DA      231 ; otherwise clear C-flag. This code has been rewritten and
E0DA      232 ; moved from 0xE07D to this location.
E0DA      233 ;
E0DA C9 5B  234 CHKASCII  cmp #'Z'+1
E0DC B0 03  235      bcs >1
E0DE      236 ;
E0DE C9 41  237      cmp #'A'
E0E0      238 ;
E0E0 60     239      rts
E0E1      240 ;
E0E1 18     241 ^1      clc
E0E2      242 ;
E0E2 60     243      rts

```

```

E0E3      244 ;
E0E3      245 ;
E0E3      246 ; Point to first array value.  ARYPNT=(LOWER) + #DIMS*2 + 5
E0E3      247 ;
E0E3 A4 9C  248 PNTARVAL ldy LOWTR+1
E0E5      249 ;
E0E5 A5 0F  250         lda NUMDIM
E0E7 0A     251         asl
E0E8 69 05  252         adc #AHDRLEN
E0EA      253 ;
E0EA 65 9B  254         adc LOWTR
E0EC 90 01  255         bcc >7
E0EE      256 ;
E0EE C8     257         iny
E0EF      258 ;
E0EF 85 94  259 ^7      sta ARYPNT
E0F1 84 95  260         sty ARYPNT+1
E0F3      261 ;
E0F3 60     262         rts
E0F4      263 ;
E0F4      264 ;
E0F4      265         dfs 11,ZERO          ; 11 bytes
E0FF      266 ;
E0FF      267 ;
E0FF      268 ; Beginning of the STRSETUP patch in order to provide three
E0FF      269 ; bytes to MID$ processing at 0xE6A2 to check for a value
E0FF      270 ; of zero for the second numerical parameter.
E0FF      271 ;
E0FF 20 B8 DE 272 STRSETUP jsr CHKCLSP
E102      273 ;
E102 4C BC E6 274         jmp STRSET2
E105      275 ;
E105      276 ;
E105 00 00   277 IVALZERO hex 0000          ; moved from 0xE09A
E107 90 80 00 278 FP8000  hex 9080000000    ; FP 32768
E10A 00 00   279 ;
E10C      280 ;
E10C 20 B1 00 281 MAKINT  jsr CHRGET
E10F 20 64 DD 282         jsr FRMNUM
E112      283 ;
E112      284 ;
E112 A5 A2   285 AYPOSINT lda FACSIGN
E114 30 0D   286         bmi IQ.ERR2
E116      287 ;
E116      288 ;
E116 A5 9D   289 AYINT    lda FACEXP
E118 C9 90   290         cmp #$90          ; maximum integer exponent
E11A 90 09   291         bcc >1
E11C      292 ;
E11C A9 07   293         lda #FP8000
E11E A0 E1   294         ldy /FP8000
E120      295 ;
E120 20 B5 EB 296         jsr FPCOMP
E123      297 ;
E123 D0 76   298 IQ.ERR2  bne IQ.ERR
E125      299 ;
E125 4C F2 EB 300 ^1      jmp FP2INT
E128      301 ;
E128      302 ;
E128      303 ; Rewrote this code in order to accelerate this routine.

```



```

E128          304 ;
E128 A5 14    305 ARRAY    lda SUBFLG
E12A D0 3F    306          bne >2
E12C          307 ;
E12C A5 10    308          lda DIMFLG
E12E 05 12    309          ora VALTYP+1
E130 48       310          pha
E131          311 ;
E131 A5 11    312          lda VALTYP
E133 48       313          pha
E134          314 ;
E134 A9 00    315          lda #ZERO
E136 85 0F    316          sta NUMDIM
E138          317 ;
E138 A5 82    318 ^1      lda VARNAM+1
E13A 48       319          pha
E13B          320 ;
E13B A5 81    321          lda VARNAM
E13D 48       322          pha
E13E          323 ;
E13E 20 0C E1 324          jsr MAKINT
E141          325 ;
E141 68       326          pla
E142 85 81    327          sta VARNAM
E144          328 ;
E144 68       329          pla
E145 85 82    330          sta VARNAM+1
E147          331 ;
E147 68       332          pla
E148 A8       333          tay
E149          334 ;
E149 68       335          pla
E14A AA       336          tax
E14B          337 ;
E14B A5 A0    338          lda FACMANT+2
E14D 48       339          pha
E14E          340 ;
E14E A5 A1    341          lda FACMANT+3
E150 48       342          pha
E151          343 ;
E151 8A       344          txa
E152 48       345          pha
E153          346 ;
E153 98       347          tya
E154 48       348          pha
E155          349 ;
E155 E6 0F    350          inc NUMDIM
E157          351 ;
E157 20 B7 00 352          jsr CHRGOT
E15A          353 ;
E15A C9 2C    354          cmp #', '
E15C F0 DA    355          beq <1
E15E          356 ;
E15E 20 B8 DE 357          jsr CHKCLSP
E161          358 ;
E161 68       359          pla
E162 85 11    360          sta VALTYP
E164          361 ;
E164 68       362          pla
E165 85 12    363          sta VALTYP+1
E167          364 ;

```

```

E167 29 7F      365      and #MSBCLR
E169 85 10      366      sta DIMFLG
E16B            367      ;
E16B            368      ;
E16B            369      ; Search Array Table for this Array Name.
E16B            370      ;
E16B A6 6B      371      ^2      ldx ARYTAB
E16D A5 6C      372      lda ARYTAB+1
E16F            373      ;
E16F 86 9B      374      ^3      stx LOWTR
E171 85 9C      375      sta LOWTR+1
E173            376      ;
E173 C5 6E      377      cmp STREND+1
E175 D0 04      378      bne >4
E177            379      ;
E177 E4 6D      380      cpx STREND
E179 F0 43      381      beq >1
E17B            382      ;
E17B A0 00      383      ^4      ldy #ZERO
E17D            384      ;
E17D B1 9B      385      lda (LOWTR),Y
E17F            386      ;
E17F C8          387      iny
E180            388      ;
E180 C5 81      389      cmp VARNAM
E182 D0 06      390      bne >5
E184            391      ;
E184 B1 9B      392      lda (LOWTR),Y
E186 C5 82      393      cmp VARNAM+1
E188 F0 1C      394      beq >6
E18A            395      ;
E18A C8          396      ^5      iny
E18B            397      ;
E18B 18          398      clc
E18C            399      ;
E18C B1 9B      400      lda (LOWTR),Y
E18E 65 9B      401      adc LOWTR
E190 AA          402      tax
E191            403      ;
E191 C8          404      iny
E192            405      ;
E192 B1 9B      406      lda (LOWTR),Y
E194 65 9C      407      adc LOWTR+1
E196 90 D7      408      bcc <3          ; always taken
E198            409      ;
E198            410      ;
E198 A2 6D      411      BS.ERR      ldx #MSG09-MESGS      ; Bad Subscript error
E19A            412      ;
E19A 2C 00 00   413      bit *-*
E19D            414      dfs !-2
E19B            415      ;
E19B A2 35      416      IQ.ERR      ldx #MSG05-MESGS      ; Illegal Quantity error
E19D            417      ;
E19D 2C 00 00   418      bit *-*
E1A0            419      dfs !-2
E19E            420      ;
E19E A2 7A      421      RA.ERR      ldx #MSG10-MESGS      ; ReDIM'd Array error
E1A0            422      ;
E1A0 2C 00 00   423      bit *-*
E1A3            424      dfs !-2
E1A1            425      ;

```

```

E1A1 A2 2A      426 OD.ERR      ldx #MSG04-MESGS      ; Out of Data error
E1A3            427 ;
E1A3 4C 12 D4    428            jmp PRterr
E1A6            429 ;
E1A6            430 ;
E1A6 A5 10      431 ^6          lda DIMFLG
E1A8 D0 F4      432            bne RA.ERR
E1AA            433 ;
E1AA A5 14      434            lda SUBFLG
E1AC F0 02      435            beq >7
E1AE            436 ;
E1AE 38         437            sec
E1AF            438 ;
E1AF 60         439            rts
E1B0            440 ;
E1B0 20 E3 E0    441 ^7          jsr PNTARVAL
E1B3            442 ;
E1B3 A5 0F      443            lda NUMDIM
E1B5            444 ;
E1B5 A0 04      445            ldy #4
E1B7            446 ;
E1B7 D1 9B      447            cmp (LOWTR),Y
E1B9 D0 DD      448            bne BS.ERR
E1BB            449 ;
E1BB 4C 4B E2    450            jmp FINDELE
E1BE            451 ;
E1BE            452 ;
E1BE            453 ; Create a new Array.
E1BE            454 ;
E1BE A5 14      455 ^1          lda SUBFLG
E1C0 D0 DF      456            bne OD.ERR
E1C2            457 ;
E1C2 20 E3 E0    458            jsr PNTARVAL
E1C5 20 E3 D3    459            jsr CKSTRSIZ
E1C8            460 ;
E1C8 A0 00      461            ldy #ZERO
E1CA 84 AE      462            sty STRING2+1
E1CC            463 ;
E1CC A2 05      464            ldx #5
E1CE            465 ;
E1CE A5 81      466            lda VARNAM
E1D0 91 9B      467            sta (LOWTR),Y
E1D2 10 01      468            bpl >2
E1D4            469 ;
E1D4 CA         470            dex
E1D5            471 ;
E1D5 C8         472 ^2          iny
E1D6            473 ;
E1D6 A5 82      474            lda VARNAM+1
E1D8 91 9B      475            sta (LOWTR),Y
E1DA 10 02      476            bpl >3
E1DC            477 ;
E1DC CA         478            dex
E1DD CA         479            dex
E1DE            480 ;
E1DE 86 AD      481 ^3          stx STRING2
E1E0            482 ;
E1E0 C8         483            iny
E1E1 C8         484            iny
E1E2 C8         485            iny
E1E3            486 ;

```

```

E1E3 A5 0F      487      lda NUMDIM
E1E5 91 9B      488      sta (LOWTR),Y
E1E7            489      ;
E1E7 A2 0B      490      ^4      ldx #DFLTDIM
E1E9 A9 00      491      lda #ZERO
E1EB            492      ;
E1EB 24 10      493      bit DIMFLG
E1ED 50 08      494      bvc >5
E1EF            495      ;
E1EF 18          496      clc
E1F0            497      ;
E1F0 68          498      pla
E1F1 69 01      499      adc #1
E1F3 AA          500      tax
E1F4            501      ;
E1F4 68          502      pla
E1F5 69 00      503      adc #ZERO
E1F7            504      ;
E1F7 C8          505      ^5      iny
E1F8            506      ;
E1F8 91 9B      507      sta (LOWTR),Y
E1FA            508      ;
E1FA C8          509      iny
E1FB            510      ;
E1FB 8A          511      txa
E1FC 91 9B      512      sta (LOWTR),Y
E1FE            513      ;
E1FE 20 AD E2    514      jsr MULSUBS
E201            515      ;
E201 86 AD      516      stx STRING2
E203 85 AE      517      sta STRING2+1
E205            518      ;
E205 A4 5E      519      ldy INDEX
E207            520      ;
E207 C6 0F      521      dec NUMDIM
E209 D0 DC      522      bne <4
E20B            523      ;
E20B 65 95      524      adc ARYPNT+1
E20D B0 5D      525      bcs OM.ERR2
E20F            526      ;
E20F 85 95      527      sta ARYPNT+1
E211            528      ;
E211 A8          529      tay
E212 8A          530      txa
E213            531      ;
E213 65 94      532      adc ARYPNT
E215 90 03      533      bcc >6
E217            534      ;
E217 C8          535      iny
E218 F0 52      536      beq OM.ERR2
E21A            537      ;
E21A 20 E3 D3    538      ^6      jsr CKSTRSIZ
E21D            539      ;
E21D 85 6D      540      sta STREND
E21F 84 6E      541      sty STREND+1
E221            542      ;
E221 A9 00      543      lda #ZERO
E223            544      ;
E223 E6 AE      545      inc STRING2+1
E225            546      ;
E225 A4 AD      547      ldy STRING2

```

```

E227 F0 05      548      beq >8
E229            549      ;
E229 88         550      ^7      dey
E22A            551      ;
E22A 91 94      552      sta (ARYPNT),Y
E22C            553      ;
E22C D0 FB      554      bne <7
E22E            555      ;
E22E C6 95      556      ^8      dec ARYPNT+1
E230            557      ;
E230 C6 AE      558      dec STRING2+1
E232 D0 F5      559      bne <7
E234            560      ;
E234 E6 95      561      inc ARYPNT+1
E236            562      ;
E236 A0 02      563      ldy #2
E238            564      ;
E238 38         565      sec
E239            566      ;
E239 A5 6D      567      lda STREND
E23B E5 9B      568      sbc LOWTR
E23D 91 9B      569      sta (LOWTR),Y
E23F            570      ;
E23F C8         571      iny
E240            572      ;
E240 A5 6E      573      lda STREND+1
E242 E5 9C      574      sbc LOWTR+1
E244 91 9B      575      sta (LOWTR),Y
E246            576      ;
E246 A5 10      577      lda DIMFLG
E248 D0 62      578      bne RTN.E2.A
E24A            579      ;
E24A C8         580      iny
E24B            581      ;
E24B            582      ;
E24B            583      ; Find specified Array Element.
E24B            584      ;
E24B B1 9B      585      FINDELE  lda (LOWTR),Y
E24D 85 0F      586      sta NUMDIM
E24F            587      ;
E24F A9 00      588      lda #ZERO
E251 85 AD      589      sta STRING2
E253            590      ;
E253 85 AE      591      ^1      sta STRING2+1
E255            592      ;
E255 C8         593      iny
E256            594      ;
E256 68         595      pla
E257 85 A0      596      sta FACMANT+2
E259 AA         597      tax
E25A            598      ;
E25A 68         599      pla
E25B 85 A1      600      sta FACMANT+3
E25D            601      ;
E25D D1 9B      602      cmp (LOWTR),Y
E25F 90 0E      603      bcc >2
E261            604      ;
E261 D0 06      605      bne BS.ERR2
E263            606      ;
E263 C8         607      iny
E264            608      ;

```

```

E264 8A          609          txa
E265 D1 9B      610          cmp (LOWTR),Y
E267 90 07      611          bcc >3
E269           612          ;
E269           613          ;
E269 4C 98 E1   614 BS.ERR2  jmp BS.ERR
E26C           615          ;
E26C 4C 10 D4   616 OM.ERR2  jmp OM.ERR
E26F           617          ;
E26F           618          ;
E26F C8         619 ^2        iny
E270           620          ;
E270 A5 AE      621 ^3        lda STRING2+1
E272 05 AD      622          ora STRING2
E274           623          ;
E274 18         624          clc
E275           625          ;
E275 F0 0A      626          beq >4
E277           627          ;
E277 20 AD E2   628          jsr MULSUBS
E27A           629          ;
E27A 8A         630          txa
E27B 65 A0      631          adc FACMANT+2
E27D AA         632          tax
E27E           633          ;
E27E 98         634          tya
E27F           635          ;
E27F A4 5E      636          ldy INDEX
E281           637          ;
E281 65 A1      638 ^4        adc FACMANT+3
E283           639          ;
E283 86 AD      640          stx STRING2
E285           641          ;
E285 C6 0F      642          dec NUMDIM
E287 D0 CA      643          bne <1
E289           644          ;
E289 85 AE      645          sta STRING2+1
E28B           646          ;
E28B A2 05      647          ldx #5
E28D           648          ;
E28D A5 81      649          lda VARNAM
E28F 10 01      650          bpl >5
E291           651          ;
E291 CA         652          dex
E292           653          ;
E292 A5 82      654 ^5        lda VARNAM+1
E294 10 02      655          bpl >6
E296           656          ;
E296 CA         657          dex
E297 CA         658          dex
E298           659          ;
E298 86 64      660 ^6        stx MULMANT+2
E29A           661          ;
E29A A9 00      662          lda #ZERO
E29C 20 B6 E2   663          jsr MULSUBS2
E29F           664          ;
E29F 8A         665          txa
E2A0 65 94      666          adc ARYPNT
E2A2 85 83      667          sta VARPNT
E2A4           668          ;
E2A4 98         669          tya

```

```

E2A5 65 95      670      adc ARYPNT+1
E2A7 85 84      671      sta VARPNT+1
E2A9            672      ;
E2A9 A8         673      tay
E2AA            674      ;
E2AA A5 83      675      lda VARPNT
E2AC            676      ;
E2AC 60         677      RTN.E2.A rts
E2AD            678      ;
E2AD            679      ;
E2AD 84 5E      680      MULSUBS  sty INDEX
E2AF            681      ;
E2AF B1 9B      682      lda (LOWTR),Y
E2B1 85 64      683      sta MULMANT+2
E2B3            684      ;
E2B3 88         685      dey
E2B4            686      ;
E2B4 B1 9B      687      lda (LOWTR),Y
E2B6            688      ;
E2B6 85 65      689      MULSUBS2 sta MULMANT+3
E2B8            690      ;
E2B8 A9 10      691      lda #16                ; number of iterations
E2BA 85 99      692      sta TEMP2+1
E2BC            693      ;
E2BC A2 00      694      ldx #ZERO
E2BE A0 00      695      ldy #ZERO
E2C0            696      ;
E2C0 8A         697      ^1 txa
E2C1 0A         698      asl
E2C2 AA         699      tax
E2C3            700      ;
E2C3 98         701      tya
E2C4 2A         702      rol
E2C5 A8         703      tay
E2C6            704      ;
E2C6 B0 A4      705      bcs OM.ERR2
E2C8            706      ;
E2C8 06 AD      707      asl STRING2
E2CA 26 AE      708      rol STRING2+1
E2CC 90 0B      709      bcc >2
E2CE            710      ;
E2CE 18         711      clc
E2CF            712      ;
E2CF 8A         713      txa
E2D0 65 64      714      adc MULMANT+2
E2D2 AA         715      tax
E2D3            716      ;
E2D3 98         717      tya
E2D4 65 65      718      adc MULMANT+3
E2D6 A8         719      tay
E2D7            720      ;
E2D7 B0 93      721      bcs OM.ERR2
E2D9            722      ;
E2D9 C6 99      723      ^2 dec TEMP2+1
E2DB D0 E3      724      bne <1
E2DD            725      ;
E2DD 60         726      RTN.E2.D rts
E2DE            727      ;
E2DE            728      ;
E2DE A5 11      729      FFRE  lda VALTYP
E2E0 F0 03      730      beq >1

```

```

E2E2      731 ;
E2E2 20 00 E6 732      jsr FREFAC
E2E5      733 ;
E2E5 20 84 E4 734 ^1      jsr GARBAG
E2E8      735 ;
E2E8 38      736      sec
E2E9      737 ;
E2E9 A5 6F 738      lda FRETOP
E2EB E5 6D 739      sbc STREND
E2ED A8      740      tay
E2EE      741 ;
E2EE A5 70 742      lda FRETOP+1
E2F0 E5 6E 743      sbc STREND+1
E2F2      744 ;
E2F2 85 9E 745 GIVAYFP sta FACMANT
E2F4 84 9F 746      sty FACMANT+1
E2F6      747 ;
E2F6 A2 00 748      ldx #ZERO
E2F8 86 11 749      stx VALTYP
E2FA      750 ;
E2FA A2 90 751      ldx #$90          ; integer exponent
E2FC      752 ;
E2FC 4C 9B EB 753      jmp FLOAT2
E2FF      754 ;
E2FF      755 ;
E2FF A4 24 756 FPOS      ldy CH          ; the LSB
E301      757 ;
E301      758 ;
E301 A9 00 759 SNGFLT      lda #ZERO          ; the MSB
E303 F0 ED 760      beq GIVAYFP          ; always taken
E305      761 ;
E305      762 ;
E305      763      dfs 1,ZERO          ; 1 byte
E306      764 ;
E306      765 ;
E306      766 ; CURLIN+1 = 0xFF if in Direct Mode, else in Running Mode.
E306      767 ;
E306 A6 76 768 CHKIFDIR ldx CURLIN+1
E308      769 ;
E308 E8      770      inx
E309 D0 D2 771      bne RTN.E2.D
E30B      772 ;
E30B      773 ;
E30B A2 99 774      ldx #MSG12-MESGS      ; Illegal Direct error
E30D      775 ;
E30D 2C 00 00 776      bit *-*
E310      777      dfs !-2
E30E      778 ;
E30E A2 E5 779 UF.ERR      ldx #MSG17-MESGS      ; Undefined Function error
E310      780 ;
E310 4C 12 D4 781      jmp PRterr
E313      782 ;
E313      783 ;
E313 20 41 E3 784 BDEF      jsr GETFNC
E316 20 06 E3 785      jsr CHKIFDIR
E319 20 BB DE 786      jsr CHKOPNP
E31C      787 ;
E31C A9 80 788      lda #$80          ; DEF FN flag value
E31E 85 14 789      sta SUBFLG
E320      790 ;
E320 20 E3 DF 791      jsr PTRGET

```



```

E323 20 67 DD      792      jsr CHKNUM
E326 20 B8 DE      793      jsr CHKCLSP
E329                794      ;
E329 A9 D0         795      lda #TK.EQUAL
E32B 20 C0 DE      796      jsr SYNTAXCHK
E32E                797      ;
E32E 48            798      pha
E32F                799      ;
E32F A5 84         800      lda VARPNT+1
E331 48            801      pha
E332                802      ;
E332 A5 83         803      lda VARPNT
E334 48            804      pha
E335                805      ;
E335 A5 B9         806      lda TXTPTR+1
E337 48            807      pha
E338                808      ;
E338 A5 B8         809      lda TXTPTR
E33A 48            810      pha
E33B                811      ;
E33B 20 95 D9      812      jsr BDATA
E33E                813      ;
E33E 4C AF E3      814      jmp FNCDATA
E341                815      ;
E341                816      ;
E341 A9 C2         817      GETFNC  lda #TK.FN
E343 20 C0 DE      818      jsr SYNTAXCHK
E346                819      ;
E346 09 80         820      ora #$80
E348 85 14         821      sta SUBFLG      ; 0xC0 < SUBFLG < 0xDB
E34A                822      ;
E34A 20 EA DF      823      jsr PTRGET3
E34D                824      ;
E34D 85 8A         825      sta FUNCNAM
E34F 84 8B         826      sty FUNCNAM+1
E351                827      ;
E351 4C 67 DD      828      jmp CHKNUM
E354                829      ;
E354                830      ;
E354 20 41 E3      831      CALLFNC  jsr GETFNC
E357                832      ;
E357 A5 8B         833      lda FUNCNAM+1
E359 48            834      pha
E35A                835      ;
E35A A5 8A         836      lda FUNCNAM
E35C 48            837      pha
E35D                838      ;
E35D 20 B2 DE      839      jsr PARENCHK
E360 20 67 DD      840      jsr CHKNUM
E363                841      ;
E363 68            842      pla
E364 85 8A         843      sta FUNCNAM
E366                844      ;
E366 68            845      pla
E367 85 8B         846      sta FUNCNAM+1
E369                847      ;
E369 A0 02         848      ldy #2
E36B                849      ;
E36B B1 8A         850      lda (FUNCNAM),Y
E36D 85 83         851      sta VARPNT
E36F AA            852      tax

```

```

E370          853 ;
E370 C8       854      iny
E371          855 ;
E371 B1 8A    856      lda (FUNCNAM),Y
E373 F0 99    857      beq UF.ERR
E375          858 ;
E375 85 84    859      sta VARPNT+1
E377          860 ;
E377 C8       861      iny
E378          862 ;
E378 B1 83    863 ^1    lda (VARPNT),Y
E37A 48       864      pha
E37B          865 ;
E37B 88       866      dey
E37C 10 FA    867      bpl <1
E37E          868 ;
E37E A4 84    869      ldy VARPNT+1
E380          870 ;
E380 20 2B EB 871      jsr COPYFAC
E383          872 ;
E383 A5 B9    873      lda TXTPTR+1
E385 48       874      pha
E386          875 ;
E386 A5 B8    876      lda TXTPTR
E388 48       877      pha
E389          878 ;
E389 B1 8A    879      lda (FUNCNAM),Y
E38B 85 B8    880      sta TXTPTR
E38D          881 ;
E38D C8       882      iny
E38E          883 ;
E38E B1 8A    884      lda (FUNCNAM),Y
E390 85 B9    885      sta TXTPTR+1
E392          886 ;
E392 A5 84    887      lda VARPNT+1
E394 48       888      pha
E395          889 ;
E395 A5 83    890      lda VARPNT
E397 48       891      pha
E398          892 ;
E398 20 64 DD 893      jsr FRMNUM
E39B          894 ;
E39B 68       895      pla
E39C 85 8A    896      sta FUNCNAM
E39E          897 ;
E39E 68       898      pla
E39F 85 8B    899      sta FUNCNAM+1
E3A1          900 ;
E3A1 20 B7 00 901      jsr CHRGOT
E3A4 F0 03    902      beq >2
E3A6          903 ;
E3A6 4C C9 DE 904      jmp SY.ERR
E3A9          905 ;
E3A9          906 ;
E3A9          907 ; Retrieve TXTPTR.
E3A9          908 ;
E3A9 68       909 ^2    pla
E3AA 85 B8    910      sta TXTPTR
E3AC          911 ;
E3AC 68       912      pla
E3AD 85 B9    913      sta TXTPTR+1

```

```

E3AF          914 ;
E3AF          915 ;
E3AF          916 ; Get five bytes from STACK and save at FUNCNAM variable
E3AF          917 ; name for DEF. These bytes are TXTPTR, TXTPTR+1, VARPNT,
E3AF          918 ; VARPNT+1, and the Applesoft statment function number.
E3AF          919 ; Otherwise, get five floating-point bytes for CALLFNC.
E3AF          920 ;
E3AF A0 00    921 FNCDATA ldy #ZERO
E3B1          922 ;
E3B1 68       923         pla
E3B2 91 8A    924         sta (FUNCNAM),Y
E3B4          925 ;
E3B4 C8       926         iny
E3B5          927 ;
E3B5 68       928         pla
E3B6 91 8A    929         sta (FUNCNAM),Y
E3B8          930 ;
E3B8 C8       931         iny
E3B9          932 ;
E3B9 68       933         pla
E3BA 91 8A    934         sta (FUNCNAM),Y
E3BC          935 ;
E3BC C8       936         iny
E3BD          937 ;
E3BD 68       938         pla
E3BE 91 8A    939         sta (FUNCNAM),Y
E3C0          940 ;
E3C0 C8       941         iny
E3C1          942 ;
E3C1 68       943         pla
E3C2 91 8A    944         sta (FUNCNAM),Y
E3C4          945 ;
E3C4 60       946         rts
E3C5          947 ;
E3C5          948 ;
E3C5          949 ; All numeric strings now start at the beginning of the
E3C5          950 ; STACK, so this routine is modified accordingly. FPOUT
E3C5          951 ; returns with (A/Y) containing the address of the STACK.
E3C5          952 ; STRLIT does not differentiate between calls to STR$,
E3C5          953 ; LINEPRT, and PRINT.
E3C5          954 ;
E3C5 20 67 DD 955 FSTR      jsr CHKNUM
E3C8          956 ;
E3C8 68       957         pla
E3C9 68       958         pla
E3CA          959 ;
E3CA 20 34 ED 960         jsr FPOUT
E3CD          961 ;
E3CD 18       962         clc
E3CE 90 13    963         bcc STRLIT1          ; always taken
E3D0          964 ;
E3D0          965 ;
E3D0 A6 A0    966 STRINI    ldx FACMANT+2
E3D2 A4 A1    967         ldy FACMANT+3
E3D4          968 ;
E3D4 86 8C    969         stx DSCPTR
E3D6 84 8D    970         sty DSCPTR+1
E3D8          971 ;
E3D8          972 ;
E3D8 20 54 E4 973 STRSPA    jsr GETSSPC
E3DB          974 ;

```

E3DB	85 9D	975	sta	FACEXP
E3DD	86 9E	976	stx	FACMANT
E3DF	84 9F	977	sty	FACMANT+1
E3E1		978	;	
E3E1	60	979	rts	
E3E2		980	;	
E3E2		981	;	
E3E2		982	icl	"E4.L"

LLOAD E4.L,A\$4000

```

E3E2      1          ttl "ROM Source Code, E4.L"
E3E2      2      ;
E3E2      3      ;
E3E2      4      ; E4.L
E3E2      5      ;
E3E2      6      ;
E3E2 38    7      STRLIT    sec
E3E3      8      ;
E3E3 A2 22  9      STRLIT1   ldx #'"'
E3E5 86 0D 10     stx CHARAC
E3E7 86 0E 11     stx ENDCHR
E3E9      12     ;
E3E9 85 AB 13     STRLIT2   sta STRING1
E3EB 84 AC 14     sty STRING1+1
E3ED      15     ;
E3ED 85 9E 16     sta DSCTMP+1
E3EF 84 9F 17     sty DSCTMP+2
E3F1      18     ;
E3F1 88    19     dey
E3F2 88    20     dey
E3F3      21     ;
E3F3 08    22     php                      ; Page 2 and STR flag status
E3F4      23     ;
E3F4 A0 FF 24     ldy #NEGONE
E3F6      25     ;
E3F6 C8    26     ^1      iny
E3F7      27     ;
E3F7 B1 AB 28     lda (STRING1),Y
E3F9 F0 0C 29     beq >3
E3FB      30     ;
E3FB C5 0D 31     cmp CHARAC
E3FD F0 04 32     beq >2
E3FF      33     ;
E3FF C5 0E 34     cmp ENDCHR
E401 D0 F3 35     bne <1
E403      36     ;
E403 C9 22 37     ^2      cmp #'"'
E405 F0 01 38     beq >4
E407      39     ;
E407 18    40     ^3      clc
E408      41     ;
E408 84 9D 42     ^4      sty DSCTMP
E40A      43     ;
E40A A6 AC 44     ldx STRING1+1
E40C      45     ;
E40C 98    46     tya
E40D      47     ;
E40D 65 AB 48     adc STRING1
E40F 85 AD 49     sta STRING2
E411 90 01 50     bcc >5
E413      51     ;
E413 E8    52     inx
E414      53     ;
E414      54     ;
E414      55     ; Since all numeric strings begin at the beginning of the
E414      56     ; STACK, the following logic tests for Page 2 strings first
E414      57     ; and then for STR strings. All other strings goto PUTNEW.
E414      58     ;
E414 86 AE 59     ^5      stx STRING2+1
E416      60     ;

```

```

E416 28          61          plp
E417 F0 02       62          beq >6          ; branch if Page 2
E419             63          ;
E419 B0 0B       64          bcs PUTNEW        ; branch if not STR
E41B             65          ;
E41B 98          66          ^6          tya          ; length of string
E41C             67          ;
E41C 20 D0 E3    68          jsr STRINI
E41F             69          ;
E41F A6 AB       70          ldx STRING1
E421 A4 AC       71          ldy STRING1+1
E423 20 E2 E5    72          jsr MOVSTR
E426             73          ;
E426             74          ;
E426 A6 52       75          PUTNEW    ldx TEMPPT
E428 E0 5E       76          cpx #TEMPST+9      ; max of 3 temp strings
E42A F0 1D       77          beq FC.ERR
E42C             78          ;
E42C A5 9D       79          lda DSCTMP
E42E 95 00       80          sta LOC0,X
E430             81          ;
E430 A5 9E       82          lda DSCTMP+1
E432 95 01       83          sta LOC1,X
E434             84          ;
E434 A5 9F       85          lda DSCTMP+2
E436 95 02       86          sta LOC2,X
E438             87          ;
E438 A0 00       88          ldy #ZERO
E43A             89          ;
E43A 86 A0       90          stx FACMANT+2
E43C 84 A1       91          sty FACMANT+3
E43E             92          ;
E43E 88          93          dey
E43F             94          ;
E43F 84 11       95          sty VALTYP
E441 86 53       96          stx LASTPT
E443             97          ;
E443 E8          98          inx
E444 E8          99          inx
E445 E8          100         inx
E446             101         ;
E446 86 52       102         stx TEMPPT
E448             103         ;
E448 60          104         rts
E449             105         ;
E449             106         ;
E449 A2 C3       107         FC.ERR    ldx #MSG15-MESGS    ; Formula too Complex error
E44B             108         ;
E44B 2C 00 00    109         bit *-*
E44E             110         dfs !-2
E44C             111         ;
E44C A2 4D       112         OM.ERR3   ldx #MSG07-MESGS    ; Out of Memory error
E44E             113         ;
E44E 4C 12 D4    114         jmp PRterr
E451             115         ;
E451             116         ;
E451             117         dfs 3,ZERO      ; 3 bytes
E454             118         ;
E454             119         ;
E454             120         ; Get space for A-reg bytes.  Return address in X/Y-reg.
E454             121         ;

```

```

E454 46 13      122 GETSSPC  lsr GARFLG
E456           123 ;
E456 48        124 ^1      pha
E457           125 ;
E457 A4 70     126          ldy FRETOP+1
E459           127 ;
E459 38        128          sec
E45A           129 ;
E45A 49 FF     130          eor #NEGONE
E45C 65 6F     131          adc FRETOP
E45E B0 01     132          bcs >2
E460           133 ;
E460 88        134          dey
E461           135 ;
E461 C4 6E     136 ^2      cpy STREND+1
E463 90 11     137          bcc >4
E465           138 ;
E465 D0 04     139          bne >3
E467           140 ;
E467 C5 6D     141          cmp STREND
E469 90 0B     142          bcc >4
E46B           143 ;
E46B 85 6F     144 ^3      sta FRETOP
E46D 84 70     145          sty FRETOP+1
E46F           146 ;
E46F 85 71     147          sta FRESPEC
E471 84 72     148          sty FRESPEC+1
E473           149 ;
E473 AA        150          tax
E474           151 ;
E474 68        152          pla
E475           153 ;
E475 60        154          rts
E476           155 ;
E476 A5 13     156 ^4      lda GARFLG
E478 30 D2     157          bmi OM.ERR3
E47A           158 ;
E47A 20 84 E4  159          jsr GARBAG
E47D           160 ;
E47D A9 80     161          lda #$80
E47F 85 13     162          sta GARFLG
E481           163 ;
E481 68        164          pla
E482 D0 D2     165          bne <1          ; always taken
E484           166 ;
E484           167 ;
E484           168 ; GARBAG has been rewritten according to the concepts that
E484           169 ; were developed by Cornelis Bongers. This routine uses
E484           170 ; the CX space at 0xC600 for the PROCVAR and PROCSPCL
E484           171 ; routines for character string garbage collection.
E484           172 ;
E484 A9 00     173 GARBAG  lda #ZERO
E486 85 CD     174          sta SPCLFLAG
E488 85 95     175          sta PROCESS
E48A 8D 07 C0  176          sta CXROMON
E48D           177 ;
E48D           178 ;
E48D A5 69     179 CHKVARS  lda VARTAB
E48F A6 6A     180          ldx VARTAB+1
E491           181 ;
E491 85 9B     182          sta LOWTR

```

```

E493 86 9C      183      stx LOWTR+1
E495           184      ;
E495 C5 6B      185      ^1      cmp ARYTAB
E497 D0 04      186      bne >2
E499           187      ;
E499 E4 6C      188      cpx ARYTAB+1
E49B F0 25      189      beq CHKARRYS
E49D           190      ;
E49D A0 00      191      ^2      ldy #ZERO
E49F           192      ;
E49F B1 9B      193      lda (LOWTR),Y
E4A1 30 10      194      bmi >3
E4A3           195      ;
E4A3 C8         196      iny
E4A4           197      ;
E4A4 B1 9B      198      lda (LOWTR),Y
E4A6 10 0B      199      bpl >3
E4A8           200      ;
E4A8 A9 02      201      lda #2
E4AA 20 67 E5   202      jsr NXTVAR
E4AD           203      ;
E4AD 20 00 C6   204      jsr PROCVAR
E4B0           205      ;
E4B0 A9 05      206      lda #SVARLEN-2
E4B2           207      ;
E4B2 2C 00 00   208      bit *-*
E4B5           209      dfs !-2
E4B3           210      ;
E4B3 A9 07      211      ^3      lda #SVARLEN
E4B5           212      ;
E4B5 20 67 E5   213      jsr NXTVAR
E4B8 90 DB      214      bcc <1
E4BA           215      ;
E4BA A5 96      216      CHKVARS2 lda HIGHTR
E4BC A6 97      217      ldx HIGHTR+1
E4BE           218      ;
E4BE 85 9B      219      sta LOWTR
E4C0 86 9C      220      stx LOWTR+1
E4C2           221      ;
E4C2           222      ;
E4C2 C5 6D      223      CHKVARS2 cmp STREND
E4C4 D0 04      224      bne >1
E4C6           225      ;
E4C6 E4 6E      226      cpx STREND+1
E4C8 F0 33      227      beq >4
E4CA           228      ;
E4CA 18         229      ^1      clc
E4CB           230      ;
E4CB A0 02      231      ldy #2
E4CD           232      ;
E4CD 71 9B      233      adc (LOWTR),Y
E4CF 85 96      234      sta HIGHTR
E4D1           235      ;
E4D1 C8         236      iny
E4D2           237      ;
E4D2 8A         238      txa
E4D3 71 9B      239      adc (LOWTR),Y
E4D5 85 97      240      sta HIGHTR+1
E4D7           241      ;
E4D7 A0 00      242      ldy #ZERO
E4D9           243      ;

```



```

E4D9 B1 9B      244      lda (LOWTR),Y
E4DB 30 DD      245      bmi CHKVAR2
E4DD            246      ;
E4DD C8         247      iny
E4DE            248      ;
E4DE B1 9B      249      lda (LOWTR),Y
E4E0 10 D8      250      bpl CHKVAR2
E4E2            251      ;
E4E2 A0 04      252      ldy #4
E4E4            253      ;
E4E4 B1 9B      254      lda (LOWTR),Y
E4E6 0A         255      asl
E4E7 69 05      256      adc #AHDRLEN
E4E9            257      ;
E4E9 20 67 E5    258      ^2      jsr NXTVAR
E4EC            259      ;
E4EC C5 96      260      cmp HIGHTR
E4EE D0 04      261      bne >3
E4F0            262      ;
E4F0 E4 97      263      cpx HIGHTR+1
E4F2 F0 CE      264      beq CHKARRYS
E4F4            265      ;
E4F4 A0 01      266      ^3      ldy #1
E4F6 20 00 C6    267      jsr PROCVAR
E4F9            268      ;
E4F9 A9 03      269      lda #AVARLEN
E4FB D0 EC      270      bne <2                ; always taken
E4FD            271      ;
E4FD 24 95      272      ^4      bit PROCESS
E4FF 10 0B      273      bpl MOVVARS
E501            274      ;
E501            275      ;
E501 A5 8A      276      GARBEXIT lda FUNCNAM
E503 A6 8B      277      ldx FUNCNAM+1
E505            278      ;
E505 85 6F      279      sta FRETOP
E507 86 70      280      stx FRETOP+1
E509            281      ;
E509 4C 3C FA    282      jmp CXOFF
E50C            283      ;
E50C            284      ;
E50C A5 73      285      MOVVARS  lda MEMSIZE
E50E A6 74      286      ldx MEMSIZE+1
E510            287      ;
E510 85 8A      288      sta FUNCNAM
E512 86 8B      289      stx FUNCNAM+1
E514            290      ;
E514 85 9B      291      sta LOWTR
E516 86 9C      292      stx LOWTR+1
E518            293      ;
E518 A0 00      294      MOVVARS2 ldy #ZERO
E51A            295      ;
E51A A5 9B      296      ^1      lda LOWTR
E51C C5 6F      297      cmp FRETOP
E51E D0 04      298      bne >2
E520            299      ;
E520 E4 70      300      cpx FRETOP+1
E522 F0 3A      301      beq >3
E524            302      ;
E524 A9 FF      303      ^2      lda #NEGONE
E526 20 8C E5    304      jsr DECPTR

```

```

E529          305 ;
E529 B1 9B    306     lda (LOWTR),Y
E52B 10 ED    307     bpl <1
E52D          308 ;
E52D 29 7F    309     and #MSBCLR
E52F 91 9B    310     sta (LOWTR),Y
E531          311 ;
E531 A9 FE    312     lda #NEGTWO
E533 20 8C E5 313     jsr DECPTR
E536          314 ;
E536 B1 9B    315     lda (LOWTR),Y
E538 85 5E    316     sta INDEX
E53A          317 ;
E53A C8       318     iny
E53B          319 ;
E53B B1 9B    320     lda (LOWTR),Y
E53D 85 5F    321     sta INDEX+1
E53F          322 ;
E53F C8       323     iny
E540          324 ;
E540 B1 5E    325     lda (INDEX),Y
E542          326 ;
E542 88       327     dey
E543          328 ;
E543 91 9B    329     sta (LOWTR),Y
E545          330 ;
E545 B1 5E    331     lda (INDEX),Y
E547          332 ;
E547 88       333     dey
E548          334 ;
E548 91 9B    335     sta (LOWTR),Y
E54A          336 ;
E54A B1 5E    337     lda (INDEX),Y
E54C 85 94    338     sta LEN
E54E          339 ;
E54E 20 73 E5 340     jsr COPYVAR
E551          341 ;
E551 C8       342     iny
E552          343 ;
E552 A5 8A    344     lda FUNCNAM
E554 91 5E    345     sta (INDEX),Y
E556          346 ;
E556 C8       347     iny
E557          348 ;
E557 A5 8B    349     lda FUNCNAM+1
E559 91 5E    350     sta (INDEX),Y
E55B          351 ;
E55B 4C 18 E5 352     jmp MOVVARS2
E55E          353 ;
E55E 24 CD    354     ^3 bit SPCLFLAG
E560 10 9F    355     bpl GARBEXIT
E562          356 ;
E562 C6 95    357     dec PROCESS
E564          358 ;
E564 4C 8D E4 359     jmp CHKVARS
E567          360 ;
E567          361 ;
E567 18       362     NXTVAR clc
E568          363 ;
E568 65 9B    364     adc LOWTR
E56A 85 9B    365     sta LOWTR

```

```

E56C 90 04      366      bcc >1
E56E            367      ;
E56E E6 9C      368      inc LOWTR+1
E570            369      ;
E570 E8         370      inx
E571            371      ;
E571 18         372      clc
E572            373      ;
E572 60         374      ^1    rts
E573            375      ;
E573            376      ;
E573 38         377      COPYVAR  sec
E574            378      ;
E574 A5 8A      379      lda FUNCNAM
E576 E5 94      380      sbc LEN
E578 85 8A      381      sta FUNCNAM
E57A            382      ;
E57A A5 8B      383      lda FUNCNAM+1
E57C E9 00      384      sbc #ZERO
E57E 85 8B      385      sta FUNCNAM+1
E580            386      ;
E580 A4 94      387      ldy LEN
E582            388      ;
E582 88         389      ^1    dey
E583            390      ;
E583 B1 9B      391      lda (LOWTR),Y
E585 91 8A      392      sta (FUNCNAM),Y
E587            393      ;
E587 C0 00      394      cpy #ZERO
E589 D0 F7      395      bne <1
E58B            396      ;
E58B 60         397      rts
E58C            398      ;
E58C            399      ;
E58C 18         400      DECPTR  clc
E58D            401      ;
E58D 65 9B      402      adc LOWTR
E58F 85 9B      403      sta LOWTR
E591 B0 03      404      bcs >1
E593            405      ;
E593 C6 9C      406      dec LOWTR+1
E595            407      ;
E595 CA         408      dex
E596            409      ;
E596 60         410      ^1    rts
E597            411      ;
E597            412      ;
E597            413      ; End of GARBAG.  The PROCVAR routine continues at 0xC600
E597            414      ; and the PROCSPCL routine is at 0xC64E and ends at 0xC66F.
E597            415      ;
E597            416      ;
E597 A5 A1      417      CAT2STR  lda FACMANT+3
E599 48         418      pha
E59A            419      ;
E59A A5 A0      420      lda FACMANT+2
E59C 48         421      pha
E59D            422      ;
E59D 20 60 DE   423      jsr FRMELMNT
E5A0 20 69 DD   424      jsr CHKSTR
E5A3            425      ;
E5A3 68         426      pla

```

```

E5A4 85 AB      427      sta STRING1
E5A6           428      ;
E5A6 68        429      pla
E5A7 85 AC      430      sta STRING1+1
E5A9           431      ;
E5A9 A0 00      432      ldy #ZERO
E5AB           433      ;
E5AB 18         434      clc
E5AC           435      ;
E5AC B1 AB      436      lda (STRING1),Y
E5AE 71 A0      437      adc (FACMANT+2),Y
E5B0 B0 1D      438      bcs SL.ERR
E5B2           439      ;
E5B2 20 D0 E3   440      jsr STRINI
E5B5 20 D4 E5   441      jsr MOVINS
E5B8           442      ;
E5B8 A5 8C      443      lda DSCPTR
E5BA A4 8D      444      ldy DSCPTR+1
E5BC 20 04 E6   445      jsr FRETMP
E5BF           446      ;
E5BF 20 E6 E5   447      jsr MOVSTR2
E5C2           448      ;
E5C2 A5 AB      449      lda STRING1
E5C4 A4 AC      450      ldy STRING1+1
E5C6 20 04 E6   451      jsr FRETMP
E5C9           452      ;
E5C9 20 26 E4   453      jsr PUTNEW
E5CC           454      ;
E5CC 4C 95 DD   455      jmp FRMEVAL3
E5CF           456      ;
E5CF           457      ;
E5CF A2 B4      458      SL.ERR ldx #MSG14-MESGS ; String too Long error
E5D1           459      ;
E5D1 4C 12 D4   460      jmp PRTErr
E5D4           461      ;
E5D4           462      ;
E5D4 A0 00      463      MOVINS ldy #ZERO
E5D6           464      ;
E5D6 B1 AB      465      lda (STRING1),Y
E5D8 48         466      pha
E5D9           467      ;
E5D9 C8         468      iny
E5DA           469      ;
E5DA B1 AB      470      lda (STRING1),Y
E5DC AA         471      tax
E5DD           472      ;
E5DD C8         473      iny
E5DE           474      ;
E5DE B1 AB      475      lda (STRING1),Y
E5E0 A8         476      tay
E5E1           477      ;
E5E1 68         478      pla
E5E2           479      ;
E5E2           480      ;
E5E2 86 5E      481      MOVSTR stx INDEX
E5E4 84 5F      482      sty INDEX+1
E5E6           483      ;
E5E6 A8         484      MOVSTR2 tay
E5E7 F0 0A      485      beq >2
E5E9           486      ;
E5E9 48         487      pha

```

```

E5EA          488 ;
E5EA 88       489 ^1    dey
E5EB          490 ;
E5EB B1 5E    491      lda (INDEX),Y
E5ED 91 71    492      sta (FRESPC),Y
E5EF          493 ;
E5EF 98       494      tya
E5F0 D0 F8    495      bne <1
E5F2          496 ;
E5F2 68       497      pla
E5F3          498 ;
E5F3 18       499 ^2    clc
E5F4          500 ;
E5F4 65 71    501      adc FRESPC
E5F6 85 71    502      sta FRESPC
E5F8 90 02    503      bcc >3
E5FA          504 ;
E5FA E6 72    505      inc FRESPC+1
E5FC          506 ;
E5FC 60       507 ^3    rts
E5FD          508 ;
E5FD          509 ;
E5FD 20 69 DD 510 FRESTR jsr CHKSTR
E600          511 ;
E600          512 ;
E600 A5 A0    513 FREFAC  lda FACMANT+2
E602 A4 A1    514      ldy FACMANT+3
E604          515 ;
E604          516 ;
E604 85 5E    517 FRETMP  sta INDEX
E606 84 5F    518      sty INDEX+1
E608          519 ;
E608 20 35 E6 520      jsr FRETMS
E60B          521 ;
E60B 08       522      php
E60C          523 ;
E60C A0 00    524      ldy #ZERO
E60E          525 ;
E60E B1 5E    526      lda (INDEX),Y
E610 48       527      pha
E611          528 ;
E611 C8       529      iny
E612          530 ;
E612 B1 5E    531      lda (INDEX),Y
E614 AA       532      tax
E615          533 ;
E615 C8       534      iny
E616          535 ;
E616 B1 5E    536      lda (INDEX),Y
E618 A8       537      tay
E619          538 ;
E619 68       539      pla
E61A          540 ;
E61A 28       541      plp
E61B D0 13    542      bne >2
E61D          543 ;
E61D C4 70    544      cpy FRETOP+1
E61F D0 0F    545      bne >2
E621          546 ;
E621 E4 6F    547      cpx FRETOP
E623 D0 0B    548      bne >2

```

```

E625          549 ;
E625 48      550 pha
E626          551 ;
E626 18      552 clc
E627          553 ;
E627 65 6F   554 adc FRETOP
E629 85 6F   555 sta FRETOP
E62B 90 02   556 bcc >1
E62D          557 ;
E62D E6 70   558 inc FRETOP+1
E62F          559 ;
E62F 68      560 ^1 pla
E630          561 ;
E630 86 5E   562 ^2 stx INDEX
E632 84 5F   563 sty INDEX+1
E634          564 ;
E634 60      565 rts
E635          566 ;
E635          567 ;
E635 C4 54   568 FRETMS cpy LASTPT+1
E637 D0 0C   569 bne >1
E639          570 ;
E639 C5 53   571 cmp LASTPT
E63B D0 08   572 bne >1
E63D          573 ;
E63D 85 52   574 sta TEMPPT
E63F          575 ;
E63F E9 03   576 sbc #3
E641 85 53   577 sta LASTPT
E643          578 ;
E643 A0 00   579 ldy #ZERO
E645          580 ;
E645 60      581 ^1 rts
E646          582 ;
E646          583 ;
E646 20 FB E6 584 FCHR jsr CONVINT
E649          585 ;
E649 8A      586 txa
E64A 48      587 pha
E64B          588 ;
E64B A9 01   589 lda #1 ; string length of 1
E64D 20 D8 E3 590 jsr STRSPA
E650          591 ;
E650 A0 00   592 ldy #ZERO
E652          593 ;
E652 68      594 pla
E653 91 9E   595 sta (FACMANT),Y
E655          596 ;
E655 68      597 pla
E656 68      598 pla
E657          599 ;
E657 4C 26 E4 600 jmp PUTNEW
E65A          601 ;
E65A          602 ;
E65A 20 FF E0 603 FLEFT jsr STRSETUP
E65D          604 ;
E65D D1 8C   605 cmp (DSCPTR),Y
E65F          606 ;
E65F 98      607 tya
E660          608 ;
E660 90 04   609 LEFT2 bcc >1

```

```

E662      610 ;
E662 B1 8C      611      lda (DSCPTR),Y
E664 AA      612      tax
E665      613 ;
E665 98      614      tya
E666      615 ;
E666 48      616 ^1      pha
E667      617 ;
E667 8A      618 LEFT3      txa
E668      619 ;
E668 48      620 LEFT4      pha
E669      621 ;
E669 20 D8 E3  622      jsr STRSPA
E66C      623 ;
E66C A5 8C      624      lda DSCPTR
E66E A4 8D      625      ldy DSCPTR+1
E670 20 04 E6  626      jsr FRETMP
E673      627 ;
E673 18      628      clc
E674      629 ;
E674 68      630      pla
E675 A8      631      tay
E676      632 ;
E676 68      633      pla
E677 65 5E      634      adc INDEX
E679 85 5E      635      sta INDEX
E67B 90 02      636      bcc >2
E67D      637 ;
E67D E6 5F      638      inc INDEX+1
E67F      639 ;
E67F 98      640 ^2      tya
E680      641 ;
E680 20 E6 E5  642      jsr MOVSTR2
E683      643 ;
E683 4C 26 E4  644      jmp PUTNEW
E686      645 ;
E686      646 ;
E686 20 FF E0  647 FRIGHT      jsr STRSETUP
E689      648 ;
E689 18      649      clc
E68A      650 ;
E68A F1 8C      651      sbc (DSCPTR),Y
E68C 49 FF      652      eor #NEGONE
E68E      653 ;
E68E 4C 60 E6  654      jmp LEFT2
E691      655 ;
E691      656 ;
E691 A9 FF      657 FMID      lda #NEGONE
E693 85 A1      658      sta FACMANT+3
E695      659 ;
E695 20 B7 00  660      jsr CHRGOT
E698      661 ;
E698 C9 29      662      cmp #'`
E69A F0 09      663      beq >1
E69C      664 ;
E69C 20 BE DE  665      jsr CHKCOM
E69F 20 F8 E6  666      jsr GETBYT
E6A2      667 ;
E6A2 8A      668      txa
E6A3 F0 4D      669      beq IQ.ERR3
E6A5      670 ;

```

; new code; check for zero

```

E6A5 20 FF E0    671  ^1      jsr STRSETUP
E6A8             672  ;
E6A8 CA         673          dex
E6A9             674  ;
E6A9 8A         675          txa
E6AA 48         676          pha
E6AB             677  ;
E6AB A2 00      678          ldx #ZERO
E6AD             679  ;
E6AD 18         680          clc
E6AE             681  ;
E6AE F1 8C      682          sbc (DSCPTR),Y
E6B0 B0 B5      683          bcs LEFT3
E6B2             684  ;
E6B2 49 FF      685          eor #NEGONE
E6B4             686  ;
E6B4 C5 A1      687          cmp FACMANT+3
E6B6 90 B0      688          bcc LEFT4
E6B8             689  ;
E6B8 A5 A1      690          lda FACMANT+3
E6BA             691  ;
E6BA B0 AC      692          bcs LEFT4          ; always taken
E6BC             693  ;
E6BC             694  ;
E6BC             695  ; Common setup for LEFT$, RIGHT$, and MID$. Requires ")",
E6BC             696  ; pops return address, pops and discards UNARY address,
E6BC             697  ; pops first numerical value, and pops descriptor address.
E6BC             698  ; It pushes return address and tests numerical value for
E6BC             699  ; zero. Modified entrance to extract three bytes that
E6BC             700  ; are used to test second numerical value for zero.
E6BC             701  ;
E6BC 68         702  STRSET2  pla
E6BD A8         703          tay
E6BE             704  ;
E6BE 68         705          pla
E6BF 85 91      706          sta RTNADR
E6C1             707  ;
E6C1 68         708          pla
E6C2 68         709          pla
E6C3             710  ;
E6C3 68         711          pla
E6C4 AA         712          tax
E6C5             713  ;
E6C5 68         714          pla
E6C6 85 8C      715          sta DSCPTR
E6C8             716  ;
E6C8 68         717          pla
E6C9 85 8D      718          sta DSCPTR+1
E6CB             719  ;
E6CB A5 91      720          lda RTNADR
E6CD 48         721          pha
E6CE             722  ;
E6CE 98         723          tya
E6CF 48         724          pha
E6D0             725  ;
E6D0 A0 00      726          ldy #ZERO
E6D2             727  ;
E6D2 8A         728          txa
E6D3 F0 1D      729          beq IQ.ERR3
E6D5             730  ;
E6D5 60         731          rts

```



```

E6D6      732 ;
E6D6      733 ;
E6D6 20 DC E6 734 FLEN      jsr GETSTRLN
E6D9      735 ;
E6D9 4C 01 E3 736          jmp SNGFLT
E6DC      737 ;
E6DC      738 ;
E6DC 20 FD E5 739 GETSTRLN jsr FRESTR
E6DF      740 ;
E6DF A2 00      741          ldx #ZERO          ; make it numeric
E6E1 86 11      742          stx VALTYP
E6E3      743 ;
E6E3 A8          744          tay
E6E4      745 ;
E6E4 60          746          rts
E6E5      747 ;
E6E5      748 ;
E6E5 20 DC E6 749 FASC      jsr GETSTRLN
E6E8 F0 08      750          beq IQ.ERR3
E6EA      751 ;
E6EA A0 00      752          ldy #ZERO
E6EC      753 ;
E6EC B1 5E      754          lda (INDEX),Y
E6EE A8          755          tay
E6EF      756 ;
E6EF 4C 01 E3 757          jmp SNGFLT
E6F2      758 ;
E6F2      759 ;
E6F2 4C 9B E1 760 IQ.ERR3  jmp IQ.ERR          ; Illegal Quantity error
E6F5      761 ;
E6F5      762 ;
E6F5 20 B1 00 763 GETBYTC  jsr CHRGET
E6F8      764 ;
E6F8      765 ;
E6F8      766 ; Reads a value less than 256 into FAC.
E6F8      767 ;
E6F8 20 64 DD 768 GETBYT  jsr FRMNUM
E6FB      769 ;
E6FB      770 ;
E6FB      771 ; Converts FAC into a single byte integer to X-reg.
E6FB      772 ;
E6FB 20 12 E1 773 CONVINT  jsr AYPOSINT
E6FE      774 ;
E6FE A6 A0      775          ldx FACMANT+2
E700 D0 F0      776          bne IQ.ERR3
E702      777 ;
E702 A6 A1      778          ldx FACMANT+3
E704      779 ;
E704 4C B7 00 780          jmp CHRGOT
E707      781 ;
E707      782 ;
E707 20 DC E6 783 FVAL      jsr GETSTRLN
E70A D0 03      784          bne >1
E70C      785 ;
E70C 4C 42 E8 786          jmp ZEROFAC
E70F      787 ;
E70F A6 B8      788 ^1      ldx TXTPTR
E711 A4 B9      789          ldy TXTPTR+1
E713      790 ;
E713 86 AD      791          stx STRING2
E715 84 AE      792          sty STRING2+1

```

```

E717          793 ;
E717 A6 5E    794      ldx INDEX
E719 A4 5F    795      ldz INDEX+1
E71B          796 ;
E71B 86 B8    797      stx TXTPTR
E71D 84 B9    798      sty TXTPTR+1
E71F          799 ;
E71F 18       800      clc
E720          801 ;
E720 65 5E    802      adc INDEX
E722 90 01    803      bcc >2
E724          804 ;
E724 C8       805      iny
E725          806 ;
E725 85 60    807      ^2 sta DEST
E727 84 61    808      sty DEST+1
E729          809 ;
E729 A0 00    810      ldz #ZERO
E72B          811 ;
E72B B1 60    812      lda (DEST),Y
E72D 48       813      pha
E72E          814 ;
E72E          815 ;
E72E          816 ; The following would cause a problem if DEST (i.e. HIMEM)
E72E          817 ; equals 0xC000. Since HIMEM is set to 0xBE00 or less in
E72E          818 ; DOS 4.5.08, the following code will not cause a problem.
E72E          819 ;
E72E A9 00    820      lda #ZERO
E730 91 60    821      sta (DEST),Y
E732          822 ;
E732 20 B7 00 823      jsr CHRGOT
E735 20 4A EC 824      jsr GETINT
E738          825 ;
E738 A0 00    826      ldz #ZERO
E73A          827 ;
E73A 68       828      pla
E73B 91 60    829      sta (DEST),Y
E73D          830 ;
E73D          831 ;
E73D A6 AD    832 STRCOPY ldz STRING2
E73F A4 AE    833      ldz STRING2+1
E741          834 ;
E741 86 B8    835      stx TXTPTR
E743 84 B9    836      sty TXTPTR+1
E745          837 ;
E745 60       838      rts
E746          839 ;
E746          840 ;
E746 20 64 DD 841 GETASNUM jsr FRMNUM
E749 20 52 E7 842      jsr GETADDR
E74C          843 ;
E74C          844 ;
E74C 20 BE DE 845 COMBYTE jsr CHKCOM
E74F          846 ;
E74F 4C F8 E6 847      jmp GETBYT
E752          848 ;
E752          849 ;
E752          850 ; Convert FAC to a 16-bit value into LINNUM.
E752          851 ;
E752 A5 9D    852 GETADDR lda FACEXP
E754 C9 91    853      cmp #$91

```

; FAC must be less than 2^16

```

E756 B0 9A      854      bcs IQ.ERR3
E758            855      ;
E758 20 F2 EB   856      jsr FP2INT
E75B            857      ;
E75B A5 A0      858      lda FACMANT+2
E75D A4 A1      859      ldy FACMANT+3
E75F            860      ;
E75F 84 50      861      sty LINNUM
E761 85 51      862      sta LINNUM+1
E763            863      ;
E763 60          864      rts
E764            865      ;
E764            866      ;
E764 A5 50      867 FPEEK   lda LINNUM
E766 48          868      pha
E767            869      ;
E767 A5 51      870      lda LINNUM+1
E769 48          871      pha
E76A            872      ;
E76A 20 52 E7   873      jsr GETADDR
E76D            874      ;
E76D A0 00      875      ldy #ZERO
E76F            876      ;
E76F B1 50      877      lda (LINNUM),Y
E771 A8          878      tay
E772            879      ;
E772 68          880      pla
E773 85 51      881      sta LINNUM+1
E775            882      ;
E775 68          883      pla
E776 85 50      884      sta LINNUM
E778            885      ;
E778 4C 01 E3   886      jmp SNGFLT
E77B            887      ;
E77B            888      ;
E77B 20 46 E7   889 BPOKE   jsr GETASNUM
E77E 8A          890      txa
E77F            891      ;
E77F A0 00      892      ldy #ZERO
E781            893      ;
E781 91 50      894      sta (LINNUM),Y
E783            895      ;
E783 60          896      rts
E784            897      ;
E784            898      ;
E784 20 46 E7   899 BWAIT   jsr GETASNUM
E787 86 85      900      stx FORPNT
E789            901      ;
E789 A2 00      902      ldx #ZERO
E78B            903      ;
E78B 20 B7 00   904      jsr CHRGOT
E78E F0 03      905      beq >1
E790            906      ;
E790 20 4C E7   907      jsr COMBYTE
E793            908      ;
E793 86 86      909 ^1      stx FORPNT+1
E795            910      ;
E795 A0 00      911      ldy #ZERO
E797            912      ;
E797 B1 50      913 ^2      lda (LINNUM),Y
E799 45 86      914      eor FORPNT+1

```

```

E79B          915 ;
E79B 25 85    916          and FORPNT
E79D F0 F8    917          beq <2
E79F          918 ;
E79F 60       919 RTN.E7.9 rts
E7A0          920 ;
E7A0          921 ;
E7A0          922          dfs 1,ZERO          ; 1 byte
E7A1          923 ;
E7A1          924 ;
E7A1 82 49 0F 925 FPPI          hex 82490FDAA2          ; 0xA221, FP PI value
E7A4 DA A2
E7A6 21       926 FFIGUARD hex 21          ; byte for FACGUARD
E7A7          927 ;
E7A7          928 ;
E7A7          929 ; FSUB routine entry point. Calculate ARG + (-FAC) -> FAC.
E7A7          930 ; Make FAC negative and recalculate FACSIGN and XORSIGN.
E7A7          931 ;
E7A7 20 E3 E9 932 FSUB          jsr LOADARG
E7AA          933 ;
E7AA          934 OMINUS:
E7AA A5 A2    935 SUB2          lda FACSIGN
E7AC 49 FF    936          eor #NEGONE
E7AE 85 A2    937          sta FACSIGN
E7B0          938 ;
E7B0 45 AA    939          eor ARGSIGN
E7B2 85 AB    940          sta XORSIGN
E7B4          941 ;
E7B4 A5 9D    942          lda FACEXP
E7B6          943 ;
E7B6 4C C1 E7 944          jmp ADD2
E7B9          945 ;
E7B9          946 ;
E7B9          947 ; Using the difference of ARG and FAC which is more than 7
E7B9          948 ; bits, shift the smaller argument one or more bytes to the
E7B9          949 ; right. Whatever is left over, shift that number of bits
E7B9          950 ; to the right. Then continue with the arithmetic.
E7B9          951 ;
E7B9 20 E5 E8 952 ^1          jsr SHFTBYT2
E7BC 90 30    953          bcc >5          ; always taken
E7BE          954 ;
E7BE          955 ;
E7BE          956 ; FADD routine entry point. Calculate ARG + FAC -> FAC.
E7BE          957 ;
E7BE 20 E3 E9 958 FADD          jsr LOADARG
E7C1          959 ;
E7C1          960 OPLUS:
E7C1 D0 03    961 ADD2          bne >2
E7C3          962 ;
E7C3 4C 53 EB 963          jmp COPYA2F          ; FAC = 0, return ARG
E7C6          964 ;
E7C6          965 ;
E7C6          966 ; Initially, set X-reg to 0xA5 and load A-reg with ARGEXP.
E7C6          967 ;
E7C6          968 ^2:
E7C6          969 ;          ldx FACGUARD          ; do not mess with guard bytes
E7C6          970 ;          stx ARGGUARD
E7C6          971 ;
E7C6 A2 A5    972          ldx #ARGEXP
E7C8 A5 A5    973          lda ARGEXP
E7CA          974 ;

```

```

E7CA      975 ;
E7CA      976 ; Test A-reg and return if zero. Subtract FACEXP and
E7CA      977 ; continue with arithmetic if equal. Branch if FAC > ARG.
E7CA      978 ;
E7CA A8    979 ADD3      tay
E7CB F0 D2 980          beq RTN.E7.9
E7CD      981 ;
E7CD 38    982          sec
E7CE      983 ;
E7CE E5 9D 984          sbc FACEXP
E7D0 F0 1C 985          beq >5
E7D2      986 ;
E7D2 90 0E 987          bcc >3
E7D4      988 ;
E7D4      989 ;
E7D4      990 ; ARG > FAC so make 2's compliment of difference, set
E7D4      991 ; FACEXP equal to ARGEXP, set FACSIGN equal to ARGSIGN,
E7D4      992 ; and set X-reg to 0x9D.
E7D4      993 ;
E7D4 84 9D 994          sty FACEXP
E7D6      995 ;
E7D6 A4 AA 996          ldy ARGSIGN
E7D8 84 A2 997          sty FACSIGN
E7DA      998 ;
E7DA 49 FF 999          eor #NEGONE
E7DC 69 00 1000         adc #ZERO
E7DE      1001 ;
E7DE      1002 ;          ldy #ZERO          ; do not mess with guard bytes
E7DE      1003 ;          sty ARGGUARD
E7DE      1004 ;
E7DE A2 9D 1005         ldx #FACEXP
E7E0 D0 00 1006         bne >4          ; always taken
E7E2      1007 ;
E7E2      1008 ;
E7E2      1009 ; FAC > ARG, X-reg is already set to 0xA5, and SHFTBYT2 or
E7E2      1010 ; SHFTBITS will shift ARG to the right.
E7E2      1011 ;
E7E2      1012 ^3:
E7E2      1013 ;          ldy #ZERO          ; do not mess with guard bytes
E7E2      1014 ;          sty FACGUARD
E7E2      1015 ;
E7E2      1016 ;
E7E2      1017 ; If exponent difference is greater than 7, branch.
E7E2      1018 ; Otherwise, set Y-reg to difference, set A-reg to
E7E2      1019 ; FACGUARD, and shift ARG or FAC that many bits right.
E7E2      1020 ;
E7E2 C9 F9 1021         ^4      cmp #!-7
E7E4 30 D3 1022         bmi <1
E7E6      1023 ;
E7E6 A8    1024         tay
E7E7      1025 ;
E7E7 A5 AC 1026         lda FACGUARD
E7E9      1027 ;
E7E9 56 01 1028         lsr MULMANT-OFFSET-0,X
E7EB      1029 ;
E7EB 20 FC E8 1030        jsr SHFTBITS
E7EE      1031 ;
E7EE      1032 ;
E7EE      1033         icl "E8.L"

```

LLOAD E8.L,A\$4000

```

E7EE      1          ttl "ROM Source Code, E8.L"
E7EE      2      ;
E7EE      3      ;
E7EE      4      ; E8.L
E7EE      5      ;
E7EE      6      ;
E7EE      7      ; If XORSIGN is positive, branch.  Set Y-reg to 0x9D.
E7EE      8      ;
E7EE 24 AB      9      ^5          bit XORSIGN
E7F0 10 57     10          bpl ADD4
E7F2      11      ;
E7F2 A0 9D     12          ldy #FACEXP
E7F4      13      ;
E7F4      14      ;
E7F4      15      ; If FAC > ARG, X-reg = 0xA5, so branch.
E7F4      16      ;
E7F4 E0 A5     17          cpx #ARGEXP
E7F6 F0 02     18          beq >6
E7F8      19      ;
E7F8      20      ;
E7F8      21      ; ARG > FAC, X-reg = 0x9D, so set Y-reg to 0xA5.
E7F8      22      ;
E7F8 A0 A5     23          ldy #ARGEXP
E7FA      24      ;
E7FA      25      ;
E7FA      26      ; A-reg contains FACGUARD from SHFTBITS routine.  Make A-
E7FA      27      ; reg negative and add in ARGGUARD.  ARGGUARD is equal to
E7FA      28      ; FACGUARD if FAC > ARG or zero if ARG > FAC.  If FAC >
E7FA      29      ; ARG, then FACGUARD is zero before ARG is shifted to the
E7FA      30      ; right.  Then subtract the mantissas and save to FAC.
E7FA      31      ; This subtraction will always leave C-flag set.
E7FA      32      ;
E7FA 38        33      ^6          sec
E7FB      34      ;
E7FB 49 FF     35          eor #NEGONE
E7FD 65 92     36          adc ARGGUARD
E7FF 85 AC     37          sta FACGUARD
E801      38      ;
E801 B9 04 00  39          lda FACMANT-FACEXP-3,Y
E804 F5 04     40          sbc ARGMANT-ARGEXP-3,X
E806 85 A1     41          sta FACMANT+3
E808      42      ;
E808 B9 03 00  43          lda FACMANT-FACEXP-2,Y
E80B F5 03     44          sbc ARGMANT-ARGEXP-2,X
E80D 85 A0     45          sta FACMANT+2
E80F      46      ;
E80F B9 02 00  47          lda FACMANT-FACEXP-1,Y
E812 F5 02     48          sbc ARGMANT-ARGEXP-1,X
E814 85 9F     49          sta FACMANT+1
E816      50      ;
E816 B9 01 00  51          lda FACMANT-FACEXP-0,Y
E819 F5 01     52          sbc ARGMANT-ARGEXP-0,X
E81B 85 9E     53          sta FACMANT
E81D      54      ;
E81D      55      ;
E81D      56      ; If C-flag is clear, make a 2's compliment of FAC.
E81D      57      ;
E81D B0 03     58      COMPFAC1 bcs NORMFAC1
E81F      59      ;
E81F 20 92 E8  60          jsr COMPFAC2

```

```

E822      61 ;
E822      62 ;
E822      63 ; Initialize the guard byte in Y-reg and the number of bits
E822      64 ; shifted in A-reg, both to zero. Initialize C-flag.
E822      65 ;
E822 A0 00 66 NORMFAC1 ldy #ZERO
E824 98    67          tya
E825      68 ;
E825 18    69          clc
E826      70 ;
E826      71 ;
E826      72 ; If FACMANT is not zero, branch. Otherwise, shift all
E826      73 ; bytes up to FACMANT setting the guard byte to zero.
E826      74 ; Can only do this four times at most; set FAC to zero.
E826      75 ;
E826 A6 9E 76 ^7      ldx FACMANT
E828 D0 4A 77          bne NORMFAC3
E82A      78 ;
E82A A6 9F 79          ldx FACMANT+1
E82C 86 9E 80          stx FACMANT
E82E      81 ;
E82E A6 A0 82          ldx FACMANT+2
E830 86 9F 83          stx FACMANT+1
E832      84 ;
E832 A6 A1 85          ldx FACMANT+3
E834 86 A0 86          stx FACMANT+2
E836      87 ;
E836 A6 AC 88          ldx FACGUARD
E838 86 A1 89          stx FACMANT+3
E83A      90 ;
E83A 84 AC 91          sty FACGUARD
E83C      92 ;
E83C 69 08 93          adc #BYTEBITS
E83E C9 20 94          cmp #MANTBITS
E840 D0 E4 95          bne <7
E842      96 ;
E842      97 ;
E842      98 ; Clear FACEXP and FACSIGN to zero.
E842      99 ;
E842 A9 00 100 ZEROFAC  lda #ZERO
E844      101 ;
E844 85 9D 102 ZEROFAC2 sta FACEXP
E846 85 A2 103          sta FACSIGN
E848      104 ;
E848 60    105          rts
E849      106 ;
E849      107 ;
E849      108 ; Add ARGGUARD and FACGUARD and the FAC and ARG mantissas.
E849      109 ; Branch if final mantissa needs to be normalized.
E849      110 ;
E849 65 92 111 ADD4     adc ARGGUARD
E84B 85 AC 112          sta FACGUARD
E84D      113 ;
E84D A5 A1 114          lda FACMANT+3
E84F 65 A9 115          adc ARGMANT+3
E851 85 A1 116          sta FACMANT+3
E853      117 ;
E853 A5 A0 118          lda FACMANT+2
E855 65 A8 119          adc ARGMANT+2
E857 85 A0 120          sta FACMANT+2
E859      121 ;

```

```

E859 A5 9F      122      lda FACMANT+1
E85B 65 A7      123      adc ARGMANT+1
E85D 85 9F      124      sta FACMANT+1
E85F           125      ;
E85F A5 9E      126      lda FACMANT
E861 65 A6      127      adc ARGMANT
E863 85 9E      128      sta FACMANT
E865           129      ;
E865 B0 1C      130      bcs NORMFAC4          ; accelerate code
E867           131      ;
E867 60         132      rts
E868           133      ;
E868           134      ;
E868           135      ; Shift FAC mantissa left and increment A-reg each time
E868           136      ; using bits from FACGUARD.
E868           137      ;
E868 69 01      138      NORMFAC2 adc #1
E86A           139      ;
E86A 06 AC      140      asl FACGUARD
E86C           141      ;
E86C 26 A1      142      rol FACMANT+3
E86E 26 A0      143      rol FACMANT+2
E870 26 9F      144      rol FACMANT+1
E872 26 9E      145      rol FACMANT
E874           146      ;
E874           147      ;
E874           148      ; A-reg = 0, branch if X-reg (or FACMANT) < 0x80.  FACMANT
E874           149      ; needs to have its MSB set.
E874           150      ;
E874           151      ; Recalculate FAC exponent using content of A-reg and 2's
E874           152      ; compliment.
E874           153      ;
E874 10 F2      154      NORMFAC3 bpl NORMFAC2
E876           155      ;
E876 38         156      sec
E877           157      ;
E877 E5 9D      158      sbc FACEXP
E879 B0 C7      159      bcs ZEROFAC
E87B           160      ;
E87B 49 FF      161      eor #NEGONE
E87D           162      ;
E87D 69 01      163      adc #1
E87F 85 9D      164      sta FACEXP
E881 90 0E      165      bcc RTN.E8.9
E883           166      ;
E883           167      ;
E883           168      ; Increment exponent, then shift mantissa right.
E883           169      ;
E883 E6 9D      170      NORMFAC4 inc FACEXP
E885 F0 42      171      beq OF.ERR          ; Overflow error
E887           172      ;
E887 66 9E      173      ror FACMANT
E889 66 9F      174      ror FACMANT+1
E88B 66 A0      175      ror FACMANT+2
E88D 66 A1      176      ror FACMANT+3
E88F           177      ;
E88F 66 AC      178      ror FACGUARD
E891           179      ;
E891 60         180      RTN.E8.9 rts
E892           181      ;
E892           182      ;

```



```

E892      183 ; Make 2's compliment of FAC sign and mantissa.
E892      184 ;
E892 A5 A2 185 COMPFAC2 lda FACSIGN
E894 49 FF 186          eor #NEGONE
E896 85 A2 187          sta FACSIGN
E898      188 ;
E898      189 ;
E898      190 ; Make 2's compliment of FAC mantissa.
E898      191 ;
E898 A5 9E 192 COMPMANT lda FACMANT
E89A 49 FF 193          eor #NEGONE
E89C 85 9E 194          sta FACMANT
E89E      195 ;
E89E A5 9F 196          lda FACMANT+1
E8A0 49 FF 197          eor #NEGONE
E8A2 85 9F 198          sta FACMANT+1
E8A4      199 ;
E8A4 A5 A0 200          lda FACMANT+2
E8A6 49 FF 201          eor #NEGONE
E8A8 85 A0 202          sta FACMANT+2
E8AA      203 ;
E8AA A5 A1 204          lda FACMANT+3
E8AC 49 FF 205          eor #NEGONE
E8AE 85 A1 206          sta FACMANT+3
E8B0      207 ;
E8B0 A5 AC 208          lda FACGUARD
E8B2 49 FF 209          eor #NEGONE
E8B4 85 AC 210          sta FACGUARD
E8B6      211 ;
E8B6 E6 AC 212          inc FACGUARD
E8B8 D0 0E 213          bne RTN.E8.D
E8BA      214 ;
E8BA      215 ;
E8BA      216 ; Increment FAC mantissa.
E8BA      217 ;
E8BA E6 A1 218 INCMANT inc FACMANT+3
E8BC D0 0A 219          bne RTN.E8.D
E8BE      220 ;
E8BE E6 A0 221          inc FACMANT+2
E8C0 D0 06 222          bne RTN.E8.D
E8C2      223 ;
E8C2 E6 9F 224          inc FACMANT+1
E8C4 D0 02 225          bne RTN.E8.D
E8C6      226 ;
E8C6 E6 9E 227          inc FACMANT
E8C8      228 ;
E8C8 60      229 RTN.E8.D rts
E8C9      230 ;
E8C9      231 ;
E8C9 A2 45 232 OF.ERR ldx #MSG06-MESGS ; Overflow error
E8CB      233 ;
E8CB 4C 12 D4 234          jmp PRERR
E8CE      235 ;
E8CE      236 ;
E8CE      237 ; Called by FMULT when A-reg is zero. OFFSET is set to
E8CE      238 ; MULMANT-1 in order to show that the bytes of MULMANT are
E8CE      239 ; being shifted right. No other routine calls SHFTBYT1.
E8CE      240 ; The bytes must be moved in this order.
E8CE      241 ;
E8CE 38      242 SHFTBYT1 sec ; escape for FMULT
E8CF      243 ;

```

```

E8CF A2 61      244      ldx #MULMANT-1
E8D1           245      ;
E8D1 B4 04      246      ^1      ld  MULMANT-OFFSET-3,X
E8D3 84 AC      247      sty  FACGUARD
E8D5           248      ;
E8D5 B4 03      249      ld  MULMANT-OFFSET-2,X
E8D7 94 04      250      sty  MULMANT-OFFSET-3,X
E8D9           251      ;
E8D9 B4 02      252      ld  MULMANT-OFFSET-1,X
E8DB 94 03      253      sty  MULMANT-OFFSET-2,X
E8DD           254      ;
E8DD B4 01      255      ld  MULMANT-OFFSET-0,X
E8DF 94 02      256      sty  MULMANT-OFFSET-1,X
E8E1           257      ;
E8E1 A4 A4      258      ld  EXTSIGN          ; zero except for FP2INT
E8E3 94 01      259      sty  MULMANT-OFFSET-0,X
E8E5           260      ;
E8E5           261      ;
E8E5           262      ; Show that eight bits have been moved to the right.
E8E5           263      ;
E8E5 69 08      264      SHFTBYT2 adc #BYTEBITS
E8E7 30 E8      265      bmi <1
E8E9           266      ;
E8E9 F0 E6      267      beq <1
E8EB           268      ;
E8EB           269      ;
E8EB           270      ; Calculate the remaining bits to be moved to the right.
E8EB           271      ;
E8EB E9 08      272      sbc #BYTEBITS
E8ED A8         273      tay
E8EE           274      ;
E8EE A5 AC      275      lda  FACGUARD
E8F0           276      ;
E8F0 B0 14      277      bcs >4          ; escape for FMULT
E8F2           278      ;
E8F2 16 01      279      ^2      asl  MULMANT-OFFSET-0,X
E8F4 90 02      280      bcc >3
E8F6           281      ;
E8F6 F6 01      282      inc  MULMANT-OFFSET-0,X
E8F8           283      ;
E8F8 76 01      284      ^3      ror  MULMANT-OFFSET-0,X
E8FA 76 01      285      ror  MULMANT-OFFSET-0,X
E8FC           286      ;
E8FC 76 02      287      SHFTBITS ror  MULMANT-OFFSET-1,X
E8FE 76 03      288      ror  MULMANT-OFFSET-2,X
E900 76 04      289      ror  MULMANT-OFFSET-3,X
E902           290      ;
E902 6A         291      ror                      ; FACGUARD
E903           292      ;
E903 C8         293      iny                      ; if Y-reg is 0x00, done
E904 D0 EC      294      bne <2
E906           295      ;
E906 18         296      ^4      clc
E907           297      ;
E907 60         298      rts
E908           299      ;
E908           300      ;
E908           301      ; Confine PDL statement to arguments of 0:3 only.
E908           302      ;
E908 E0 04      303      PDL2      cpx #MAXPDL+1
E90A B0 03      304      bcs >1

```

```

E90C          305 ;
E90C 4C 1E FB 306      jmp PREAD
E90F          307 ;
E90F 4C 9B E1 308 ^1    jmp IQ.ERR
E912          309 ;
E912          310 ;
E912          311      dfs 1,ZERO          ; 1 byte
E913          312 ;
E913          313 ;
E913          314 ; Changed order of variables.  Removed first FP1.0 since
E913          315 ; there are duplicates in other locations and put FPLOGE
E913          316 ; here.
E913          317 ;
E913 7F 5E 5B 318 FPLOGE  hex 7F5E5BD8A9          ; FP LOG(e) = 0.434294481904
E916 D8 A9
E918 80 35 04 319 FPSQR0.5 hex 803504F334          ; FP 0.5^0.5 = 0.707106781
E91B F3 34
E91D 81 35 04 320 FPSQR2.0 hex 813504F334          ; FP 2.0^0.5 = 1.414213562
E920 F3 34
E922 80 80 00 321 FPN0.5  hex 8080000000          ; FP -0.5
E925 00 00
E927 80 31 72 322 FPLN2   hex 80317217F8          ; FP LN(2) = 0.693147181
E92A 17 F8
E92C          323 ;
E92C          324 ;
E92C 03        325 POLY.LOG hex 03          ; number of coefficients-1
E92D 7F 5E 56 326      hex 7F5E56CB79          ; FP 0.434255942 * X^7 +
E930 CB 79
E932 80 13 9B 327      hex 80139B0B64          ; FP 0.576584541 * X^5 +
E935 0B 64
E937 80 76 38 328      hex 8076389316          ; FP 0.961800759 * X^3 +
E93A 93 16
E93C 82 38 AA 329      hex 8238AA3B20          ; FP 2.88539007 * X
E93F 3B 20
E941          330 ;
E941          331 ;
E941          332 ; This was originally the FLOG entry.  However, this code
E941          333 ; calculates the natural logarithm of FAC leaving the
E941          334 ; result in FAC.  The FLOG conversion routine is at 0xF1CB.
E941          335 ; Fall into FMULT to multiply the final result with ln(2).
E941          336 ;
E941 20 82 EB 337 FLN      jsr SIGNCHK
E944 F0 02     338      beq >1
E946          339 ;
E946 10 03     340      bpl >2          ; C-flag is clear
E948          341 ;
E948 4C 9B E1 342 ^1      jmp IQ.ERR
E94B          343 ;
E94B A5 9D     344 ^2      lda FACEXP          ; save unbiased exponent
E94D E9 7F     345      sbc #EXPBIAS-1
E94F 48        346      pha
E950          347 ;
E950 A9 80     348      lda #EXPBIAS          ; normalize from 0.5 to 1.0
E952 85 9D     349      sta FACEXP
E954          350 ;
E954 A9 18     351      lda #FPSQR0.5
E956 A0 E9     352      ldy /FPSQR0.5
E958 20 BE E7 353      jsr FADD
E95B          354 ;
E95B A9 1D     355      lda #FPSQR2.0
E95D A0 E9     356      ldy /FPSQR2.0

```

```

E95F 20 66 EA 357      jsr FDIV
E962          358      ;
E962 A9 04      359      lda #FP1.0
E964 A0 EF      360      ldy /FP1.0
E966 20 A7 E7   361      jsr FSUB
E969          362      ;
E969 A9 2C      363      lda #POLY.LOG
E96B A0 E9      364      ldy /POLY.LOG
E96D 20 5B EF   365      jsr POLYPROC
E970          366      ;
E970 A9 22      367      lda #FPN0.5
E972 A0 E9      368      ldy /FPN0.5
E974 20 BE E7   369      jsr FADD
E977          370      ;
E977 68          371      pla
E978 20 F6 EC   372      jsr ADD2FAC
E97B          373      ;
E97B A9 27      374      lda #FPLN2          ; FP LN(2)
E97D A0 E9      375      ldy /FPLN2
E97F          376      ;
E97F          377      ;
E97F          378      ; FMULT routine entry point; product = ARG * FAC -> FAC.
E97F          379      ; The entry to MULT2 (OMULT) is slightly modified.
E97F          380      ;
E97F 20 E3 E9   381      FMULT      jsr LOADARG
E982          382      ;
E982          383      OMULT:
E982 F0 5E      384      MULT2      beq >3          ; value from FACEXP
E984          385      ;
E984 20 10 EA    386      jsr PROCEXP          ; process exponents
E987          387      ;
E987 A0 00      388      ldy #ZERO          ; initialize MULMANT
E989 20 CC F1    389      jsr CLEARMUL
E98C          390      ;
E98C          391      ;
E98C          392      ; Utilize the FACGUARD content from a previous calculation.
E98C          393      ;
E98C A5 AC      394      lda FACGUARD
E98E 84 AC      395      sty FACGUARD          ; initialize to zero
E990 20 AA E9    396      jsr TSTMULT
E993          397      ;
E993 A5 A1      398      lda FACMANT+3
E995 20 AA E9    399      jsr TSTMULT
E998          400      ;
E998 A5 A0      401      lda FACMANT+2
E99A 20 AA E9    402      jsr TSTMULT
E99D          403      ;
E99D A5 9F      404      lda FACMANT+1
E99F 20 AA E9    405      jsr TSTMULT
E9A2          406      ;
E9A2 A5 9E      407      lda FACMANT          ; multiply FAC by all 8 bits
E9A4 20 AF E9    408      jsr BYTMULT
E9A7          409      ;
E9A7 4C E6 EA    410      jmp COPYM2F          ; finish, copy MULMANT to FAC
E9AA          411      ;
E9AA          412      ;
E9AA          413      ; If zero, copy MULMANT bytes to the right; faster process.
E9AA          414      ;
E9AA D0 03      415      TSTMULT      bne BYTMULT
E9AC          416      ;
E9AC 4C CE E8    417      jmp SHFTBYT1          ; copy MULMANT bytes right

```

```

E9AF          418 ;
E9AF          419 ;
E9AF          420 ; Multiply FAC with bits in A-reg. Use A-reg as an 8-bit
E9AF          421 ; counter by shifting right and setting its MSB.
E9AF          422 ;
E9AF 4A       423 BYTMULT lsr                ; put LSB into carry
E9B0          424 ;
E9B0 09 80    425         ora #MSBSET        ; setup 8-bit counter
E9B2          426 ;
E9B2 A8       427 ^1      tay                ; save for below; or PHA/PLA
E9B3          428 ;
E9B3 90 1F    429         bcc >2            ; bypass addition
E9B5          430 ;
E9B5 18       431         clc                ; add ARG to MULMANT
E9B6          432 ;
E9B6 A5 AC    433         lda FACGUARD
E9B8 65 92    434         adc ARGGUARD
E9BA 85 AC    435         sta FACGUARD
E9BC          436 ;
E9BC A5 65    437         lda MULMANT+3
E9BE 65 A9    438         adc ARGMANT+3
E9C0 85 65    439         sta MULMANT+3
E9C2          440 ;
E9C2 A5 64    441         lda MULMANT+2
E9C4 65 A8    442         adc ARGMANT+2
E9C6 85 64    443         sta MULMANT+2
E9C8          444 ;
E9C8 A5 63    445         lda MULMANT+1
E9CA 65 A7    446         adc ARGMANT+1
E9CC 85 63    447         sta MULMANT+1
E9CE          448 ;
E9CE A5 62    449         lda MULMANT
E9D0 65 A6    450         adc ARGMANT
E9D2 85 62    451         sta MULMANT
E9D4          452 ;
E9D4          453 ;
E9D4          454 ; Shift MULMANT and FACGUARD right for next addition.
E9D4          455 ;
E9D4 66 62    456 ^2      ror MULMANT
E9D6 66 63    457         ror MULMANT+1
E9D8 66 64    458         ror MULMANT+2
E9DA 66 65    459         ror MULMANT+3
E9DC          460 ;
E9DC 66 AC    461         ror FACGUARD
E9DE          462 ;
E9DE          463 ;
E9DE          464 ; Recall A-reg, shift right, and put next bit into carry.
E9DE          465 ; When the 8-bit counter is complete, this byte is done.
E9DE          466 ;
E9DE 98       467         tya
E9DF          468 ;
E9DF 4A       469         lsr
E9E0 D0 D0    470         bne <1
E9E2          471 ;
E9E2 60       472 ^3      rts
E9E3          473 ;
E9E3          474 ;
E9E3          475 ; LOADARG routine entry point, (A/Y) -> ARG.
E9E3          476 ;
E9E3 85 5E    477 LOADARG sta INDEX
E9E5 84 5F    478         sty INDEX+1

```

```
E9E7          479 ;
E9E7 A0 04    480      ldy #4
E9E9          481 ;
E9E9 B1 5E    482      lda (INDEX),Y
E9EB 85 A9    483      sta ARGMANT+3
E9ED          484 ;
E9ED 88       485      dey
E9EE          486 ;
E9EE B1 5E    487      lda (INDEX),Y
E9F0 85 A8    488      sta ARGMANT+2
E9F2          489 ;
E9F2 88       490      dey
E9F3          491 ;
E9F3 B1 5E    492      lda (INDEX),Y
E9F5 85 A7    493      sta ARGMANT+1
E9F7          494 ;
E9F7 88       495      dey
E9F8          496 ;
E9F8 B1 5E    497      lda (INDEX),Y
E9FA 85 AA    498      sta ARGSIGN
E9FC          499 ;
E9FC 45 A2    500      eor FACSIGN
E9FE 85 AB    501      sta XORSIGN
EA00          502 ;
EA00 A5 AA    503      lda ARGSIGN
EA02 09 80    504      ora #EXPBIAS
EA04 85 A6    505      sta ARGMANT
EA06          506 ;
EA06 88       507      dey
EA07          508 ;
EA07 B1 5E    509      lda (INDEX),Y
EA09 85 A5    510      sta ARGEXP
EA0B          511 ;
EA0B 84 92    512      sty ARGGUARD      ; initialize to zero
EA0D          513 ;
EA0D A5 9D    514      lda FACEXP
EA0F          515 ;
EA0F 60       516      rts
EA10          517 ;
EA10          518 ;
EA10          519      icl "EA.L"
```

LLOAD EA.L,A\$4000

```

EA10      1          ttl "ROM Source Code, EA.L"
EA10      2      ;
EA10      3      ;
EA10      4      ; EA.L
EA10      5      ;
EA10      6      ;
EA10      7      ; Process exponents entry point; initialize FACSIGN.
EA10      8      ; PROCEXP has been rewritten and is 3 bytes shorter.
EA10      9      ; If FACEXP is zero, do not make FACSIGN positive.
EA10     10      ;
EA10 A5 A5     11 PROCEXP  lda ARGEXP
EA12      12      ;
EA12 F0 14     13 PROCEXP2 beq PROCEXP3
EA14      14      ;
EA14 18       15          clc
EA15      16      ;
EA15 65 9D     17          adc FACEXP
EA17 90 04     18          bcc >1
EA19      19      ;
EA19 30 16     20          bmi OF.ERR2          ; Overflow error
EA1B      21      ;
EA1B 18       22          clc
EA1C      23      ;
EA1C 2C 00 00  24          bit *-*          ; bypass next instruction
EA1F      25          dfs !-2
EA1D      26      ;
EA1D 10 09     27 ^1      bpl PROCEXP3
EA1F      28      ;
EA1F 69 80     29 ^2      adc #EXPBIAS
EA21 85 9D     30          sta FACEXP          ; even if zero use XORSIGN
EA23      31      ;
EA23 A5 AB     32          lda XORSIGN
EA25 85 A2     33          sta FACSIGN
EA27      34      ;
EA27 60       35          rts
EA28      36      ;
EA28      37      ;
EA28      38      ; Adjust STACK pointer; set FACEXP and FACSIGN to zero.
EA28      39      ;
EA28 68       40 PROCEXP3 pla
EA29 68       41          pla
EA2A      42      ;
EA2A 4C 42 E8  43          jmp ZEROFAC
EA2D      44      ;
EA2D      45      ;
EA2D      46      ; Check for zero or overflow error.
EA2D      47      ;
EA2D A5 A2     48 ZEROFERR lda FACSIGN
EA2F 30 F7     49          bmi PROCEXP3
EA31      50      ;
EA31      51      ;
EA31 A2 45     52 OF.ERR2 ldx #MSG06-MESGS      ; Overflow error
EA33      53      ;
EA33 2C 00 00  54          bit *-*
EA36      55          dfs !-2
EA34      56      ;
EA34 A2 89     57 DZ.ERR  ldx #MSG11-MESGS      ; Division by Zero error
EA36      58      ;
EA36 4C 12 D4  59          jmp PRterr
EA39      60      ;

```

```

EA39      61 ;
EA39      62 ; COPYF2A does not call RNDUP and returns with FACEXP in
EA39      63 ; A-reg.  FACGUARD is copied to ARGGUARD.
EA39      64 ;
EA39 20 63 EB 65 MULFAC10 jsr COPYF2A
EA3C F0 10    66          beq >1
EA3E      67 ;
EA3E 18      68          clc
EA3F      69 ;
EA3F 69 02    70          adc #2
EA41 B0 EE    71          bcs OF.ERR2
EA43      72 ;
EA43 A2 00    73          ldx #ZERO
EA45 86 AB    74          stx XORSIGN
EA47      75 ;
EA47 20 CA E7 76          jsr ADD3
EA4A      77 ;
EA4A E6 9D    78          inc FACEXP
EA4C F0 E3    79          beq OF.ERR2
EA4E      80 ;
EA4E 60      81 ^1      rts
EA4F      82 ;
EA4F      83 ;
EA4F 84 20 00 84 FP10.0  hex 8420000000      ; FP 10.0
EA52 00 00
EA54      85 ;
EA54      86 ;
EA54      87 ; Move FAC to ARG in order to divide ARG by 10 -> FAC.
EA54      88 ;
EA54 20 63 EB 89 DIVFAC10 jsr COPYF2A
EA57      90 ;
EA57 A9 4F    91          lda #FP10.0
EA59 A0 EA    92          ldy /FP10.0
EA5B 20 F9 EA 93          jsr LOADFAC
EA5E      94 ;
EA5E 84 AB    95          sty XORSIGN      ; Y-reg = 0
EA60      96 ;
EA60 4C 69 EA 97          jmp DIV2
EA63      98 ;
EA63      99 ;
EA63     100          dfs 3,ZERO      ; 3 bytes
EA66     101 ;
EA66     102 ;
EA66     103 ; FDIV routine entry point; quotient = ARG / FAC -> FAC.
EA66     104 ; This routine has been rewritten and reorganized in order
EA66     105 ; to produce an 8-bit FACGUARD.  FACGUARD and ARGGUARD are
EA66     106 ; both utilized so RNDUP call deleted.
EA66     107 ;
EA66 20 E3 E9 108 FDIV      jsr LOADARG
EA69     109 ;
EA69     110 ODIVIDE:
EA69 F0 C9    111 DIV2      beq DZ.ERR      ; value from ARGEXP or FACEXP
EA6B     112 ;
EA6B     113 ;
EA6B     114 ; ARGGUARD has already been initialized appropriately.
EA6B     115 ; Make 2's compliment of FACEXP and then process exponents.
EA6B     116 ;
EA6B 38      117          sec
EA6C     118 ;
EA6C A9 00    119          lda #ZERO
EA6E E5 9D    120          sbc FACEXP

```



```

EA70 85 9D      121      sta FACEXP
EA72           122      ;
EA72 20 10 EA   123      jsr PROCEXP
EA75           124      ;
EA75 E6 9D     125      inc FACEXP
EA77 F0 B8     126      beq OF.ERR2
EA79           127      ;
EA79           128      ;
EA79           129      ; Setup a 5-byte counter in X-reg and an 8-bit counter in
EA79           130      ; A-reg. Test if FAC can be subtracted from ARG and rol
EA79           131      ; carry into A-reg as an LSB. If carry is set, perform
EA79           132      ; the subtraction. A-reg contains the quotient byte.
EA79           133      ;
EA79 A2 FB     134      ldx #!-5                ; byte counter for MULMANT
EA7B           135      ;
EA7B A9 01     136      lda #1                ; bit counter for byte
EA7D           137      ;
EA7D A4 A6     138      ^1 ldy ARGMANT
EA7F C4 9E     139      cpy FACMANT
EA81 D0 16     140      bne >2
EA83           141      ;
EA83 A4 A7     142      ldy ARGMANT+1
EA85 C4 9F     143      cpy FACMANT+1
EA87 D0 10     144      bne >2
EA89           145      ;
EA89 A4 A8     146      ldy ARGMANT+2
EA8B C4 A0     147      cpy FACMANT+2
EA8D D0 0A     148      bne >2
EA8F           149      ;
EA8F A4 A9     150      ldy ARGMANT+3
EA91 C4 A1     151      cpy FACMANT+3
EA93 D0 04     152      bne >2
EA95           153      ;
EA95 A4 92     154      ldy ARGGUARD
EA97 C4 AC     155      cpy FACGUARD
EA99           156      ;
EA99 08        157      ^2 php                ; save comparison in carry
EA9A           158      ;
EA9A 2A        159      rol                ; save carry value
EA9B 90 07     160      bcc >3                ; counter expired?
EA9D           161      ;
EA9D           162      ;
EA9D           163      ; Bit counter expired; save the quotient byte in MULMANT.
EA9D           164      ;
EA9D E8        165      inx
EA9E F0 43     166      beq >5
EAA0           167      ;
EAA0 95 66     168      sta MULMANT+4,X
EAA2           169      ;
EAA2 A9 01     170      lda #1                ; load bit counter again
EAA4           171      ;
EAA4           172      ;
EAA4           173      ; Recall carry status; if set, do subtraction.
EAA4           174      ;
EAA4 28        175      ^3 plp
EAA5 90 20     176      bcc >4
EAA7           177      ;
EAA7           178      ;
EAA7           179      ; Copy A-reg, subtract FAC from ARC including their guard
EAA7           180      ; bytes, and restore A-reg.
EAA7           181      ;

```

```

EAA7 A8          182          tay
EAA8            183          ;
EAA8 A5 92       184          lda ARGGUARD
EAAA E5 AC       185          sbc FACGUARD
EAAC 85 92       186          sta ARGGUARD
EAAE            187          ;
EAAE A5 A9       188          lda ARGMANT+3
EAB0 E5 A1       189          sbc FACMANT+3
EAB2 85 A9       190          sta ARGMANT+3
EAB4            191          ;
EAB4 A5 A8       192          lda ARGMANT+2
EAB6 E5 A0       193          sbc FACMANT+2
EAB8 85 A8       194          sta ARGMANT+2
EABA            195          ;
EABA A5 A7       196          lda ARGMANT+1
EABC E5 9F       197          sbc FACMANT+1
EABE 85 A7       198          sta ARGMANT+1
EAC0            199          ;
EAC0 A5 A6       200          lda ARGMANT
EAC2 E5 9E       201          sbc FACMANT
EAC4 85 A6       202          sta ARGMANT
EAC6            203          ;
EAC6 98          204          tya
EAC7            205          ;
EAC7            206          ;
EAC7            207          ; Shift ARG one bit left.
EAC7            208          ;
EAC7 06 92       209          ^4      asl ARGGUARD
EAC9            210          ;
EAC9 26 A9       211          rol ARGMANT+3
EACB 26 A8       212          rol ARGMANT+2
EACD 26 A7       213          rol ARGMANT+1
EACF 26 A6       214          rol ARGMANT
EAD1            215          ;
EAD1 B0 C6       216          bcs <2          ; capture carry to subtract
EAD3 10 C4       217          bpl <2          ; ARG < FAC, capture carry
EAD5            218          ;
EAD5 30 A6       219          bmi <1          ; always, check if ARG > FAC
EAD7            220          ;
EAD7            221          ;
EAD7 7F 00 00    222          FP.25      hex 7F00000000          ; FP 0.25
EADA 00 00       223          FP1.0E9    hex 9E6E6B2800          ; FP 1.0E+09 = 1000000000.0
EADF 28 00       224          ;
EAE1            225          ;
EAE1            226          dfs 2,ZERO          ; 2 bytes
EAE3            227          ;
EAE3            228          ;
EAE3 28          229          ^5      plp          ; adjust STACK pointer
EAE4            230          ;
EAE4 85 AC       231          sta FACGUARD          ; a complete byte
EAE6            232          ;
EAE6            233          ;
EAE6            234          ; Copy MULMANT to FACMANT, MUL -> FAC.
EAE6            235          ;
EAE6 A5 62       236          COPYM2F      lda MULMANT
EAE8 85 9E       237          sta FACMANT
EAEA            238          ;
EAEA A5 63       239          lda MULMANT+1
EAEC 85 9F       240          sta FACMANT+1

```

```

EAEE          241 ;
EAEE A5 64     242         lda MULMANT+2
EAF0 85 A0     243         sta FACMANT+2
EAF2          244 ;
EAF2 A5 65     245         lda MULMANT+3
EAF4 85 A1     246         sta FACMANT+3
EAF6          247 ;
EAF6 4C 22 E8  248         jmp NORMFAC1             ; finalize exponent
EAF9          249 ;
EAF9          250 ;
EAF9          251 ; LOADFAC routine entry point, (A/Y) -> FAC.
EAF9          252 ;
EAF9 85 5E     253 LOADFAC sta INDEX
EAFB 84 5F     254         sty INDEX+1
EAFD          255 ;
EAFD A0 04     256         ldy #4
EAFF          257 ;
EAFF B1 5E     258         lda (INDEX),Y
EB01 85 A1     259         sta FACMANT+3
EB03          260 ;
EB03 88        261         dey
EB04          262 ;
EB04 B1 5E     263         lda (INDEX),Y
EB06 85 A0     264         sta FACMANT+2
EB08          265 ;
EB08 88        266         dey
EB09          267 ;
EB09 B1 5E     268         lda (INDEX),Y
EB0B 85 9F     269         sta FACMANT+1
EB0D          270 ;
EB0D 88        271         dey
EB0E          272 ;
EB0E B1 5E     273         lda (INDEX),Y
EB10 85 A2     274         sta FACSIGN
EB12          275 ;
EB12 09 80     276         ora #EXPBIAS
EB14 85 9E     277         sta FACMANT
EB16          278 ;
EB16 88        279         dey
EB17          280 ;
EB17 B1 5E     281         lda (INDEX),Y
EB19 85 9D     282         sta FACEXP
EB1B          283 ;
EB1B 84 AC     284         sty FACGUARD             ; make it zero
EB1D          285 ;
EB1D 60        286         rts
EB1E          287 ;
EB1E          288 ;
EB1E A2 98     289 COPYF2T2 ldx #TEMP2             ; FAC -> (TEMP2)
EB20          290 ;
EB20 2C 00 00  291         bit *-*
EB23          292         dfs !-2
EB21          293 ;
EB21 A2 93     294 COPYF2T1 ldx #TEMP1             ; FAC -> (TEMP1)
EB23          295 ;
EB23 A0 00     296         ldy /TEMP1
EB25 F0 04     297         beq COPYFAC             ; always taken
EB27          298 ;
EB27 A6 85     299 COPYF2FR ldx FORPNT             ; FAC -> (FORPNT)
EB29 A4 86     300         ldy FORPNT+1
EB2B          301 ;

```

```

EB2B          302 ;
EB2B          303 ; COPYFAC routine entry point, FAC -> (X/Y).  Leave
EB2B          304 ; FACGUARD alone.
EB2B          305 ;
EB2B 20 70 EB 306 COPYFAC jsr RNDUP ; process FACGUARD for roundup
EB2E          307 ;
EB2E 86 5E     308 COPYFAC2 stx INDEX
EB30 84 5F     309 sty INDEX+1
EB32          310 ;
EB32 A6 AC     311 ldx FACGUARD
EB34          312 ;
EB34 A0 04     313 ldy #4
EB36          314 ;
EB36 A5 A1     315 lda FACMANT+3
EB38 91 5E     316 sta (INDEX),Y
EB3A          317 ;
EB3A 88        318 dey
EB3B          319 ;
EB3B A5 A0     320 lda FACMANT+2
EB3D 91 5E     321 sta (INDEX),Y
EB3F          322 ;
EB3F 88        323 dey
EB40          324 ;
EB40 A5 9F     325 lda FACMANT+1
EB42 91 5E     326 sta (INDEX),Y
EB44          327 ;
EB44 88        328 dey
EB45          329 ;
EB45 A5 A2     330 lda FACSIGN
EB47 09 7F     331 ora #$7F
EB49          332 ;
EB49 25 9E     333 and FACMANT
EB4B 91 5E     334 sta (INDEX),Y
EB4D          335 ;
EB4D 88        336 dey
EB4E          337 ;
EB4E A5 9D     338 lda FACEXP
EB50 91 5E     339 sta (INDEX),Y
EB52          340 ;
EB52 60        341 rts
EB53          342 ;
EB53          343 ;
EB53          344 ; Copy ARG to FAC routine entry point, ARG -> FAC.  This
EB53          345 ; routine is used by ADD2 when FACEXP is zero and by POWER.
EB53          346 ;
EB53 A5 AA     347 COPYA2F lda ARGSIGN
EB55          348 ;
EB55 85 A2     349 COPYA2F2 sta FACSIGN
EB57          350 ;
EB57 A2 05     351 ldx #FACLEN
EB59          352 ;
EB59 B5 A4     353 ^1 lda ARGEXP-1,X
EB5B 95 9C     354 sta FACEXP-1,X
EB5D          355 ;
EB5D CA       356 dex
EB5E D0 F9     357 bne <1
EB60          358 ;
EB60 4C 8E F6 359 jmp COPYA2F3 ; continue the routine
EB63          360 ;
EB63          361 ;
EB63          362 ; Copy FAC to ARG routine entry point, FAC -> ARG.  This

```

```

EB63      363 ; routine has been expanded and modified. The call to
EB63      364 ; RNDUP has been removed and FACGUARD -> ARGGUARD.
EB63      365 ;
EB63 A5 A2 366 COPYF2A lda FACSIGN
EB65 85 AA 367          sta ARGSIGN
EB67      368 ;
EB67 A2 00 369          ldx #ZERO          ; for FEXP
EB69      370 ;
EB69 A5 A1 371          lda FACMANT+3
EB6B 85 A9 372          sta ARGMANT+3
EB6D      373 ;
EB6D 4C 93 F6 374          jmp COPYF2A2          ; continue the routine
EB70      375 ;
EB70      376 ;
EB70      377 ; Roundup routine entry point; test FACGUARD before
EB70      378 ; modifying its value.
EB70      379 ;
EB70 A5 9D 380 RNDUP      lda FACEXP
EB72 F0 1B 381          beq RTN.EB.8
EB74      382 ;
EB74 A5 AC 383          lda FACGUARD
EB76 10 17 384          bpl RTN.EB.8
EB78      385 ;
EB78 06 AC 386          asl FACGUARD          ; now extract the MSB
EB7A      387 ;
EB7A 20 BA E8 388          jsr INCMANT          ; increment FAC mantissa
EB7D D0 10 389          bne RTN.EB.8
EB7F      390 ;
EB7F 4C 83 E8 391          jmp NORMFAC4          ; C-flag still set
EB82      392 ;
EB82      393 ;
EB82      394 ; Test FAC for negative, zero, and positive.
EB82      395 ;
EB82 A5 9D 396 SIGNCHK   lda FACEXP
EB84 F0 09 397          beq RTN.EB.8
EB86      398 ;
EB86 A5 A2 399 SIGNCHK2  lda FACSIGN
EB88      400 ;
EB88 2A      401 SIGNCHK3  rol
EB89      402 ;
EB89 A9 01 403          lda #1
EB8B      404 ;
EB8B 90 02 405          bcc RTN.EB.8
EB8D      406 ;
EB8D A9 FF 407          lda #NEGONE
EB8F      408 ;
EB8F 60      409 RTN.EB.8 rts
EB90      410 ;
EB90      411 ;
EB90      412 ; Routine to evaluate FACSIGN and FACMANT for < 0, = 0, or
EB90      413 ; > 0. Also, set C-flag for < 0 and clear C-flag for > 0.
EB90      414 ;
EB90 20 82 EB 415 FSGN      jsr SIGNCHK
EB93      416 ;
EB93 85 9E 417 FLOAT      sta FACMANT
EB95      418 ;
EB95 A9 00 419          lda #ZERO
EB97 85 9F 420          sta FACMANT+1
EB99      421 ;
EB99 A2 88 422          ldx #$88          ; byte exponent, 2^9
EB9B      423 ;

```

```

EB9B A5 9E      424  FLOAT2    lda FACMANT
EB9D 49 FF      425          eor #NEGONE
EB9F           426  ;
EB9F 2A         427          rol                ; initialize C-flag
EBA0           428  ;
EBA0 86 9D      429  FLOAT3    stx FACEXP
EBA2           430  ;
EBA2 A9 00      431          lda #ZERO
EBA4 85 A0      432          sta FACMANT+2
EBA6 85 A1      433          sta FACMANT+3
EBA8 85 AC      434          sta FACGUARD
EBAA           435  ;
EBAA 85 A2      436          sta FACSIGN
EBAC           437  ;
EBAC 4C 1D E8   438          jmp COMPFAC1
EBAF           439  ;
EBAF           440  ;
EBAF           441  ; Calculate ABS.
EBAF           442  ;
EBAF 46 A2      443  FABS      lsr FACSIGN
EBB1           444  ;
EBB1 60         445          rts
EBB2           446  ;
EBB2           447  ;
EBB2           448  ; Routine that compares FAC to packed number at (A/Y).
EBB2           449  ; Return A=1, 0, or -1 if (A/Y) is <, =, > than FAC.
EBB2           450  ;
EBB2 A6 92      451  FPCOMP0    ldx ARGGUARD
EBB4           452  ;
EBB4 2C 00 00   453          bit *-*
EBB7           454          dfs !-2
EBB5           455  ;
EBB5 A6 AC      456  FPCOMP    ldx FACGUARD
EBB7           457  ;
EBB7 85 60      458  FPCOMP2    sta DEST
EBB9 84 61      459          sty DEST+1
EBBB           460  ;
EBBB A0 00      461          ldy #ZERO
EBBD           462  ;
EBBD B1 60      463          lda (DEST),Y
EBBF F0 C1      464          beq SIGNCHK
EBC1           465  ;
EBC1 C5 9D      466          cmp FACEXP
EBC3 D0 26      467          bne >1
EBC5           468  ;
EBC5 C8         469          iny
EBC6           470  ;
EBC6 B1 60      471          lda (DEST),Y
EBC8 45 A2      472          eor FACSIGN
EBCA 30 BA      473          bmi SIGNCHK2
EBCC           474  ;
EBCC B1 60      475          lda (DEST),Y
EBCE 09 80      476          ora #EXPBIAS
EBD0           477  ;
EBD0 C5 9E      478          cmp FACMANT
EBD2 D0 17      479          bne >1
EBD4           480  ;
EBD4 C8         481          iny
EBD5           482  ;
EBD5 B1 60      483          lda (DEST),Y
EBD7 C5 9F      484          cmp FACMANT+1

```

```

EBD9 D0 10      485      bne >1
EBDB           486      ;
EBDB C8         487      iny
EBDC           488      ;
EBDC B1 60      489      lda (DEST),Y
EBDE C5 A0      490      cmp FACMANT+2
EBE0 D0 09      491      bne >1
EBE2           492      ;
EBE2 C8         493      iny
EBE3           494      ;
EBE3 E4 AC      495      cpx FACGUARD
EBE5           496      ;
EBE5 B1 60      497      lda (DEST),Y
EBE7 E5 A1      498      sbc FACMANT+3
EBE9 F0 A4      499      beq RTN.EB.8
EBEB           500      ;
EBEB 6A         501      ^1      ror
EBEC           502      ;
EBEC 45 A2      503      eor FACSIGN
EBEE           504      ;
EBEE 4C 88 EB   505      jmp SIGNCHK3
EBF1           506      ;
EBF1           507      ;
EBF1           508      dfs 1,ZERO      ; 1 byte
EBF2           509      ;
EBF2           510      ;
EBF2           511      ; FP2INT quick integer fuction converts FAC to an integer
EBF2           512      ; value by shifting right with sign extension until all
EBF2           513      ; of the fractional bits are out. This routine assumes
EBF2           514      ; that the exponent is less than 32.
EBF2           515      ;
EBF2 A5 9D      516      FP2INT      lda FACEXP
EBF4 F0 4A      517      beq CLRMANT
EBF6           518      ;
EBF6 38         519      sec
EBF7           520      ;
EBF7 E9 A0      521      sbc #$A0
EBF9           522      ;
EBF9 24 A2      523      bit FACSIGN
EBFB 10 09      524      bpl >1
EBFD           525      ;
EBFD AA         526      tax
EBFE           527      ;
EBFE A9 FF      528      lda #NEGONE
EC00 85 A4      529      sta EXTSIGN
EC02           530      ;
EC02 20 98 E8   531      jsr COMPMANT
EC05           532      ;
EC05 8A         533      txa
EC06           534      ;
EC06 A2 9D      535      ^1      ldx #FACEXP      ; use FAC indexing
EC08           536      ;
EC08 C9 F9      537      cmp #!-7
EC0A 10 06      538      bpl >2
EC0C           539      ;
EC0C 20 E5 E8   540      jsr SHFTBYT2
EC0F 84 A4      541      sty EXTSIGN      ; Y-reg = 0
EC11           542      ;
EC11 60         543      rts
EC12           544      ;
EC12 A8         545      ^2      tay

```

```

EC13          546 ;
EC13 A5 A2    547      lda FACSIGN
EC15 29 80    548      and #MSBSET
EC17          549 ;
EC17 46 9E    550      lsr FACMANT
EC19          551 ;
EC19 05 9E    552      ora FACMANT
EC1B 85 9E    553      sta FACMANT
EC1D          554 ;
EC1D 20 FC E8 555      jsr SHFTBITS
EC20 84 A4    556      sty EXTSIGN          ; Y-reg = 0
EC22          557 ;
EC22 60       558      rts
EC23          559 ;
EC23          560 ;
EC23          561 ; FINT function to convert FAC to integer form and then
EC23          562 ; refloat the integer to floating point. A faster
EC23          563 ; approach would be to simply clear the fractional bits by
EC23          564 ; setting them to zero. Exponent must be less than 32.
EC23          565 ;
EC23 A5 9D    566 FINT      lda FACEXP
EC25 C9 A0    567      cmp #$A0
EC27 B0 20    568      bcs RTN.EC.4
EC29          569 ;
EC29 20 F2 EB 570      jsr FP2INT
EC2C 84 AC    571      sty FACGUARD          ; Y-reg = 0
EC2E          572 ;
EC2E A5 A2    573      lda FACSIGN
EC30 84 A2    574      sty FACSIGN
EC32          575 ;
EC32 49 80    576      eor #EXPBIAS
EC34 2A       577      rol          ; init C-flag for COMPFAC1
EC35          578 ;
EC35 A5 A1    579      lda FACMANT+3          ; set aside integer byte
EC37 85 0D    580      sta BYTVALUE          ; for EXP and POWER
EC39          581 ;
EC39 A9 A0    582      lda #$A0
EC3B 85 9D    583      sta FACEXP
EC3D          584 ;
EC3D 4C 1D E8 585      jmp COMPFAC1          ; normalize the conversion
EC40          586 ;
EC40          587 ;
EC40 85 9E    588 CLRMANT   sta FACMANT
EC42 85 9F    589      sta FACMANT+1
EC44 85 A0    590      sta FACMANT+2
EC46 85 A1    591      sta FACMANT+3
EC48          592 ;
EC48 A8       593      tay
EC49          594 ;
EC49 60       595 RTN.EC.4 rts
EC4A          596 ;
EC4A          597 ;
EC4A          598      icl "EC.L"

```

LLOAD EC.L,A\$4000


```

EC4A          1          ttl "ROM Source Code, EC.L"
EC4A          2          ;
EC4A          3          ;
EC4A          4          ; EC.L
EC4A          5          ;
EC4A          6          ;
EC4A A2 0A     7  GETINT  ldx #10                ; counter
EC4C A0 00     8          ldy #ZERO              ; value to save
EC4E          9          ;
EC4E 94 99    10 ^1      sty TEMP2+1,X          ; clear 0x99 to 0xA3
EC50         11          ;
EC50 CA       12          dex
EC51 10 FB    13          bpl <1
EC53         14          ;
EC53 90 6B    15          bcc GETINT5
EC55         16          ;
EC55 C9 2D    17          cmp #'-'
EC57 D0 04    18          bne >2
EC59         19          ;
EC59 86 A3    20          stx MINUSLOC            ; minus sign location
EC5B         21          ;
EC5B F0 04    22          beq GETINT2            ; always taken
EC5D         23          ;
EC5D C9 2B    24 ^2      cmp #'+'
EC5F D0 05    25          bne >3
EC61         26          ;
EC61 20 B1 00 27  GETINT2 jsr CHRGET
EC64 90 5A    28          bcc GETINT5
EC66         29          ;
EC66 C9 2E    30 ^3      cmp #'.'
EC68 F0 2E    31          beq >6
EC6A         32          ;
EC6A C9 45    33          cmp #'E'              ; exponent part
EC6C D0 30    34          bne >7
EC6E         35          ;
EC6E 20 B1 00 36          jsr CHRGET
EC71 90 61    37          bcc GETINT6            ; modified
EC73         38          ;
EC73 C9 C9    39          cmp #TK.MINUS
EC75 F0 0E    40          beq >4
EC77         41          ;
EC77 C9 2D    42          cmp #'-'
EC79 F0 0A    43          beq >4
EC7B         44          ;
EC7B C9 C8    45          cmp #TK.PLUS
EC7D F0 08    46          beq GETINT3
EC7F         47          ;
EC7F C9 2B    48          cmp #'+'
EC81 F0 04    49          beq GETINT3
EC83         50          ;
EC83 D0 07    51          bne >5                ; always taken
EC85         52          ;
EC85 66 9C    53 ^4      ror EXPSIGN
EC87         54          ;
EC87 20 B1 00 55  GETINT3 jsr CHRGET
EC8A 90 48    56          bcc GETINT6
EC8C         57          ;
EC8C 24 9C    58 ^5      bit EXPSIGN
EC8E 10 0E    59          bpl >7
EC90         60          ;

```

```

EC90 38          61          sec
EC91          62          ;
EC91 A9 00      63          lda #ZERO
EC93 E5 9A      64          sbc EXPCOUNT
EC95          65          ;
EC95 4C A0 EC   66          jmp GETINT4
EC98          67          ;
EC98 66 9B      68 ^6      ror DPFLAG
EC9A          69          ;
EC9A 24 9B      70          bit DPFLAG
EC9C 50 C3      71          bvc GETINT2
EC9E          72          ;
EC9E A5 9A      73 ^7      lda EXPCOUNT
ECA0          74          ;
ECA0 38          75 GETINT4  sec
ECA1          76          ;
ECA1 E5 99      77          sbc TEMP2+1
ECA3 85 9A      78          sta EXPCOUNT
ECA5 F0 12      79          beq >1
ECA7          80          ;
ECA7 10 09      81          bpl >9
ECA9          82          ;
ECA9 20 54 EA   83 ^8      jsr DIVFAC10
ECAC          84          ;
ECAC E6 9A      85          inc EXPCOUNT
ECAE D0 F9      86          bne <8
ECB0          87          ;
ECB0 F0 07      88          beq >1          ; always taken
ECB2          89          ;
ECB2 20 39 EA   90 ^9      jsr MULFAC10
ECB5          91          ;
ECB5 C6 9A      92          dec EXPCOUNT
ECB7 D0 F9      93          bne <9
ECB9          94          ;
ECB9 A5 A3      95 ^1      lda MINUSLOC
ECBB 10 8C      96          bpl RTN.EC.4          ; accelerate code
ECBD          97          ;
ECBD 4C D0 EE   98          jmp NEGFAC
ECC0          99          ;
ECC0          100         ;
ECC0          101         ; Accumulate a digit into FAC.
ECC0          102         ;
ECC0 48          103 GETINT5  pha
ECC1          104         ;
ECC1 24 9B      105         bit DPFLAG
ECC3 10 02      106         bpl >2
ECC5          107         ;
ECC5 E6 99      108         inc TEMP2+1
ECC7          109         ;
ECC7 20 39 EA   110 ^2      jsr MULFAC10
ECCA          111         ;
ECCA 38          112         sec
ECCB          113         ;
ECCB 68          114         pla
ECCC E9 30      115         sbc #'0'
ECCE          116         ;
ECCE 20 F6 EC   117         jsr ADD2FAC
ECD1          118         ;
ECD1 4C 61 EC   119         jmp GETINT2
ECD4          120         ;
ECD4          121         ;

```

```

ECD4      122  ; Accumulate digit of exponent.  Moved from 0xECE7.
ECD4      123  ;
ECD4 A5 9A   124 GETINT6  lda EXPCOUNT
ECD6 C9 0A   125          cmp #10
ECD8 90 09   126          bcc >3
ECDA      127  ;
ECDA A9 64   128          lda #100
ECDC      129  ;
ECDC 24 9C   130          bit EXPSIGN
ECDE 30 11   131          bmi >4
ECE0      132  ;
ECE0 4C C9 E8 133          jmp OF.ERR
ECE3      134  ;
ECE3 0A      135 ^3      asl
ECE4 0A      136          asl
ECE5      137  ;
ECE5 18      138          clc
ECE6      139  ;
ECE6 65 9A   140          adc EXPCOUNT
ECE8 0A      141          asl
ECE9      142  ;
ECE9 18      143          clc
ECEA      144  ;
ECEA A0 00   145          ldy #ZERO
ECEC      146  ;
ECEC 71 B8   147          adc (TXTPTR),Y
ECEE      148  ;
ECEE 38      149          sec
ECF0      150  ;
ECF0 E9 30   151          sbc #'0'
ECF1      152  ;
ECF1 85 9A   153 ^4      sta EXPCOUNT
ECF3      154  ;
ECF3 4C 87 EC 155          jmp GETINT3
ECF6      156  ;
ECF6      157  ;
ECF6      158 ; Add A-reg to FAC.  Moved from 0xECD4.
ECF6      159  ;
ECF6 48      160 ADD2FAC pha
ECF7      161  ;
ECF7 20 63 EB 162          jsr COPYF2A
ECFA      163  ;
ECFA 68      164          pla
ECFB      165  ;
ECFB 20 93 EB 166          jsr FLOAT
ECFE      167  ;
ECFE A5 AA   168          lda ARGSIGN
ED00 45 A2   169          eor FACSIGN
ED02 85 AB   170          sta XORSIGN
ED04      171  ;
ED04 A6 9D   172          ldx FACEXP
ED06      173  ;
ED06 4C C1 E7 174          jmp ADD2
ED09      175  ;
ED09      176  ;
ED09      177 ; Fall into LINEPRT.
ED09      178  ;
ED09 60      179 ^0      rts
ED0A      180  ;
ED0A C8      181 PRTMSG19 iny
ED0B F0 FC   182          beq <0

```

```

ED0D          183 ;
ED0D A9 25    184      lda #MSG19          ; ' in '
ED0F A0 ED    185      ldy /MSG19
ED11 20 3B DB 186      jsr STROUT
ED14          187 ;
ED14 A6 75    188      ldx CURLIN
ED16 A5 76    189      lda CURLIN+1
ED18          190 ;
ED18          191 ;
ED18 85 9E    192 LINEPRT sta FACMANT      ; put upper half here
ED1A 86 9F    193      stx FACMANT+1      ; put lower half here
ED1C          194 ;
ED1C 38       195      sec          ; normal FAC
ED1D          196 ;
ED1D A2 90    197      ldx #$90          ; integer exponent, 2^16
ED1F 20 A0 EB 198      jsr FLOAT3
ED22          199 ;
ED22 4C 38 DB 200      jmp LINEOUT
ED25          201 ;
ED25          202 ;
ED25 20 69 6E 203 MSG19 asc ' in '
ED28 20
ED29 00       204      byt ZERO
ED2A          205 ;
ED2A          206 ;
ED2A          207 ; Floating point values used in FPOUT. Moved from 0xED09.
ED2A          208 ;
ED2A 9B 3E BC 209 FP9.9E7 hex 9B3EBC1FFD      ; FP 999999999.9
ED2D 1F FD
ED2F 9E 6E 6B 210 FP9.9E8 hex 9E6E6B27FC      ; FP 999999999.0
ED32 27 FC
ED34          211 ;
ED34          212 ;
ED34          213 ; Entry point for the PRINT and the STR$ statements. This
ED34          214 ; routine has been rewritten in order to add enhancements.
ED34          215 ; All numerical strings are printed from start of STACK.
ED34          216 ; All floating point decimal numbers are prefaced with a
ED34          217 ; zero to the left of the decimal point.
ED34          218 ;
ED34          219 ; Convert FAC to a numerical string at the start of STACK.
ED34          220 ;
ED34 A0 00    221 FPOUT ldy #ZERO
ED36 84 AD    222      sty SAVY
ED38          223 ;
ED38 A5 A2    224      lda FACSIGN
ED3A 10 07    225      bpl >1
ED3C          226 ;
ED3C 84 A2    227      sty FACSIGN      ; make FACSIGN positive
ED3E          228 ;
ED3E A9 2D    229      lda #'-'
ED40 20 54 EE 230      jsr SAV2STK
ED43          231 ;
ED43 A5 9D    232 ^1      lda FACEXP      ; is FAC zero
ED45 D0 05    233      bne >2
ED47          234 ;
ED47 A9 30    235      lda #'0'          ; yes
ED49          236 ;
ED49 4C 49 EE 237      jmp FPOUT2      ; exit FPOUT
ED4C          238 ;
ED4C C9 81    239 ^2      cmp #$81
ED4E B0 0A    240      bcs >3          ; FAC > 0x80

```

```

ED50          241 ;
ED50          242 ;
ED50          243 ; FAC < 0x81.  Multiply FAC by 1.0E+09.  Set counter to -9.
ED50          244 ;
ED50 A9 DC    245         lda #FP1.0E9
ED52 A0 EA    246         ldy /FP1.0E9
ED54 20 7F E9 247         jsr FMULT
ED57          248 ;
ED57 A9 F7    249         lda #!-9
ED59          250 ;
ED59 2C 00 00 251         bit *-*
ED5C          252         dfs !-2
ED5A          253 ;
ED5A          254 ;
ED5A          255 ; FAC > 0x80.  No pre-multiplying.  Set counter to 0.
ED5A          256 ;
ED5A A9 00    257 ^3      lda #ZERO
ED5C          258 ;
ED5C 85 99    259         sta COUNTER          ; multiply/divide counter
ED5E          260 ;
ED5E          261 ;
ED5E          262 ; Multiply FAC by 10 or divide FAC by 10 until FAC is in
ED5E          263 ; the range of 1.0E+08-1 to 1.0E+09-1.
ED5E          264 ;
ED5E A9 2F    265 ^4      lda #FP9.9E8          ; FP 999,999,999.0
ED60 A0 ED    266         ldy /FP9.9E8
ED62          267 ;
ED62 20 B5 EB 268         jsr FPCOMP
ED65 F0 20    269         beq >8          ; done, bypass roundup
ED67          270 ;
ED67 10 10    271         bpl >6          ; FAC > FP9.9E8
ED69          272 ;
ED69 A9 2A    273 ^5      lda #FP9.9E7          ; FP 99,999,999.9
ED6B A0 ED    274         ldy /FP9.9E7
ED6D          275 ;
ED6D 20 B5 EB 276         jsr FPCOMP
ED70 90 0E    277         bcc >7          ; FAC > FP9.9E7
ED72          278 ;
ED72 20 39 EA 279         jsr MULFAC10
ED75          280 ;
ED75 C6 99    281         dec COUNTER
ED77 D0 F0    282         bne <5          ; always taken
ED79          283 ;
ED79 20 54 EA 284 ^6      jsr DIVFAC10
ED7C          285 ;
ED7C E6 99    286         inc COUNTER
ED7E D0 DE    287         bne <4          ; always taken
ED80          288 ;
ED80 A9 61    289 ^7      lda #FP0.5          ; a trivial roundup
ED82 A0 F0    290         ldy /FP0.5
ED84 20 BE E7 291         jsr FADD
ED87          292 ;
ED87          293 ;
ED87          294 ; Normalize the mantissa so that its exponent is 0xA0.
ED87          295 ; Initialize X-reg to display at least one whole digit.
ED87          296 ; Up to nine decimal numbers are displayed, then 'E' is
ED87          297 ; displayed.  Calculate exponent value from COUNTER.
ED87          298 ;
ED87 20 F2 EB 299 ^8      jsr FP2INT          ; C-flag clear on exit
ED8A          300 ;
ED8A A2 01    301         ldx #1

```

```

ED8C      302 ;
ED8C A5 99  303      lda COUNTER
ED8E 69 0A  304      adc #10
ED90 30 08  305      bmi >1                ; if decimal number
ED92      306 ;
ED92 C9 0B  307      cmp #11
ED94 B0 05  308      bcs >2                ; if > 10 whole digits
ED96      309 ;
ED96      310 ;
ED96      311 ; Number of whole digits < 10, so decrement and display up
ED96      312 ; to 9 whole digits by forcing EXPCOUNT to be zero.
ED96      313 ;
ED96 AA     314      tax
ED97 CA     315      dex
ED98      316 ;
ED98 A9 02  317      lda #2
ED9A      318 ;
ED9A 38     319 ^1      sec
ED9B      320 ;
ED9B E9 02  321 ^2      sbc #2
ED9D      322 ;
ED9D 86 99  323      stx COUNTER          ; whole digit counter
ED9F 85 9A  324      sta EXPCOUNT        ; exponent value
EDA1      325 ;
EDA1      326 ;
EDA1      327 ; If the whole digit counter is =< zero, write '0.'. The
EDA1      328 ; original code wrote ' '. X-reg is 0xFF, 0x00, or > 0x00.
EDA1      329 ; The values of 0xFF and 0x00 are only of interest here.
EDA1      330 ;
EDA1 CA     331      dex
EDA2 10 12  332      bpl >3
EDA4      333 ;
EDA4 A9 30  334      lda #'0'
EDA6 20 54 EE 335      jsr SAV2STK
EDA9      336 ;
EDA9 A9 2E  337      lda #'.'
EDAB 20 54 EE 338      jsr SAV2STK
EDAE      339 ;
EDAE      340 ;
EDAE      341 ; If COUNTER is still negative after an increment, write
EDAE      342 ; the tenths digit as zero.
EDAE      343 ;
EDAE E8     344      inx
EDAF F0 05  345      beq >3
EDB1      346 ;
EDB1 A9 30  347      lda #'0'
EDB3 20 54 EE 348      jsr SAV2STK
EDB6      349 ;
EDB6      350 ;
EDB6      351 ; Digit counting loop. When X-reg is minus, negative
EDB6      352 ; values are added to FAC and X-reg is saved as a digit.
EDB6      353 ; When X-reg is plus, positive values are added to FAC and
EDB6      354 ; 10 - X-reg is saved as a digit.
EDB6      355 ;
EDB6 A9 00  356 ^3      lda #ZERO                ; initial FPDECTBL index
EDB8 A2 80  357      ldx #MSBSET            ; initial digit counter
EDBA      358 ;
EDBA 48     359 ^4      pha                ; save FPDECTBL index
EDBB      360 ;
EDBB A8     361      tay
EDBC      362 ;

```

```

EDBC 18          363 ^5      clc
EDBD             364 ;
EDBD A5 A1       365      lda FACMANT+3
EDBF 79 5F EE    366      adc FPDECTBL+3,Y
EDC2 85 A1       367      sta FACMANT+3
EDC4             368 ;
EDC4 A5 A0       369      lda FACMANT+2
EDC6 79 5E EE    370      adc FPDECTBL+2,Y
EDC9 85 A0       371      sta FACMANT+2
EDCB             372 ;
EDCB A5 9F       373      lda FACMANT+1
EDCD 79 5D EE    374      adc FPDECTBL+1,Y
EDD0 85 9F       375      sta FACMANT+1
EDD2             376 ;
EDD2 A5 9E       377      lda FACMANT
EDD4 79 5C EE    378      adc FPDECTBL,Y
EDD7 85 9E       379      sta FACMANT
EDD9             380 ;
EDD9 E8          381      inx                ; increment digit counter
EDDA             382 ;
EDDA B0 04       383      bcs >6                ; when adding - values or
EDDC             384 ;                        done adding + values
EDDC             385 ;
EDDC 10 DE       386      bpl <5                ; when adding + values
EDDE 30 02       387      bmi >7                ; done adding - values
EDE0             388 ;
EDE0 30 DA       389 ^6      bmi <5                ; not done adding - values
EDE2             390 ;
EDE2 8A          391 ^7      txa                ; recall digit counter
EDE3             392 ;
EDE3 90 04       393      bcc >8                ; when adding - values
EDE5             394 ;
EDE5 49 FF       395      eor #NEGONE            ; make negative
EDE7 69 0A       396      adc #10              ; subtract from 10
EDE9             397 ;
EDE9 69 2F       398 ^8      adc #'0'-1          ; make into ASCII digit
EDEB 29 7F       399      and #MSBCLR          ; save digit
EDED 20 54 EE    400      jsr SAV2STK
EDF0             401 ;
EDF0 C6 99       402      dec COUNTER            ; whole digit counter
EDF2 D0 05       403      bne >9
EDF4             404 ;
EDF4             405 ;
EDF4             406 ; Done saving whole digits. Write decimal point. Process
EDF4             407 ; remaining table entries.
EDF4             408 ;
EDF4 A9 2E       409      lda #'. '
EDF6 20 54 EE    410      jsr SAV2STK
EDF9             411 ;
EDF9 8A          412 ^9      txa                ; recall digit counter
EDFA 29 80       413      and #MSBSET          ; extract the MSB
EDFC 49 80       414      eor #MSBSET          ; toggle the MSB
EDFE AA          415      tax
EDFF             416 ;
EDFF 68          417      pla                ; recall table index
EE00 69 04       418      adc #4              ; point to next entry
EE02             419 ;
EE02 C9 24       420      cmp #DECTBLLEN
EE04 D0 B4       421      bne <4
EE06             422 ;
EE06             423 ;

```

```

EE06      424 ; Remove unnecessary zeros. Leave an ASCII period/zero if
EE06      425 ; only have single digit and 'E'.
EE06      426 ;
EE06 A2 07      427          ldx #7          ; ASCII zero counter
EE08 A4 AD      428          ldy SAVY          ; recall insertion location
EE0A      429 ;
EE0A 88      430 ^1          dey
EE0B      431 ;
EE0B B9 00 01   432          lda STACK,Y
EE0E      433 ;
EE0E CA      434          dex
EE0F D0 04      435          bne >2
EE11      436 ;
EE11 A6 9A      437          ldx EXPCOUNT
EE13 D0 09      438          bne >3
EE15      439 ;
EE15 C9 30      440 ^2          cmp #'0'
EE17 F0 F1      441          beq <1          ; found ASCII zero
EE19      442 ;
EE19 C9 2E      443          cmp #'.'
EE1B F0 01      444          beq >3          ; found ASCII period
EE1D      445 ;
EE1D C8      446          iny
EE1E      447 ;
EE1E 84 AD      448 ^3          sty SAVY          ; insertion location
EE20      449 ;
EE20      450 ;
EE20      451 ; If EXPCOUNT is zero, process is done. Otherwise, save
EE20      452 ; the exponent value after 'E' and the EXPCOUNT sign.
EE20      453 ;
EE20 A6 9A      454          ldx EXPCOUNT
EE22 F0 28      455          beq >6
EE24      456 ;
EE24 A9 45      457          lda #'E'
EE26 20 54 EE    458          jsr SAV2STK
EE29      459 ;
EE29 8A      460          txa          ; recall EXPCOUNT
EE2A 10 07      461          bpl >4
EE2C      462 ;
EE2C 49 FF      463          eor #NEGONE      ; make 2's compliment
EE2E AA      464          tax
EE2F      465 ;
EE2F E8      466          inx
EE30      467 ;
EE30 A9 2D      468          lda #'-'          ; get ASCII minus
EE32      469 ;
EE32 2C 00 00    470          bit *-*
EE35      471          dfs !-2
EE33      472 ;
EE33 A9 2B      473 ^4          lda #'+'          ; get ASCII plus
EE35      474 ;
EE35 20 54 EE    475          jsr SAV2STK
EE38      476 ;
EE38      477 ;
EE38      478 ; Convert exponent value into base 10.
EE38      479 ;
EE38 8A      480          txa          ; recall exponent value
EE39      481 ;
EE39 A2 2F      482          ldx #'0'-1      ; init 10's ASCII digit
EE3B      483 ;
EE3B 38      484          sec

```



```

EE3C          485 ;
EE3C E8       486 ^5      inx
EE3D          487 ;
EE3D E9 0A    488          sbc #10          ; successive subtraction
EE3F B0 FB    489          bcs <5          ; still more to subtract
EE41          490 ;
EE41 69 3A    491          adc #'0'+10      ; make ASCII 1's digit
EE43 48       492          pha              ; save
EE44          493 ;
EE44 8A       494          txa              ; write 10's digit
EE45 20 54 EE 495          jsr SAV2STK
EE48          496 ;
EE48 68       497          pla              ; write 1's digit
EE49          498 ;
EE49 20 54 EE 499 FPOUT2   jsr SAV2STK
EE4C          500 ;
EE4C A9 00    501 ^6      lda #ZERO          ; terminate ASCII data
EE4E 20 54 EE 502          jsr SAV2STK
EE51          503 ;
EE51          504 ;
EE51          505 ; Return to caller with address of STACK in (A/Y).
EE51          506 ;
EE51          507 ;          lda #STACK          ; already zero
EE51 A0 01    508          ldy /STACK
EE53          509 ;
EE53 60       510          rts
EE54          511 ;
EE54          512 ;
EE54 A4 AD    513 SAV2STK   ldy SAVY
EE56          514 ;
EE56 99 00 01 515          sta STACK,Y
EE59          516 ;
EE59 E6 AD    517          inc SAVY
EE5B          518 ;
EE5B 60       519          rts
EE5C          520 ;
EE5C          521 ;
EE5C FA 0A 1F 522 FPDECTBL hex FA0A1F00      ; -100,000,000
EE5F 00
EE60 00 98 96 523          hex 00989680      ; 10,000,000
EE63 80
EE64 FF F0 BD 524          hex FFF0BDC0      ; - 1,000,000
EE67 C0
EE68 00 01 86 525          hex 000186A0      ; 100,000
EE6B A0
EE6C FF FF D8 526          hex FFFFD8F0      ; - 10,000
EE6F F0
EE70 00 00 03 527          hex 000003E8      ; 1,000
EE73 E8
EE74 FF FF FF 528          hex FFFFFFF9C      ; - 100
EE77 9C
EE78 00 00 00 529          hex 0000000A      ; 10
EE7B 0A
EE7C FF FF FF 530          hex FFFFFFFF      ; - 1
EE7F FF
EE80          531 ;
0024          532 DECTBLN equ *-FPDECTBL
EE80          533 ;
EE80          534 ;
EE80          535 ; FSQR routine entry point; take square root of FAC.
EE80          536 ; Use the Newton-Raphson iteration method which is

```

```

EE80          537 ; R = [(N/X) + X] / 2 until R = X, else X = R.
EE80          538 ; Calculate the exponent of the starting initial value.
EE80          539 ;
EE80 20 21 EB 540 ^1      jsr COPYF2T1
EE83          541 ;
EE83 18       542      clc
EE84          543 ;
EE84 49 80    544      eor #EXPBIAS
EE86 30 01    545      bmi >2
EE88          546 ;
EE88 38       547      sec
EE89          548 ;
EE89 6A       549 ^2      ror
EE8A          550 ;
EE8A 4C 66 F6 551      jmp SQR2
EE8D          552 ;
EE8D          553 ;
EE8D 20 82 EB 554 FSQR    jsr SIGNCHK
EE90 F0 48    555      beq RTN.EE.D
EE92          556 ;
EE92 10 EC    557      bpl <1
EE94          558 ;
EE94 4C 9B E1 559      jmp IQ.ERR
EE97          560 ;
EE97          561 ;
EE97          562 ; This is the ^ operator entry point for Applesoft. POWER
EE97          563 ; takes the EXP of LN( ARG ) * FAC leaving the result in
EE97          564 ; FAC, or ARG ^ FAC = EXP( LN( ARG ) * FAC ). Rewritten to
EE97          565 ; utilize TEMP3 guard byte in MULT2.
EE97          566 ;
EE97 F0 70    567 OPOWER   beq FEXP
EE99          568 ;
EE99 A5 A5    569      lda ARGEXP
EE9B F0 2C    570      beq >2
EE9D          571 ;
EE9D 20 BA F1 572      jsr COPYF2T3          ; Y-reg = 0
EEA0          573 ;
EEA0 A5 AA    574      lda ARGSIGN
EEA2 10 0F    575      bpl >1
EEA4          576 ;
EEA4 20 23 EC 577      jsr FINT          ; negative, must be INT value
EEA7          578 ;
EEA7 A9 8A    579      lda #TEMP3
EEA9 A0 00    580      ldy /TEMP3
EEAB          581 ;
EEAB 20 4D DD 582      jsr FPCOMPT3
EEAE D0 03    583      bne >1
EEB0          584 ;
EEB0 98       585      tya          ; Y-reg=4 (FPCOMP) -> FACSIGN
EEB1          586 ;
EEB1 A4 0D    587      ldy BYTVALUE      ; Y-reg = 4 from FPCOMP or =
EEB3          588 ;          signed integer byte value
EEB3 20 55 EB 589 ^1      jsr COPYA2F2
EEB6          590 ;
EEB6 98       591      tya          ; 0, 4 or signed integer value
EEB7 48       592      pha
EEB8          593 ;
EEB8 20 41 E9 594      jsr FLN
EEBB          595 ;
EEBB 20 A8 F6 596      jsr COPYT32A
EEBE 20 82 E9 597      jsr MULT2

```

```

EEC1      598 ;
EEC1 20 09 EF 599      jsr FEXP
EEC4      600 ;
EEC4 68      601      pla
EEC5      602 ;
EEC5 4A      603      lsr                ; test oddness for neg. value
EEC6 B0 08    604      bcs NEGFAC
EEC8      605 ;
EEC8 60      606      rts
EEC9      607 ;
EEC9 4C 44 E8 608      ^2      jmp ZEROFAC2
EECC      609 ;
EECC      610 ;
EECC      611      dfs 1,ZERO                ; 1 byte
EECD      612 ;
EECD      613 ;
EECD 4C 2D EA 614 FEXP2      jmp ZEROFERR                ; handle 0 or overflow error
EED0      615 ;
EED0      616 ;
EED0      617 ; This routine negates the value in FAC and this routine is
EED0      618 ; also the ">" operator entry point.
EED0      619 ;
EED0      620 OGT:
EED0 A5 9D    621 NEGFAC      lda FACEXP
EED2 F0 06    622              beq RTN.EE.D
EED4      623 ;
EED4 A5 A2    624              lda FACSIGN
EED6 49 FF    625              eor #NEGONE
EED8 85 A2    626              sta FACSIGN
EEDA      627 ;
EEDA 60      628 RTN.EE.D rts
EEDB      629 ;
EEDB      630 ;
EEDB 81 38 AA 631 FPINVLN2 hex 8138AA3B28                ; FP 1/ln(2) = 1.44269504
EEDE 3B 28    632 ;
EEE0      633 POLY.EXP hex 07                ; number of coefficients-1
EEE1 71 34 58 634              hex 7134583E56                ; FP (LOG(2)^7)/8!
EEE4 3E 56    635              hex 74167EB31B                ; FP (LOG(2)^6)/7!
EEE6 74 16 7E 636              hex 772FEEEE385                ; FP (LOG(2)^5)/6!
EEE9 B3 1B    637              hex 7A1D841C2A                ; FP (LOG(2)^4)/5!
EEEB 77 2F EE 638              hex 7C6359580A                ; FP (LOG(2)^3)/4!
EEEE E3 85    639              hex 7E75FDE7C6                ; FP (LOG(2)^2)/3!
EEF0 7A 1D 84 640              hex 8031721810                ; FP LOG(2)/2!
EEF3 1C 2A    641 FP1.0      hex 8100000000                ; FP 1.0
EEF5 7C 63 59 642 ;
EEF8 58 0A    643 ;
EEFA 7E 75 FD 644 ; FEXP routine entry point; take e to the power of FAC
EEFD E7 C6    645 ; leaving the result in FAC, or e ^ FAC -> FAC. It is not
EEFF 80 31 72 646 ; necessary to swap registers and negate their subtraction.
EF02 18 10    647 ; Modified zero and overflow handling.
EF04 81 00 00 648 ;
EF07 00 00    649 FEXP      lda #FPINVLN2
EF09      642 ;
EF09      643 ;
EF09      644 ; FEXP routine entry point; take e to the power of FAC
EF09      645 ; leaving the result in FAC, or e ^ FAC -> FAC. It is not
EF09      646 ; necessary to swap registers and negate their subtraction.
EF09      647 ; Modified zero and overflow handling.
EF09      648 ;
EF09 A9 DB    649 FEXP      lda #FPINVLN2

```

```

EF0B A0 EE      650      ldy /FPINVLN2
EF0D 20 7F E9   651      jsr FMULT
EF10            652      ;
EF10 20 70 EB   653      jsr RNDUP          ; use standard round up test
EF13            654      ;
EF13 20 63 EB   655      jsr COPYF2A        ; returns X-reg = 0, FACEXP
EF16 86 AC      656      stx FACGUARD      ; to force INCMANT in FINT
EF18            657      ;
EF18 C9 88      658      cmp #$88          ; exponent value less than 90
EF1A B0 B1      659      bcs FEXP2
EF1C            660      ;
EF1C 20 23 EC   661      jsr FINT
EF1F            662      ;
EF1F 18         663      clc
EF20            664      ;
EF20 A5 0D      665      lda BYTVALUE      ; signed integer byte value
EF22 69 81      666      adc #$81         ; if exponent value > 90
EF24 F0 A7      667      beq FEXP2
EF26            668      ;
EF26 38         669      sec
EF27            670      ;
EF27 E9 01      671      sbc #1
EF29 48         672      pha
EF2A            673      ;
EF2A            674      ;
EF2A            675      ; Removed register swap and call to NEGFAC after SUB2.
EF2A            676      ;
EF2A 20 AA E7   677      jsr SUB2
EF2D            678      ;
EF2D A9 E0      679      lda #POLY.EXP
EF2F A0 EE      680      ldy /POLY.EXP
EF31            681      ;
EF31 85 AD      682      sta COEFPTR
EF33 84 AE      683      sty COEFPTR+1
EF35            684      ;
EF35 20 71 EF   685      jsr POLYNOM
EF38 84 AB      686      sty XORSIGN        ; reg-Y = 0 from POLYNOM
EF3A            687      ;
EF3A 68         688      pla
EF3B            689      ;
EF3B 4C 12 EA   690      jmp PROCEXP2      ; accelerate code
EF3E            691      ;
EF3E            692      ;
EF3E            693      ; Convert natural log LN to base-10 log LOG.
EF3E            694      ;
EF3E 20 41 E9   695      FLOG      jsr FLN
EF41            696      ;
EF41 A9 13      697      lda #FPLOGE
EF43 A0 E9      698      ldy /FPLOGE
EF45            699      ;
EF45 4C 7F E9   700      jmp FMULT
EF48            701      ;
EF48            702      ;
EF48            703      ; Load FAC and FACGUARD with the value of PI.
EF48            704      ;
EF48 A9 A1      705      FPI      lda #FPPI
EF4A A0 E7      706      ldy /FPPI
EF4C 20 F9 EA   707      jsr LOADFAC
EF4F            708      ;
EF4F 84 A2      709      sty FACSIGN        ; Y-reg = 0
EF51            710      ;

```

```

EF51 AD A6 E7 711          lda FPIGUARD          ; get byte for FACGUARD
EF54 85 AC      712          sta FACGUARD
EF56           713          ;
EF56 60         714          rts
EF57           715          ;
EF57           716          ;
EF57           717          ; Odd polynomial processing. First byte is the number of
EF57           718          ; FP coefficients minus one that must be processed.
EF57           719          ;
EF57           720          ; F(X) = X * P( X^2 ) where X = FAC and (A/Y) point to a
EF57           721          ; coefficient table with highest power first. P( X^2 )
EF57           722          ; uses the normal polynomial function.
EF57           723          ;
EF57 A9 66      724 POLYSIN  lda #POLY.SIN          ; standard SINE polynomials
EF59 A0 F0      725          ldy /POLY.SIN
EF5B           726          ;
EF5B 85 AD      727 POLYPROC sta COEFPTR          ; save pointer to coefficients
EF5D 84 AE      728          sty COEFPTR+1
EF5F           729          ;
EF5F 20 21 EB   730          jsr COPYF2T1          ; save (X) term
EF62           731          ;
EF62 A9 93      732          lda #TEMP1          ; point to (X) term
EF64           733          ;          ldy /TEMP1          ; Y-reg = 0 from COPYF2T1
EF64 20 7F E9   734          jsr FMULT          ; create (X*X) term
EF67           735          ;
EF67 20 71 EF   736          jsr POLYNOM          ; add in all coefficients
EF6A           737          ;
EF6A A9 93      738          lda #TEMP1          ; point to (X) term
EF6C A0 00      739          ldy /TEMP1
EF6E           740          ;
EF6E 4C 7F E9   741          jmp FMULT          ; final multiply by (X)
EF71           742          ;
EF71           743          ;
EF71           744          ; Normal polynomial processing. Slightly modified start
EF71           745          ; and end of routine.
EF71           746          ;
EF71           747          ; P(X) = C(0) * X^N + C(1) * X^(N-1) + ... + C(N) where
EF71           748          ; X = FAC and (A/Y) point to a coefficient table with
EF71           749          ; highest power first.
EF71           750          ;
EF71 20 1E EB   751 POLYNOM  jsr COPYF2T2          ; save (X) or (X*X) term
EF74 86 45      752          stx T2GUARD          ; save guard byte
EF76           753          ;
EF76 B1 AD      754          lda (COEFPTR),Y      ; Y-reg = 0 from COPYF2T2
EF78 85 A3      755          sta COEFNUM          ; save coefficients-1 number
EF7A           756          ;
EF7A 20 C5 F1   757          jsr INCCOEF          ; increment coef pointer
EF7D           758          ;
EF7D A2 00      759          ldx #ZERO          ; initial value for ARGGUARD
EF7F           760          ;
EF7F A5 AD      761          lda COEFPTR          ; point to first coefficient
EF81 A4 AE      762          ldy COEFPTR+1
EF83           763          ;
EF83 20 E3 E9   764 ^1      jsr LOADARG          ; load the (X) or (X*X) term
EF86 86 92      765          stx ARGGUARD          ; load the guard byte
EF88           766          ;
EF88 20 82 E9   767          jsr MULT2          ; multiply the term by FAC
EF8B           768          ;
EF8B 18         769          clc
EF8C           770          ;
EF8C A5 AD      771          lda COEFPTR          ; point to next coefficient

```

```

EF8E 69 05      772      adc #FACLEN
EF90            773      ;
EF90 85 AD      774      sta COEFPTR
EF92 90 02      775      bcc >2
EF94            776      ;
EF94 E6 AE      777      inc COEFPTR+1
EF96            778      ;
EF96 A4 AE      779      ^2      ldy COEFPTR+1
EF98            780      ;
EF98 20 BE E7    781      jsr FADD          ; add in next coefficient
EF9B            782      ;
EF9B A6 45      783      ldx T2GUARD      ; retrieve guard byte
EF9D            784      ;
EF9D A9 98      785      lda #TEMP2      ; point to the (X) term
EF9F A0 00      786      ldy /TEMP2
EFA1            787      ;
EFA1 C6 A3      788      dec COEFNUM      ; decrement counter
EFA3 D0 DE      789      bne <1
EFA5            790      ;
EFA5 60          791      RTN.EF.A rts
EFA6            792      ;
EFA6            793      ;
EFA6            794      ; RANDVAL1 hex 9835447A      ; integer value = 2553627770
EFA6            795      ; RANDVAL2 hex 6828B146      ; integer value = 1747497286
EFA6            796      ;
EFA6            797      ; Donald Knuth specified the values to use for this
EFA6            798      ; equation:  $X(n+1) = (X(n) * A + C) \bmod M$ ,  $M = 2^{32}$ .
EFA6            799      ;
EFA6            800      ; Any negative integer value is saved as the new IRAND.
EFA6            801      ; A zero value returns the current integer value of IRAND.
EFA6            802      ; A value of one returns a random number fraction.
EFA6            803      ; Any integer value greater than 1 is for the Range value.
EFA6            804      ;
EFA6 12 B9 B0    805      RANDVAL1 hex 12B9B0A5      ; 314159269 integer value A
EFA9 A5          806
EFAA 36 19 62    806      RANDVAL2 hex 361962EA      ; 907633386 integer value C
EFAD EA          807
EFAE            807      ;
EFAE 20 82 EB    808      FRND      jsr SIGNCHK
EFB1 F0 1A      809      beq >3          ; display IRAND
EFB3            810      ;
EFB3 10 2A      811      bpl >6          ; generate a new IRAND
EFB5            812      ;
EFB5            813      ;
EFB5            814      ; Prepare a new IRAND value. The IRAND is any integer
EFB5            815      ; value from 0 to less than  $2^{32}$ .
EFB5            816      ;
EFB5 46 A2      817      lsr FACSIGN      ; make positive
EFB7            818      ;
EFB7 A9 A0      819      lda #$A0      ; limit FACEXP to  $< 2^{32}$ 
EFB9 C5 9D      820      cmp FACEXP
EFBB B0 02      821      bcs >1
EFBD            822      ;
EFBD 85 9D      823      sta FACEXP
EFBF            824      ;
EFBF 20 F2 EB    825      ^1      jsr FP2INT      ; convert FAC to an integer
EFC2            826      ;
EFC2            827      ;
EFC2            828      ; Copy the integer that resides in FAC back to IRAND.
EFC2            829      ;
EFC2 A2 04      830      ldx #4

```

```

EFC4      831 ;
EFC4 B5 9D 832 ^2      lda FACMANT-1,X
EFC6 95 C8 833      sta IRAND-1,X
EFC8      834 ;
EFC8 CA    835      dex
EFC9 D0 F9 836      bne <2
EFCB      837 ;
EFCB F0 09 838      beq >5          ; always taken
EFCD      839 ;
EFCD      840 ;
EFCD      841 ; Display the current IRAND value as an integer.
EFCD      842 ;
EFCD A2 04 843 ^3      ldx #4
EFCF      844 ;
EFCF B5 C8 845 ^4      lda IRAND-1,X
EFD1 95 9D 846      sta FACMANT-1,X
EFD3      847 ;
EFD3 CA    848      dex
EFD4 D0 F9 849      bne <4
EFD6      850 ;
EFD6 86 AC 851 ^5      stx FACGUARD
EFD8      852 ;
EFD8 A9 A0 853      lda #$A0
EFDA 85 9D 854      sta FACEXP
EFDC      855 ;
EFDC 4C 22 E8 856      jmp NORMFAC1
EFDF      857 ;
EFDF      858 ;
EFDF      859 ; Save the value currently in FAC as the desired Range.
EFDF      860 ; Copy the RANDVAL1 multiplier to ARGMANT, the RANDVAL2
EFDF      861 ; constant to FACMANT, and IRAND to MULMANT.
EFDF      862 ;
EFDF 20 21 EB 863 ^6      jsr COPYF2T1
EFE2      864 ;
EFE2 A2 03 865      ldx #3          ; multiply values copy counter
EFE4 A0 20 866      ldy #32        ; multiply iteration counter
EFE6      867 ;
EFE6 4C 2D F7 868      jmp RND2
EFE9      869 ;
EFE9      870 ;
EFE9      871      dfs 1,ZERO          ; 1 byte
EFEA      872 ;
EFEA      873 ;
EFEA      874 ; FCOS routine entry point; compute COSINE of FAC -> FAC.
EFEA      875 ; Not an identity, COS(X) = SIN(X + PI/2) in this routine.
EFEA      876 ;
EFEA A9 5C 877 FCOS      lda #FPIDIV2          ; FP PI/2
EFEC A0 F0 878      ldy /FPIDIV2
EFEE 20 BE E7 879      jsr FADD
EFF1      880 ;
EFF1      881 ;
EFF1      882 ; FSIN routine entry point; compute SINE of FAC -> FAC.
EFF1      883 ; Modified in order to provide a better entry for TAN.
EFF1      884 ;
EFF1 20 63 EB 885 FSIN      jsr COPYF2A
EFF4      886 ;
EFF4 A6 AA 887      ldx ARGSIGN          ; same as FACSIGN
EFF6 86 AB 888      stx XORSIGN
EFF8      889 ;
EFF8 A9 99 890      lda #FPIMUL2          ; FP PI*2
EFFA A0 F0 891      ldy /FPIMUL2

```

```
EFFC 20 F9 EA    892      jsr LOADFAC
EFFF           893      ;
EFFF 20 69 EA    894      jsr DIV2
F002           895      ;
F002           896      ;

BSAVE E0ROM,D1,A$1000,B,L$1002

F002           897      usr E0ROM,D1
F002           898      ;
F002           899      ;
F002           900      icl "F0.L,D2"

LLOAD F0.L,D2,A$4000
```



```

F002      1      ttl "ROM Source Code, F0.L"
F002      2      ;
F002      3      ;
F002      4      ; F0.L
F002      5      ;
F002      6      ;
F002      7      obj PAGE10
F002      8      usr
F002      9      ;
F002     10      ;
F002 20 63 EB    11      jsr COPYF2A
F005 20 23 EC    12      jsr FINT
F008 20 AA E7    13      jsr SUB2
F00B     14      ;
F00B A9 D7      15      lda #FP.25
F00D A0 EA      16      ldy /FP.25
F00F 20 A7 E7    17      jsr FSUB
F012     18      ;
F012 A5 A2      19      lda FACSIGN
F014 85 CD      20      sta SIGNFLG
F016 10 0D      21      bpl SIN2          ; range already 0 to 1/4
F018     22      ;
F018 A9 61      23      lda #FP0.5          ; shift range to 1/4 to 1/2
F01A A0 F0      24      ldy /FP0.5
F01C 20 BE E7    25      jsr FADD
F01F     26      ;
F01F A5 A2      27      lda FACSIGN
F021 30 05      28      bmi >5
F023     29      ;
F023 C6 16      30      dec TOGLFLG          ; make it negative
F025     31      ;
F025 20 D0 EE    32      SIN2      jsr NEGFAC          ; TAN entry to get COS value
F028     33      ;
F028 A9 D7      34      ^5      lda #FP.25          ; shift range to -1/4 to 1/4
F02A A0 EA      35      ldy /FP.25
F02C 20 BE E7    36      jsr FADD
F02F     37      ;
F02F A5 CD      38      lda SIGNFLG
F031 10 03      39      bpl >6
F033     40      ;
F033 20 D0 EE    41      jsr NEGFAC          ; make range 0 to 1/4
F036     42      ;
F036 4C 57 EF    43      ^6      jmp POLYSIN          ; process SINE polynomials
F039     44      ;
F039     45      ;
F039     46      dfs 1,ZERO          ; 1 byte
F03A     47      ;
F03A     48      ;
F03A     49      ; FTAN routine entry point; compute TANGENT of FAC leaving
F03A     50      ; the result in FAC. Modified routine to use SIN directly.
F03A     51      ;
F03A A9 00      52      FTAN      lda #ZERO
F03C 85 16      53      sta TOGLFLG
F03E     54      ;
F03E 20 F1 EF    55      jsr FSIN
F041 20 BA F1    56      jsr COPYF2T3
F044     57      ;
F044 A9 93      58      lda #TEMP1          ; recall POLYPROC SIN value
F046 A0 00      59      ldy /TEMP1
F048 20 F9 EA    60      jsr LOADFAC

```

```

F04B      61 ;
F04B A9 00      62      lda #ZERO
F04D 85 A2      63      sta FACSIGN
F04F      64 ;
F04F A5 16      65      lda TOGLFLG
F051 85 CD      66      sta SIGNFLG
F053      67 ;
F053 20 25 F0    68      jsr SIN2
F056 20 A8 F6    69      jsr COPYT32A
F059      70 ;
F059 4C 69 EA    71      jmp DIV2
F05C      72 ;
F05C      73 ;
F05C      74 ; Floating point values. Only need one labeled FPIMUL2.
F05C      75 ; Moved FP.25 to 0xEA4D. Moved FP0.5 from 0xED27.
F05C      76 ;
F05C 81 49 0F    77 FPIDIV2 hex 81490FDAA2      ; FP PI/2 = 1.57079633
F05F DA A2
F061 80 00 00    78 FP0.5 hex 8000000000      ; FP 0.5
F064 00 00
F066      79 ;
F066      80 ;
F066      81 ; SIN function polynomials. The absolute SIN coefficients
F066      82 ; are given using the unreferenced space above and below.
F066      83 ; Calculated these coefficients using FLT64 routines in C.
F066      84 ; The guard byte is shown in the comments if it is used.
F066      85 ;
F066 0A          86 POLY.SIN hex 0A          ; number of coefficients-1
F067 77 14 3B    87      hex 77143B8107      ; 0x06AA, FP (2PI)^21/21!
F06A 81 07
F06C 7A C5 20    88      hex 7AC5202109      ; 0x08FD, FP -(2PI)^19/19!
F06F 21 09
F071 7D 55 76    89      hex 7D55761958      ; 0x57CA, FP (2PI)^17/17!
F074 19 58
F076 80 B7 D6    90      hex 80B7D6DCF9      ; 0xF8AB, FP -(2PI)^15/15!
F079 DC F9
F07B 82 74 7A    91      hex 82747A1A68      ; 0x680C, FP (2PI)^13/13!
F07E 1A 68
F080 84 F1 83    92      hex 84F183A7EF      ; 0xEF44, FP -(2PI)^11/11!
F083 A7 EF
F085 86 28 3C    93      hex 86283C1A44      ; 0x43F7, FP (2PI)^9/9!
F088 1A 44
F08A 87 99 69    94      hex 8799696673      ; 0x7316, FP -(2PI)^7/7!
F08D 66 73
F08F 87 23 35    95      hex 872335E33C      ; 0x3BAD, FP (2PI)^5/5!
F092 E3 3C
F094 86 A5 5D    96      hex 86A55DE731      ; 0x312E, FP -(2PI)^3/3!
F097 E7 31
F099 83 49 0F    97 FPIMUL2 hex 83490FDAA2      ; 0xA221, FP (2PI)
F09C DA A2
F09E      98 ;
F09E      99 ;
F09E     100 ; FATAN routine entry point; compute ARCTANGENT of FAC
F09E     101 ; leaving the result in FAC.
F09E     102 ;
F09E A5 A2     103 FATAN      lda FACSIGN
F0A0 48        104      pha
F0A1        105 ;
F0A1 10 03     106      bpl >1
F0A3        107 ;
F0A3 20 D0 EE   108      jsr NEGFAC

```

```

F0A6      109 ;
F0A6 A5 9D      110 ^1      lda FACEXP      ; check if > 1
F0A8 C9 81      111      cmp #$81      ; first value at FP1.0
F0AA 08      112      php
F0AB      113 ;
F0AB 90 07      114      bcc >2
F0AD      115 ;
F0AD A9 04      116      lda #FP1.0
F0AF A0 EF      117      ldy /FP1.0
F0B1 20 66 EA    118      jsr FDIV
F0B4      119 ;
F0B4 A9 CC      120 ^2      lda #POLY.ATN
F0B6 A0 F0      121      ldy /POLY.ATN
F0B8 20 5B EF    122      jsr POLYPROC
F0BB      123 ;
F0BB 28      124      plp
F0BC 90 07      125      bcc >3
F0BE      126 ;
F0BE A9 5C      127      lda #FPIDIV2
F0C0 A0 F0      128      ldy /FPIDIV2
F0C2 20 A7 E7    129      jsr FSUB
F0C5      130 ;
F0C5 68      131 ^3      pla
F0C6 30 01      132      bmi >4
F0C8      133 ;
F0C8 60      134      rts
F0C9      135 ;
F0C9 4C D0 EE    136 ^4      jmp NEGFAC
F0CC      137 ;
F0CC      138 ;
F0CC 0B      139 POLY.ATN hex 0B      ; number of coefficients-1
F0CD 76 B3 83    140      hex 76B383BDD3      ; FP -6.84793912E-04
F0D0 BD D3
F0D2 79 1E F4    141      hex 791EF4A6F5      ; FP 4.85094216E-03
F0D5 A6 F5
F0D7 7B 83 FC    142      hex 7B83FCB010      ; FP -0.0161117018
F0DA B0 10
F0DC 7C 0C 1F    143      hex 7C0C1F67CA      ; FP 0.034209638
F0DF 67 CA
F0E1 7C DE 53    144      hex 7CDE53CBC1      ; FP -0.0542791328
F0E4 CB C1
F0E6 7D 14 64    145      hex 7D1464704C      ; FP 0.0724571965
F0E9 70 4C
F0EB 7D B7 EA    146      hex 7DB7EA517A      ; FP -0.0898023954
F0EE 51 7A
F0F0 7D 63 30    147      hex 7D6330887E      ; FP 0.110932413
F0F3 88 7E
F0F5 7E 92 44    148      hex 7E9244993A      ; FP -0.142839808
F0F8 99 3A
F0FA 7E 4C CC    149      hex 7E4CCC91C7      ; FP 0.19999912
F0FD 91 C7
F0FF 7F AA AA    150      hex 7FAAAAAA13      ; FP -0.333333316
F102 AA 13
F104 81 00 00    151      hex 8100000000      ; FP 1.000000000
F107 00 00
F109      152 ;
F109      153 ;
F109      154 ; CHRGET and CHRGOT routines and FPRAND variable.
F109      155 ; C-flag is set for 00:2F and 3A:FF, clear for 30:39.
F109      156 ; Only need four bytes for FPRAND, so 0xCD is used by TAN.
F109      157 ;

```

```

F109      158 PGZCODE:
F109      159      phs CHRGOTADR
00B1      160      ;
00B1 E6 B8 161 CHRGET      inc TXTPTR
00B3 D0 02 162      bne CHRGOT
00B5      163      ;
00B5 E6 B9 164      inc TXTPTR+1
00B7      165      ;
00B7 AD 00 00 166 CHRGOT      lda *-*          ; TXTPTR
00BA C9 3A 167      cmp #' ':'
00BC B0 0A 168      bcs >1
00BE      169      ;
00BE C9 20 170      cmp #SPACE&MSBCLR
00C0 F0 EF 171      beq CHRGET
00C2      172      ;
00C2 38 173      sec
00C3      174      ;
00C3 E9 30 175      sbc #'0'
00C5      176      ;
00C5 38 177      sec
00C6      178      ;
00C6 E9 D0 179      sbc #0-'0'
00C8      180      ;
00C8 60 181      ^1      rts
00C9      182      ;
00C9      183      ;
00C9      184      ; Need four bytes for the random number generator seed.
00C9      185      ;
00C9 4F C7 52 186 FPRAND      hex 4FC75258          ; 1338462808 integer value
00CC 58
00CD      187      ;
001C      188 ZPCDLEN      equ *-CHRGET
00CD      189      ;
00CD      190      phs PGZCODE+ZPCDLEN
F125      191      ;
F125      192      ;
F125      193      ; Many of the COLDSTRT initializations are ridiculous and
F125      194      ; they are of no benefit. They have been removed.
F125      195      ;
F125 A2 FF 196 COLDSTRT      ldx #NEGONE          ; set Direct Mode flag
F127 86 76 197      stx CURLIN+1
F129      198      ;
F129 A2 FB 199      ldx #$FB          ; leave room for line buffer
F12B 9A 200      txs
F12C      201      ;
F12C      202      ;
F12C      203      ; Two vector initializations have been removed here.
F12C      204      ;
F12C 20 73 F2 205      jsr BNORMAL
F12F      206      ;
F12F A9 00 207      lda #*-*          ; get jmp instruction
F131      208      dfs !-1
F130 4C 00 00 209      jmp *-*          ; to form jmp vectors
F133      210      dfs !-2
F131      211      ;
F131 85 00 212      sta GOWARM          ; USER vector
F133 85 03 213      sta GOSTROUT        ; USER vector
F135 85 0A 214      sta GOUSR
F137 85 90 215      sta JMPADRS
F139      216      ;
F139 A9 9B 217      lda #IQ.ERR

```

```

F13B A0 E1      218      ldy /IQ.ERR
F13D            219      ;
F13D 85 0B      220      sta GOUSR+1
F13F 84 0C      221      sty GOUSR+2
F141            222      ;
F141            223      ;
F141            224      ; Copy CHRGET and FPRAND to page-zero.  The original loop
F141            225      ; value is now correct.
F141            226      ;
F141 A2 1C      227      ldx #ZPCDLEN          ; correct value
F143            228      ;
F143 BD 08 F1    229      ^1      lda PGZCODE-1,X
F146 95 B0      230      sta CHRGTADR-1,X
F148            231      ;
F148 CA         232      dex
F149 D0 F8      233      bne <1
F14B            234      ;
F14B 86 54      235      stx LASTPT+1
F14D 86 A4      236      stx EXTSIGN
F14F 86 F1      237      stx SPEEDBYT
F151 86 F2      238      stx TRACEFLG
F153            239      ;
F153 E8         240      inx          ; make it one
F154            241      ;
F154 86 E7      242      stx HRSCALE          ; init HIRES scale factor
F156            243      ;
F156 8E FD 01    244      stx INPUT-3          ; fake forward link
F159 8E FC 01    245      stx INPUT-4
F15C            246      ;
F15C A2 55      247      ldx #TEMPST
F15E 86 52      248      stx TEMPPT
F160            249      ;
F160 20 50 DB    250      jsr PRTCR
F163            251      ;
F163            252      ;
F163            253      ; Determine end of RAM.  This routine has been modified.
F163            254      ;
F163 A0 00      255      ldy #PAGE08
F165 A9 08      256      lda /PAGE08
F167            257      ;
F167 84 50      258      sty LINNUM
F169 85 51      259      sta LINNUM+1
F16B            260      ;
F16B E6 51      261      ^2      inc LINNUM+1
F16D            262      ;
F16D B1 50      263      lda (LINNUM),Y
F16F 49 FF      264      eor #NEGONE
F171 91 50      265      sta (LINNUM),Y
F173            266      ;
F173 D1 50      267      cmp (LINNUM),Y
F175 D0 08      268      bne >3
F177            269      ;
F177 49 FF      270      eor #NEGONE
F179 91 50      271      sta (LINNUM),Y
F17B            272      ;
F17B D1 50      273      cmp (LINNUM),Y
F17D F0 EC      274      beq <2
F17F            275      ;
F17F A5 51      276      ^3      lda LINNUM+1
F181 29 F0      277      and #$F0
F183            278      ;

```

```

F183 84 73      279      sty MEMSIZE
F185 85 74      280      sta MEMSIZE+1
F187           281      ;
F187 84 6F      282      sty FRETOP
F189 85 70      283      sta FRETOP+1
F18B           284      ;
F18B           285      ;
F18B           286      ; Initialize Applesoft program start location to 0x0801.
F18B           287      ; This routine has been modified. Any address may replace
F18B           288      ; PAGE08 as the start address for Applesoft. RUNFLAG is
F18B           289      ; cleared in SCRTCH.
F18B           290      ;
F18B A2 00      291      ldx #ZERO
F18D 8E 00 08   292      stx PAGE08
F190           293      ;
F190 A9 01      294      lda #PAGE08+1
F192 A0 08      295      ldy /PAGE08+1
F194           296      ;
F194 85 67      297      sta PRGTAB
F196 84 68      298      sty PRGTAB+1
F198           299      ;
F198 20 E3 D3   300      jsr CKSTRSIZ
F19B 20 4B D6   301      jsr SCRTCH
F19E           302      ;
F19E           303      ;
F19E           304      ; Create two USER vectors.
F19E           305      ;
F19E A9 3B      306      lda #STROUT
F1A0 A0 DB      307      ldy /STROUT
F1A2           308      ;
F1A2 85 04      309      sta GOSTROUT+1
F1A4 84 05      310      sty GOSTROUT+2
F1A6           311      ;
F1A6 A9 3C      312      lda #RESTART
F1A8 A0 D4      313      ldy /RESTART
F1AA           314      ;
F1AA 85 01      315      sta GOWARM+1
F1AC 84 02      316      sty GOWARM+2
F1AE           317      ;
F1AE 4C 3C D4   318      jmp RESTART          ; accelerate code
F1B1           319      ;
F1B1           320      ;
F1B1           321      ; Continuation of FRMSTAK3.
F1B1           322      ;
F1B1 A5 9E      323      FRMSTAK4 lda FACMANT
F1B3 48         324      pha
F1B4           325      ;
F1B4 A5 9D      326      lda FACEXP
F1B6 48         327      pha
F1B7           328      ;
F1B7 6C 5E 00   329      jmp (INDEX)
F1BA           330      ;
F1BA           331      ;
F1BA           332      ; COPYF2T3 routine entry point. Copy FACGUARD to T3GUARD
F1BA           333      ; and FAC -> T3.
F1BA           334      ;
F1BA A5 AC      335      COPYF2T3 lda FACGUARD
F1BC 85 8F      336      sta T3GUARD
F1BE           337      ;
F1BE A2 8A      338      ldx #TEMP3
F1C0 A0 00      339      ldy /TEMP3

```

```

F1C2      340 ;
F1C2 4C 2E EB 341      jmp COPYFAC2
F1C5      342 ;
F1C5      343 ;
F1C5      344 ; Increment the coefficient pointer.
F1C5      345 ;
F1C5 E6 AD 346 INCCOEF inc COEFPTR
F1C7 D0 02 347      bne >1
F1C9      348 ;
F1C9 E6 AE 349      inc COEFPTR+1
F1CB      350 ;
F1CB 60 351 ^1      rts
F1CC      352 ;
F1CC      353 ;
F1CC      354 ; Clear MULMANT for FMULT in order to provide room for
F1CC      355 ; using guard bytes in this function.
F1CC      356 ;
F1CC 84 62 357 CLEARMUL sty MULMANT
F1CE 84 63 358      sty MULMANT+1
F1D0 84 64 359      sty MULMANT+2
F1D2 84 65 360      sty MULMANT+3
F1D4      361 ;
F1D4 60 362      rts
F1D5      363 ;
F1D5      364 ;
F1D5 20 64 DD 365 BCALL jsr FRMNUM
F1D8 20 52 E7 366      jsr GETADDR
F1DB      367 ;
F1DB 6C 50 00 368      jmp (LINNUM)
F1DE      369 ;
F1DE      370 ;
F1DE 20 F8 E6 371 BIN jsr GETBYT
F1E1      372 ;
F1E1 8A 373      txa
F1E2      374 ;
F1E2 4C 8B FE 375      jmp INPORT
F1E5      376 ;
F1E5      377 ;
F1E5 20 F8 E6 378 BPR jsr GETBYT
F1E8      379 ;
F1E8 8A 380      txa
F1E9      381 ;
F1E9 4C 95 FE 382      jmp OUTPORT
F1EC      383 ;
F1EC      384 ;
F1EC      385 ; LORES comma separated coordinates for H2 and V2.
F1EC      386 ;
F1EC 20 F8 E6 387 PLOTFNS jsr GETBYT
F1EF      388 ;
F1EF E0 30 389      cpx #48 ; must be less than 48
F1F1 B0 13 390      bcs IQ.ERR4
F1F3      391 ;
F1F3 86 F0 392      stx FIRST
F1F5      393 ;
F1F5 A9 2C 394      lda #', ' ; why not use CHKCOM?
F1F7 20 C0 DE 395      jsr SYNTAXCHK
F1FA      396 ;
F1FA 20 F8 E6 397      jsr GETBYT
F1FD      398 ;
F1FD E0 30 399      cpx #48 ; must be less than 48
F1FF B0 05 400      bcs IQ.ERR4

```

```

F201          401 ;
F201 86 2C    402          stx H2
F203 86 2D    403          stx V2
F205          404 ;
F205 60       405          rts
F206          406 ;
F206          407 ;
F206 4C 9B E1 408 IQ.ERR4  jmp IQ.ERR          ; Illegal Quantity error
F209          409 ;
F209          410 ;
F209          411 ; Gets A,B at C values for HLIN and VLIN.
F209          412 ;
F209 20 EC F1 413 LINCOOR  jsr PLOTFNS
F20C          414 ;
F20C E4 F0    415          cpx FIRST
F20E B0 08    416          bcs >1
F210          417 ;
F210 A5 F0    418          lda FIRST
F212 85 2C    419          sta H2
F214 85 2D    420          sta V2
F216          421 ;
F216 86 F0    422          stx FIRST
F218          423 ;
F218 A9 C5    424 ^1      lda #TK.AT
F21A 20 C0 DE 425          jsr SYNTAXCHK
F21D          426 ;
F21D 20 F8 E6 427          jsr GETBYT
F220          428 ;
F220 E0 30    429          cpx #48          ; must be less than 48
F222 B0 E2    430          bcs IQ.ERR4
F224          431 ;
F224 60       432          rts
F225          433 ;
F225          434 ;
F225 20 EC F1 435 BPLOT    jsr PLOTFNS
F228          436 ;
F228 8A       437          txa
F229          438 ;
F229 A4 F0    439          ldy FIRST
F22B C0 28    440          cpy #40          ; must be less than 40
F22D B0 D7    441          bcs IQ.ERR4
F22F          442 ;
F22F 4C 00 F8 443          jmp PLOT
F232          444 ;
F232          445 ;
F232 20 09 F2 446 BHLIN    jsr LINCOOR
F235          447 ;
F235 8A       448          txa
F236          449 ;
F236 A4 2C    450          ldy H2
F238 C0 28    451          cpy #40          ; must be less than 40
F23A B0 CA    452          bcs IQ.ERR4
F23C          453 ;
F23C A4 F0    454          ldy FIRST
F23E          455 ;
F23E 4C 19 F8 456          jmp HLINE
F241          457 ;
F241          458 ;
F241 20 09 F2 459 BVLIN    jsr LINCOOR
F244          460 ;
F244 8A       461          txa

```



```

F245 A8          462          tay
F246            463          ;
F246 C0 28      464          cpy #40          ; must be less than 40
F248 B0 BC      465          bcs IQ.ERR4
F24A            466          ;
F24A A5 F0      467          lda FIRST
F24C            468          ;
F24C 4C 28 F8   469          jmp VLINE
F24F            470          ;
F24F            471          ;
F24F 20 F8 E6   472 BCOLOR   jsr GETBYT
F252            473          ;
F252 8A         474          txa
F253            475          ;
F253 4C 64 F8   476          jmp SETCOL
F256            477          ;
F256            478          ;
F256 20 F8 E6   479 BVTAB    jsr GETBYT
F259            480          ;
F259 CA         481          dex
F25A            482          ;
F25A 8A         483          txa
F25B C9 18      484          cmp #24          ; must only be 0 to 23
F25D B0 A7      485          bcs IQ.ERR4
F25F            486          ;
F25F 4C 5B FB   487          jmp TABV
F262            488          ;
F262            489          ;
F262            490          ; SPEEDBYT = SPEED ^ 0xFF. Modified range to include zero.
F262            491          ; See 0xDB6A. If SPEEDBYT = 0, then WAIT is bypassed.
F262            492          ;
F262 20 F8 E6   493 BSPEED   jsr GETBYT
F265            494          ;
F265 8A         495          txa
F266 49 FF      496          eor #NEGONE
F268 85 F1      497          sta SPEEDBYT
F26A            498          ;
F26A 60         499          rts
F26B            500          ;
F26B            501          ;
F26B            502          dfs 2,ZERO          ; 2 bytes
F26D            503          ;
F26D            504          ;
F26D 38         505 BTRACE   sec
F26E            506          ;
F26E 90 00      507          bcc *+2
F270            508          dfs !-1
F26F            509          ;
F26F            510          ;
F26F 18         511 BNOTRACE clc
F270            512          ;
F270 66 F2      513          ror TRACEFLG
F272            514          ;
F272 60         515          rts
F273            516          ;
F273            517          ;
F273 A9 FF      518 BNORMAL  lda #NEGONE
F275 D0 02      519          bne INVERSE2          ; always taken
F277            520          ;
F277            521          ;
F277 A9 3F      522 BINVERSE lda #INVERSE

```

```

F279          523 ;
F279 A2 00     524 INVERSE2 ldx #ZERO          ; flash OFF
F27B          525 ;
F27B 85 32     526 INVERSE3 sta INVFLG
F27D 86 F3     527          stx FLASHBYT
F27F          528 ;
F27F 60        529          rts
F280          530 ;
F280          531 ;
F280 A9 7F     532 BFLASH   lda #FLASH
F282          533 ;
F282 A2 40     534          ldx #$40          ; flash ON
F284 D0 F5     535          bne INVERSE3      ; always taken
F286          536 ;
F286          537 ;
F286          538 ; Verify requested HIMEM is above all variables and arrays.
F286          539 ; Modified this routine slightly.
F286          540 ;
F286 20 64 DD   541 BHIMEM   jsr FRMNUM
F289 20 52 E7   542          jsr GETADDR
F28C          543 ;
F28C A5 50     544          lda LINNUM
F28E C5 6D     545          cmp STREND
F290          546 ;
F290 A5 51     547          lda LINNUM+1
F292 E5 6E     548          sbc STREND+1
F294 90 0D     549          bcc OM.ERR4
F296          550 ;
F296          551 ;
F296          552 ; Save new HIMEM value. String variables are NOT cleared
F296          553 ; and this could produce a huge problem.
F296          554 ;
F296 A5 50     555          lda LINNUM
F298 85 73     556          sta MEMSIZE
F29A 85 6F     557          sta FRETOP
F29C          558 ;
F29C A5 51     559          lda LINNUM+1
F29E 85 74     560          sta MEMSIZE+1
F2A0 85 70     561          sta FRETOP+1
F2A2          562 ;
F2A2 60        563          rts
F2A3          564 ;
F2A3          565 ;
F2A3 4C 10 D4   566 OM.ERR4  jmp OM.ERR          ; Out of Memory error
F2A6          567 ;
F2A6          568 ;
F2A6          569 ; Verify requested LOMEM is below HIMEM and above program.
F2A6          570 ;
F2A6 20 64 DD   571 BLOMEM   jsr FRMNUM
F2A9 20 52 E7   572          jsr GETADDR
F2AC          573 ;
F2AC A5 50     574          lda LINNUM
F2AE C5 73     575          cmp MEMSIZE
F2B0          576 ;
F2B0 A5 51     577          lda LINNUM+1
F2B2 E5 74     578          sbc MEMSIZE+1
F2B4 B0 ED     579          bcs OM.ERR4
F2B6          580 ;
F2B6 A5 50     581          lda LINNUM
F2B8 C5 69     582          cmp VARTAB
F2BA          583 ;

```

```

F2BA A5 51      584      lda LINNUM+1
F2BC E5 6A      585      sbc VARTAB+1
F2BE 90 E3      586      bcc OM.ERR4
F2C0           587      ;
F2C0           588      ;
F2C0           589      ; Save new LOMEM value and clear all variables and arrays.
F2C0           590      ;
F2C0 A5 50      591      lda LINNUM
F2C2 85 69      592      sta VARTAB
F2C4           593      ;
F2C4 A5 51      594      lda LINNUM+1
F2C6 85 6A      595      sta VARTAB+1
F2C8           596      ;
F2C8 4C 6C D6   597      jmp CLEARC
F2CB           598      ;
F2CB           599      ;
F2CB A9 AB      600      BONERR  lda #$80+BSGOTO/2      ; must be GOTO statement
F2CD 20 C0 DE   601      jsr SYNTAXCHK
F2D0           602      ;
F2D0 A5 B8      603      lda TXTPTR
F2D2 85 F4      604      sta TXTPTRSV
F2D4           605      ;
F2D4 A5 B9      606      lda TXTPTR+1
F2D6 85 F5      607      sta TXTPTRSV+1
F2D8           608      ;
F2D8 38         609      sec
F2D9 66 D8      610      ror ERRFLG      ; set MSB of ERRFLG
F2DB           611      ;
F2DB A5 75      612      lda CURLIN
F2DD 85 F6      613      sta CURLINSV
F2DF           614      ;
F2DF A5 76      615      lda CURLIN+1
F2E1 85 F7      616      sta CURLINSV+1
F2E3           617      ;
F2E3 20 A6 D9   618      jsr DATSCAN2      ; why ignore rest of line???
F2E6           619      ;
F2E6 4C 98 D9   620      jmp DATA2      ; continue processing program
F2E9           621      ;
F2E9           622      ;
F2E9           623      ; This routine handles errors if the ONERR GOTO is active.
F2E9           624      ;
F2E9 86 DE      625      HANDLERR stx ERRNUM
F2EB           626      ;
F2EB A6 F8      627      ldx REMSTK
F2ED 86 DF      628      stx ERRSTK
F2EF           629      ;
F2EF A5 75      630      lda CURLIN      ; save offending line number
F2F1 85 DA      631      sta ERRLIN
F2F3           632      ;
F2F3 A5 76      633      lda CURLIN+1
F2F5 85 DB      634      sta ERRLIN+1
F2F7           635      ;
F2F7 A5 79      636      lda TEXTPTR      ; save offending line position
F2F9 85 DC      637      sta ERRPOS
F2FB           638      ;
F2FB A5 7A      639      lda TEXTPTR+1
F2FD 85 DD      640      sta ERRPOS+1
F2FF           641      ;
F2FF A5 F4      642      lda TXTPTRSV      ; get target line number
F301 85 B8      643      sta TXTPTR
F303           644      ;

```

```

F303 A5 F5      645      lda TXTPTRSV+1
F305 85 B9      646      sta TXTPTR+1
F307            647      ;
F307 A5 F6      648      lda CURLINSV          ; get ON ERR line number
F309 85 75      649      sta CURLIN
F30B            650      ;
F30B A5 F7      651      lda CURLINSV+1
F30D 85 76      652      sta CURLIN+1
F30F            653      ;
F30F 20 B7 00    654      jsr CHRGOT          ; start conversion
F312 20 3E D9    655      jsr BGOTO          ; go to specified ON ERR line
F315            656      ;
F315 4C D2 D7    657      jmp NEWSTT
F318            658      ;
F318            659      ;
F318            660      ; Entry for ON ERR correction, Applesoft manual, CALL -3288
F318            661      ; which is 0xF328 below at the ldx ERRSTK.
F318            662      ;
F318            663      ; The Applesoft manual on page 82 suggests the following
F318            664      ; code as part of an error-handling routine. This code
F318            665      ; will execute at any address. The net effect of these
F318            666      ; instructions is the same with the current lines of code.
F318            667      ;
F318            668      ;      pla          ; 0x68, 104
F318            669      ;      tay          ; 0xA8, 168
F318            670      ;
F318            671      ;      pla          ; 0x68, 104
F318            672      ;
F318            673      ;      ldx $DF      ; 0xA6, 0xDF; 166, 223
F318            674      ;      txs          ; 0x9A, 154
F318            675      ;
F318            676      ;      pha          ; 0x48, 72
F318            677      ;
F318            678      ;      tya          ; 0x98, 152
F318            679      ;      pha          ; 0x48 72
F318            680      ;
F318            681      ;      rts          ; 0x60, 96
F318            682      ;
F318            683      ;
F318            684      ; Restore line number and text pointer and retry offending
F318            685      ; line.
F318            686      ;
F318 A5 DA      687      BRESUME  lda ERRLIN
F31A 85 75      688      sta CURLIN
F31C            689      ;
F31C A5 DB      690      lda ERRLIN+1
F31E 85 76      691      sta CURLIN+1
F320            692      ;
F320 A5 DC      693      lda ERRPOS
F322 85 B8      694      sta TXTPTR
F324            695      ;
F324 A5 DD      696      lda ERRPOS+1
F326 85 B9      697      sta TXTPTR+1
F328            698      ;
F328 A6 DF      699      ldx ERRSTK          ; entry for ON ERR correction
F32A 9A          700      txs
F32B            701      ;
F32B 4C D2 D7    702      jmp NEWSTT
F32E            703      ;
F32E            704      ;
F32E 4C C9 DE    705      SY.ERR6  jmp SY.ERR          ; Syntax error

```

```

F331          706 ;
F331          707 ;
F331 B0 FB    708 BDEL      bcs SY.ERR6          ; must specify a line number
F333          709 ;
F333 A6 AF    710          ldx PRGEND
F335 86 69    711          stx VARTAB
F337          712 ;
F337 A6 B0    713          ldx PRGEND+1
F339 86 6A    714          stx VARTAB+1
F33B          715 ;
F33B 20 0C DA 716          jsr LINGET
F33E 20 1A D6 717          jsr FNDLIN
F341          718 ;
F341 A5 9B    719          lda LOWTR
F343 85 60    720          sta DEST
F345          721 ;
F345 A5 9C    722          lda LOWTR+1
F347 85 61    723          sta DEST+1
F349          724 ;
F349 A9 2C    725          lda #', '
F34B 20 C0 DE 726          jsr SYNTAXCHK
F34E          727 ;
F34E 20 0C DA 728          jsr LINGET          ; get end range
F351          729 ;
F351 E6 50    730          inc LINNUM
F353 D0 02    731          bne >2
F355          732 ;
F355 E6 51    733          inc LINNUM+1
F357          734 ;
F357 20 1A D6 735 ^2          jsr FNDLIN
F35A          736 ;
F35A A5 9B    737          lda LOWTR
F35C C5 60    738          cmp DEST
F35E          739 ;
F35E A5 9C    740          lda LOWTR+1
F360 E5 61    741          sbc DEST+1
F362 90 2B    742          bcc >8
F364          743 ;
F364 A0 00    744          ldy #ZERO
F366          745 ;
F366 B1 9B    746 ^4          lda (LOWTR),Y
F368 91 60    747          sta (DEST),Y
F36A          748 ;
F36A E6 9B    749          inc LOWTR
F36C D0 02    750          bne >5
F36E          751 ;
F36E E6 9C    752          inc LOWTR+1
F370          753 ;
F370 E6 60    754 ^5          inc DEST
F372 D0 02    755          bne >6
F374          756 ;
F374 E6 61    757          inc DEST+1
F376          758 ;
F376 A5 69    759 ^6          lda VARTAB
F378 C5 9B    760          cmp LOWTR
F37A          761 ;
F37A A5 6A    762          lda VARTAB+1
F37C E5 9C    763          sbc LOWTR+1
F37E B0 E6    764          bcs <4
F380          765 ;
F380 A6 61    766          ldx DEST+1

```

```

F382          767 ;
F382 A4 60    768     ldy DEST
F384 D0 01    769     bne >7
F386          770 ;
F386 CA      771     dex
F387          772 ;
F387 88      773 ^7    dey
F388          774 ;
F388 86 6A    775     stx VARTAB+1
F38A 84 69    776     sty VARTAB
F38C          777 ;
F38C 4C F2 D4 778     jmp ASENTER
F38F          779 ;
F38F 60      780 ^8    rts
F390          781 ;
F390          782 ;
F390 AD 56 C0 783 BGR   lda HIRESOFF
F393          784     lda MIXEDON           ; done in SETGR
F393          785 ;
F393 4C 40 FB 786     jmp SETGR
F396          787 ;
F396          788 ;
F396          789     dfs 3,ZERO           ; 3 bytes
F399          790 ;
F399          791 ;
F399          792 ; Modified routine with substantially better logic.
F399          793 ;
F399 4C 33 FB 794 BTEXT  jmp INIT2
F39C          795 ;
F39C          796 ;
F39C          797 ; Removed STORE and RECALL statements since they depend on
F39C          798 ; reading and writing data to and from the cassette ports
F39C          799 ; that are no longer useful to the Apple //e user.
F39C          800 ;
F39C          801 ;
F39C          802 ; Read audio waveform for HEADER, SYNC, and binary DATA to
F39C          803 ; the address in A1 until the address in A2. Wait until
F39C          804 ; three seconds of HEADER has past before looking for SYNC.
F39C          805 ; This routine was originally found at 0xC5D1 and modified.
F39C          806 ;
F39C 20 FF D8 807 CXREAD jsr RD2BIT           ; wait for waveform to change
F39F          808 ;
F39F A9 FF    809     lda #NEGONE           ; initialize CHECKSUM
F3A1 85 2E    810     sta CHKSUM
F3A3          811 ;
F3A3 A2 12    812     ldx #18
F3A5          813 ;
F3A5 20 A8 FC 814 ^1    jsr WAIT           ; waste 167309 cycles
F3A8          815 ;
F3A8 CA      816     dex
F3A9 D0 FA    817     bne <1
F3AB          818 ;
F3AB 20 FF D8 819     jsr RD2BIT           ; wait for waveform to change
F3AE          820 ;
F3AE A0 24    821 ^2    ldy #$24           ; waveform change, 432 cycles
F3B0          822 ;
F3B0 20 02 D9 823     jsr RDBIT
F3B3 B0 F9    824     bcs <2
F3B5          825 ;
F3B5 20 02 D9 826     jsr RDBIT
F3B8          827 ;

```

```

F3B8 A0 3B      828      ldy #$3B          ; waveform change, 708 cycles
F3BA           829      ;
F3BA 20 BB D8   830      ^3      jsr RDBYTE      ; read DATA, MSB first
F3BD           831      ;
F3BD 81 3C      832      sta (A1L,X)          ; save data, X-reg = 0
F3BF           833      ;
F3BF 45 2E      834      eor CHKSUM
F3C1 85 2E      835      sta CHKSUM          ; update CHECKSUM
F3C3           836      ;
F3C3 20 BA FC   837      jsr NEXTA1          ; increment A1, compare to A2
F3C6           838      ;
F3C6 A0 35      839      ldy #$35          ; waveform change, 636 cycles
F3C8           840      ;
F3C8 90 F0      841      bcc <3              ; C-flag from NEXTA1
F3CA           842      ;
F3CA 20 BB D8   843      jsr RDBYTE          ; read CHECKSUM data byte
F3CD           844      ;
F3CD C5 2E      845      cmp CHKSUM          ; does it compare?
F3CF           846      ;
F3CF 60         847      rts                  ; return to CALLER
F3D0           848      ;
F3D0           849      ;
F3D0           850      ; Modified the HGR and HGR2 routines.
F3D0           851      ;
F3D0 2C 52 C0   852      ^1      bit MIXEDOFF
F3D3           853      ;
F3D3 60         854      rts
F3D4           855      ;
F3D4           856      ;
F3D4 2C 53 C0   857      ^2      bit MIXEDON
F3D7           858      ;
F3D7 60         859      rts
F3D8           860      ;
F3D8           861      ;
F3D8 A9 40      862      BHGR2      lda /PAGE40
F3DA 20 EC F3   863      jsr CLRHIREs
F3DD           864      ;
F3DD 2C 55 C0   865      bit PAGE2ON
F3E0           866      ;
F3E0 90 EE      867      bcc <1              ; always taken
F3E2           868      ;
F3E2           869      ;
F3E2 A9 20      870      BHGR      lda /PAGE20
F3E4 20 EC F3   871      jsr CLRHIREs
F3E7           872      ;
F3E7 2C 54 C0   873      bit PAGE1ON
F3EA           874      ;
F3EA 90 E8      875      bcc <2              ; always taken
F3EC           876      ;
F3EC           877      ;
F3EC           878      ; Discretely and quickly clear the selected HIRES screen.
F3EC           879      ;
F3EC A2 00      880      CLRHIREs ldx #ZERO          ; set for black background
F3EE           881      ;
F3EE 86 1C      882      SETHIREs stx COLBITS
F3F0           883      ;
F3F0 A0 00      884      ldy #ZERO
F3F2 84 1A      885      sty SHAPE
F3F4           886      ;
F3F4 85 1B      887      sta SHAPE+1
F3F6 85 E6      888      sta HRPAG          ; selected HIRES secreen

```

```
F3F8      889  ;
F3F8 A2 20  890      ldx /PAGE40-PAGE20
F3FA      891  ;
F3FA A5 1C  892 ^1    lda COLBITS
F3FC 91 1A  893      sta (SHAPE),Y
F3FE      894  ;
F3FE 20 7E F4 895      jsr COLSHIFT
F401      896  ;
F401 C8     897      iny
F402 D0 F6  898      bne <1
F404      899  ;
F404 E6 1B  900      inc SHAPE+1
F406      901  ;
F406 CA     902      dex
F407 D0 F1  903      bne <1
F409      904  ;
F409 2C 57 C0 905      bit HIRESON
F40C 2C 50 C0 906      bit TEXTOFF
F40F      907  ;
F40F 18     908      clc                      ; for branch always taken
F410      909  ;
F410 60     910      rts
F411      911  ;
F411      912  ;
F411      913      icl "F4.L"
```

LLOAD F4.L,A\$4000


```

F411      1          ttl "ROM Source Code, F4.L"
F411      2      ;
F411      3      ;
F411      4      ; F4.L
F411      5      ;
F411      6      ;
F411      7      ; Set the HIRES cursor position. (X/Y) for horizontal for
F411      8      ; 0-279 and (A) for vertical for 0-191.
F411      9      ;
F411     10      ;          ---A-reg-- -GBASL-- -GBASH--
F411  86 E0     11  HPOSN      stx HRXCOOR
F413  84 E1     12              sty HRXCOOR+1
F415  85 E2     13              sta HRYCOOR          ; --ABCDEFGH -----
F417     14      ;
F417  48        15              pha          ; --ABCDEFGH -----
F418     16      ;
F418  29 C0     17              and #$C0          ; --AB000000 -----
F41A  85 26     18              sta GBASL          ; --AB000000 AB000000 -----
F41C     19      ;
F41C  4A        20              lsr          ; 0-0AB00000 AB000000 -----
F41D  4A        21              lsr          ; 0-00AB0000 AB000000 -----
F41E     22      ;
F41E  05 26     23              ora GBASL          ; 0-ABAB0000 AB000000 -----
F420  85 26     24              sta GBASL          ; 0-ABAB0000 ABAB0000 -----
F422     25      ;
F422  68        26              pla          ; 0-ABCDEFGH ABAB0000 -----
F423  85 27     27              sta GBASH          ; 0-ABCDEFGH ABAB0000 ABCDEFGH
F425     28      ;
F425  0A        29              asl          ; A-BCDEFGH0 ABAB0000 ABCDEFGH
F426  0A        30              asl          ; B-CDEFGH00 ABAB0000 ABCDEFGH
F427     31      ;
F427  0A        32              asl          ; C-DEFGH000 ABAB0000 ABCDEFGH
F428  26 27     33              rol GBASH          ; A-DEFGH000 ABAB0000 BCDEFGHC
F42A     34      ;
F42A  0A        35              asl          ; D-EFGH0000 ABAB0000 BCDEFGHC
F42B  26 27     36              rol GBASH          ; B-EFGH0000 ABAB0000 CDEFGHCD
F42D     37      ;
F42D  0A        38              asl          ; E-FGH00000 ABAB0000 CDEFGHCD
F42E  66 26     39              ror GBASL          ; 0-FGH00000 EABAB000 CDEFGHCD
F430     40      ;
F430  A5 27     41              lda GBASH          ; 0-CDEFGHCD EABAB000 CDEFGHCD
F432  29 1F     42              and #$1F          ; 0-000FGHCD EABAB000 CDEFGHCD
F434     43      ;
F434  05 E6     44              ora HRPAG          ; 0-PPPPFGHCD EABAB000 CDEFGHCD
F436  85 27     45              sta GBASH          ; 0-PPPPFGHCD EABAB000 PPPFGHCD
F438     46      ;
F438  8A        47              txa
F439     48      ;
F439  C0 00     49              cpy #ZERO
F43B  F0 05     50              beq >2          ; carry set either way
F43D     51      ;
F43D     52      ;
F43D     53      ; Pixel 256 is 256/7 = line byte #36 with remainder 4 bits.
F43D     54      ;
F43D  A0 23     55              ldy #35
F43F     56      ;
F43F  69 04     57              adc #4
F441     58      ;
F441  C8        59      ^1      iny
F442     60      ;

```

```

F442 E9 07      61  ^2      sbc #PXLSBYTE
F444 B0 FB      62          bcs <1
F446           63  ;
F446 84 E5      64          sty HRHORZ
F448           65  ;
F448 AA         66          tax
F449           67  ;
F449           68  ;
F449           69  ; X-reg ranges from $F9 to $FF so it can be used as an
F449           70  ; index of 0 to 6, respectively.
F449           71  ;
F449 BD 37 F4   72          lda BITABLE-$F9,X
F44C 85 30      73          sta COLOR
F44E           74  ;
F44E 98         75          tya
F44F 4A         76          lsr
F450           77  ;
F450 A5 E4      78          lda HRCOLOR
F452 85 1C      79          sta COLBITS
F454           80  ;
F454 B0 28      81          bcs COLSHIFT
F456           82  ;
F456 60         83          rts
F457           84  ;
F457           85  ;
F457           86  ; Enter HRPLOT with the following start coordinates:
F457           87  ;
F457           88  ; X-reg, HIRES X-coordinate, low order HORZ byte (0-255)
F457           89  ; Y-reg, HIRES X-coordinate, high order HORZ byte (256-279)
F457           90  ; A-reg, HIRES Y-coordinate VERT byte (0-191)
F457           91  ;
F457           92  ; In HIRES drawing, pixel bits are displayed backwards from
F457           93  ; their bit locations in byte pairs for color conservation.
F457           94  ;
F457           95  ; color byte:   byte N   byte N+1           where s is the
F457           96  ; bit pixel:  s6543210  sEDCBA98           color group bit
F457           97  ;
F457 20 11 F4   98  HRPLOT   jsr HPOSN
F45A           99  ;
F45A B1 26     100          lda (GBASL),Y
F45C 45 1C     101          eor COLBITS
F45E 25 30     102          and COLOR
F460           103  ;
F460 51 26     104          eor (GBASL),Y
F462 91 26     105          sta (GBASL),Y
F464           106  ;
F464 60         107          rts
F465           108  ;
F465           109  ;
F465 10 23     110  HRMOVLf  bpl HRMOVRT           ; Y-reg = HRHORZ
F467           111  ;
F467 A5 30     112          lda COLOR
F469           113  ;
F469 4A         114          lsr                     ; bit 0 pixel test
F46A B0 05     115          bcs >1
F46C           116  ;
F46C 49 C0     117          eor #$C0           ; add bit 7 pixel
F46E           118  ;
F46E 85 30     119  HRMOVLf2 sta COLOR
F470           120  ;
F470 60         121          rts

```

```

F471      122 ;
F471 88    123 ^1      dey                ; move left
F472 10 02  124      bpl >2
F474      125 ;
F474 A0 27  126      ldy #39                ; continue on screen right
F476      127 ;
F476 A9 C0  128 ^2      lda #$C0            ; mask for bit 6 pixel
F478      129 ;
F478 85 30  130 HRMOVL3 sta COLOR
F47A 84 E5  131      sty HRHORZ
F47C      132 ;
F47C A5 1C  133      lda COLBITS
F47E      134 ;
F47E      135 ;
F47E      136 ; No branch if 0x1F<COLBITS<0xE0.
F47E      137 ;
F47E 0A     138 COLSHIFT asl
F47F      139 ;
F47F C9 C0  140      cmp #$C0
F481 10 06  141      bpl >3
F483      142 ;
F483 A5 1C  143      lda COLBITS
F485 49 7F  144      eor #MSBCLR            ; make inverse for odd byte
F487 85 1C  145      sta COLBITS
F489      146 ;
F489 60     147 ^3      rts
F48A      148 ;
F48A      149 ;
F48A A5 30  150 HRMOVRT lda COLOR
F48C 0A     151      asl                    ; bit 7 pixel test
F48D      152 ;
F48D 49 80  153      eor #MSBSET
F48F 30 DD  154      bmi HRMOVL3          ; all except bit 7 pixel
F491      155 ;
F491 A9 81  156      lda #$81            ; get bit 0 pixel
F493      157 ;
F493 C8     158      iny                    ; move right
F494      159 ;
F494 C0 28  160      cpy #40
F496 90 E0  161      bcc HRMOVL3
F498      162 ;
F498 A0 00  163      ldy #ZERO            ; continue on screen left
F49A      164 ;
F49A B0 DC  165      bcs HRMOVL3          ; always taken
F49C      166 ;
F49C      167 ;
F49C      168 ; The Draw Shape routines have been heavily modified. The
F49C      169 ; DRAWHDR common header uses the MSB of OPRND to fall into
F49C      170 ; either XDRAWIT or DRAWIT. Each has specific code before
F49C      171 ; falling into common code. The XDRAWIT header was flawed
F49C      172 ; and generated anomalies. The common code was duplicated.
F49C      173 ;
F49C A5 D0  174 DRAWHDR lda SHPVAL
F49E 29 04  175      and #PLOTMASK
F4A0 F0 24  176      beq >3
F4A2      177 ;
F4A2 24 44  178      bit OPRND                    ; XDRAWIT/DRAWIT flag
F4A4 10 12  179      bpl DRAWIT
F4A6      180 ;
F4A6      181 ;
F4A6      182 ; XDRAW code.

```

```

F4A6          183 ;
F4A6 B1 26    184 XDRAWIT lda (GBASL),Y
F4A8 25 1C    185          and COLBITS          ; added to support color
F4AA 25 30    186          and COLOR
F4AC 29 7F    187          and #MSBCLR
F4AE D0 12    188          bne >2
F4B0          189 ;
F4B0 A5 1C    190          lda COLBITS          ; added to support color
F4B2 25 30    191          and COLOR
F4B4 29 7F    192          and #MSBCLR
F4B6 10 08    193          bpl >1          ; always taken
F4B8          194 ;
F4B8          195 ;
F4B8          196 ; DRAW code. Same drawing logic that is used in HPLOT.
F4B8          197 ;
F4B8 B1 26    198 DRAWIT  lda (GBASL),Y
F4BA 45 1C    199          eor COLBITS
F4BC 25 30    200          and COLOR
F4BE D0 02    201          bne >2
F4C0          202 ;
F4C0 E6 EA    203 ^1      inc HRCOLCNT          ; collision counter
F4C2          204 ;
F4C2          205 ;
F4C2          206 ; Common XDRAW/DRAW code.
F4C2          207 ;
F4C2 51 26    208 ^2      eor (GBASL),Y
F4C4 91 26    209          sta (GBASL),Y
F4C6          210 ;
F4C6          211 ;
F4C6          212 ; C-flag clear in horz summation and set in vert summation.
F4C6          213 ;
F4C6 A5 D0    214 ^3      lda SHPVAL
F4C8 65 D1    215          adc ROTQVAL
F4CA          216 ;
F4CA 29 03    217          and #MOVEMASK
F4CC C9 02    218          cmp #2          ; for down or left
F4CE          219 ;
F4CE          220 ;
F4CE          221 ; ROR sets MSB if move direction is down or left and C-flag
F4CE          222 ; is set if move direction is left or right.
F4CE          223 ;
F4CE 6A      224          ror
F4CF B0 94    225          bcs HRMOVLF
F4D1          226 ;
F4D1          227 ;
F4D1          228 ; Move up one scan line. Enters with C-flag clear.
F4D1          229 ; Scan line 00 at 0x2000 -> 0x3FD0.
F4D1          230 ;
F4D1 30 2E    231 HRMOVUP bmi HRMOVDN          ; branch if down
F4D3          232 ;
F4D3 A5 27    233          lda GBASH
F4D5 2C 2F F5 234          bit BITBYT1C
F4D8 D0 22    235          bne >5
F4DA          236 ;
F4DA 06 26    237          asl GBASL          ; 00, 08, 10, 18, 20, 28 ...
F4DC B0 1A    238          bcs >3
F4DE          239 ;
F4DE 2C 2D F5 240          bit BITBYT03          ; 00, 10, 20, 30, 40, 50 ...
F4E1 F0 05    241          beq >1
F4E3          242 ;
F4E3 69 1F    243          adc #$1F          ; 10, 20, 30, 50, 60, 70 ...

```

```

F4E5          244 ;
F4E5 38       245      sec
F4E6 B0 12    246      bcs >4                ; always taken
F4E8          247 ;
F4E8 69 23    248 ^1      adc #$23                ; 00, 40, 80 only
F4EA 48       249      pha
F4EB          250 ;
F4EB A5 26    251      lda GBASL
F4ED 69 B0    252      adc #$B0
F4EF B0 02    253      bcs >2
F4F1          254 ;
F4F1 69 F0    255      adc #$F0                ; 00 only
F4F3          256 ;
F4F3 85 26    257 ^2      sta GBASL
F4F5          258 ;
F4F5 68       259      pla
F4F6          260 ;
F4F6 B0 02    261      bcs >4                ; always taken
F4F8          262 ;
F4F8 69 1F    263 ^3      adc #$1F
F4FA          264 ;
F4FA 66 26    265 ^4      ror GBASL
F4FC          266 ;
F4FC 69 FC    267 ^5      adc #!-4                ; effectively, subtracts four
F4FE          268 ;
F4FE 85 27    269      sta GBASH
F500          270 ;
F500 60       271      rts
F501          272 ;
F501          273 ;
F501          274 ; Move down one scan line. Enters with C-flag clear.
F501          275 ; Scan line BF at 0x3FD0 -> 0x2000.
F501          276 ;
F501 A5 27    277 HRMOVDN  lda GBASH
F503 69 04    278      adc #4
F505          279 ;
F505 2C 2F F5 280      bit BITBYT1C
F508 D0 20    281      bne >4
F50A          282 ;
F50A 06 26    283      asl GBASL                ; 07, 0F, 17, 1F, 27, 2F ...
F50C 90 18    284      bcc >2
F50E          285 ;
F50E 69 E0    286      adc #$E0                ; 0F, 1F, 2F, 3F, 4F, 5F ...
F510          287 ;
F510 18       288      clc
F511          289 ;
F511 2C 2E F5 290      bit BITBYT04
F514 F0 12    291      beq >3
F516          292 ;
F516 A5 26    293      lda GBASL                ; 3F, 7F, BF only
F518 69 50    294      adc #$50
F51A          295 ;
F51A 49 F0    296      eor #$F0
F51C F0 02    297      beq >1
F51E          298 ;
F51E 49 F0    299      eor #$F0                ; 3F, 7F only
F520          300 ;
F520 85 26    301 ^1      sta GBASL
F522          302 ;
F522 A5 E6    303      lda HRPAG
F524          304 ;

```

```

F524 90 02      305      bcc >3
F526           306      ;
F526 69 E0      307      ^2      adc #$E0
F528           308      ;
F528 66 26      309      ^3      ror GBASL
F52A           310      ;
F52A 85 27      311      ^4      sta GBASH
F52C           312      ;
F52C 60         313      RTN.F5.2 rts
F52D           314      ;
F52D           315      ;
F52D 03         316      BITBYT03 hex 03
F52E 04         317      BITBYT04 hex 04
F52F 1C         318      BITBYT1C hex 1C
F530           319      ;
F530           320      ;
F530 81         321      BITABLE   byt %10000001
F531 82         322      byt %10000010
F532 84         323      byt %10000100
F533 88         324      byt %10001000
F534 90         325      byt %10010000
F535 A0         326      byt %10100000
F536 C0         327      byt %11000000
F537           328      ;
F537           329      ;
F537           330      dfs 3,ZERO      ; 3 bytes
F53A           331      ;
F53A           332      ;
F53A           333      ; Enter HLIN routine with the following end coordinates:
F53A           334      ;
F53A           335      ; A-reg, HIRES X-coordinate, low order byte (0-255)
F53A           336      ; X-reg, HIRES X-coordinate, high order byte (256-279)
F53A           337      ; Y-reg, HIRES Y-coordinate byte (0-191)
F53A           338      ;
F53A           339      ; This routine is modified using the HLINMOD directive as
F53A           340      ; documented in DOS 4.5 Volume and File Disk Management
F53A           341      ; System Second Edition.
F53A           342      ;
F53A 48         343      HLIN      pha
F53B           344      ;
F53B 38         345      sec
F53C           346      ;
F53C E5 E0      347      sbc HRXCOOR
F53E 48         348      pha
F53F           349      ;
F53F 8A         350      txa
F540           351      ;
F540 E5 E1      352      sbc HRXCOOR+1
F542 85 D3      353      sta HRFLAG      ; 0x00 - right, 0xFF - left
F544 B0 0A      354      bcs >1
F546           355      ;
F546 68         356      pla      ; make 2's compliment
F547 49 FF      357      eor #NEGONE
F549 69 01      358      adc #1
F54B 48         359      pha
F54C           360      ;
F54C A9 00      361      lda #ZERO
F54E E5 D3      362      sbc HRFLAG
F550           363      ;
F550 85 D1      364      ^1      sta HRXDELTA+1
F552 85 D5      365      sta HRWORK+1

```

```

F554          366 ;
F554 68       367      pla
F555 85 D0    368      sta HRXDELTA
F557 85 D4    369      sta HRWORK
F559          370 ;
F559 68       371      pla
F55A 85 E0    372      sta HRXCOOR
F55C 86 E1    373      stx HRXCOOR+1
F55E          374 ;
F55E 18       375      clc
F55F          376 ;
F55F 98       377      tya
F560 E5 E2    378      sbc HRYCOOR
F562 90 04    379      bcc >2                ; clr - up, set - down
F564          380 ;
F564 49 FF    381      eor #NEGONE
F566 69 FE    382      adc #!-2
F568          383 ;
F568 85 D2    384 ^2      sta HRYDELTA
F56A 84 E2    385      sty HRYCOOR
F56C          386 ;
F56C 66 D3    387      ror HRFLAG
F56E          388 ;
F56E 38       389      sec
F56F          390 ;
F56F E5 D0    391      sbc HRXDELTA
F571 AA       392      tax
F572          393 ;
F572 A9 FF    394      lda #NEGONE
F574 E5 D1    395      sbc HRXDELTA+1
F576 85 1D    396      sta COLCOUNT
F578          397 ;
F578 A4 E5    398      ldy HRHORZ
F57A          399 ;
F57A          400      .if HLINMOD
F57A          401 ;
F57A B0 04    402      bcs >4
F57C          403 ;
F57C          404 ;
F57C          405 ; This is the draw line loop from horizontal processing.
F57C          406 ;
F57C 0A       407 ^3      asl
F57D          408 ;
F57D 20 65 F4 409      jsr HRMOVLF
F580          410 ;
F580 18       411 ^4      clc
F581          412 ;
F581 A5 D4    413      lda HRWORK
F583          414 ;
F583          415      .el
F583          416 ;
F583          417      bcs >4
F583          418 ;
F583          419 ^3      asl
F583          420 ;
F583          421      jsr HRMOVLF
F583          422 ;
F583          423      sec
F583          424 ;
F583          425 ^4      lda HRWORK
F583          426 ;

```

```

F583          427          .fi
F583          428      ;
F583 65 D2    429          adc HRYDELTA
F585 85 D4    430          sta HRWORK
F587          431      ;
F587 A5 D5    432          lda HRWORK+1
F589 E9 00    433          sbc #ZERO
F58B          434      ;
F58B          435      ;
F58B          436      ; This is the draw line loop from vertical processing.
F58B          437      ;
F58B 85 D5    438      ^5          sta HRWORK+1
F58D          439      ;
F58D B1 26    440          lda (GBASL),Y
F58F 45 1C    441          eor COLBITS
F591 25 30    442          and COLOR
F593          443      ;
F593 51 26    444          eor (GBASL),Y
F595 91 26    445          sta (GBASL),Y
F597          446      ;
F597 E8       447          inx
F598 D0 04    448          bne >6
F59A          449      ;
F59A E6 1D    450          inc COLCOUNT
F59C F0 8E    451          beq RTN.F5.2
F59E          452      ;
F59E A5 D3    453      ^6          lda HRFLAG
F5A0          454      ;
F5A0 B0 DA    455          bcs <3
F5A2          456      ;
F5A2          457      ;
F5A2          458      ; Verical processing.
F5A2          459      ;
F5A2 20 D1 F4 460          jsr HRMOVUP
F5A5          461      ;
F5A5          462          .if HLINMOD
F5A5          463      ;
F5A5 38       464          sec
F5A6          465      ;
F5A6          466          .el
F5A6          467      ;
F5A6          468          clc
F5A6          469      ;
F5A6          470          .fi
F5A6          471      ;
F5A6 A5 D4    472          lda HRWORK
F5A8 65 D0    473          adc HRXDELTA
F5AA 85 D4    474          sta HRWORK
F5AC          475      ;
F5AC A5 D5    476          lda HRWORK+1
F5AE 65 D1    477          adc HRXDELTA+1
F5B0          478      ;
F5B0 4C 8B F5 479          jmp <5          ; replaced bvc instruction
F5B3          480      ;
F5B3          481      ;
F5B3          482      ; Rotational values in 5.625 degrees based on 360/64.
F5B3          483      ; These values are COS( 90 * X/16 ) * 0x100 rather than
F5B3          484      ; * 0xFF. These value are more precise.
F5B3          485      ;
F5B3 00       486      ROTATBL hex 00          ; no rotation
F5B4 FF       487          hex FF

```



```

F5B5 FB          488          hex FB
F5B6 F5          489          hex F5
F5B7 ED          490          hex ED
F5B8 E2          491          hex E2
F5B9 D5          492          hex D5
F5BA C6          493          hex C6
F5BB B5          494          hex B5          ; 45 degrees
F5BC A2          495          hex A2
F5BD 8E          496          hex 8E
F5BE 79          497          hex 79
F5BF 62          498          hex 62
F5C0 4A          499          hex 4A
F5C1 32          500          hex 32
F5C2 19          501          hex 19
F5C3 00          502          hex 00          ; no rotation
F5C4             503          ;
F5C4             504          ;
F5C4 4C 9B E1    505  IQ.ERR5  jmp IQ.ERR          ; Illegal Quantity error
F5C7             506          ;
F5C7             507          ;
F5C7             508          ; This is the DRAWCMD routine.
F5C7             509          ;
F5C7 20 F8 E6    510  DRAWCMD  jsr GETBYT          ; get requested SHAPE number
F5CA             511          ;
F5CA A5 E8       512          lda HRSHPTBL
F5CC 85 1A       513          sta SHAPE
F5CE             514          ;
F5CE A5 E9       515          lda HRSHPTBL+1
F5D0 85 1B       516          sta SHAPE+1
F5D2             517          ;
F5D2 8A          518          txa          ; SHAPE number to draw
F5D3             519          ;
F5D3 A2 00       520          ldx #ZERO
F5D5             521          ;
F5D5 C1 1A       522          cmp (SHAPE,X)
F5D7 F0 02       523          beq >1
F5D9             524          ;
F5D9 B0 E9       525          bcs IQ.ERR5
F5DB             526          ;
F5DB 0A          527  ^1      asl
F5DC 90 03       528          bcc >2
F5DE             529          ;
F5DE E6 1B       530          inc SHAPE+1
F5E0             531          ;
F5E0 18          532          clc
F5E1             533          ;
F5E1 A8          534  ^2      tay
F5E2             535          ;
F5E2 B1 1A       536          lda (SHAPE),Y
F5E4 65 E8       537          adc HRSHPTBL
F5E6 AA          538          tax
F5E7             539          ;
F5E7 C8          540          iny
F5E8             541          ;
F5E8 B1 1A       542          lda (SHAPE),Y
F5EA 65 E9       543          adc HRSHPTBL+1
F5EC             544          ;
F5EC 86 1A       545          stx SHAPE
F5EE 85 1B       546          sta SHAPE+1
F5F0             547          ;
F5F0 20 B7 00    548          jsr CHRGOT

```

```

F5F3          549 ;
F5F3 C9 C5    550      cmp #TK.AT
F5F5 D0 09    551      bne >3
F5F7          552 ;
F5F7 20 C0 DE 553      jsr SYNTAXCHK
F5FA 20 B9 F6 554      jsr GETFNS          ; get coordinants
F5FD 20 11 F4 555      jsr HPOSN          ; set GBASL/GBASH, E0:E2, E5
F600          556 ;
F600          557 ;
F600          558 ; This is the DRAWSHP routine.
F600          559 ;
F600 A5 F9    560 ^3      lda HRROT
F602          561 ;
F602 4A       562      lsr
F603 4A       563      lsr
F604 4A       564      lsr
F605 4A       565      lsr
F606          566 ;
F606 85 D1    567      sta ROTQVAL
F608          568 ;
F608 A5 F9    569      lda HRROT
F60A 29 0F    570      and #SROTMASK
F60C AA       571      tax
F60D          572 ;
F60D BC B3 F5 573      ldy ROTATBL,X
F610 88       574      dey
F611 84 D2    575      sty ROTHVAL
F613          576 ;
F613 49 0F    577      eor #SROTMASK
F615 AA       578      tax
F616          579 ;
F616 BC B4 F5 580      ldy ROTATBL+1,X
F619 84 D3    581      sty ROTVVAL
F61B          582 ;
F61B A4 E5    583      ldy HRHORZ
F61D          584 ;
F61D A2 FF    585      ldx #NEGONE
F61F 86 D7    586      stx SHPOLD
F621          587 ;
F621 E8       588      inx
F622 86 EA    589      stx HRCOLCNT          ; initialize to zero
F624          590 ;
F624 A1 1A    591      lda (SHAPE,X)
F626          592 ;
F626          593 ;
F626          594 ; If there is no change in direction, keep summing
F626          595 ; coordinate values.
F626          596 ;
F626 85 D0    597 ^4      sta SHPVAL
F628 29 07    598      and #SCMDMASK
F62A          599 ;
F62A C5 D7    600      cmp SHPOLD
F62C F0 08    601      beq >5
F62E          602 ;
F62E 85 D7    603      sta SHPOLD
F630          604 ;
F630 A9 00    605      lda #ZERO
F632 85 D4    606      sta ROTHSUM
F634 85 D5    607      sta ROTVSUM
F636          608 ;
F636 A6 E7    609 ^5      ldx HRSCALE

```

```

F638      610 ;
F638      611 ;
F638      612 ; Horizontal summation.
F638      613 ;
F638 38    614 ^6      sec
F639      615 ;
F639 A5 D4  616      lda ROTHSUM
F63B 65 D2  617      adc ROTHVAL
F63D 85 D4  618      sta ROTHSUM
F63F 90 05  619      bcc >7
F641      620 ;
F641 18     621      clc
F642      622 ;
F642 20 9C F4 623      jsr DRAWHDR
F645      624 ;
F645 18     625      clc
F646      626 ;
F646      627 ;
F646      628 ; Vertical summation.
F646      629 ;
F646 A5 D5  630 ^7      lda ROTVSUM
F648 65 D3  631      adc ROTVVAL
F64A 85 D5  632      sta ROTVSUM
F64C 90 03  633      bcc >8
F64E      634 ;
F64E 20 9C F4 635      jsr DRAWHDR
F651      636 ;
F651 CA     637 ^8      dex
F652 D0 E4  638      bne <6
F654      639 ;
F654 A5 D0  640      lda SHPVAL
F656      641 ;
F656 4A     642      lsr
F657 4A     643      lsr
F658 4A     644      lsr
F659      645 ;
F659 D0 CB  646      bne <4
F65B      647 ;
F65B      648 ;
F65B      649 ; Point to next shape in table.
F65B      650 ;
F65B E6 1A  651      inc SHAPE
F65D D0 02  652      bne >9
F65F      653 ;
F65F E6 1B  654      inc SHAPE+1
F661      655 ;
F661 A1 1A  656 ^9      lda (SHAPE,X)
F663 D0 C1  657      bne <4
F665      658 ;
F665 60     659      rts
F666      660 ;
F666      661 ;
F666      662 ; Continuation of FSQR.
F666      663 ;
F666 85 9D  664 SQR2      sta FACEXP
F668      665 ;
F668 A0 07  666      ldy #7          ; max iteration counter value
F66A 84 AD  667      sty SAVY
F66C      668 ;
F66C 20 BA F1 669 ^3      jsr COPYF2T3
F66F      670 ;

```

```

F66F A9 93      671      lda #TEMP1
F671 A0 00      672      ldy /TEMP1
F673 20 66 EA    673      jsr FDIV
F676           674      ;
F676 20 A8 F6    675      jsr COPYT32A
F679 20 C1 E7    676      jsr ADD2
F67C           677      ;
F67C C6 9D      678      dec FACEXP
F67E           679      ;
F67E C6 AD      680      dec SAVY
F680 F0 09      681      beq >4
F682           682      ;
F682 A9 8A      683      lda #TEMP3
F684 A0 00      684      ldy /TEMP3
F686           685      ;
F686 20 B5 EB    686      jsr FPCOMP
F689 D0 E1      687      bne <3
F68B           688      ;
F68B 4C 70 EB    689      ^4      jmp RNDUP
F68E           690      ;
F68E           691      ;
F68E           692      ; Continuation of COPYA2F.
F68E           693      ;
F68E A5 92      694      COPYA2F3 lda ARGGUARD
F690 85 AC      695      sta FACGUARD
F692           696      ;
F692 60         697      rts
F693           698      ;
F693           699      ;
F693           700      ; Continuation of COPYF2A.
F693           701      ;
F693 A5 A0      702      COPYF2A2 lda FACMANT+2
F695 85 A8      703      sta ARGMANT+2
F697           704      ;
F697 A5 9F      705      lda FACMANT+1
F699 85 A7      706      sta ARGMANT+1
F69B           707      ;
F69B A5 9E      708      lda FACMANT
F69D 85 A6      709      sta ARGMANT
F69F           710      ;
F69F A5 AC      711      lda FACGUARD
F6A1 85 92      712      sta ARGGUARD
F6A3           713      ;
F6A3 A5 9D      714      lda FACEXP
F6A5 85 A5      715      sta ARGEXP
F6A7           716      ;
F6A7 60         717      rts
F6A8           718      ;
F6A8           719      ;
F6A8           720      ; COPYT32A routine entry point. Copy T3GUARD to ARGGUARD
F6A8           721      ; using (A/Y) -> ARG.
F6A8           722      ;
F6A8 A6 8F      723      COPYT32A ldx T3GUARD
F6AA           724      ;
F6AA A9 8A      725      lda #TEMP3
F6AC A0 00      726      ldy /TEMP3
F6AE 20 E3 E9    727      jsr LOADARG
F6B1           728      ;
F6B1 86 92      729      stx ARGGUARD      ; maintains FACEXP validity
F6B3           730      ;
F6B3 60         731      rts

```

```

F6B4      732 ;
F6B4      733 ;
F6B4      734      dfs 5,ZERO      ; 5 bytes
F6B9      735 ;
F6B9      736 ;
F6B9      737 ; Get HPOSN coordinates: (X/Y) = HORZ, A-reg = VERT.
F6B9      738 ;
F6B9 20 64 DD 739 GETFNS      jsr FRMNUM
F6BC 20 52 E7 740      jsr GETADDR
F6BF      741 ;
F6BF A6 50 742      ldx LINNUM
F6C1 A4 51 743      ldY LINNUM+1
F6C3      744 ;
F6C3 C0 01 745      cpy /280      ; must be less than 280
F6C5 90 06 746      bcc >1
F6C7      747 ;
F6C7 D0 1D 748      bne IQ.ERR6
F6C9      749 ;
F6C9 E0 18 750      cpx #280
F6CB B0 19 751      bcs IQ.ERR6
F6CD      752 ;
F6CD 8A 753      ^1      txa
F6CE 48 754      pha
F6CF      755 ;
F6CF 98 756      tya
F6D0 48 757      pha
F6D1      758 ;
F6D1 A9 2C 759      lda #', '
F6D3 20 C0 DE 760      jsr SYNTAXCHK
F6D6      761 ;
F6D6 20 F8 E6 762      jsr GETBYT
F6D9      763 ;
F6D9 E0 C0 764      cpx #192      ; maximum vertical line
F6DB B0 09 765      bcs IQ.ERR6
F6DD      766 ;
F6DD 86 9D 767      stx DSCTMP
F6DF      768 ;
F6DF 68 769      pla
F6E0 A8 770      tay
F6E1      771 ;
F6E1 68 772      pla
F6E2 AA 773      tax
F6E3      774 ;
F6E3 A5 9D 775      lda DSCTMP
F6E5      776 ;
F6E5 60 777      rts
F6E6      778 ;
F6E6      779 ;
F6E6 4C 9B E1 780 IQ.ERR6      jmp IQ.ERR      ; Illegal Quantity error
F6E9      781 ;
F6E9      782 ;
F6E9 20 F8 E6 783 BHCOLOR      jsr GETBYT
F6EC      784 ;
F6EC E0 08 785      cpx #8
F6EE B0 F6 786      bcs IQ.ERR6
F6F0      787 ;
F6F0 BD F6 F6 788      lda HRCOLTBL,X
F6F3 85 E4 789      sta HRCOLOR
F6F5      790 ;
F6F5 60 791      RTN.F6.F rts
F6F6      792 ;

```

```

F6F6          793 ;
F6F6 00 2A 55 794 HRCOLTBL hex 002A557F ; MSB off color group
F6F9 7F
F6FA 80 AA D5 795          hex 80AAD5FF ; MSB on color group
F6FD FF
F6FE          796 ;
F6FE          797 ;
F6FE C9 C1    798 BHPlot      cmp #TK.TO
F700 F0 0D    799          beq >1
F702          800 ;
F702 20 B9 F6 801          jsr GETFNS
F705 20 57 F4 802          jsr HRPlot
F708          803 ;
F708 20 B7 00 804 HPlot2     jsr CHRGOT
F70B          805 ;
F70B C9 C1    806          cmp #TK.TO
F70D D0 E6    807          bne RTN.F6.F
F70F          808 ;
F70F 20 C0 DE 809 ^1        jsr SYNTAXCHK
F712 20 B9 F6 810          jsr GETFNS
F715          811 ;
F715 84 9D    812          sty DSCTMP
F717          813 ;
F717 A8       814          tay
F718 8A       815          txa
F719          816 ;
F719 A6 9D    817          ldx DSCTMP
F71B          818 ;
F71B 20 3A F5 819          jsr HLIN
F71E          820 ;
F71E 4C 08 F7 821          jmp HPlot2 ; beq HPlot2 could be used
F721          822 ;
F721          823 ;
F721 20 F8 E6 824 BROT      jsr GETBYT
F724 86 F9    825          stx HRROT
F726          826 ;
F726 60       827          rts
F727          828 ;
F727          829 ;
F727 20 F8 E6 830 BSCALE   jsr GETBYT
F72A 86 E7    831          stx HRSCALE
F72C          832 ;
F72C 60       833          rts
F72D          834 ;
F72D          835 ;
F72D          836 ; Continuation of the FRND routine.
F72D          837 ;
F72D          838 RND2:
F72D          839 ;
F72D          840 ^1        lda RANDVAL1,X
F730 95 62    841          sta MULMANT,X
F732          842 ;
F732 BD AA EF 843          lda RANDVAL2,X
F735 95 9E    844          sta FACMANT,X
F737          845 ;
F737 B5 C9    846          lda IRAND,X
F739 95 A6    847          sta ARGMANT,X
F73B          848 ;
F73B CA       849          dex
F73C 10 EF    850          bpl <1
F73E          851 ;

```

```

F73E      852 ;
F73E      853 ; Multiply ARGMANT and MULMANT using the Peasant algorithm
F73E      854 ; for multiplication and save result into FAC and IRAND.
F73E      855 ;
F73E 46 62 856 ^2      lsr MULMANT
F740 66 63 857          ror MULMANT+1
F742 66 64 858          ror MULMANT+2
F744 66 65 859          ror MULMANT+3
F746      860 ;
F746 90 0E 861          bcc >4
F748      862 ;
F748 18    863          clc
F749      864 ;
F749 A2 03 865          ldx #3
F74B      866 ;
F74B B5 9E 867 ^3      lda FACMANT,X
F74D 75 A6 868          adc ARGMANT,X
F74F 95 9E 869          sta FACMANT,X
F751 95 C9 870          sta IRAND,X
F753      871 ;
F753 CA    872          dex
F754 10 F5 873          bpl <3
F756      874 ;
F756 06 A9 875 ^4      asl ARGMANT+3
F758 26 A8 876          rol ARGMANT+2
F75A 26 A7 877          rol ARGMANT+1
F75C 26 A6 878          rol ARGMANT
F75E      879 ;
F75E 88    880          dey
F75F D0 DD 881          bne <2
F761      882 ;
F761 84 A2 883          sty FACSIGN
F763 84 AC 884          sty FACGUARD
F765      885 ;
F765 A9 80 886          lda #$80
F767 D0 0C 887          bne RND3                ; always taken
F769      888 ;
F769      889 ;
F769      890 ; DRAW and XDRAW set the value of OPRND and they enter
F769      891 ; DRAWCMD. DRAWCMD is followed by DRAWSHP which was
F769      892 ; duplicated in the original code. Now, DRAWHDR combines
F769      893 ; two separate headers that operates based on the MSB of
F769      894 ; OPRND. DRAW and XDRAW are at their original location.
F769      895 ;
F769 18    896 BDRAW    clc
F76A      897 ;
F76A 66 44 898          ror OPRND
F76C      899 ;
F76C 4C C7 F5 900          jmp DRAWCMD
F76F      901 ;
F76F 38    902 BXDRAW    sec
F770      903 ;
F770 66 44 904          ror OPRND
F772      905 ;
F772 4C C7 F5 906          jmp DRAWCMD
F775      907 ;
F775      908 ;
F775      909 ; Convert the value in FAC to a floating point fraction.
F775      910 ;
F775 85 9D 911 RND3      sta FACEXP
F777      912 ;

```

```

F777 20 22 E8    913      jsr NORMFAC1
F77A             914      ;
F77A             915      ;
F77A             916      ; If Range is less than two, return the value in FAC as is,
F77A             917      ; otherwise multiply by TEMP1 Range and return as integer.
F77A             918      ;
F77A A5 93       919      lda TEMP1
F77C C9 82       920      cmp #$82
F77E 90 66       921      bcc RTN.F7.E          ; branch if Range < 2
F780             922      ;
F780 A9 93       923      lda #TEMP1
F782 A0 00       924      ldy /TEMP1
F784 20 7F E9    925      jsr FMULT
F787             926      ;
F787 4C 23 EC    927      jmp FINT
F78A             928      ;
F78A             929      ;
F78A             930      dfs 7,ZERO          ; 7 bytes
F791             931      ;
F791             932      ;
F791 C1 F0 F0    933      TITLE asc "Apple //e+"
F794 EC E5 A0
F797 AF AF E5
F79A AB
F79B             934      ;
000A             935      TITLEN equ *-TITLE
001C             936      DELTITLE equ 40-TITLEN+2
000E             937      OFFTITLE equ DELTITLE/2
F79B             938      ;
F79B             939      ;
F79B             940      ; Modifications to Applesoft in order to support an 80
F79B             941      ; column display and Applesoft program entry in lower case.
F79B             942      ;
F79B BD 01 02    943      PARSIEX1 lda INPUT+1,X
F79E 10 11       944      bpl >1
F7A0             945      ;
F7A0 A5 0E       946      PARSIEX2 lda ENDCHR
F7A2 F0 16       947      beq >3
F7A4             948      ;
F7A4 C9 22       949      cmp #'"'
F7A6 F0 12       950      beq >3
F7A8             951      ;
F7A8 A5 13       952      lda DATAFLG
F7AA C9 49       953      cmp #$46+BSDATA/2
F7AC F0 0C       954      beq >3
F7AE             955      ;
F7AE BD 00 02    956      PARSIEX3 lda INPUT,X
F7B1             957      ;
F7B1 08          958      ^1 php
F7B2             959      ;
F7B2 C9 61       960      cmp #$61
F7B4 90 02       961      bcc >2
F7B6             962      ;
F7B6 29 5F       963      and #$5F
F7B8             964      ;
F7B8 28          965      ^2 plp
F7B9             966      ;
F7B9 60          967      rts
F7BA             968      ;
F7BA BD 00 02    969      ^3 lda INPUT,X
F7BD             970      ;

```



```

F7BD 60          971          rts
F7BE            972          ;
F7BE            973          ;
F7BE 48          974 LISTEX1 pha
F7BF            975          ;
F7BF 20 53 DB    976          jsr OUTSPC
F7C2            977          ;
F7C2 68          978          pla
F7C3            979          ;
F7C3 4C 18 ED    980          jmp LINEPRT
F7C6            981          ;
F7C6            982          ;
F7C6            983 LISTEX2:
F7C6 A5 24       984 PRTCRESX1 lda CH
F7C8 C9 24       985          cmp #36          ; was 33
F7CA            986          ;
F7CA 2C 1F C0    987          bit RDVID80
F7CD 10 05       988          bpl >1
F7CF            989          ;
F7CF AD 7B 05    990          lda OURCH
F7D2 C9 4A       991          cmp #74          ; was 73
F7D4            992          ;
F7D4 60          993 ^1          rts
F7D5            994          ;
F7D5            995          ;
F7D5 8A          996 PRTCRESX2 txa
F7D6            997          ;
F7D6 2C 1F C0    998          bit RDVID80
F7D9 30 08       999          bmi >1
F7DB            1000         ;
F7DB 2C 00 00    1001         bit *-*
F7DE            1002         dfs !-2
F7DC            1003         ;
F7DC 85 24       1004 PRTCRESX3 sta CH
F7DE            1005         ;
F7DE 38          1006         sec
F7DF            1007         ;
F7DF 8A          1008         txa
F7E0 E5 24       1009         sbc CH
F7E2            1010         ;
F7E2 60          1011         rts
F7E3            1012         ;
F7E3 ED 7B 05    1013 ^1          sbc OURCH
F7E6            1014         ;
F7E6 60          1015 RTN.F7.E rts
F7E7            1016         ;
F7E7            1017         ;
F7E7 20 F8 E6    1018 BHTAB      jsr GETBYT
F7EA            1019         ;
F7EA CA          1020         dex
F7EB            1021         ;
F7EB A9 28       1022 ^1          lda #40
F7ED C5 21       1023          cmp WNDWDTH
F7EF B0 02       1024          bcs >2
F7F1            1025         ;
F7F1 A5 21       1026          lda WNDWDTH
F7F3            1027         ;
F7F3 20 DC F7    1028 ^2          jsr PRTCRESX3
F7F6            1029         ;
F7F6 86 24       1030          stx CH
F7F8            1031         ;

```

F7F8 90 EC	1032	bcc RTN.F7.E	
F7FA	1033		;
F7FA AA	1034	tax	
F7FB	1035		;
F7FB 20 50 DB	1036	jsr PRTCR	
F7FE D0 EB	1037	bne <1	; always taken
F800	1038		;
F800	1039		;
F800	1040	icl "F8.L"	

LLOAD F8.L,A\$4000

```

F800          1          ttl "ROM Source Code, F8.L"
F800          2          ;
F800          3          ;
F800          4          ; F8.L
F800          5          ;
F800          6          ;
F800          7          F8SPACE:
F800          8          ;
F800          9          ;
F800 4A       10         PLOT      lsr
F801 08       11          php                      ; save LSB in C-flag
F802          12          ;
F802 20 47 F8 13          jsr GBASCALC
F805          14          ;
F805 28       15          plp                      ; recall LSB
F806          16          ;
F806 A9 0F    17          lda #$0F                  ; mask 0x0F if even
F808          18          ;
F808 90 02    19          bcc RTMASK
F80A          20          ;
F80A 69 E0    21          adc #$E0                  ; mask 0xF0 if odd
F80C          22          ;
F80C 85 2E    23         RTMASK   sta MASK
F80E          24          ;
F80E B1 26    25         PLOT1    lda (GBASL),Y      ; get data
F810 45 30    26          eor COLOR                ; apply color
F812          27          ;
F812 25 2E    28          and MASK                ; select bits
F814          29          ;
F814 51 26    30          eor (GBASL),Y            ; apply data
F816 91 26    31          sta (GBASL),Y           ; save it
F818          32          ;
F818 60       33          rts
F819          34          ;
F819          35          ;
F819 20 00 F8 36         HLINE    jsr PLOT          ; plot square
F81C          37          ;
F81C C4 2C    38         HLINE1   cpy H2
F81E B0 11    39          bcs >2
F820          40          ;
F820 C8       41          iny
F821          42          ;
F821 20 0E F8 43          jsr PLOT1                ; plot next square
F824 90 F6    44          bcc HLINE1
F826          45          ;
F826 69 01    46          ^1      adc #1            ; next Y-coord
F828          47          ;
F828 48       48         VLINE    pha
F829          49          ;
F829 20 00 F8 50          jsr PLOT          ; plot square
F82C          51          ;
F82C 68       52          pla
F82D C5 2D    53          cmp V2
F82F 90 F5    54          bcc <1
F831          55          ;
F831 60       56          ^2      rts
F832          57          ;
F832          58          ;
F832 A0 2F    59         CLRSCR   ldy #$2F          ; max Y, full scrn clr
F834 D0 02    60          bne >3

```

```

F836      61 ;
F836 A0 27 62 CLRTOP    ldy #$27          ; max Y, top scrn clr
F838      63 ;
F838 84 2D 64 ^3      sty V2
F83A      65 ;
F83A A0 27 66          ldy #$27          ; rightmost X-coord (column)
F83C      67 ;
F83C A9 00 68 ^4      lda #ZERO          ; top coord for VLINE
F83E 85 30 69          sta COLOR          ; clear color to black
F840      70 ;
F840 20 28 F8 71        jsr VLINE          ; draw VLINE
F843      72 ;
F843 88      73          dey
F844 10 F6 74          bpl <4
F846      75 ;
F846 60      76          rts
F847      77 ;
F847      78 ;
F847 48      79 GBASCALC pha          ; for input ODDEF GH
F848      80 ;
F848 4A      81          lsr
F849      82 ;
F849 29 03 83          and #3
F84B 09 04 84          ora #4
F84D 85 27 85          sta GBASH          ; = 000001FG
F84F      86 ;
F84F 68      87          pla          ; = HDEDE000
F850 29 18 88          and #$18
F852      89 ;
F852 90 02 90          bcc >5
F854      91 ;
F854 69 7F 92          adc #$7F
F856      93 ;
F856 85 26 94 ^5      sta GBASL
F858      95 ;
F858 0A      96          asl
F859 0A      97          asl
F85A      98 ;
F85A 05 26 99          ora GBASL
F85C 85 26 100         sta GBASL
F85E      101 ;
F85E 60      102         rts
F85F      103 ;
F85F      104 ;
F85F A5 30 105 NXTCOL   lda COLOR          ; increment by 3
F861      106 ;
F861 18      107         clc
F862      108 ;
F862 69 03 109         adc #3
F864      110 ;
F864 29 0F 111 SETCOL   and #$0F          ; COLOR=17*A MOD 16m 114
F866 85 30 112         sta COLOR
F868      113 ;
F868 0A      114         asl          ; both half bytes equal
F869 0A      115         asl
F86A 0A      116         asl
F86B 0A      117         asl
F86C      118 ;
F86C 05 30 119         ora COLOR
F86E 85 30 120         sta COLOR
F870      121 ;

```

```

F870 60          122          rts
F871          123          ;
F871          124          ;
F871 4A          125  SCRN    lsr                ; screen Y-coord/2
F872 08          126          php                ; save LSB
F873          127          ;
F873 20 47 F8    128          jsr GBASCALC        ; calc base address
F876          129          ;
F876 B1 26       130          lda (GBASL),Y      ; get data
F878          131          ;
F878 28          132          plp                ; recall LSB
F879          133          ;
F879 90 04       134  SCRN2   bcc >6            ; if even use LO half
F87B          135          ;
F87B 4A          136          lsr                ; shift HI half down
F87C 4A          137          lsr
F87D 4A          138          lsr
F87E 4A          139          lsr
F87F          140          ;
F87F 29 0F       141  ^6      and #$0F          ; mask 4 bits
F881          142          ;
F881 60          143          rts
F882          144          ;
F882          145          ;
F882 A6 3A       146  INSDS1   ldx PCL            ; print PCL,PCH
F884 A4 3B       147          ldy PCH
F886          148          ;
F886 20 96 FD    149          jsr PRXY
F889 20 48 F9    150          jsr PRBLNK
F88C          151          ;
F88C A1 3A       152          lda (PCL,X)        ; get opcode
F88E          153          ;
F88E A8          154  INSDS2   tay
F88F          155          ;
F88F 4A          156          lsr                ; even/odd test
F890 90 05       157          bcc >7
F892          158          ;
F892 6A          159          ror                ; bit 1 test
F893 B0 0C       160          bcs >8            ; XXXXXX11 invalid OP
F895          161          ;
F895 29 87       162          and #$87          ; mask bits
F897          163          ;
F897 4A          164  ^7      lsr                ; C-flag = LSB for later
F898 AA          165          tax
F899          166          ;
F899 BD 62 F9    167          lda FMT1,X        ; get FORMAT index
F89C          168          ;
F89C 20 79 F8    169          jsr SCRN2
F89F D0 04       170          bne GETFMT
F8A1          171          ;
F8A1 A0 03       172  ^8      ldy #3            ; invalid OP
F8A3 A9 00       173          lda #ZERO         ; error FORMAT
F8A5          174          ;
F8A5          175          ;
F8A5          176          ; Move remaining code to 0xC1-0xC2 because the code that
F8A5          177          ; tests the ROM in slot 3 must be in the 0xF8 ROM space.
F8A5          178          ;
F8A5 AA          179  GETFMT   tax
F8A6          180          ;
F8A6 BD C7 FB    181          lda FMT2,X        ; get instruction format
F8A9 C9 49       182          cmp #$49         ; is it relative (zpage)

```

```

F8AB D0 01      183      bne >1
F8AD           184      ;
F8AD 88         185      dey                ; correct index
F8AE           186      ;
F8AE AA         187      ^1      tax
F8AF           188      ;
F8AF 84 2A      189      sty BAS2L
F8B1           190      ;
F8B1 A0 10      191      ldy #16
F8B3           192      ;
F8B3 4C B4 FB   193      jmp GOTOROM
F8B6           194      ;
F8B6           195      ;
F8B6           196      ; Test slot 3 for a card containing a ROM.  If there is one
F8B6           197      ; the internal slot 3 firmware for 80 columns will not be
F8B6           198      ; used.  Y-reg is set to a reasonable value.  Bytes checked
F8B6           199      ; are 0xC305 for 0x38 (sec) and 0xC307 for 0x18 (clc).
F8B6           200      ;
F8B6 8D 06 C0   201      TESTROM sta CXROMOFF        ; enable slots
F8B9           202      ;
F8B9 A0 08      203      ldy #8
F8BB           204      ;
F8BB A2 02      205      ^1      ldx #2
F8BD           206      ;
F8BD BD 05 C3   207      ^2      lda BASICIN,X
F8C0 DD 9C FC   208      cmp CLREOL,X
F8C3 D0 07      209      bne >3
F8C5           210      ;
F8C5 CA         211      dex
F8C6 CA         212      dex
F8C7           213      ;
F8C7 10 F4      214      bpl <2
F8C9           215      ;
F8C9 88         216      dey
F8CA D0 EF      217      bne <1
F8CC           218      ;
F8CC 8D 07 C0   219      ^3      sta CXROMON        ; swap in internal ROM
F8CF           220      ;
F8CF 60         221      rts
F8D0           222      ;
F8D0           223      ;
F8D0           224      ; Disassemble instruction.
F8D0           225      ;
F8D0 20 82 F8   226      INSTDSP jsr INSDS1        ; get format, length
F8D3           227      ;
F8D3 48         228      pha
F8D4           229      ;
F8D4 B1 3A      230      PRNTOP  lda (PCL),Y
F8D6           231      ;
F8D6 20 DA FD   232      jsr PRBYTE
F8D9           233      ;
F8D9 A2 01      234      ldx #1                ; print 2 blanks
F8DB           235      ;
F8DB 20 4A F9   236      PRNTBL  jsr PRBL2
F8DE           237      ;
F8DE C4 2F      238      cpy LENGTH
F8E0           239      ;
F8E0 C8         240      iny
F8E1           241      ;
F8E1 90 F1      242      bcc PRNTOP
F8E3           243      ;

```

```

F8E3 A2 03      244      ldx #3                ; 12 chr field
F8E5            245      ;
F8E5 C0 04      246      cpy #4
F8E7 90 F2      247      bcc PRNTBL
F8E9            248      ;
F8E9 68          249      pla
F8EA A8          250      tay
F8EB            251      ;
F8EB B9 A6 F9    252      lda MNEML,Y          ; get mnemonic
F8EE 85 2C      253      sta LMNEM
F8F0            254      ;
F8F0 B9 EB F9    255      lda MNEMR,Y
F8F3 85 2D      256      sta RMNEM
F8F5            257      ;
F8F5 A9 00      258      ^4      lda #ZERO
F8F7 A0 05      259      ldy #5
F8F9            260      ;
F8F9 06 2D      261      ^5      asl RMNEM
F8FB 26 2C      262      rol LMNEM
F8FD            263      ;
F8FD 2A          264      rol
F8FE            265      ;
F8FE 88          266      dey
F8FF D0 F8      267      bne <5
F901            268      ;
F901 69 BF      269      adc #"@"-1          ; add in offset
F903            270      ;
F903 20 ED FD    271      jsr COUT
F906            272      ;
F906 CA          273      dex
F907 D0 EC      274      bne <4
F909            275      ;
F909 20 48 F9    276      jsr PRBLNK          ; print 3 blanks
F90C            277      ;
F90C A4 2F      278      ldy LENGTH
F90E A2 06      279      ldx #6                ; 6 format bits
F910            280      ;
F910 E0 03      281      ^1      cpx #3
F912 F0 1C      282      beq >5
F914            283      ;
F914 06 2E      284      ^2      asl FORMAT
F916 90 0E      285      bcc >3
F918            286      ;
F918 BD 2F FA    287      lda CHAR1-1,X
F91B            288      ;
F91B 20 ED FD    289      jsr COUT
F91E            290      ;
F91E BD 35 FA    291      lda CHAR2-1,X
F921 F0 03      292      beq >3
F923            293      ;
F923 20 ED FD    294      jsr COUT
F926            295      ;
F926 CA          296      ^3      dex
F927 D0 E7      297      bne <1
F929            298      ;
F929 60          299      rts
F92A            300      ;
F92A            301      ;
F92A 88          302      ^4      dey
F92B 30 E7      303      bmi <2
F92D            304      ;

```

```

F92D 20 DA FD      305      jsr PRBYTE
F930                306      ;
F930 A5 2E        307      ^5      lda FORMAT
F932 C9 E8        308      cmp #$E8      ; relative address mode
F934                309      ;
F934 B1 3A        310      lda (PCL),Y      ; print target, not offset
F936                311      ;
F936 90 F2        312      bcc <4
F938                313      ;
F938 20 56 F9     314      jsr PCADJ3
F93B                315      ;
F93B AA          316      tax
F93C                317      ;
F93C E8          318      inx
F93D D0 01       319      bne PRNTYX
F93F                320      ;
F93F C8          321      iny
F940                322      ;
F940 98          323      PRNTYX      tya
F941                324      ;
F941 20 DA FD     325      PRNTAX      jsr PRBYTE
F944                326      ;
F944 8A          327      PRNTAX      txa
F945                328      ;
F945 4C DA FD     329      jmp PRBYTE
F948                330      ;
F948                331      ;
F948 A2 03       332      PRBLNK      ldx #3      ; blank count
F94A                333      ;
F94A A9 A0       334      PRBL2      lda #SPACE
F94C 20 ED FD     335      jsr COUT
F94F                336      ;
F94F CA          337      dex
F950 D0 F8       338      bne PRBL2
F952                339      ;
F952 60          340      rts
F953                341      ;
F953                342      ;
F953                343      ; A-reg = PCL + LENGTH + 1, carry into Y-reg (PCH).
F953                344      ;
F953 38          345      PCADJ      sec
F954                346      ;
F954 A5 2F       347      PCADJ2      lda LENGTH
F956                348      ;
F956 A4 3B       349      PCADJ3      ldY PCH
F958                350      ;
F958 AA          351      tax
F959 10 01       352      bpl >6
F95B                353      ;
F95B 88          354      dey
F95C                355      ;
F95C 65 3A       356      ^6      adc PCL
F95E 90 01       357      bcc RTS2
F960                358      ;
F960 C8          359      iny
F961                360      ;
F961 60          361      RTS2      rts
F962                362      ;
F962                363      ;
F962                364      ; Opcodes of the form XXXX XXY0:
F962                365      ;

```



```

F962      366 ; Use XXXXXX for index into FMT1.
F962      367 ;
F962      368 ; If Y=0, use lower nibble for index into FMT2.
F962      369 ; If Y=1, use upper nibble for index into FMT2.
F962      370 ;
F962      371 FMT1:
F962 04 22 54 372      hex 04225433ED824493
F965 33 ED 82
F968 44 93
F96A 03 22 54 373      hex 03225433ED884499
F96D 33 ED 88
F970 44 99
F972 04 20 54 374      hex 04205433ED804490
F975 33 ED 80
F978 44 90
F97A 04 22 54 375      hex 0422543BED88449F
F97D 3B ED 88
F980 44 9F
F982 0D 22 44 376      hex 0D224433EDC84493
F985 33 ED C8
F988 44 93
F98A 11 22 44 377      hex 11224433EDC844A9
F98D 33 ED C8
F990 44 A9
F992 01 22 44 378      hex 01224433ED804490
F995 33 ED 80
F998 44 90
F99A 01 22 44 379      hex 01224433ED804490
F99D 33 ED 80
F9A0 44 90
F9A2      380 ;
F9A2      381 ;
F9A2      382 ; Opcodes of the form ZZXX XY01:
F9A2      383 ;
F9A2 26 31 87 384      hex 2631879A
F9A5 9A
F9A6      385 ;
F9A6      386 ;
F9A6      387 ; Mnemonic extraction from MNEML and MNEMR:
F9A6      388 ;
F9A6      389 ; Table: <MNEML-><MNEMR->
F9A6      390 ; Bit: 7654321076543210
F9A6      391 ; -----
F9A6      392 ; Chr1 Chr2 Chr3
F9A6      393 ;
F9A6      394 ; Mnemonic = Chr1+0xBF Chr2+0xBF Chr3+0xBF
F9A6      395 ;
F9A6      396 MNEML:
F9A6 1C 8A 1C 397      hex 1C8A1C235D8B1BA1
F9A9 23 5D 8B
F9AC 1B A1
F9AE 9D 8A 1D 398      hex 9D8A1D239D8B1DA1
F9B1 23 9D 8B
F9B4 1D A1
F9B6 1C 29 19 399      hex 1C2919AE69A81923
F9B9 AE 69 A8
F9BC 19 23
F9BE 24 53 1B 400      hex 24531B23245319A1
F9C1 23 24 53
F9C4 19 A1
F9C6 AD 1A 5B 401      hex AD1A5B5BA5692424

```

```

F9C9 5B A5 69
F9CC 24 24
F9CE AE AE A8    402          hex AEAEA8AD298A7C8B
F9D1 AD 29 8A
F9D4 7C 8B
F9D6 15 9C 6D    403          hex 159C6D9CA5692953
F9D9 9C A5 69
F9DC 29 53
F9DE 84 13 34    404          hex 84133411A56923A0
F9E1 11 A5 69
F9E4 23 A0
F9E6 00 8A 8B    405          hex 008A8BACA5
F9E9 AC A5
F9EB              406          ;
F9EB              407          MNEMR:
F9EB D8 62 5A    408          hex D8625A4826629488
F9EE 48 26 62
F9F1 94 88
F9F3 54 44 C8    409          hex 5444C8546844E894
F9F6 54 68 44
F9F9 E8 94
F9FB C4 B4 08    410          hex C4B4088474B4286E
F9FE 84 74 B4
FA01 28 6E
FA03 74 F4 CC    411          hex 74F4CC4A72F2A48A
FA06 4A 72 F2
FA09 A4 8A
FA0B 06 AA A2    412          hex 06AAA2A274747472
FA0E A2 74 74
FA11 74 72
FA13 44 68 B2    413          hex 4468B232B2722272
FA16 32 B2 72
FA19 22 72
FA1B 1A 1A 26    414          hex 1A1A2626727288C8
FA1E 26 72 72
FA21 88 C8
FA23 C4 CA 26    415          hex C4CA26484444A2C8
FA26 48 44 44
FA29 A2 C8
FA2B 00 74 74    416          hex 007474C676
FA2E C6 76
FA30              417          ;
FA30              418          ;
FA30              419          ; Characters used to disassemble instructions.
FA30              420          ;
FA30 AC A9 AC    421          CHAR1    hex ACA9ACA3A8A4    ; ,),#($
FA33 A3 A8 A4
FA36 D9 00 D8    422          CHAR2    hex D900D8A4A400    ; Y X$$
FA39 A4 A4 00
FA3C              423          ;
FA3C              424          ;
FA3C 8D 06 C0    425          CXOFF    sta CXROMOFF    ; enable slots
FA3F              426          ;
FA3F 60          427          CXRTN    rts
FA40              428          ;
FA40              429          ;
FA40              430          ; This code is useless, and probably only supports the old
FA40              431          ; BREAK vector at 0xFA40. It assumes that RDCXROM is off.
FA40              432          ;
FA40 85 45        433          OLDIRQ    sta AREG
FA42 A5 45        434          lda AREG

```

```

FA44      435 ;
FA44 4C FA C3 436      jmp IRQRTN
FA47      437 ;
FA47      438 ;
FA47 8D 06 C0 439 NEWBREAK sta CXROMOFF      ; enable slots
FA4A      440 ;
FA4A 85 45      441      sta AREG
FA4C      442 ;
FA4C 28      443 BREAK      plp
FA4D      444 ;
FA4D 20 4C FF 445      jsr SAVE2      ; save registers
FA50      446 ;
FA50 68      447      pla
FA51 85 3A      448      sta PCL
FA53      449 ;
FA53 68      450      pla
FA54 85 3B      451      sta PCH
FA56      452 ;
FA56 6C F0 03 453      jmp (AUTOBRK+1)
FA59      454 ;
FA59      455 ;
FA59 20 82 F8 456 OLDBRK      jsr INSDS1      ; print PC
FA5C 20 DA FA 457      jsr REGDSP2      ; print registers
FA5F      458 ;
FA5F 4C 65 FF 459      jmp MON      ; enter Monitor
FA62      460 ;
FA62      461 ;
FA62 AD 58 C0 462 RESET      lda ANN1OFF
FA65 AD 5A C0 463      lda ANN2OFF
FA68      464 ;
FA68 AD 5D C0 465      lda ANN3ON
FA6B AD 5F C0 466      lda ANN4ON
FA6E      467 ;
FA6E A0 09      468      ldy #9      ; //e init
FA70 D0 0C      469      bne RESET2      ; always taken
FA72      470 ;
FA72      471 ;
FA72 8D 07 C0 472 SWEET16      sta CXROMON      ; enable CX ROM
FA75      473 ;
FA75 4C 70 C6 474      jmp SW16
FA78      475 ;
FA78 8D 06 C0 476 SW16RTN      sta CXROMOFF      ; enable slots
FA7B      477 ;
FA7B 6C 1E 00 478      jmp (R15L)
FA7E      479 ;
FA7E      480 ;
FA7E 20 B4 FB 481 RESET2      jsr GOTOROM
FA81      482 ;
FA81      483 ;
FA81 D8      484 NEWMON      cld
FA82      485 ;
FA82 20 3A FF 486      jsr BELL
FA85      487 ;
FA85 AD F3 03 488      lda AUTORSET+1      ; check for power up
FA88 49 A5      489      eor #PWRUPBYT
FA8A      490 ;
FA8A CD F4 03 491      cmp PWRSTATE
FA8D D0 17      492      bne PWRUP
FA8F      493 ;
FA8F AD F2 03 494      lda AUTORSET      ; BASIC coldstart?
FA92 D0 0F      495      bne >1

```

```

FA94          496 ;
FA94 A9 E0    497     lda /BASIC
FA96 CD F3 03 498     cmp AUTORSET+1
FA99 D0 08    499     bne >1
FA9B          500 ;
FA9B A0 03    501 NEWMON2  ldy #BASIC2
FA9D 8C F2 03 502     sty AUTORSET
FAA0          503 ;
FAA0 4C 00 E0 504     jmp BASIC
FAA3          505 ;
FAA3 6C F2 03 506     ^1     jmp (AUTORSET)
FAA6          507 ;
FAA6          508 ;
FAA6          509 ; Setup PAGE03 vectors.
FAA6          510 ;
FAA6 20 60 FB 511 PWRUP    jsr APPLE2
FAA9          512 ;
FAA9 A2 05    513     ldx #PWRCNLEN
FAAB          514 ;
FAAB BD FC FA 515     ^1     lda PWRCON-1,X
FAAE 9D EF 03 516     sta AUTOBRK,X
FAB1          517 ;
FAB1 CA       518     dex
FAB2 D0 F7    519     bne <1
FAB4          520 ;
FAB4 A9 C8    521     lda /PAGEC8           ; high slot + 1
FAB6          522 ;
FAB6 86 00    523     stx LOC0           ; X-reg = 0
FAB8 85 01    524     sta LOC1
FABA          525 ;
FABA          526 ;
FABA          527 ; Check 3 ID bytes, not 4, to find all bootable devices.
FABA          528 ;
FABA A0 03    529     ^2     ldy #DISKIDLN-3
FABC          530 ;
FABC C6 01    531     dec LOC1
FABE          532 ;
FABE A5 01    533     lda LOC1
FAC0 8D F8 07 534     sta MSLOT
FAC3          535 ;
FAC3 C9 C0    536     cmp /PAGEC0           ; last slot?
FAC5 F0 D4    537     beq NEWMON2
FAC7          538 ;
FAC7 B1 00    539     ^3     lda (LOC0),Y
FAC9 D9 02 FB 540     cmp DISKID,Y
FACC D0 EC    541     bne <2
FACE          542 ;
FACE 88       543     dey
FACF 88       544     dey
FAD0          545 ;
FAD0 10 F5    546     bpl <3
FAD2          547 ;
FAD2 6C 00 00 548     jmp (LOC0)
FAD5          549 ;
FAD5          550 ;
FAD5          551     dfs 2,ZERO           ; 2 bytes
FAD7          552 ;
FAD7          553 ;
FAD7          554 ; Display register contents with labels.
FAD7          555 ;
FAD7 20 8E FD 556 REGDSP  jsr CROUT

```

```

FADA          557 ;
FADA A9 45     558 REGDSP2  lda #AREG
FADC 85 40     559          sta A3L
FADE          560 ;
FADE A9 00     561          lda /AREG
FAE0 85 41     562          sta A3H
FAE2          563 ;
FAE2 A2 FB     564          ldx #$100-REGTBLEN
FAE4          565 ;
FAE4 A9 A0     566 ^1      lda #SPACE
FAE6 20 ED FD  567          jsr COUT
FAE9          568 ;
FAE9 BD 1E FA  569          lda REGTBL-$100-REGTBLEN,X
FAEC 20 ED FD  570          jsr COUT
FAEF          571 ;
FAEF A9 BD     572          lda #""
FAF1 20 ED FD  573          jsr COUT
FAF4          574 ;
FAF4 B5 4A     575          lda AREG+5,X          ; REGTBLEN
FAF6 20 DA FD  576          jsr PRBYTE
FAF9          577 ;
FAF9 E8        578          inx
FAFA 30 E8     579          bmi <1
FAFC          580 ;
FAFC 60        581          rts
FAFD          582 ;
FAFD          583 ;
FAFD 59 FA     584 PWRCON   adr OLDBRK
FAFF 00 E0     585          adr BASIC
FB01 45        586          byt PWRUPBYT^BASIC/PAGESIZE
FB02          587 ;
0005          588 PWRCNLEN equ *-PWRCON
FB02          589 ;
FB02          590 ;
FB02          591 ; Only 3 of 4 ID bytes are verified.
FB02          592 ;
FB02 FF 20     593 DISKID   hex FF20
FB04 FF 00     594          hex FF00
FB06 FF 03     595          hex FF03
FB08          596          hex FF3C          ; not verified
FB08          597 ;
0006          598 DISKIDLN equ *-DISKID
FB08          599 ;
FB08          600 ;
FB08 20 84 FE  601 RSETINIT jsr SETNORM
FB0B 20 2F FB  602          jsr INIT
FB0E 20 93 FE  603          jsr SETVID
FB11          604 ;
FB11 4C 89 FE  605          jmp SETKBD
FB14          606 ;
FB14          607 ;
FB14          608 ; Translate table for IJKLM to DBALC.
FB14          609 ;
FB14 C4 C2 C1  610 XLATBL   asc "DBALC"
FB17 CC C3
FB19          611 ;
FB19          612 ;
FB19          613 ; Display register table.
FB19          614 ;
FB19 C1 D8 D9  615 REGTBL   asc "AXYPS"
FB1C D0 D3

```

```

FB1E      616 ;
0005      617 REGTBLEN equ *-REGTBL
FB1E      618 ;
FB1E      619 ;
FB1E      620 ; Paddle read routine.  Count Y-reg every 11 usec.
FB1E      621 ;
FB1E AD 70 C0 622 PREAD      lda GCTOGL
FB21      623 ;
FB21 A0 00    624          ldy #ZERO
FB23      625 ;
FB23 EA      626          nop                ; compensate for 1st count
FB24 EA      627          nop
FB25      628 ;
FB25 BD 64 C0 629 ^1      lda GC1IN,X
FB28 10 04    630          bpl >2
FB2A      631 ;
FB2A C8      632          iny
FB2B D0 F8    633          bne <1
FB2D      634 ;
FB2D 88      635          dey
FB2E      636 ;
FB2E 60      637 ^2      rts
FB2F      638 ;
FB2F      639 ;
FB2F      640 ; Initialize status, text mode, graphics mode, and window.
FB2F      641 ;
FB2F A9 00    642 INIT      lda #ZERO
FB31 85 48    643          sta PREG
FB33      644 ;
FB33 AD 56 C0 645 INIT2     lda HIRESOFF
FB36 AD 54 C0 646          lda PAGE1ON
FB39      647 ;
FB39 AD 51 C0 648 SETEXT    lda TEXTON
FB3C      649 ;
FB3C A9 00    650          lda #ZERO
FB3E F0 0B    651          beq SETWND                ; always taken
FB40      652 ;
FB40 AD 50 C0 653 SETGR     lda TEXTOFF
FB43 AD 53 C0 654          lda MIXEDON
FB46      655 ;
FB46 20 36 F8 656          jsr CLRTOP
FB49      657 ;
FB49 A9 14    658          lda #20
FB4B      659 ;
FB4B 85 22    660 SETWND    sta WNDTOP
FB4D      661 ;
FB4D A9 00    662          lda #ZERO
FB4F 85 20    663          sta WNDLFT
FB51      664 ;
FB51 A0 0C    665          ldy #12
FB53 D0 5F    666          bne GOTOROM                ; always taken
FB55      667 ;
FB55      668 ;
FB55      669          dfs 6,ZERO                ; 6 bytes
FB5B      670 ;
FB5B      671 ;
FB5B 85 25    672 TABV     sta CV
FB5D      673 ;
FB5D 4C 22 FC 674          jmp VTAB
FB60      675 ;
FB60      676 ;

```

```

FB60      677 ; TITLE was moved and is 1 byte longer.
FB60      678 ;
FB60 20 58 FC 679 APPLE2    jsr HOME
FB63      680 ;
FB63 A0 0A    681          ldy #TITLEN
FB65      682 ;
FB65 B9 90 F7 683 STITLE    lda TITLE-1,Y
FB68 99 0E 04 684          sta TEXTPG1+OFFTITL,Y
FB6B      685 ;
FB6B 88      686          dey
FB6C D0 F7    687          bne STITLE
FB6E      688 ;
FB6E 60      689          rts
FB6F      690 ;
FB6F      691 ;
FB6F      692 ; Routine to calculate power-up byte from RESET vector.
FB6F      693 ; Not referenced.
FB6F      694 ;
FB6F AD F3 03 695 SETPWRUP  lda AUTORSET+1
FB72 49 A5    696          eor #PWRUPBYT
FB74 8D F4 03 697          sta PWRSTATE
FB77      698 ;
FB77 60      699          rts
FB78      700 ;
FB78      701 ;
FB78 C9 8D    702 VIDWAIT   cmp #RETURN
FB7A D0 18    703          bne >1
FB7C      704 ;
FB7C AC 00 C0 705          ldy KEY
FB7F 10 13    706          bpl >1
FB81      707 ;
FB81 C0 93    708          cpy #CTRLS          ; ctrl-S pressed?
FB83 D0 0F    709          bne >1
FB85      710 ;
FB85 2C 10 C0 711          bit CLRKEY
FB88      712 ;
FB88      713 ;
FB88      714 ; Nothing is done with CTRLC.
FB88      715 ;
FB88 AC 00 C0 716 KBDWAIT   ldy KEY
FB8B 10 FB    717          bpl KBDWAIT
FB8D      718 ;
FB8D C0 83    719          cpy #CTRLC          ; ctrl-C pressed?
FB8F F0 03    720          beq >1
FB91      721 ;
FB91 2C 10 C0 722          bit CLRKEY
FB94      723 ;
FB94 4C FD FB 724 ^1        jmp VIDOUT
FB97      725 ;
FB97      726 ;
FB97 38      727 ESCOLD    sec          ; insure carry set
FB98      728 ;
FB98 4C 2C FC 729          jmp ESCOLD2
FB9B      730 ;
FB9B      731 ;
FB9B A8      732 ^1        tay          ; character is index
FB9C      733 ;
FB9C B9 4B FA 734          lda XLATBL-"I",Y    ; IJKLM -> CBALD
FB9F      735 ;
FB9F 20 97 FB 736          jsr ESCOLD
FBA2 20 21 FD 737          jsr RDESC

```

```

FBA5          738 ;
FBA5          739 ;
FBA5 C9 CE    740 ESCNEW    cmp #"N"
FBA7 B0 EE    741          bcs ESCOLD          ; exclude >M
FBA9          742 ;
FBA9 C9 C9    743          cmp #"I"
FBAB 90 EA    744          bcc ESCOLD          ; exclude <I
FBAD          745 ;
FBAD C9 CC    746          cmp #"L"
FBAF F0 E6    747          beq ESCOLD          ; exclude L
FBB1          748 ;
FBB1 D0 E8    749          bne <1              ; always taken
FBB3          750 ;
FBB3          751 ;
FBB3 06       752 SIGROM    hex 06              ; ROM ID byte
FBB4          753 ;
FBB4          754 ;
FBB4          755 ; Save state of CX ROM.  If +, return via CXOFF and turn CX
FBB4          756 ; ROM off.  If -, return via CXRTN and leave CX ROM on.
FBB4          757 ;
FBB4 2C 15 C0 758 GOTOROM  bit RDCXROM          ; get CX ROM state
FBB7          759 ;
FBB7 08       760          php                  ; save state
FBB8          761 ;
FBB8 8D 07 C0 762          sta CXROMON          ; enable CX ROM
FBBB          763 ;
FBBB 4C 00 C1 764          jmp DOCXCMD
FBBE          765 ;
FBBE          766 ;
FBBE          767          dfs 2,ZERO          ; 2 bytes
FBC0          768 ;
FBC0          769 ;
FBC0          770 ; Signature byte.  Original Apple //e uses 0xEA.
FBC0          771 ; Enhanced Apple //e uses 0xE0.
FBC0          772 ;
FBC0 E0       773 SIGBYTE  hex E0              ; //e ROM ID byte
FBC1          774 ;
FBC1          775 ;
FBC1          776 ; The same routine is at 0xCABA, so use that routine.
FBC1          777 ;
FBC1 84 28    778 BASCALC  sty BASL
FBC3          779 ;
FBC3 A0 02    780          ldy #2
FBC5 D0 ED    781          bne GOTOROM          ; always taken
FBC7          782 ;
FBC7          783 ;
FBC7          784 ; FMT2 bits: 6543 21LL
FBC7          785 ;
FBC7          786 ; LL sets value/address length:
FBC7          787 ;
FBC7          788 ; 0 - no extra bytes (1 byte)
FBC7          789 ; 1 - value or zpage (2 bytes)
FBC7          790 ; 2 - absolute address (3 bytes)
FBC7          791 ; 3 - 1 byte relative branch for abs. addr. (2 bytes)
FBC7          792 ;
FBC7          793 ; Xreg index into CHAR1/CHAR2 tables for bit set:
FBC7          794 ;
FBC7          795 ; 6 - $ |
FBC7          796 ; 5 - ($ } left of value/address
FBC7          797 ; 4 - #$ |
FBC7          798 ;

```



```

FBC7          799 ;      3 - ,X |
FBC7          800 ;      2 - )   } right of value/address
FBC7          801 ;      1 - ,Y |
FBC7          802 ;
FBC7          803 ;
FBC7          804 FMT2:
FBC7 00        805      byt %00000000      ; error
FBC8 21        806      byt %00100001      ; immediate; 1 byte
FBC9 81        807      byt %10000001      ; zpage; 1 byte
FBCA 82        808      byt %10000010      ; absolute; 2 bytes
FBCB 00        809      byt %00000000      ; implied; 0 bytes
FBCC 00        810      byt %00000000      ; accumulator; 0 bytes
FBCD 59        811      byt %01011001      ; (zpage,X); 1 byte
FBCE 4D        812      byt %01001101      ; (zpage),Y; 1 byte
FBCF 91        813      byt %10010001      ; zpage,X; 1 byte
FBD0 92        814      byt %10010010      ; absolute,X; 2 bytes
FBD1 86        815      byt %10000110      ; absolute,Y; 2 bytes
FBD2 4A        816      byt %01001010      ; (absolute); 2 bytes
FBD3 85        817      byt %10000101      ; zpage,Y; 1 byte
FBD4 9D        818      byt %10011101      ; relative; 1 byte
FBD5 49        819      byt %01001001      ; (zpage); 1 byte (was 0x4B)
FBD6 5A        820      byt %01011010      ; (absolute,X); 2 bytes
FBD7          821 ;
FBD7          822 ;
FBD7          823      dfs 2,ZERO          ; 2 bytes
FBD9          824 ;
FBD9          825 ;
FBD9 C9 87     826 BELL2      cmp #ASCIBELL      ; ^G
FBDB D0 12     827          bne >2
FBDD          828 ;
FBDD A9 40     829 RINGBELL   lda #$40          ; delay 0.01 seconds
FBDF 20 A8 FC  830          jsr WAIT
FBE2          831 ;
FBE2 A0 C0     832          ldy #$C0
FBE4          833 ;
FBE4 A9 0C     834 ^1      lda #12          ; 1 KHz for 1 second
FBE6 20 A8 FC  835          jsr WAIT
FBE9          836 ;
FBE9 AD 30 C0  837          lda SPKRTOGL
FBEC          838 ;
FBEC 88        839          dey
FBED D0 F5     840          bne <1
FBEF          841 ;
FBEF 60        842 ^2      rts
FBF0          843 ;
FBF0          844 ;
FBF0          845 ; Store character on screen and advance cursor.
FBF0          846 ;
FBF0 A4 24     847 STORADV   ldy CH
FBF2          848 ;
FBF2 91 28     849          sta (BASL),Y
FBF4          850 ;
FBF4 E6 24     851 ADVANCE   inc CH
FBF6          852 ;
FBF6 A5 24     853          lda CH
FBF8 C5 21     854          cmp WNDWDTH
FBFA B0 66     855          bcs CR
FBFC          856 ;
FBFC 60        857 RTN.FB.F rts
FBFD          858 ;
FBFD          859 ;

```

FBFD 860 icl "FC.L"

LLOAD FC.L,A\$4000

```

FBFD      1          ttl "ROM Source Code, FC.L"
FBFD      2      ;
FBFD      3      ;
FBFD      4      ; FC.L
FBFD      5      ;
FBFD      6      ;
FBFD      7      ; Check for NORMAL, INVERSE, and control characters.
FBFD      8      ;
FBFD C9 A0      9  VIDOUT    cmp #SPACE
FBFF B0 EF     10          bcs STORADV
FC01      11      ;
FC01 A8       12          tay
FC02 10 EC     13          bpl STORADV
FC04      14      ;
FC04 C9 8D     15          cmp #RETURN
FC06 F0 5A     16          beq CR
FC08      17      ;
FC08 C9 8A     18          cmp #DARROW
FC0A F0 5A     19          beq LF
FC0C      20      ;
FC0C C9 88     21          cmp #LARROW
FC0E D0 C9     22          bne BELL2
FC10      23      ;
FC10 C6 24     24  BS      dec CH
FC12 10 E8     25          bpl RTN.FB.F
FC14      26      ;
FC14 A5 21     27          lda WNDWDTH
FC16 85 24     28          sta CH
FC18      29      ;
FC18 C6 24     30          dec CH
FC1A      31      ;
FC1A A5 22     32  UP      lda WNDTOP
FC1C C5 25     33          cmp CV
FC1E B0 DC     34          bcs RTN.FB.F
FC20      35      ;
FC20 C6 25     36          dec CV
FC22      37      ;
FC22      38      ;
FC22 A5 25     39  VTAB    lda CV
FC24      40      ;
FC24 84 28     41  VTAB2   sty BASL
FC26      42      ;
FC26 A0 04     43          ldy #4
FC28 D0 8A     44          bne GOTOROM          ; always taken
FC2A      45      ;
FC2A      46      ;
FC2A      47          dfs 2,ZERO          ; 2 bytes
FC2C      48      ;
FC2C      49      ;
FC2C 49 C0     50  ESCOLD2  eor #"@"          ; check esc-@
FC2E F0 28     51          beq HOME
FC30      52      ;
FC30 69 FD     53          adc #!-3
FC32 90 C0     54          bcc ADVANCE          ; check esc-A
FC34      55      ;
FC34 F0 DA     56          beq BS          ; check esc-B
FC36      57      ;
FC36 69 FD     58          adc #!-3
FC38 90 2C     59          bcc LF          ; check esc-C
FC3A      60      ;

```

```

FC3A F0 DE      61          beq UP                ; check esc-D
FC3C            62          ;
FC3C 69 FD      63          adc #!-3
FC3E 90 5C      64          bcc CLREOL            ; check esc-E
FC40            65          ;
FC40 D0 BA      66          bne RTN.FB.F          ; not esc-F
FC42            67          ;
FC42 A0 0A      68  CLREOP   ldy #10              ; handle esc-F
FC44 D0 14      69          bne GOTOROM2         ; always taken
FC46            70          ;
FC46            71          ;
FC46            72          ; New VIDWAIT to handle 40 and 80 columns.
FC46            73          ;
FC46 2C 1F C0   74  NEWVW    bit RDVID80          ; 80 column enabled?
FC49 10 04      75          bpl >1
FC4B            76          ;
FC4B A0 00      77          ldy #ZERO            ; print 80 column
FC4D F0 0B      78          beq GOTOROM2         ; always taken
FC4F            79          ;
FC4F            80          ;
FC4F            81          ; Print 40 column.
FC4F            82          ;
FC4F 98         83          ^1      tya
FC50 48         84          pha
FC51            85          ;
FC51 20 78 FB   86          jsr VIDWAIT
FC54            87          ;
FC54 68         88          pla
FC55            89          ;
FC55 A4 35      90          ldy YSAV1
FC57            91          ;
FC57 60         92          rts
FC58            93          ;
FC58            94          ;
FC58 A0 05      95  HOME     ldy #5
FC5A            96          ;
FC5A 4C B4 FB   97  GOTOROM2 jmp GOTOROM
FC5D            98          ;
FC5D            99          ;
FC5D            100         dfs 2,ZERO            ; 2 bytes
FC5F            101         ;
FC5F            102         ;
FC5F            103         ; Return from BRK instruction in STEP/TRACE.
FC5F            104         ;
FC5F 4C 59 FA   105  STEPRTN2 jmp OLDBRK
FC62            106         ;
FC62            107         ;
FC62 A9 00      108  CR       lda #ZERO
FC64 85 24      109          sta CH
FC66            110         ;
FC66 E6 25      111  LF       inc CV
FC68            112         ;
FC68 A5 25      113          lda CV
FC6A C5 23      114          cmp WNDBTM
FC6C 90 B6      115          bcc VTAB2
FC6E            116         ;
FC6E C6 25      117          dec CV
FC70            118         ;
FC70 A0 06      119  SCROLL   ldy #6
FC72 D0 E6      120          bne GOTOROM2         ; always taken
FC74            121         ;

```

```

FC74      122 ;
FC74 8D 06 C0 123 GOTOIRQ sta CXROMOFF ; enable slots
FC77      124 ;
FC77 6C FE 03 125 jmp (MASKIRQ)
FC7A      126 ;
FC7A      127 ;
FC7A      128 ; After an interrupt, IRQDONE (0xC3F4) jumps here because
FC7A      129 ; this code cannot execute in CX ROM space.
FC7A      130 ;
FC7A 68      131 IRQDONE2 pla
FC7B 8D F8 07 132 sta MSLOT
FC7E      133 ;
FC7E C9 C1    134 cmp /PAGEC1
FC80 90 0D    135 bcc >1
FC82      136 ;
FC82 8D FF CF 137 sta CLRROM
FC85      138 ;
FC85 A0 00    139 ldy #ZERO
FC87 A6 01    140 ldx LOC1
FC89      141 ;
FC89 85 01    142 sta LOC1
FC8B      143 ;
FC8B B1 00    144 lda (LOC0),Y
FC8D      145 ;
FC8D 86 01    146 stx LOC1
FC8F      147 ;
FC8F 8D 07 C0 148 ^1 sta CXROMON ; enable CX ROM
FC92      149 ;
FC92 4C 7C C4 150 jmp NEWIRQ ; restore the machine
FC95      151 ;
FC95      152 ;
FC95 90 02    153 CHKINV bcc >1
FC97      154 ;
FC97 25 32    155 and INVFLG
FC99      156 ;
FC99 4C F7 FD 157 ^1 jmp COUTZ2
FC9C      158 ;
FC9C      159 ;
FC9C 38      160 CLREOL sec
FC9D      161 ;
FC9D 90 00    162 bcc *+2
FC9F      163 dfs !-1
FC9E      164 ;
FC9E 18      165 CLREOL2 clc
FC9F      166 ;
FC9F 84 2A    167 sty BAS2L
FCA1      168 ;
FCA1 A0 07    169 ldy #7
FCA3      170 ;
FCA3 B0 B5    171 bcs GOTOROM2
FCA5      172 ;
FCA5 C8      173 iny
FCA6 D0 B2    174 bne GOTOROM2 ; always taken
FCA8      175 ;
FCA8      176 ;
FCA8      177 ; A-reg = delay value. Clock is 1,020,484 cycles/second.
FCA8      178 ;
FCA8      179 ; Delay = 2.5*A*A + 13.5*A + 13 cycles.
FCA8      180 ;
FCA8      181 ; A = ( Delay / 2.5 + 2.09 ) ^ 0.5 - 2.7.
FCA8      182 ;

```

```

FCA8 38          183 WAIT      sec
FCA9            184 ;
FCA9 48          185 ^1      pha
FCAA            186 ;
FCAA E9 01       187 ^2      sbc #1
FCAC D0 FC       188          bne <2
FCAE            189 ;
FCAE 68          190          pla
FCAF            191 ;
FCAF E9 01       192          sbc #1
FCB1 D0 F6       193          bne <1
FCB3            194 ;
FCB3 60          195          rts
FCB4            196 ;
FCB4            197 ;
FCB4            198 ; Increment A4, compare A1 to A2, increment A1.
FCB4            199 ;
FCB4 E6 42       200 NEXTA4   inc A4L
FCB6 D0 02       201          bne NEXTA1
FCB8            202 ;
FCB8 E6 43       203          inc A4H
FCBA            204 ;
FCBA A5 3C       205 NEXTA1   lda A1L
FCBC C5 3E       206          cmp A2L
FCBE            207 ;
FCBE A5 3D       208          lda A1H
FCC0 E5 3F       209          sbc A2H
FCC2            210 ;
FCC2 E6 3C       211          inc A1L
FCC4 D0 02       212          bne >1
FCC6            213 ;
FCC6 E6 3D       214          inc A1H
FCC8            215 ;
FCC8 60          216 ^1      rts
FCC9            217 ;
FCC9            218 ;
FCC9 60          219 HEADR    rts                ; cassette HEADR removed
FCCA            220 ;
FCCA            221 ;
FCCA            222 ; Return from STEP/TRACE.  If carry set, then BRK.
FCCA            223 ;
FCCA 8D 06 C0    224 STEPRTN  sta CXROMOFF
FCCD            225 ;
FCCD B0 90       226          bcs STEPRTN2
FCCF            227 ;
FCCF 4C 73 FF    228          jmp NEXTITM
FCD2            229 ;
FCD2            230 ;
FCD2 8D 06 C0    231 XERR     sta CXROMOFF                ; enable slots
FCD5            232 ;
FCD5 20 4A F9    233          jsr PRBL2                ; tab to error
FCD8            234 ;
FCD8 A9 DE       235          lda #"^"
FCDA 20 ED FD    236          jsr COUT
FCDD            237 ;
FCDD 20 3A FF    238          jsr BELL
FCE0            239 ;
FCE0 4C F0 FC    240          jmp NXTLINE
FCE3            241 ;
FCE3            242 ;
FCE3            243 ; End of Mini-Assembler instruction processing.

```

```

FCE3      244 ;
FCE3 8D 06 C0 245 FINDOP      sta CXROMOFF      ; enable slots
FCE6      246 ;
FCE6 20 D0 F8 247             jsr INSTDSP
FCE9 20 53 F9 248             jsr PCADJ
FCEC      249 ;
FCEC 84 3B    250             sty PCH
FCEE 85 3A    251             sta PCL
FCF0      252 ;
FCF0      253 ;
FCF0      254 ; Start of Mini-Assembler instruction processing.
FCF0      255 ;
FCF0 A9 A1    256 NXTLINE     lda #"!"
FCF2 85 33    257             sta PROMPT
FCF4      258 ;
FCF4 20 67 FD 259             jsr GETLINE2
FCF7      260 ;
FCF7 8D 07 C0 261             sta CXROMON      ; enable CX ROM
FCFA      262 ;
FCFA 4C 9C CF 263             jmp NXTLINE2
FCFD      264 ;
FCFD      265 ;
FCFD      266 ; Change lowercase to uppercase.
FCFD      267 ;
FCFD B9 00 02 268 UPRMON      lda INPUT,Y
FD00      269 ;
FD00 C8       270             iny
FD01      271 ;
FD01 C9 E1    272 UPRCASE     cmp #"a"
FD03 90 06    273             bcc >1
FD05      274 ;
FD05 C9 FB    275             cmp #"z"+1
FD07 B0 02    276             bcs >1
FD09      277 ;
FD09 29 DF    278             and #LWRMASK
FD0B      279 ;
FD0B 60       280 ^1         rts
FD0C      281 ;
FD0C      282 ;
FD0C 4C 13 FD 283 RDKEY      jmp RDKEY2
FD0F      284 ;
FD0F      285 ;
FD0F      286             dfs 1,ZERO      ; 1 byte
FD10      287 ;
FD10      288 ;
FD10      289 ; Preserve this entry point, but do nothing.
FD10      290 ;
FD10 6C 38 00 291             jmp (KSWL)
FD13      292 ;
FD13      293 ;
FD13 A0 0B    294 RDKEY2     ldy #11
FD15 20 B4 FB 295             jsr GOTOROM
FD18      296 ;
FD18 6C 38 00 297 RDKEY3     jmp (KSWL)
FD1B      298 ;
FD1B      299 ;
FD1B A0 03    300 KEYIN      ldy #3
FD1D      301 ;
FD1D 4C B4 FB 302 GOTOROM4  jmp GOTOROM
FD20      303 ;
FD20      304 ;

```

```

FD20          305          dfs 1,ZERO          ; 1 byte
FD21          306          ;
FD21          307          ;
FD21 20 13 FD 308 RDESC      jsr RDKEY2          ; get a key
FD24          309          ;
FD24 A0 01     310          ldy #1
FD26 D0 F5     311          bne GOTOROM4        ; always taken
FD28          312          ;
FD28          313          ;
FD28          314          ; All slot cards must save their slot number in MSLOT and
FD28          315          ; set its MSB.  If the MSB of MSLOT is clear, that is a
FD28          316          ; flag to the video firmware that escapes are allowed.
FD28          317          ;
FD28 4E F8 07 318 NEWRDKEY lsr MSLOT            ; allow escapes
FD2B          319          ;
FD2B 4C 13 FD 320          jmp RDKEY2          ; now read the key
FD2E          321          ;
FD2E          322          ;
FD2E          323          dfs 1,ZERO          ; 1 byte
FD2F          324          ;
FD2F          325          ;
FD2F 20 21 FD 326 ESC        jsr RDESC          ; remap keys
FD32 20 A5 FB 327          jsr ESCNEW          ; do ESC function
FD35          328          ;
FD35          329          ;
FD35 20 28 FD 330 RDCHAR     jsr NEWRDKEY
FD38          331          ;
FD38 C9 9B     332          cmp #ESCAPE
FD3A F0 F3     333          beq ESC
FD3C          334          ;
FD3C 60        335          rts
FD3D          336          ;
FD3D          337          ;
FD3D          338          ; Do 80 column pick and fix.
FD3D          339          ;
FD3D A0 0F     340 PICKFIX   ldy #15
FD3F 20 B4 FB 341          jsr GOTOROM
FD42          342          ;
FD42 A4 24     343          ldy CH
FD44          344          ;
FD44 9D 00 02 345          sta INPUT,X
FD47          346          ;
FD47 20 ED FD 347 NOTCR     jsr COUT
FD4A          348          ;
FD4A BD 00 02 349          lda INPUT,X
FD4D C9 88     350          cmp #LARROW
FD4F F0 20     351          beq BCKSPC
FD51          352          ;
FD51 C9 98     353          cmp #CTRLX          ; ctrl-X pressed?
FD53 F0 0D     354          beq CANCEL
FD55          355          ;
FD55 E0 F8     356          cpx #$F8
FD57 90 06     357          bcc >1            ; margin?
FD59          358          ;
FD59 20 3A FF 359          jsr BELL
FD5C          360          ;
FD5C EA       361          nop
FD5D EA       362          nop
FD5E EA       363          nop
FD5F          364          ;
FD5F E8       365          ^1      inx

```



```

FD60 D0 13      366      bne NXTCHAR      ; always taken
FD62            367      ;
FD62            368      ;
FD62            369      ; Print a backslash after a cancelled line.
FD62            370      ;
FD62 A9 DC      371 CANCEL    lda #"\ "
FD64 20 ED FD    372          jsr COUT
FD67            373      ;
FD67 20 8E FD    374 GETLINE2 jsr CROUT
FD6A            375      ;
FD6A A5 33      376 GETLINE    lda PROMPT
FD6C 20 ED FD    377          jsr COUT
FD6F            378      ;
FD6F A2 01      379          ldx #1
FD71            380      ;
FD71 8A          381 BCKSPC    txa
FD72 F0 F3      382          beq GETLINE2
FD74            383      ;
FD74 CA          384          dex
FD75            385      ;
FD75 20 35 FD    386 NXTCHAR  jsr RDCHAR
FD78            387      ;
FD78            388      ;
FD78            389      ; For ^U processing use the screen character, either 40
FD78            390      ; column PICK or 80 column PICKFIX.
FD78            391      ;
FD78 C9 95      392          cmp #CTRLU
FD7A D0 08      393          bne ADDINP
FD7C            394      ;
FD7C B1 28      395          lda (BASL),Y      ; 40 column pick
FD7E            396      ;
FD7E            397      ;
FD7E            398      ; CAPTST entry gone.
FD7E            399      ;
FD7E            400      ;
FD7E 2C 1F C0    401          bit RDVID80
FD81 30 BA      402          bmi PICKFIX      ; 80 column pick
FD83            403      ;
FD83 EA          404          nop
FD84            405      ;
FD84            406      ;
FD84            407      ; Add to input buffer.
FD84            408      ;
FD84 9D 00 02    409 ADDINP    sta INPUT,X
FD87 C9 8D      410          cmp #RETURN
FD89 D0 BC      411          bne NOTCR
FD8B            412      ;
FD8B 20 9C FC    413          jsr CLREOL      ; clear to EOL after CR
FD8E            414      ;
FD8E            415      ;
FD8E A9 8D      416 CROUT     lda #RETURN
FD90 D0 5B      417          bne COUT      ; always taken
FD92            418      ;
FD92            419      ;
FD92            420      ; Print CR, then A1 in HEX.
FD92            421      ;
FD92 A6 3C      422 PRA1      ldx A1L
FD94 A4 3D      423          ldy A1H
FD96            424      ;
FD96 20 8E FD    425 PRXY      jsr CROUT
FD99 20 40 F9    426          jsr PRNTYX

```

```

FD9C          427 ;
FD9C A0 00     428          ldy #ZERO
FD9E          429 ;
FD9E A9 AD     430          lda #"-"
FDA0          431 ;
FDA0 4C ED FD  432          jmp COUT
FDA3          433 ;
FDA3          434 ;
FDA3 A5 3C     435 XAM8      lda A1L
FDA5 09 07     436          ora #7                ; set to finish at MOD 8=7
FDA7 85 3E     437          sta A2L
FDA9          438 ;
FDA9 A5 3D     439          lda A1H
FDAB 85 3F     440          sta A2H
FDAD          441 ;
FDAD A5 3C     442 MOD8CHK   lda A1L
FDAF 29 07     443          and #7
FDB1 D0 03     444          bne DATAOUT
FDB3          445 ;
FDB3 20 92 FD  446 XAM      jsr PRA1
FDB6          447 ;
FDB6 A9 A0     448 DATAOUT  lda #SPACE
FDB8 20 ED FD  449          jsr COUT
FDBB          450 ;
FDBB B1 3C     451          lda (A1L),Y
FDBD 20 DA FD  452          jsr PRBYTE
FDC0          453 ;
FDC0 20 BA FC  454          jsr NEXTA1
FDC3 90 E8     455          bcc MOD8CHK
FDC5          456 ;
FDC5 60        457          rts
FDC6          458 ;
FDC6          459 ;
FDC6          460 ; Determine if monitor mode is examine, add, or subtract.
FDC6          461 ;
FDC6 4A        462 XAMPM     lsr
FDC7 90 EA     463          bcc XAM
FDC9          464 ;
FDC9 4A        465          lsr
FDCA 4A        466          lsr
FDCB          467 ;
FDCB A5 3E     468          lda A2L
FDCD          469 ;
FDCD 90 02     470          bcc ADD
FDCF          471 ;
FDCF 49 FF     472          eor #NEGONE                ; 2's complement for subtract
FDD1          473 ;
FDD1 65 3C     474 ADD      adc A1L
FDD3 48        475          pha
FDD4          476 ;
FDD4 A9 BD     477          lda #"="
FDD6 20 ED FD  478          jsr COUT
FDD9          479 ;
FDD9 68        480          pla
FDDA          481 ;
FDDA          482 ;
FDDA          483 ; Print byte as 2 HEX digits.
FDDA          484 ;
FDDA 48        485 PRBYTE   pha
Fddb          486 ;
Fddb 4A        487          lsr

```

```

FDDC 4A          488          lsr
FDDD 4A          489          lsr
FDDE 4A          490          lsr
FDDF           491          ;
FDDF 20 E5 FD    492          jsr PRHEX2
FDE2           493          ;
FDE2 68          494          pla
FDE3           495          ;
FDE3           496          ;
FDE3           497          ; Print HEX digit in A-reg.
FDE3           498          ;
FDE3 29 0F       499 PRHEX      and #$0F
FDE5           500          ;
FDE5 09 B0       501 PRHEX2    ora #"0"
FDE7           502          ;
FDE7 C9 BA       503          cmp #"9"+1
FDE9 90 02       504          bcc COUT
FDEB           505          ;
FDEB 69 06       506          adc #6                      ; for HEX letters
FDED           507          ;
FDED           508          ;
FDED           509          ; Vector to user output routing.
FDED           510          ;
FDED 6C 36 00    511 COUT      jmp (CSWL)
FDF0           512          ;
FDF0           513          ;
FDF0 48          514 COUT2     pha
FDF1           515          ;
FDF1 C9 A0       516          cmp #SPACE
FDF3           517          ;
FDF3 4C 95 FC    518          jmp CHKINV
FDF6           519          ;
FDF6           520          ;
FDF6 48          521 COUTZ     pha
FDF7           522          ;
FDF7 84 35       523 COUTZ2    sty YSAV1
FDF9           524          ;
FDF9 A8          525          tay                      ; masked character
FDFA           526          ;
FDFA 68          527          pla                      ; original character
FDFB           528          ;
FDFB 4C 46 FC    529          jmp NEWVW
FDFE           530          ;
FDFE           531          ;
FDFE           532          dfs 2,ZERO                  ; 2 bytes
FE00           533          ;
FE00           534          ;
FE00 C6 34       535 BL1       dec YSAV
FE02 F0 9F       536          beq XAM8
FE04           537          ;
FE04 CA          538 BLANK     dex
FE05 D0 16       539          bne SETMDZ
FE07           540          ;
FE07 C9 BA       541          cmp #":"
FE09 D0 BB       542          bne XAMPM
FE0B           543          ;
FE0B 85 31       544 STOR      sta MODE
FE0D           545          ;
FE0D A5 3E       546          lda A2L
FE0F           547          ;
FE0F 91 40       548 STOR2     sta (A3L),Y

```

```

FE11          549 ;
FE11 E6 40    550 NEXTA3   inc A3L
FE13 D0 02    551          bne >1
FE15          552 ;
FE15 E6 41    553          inc A3H
FE17          554 ;
FE17 60       555 ^1      rts
FE18          556 ;
FE18          557 ;
FE18          558 ; Save converted `:`, `+`, `-`, or `.` as MODE.
FE18          559 ;
FE18 A4 34    560 SETMODE   ldy YSAV
FE1A          561 ;
FE1A B9 FF 01 562          lda INPUT-1,Y
FE1D          563 ;
FE1D 85 31    564 SETMDZ   sta MODE
FE1F          565 ;
FE1F 60       566          rts
FE20          567 ;
FE20          568 ;
FE20          569 ; Copy A2 to A4 and A5.
FE20          570 ;
FE20 A2 01    571 LT       ldx #1
FE22          572 ;
FE22 B5 3E    573 ^1      lda A2L,X
FE24 95 42    574          sta A4L,X
FE26 95 44    575          sta A5L,X
FE28          576 ;
FE28 CA       577          dex
FE29 10 F7    578          bpl <1
FE2B          579 ;
FE2B 60       580          rts
FE2C          581 ;
FE2C          582 ;
FE2C          583 ; Move A1 through A2 to A4.
FE2C          584 ;
FE2C B1 3C    585 MOVE     lda (A1L),Y
FE2E 91 42    586          sta (A4L),Y
FE30          587 ;
FE30 20 B4 FC 588          jsr NEXTA4
FE33 90 F7    589          bcc MOVE
FE35          590 ;
FE35 60       591          rts
FE36          592 ;
FE36          593 ;
FE36          594 ; Verify A1 through A2 with A4.
FE36          595 ;
FE36 B1 3C    596 VERIFY   lda (A1L),Y
FE38 D1 42    597          cmp (A4L),Y
FE3A F0 1C    598          beq >1
FE3C          599 ;
FE3C 20 92 FD 600          jsr PRA1
FE3F          601 ;
FE3F B1 3C    602          lda (A1L),Y
FE41 20 DA FD 603          jsr PRBYTE
FE44          604 ;
FE44 A9 A0    605          lda #SPACE
FE46 20 ED FD 606          jsr COUT
FE49          607 ;
FE49 A9 A8    608          lda #"("
FE4B 20 ED FD 609          jsr COUT

```

```

FE4E      610 ;
FE4E B1 42      611      lda (A4L),Y
FE50 20 DA FD   612      jsr PRBYTE
FE53      613 ;
FE53 A9 A9      614      lda #")"
FE55 20 ED FD   615      jsr COUT
FE58      616 ;
FE58 20 B4 FC   617 ^1      jsr NEXTA4
FE5B 90 D9      618      bcc VERIFY
FE5D      619 ;
FE5D 60         620      rts
FE5E      621 ;
FE5E      622 ;
FE5E      623 ; Move A1 to PC if specified and disassemble 20
FE5E      624 ; instructions.
FE5E      625 ;
FE5E 20 75 FE   626 LIST      jsr A1PC
FE61      627 ;
FE61 A9 14      628      lda #20
FE63      629 ;
FE63 48         630 ^1      pha
FE64      631 ;
FE64 20 D0 F8   632      jsr INSTDSP
FE67 20 53 F9   633      jsr PCADJ
FE6A      634 ;
FE6A 85 3A      635      sta PCL
FE6C 84 3B      636      sty PCH
FE6E      637 ;
FE6E 68         638      pla
FE6F      639 ;
FE6F 38         640      sec
FE70      641 ;
FE70 E9 01      642      sbc #1
FE72 D0 EF      643      bne <1
FE74      644 ;
FE74 60         645      rts
FE75      646 ;
FE75      647 ;
FE75      648 ; If an address was specified copy it from A1 to PC.
FE75      649 ;
FE75 8A         650 A1PC      txa
FE76 F0 07      651      beq >1
FE78      652 ;
FE78 B5 3C      653 A1PCLP   lda A1L,X
FE7A 95 3A      654      sta PCL,X
FE7C      655 ;
FE7C CA         656      dex
FE7D 10 F9      657      bpl A1PCLP
FE7F      658 ;
FE7F 60         659 ^1      rts
FE80      660 ;
FE80      661 ;
FE80      662 ; Set INVERSE/NORMAL video.
FE80      663 ;
FE80 A0 3F      664 SETINV   ldy #INVERSE
FE82 D0 02      665      bne >1
FE84      666 ;
FE84 A0 FF      667 SETNORM  ldy #NEGONE
FE86      668 ;
FE86 84 32      669 ^1      sty INVFLG
FE88      670 ;

```

; always taken

```

FE88 60          671          rts
FE89          672          ;
FE89          673          ;
FE89 A9 00      674 SETKBD    lda #ZERO          ; IN#0
FE8B          675          ;
FE8B 85 3E      676 INPORT    sta A2L          ; IN#n
FE8D          677          ;
FE8D A2 38      678 INPORT2   ldx #KSWL
FE8F A0 1B      679          ldy #KEYIN
FE91          680          ;
FE91 D0 08      681          bne IOPORT          ; always taken
FE93          682          ;
FE93          683          ;
FE93 A9 00      684 SETVID    lda #ZERO          ; PR#0
FE95          685          ;
FE95 85 3E      686 OUTPORT   sta A2L          ; PR#n
FE97          687          ;
FE97 A2 36      688 OUTPORT2  ldx #CSWL
FE99 A0 F0      689          ldy #COUT2
FE9B          690          ;
FE9B A5 3E      691 IOPORT    lda A2L
FE9D 29 0F      692          and #$0F
FE9F F0 04      693          beq >1
FEA1          694          ;
FEA1 09 C0      695          ora /PAGEC0
FEA3 A0 00      696          ldy #ZERO
FEA5          697          ;
FEA5 94 00      698 ^1      sty LOC0,X
FEA7 95 01      699          sta LOC1,X
FEA9          700          ;
FEA9 A0 0E      701          ldy #14
FEAB          702          ;
FEAB 4C B4 FB   703          jmp GOTOROM
FEAE          704          ;
FEAE          705          ;
FEAE F0 55      706 ZAPMEM    beq ZAPMEM2          ; Y-reg = 0, always taken
FEB0          707          ;
FEB0          708          ;
FEB0 4C 00 E0   709 XBASIC    jmp BASIC          ; cold start
FEB3          710          ;
FEB3          711          ;
FEB3 4C 03 E0   712 BASCONT   jmp BASIC2          ; warm start
FEB6          713          ;
FEB6          714          ;
FEB6 20 75 FE   715 GO        jsr A1PC
FEB9 20 3F FF   716          jsr RESTORE
FEBF          717          ;
FEBF 6C 3A 00   718          jmp (PCL)
FEBF          719          ;
FEBF          720          ;
FEBF 4C D7 FA   721 REGZ      jmp REGDSP
FEC2          722          ;
FEC2          723          ;
FEC2 C6 34      724 TRACE    dec YSAV
FEC4          725          ;
FEC4 8D 07 C0   726 STEPZ    sta CXROMON
FEC7          727          ;
FEC7 4C 08 C5   728          jmp CXSTEP
FECA          729          ;
FECA          730          ;
FECA          731          ; Jump to user ^Y vector.

```

```

FECA          732 ;
FECA 4C F8 03 733 USR      jmp USRYHAND
FECD          734 ;
FECD          735 ;
FECD          736 ; Enter the cassette WRITE routine (removed).
FECD          737 ;
FECD 60       738 WRITE    rts
FECE          739 ;
FECE          740 ;
FECE C9 F1    741 CHRTBLX3 cmp #$89+$B0^"X"      ; SEARCH command
FED0 D0 3C    742          bne RTN.FF.0
FED2          743 ;
FED2          744 ; New 'X' command, used in the following ways:
FED2          745 ;
FED2          746 ; 'char<$strt.$end X <cr>      "char<$strt.$end X <cr>
FED2          747 ; 'char'char<$strt.$end X <cr> "char"char<$strt.$end X <cr>
FED2          748 ; 'char"char<$strt.$end X <cr> "char'char<$strt.$end X <cr>
FED2          749 ;
FED2          750 ; $val<$strt.$end X <cr> where $val is MSB/LSB
FED2          751 ;
FED2 A0 01    752 SEARCH   ldy #1
FED4          753 ;
FED4 A5 43    754          lda A4H
FED6 F0 04    755          beq >1
FED8          756 ;
FED8 D1 3C    757          cmp (A1L),Y
FEDA D0 0A    758          bne >2
FEDC          759 ;
FEDC 88       760 ^1      dey
FEDD          761 ;
FEDD A5 42    762          lda A4L
FEDF D1 3C    763          cmp (A1L),Y
FEE1 D0 03    764          bne >2
FEE3          765 ;
FEE3 20 92 FD 766          jsr PRA1
FEE6          767 ;
FEE6 20 BA FC 768 ^2      jsr NEXTA1
FEE9 90 E7    769          bcc SEARCH
FEEB          770 ;
FEEB 20 8E FD 771          jsr CROUT
FEEE          772 ;
FEEE 4C F9 FE 773          jmp CRMON2          ; fix program counter
FEF1          774 ;
FEF1          775 ;
FEF1          776 ; Enter the Mini-Assembler.  Fall into CRMON.
FEF1          777 ;
FEF1 A0 0D    778 MINIASM  ldy #13
FEF3          779 ;
FEF3 20 B4 FB 780          jsr GOTOROM
FEF6          781 ;
FEF6          782 ;
FEF6 20 00 FE 783 CRMON    jsr BL1
FEF9          784 ;
FEF9 68       785 CRMON2   pla
FEFA 68       786          pla
FEFB          787 ;
FEFB D0 6C    788          bne MON2          ; always taken
FEFD          789 ;
FEFD          790 ;
FEFD          791 ; Enter the cassette READ routine.  If CHKSUM compares then
FEFD          792 ; sound BELL and return to CALLER or enter PRNTERR.

```

```

FEFD      793 ;
FEFD      794 ; This command is used in the following way:
FEFD      795 ;
FEFD      796 ; $strt.$end R <cr>
FEFD      797 ;
FEFD 20 9C F3 798 READ      jsr CXREAD
FF00      799 ;
FF00 F0 38   800          beq BELL
FF02      801 ;
FF02 D0 29   802          bne PRNTERR          ; always taken
FF04      803 ;
FF04      804 ;
FF04      805          dfs 1,ZERO          ; 1 byte
FF05      806 ;
FF05      807 ;
FF05      808 ; New 'Z' command, used in the following way:
FF05      809 ;
FF05      810 ; $val<$strt.$end Z <cr>
FF05      811 ;
FF05 A5 42   812 ZAPMEM2   lda A4L
FF07 91 3C   813          sta (A1L),Y
FF09      814 ;
FF09 20 BA FC 815          jsr NEXTA1
FF0C 90 F7   816          bcc ZAPMEM2
FF0E      817 ;
FF0E 60      818 RTN.FF.0 rts
FF0F      819 ;
FF0F      820 ;
FF0F C9 EB   821 CHRTBLX2  cmp #$89+$B0^"R"          ; READ command
FF11 F0 EA   822          beq READ
FF13      823 ;
FF13 C9 A0   824          cmp #$89+$B0^"'"          ; APOSTROPHE command
FF15 D0 B7   825          bne CHRTBLX3
FF17      826 ;
FF17 18      827          clc
FF18      828 ;
FF18 08      829 LOOKASC   php
FF19      830 ;
FF19 B9 00 02 831          lda INPUT,Y
FF1C      832 ;
FF1C C9 8D   833          cmp #RETURN
FF1E F0 0A   834          beq >2
FF20      835 ;
FF20 28      836          plp
FF21 B0 02   837          bcs >1
FF23      838 ;
FF23 29 7F   839          and #MSBCLR
FF25      840 ;
FF25 A2 07   841 ^1      ldx #7
FF27      842 ;
FF27 C8      843          iny
FF28 D0 66   844          bne NEXTBIT          ; always taken
FF2A      845 ;
FF2A 28      846 ^2      plp
FF2B F0 7A   847          beq GETNUM          ; always taken
FF2D      848 ;
FF2D      849 ;
FF2D      850 ; Print "ERR" and fall into BELL.
FF2D      851 ;
FF2D A9 C5   852 PRNTERR   lda #"E"
FF2F 20 ED FD 853          jsr COUT

```



```

FF32      854 ;
FF32 A9 D2      855      lda #"R"
FF34 20 ED FD    856      jsr COUT
FF37 20 ED FD    857      jsr COUT
FF3A      858 ;
FF3A      859 ;
FF3A A9 87      860 BELL      lda #ASCIBELL
FF3C      861 ;
FF3C 4C ED FD    862      jmp COUT
FF3F      863 ;
FF3F      864 ;
FF3F      865 ; Restore the 65C02 registers.
FF3F      866 ;
FF3F A5 48      867 RESTORE   lda PREG
FF41 48          868      pha
FF42      869 ;
FF42 A5 45      870      lda AREG
FF44 A6 46      871      ldx XREG
FF46 A4 47      872      ldy YREG
FF48      873 ;
FF48 28          874      plp
FF49      875 ;
FF49 60          876      rts
FF4A      877 ;
FF4A      878 ;
FF4A      879 ; Save the 65C02 registers.
FF4A      880 ;
FF4A 85 45      881 SAVE      sta AREG
FF4C      882 ;
FF4C 86 46      883 SAVE2     stx XREG
FF4E 84 47      884      sty YREG
FF50      885 ;
FF50 08          886      php
FF51      887 ;
FF51 68          888      pla
FF52 85 48      889      sta PREG
FF54      890 ;
FF54 BA          891      tsx
FF55 86 49      892      stx SPNT
FF57      893 ;
FF57 D8          894      cld
FF58      895 ;
FF58 60          896 IORTS     rts
FF59      897 ;
FF59      898 ;
FF59 20 08 FB    899 OLDRST   jsr RSETINIT      ; not referenced
FF5C      900 ;
FF5C 4C 65 FF    901      jmp MON
FF5F      902 ;
FF5F      903 ;
FF5F C9 9B      904 CHRTBLX1 cmp #$89+$B0^""      ; QUOTE command
FF61 F0 B5      905      beq LOOKASC
FF63      906 ;
FF63 D0 AA      907      bne CHRTBLX2      ; always taken
FF65      908 ;
FF65      909 ;
FF65      910 ; Monitor entry point.
FF65      911 ;
FF65 D8          912 MON      cld
FF66      913 ;
FF66 20 3A FF    914      jsr BELL

```

```

FF69          915 ;
FF69 A9 AA    916 MON2      lda #"*"
FF6B 85 33    917          sta PROMPT
FF6D          918 ;
FF6D 20 67 FD 919          jsr GETLINE2
FF70 20 C7 FF 920          jsr ZMODE
FF73          921 ;
FF73          922 ;
FF73 20 A7 FF 923 NEXTITM  jsr GETNUM
FF76          924 ;
FF76 84 34    925          sty YSAV
FF78          926 ;
FF78 A0 17    927          ld y #SUBTBL-CHRTBL
FF7A          928 ;
FF7A          929 ;
FF7A 88       930 CHRSRCH  dey
FF7B 30 E8    931          bmi MON
FF7D          932 ;
FF7D D9 CC FF 933          cmp CHRTBL,Y
FF80 D0 F8    934          bne CHRSRCH
FF82          935 ;
FF82 20 BE FF 936          jsr TOSUBR
FF85          937 ;
FF85 A4 34    938          ld y YSAV          ; process next entry
FF87          939 ;
FF87 4C 73 FF 940          jmp NEXTITM
FF8A          941 ;
FF8A          942 ;
FF8A A2 03    943 DIG      ldx #3
FF8C          944 ;
FF8C 0A       945          asl
FF8D 0A       946          asl
FF8E 0A       947          asl
FF8F 0A       948          asl
FF90          949 ;
FF90          950 ;
FF90 0A       951 NEXTBIT  asl
FF91          952 ;
FF91 26 3E    953          rol A2L
FF93 26 3F    954          rol A2H
FF95          955 ;
FF95 CA       956          dex
FF96 10 F8    957          bpl NEXTBIT
FF98          958 ;
FF98          959 ;
FF98          960 ; If MODE is 0x00 copy A2 to A1 and A3.
FF98          961 ;
FF98 A5 31    962 ^1      lda MODE
FF9A D0 06    963          bne >2
FF9C          964 ;
FF9C B5 3F    965          lda A2H,X
FF9E 95 3D    966          sta A1H,X
FFA0 95 41    967          sta A3H,X
FFA2          968 ;
FFA2 E8       969 ^2      inx
FFA3 F0 F3    970          beq <1
FFA5          971 ;
FFA5 D0 06    972          bne NEXTCHR          ; always taken
FFA7          973 ;
FFA7          974 ;
FFA7 A2 00    975 GETNUM  ldx #ZERO

```

```

FFA9 86 3E      976      stx A2L
FFAB 86 3F      977      stx A2H
FFAD           978      ;
FFAD           979      ;
FFAD 20 FD FC   980  NEXTCHR  jsr UPRMON
FFB0           981      ;
FFB0 49 B0      982      eor #"0"
FFB2           983      ;
FFB2 C9 0A      984      cmp #10
FFB4 90 D4      985      bcc DIG
FFB6           986      ;
FFB6 69 88      987      adc #$88
FFB8 C9 FA      988      cmp #$FA
FFBA B0 CE      989      bcs DIG
FFBC           990      ;
FFBC 90 A1      991      bcc CHRTBLX1      ; always taken
FFBE           992      ;
FFBE           993      ;
FFBE           994      ; Get address of PAGEFE subroutine.
FFBE           995      ;
FFBE A9 FE      996  TOSUBR   lda /PAGEFE
FFC0 48         997      pha
FFC1           998      ;
FFC1 B9 E3 FF   999      lda SUBTBL,Y
FFC4 48        1000     pha
FFC5           1001     ;
FFC5 A5 31      1002     lda MODE
FFC7           1003     ;
FFC7           1004     ;
FFC7 A0 00      1005  ZMODE   ldy #ZERO
FFC9 84 31      1006     sty MODE
FFCB           1007     ;
FFCB 60         1008     rts
FFCC           1009     ;
FFCC           1010     ;
FFCC           1011     CHRTBL:
FFCC BC        1012     byt $89+$B0^CTRLC      ; ctrl-C, jmp 0xE003
FFCD B2        1013     byt $89+$B0^CTRLY      ; ctrl-Y, jmp 0x3F8
FFCE BE        1014     byt $89+$B0^CTRLE      ; ctrl-E, display registers
FFCF ED        1015     byt $89+$B0^"T"        ; T, trace
FFD0 EF        1016     byt $89+$B0^"V"        ; V, verify memory
FFD1 C4        1017     byt $89+$B0^CTRLK      ; ctrl-K, set input device
FFD2 EC        1018     byt $89+$B0^"S"        ; S, step
FFD3 A9        1019     byt $89+$B0^CTRLP      ; ctrl-P, set output device
FFD4 BB        1020     byt $89+$B0^CTRLB      ; ctrl-B, jmp 0xE000
FFD5 A6        1021     byt $89+$B0^"- "       ; -, math operator
FFD6 A4        1022     byt $89+$B0^"+ "       ; +, math operator
FFD7 06        1023     byt $89+$B0^"M"        ; M, move memory
FFD8 95        1024     byt $89+$B0^"<"       ; <, move direction
FFD9 07        1025     byt $89+$B0^"N"        ; N, set normal
FFDA 02        1026     byt $89+$B0^"I"        ; I, set inverse
FFDB 05        1027     byt $89+$B0^"L"        ; L, list memory
FFDC 9A        1028     byt $89+$B0^"! "       ; !, enter mini-assembler
FFDD 00        1029     byt $89+$B0^"G"        ; G, jmp to memory
FFDE F3        1030     byt $89+$B0^"Z"        ; Z, fill memory with value
FFDF 93        1031     byt $89+$B0^": "       ; :, input instruction mode
FFE0 A7        1032     byt $89+$B0^". "       ; ., memory delimiter
FFE1 C6        1033     byt $89+$B0^RETURN    ; CR, carriage return
FFE2 99        1034     byt $89+$B0^" "       ; space, input instruct mode
FFE3           1035     ;
FFE3           1036     ;

```

```

FFE3          1037  SUBTBL:
FFE3 B2        1038      byt BASCONT-1      ; ctrl-C <cr>
FFE4 C9        1039      byt USR-1           ; ctrl-Y <cr>
FFE5 BE        1040      byt REGZ-1          ; ctrl-E <cr>
FFE6 C1        1041      byt TRACE-1         ; trace
FFE7 35        1042      byt VERIFY-1        ; $dst<$strt.$end V <cr>
FFE8 8C        1043      byt INPORT2-1       ; #slot ctrl-K <cr>
FFE9 C3        1044      byt STEPZ-1         ; step
FFEA 96        1045      byt OUTPORT2-1      ; #slot ctrl-P <cr>
FFEB AF        1046      byt XBASIC-1        ; ctrl-B <cr>
FFEC 17        1047      byt SETMODE-1       ; $val-$val <cr>
FFED 17        1048      byt SETMODE-1       ; $val+$val <cr>
FFEE 2B        1049      byt MOVE-1          ; $dst<$strt.$end M <cr>
FFEF 1F        1050      byt LT-1            ; $dst<$strt ::: <cr>
FFF0 83        1051      byt SETNORM-1       ; N <cr>
FFF1 7F        1052      byt SETINV-1        ; I <cr>
FFF2 5D        1053      byt LIST-1          ; $adr L <cr>
FFF3 F0        1054      byt MINIASM-1       ; mini-assembler
FFF4 B5        1055      byt GO-1            ; $adr G <cr>
FFF5 AD        1056      byt ZAPMEM-1        ; $val<$strt.$end Z <cr>
FFF6 17        1057      byt SETMODE-1       ; $adr:$val ... <cr>
FFF7 17        1058      byt SETMODE-1       ; $strt.$end
FFF8 F5        1059      byt CRMON-1         ; <cr>
FFF9 03        1060      byt BLANK-1         ; <space>
FFFA          1061      ;
FFFA          1062      ;
FFFA FB 03     1063      adr NMASKIRQ
FFFC 62 FA     1064      adr RESET
FFFE FA C3     1065      adr IRQRTN
0000          1066      ;
0000          1067      ;

```

```
BSAVE F0ROM,D1,A$1000,B,L$0FFE
```

```

0000          1068      usr F0ROM,D1
0000          1069      ;
0000          1070      ;      dcm "CD D2"
0000          1071      ;
0000          1072      ;
0000          1073      stt "ROM2E Symbol Table"
0000          1074      ;
0000          1075      ;
0000          1076      end 111

```

```
*** End of Assembly
```

Symbol List starts at 0x7800, ends at 0xABAE, used 0x33AE, remaining 0x0276

Symbols unsorted:

LOC0	0000	LOC1	0001	LOC2	0002	R0L	0000	R0H	0001
R12L	0018	R12H	0019	R14L	001C	R14H	001D	R15L	001E
R15H	001F	GOWARM	0000	GOSTROUT	0003	GOUSR	000A	JMPADRS	0090
BYTVALUE	000D	CHARAC	000D	ENDCHR	000E	EOLPTR	000F	NUMDIM	000F
TOKNCNTR	000F	DIMFLG	0010	VALTYP	0011	DATAFLG	0013	GARFLG	0013
SUBFLG	0014	INPUTFLG	0015	CPRMASK	0016	TOGLFLG	0016	SHAPE	001A
COLBITS	001C	COLCOUNT	001D	WNDLFT	0020	WNDWDTH	0021	WNDTOP	0022
WNCBTM	0023	CH	0024	CV	0025	GBASL	0026	GBASH	0027
BASL	0028	BASH	0029	BAS2L	002A	BAS2H	002B	H2	002C
LMNEM	002C	V2	002D	RMNEM	002D	CHKSUM	002E	FORMAT	002E
MASK	002E	LENGTH	002F	LASTIN	002F	SIGN	002F	COLOR	0030
HMASK	0030	MODE	0031	INVFLG	0032	PROMPT	0033	YSAV	0034
YSAV1	0035	CSWL	0036	CSWH	0037	KSWL	0038	KSWH	0039
PCL	003A	PCH	003B	A1L	003C	A1H	003D	A2L	003E
A2H	003F	A3L	0040	A3H	0041	A4L	0042	A4H	0043
A5L	0044	MACSTAT	0044	OPRND	0044	T2GUARD	0045	AREG	0045
XREG	0046	YREG	0047	PREG	0048	SPNT	0049	RNDL	004E
RNDH	004F	LINNUM	0050	ACL	0050	ACH	0051	TEMPPT	0052
LASTPT	0053	TEMPST	0055	INDEX	005E	DEST	0060	OFFSET	0061
MULMANT	0062	MULGUARD	0066	PRGTAB	0067	VARTAB	0069	ARYTAB	006B
STREND	006D	FRETOP	006F	FRESPC	0071	MEMSIZE	0073	CURLIN	0075
OLDLIN	0077	TEXTPTR	0079	DATLIN	007B	DATPTR	007D	SRCPTR	007F
VARNAM	0081	VARPNT	0083	FORPNT	0085	LASTOP	0087	TXPTRSAV	0087
CPRTYPE	0089	FUNCNAM	008A	TEMP3	008A	DSCPTR	008C	T3GUARD	008F
RTNADR	0091	ARGGUARD	0092	TEMP1	0093	ARYPNT	0094	HIGHDS	0094
LEN	0094	PROCESS	0095	HIGHTR	0096	TEMP2	0098	COUNTER	0099
EXPCOUNT	009A	DPFLAG	009B	LOWTR	009B	EXPSIGN	009C	DSCTMP	009D
FACEXP	009D	FACMANT	009E	VARPTR	00A0	FACSIGN	00A2	COEFNUM	00A3
MINUSLOC	00A3	EXTSIGN	00A4	ARGEXP	00A5	ARGMANT	00A6	ARGSIGN	00AA
XORSIGN	00AB	FACGUARD	00AC	STRING1	00AB	STRING2	00AD	COEFPTR	00AD
SAVY	00AD	PRGEND	00AF	CHRGADR	00B1	TXTPTR	00B8	IRAND	00C9
SIGNFLG	00CD	SPCLFLAG	00CD	HRXDELTA	00D0	HRDELTA	00D2	HRFLAG	00D3
HRWORK	00D4	SHPVAL	00D0	ROTQVAL	00D1	ROTHVAL	00D2	ROTVVAL	00D3
ROTHSUM	00D4	ROTVSUM	00D5	SHPOLD	00D7	RUNFLAG	00D6	ERRFLG	00D8
ERRLIN	00DA	ERRPOS	00DC	ERRNUM	00DE	ERRSTK	00DF	HRXCOOR	00E0
HRXCOOR	00E2	HRCOLOR	00E4	HRHORZ	00E5	HRPAG	00E6	HRSCALE	00E7
HRSHPTBL	00E8	HRCOLCNT	00EA	FIRST	00F0	SPEEDBYT	00F1	TRACEFLG	00F2
FLASHBYT	00F3	TXTPTRSV	00F4	CURLINSV	00F6	REMSTK	00F8	HRROT	00F9
HLINMOD	0001	FPCDMOD	0001	ZERO	0000	IVARLEN	0002	AVARLEN	0003
MAXPDL	0003	MANTLEN	0004	AHDRLEN	0005	FACLEN	0005	FACSIZE	0006
SVARLEN	0007	PXLSBYTE	0007	BYTEBITS	0008	DFLTDIM	000B	FRAMSIZE	0014
MANTBITS	0020	MOVEMASK	0003	PLOTMASK	0004	SCMDMASK	0007	SROTMASK	000F
INVERSE	003F	FLASH	007F	INVRSE80	007F	MSBCLR	007F	MSBSET	0080
EXPBIAS	0080	LWRMASK	00DF	MAXINPUT	00EF	NEG TWO	00FE	NEGONE	00FF
CTRLB	0082	CTRLC	0083	CTRLE	0085	ASCIBELL	0087	CTRLH	0088
LARROW	0088	DARROW	008A	CTRLJ	008A	CTRLK	008B	UARROW	008B
RETURN	008D	CTRLP	0090	CTRLS	0093	CTRLU	0095	RARROW	0095
CTRLX	0098	CTRLY	0099	ESCAPE	009B	SPACE	00A0	GOODF8	0006
PWRUPBYT	00A5	ERROR.1	00FE	ERROR.2	00FF	TK.TAB	00C0	TK.TO	00C1
TK.FN	00C2	TK.SPC	00C3	TK.THEN	00C4	TK.AT	00C5	TK.NOT	00C6
TK.STEP	00C7	TK.PLUS	00C8	TK.MINUS	00C9	TK.GRTR	00CF	TK.EQUAL	00D0
TK.SGN	00D2	TK.SCRN	00D7	OTV.OR	0046	OTV.AND	0050	OTV.REL	0064
OTV.ADD	0079	OTV.MUL	007B	OTV.PWR	007D	OTV.NEQ	007F	PAGESIZE	0100
STACK	0100	INPUT	0200	XFERADR	03ED	AUTOBRK	03EF	AUTORSET	03F2
PWRSTATE	03F4	USRAHAND	03F5	USRYHAND	03F8	NMASKIRQ	03FB	MASKIRQ	03FE
TEXTPG1	0400	PG1TXLOC	05B0	OLDCH	047B	XMODE	04FB	OURCH	057B

OURCV	05FB	XCHAR	067B	XCOORD	06FB	XTEMP1	077B	OLDBASL	077B
XTEMP2	07FB	OLDBASH	07FB	MSLOT	07F8	PAGE08	0800	PAGE0C	0C00
PAGE10	1000	PAGE20	2000	PAGE40	4000	PAGEBD	BD00	PAGEBF	BF00
PAGEC0	C000	PAGEC1	C100	PAGEC8	C800	PAGEDO	D000	PAGEF0	F000
PAGEFE	FE00	KEY	C000	STR80OFF	C000	STR80ON	C001	RAMRDOFF	C002
RAMRDON	C003	RAMWROFF	C004	RAMWRON	C005	CXROMOFF	C006	CXROMON	C007
AUXZPOFF	C008	AUXZPON	C009	C3ROMOFF	C00A	C3ROMON	C00B	VID80OFF	C00C
VID80ON	C00D	ALTCHOFF	C00E	ALTCHON	C00F	CLRKEY	C010	RDBANK2	C011
RDLGRAM	C012	RDRAMRD	C013	RDRAMWR	C014	RDCXROM	C015	RDAUXZP	C016
RDC3ROM	C017	RDSTR80	C018	RDVRTBLK	C019	RDTEXT	C01A	RDMIXED	C01B
RDPAGE2	C01C	RDHIRES	C01D	RDALTCH	C01E	RDVID80	C01F	TAPEOUT	C020
SPKRTOGL	C030	TEXTOFF	C050	TEXTON	C051	MIXEDOFF	C052	MIXEDON	C053
PAGE1ON	C054	PAGE2ON	C055	HIRESOFF	C056	HIRESON	C057	ANN1OFF	C058
ANN2OFF	C05A	ANN3ON	C05D	ANN4ON	C05F	TAPEIN	C060	PB1IN	C061
PB2IN	C062	GC1IN	C064	GCTOGL	C070	RAM2WP	C080	ROM2WE	C081
ROM2WP	C082	RAM2WE	C083	RAM1WP	C088	RAM1WE	C08B	M.6	0040
M.CTL2	0020	M.4	0010	M.CTL	0008	M.2	0004	M.1	0002
M.MOUSE	0001	M.PASCAL	0080	M.CURSOR	0010	M.GOXY	0008	M.VMODE	0004
M.PAS1.0	0002	IOSPACE	C000	DOCXCMD	C100	XCLREOP	C103	XHOME	C119
XSCROLL	C123	XSETWND	C152	XCLREOL2	C160	YSCROLL	C165	YCLREOL	C168
YCLREOL2	C16B	YCLREOP	C170	YSETWND	C173	XRESET	C176	YRESET	C176
YRDKEY	C179	YHOME	C17C	YIOPORT	C18A	XIOPORT	C195	ISO	C1A0
XRDKEY	C1A9	XBASCLC	C1B6	XRDESC	C1BD	XNEWVW	C1CF	XGETFMT	C1D5
YGETFMT	C1D5	XGOMINI	C1E1	YGOMINI	C1E1	XPICKFIX	C1E8	YPICKFIX	C1E8
XCLREOL	C1F1	XCLREOL1	C1F3	XVTAB	C203	CXEXIT	C208	XDOCMD	C211
YREGTBLX	C243	YREGTBLY	C24F	XKEYIN	C25B	XSETWNDX	C2A5	XRESETX	C2B5
CXRESET	C2DD	KBDTBL	C2EF	KBDOUT	C2F3	C3SPACE	C300	BASICINT	C300
BASICIN	C305	BASICOUT	C307	AUXMOVE	C311	XFER	C314	BASICENT	C317
JC8	C344	JBASINIT	C347	JPINIT	C34A	JPREAD	C350	JPWRITE	C356
JPSTAT	C35C	SETC8	C36D	DOMOVE	C376	DOXFER	C3C3	IRQDONE	C3F4
IRQRTN	C3FA	NEWIRQ	C47C	RAMSWTBL	C4C1	SWTBLEN	0006	FORM	C4C8
GETNSP	C500	CXSTEP	C508	XJMPX	C575	XRTI	C584	XRTS	C588
PCINC2	C58C	PCINC3	C58E	XJSR	C598	XJMP	C5A3	XJMPAT	C5A4
BRANCH	C5B9	BRANCH2	C5C7	INITBL	C5CD	INITBLEN	0008	XGETFMT2	C5D5
PROCVAR	C600	PROCSPCL	C64E	SW16	C670	SW16B	C679	SW16C	C67F
TOBR	C6A2	SETCMD2	C6B2	RSCMD2	C6C6	BSNSCMD2	C6D7	BRTBL	C6E2
OPTBL	C6E3	RTNCMD	C701	SETCMD	C709	RSCMD	C70B	BSNSCMD	C70D
LDCMD	C70F	STCMD	C718	LD@CMD	C721	ST@CMD	C72B	ST@CMD2	C72D
INRCMD	C733	LDD@CMD	C73A	STD@CMD	C743	POPD@CMD	C74C	POP@CMD	C753
STP@CMD	C763	ADDCMD	C76C	SUBCMD	C77A	CPRCMD	C77C	BSCMD	C790
BRCMD	C79C	BNCCMD	C79D	BCCMD	C7AE	BPCMD	C7B1	BMCMD	C7B6
BZCMD	C7BB	BNZCMD	C7C2	BM1CMD	C7C9	BNM1CMD	C7D2	SOUTCMD	C7DB
RSNSCMD	C7E0	SJMPCMD	C7E9	DCRCMD	C7F7	PXINIT	C800	BASCINIT	C803
C3HOOKS	C82A	C3IN	C832	PXREAD	C84D	CSETUP	C850	CXNEWVW	C870
CXNEWVW2	C874	CXVIDCK3	C87C	CXVIDCK4	C87E	CTRLON	C8BD	BIORET	C8C5
XINPUT	C8E6	ESCCHAR	C96B	ESCTABL	C97C	CXKEYIN	C98D	PXWRITE	C9AA
PXINIT2	C9B0	PPINIT	C9B4	PPREAD	C9D6	PPWRITE	C9F0	DOBASL	CA1F
PWRITER	CA29	BITBYT60	CA2E	OPTBLC	CA71	OPTBLL	CA7D	TSTROMCD	CA89
TESTCARD	CA90	XBASCALC	CABA	CTRLCHR2	CAD2	CTRLCHR	CAD6	CTRLXFER	CB07
PCURON	CB0D	PCUROFF	CB18	XBS	CB40	XCR	CB51	XEM	CB5F
XFS	CB6B	XUS	CB79	XSO	CB84	XSI	CB8F	CTRLADRL	CB9E
CTRLADRH	CBB9	SCROLLDN	CBD4	XLF	CBD8	SCROLLUP	CBEB	XFF	CC74
XVT	CC77	XSUB	CC93	XGS	CC97	XGSEOLZ	CC9A	CLR40	CCA5
CLRHAF	CCAD	CLR80	CCBA	CLR2	CCD2	XDC1	CCE7	XDC1.2	CCEC
XDC2	CCF9	DO40	CD2B	SETTOP	CD2E	MOUSEOFF	CD3A	MOUSEON	CD41
XNAK	CD4A	SETKEYIN	CD58	SETCOUT2	CD61	FULL40	CD6A	FULL80	CD6E
QUIT	CD7D	SCRN84	CD8E	SCRN48	CDC1	SCRNRET	CDF5	XVTAB2	CDFB
XVTAB3	CE00	XRDKEY2	CE11	PASINV	CE1F	INVERT	CE26	STORCHAR	CE38
PICK	CE44	STORIT	CE70	ESCON	CEB1	ESCOFF	CEC4	ESCRTN	CECD
PSETUP	CED4	COPYROM	CEF4	REL	CF3A	TRYNEXT	CF6B	NXTLINE2	CF9C
CLRROM	CFFF	BASADDR	D000	BSFOR	D002	BSDATA	D006	BSPOP	D042

BSGOTO	D056	BSGOSUB	D060	BSREM	D064	BSPRINT	D074	FN1ADDR	D080
FS1SGN	D080	FS1SCRN	D08A	FS2LEFT	D0AC	FS3PI	D0B2	TAGADDR	D0B6
TAG.NEG	D0CB	TAG.EQU	D0CE	TAG.REL	D0D1	BASNAME	D0D4	MESGS	D25B
MESG01	D25B	MESG02	D26B	MESG03	D271	MESG04	D285	MESG05	D290
MESG06	D2A0	MESG07	D2A8	MESG08	D2B5	MESG09	D2C8	MESG10	D2D5
MESG11	D2E4	MESG12	D2F4	MESG13	D302	MESG14	D30F	MESG15	D31E
MESG16	D331	MESG17	D340	MESG18	D352	MESG20	D35A	GTFORPNT	D362
BLTU	D393	CKSTKSIZ	D3D6	CKSTRSIZ	D3E3	OM.ERR	D410	PRTERR	D412
PRLINUM	D431	ASROMWRM	D43C	RESTART	D43C	ASROMRST	D4F2	ASENTER	D4F2
INLIN	D52C	INLIN2	D52E	PARSINPT	D559	PARSE	D56C	PARSE2	D56D
PARSE3	D5A8	PARSE4	D5CD	PARSE5	D5F2	PARSE6	D5F9	PARSE7	D610
FNDLIN	D61A	FNDLIN2	D61E	RTN.D6.4	D648	BNEW	D649	SCRATCH	D64B
ASROMCLR	D665	SETPTRS	D665	BCLEAR	D66A	CLEARC	D66C	STKINIT	D683
RTN.D6.9	D696	STXTPTR	D697	RTN.D6.A	D6A4	BLIST	D6A5	LIST2	D6DA
LIST3	D6FC	LIST4	D700	LIST5	D721	LIST6	D727	GETCHR	D758
BFOR	D766	STEP	D7AF	ASROMNEW	D7D2	NEWSTT	D7D2	DOTRACE	D805
GOEND3	D826	DOSTAMT	D828	DOSTAMT2	D82A	DOSTAMT3	D842	BRESTORE	D849
SETDAPTR	D853	RTN.D8.5	D857	ISCNTLC	D858	DOCTRL.C	D863	ASROMERR	D865
BSTOP	D86E	BEND	D870	END2	D871	END3	D88A	BCONT	D896
RTN.D8.A	D8AF	DOHANDLR	D8B0	PULL3A	D8B3	RDBYTE	D8BB	BLOAD	D8C9
RD2BIT	D8FF	RDBIT	D902	BRUN	D912	RUN2	D91E	BGOSUB	D921
GOSUB2	D935	BGOTO	D93E	ASROMSET	D955	RTN.D9.6	D96A	BRETURN	D96B
BPOP	D96B	US.ERR	D97C	PULL3	D981	BDATA	D995	DATA2	D998
RTN.D9.A	D9A2	DATSCAN	D9A3	DATSCAN2	D9A6	SY.ERR2	D9C5	BIF	D9C9
BREM	D9DC	REM2	D9E1	BON	D9EC	RTN.DA.0	DA0B	LINGET	DA0C
LINGET2	DA12	BLET	DA46	LET2	DA63	PUTSTR	DA7B	COPYSTR	DAB7
RTN.DA.C	DACE	PRSTRING	DACF	BPRINT	DAD5	PRINT2	DAD7	UNARY2	DB32
LINEOUT	DB38	STROUT	DB3B	STRPRT	DB3E	PRTCR	DB50	OUTSPC	DB53
OUTPROMT	DB56	OUTCHR	DB58	INPUTERR	DB6F	READERR	DB79	ERRLINN	DB7D
SY.ERR3	DB81	INPERR	DB86	RESPERR	DB87	BGET	DBA0	BINPUT	DBB2
HEXTIN	DBDC	BREAD	DBE2	READ2	DBE9	INPTLIST	DBEB	INPTITEM	DBF1
INSTART	DC2B	INSTART2	DC69	INSTART3	DC72	INPTFLG	DC99	FINDATA	DCA0
INPTDONE	DCC7	MESG21	DCDF	MESG22	DCEF	BNEXT	DCF9	NEXT2	DCFB
RTN.DD.4	DD4C	FPCOMPT3	DD4D	FRMNUM	DD64	CHKNUM	DD67	CHKSTR	DD69
CHKVAL	DD6A	TM.ERR	DD73	NF.ERR	DD76	FRMEVAL	DD7B	FRMEVAL2	DD86
FRMEVAL3	DD95	FRMEVAL4	DD98	SAVOP	DDD7	FRMRECUR	DDFD	SY.ERR4	DE0D
FRMSTAK	DE10	FRMSTAK2	DE15	FRMSTAK3	DE23	NOTMATH	DE32	NOTMATH2	DE35
NOTMATH3	DE37	NOTMATH4	DE40	FRMELMNT	DE60	STRTXT	DE81	NOTFUNC	DE90
OEQUAL	DE98	FNFUNC	DEA7	SGNFUNC	DEAE	PARENCHK	DEB2	CHKCLSP	DEB8
CHKOPNP	DEBB	CHKCOM	DEBE	SYNTAXCHK	DEC0	SY.ERR	DEC9	MINUFUNC	DECE
EQLFUNC	DED0	GETIVAL	DED5	FSCREEN	DEF9	UNARY	DF09	UNARY3	DF3F
OOR	DF4F	OAND	DF55	FALSE	DF5D	TRUE	DF60	OLT	DF65
STRCMP	DF7D	NUMCMP	DFB0	FPDL	DFCD	BDIM	DFD9	PTRGET	DFE3
PTRGET2	DFE8	PTRGET3	DFEA	SY.ERR5	DFF4	BASIC	E000	BASIC2	E003
PTRGET4	E006	PTRGET5	E076	PTRGET6	E089	PTRGET7	E0CB	CHKASCI	E0DA
PNTARVAL	E0E3	STRSETUP	E0FF	IVALZERO	E105	FP8000	E107	MAKINT	E10C
AYPOSINT	E112	AYINT	E116	IQ.ERR2	E123	ARRAY	E128	BS.ERR	E198
IQ.ERR	E19B	RA.ERR	E19E	OD.ERR	E1A1	FINDELE	E24B	BS.ERR2	E269
OM.ERR2	E26C	RTN.E2.A	E2AC	MULSUBS	E2AD	MULSUBS2	E2B6	RTN.E2.D	E2DD
FFRE	E2DE	GIVAYFP	E2F2	FPOS	E2FF	SNGFLT	E301	CHKIFDIR	E306
UF.ERR	E30E	BDEF	E313	GETFNC	E341	CALLFNC	E354	FNCDATA	E3AF
FSTR	E3C5	STRINI	E3D0	STRSPA	E3D8	STRLIT	E3E2	STRLIT1	E3E3
STRLIT2	E3E9	PUTNEW	E426	FC.ERR	E449	OM.ERR3	E44C	GETSSPC	E454
GARBAG	E484	CHKVARS	E48D	CHKVARS2	E4BA	CHKARRYS	E4C2	GARBEXIT	E501
MOVVARS	E50C	MOVVARS2	E518	NXTVAR	E567	COPYVAR	E573	DECPTR	E58C
CAT2STR	E597	SL.ERR	E5CF	MOVINS	E5D4	MOVSTR	E5E2	MOVSTR2	E5E6
FRESTR	E5FD	FREFAC	E600	FRETMP	E604	FRETMS	E635	FCHR	E646
FLEFT	E65A	LEFT2	E660	LEFT3	E667	LEFT4	E668	FRIGHT	E686
FMID	E691	STRSET2	E6BC	FLEN	E6D6	GETSTRLN	E6DC	FASC	E6E5
IQ.ERR3	E6F2	GETBYTC	E6F5	GETBYT	E6F8	CONVINT	E6FB	FVAL	E707
STRCOPY	E73D	GETASNUM	E746	COMBYTE	E74C	GETADDR	E752	FPEEK	E764

BPOKE	E77B	BWAIT	E784	RTN.E7.9	E79F	FPPI	E7A1	FPIGUARD	E7A6
FSUB	E7A7	OMINUS	E7AA	SUB2	E7AA	FADD	E7BE	OPLUS	E7C1
ADD2	E7C1	ADD3	E7CA	COMPFAC1	E81D	NORMFAC1	E822	ZEROFAC	E842
ZEROFAC2	E844	ADD4	E849	NORMFAC2	E868	NORMFAC3	E874	NORMFAC4	E883
RTN.E8.9	E891	COMPFAC2	E892	COMPMANT	E898	INCMANT	E8BA	RTN.E8.D	E8C8
OF.ERR	E8C9	SHFTBYT1	E8CE	SHFTBYT2	E8E5	SHFTBITS	E8FC	PDL2	E908
FPLOGE	E913	FPSQR0.5	E918	FPSQR2.0	E91D	FPN0.5	E922	FPLN2	E927
POLY.LOG	E92C	FLN	E941	FMULT	E97F	OMULT	E982	MULT2	E982
TSTMULT	E9AA	BYTMULT	E9AF	LOADARG	E9E3	PROCEXP	EA10	PROCEXP2	EA12
PROCEXP3	EA28	ZEROFERR	EA2D	OF.ERR2	EA31	DZ.ERR	EA34	MULFAC10	EA39
FP10.0	EA4F	DIVFAC10	EA54	FDIV	EA66	ODIVIDE	EA69	DIV2	EA69
FP.25	EAD7	FP1.0E9	EADC	COPYM2F	EAE6	LOADFAC	EAF9	COPYF2T2	EB1E
COPYF2T1	EB21	COPYF2FR	EB27	COPYFAC	EB2B	COPYFAC2	EB2E	COPYA2F	EB53
COPYA2F2	EB55	COPYF2A	EB63	RNDUP	EB70	SIGNCHK	EB82	SIGNCHK2	EB86
SIGNCHK3	EB88	RTN.EB.8	EB8F	FSGN	EB90	FLOAT	EB93	FLOAT2	EB9B
FLOAT3	EBA0	FABS	EBAF	FPCOMP0	EBB2	FPCOMP	EBB5	FPCOMP2	EBB7
FP2INT	EBF2	FINT	EC23	CLRMANT	EC40	RTN.EC.4	EC49	GETINT	EC4A
GETINT2	EC61	GETINT3	EC87	GETINT4	ECA0	GETINT5	ECC0	GETINT6	ECD4
ADD2FAC	ECF6	PRTMSG19	ED0A	LINEPRT	ED18	MESG19	ED25	FP9.9E7	ED2A
FP9.9E8	ED2F	FPOUT	ED34	FPOUT2	EE49	SAV2STK	EE54	FPDECTBL	EE5C
DECTBLEN	0024	FSQR	EE8D	OPOWER	EE97	FEXP2	EECD	OGT	EED0
NEGFAC	EED0	RTN.EE.D	EEDA	FPINVLN2	EEDB	POLY.EXP	EEE0	FP1.0	EF04
FEXP	EF09	FLOG	EF3E	FPI	EF48	POLYSIN	EF57	POLYPROC	EF5B
POLYNOM	EF71	RTN.EF.A	EFA5	RANDVAL1	EFA6	RANDVAL2	EFAA	FRND	EFAE
FCOS	EFEA	FSIN	EFF1	SIN2	F025	FTAN	F03A	FPIDIV2	F05C
FP0.5	F061	POLY.SIN	F066	FPIMUL2	F099	FATAN	F09E	POLY.ATN	F0CC
PGZCODE	F109	CHRGET	00B1	CHRGOT	00B7	FPRAND	00C9	ZPCDLEN	001C
COLDSTRT	F125	FRMSTAK4	F1B1	COPYF2T3	F1BA	INCCOEF	F1C5	CLEARMUL	F1CC
BCALL	F1D5	BIN	F1DE	BPR	F1E5	PLOTFNS	F1EC	IQ.ERR4	F206
LINCOOR	F209	BPLOT	F225	BHLIN	F232	BVLIN	F241	BCOLOR	F24F
BVTAB	F256	BSPEED	F262	BTRACE	F26D	BNOTRACE	F26F	BNORMAL	F273
BINVERSE	F277	INVERSE2	F279	INVERSE3	F27B	BFLASH	F280	BHIMEM	F286
OM.ERR4	F2A3	BLOMEM	F2A6	BONERR	F2CB	HANDLERR	F2E9	BRESUME	F318
SY.ERR6	F32E	BDEL	F331	BGR	F390	BTEXT	F399	CXREAD	F39C
BHGR2	F3D8	BHGR	F3E2	CLRHIRE	F3EC	SETHIRE	F3EE	HPOSN	F411
HRPLOT	F457	HRMOVLF	F465	HRMOVLF2	F46E	HRMOVLF3	F478	COLSHIFT	F47E
HRMOVRT	F48A	DRAWHDR	F49C	XDRAWIT	F4A6	DRAWIT	F4B8	HRMOVUP	F4D1
HRMOVDN	F501	RTN.F5.2	F52C	BITBYT03	F52D	BITBYT04	F52E	BITBYT1C	F52F
BITABLE	F530	HLIN	F53A	ROTATBL	F5B3	IQ.ERR5	F5C4	DRAWCMD	F5C7
SQR2	F666	COPYA2F3	F68E	COPYF2A2	F693	COPYT32A	F6A8	GETFNS	F6B9
IQ.ERR6	F6E6	BHCOLOR	F6E9	RTN.F6.F	F6F5	HRCOLTBL	F6F6	BHPLOT	F6FE
HPlot2	F708	BROT	F721	BSCALE	F727	RND2	F72D	BDRAW	F769
BXDRAW	F76F	RND3	F775	TITLE	F791	TITLEN	000A	DELTITLE	001C
OFFTITLE	000E	PARSIEX1	F79B	PARSIEX2	F7A0	PARSIEX3	F7AE	LISTEX1	F7BE
LISTEX2	F7C6	PRTCRESX1	F7C6	PRTCRESX2	F7D5	PRTCRESX3	F7DC	RTN.F7.E	F7E6
BHTAB	F7E7	F8SPACE	F800	PLOT	F800	RTMASK	F80C	PLOT1	F80E
HLINE	F819	HLINE1	F81C	VLINE	F828	CLRSCR	F832	CLRTOP	F836
GBASCALC	F847	NXTCOL	F85F	SETCOL	F864	SCRN	F871	SCRN2	F879
INSDS1	F882	INSDS2	F88E	GETFMT	F8A5	TESTROM	F8B6	INSTDSP	F8D0
PRNTOP	F8D4	PRNTBL	F8DB	PRNTYX	F940	PRNTAX	F941	PRNTX	F944
PRBLNK	F948	PRBL2	F94A	PCADJ	F953	PCADJ2	F954	PCADJ3	F956
RTS2	F961	FMT1	F962	MNEML	F9A6	MNEMR	F9EB	CHAR1	FA30
CHAR2	FA36	CXOFF	FA3C	CXRTN	FA3F	OLDIRQ	FA40	NEWBREAK	FA47
BREAK	FA4C	OLDBRK	FA59	RESET	FA62	SWEET16	FA72	SW16RTN	FA78
RESET2	FA7E	NEWMON	FA81	NEWMON2	FA9B	PWRUP	FAA6	REGDSP	FAD7
REGDSP2	FADA	PWRCON	FAFD	PWRCNLEN	0005	DISKID	FB02	DISKIDLN	0006
RSETINIT	FB08	XLATBL	FB14	REGTBL	FB19	REGTBLN	0005	PREAD	FB1E
INIT	FB2F	INIT2	FB33	SETEXT	FB39	SETGR	FB40	SETWND	FB4B
TABV	FB5B	APPLE2	FB60	STITLE	FB65	SETPWRUP	FB6F	VIDWAIT	FB78
KBDWAIT	FB88	ESCOLD	FB97	ESCNEW	FBA5	SIGROM	FBB3	GOTOROM	FBB4
SIGBYTE	FBC0	BASCALC	FBC1	FMT2	FBC7	BELL2	FBD9	RINGBELL	FBD9

STORADV	FBF0	ADVANCE	FBF4	RTN.FB.F	FBFC	VIDOUT	FBFD	BS	FC10
UP	FC1A	VTAB	FC22	VTAB2	FC24	ESCOLD2	FC2C	CLREOP	FC42
NEWVW	FC46	HOME	FC58	GOTOROM2	FC5A	STEPRTN2	FC5F	CR	FC62
LF	FC66	SCROLL	FC70	GOTOIRQ	FC74	IRQDONE2	FC7A	CHKINV	FC95
CLREOL	FC9C	CLREOL2	FC9E	WAIT	FCA8	NEXTA4	FCB4	NEXTA1	FCBA
HEADR	FCC9	STEPRTN	FCCA	XERR	FCD2	FINDOP	FCE3	NXTLINE	FCF0
UPRMON	FCFD	UPRCASE	FD01	RDKEY	FD0C	RDKEY2	FD13	RDKEY3	FD18
KEYIN	FD1B	GOTOROM4	FD1D	RDESC	FD21	NEWRDKEY	FD28	ESC	FD2F
RDCHAR	FD35	PICKFIX	FD3D	NOTCR	FD47	CANCEL	FD62	GETLINE2	FD67
GETLINE	FD6A	BCKSPC	FD71	NXTCHAR	FD75	ADDINP	FD84	CROUT	FD8E
PRA1	FD92	PRXY	FD96	XAM8	FDA3	MOD8CHK	FDAD	XAM	FDB3
DATAOUT	FDB6	XAMPM	FDC6	ADD	FDD1	PRBYTE	FDDA	PRHEX	FDE3
PRHEX2	FDE5	COUT	FDED	COUT2	FDF0	COUTZ	FDF6	COUTZ2	FDF7
BL1	FE00	BLANK	FE04	STOR	FE0B	STOR2	FE0F	NEXTA3	FE11
SETMODE	FE18	SETMDZ	FE1D	LT	FE20	MOVE	FE2C	VERIFY	FE36
LIST	FE5E	A1PC	FE75	A1PCLP	FE78	SETINV	FE80	SETNORM	FE84
SETKBD	FE89	INPORT	FE8B	INPORT2	FE8D	SETVID	FE93	OUTPORT	FE95
OUTPORT2	FE97	IOPORT	FE9B	ZAPMEM	FEAE	XBASIC	FEB0	BASCONT	FEB3
GO	FEB6	REGZ	FEBF	TRACE	FEC2	STEPZ	FEC4	USR	FECA
WRITE	FECD	CHRTBLX3	FECE	SEARCH	FED2	MINIASM	FEF1	CRMON	FEF6
CRMON2	FEF9	READ	FEFD	ZAPMEM2	FF05	RTN.FF.0	FF0E	CHRTBLX2	FF0F
LOOKASC	FF18	PRNTERR	FF2D	BELL	FF3A	RESTORE	FF3F	SAVE	FF4A
SAVE2	FF4C	IORTS	FF58	OLDRST	FF59	CHRTBLX1	FF5F	MON	FF65
MON2	FF69	NEXTITM	FF73	CHRSRCH	FF7A	DIG	FF8A	NEXTBIT	FF90
GETNUM	FFA7	NEXTCHR	FFAD	TOSUBR	FFBE	ZMODE	FFC7	CHRTBL	FFCC
SUBTBL	FFE3								

Symbols alphabetically sorted:

A1H	003D	A1L	003C	A1PC	FE75	A1PCLP	FE78	A2H	003F
A2L	003E	A3H	0041	A3L	0040	A4H	0043	A4L	0042
A5L	0044	ACH	0051	ACL	0050	ADD	FDD1	ADD2	E7C1
ADD2FAC	ECF6	ADD3	E7CA	ADD4	E849	ADDCMD	C76C	ADDINP	FD84
ADVANCE	FBF4	AHDRLEN	0005	ALTCHOFF	C00E	ALTCHON	C00F	ANN1OFF	C058
ANN2OFF	C05A	ANN3ON	C05D	ANN4ON	C05F	APPLE2	FB60	AREG	0045
ARGEXP	00A5	ARGGUARD	0092	ARGMANT	00A6	ARGSIGN	00AA	ARRAY	E128
ARYPNT	0094	ARYTAB	006B	ASCIBELL	0087	ASENTER	D4F2	ASROMCLR	D665
ASROMERR	D865	ASROMNEW	D7D2	ASROMRST	D4F2	ASROMSET	D955	ASROMWRM	D43C
AUTOBRK	03EF	AUTORSET	03F2	AUXMOVE	C311	AUXZPOFF	C008	AUXZPON	C009
AVARLEN	0003	AYINT	E116	AYPOSINT	E112	BAS2H	002B	BAS2L	002A
BASADDR	D000	BASCALC	FBC1	BASCINIT	C803	BASCONT	FEB3	BASH	0029
BASIC	E000	BASIC2	E003	BASICENT	C317	BASICIN	C305	BASICINT	C300
BASICOUT	C307	BASL	0028	BASNAME	D0D4	BCALL	F1D5	BCCMD	C7AE
BCKSPC	FD71	BCLEAR	D66A	BCOLOR	F24F	BCONT	D896	BDATA	D995
BDEF	E313	BDEL	F331	BDIM	DFD9	BDRAW	F769	BELL	FF3A
BELL2	FBD9	BEND	D870	BFLASH	F280	BFOR	D766	BGET	DBA0
BGOSUB	D921	BGOTO	D93E	BGR	F390	BHCOLOR	F6E9	BHGR	F3E2
BHGR2	F3D8	BHIMEM	F286	BHLIN	F232	BHPLOT	F6FE	BHTAB	F7E7
BIF	D9C9	BIN	F1DE	BINPUT	DBB2	BINVERSE	F277	BIORET	C8C5
BITABLE	F530	BITBYT03	F52D	BITBYT04	F52E	BITBYT1C	F52F	BITBYT60	CA2E
BL1	FE00	BLANK	FE04	BLET	DA46	BLIST	D6A5	BLOAD	D8C9
BLOMEM	F2A6	BLTU	D393	BM1CMD	C7C9	BMCMD	C7B6	BNCCMD	C79D
BNEW	D649	BNEXT	DCF9	BNM1CMD	C7D2	BNORMAL	F273	BNOTRACE	F26F
BNZCMD	C7C2	BON	D9EC	BONERR	F2CB	BPCMD	C7B1	BPLOT	F225
BPOKE	E77B	BPOP	D96B	BPR	F1E5	BPRINT	DAD5	BRANCH	C5B9
BRANCH2	C5C7	BRCMD	C79C	BREAD	DBE2	BREAK	FA4C	BREM	D9DC
BRESTORE	D849	BRESUME	F318	BRETURN	D96B	BROT	F721	BRTBL	C6E2
BRUN	D912	BS	FC10	BS.ERR	E198	BS.ERR2	E269	BSCALE	F727
BSCMD	C790	BSDATA	D006	BSFOR	D002	BSGOSUB	D060	BSGOTO	D056
BSNSCMD	C70D	BSNSCMD2	C6D7	BSPEED	F262	BSPOP	D042	BSPRINT	D074

BSREM	D064	BSTOP	D86E	BTEXT	F399	BTRACE	F26D	BVLIN	F241
BVTAB	F256	BWAIT	E784	BXDRAW	F76F	BYTEBITS	0008	BYTMULT	E9AF
BYTVALUE	000D	BZCMD	C7BB	C3HOOKS	C82A	C3IN	C832	C3ROMOFF	C00A
C3ROMON	C00B	C3SPACE	C300	CALLFNC	E354	CANCEL	FD62	CAT2STR	E597
CH	0024	CHAR1	FA30	CHAR2	FA36	CHARAC	000D	CHKARRYS	E4C2
CHKASCI	E0DA	CHKCLSP	DEB8	CHKCOM	DEBE	CHKIFDIR	E306	CHKINV	FC95
CHKNUM	DD67	CHKOPNP	DEBB	CHKSTR	DD69	CHKSUM	002E	CHKVAL	DD6A
CHKVARS	E48D	CHKVARS2	E4BA	CHRGET	00B1	CHRGOT	00B7	CHRGTADR	00B1
CHRSRCH	FF7A	CHRTBL	FFCC	CHRTBLX1	FF5F	CHRTBLX2	FF0F	CHRTBLX3	FECE
CKSTKSIZ	D3D6	CKSTRSIZ	D3E3	CLEARC	D66C	CLEARMUL	F1CC	CLR2	CCD2
CLR40	CCA5	CLR80	CCBA	CLREOL	FC9C	CLREOL2	FC9E	CLREOP	FC42
CLRHAF	CCAD	CLRHIRE	F3EC	CLRKEY	C010	CLRMANT	EC40	CLRROM	CFFF
CLRSCR	F832	CLRTOP	F836	COEFNUM	00A3	COEFPTR	00AD	COLBITS	001C
COLCOUNT	001D	COLDSTRT	F125	COLOR	0030	COLSHIFT	F47E	COMBYTE	E74C
COMPFAC1	E81D	COMPFAC2	E892	COMPMANT	E898	CONVINT	E6FB	COPYA2F	EB53
COPYA2F2	EB55	COPYA2F3	F68E	COPYF2A	EB63	COPYF2A2	F693	COPYF2FR	EB27
COPYF2T1	EB21	COPYF2T2	EB1E	COPYF2T3	F1BA	COPYFAC	EB2B	COPYFAC2	EB2E
COPYM2F	EAE6	COPYROM	CEF4	COPYSTR	DAB7	COPYT32A	F6A8	COPYVAR	E573
COUNTER	0099	COUT	FDED	COUT2	FDF0	COUTZ	FDF6	COUTZ2	FDF7
CPRCMD	C77C	CPRMASK	0016	CPRTYPE	0089	CR	FC62	CRMON	FEF6
CRMON2	FEF9	CROUT	FD8E	CSETUP	C850	CSWH	0037	CSWL	0036
CTRLADRH	CBB9	CTRLADRL	CB9E	CTRLB	0082	CTRLC	0083	CTRLCHR	CAD6
CTRLCHR2	CAD2	CTRLE	0085	CTRLH	0088	CTRLJ	008A	CTRLK	008B
CTRLON	C8BD	CTRLP	0090	CTRLS	0093	CTRLU	0095	CTRLX	0098
CTRLXFER	CB07	CTRLY	0099	CURLIN	0075	CURLINSV	00F6	CV	0025
CXEXIT	C208	CXKEYIN	C98D	CXNEWVW	C870	CXNEWVW2	C874	CXOFF	FA3C
CXREAD	F39C	CXRESET	C2DD	CXROMOFF	C006	CXROMON	C007	CXRTN	FA3F
CXSTEP	C508	CXVIDCK3	C87C	CXVIDCK4	C87E	DARROW	008A	DATA2	D998
DATAFLG	0013	DATAOUT	FDB6	DATLIN	007B	DATPTR	007D	DATSCAN	D9A3
DATSCAN2	D9A6	DCRCMD	C7F7	DECPTR	E58C	DECTBLEN	0024	DELTITLE	001C
DEST	0060	DFLTDIM	000B	DIG	FF8A	DIMFLG	0010	DISKID	FB02
DISKIDLN	0006	DIV2	EA69	DIVFAC10	EA54	DO40	CD2B	DOBASL	CA1F
DOCTRL.C	D863	DOCXCMD	C100	DOHANDLR	D8B0	DOMOVE	C376	DOSTAMT	D828
DOSTAMT2	D82A	DOSTAMT3	D842	DOTRACE	D805	DOXFER	C3C3	DPFLAG	009B
DRAWCMD	F5C7	DRAWHDR	F49C	DRAWIT	F4B8	DSCPTR	008C	DSCTMP	009D
DZ.ERR	EA34	END2	D871	END3	D88A	ENDCHR	000E	EOLPTR	000F
EQLFUNC	DED0	ERRFLG	00D8	ERRLIN	00DA	ERRLINN	DB7D	ERRNUM	00DE
ERROR.1	00FE	ERROR.2	00FF	ERRPOS	00DC	ERRSTK	00DF	ESC	FD2F
ESCAPE	009B	ESCCHAR	C96B	ESCNEW	FBA5	ESCOFF	CEC4	ESCOLD	FB97
ESCOLD2	FC2C	ESCON	CEB1	ESCRTN	CECD	ESCTABL	C97C	EXPBIAS	0080
EXPCOUNT	009A	EXPSIGN	009C	EXTSIGN	00A4	F8SPACE	F800	FABS	EBAF
FACEXP	009D	FACGUARD	00AC	FACLEN	0005	FACMANT	009E	FACSIGN	00A2
FACSIZE	0006	FADD	E7BE	FALSE	DF5D	FASC	E6E5	FATAN	F09E
FC.ERR	E449	FCHR	E646	FCOS	EFEA	FDIV	EA66	FEXP	EF09
FEXP2	EECD	FFRE	E2DE	FINDATA	DCA0	FINDELE	E24B	FINDOP	FCE3
FINT	EC23	FIRST	00F0	FLASH	007F	FLASHBYT	00F3	FLEFT	E65A
FLEN	E6D6	FLN	E941	FLOAT	EB93	FLOAT2	EB9B	FLOAT3	EBA0
FLOG	EF3E	FMID	E691	FMT1	F962	FMT2	FBC7	FMULT	E97F
FN1ADDR	D080	FNCDATA	E3AF	FNDLIN	D61A	FNDLIN2	D61E	FNFUNC	DEA7
FORM	C4C8	FORMAT	002E	FORPNT	0085	FP.25	EAD7	FP0.5	F061
FP1.0	EF04	FP1.0E9	EADC	FP10.0	EA4F	FP2INT	EBF2	FP8000	E107
FP9.9E7	ED2A	FP9.9E8	ED2F	FPCDMOD	0001	FPCOMP	EBB5	FPCOMP0	EBB2
FPCOMP2	EBB7	FPCOMPT3	DD4D	FPDECTBL	EE5C	FPDL	DFCD	FPEEK	E764
FPI	EF48	FPIDIV2	F05C	FPIGUARD	E7A6	FPIMUL2	F099	FPINVLN2	EEDB
FPLN2	E927	FPLOGE	E913	FPN0.5	E922	FPOS	E2FF	FPOUT	ED34
FPOUT2	EE49	FPPI	E7A1	FPRAND	00C9	FPSQR0.5	E918	FPSQR2.0	E91D
FRAMSIZE	0014	FREFAC	E600	FRESPC	0071	FRESTR	E5FD	FRETMP	E604
FRETMS	E635	FRETOP	006F	FRIGHT	E686	FRMELMNT	DE60	FRMEVAL	DD7B
FRMEVAL2	DD86	FRMEVAL3	DD95	FRMEVAL4	DD98	FRMNUM	DD64	FRMRECUR	DDFD
FRMSTAK	DE10	FRMSTAK2	DE15	FRMSTAK3	DE23	FRMSTAK4	F1B1	FRND	EFAE
FS1SCRN	D08A	FS1SGN	D080	FS2LEFT	D0AC	FS3PI	D0B2	FSCREEN	DEF9

FSGN	EB90	FSIN	EFF1	FSQR	EE8D	FSTR	E3C5	FSUB	E7A7
FTAN	F03A	FULL40	CD6A	FULL80	CD6E	FUNCNAM	008A	FVAL	E707
GARBAG	E484	GARBEXIT	E501	GARFLG	0013	GBASCALC	F847	GBASH	0027
GBASL	0026	GC1IN	C064	GCTOGL	C070	GETADDR	E752	GETASNUM	E746
GETBYT	E6F8	GETBYTC	E6F5	GETCHR	D758	GETFMT	F8A5	GETFNC	E341
GETFNS	F6B9	GETINT	EC4A	GETINT2	EC61	GETINT3	EC87	GETINT4	ECA0
GETINT5	ECC0	GETINT6	ECD4	GETIVAL	DED5	GETLINE	FD6A	GETLINE2	FD67
GETNSP	C500	GETNUM	FFA7	GETSSPC	E454	GETSTRLN	E6DC	GIVAYFP	E2F2
GO	FEB6	GOEND3	D826	GOODF8	0006	GOSTROUT	0003	GOSUB2	D935
GOTOIRQ	FC74	GOTOROM	FBB4	GOTOROM2	FC5A	GOTOROM4	FD1D	GOUSR	000A
GOWARM	0000	GTFORPNT	D362	H2	002C	HANDLERR	F2E9	HEADR	FCC9
HEXTIN	DBDC	HIGHDS	0094	HIGHTR	0096	HIRESOFF	C056	HIRESON	C057
HLIN	F53A	HLINE	F819	HLINE1	F81C	HLINMOD	0001	HMASK	0030
HOME	FC58	HPLOT2	F708	HPOSN	F411	HRCOLCNT	00EA	HRCOLOR	00E4
HRCOLTBL	F6F6	HRFLAG	00D3	HRHORZ	00E5	HRMOVDN	F501	HRMOVLF	F465
HRMOVLF2	F46E	HRMOVLF3	F478	HRMOVRT	F48A	HRMOVUP	F4D1	HRPAG	00E6
HRPLOT	F457	HRROT	00F9	HRSCALE	00E7	HRSHPTBL	00E8	HRWORK	00D4
HRXCOOR	00E0	HRXDELTA	00D0	HRYCOOR	00E2	HRYDELTA	00D2	INCCOEF	F1C5
INCMANT	E8BA	INDEX	005E	INIT	FB2F	INIT2	FB33	INITBL	C5CD
INITBLN	0008	INLIN	D52C	INLIN2	D52E	INPERR	DB86	INPORT	FE8B
INPORT2	FE8D	INPTDONE	DCC7	INPTFLG	DC99	INPTITEM	DBF1	INPTLIST	DBEB
INPUT	0200	INPUTERR	DB6F	INPUTFLG	0015	INRCMD	C733	INSDS1	F882
INSDS2	F88E	INSTART	DC2B	INSTART2	DC69	INSTART3	DC72	INSTDSP	F8D0
INVERSE	003F	INVERSE2	F279	INVERSE3	F27B	INVERT	CE26	INVFLG	0032
INVRSE80	007F	IOPORT	FE9B	IORTS	FF58	IOSPACE	C000	IQ.ERR	E19B
IQ.ERR2	E123	IQ.ERR3	E6F2	IQ.ERR4	F206	IQ.ERR5	F5C4	IQ.ERR6	F6E6
IRAND	00C9	IRQDONE	C3F4	IRQDONE2	FC7A	IRQRTN	C3FA	ISCNTLC	D858
ISO	C1A0	IVALZERO	E105	IVARLEN	0002	JBASINIT	C347	JC8	C344
JMPADRS	0090	JPINIT	C34A	JPREAD	C350	JPSTAT	C35C	JPWRITE	C356
KBDOUT	C2F3	KBDTBL	C2EF	KBDWAIT	FB88	KEY	C000	KEYIN	FD1B
KSWH	0039	KSWL	0038	LARROW	0088	LASTIN	002F	LASTOP	0087
LASTPT	0053	LD@CMD	C721	LDCMD	C70F	LDD@CMD	C73A	LEFT2	E660
LEFT3	E667	LEFT4	E668	LEN	0094	LENGTH	002F	LET2	DA63
LF	FC66	LINCOOR	F209	LINEOUT	DB38	LINEPRT	ED18	LINGET	DA0C
LINGET2	DA12	LINNUM	0050	LIST	FE5E	LIST2	D6DA	LIST3	D6FC
LIST4	D700	LIST5	D721	LIST6	D727	LISTEX1	F7BE	LISTEX2	F7C6
LMNEM	002C	LOADARG	E9E3	LOADFAC	EAF9	LOC0	0000	LOC1	0001
LOC2	0002	LOOKASC	FF18	LOWTR	009B	LT	FE20	LWRMASK	00DF
M.1	0002	M.2	0004	M.4	0010	M.6	0040	M.CTL	0008
M.CTL2	0020	M.CURSOR	0010	M.GOXY	0008	M.MOUSE	0001	M.PAS1.0	0002
M.PASCAL	0080	M.VMODE	0004	MACSTAT	0044	MAKINT	E10C	MANTBITS	0020
MANTLEN	0004	MASK	002E	MASKIRQ	03FE	MAXINPUT	00EF	MAXPDL	0003
MEMSIZE	0073	MESG01	D25B	MESG02	D26B	MESG03	D271	MESG04	D285
MESG05	D290	MESG06	D2A0	MESG07	D2A8	MESG08	D2B5	MESG09	D2C8
MESG10	D2D5	MESG11	D2E4	MESG12	D2F4	MESG13	D302	MESG14	D30F
MESG15	D31E	MESG16	D331	MESG17	D340	MESG18	D352	MESG19	ED25
MESG20	D35A	MESG21	DCDF	MESG22	DCEF	MESGS	D25B	MINIASM	FEF1
MINUFUNC	DECE	MINUSLOC	00A3	MIXEDOFF	C052	MIXEDON	C053	MNEML	F9A6
MNEMR	F9EB	MOD8CHK	FDAD	MODE	0031	MON	FF65	MON2	FF69
MOUSEOFF	CD3A	MOUSEON	CD41	MOVE	FE2C	MOVEMASK	0003	MOVINS	E5D4
MOVSTR	E5E2	MOVSTR2	E5E6	MOVVARS	E50C	MOVVARS2	E518	MSBCLR	007F
MSBSET	0080	MSLOT	07F8	MULFAC10	EA39	MULGUARD	0066	MULMANT	0062
MULSUBS	E2AD	MULSUBS2	E2B6	MULT2	E982	NEGFAC	EED0	NEGONE	00FF
NEGTWO	00FE	NEWBREAK	FA47	NEWIRQ	C47C	NEWMON	FA81	NEWMON2	FA9B
NEWRDKEY	FD28	NEWSTT	D7D2	NEWVW	FC46	NEXT2	DCFB	NEXTA1	FCBA
NEXTA3	FE11	NEXTA4	FCB4	NEXTBIT	FF90	NEXTCHR	FFAD	NEXTITM	FF73
NF.ERR	DD76	NMASKIRQ	03FB	NORMFAC1	E822	NORMFAC2	E868	NORMFAC3	E874
NORMFAC4	E883	NOTCR	FD47	NOTFUNC	DE90	NOTMATH	DE32	NOTMATH2	DE35
NOTMATH3	DE37	NOTMATH4	DE40	NUMCMP	DFB0	NUMDIM	000F	NXTCHAR	FD75
NXTCOL	F85F	NXTLINE	FCF0	NXTLINE2	CF9C	NXTVAR	E567	OAND	DF55
OD.ERR	E1A1	ODIVIDE	EA69	OEQUAL	DE98	OF.ERR	E8C9	OF.ERR2	EA31

OFFSET	0061	OFFTITLE	000E	OGT	EED0	OLDBASH	07FB	OLDBASL	077B
OLDBRK	FA59	OLDCH	047B	OLDIRQ	FA40	OLDLIN	0077	OLDRST	FF59
OLT	DF65	OM.ERR	D410	OM.ERR2	E26C	OM.ERR3	E44C	OM.ERR4	F2A3
OMINUS	E7AA	OMULT	E982	OOR	DF4F	OPLUS	E7C1	OPOWER	EE97
OPRND	0044	OPTBL	C6E3	OPTBLC	CA71	OPTBLL	CA7D	OTV.ADD	0079
OTV.AND	0050	OTV.MUL	007B	OTV.NEQ	007F	OTV.OR	0046	OTV.PWR	007D
OTV.REL	0064	OURCH	057B	OURCV	05FB	OUTCHR	DB58	OUTPORT	FE95
OUTPORT2	FE97	OUTPROMT	DB56	OUTSPC	DB53	PAGE08	0800	PAGE0C	0C00
PAGE10	1000	PAGE1ON	C054	PAGE20	2000	PAGE2ON	C055	PAGE40	4000
PAGEBD	BD00	PAGEBF	BF00	PAGEC0	C000	PAGEC1	C100	PAGEC8	C800
PAGEDO	D000	PAGEF0	F000	PAGEFE	FE00	PAGESIZE	0100	PARENCHK	DEB2
PARSE	D56C	PARSE2	D56D	PARSE3	D5A8	PARSE4	D5CD	PARSE5	D5F2
PARSE6	D5F9	PARSE7	D610	PARSIEX1	F79B	PARSIEX2	F7A0	PARSIEX3	F7AE
PARSINPT	D559	PASINV	CE1F	PB1IN	C061	PB2IN	C062	PCADJ	F953
PCADJ2	F954	PCADJ3	F956	PCH	003B	PCINC2	C58C	PCINC3	C58E
PCL	003A	PCUROFF	CB18	PCURON	CB0D	PDL2	E908	PG1TXLOC	05B0
PGZCODE	F109	PICK	CE44	PICKFIX	FD3D	PLOT	F800	PLOT1	F80E
PLOTFNS	F1EC	PLOTMASK	0004	PNTARVAL	E0E3	POLY.ATN	F0CC	POLY.EXP	EEE0
POLY.LOG	E92C	POLY.SIN	F066	POLYNOM	EF71	POLYPROC	EF5B	POLYSIN	EF57
POP@CMD	C753	POP@CMD	C74C	PPINIT	C9B4	PPREAD	C9D6	PPWRITE	C9F0
PRA1	FD92	PRBL2	F94A	PRBLNK	F948	PRBYTE	FDDA	PREAD	FB1E
PREG	0048	PRGEND	00AF	PRGTAB	0067	PRHEX	FDE3	PRHEX2	FDE5
PRINT2	DAD7	PRLINUM	D431	PRNTAX	F941	PRNTBL	F8DB	PRNTERR	FF2D
PRNTOP	F8D4	PRNTX	F944	PRNTYX	F940	PROCESS	0095	PROCEXP	EA10
PROCEXP2	EA12	PROCEXP3	EA28	PROCSPCL	C64E	PROCVAR	C600	PROMPT	0033
PRSTRING	DACF	PRTCR	DB50	PRTCRESX1	F7C6	PRTCRESX2	F7D5	PRTCRESX3	F7DC
PRTERR	D412	PRTMSG19	ED0A	PRXY	FD96	PSETUP	CED4	PTRGET	DFE3
PTRGET2	DFE8	PTRGET3	DFA	PTRGET4	E006	PTRGET5	E076	PTRGET6	E089
PTRGET7	E0CB	PULL3	D981	PULL3A	D8B3	PUTNEW	E426	PUTSTR	DA7B
PWRCNLEN	0005	PWRCON	FAFD	PWRITER	CA29	PWRSTATE	03F4	PWRUP	FAA6
PWRUPBYT	00A5	PXINIT	C800	PXINIT2	C9B0	PXLSBYTE	0007	PXREAD	C84D
PXWRITE	C9AA	QUIT	CD7D	R0H	0001	R0L	0000	R12H	0019
R12L	0018	R14H	001D	R14L	001C	R15H	001F	R15L	001E
RA.ERR	E19E	RAM1WE	C08B	RAM1WP	C088	RAM2WE	C083	RAM2WP	C080
RAMRDOFF	C002	RAMRDON	C003	RAMSWTBL	C4C1	RAMWROFF	C004	RAMWRON	C005
RANDVAL1	EFA6	RANDVAL2	EFAA	RARROW	0095	RD2BIT	D8FF	RDALTCH	C01E
RDAUXZP	C016	RDBANK2	C011	RDBIT	D902	RDBYTE	D8BB	RDC3ROM	C017
RDCHAR	FD35	RDCXROM	C015	RDESC	FD21	RDHIRES	C01D	RDKEY	FD0C
RDKEY2	FD13	RDKEY3	FD18	RDLCRAM	C012	RDMIXED	C01B	RDPAGE2	C01C
RDRAMRD	C013	RDRAMWR	C014	RDSTR80	C018	RDTEXT	C01A	RDVID80	C01F
RDVRTBLK	C019	READ	FEFD	READ2	DBE9	READERR	DB79	REGDSP	FAD7
REGDSP2	FADA	REGTBL	FB19	REGTBLEN	0005	REGZ	FEBF	REL	CF3A
REM2	D9E1	REMSTK	00F8	RESET	FA62	RESET2	FA7E	RESPERR	DB87
RESTART	D43C	RESTORE	FF3F	RETURN	008D	RINGBELL	FBDD	RMNEM	002D
RND2	F72D	RND3	F775	RNDH	004F	RNDL	004E	RNDUP	EB70
ROM2WE	C081	ROM2WP	C082	ROTATBL	F5B3	ROTHSUM	00D4	ROTHVAL	00D2
ROTQVAL	00D1	ROTVSUM	00D5	ROTVVAL	00D3	RSCMD	C70B	RSCMD2	C6C6
RSETINIT	FB08	RSNSCMD	C7E0	RTMASK	F80C	RTN.D6.4	D648	RTN.D6.9	D696
RTN.D6.A	D6A4	RTN.D8.5	D857	RTN.D8.A	D8AF	RTN.D9.6	D96A	RTN.D9.A	D9A2
RTN.DA.0	DA0B	RTN.DA.C	DACE	RTN.DD.4	DD4C	RTN.E2.A	E2AC	RTN.E2.D	E2DD
RTN.E7.9	E79F	RTN.E8.9	E891	RTN.E8.D	E8C8	RTN.EB.8	EB8F	RTN.EC.4	EC49
RTN.EE.D	EEDA	RTN.EF.A	EFA5	RTN.F5.2	F52C	RTN.F6.F	F6F5	RTN.F7.E	F7E6
RTN.FB.F	FBFC	RTN.FF.0	FF0E	RTNADR	0091	RTNCMD	C701	RTS2	F961
RUN2	D91E	RUNFLAG	00D6	SAV2STK	EE54	SAVE	FF4A	SAVE2	FF4C
SAVOP	DDD7	SAVY	00AD	SCMDMASK	0007	SCRN	F871	SCRN2	F879
SCRN48	CDC1	SCRN84	CD8E	SCRNRET	CDF5	SCROLL	FC70	SCROLLDN	CBD4
SCROLLUP	CBEB	SCRTH	D64B	SEARCH	FED2	SETC8	C36D	SETCMD	C709
SETCMD2	C6B2	SETCOL	F864	SETCOUT2	CD61	SETDAPTR	D853	SETEXT	FB39
SETGR	FB40	SETHIRES	F3EE	SETINV	FE80	SETKBD	FE89	SETKEYIN	CD58
SETMDZ	FE1D	SETMODE	FE18	SETNORM	FE84	SETPTRS	D665	SETPWRUP	FB6F
SETTOP	CD2E	SETVID	FE93	SETWND	FB4B	SGNFUNC	DEAE	SHAPE	001A

SHFTBITS	E8FC	SHFTBYT1	E8CE	SHFTBYT2	E8E5	SHPOLD	00D7	SHPVAL	00D0
SIGBYTE	FBC0	SIGN	002F	SIGNCHK	EB82	SIGNCHK2	EB86	SIGNCHK3	EB88
SIGNFLG	00CD	SIGROM	FBB3	SIN2	F025	SJMPCMD	C7E9	SL.ERR	E5CF
SNGFLT	E301	SOUTCMD	C7DB	SPACE	00A0	SPCLFLAG	00CD	SPEEDBYT	00F1
SPKRTOGL	C030	SPNT	0049	SQR2	F666	SRCPTR	007F	SROTMASK	000F
ST@CMD	C72B	ST@CMD2	C72D	STACK	0100	STCMD	C718	STD@CMD	C743
STEP	D7AF	STEPRTN	FCCA	STEPRTN2	FC5F	STEPZ	FEC4	STITLE	FB65
STKINIT	D683	STOR	FE0B	STOR2	FE0F	STORADV	FBF0	STORCHAR	CE38
STORIT	CE70	STP@CMD	C763	STR80OFF	C000	STR80ON	C001	STRCMP	DF7D
STRCOPY	E73D	STREND	006D	STRING1	00AB	STRING2	00AD	STRINI	E3D0
STRLIT	E3E2	STRLIT1	E3E3	STRLIT2	E3E9	STROUT	DB3B	STRPRT	DB3E
STRSET2	E6BC	STRSETUP	E0FF	STRSPA	E3D8	STRTXT	DE81	STXTPTR	D697
SUB2	E7AA	SUBCMD	C77A	SUBFLG	0014	SUBTBL	FFE3	SVARLEN	0007
SW16	C670	SW16B	C679	SW16C	C67F	SW16RTN	FA78	SWEET16	FA72
SWTBLEN	0006	SY.ERR	DEC9	SY.ERR2	D9C5	SY.ERR3	DB81	SY.ERR4	DE0D
SY.ERR5	DFF4	SY.ERR6	F32E	SYNTAXCHK	DEC0	T2GUARD	0045	T3GUARD	008F
TABV	FB5B	TAG.EQU	D0CE	TAG.NEG	D0CB	TAG.REL	D0D1	TAGADDR	D0B6
TAPEIN	C060	TAPEOUT	C020	TEMP1	0093	TEMP2	0098	TEMP3	008A
TEMPPT	0052	TEMPST	0055	TESTCARD	CA90	TESTROM	F8B6	TEXTOFF	C050
TEXTON	C051	TEXTPG1	0400	TEXTPTR	0079	TITLE	F791	TITLEN	000A
TK.AT	00C5	TK.EQUAL	00D0	TK.FN	00C2	TK.GRTR	00CF	TK.MINUS	00C9
TK.NOT	00C6	TK.PLUS	00C8	TK.SCRN	00D7	TK.SGN	00D2	TK.SPC	00C3
TK.STEP	00C7	TK.TAB	00C0	TK.THEN	00C4	TK.TO	00C1	TM.ERR	DD73
TOBR	C6A2	TOGLFLG	0016	TOKNCNTR	000F	TOSUBR	FFBE	TRACE	FEC2
TRACEFLG	00F2	TRUE	DF60	TRYNEXT	CF6B	TSTMULT	E9AA	TSTROMCD	CA89
TXPTRSAV	0087	TXTPTR	00B8	TXTPTRSV	00F4	UARROW	008B	UF.ERR	E30E
UNARY	DF09	UNARY2	DB32	UNARY3	DF3F	UP	FC1A	UPRCASE	FD01
UPRMON	FCFD	US.ERR	D97C	USR	FECA	USRAHAND	03F5	USRYHAND	03F8
V2	002D	VALTYP	0011	VARNAM	0081	VARPNT	0083	VARPTR	00A0
VARTAB	0069	VERIFY	FE36	VID80OFF	C00C	VID80ON	C00D	VIDOUT	FBFD
VIDWAIT	FB78	VLIN	F828	VTAB	FC22	VTAB2	FC24	WAIT	FCA8
WNBDM	0023	WNDLFT	0020	WNDTOP	0022	WNDWDTH	0021	WRITE	FECF
XAM	FDB3	XAM8	FDA3	XAMPM	FDC6	XBASCALC	CABA	XBASCLC	C1B6
XBASIC	FEB0	XBS	CB40	XCHAR	067B	XCLREOL	C1F1	XCLREOL1	C1F3
XCLREOL2	C160	XCLREOP	C103	XCOORD	06FB	XCR	CB51	XDC1	CCE7
XDC1.2	CCEC	XDC2	CCF9	XDOCMD	C211	XDRAWIT	F4A6	XEM	CB5F
XERR	FCD2	XFER	C314	XFERADR	03ED	XFF	CC74	XFS	CB6B
XGETFMT	C1D5	XGETFMT2	C5D5	XGOMINI	C1E1	XGS	CC97	XGSEOLZ	CC9A
XHOME	C119	XINPUT	C8E6	XIOPORT	C195	XJMP	C5A3	XJMPAT	C5A4
XJMPX	C575	XJSR	C598	XKEYIN	C25B	XLATBL	FB14	XLIF	CBD8
XMODE	04FB	XNAK	CD4A	XNEWVW	C1CF	XORSIGN	00AB	XPICKFIX	C1E8
XRDESC	C1BD	XRDKEY	C1A9	XRDKEY2	CE11	XREG	0046	XRESET	C176
XRESETX	C2B5	XRTI	C584	XRTS	C588	XSCROLL	C123	XSETWND	C152
XSETWNDX	C2A5	XSI	CB8F	XSO	CB84	XSUB	CC93	XTEMP1	077B
XTEMP2	07FB	XUS	CB79	XVT	CC77	XVTAB	C203	XVTAB2	CDFB
XVTAB3	CE00	YCLREOL	C168	YCLREOL2	C16B	YCLREOP	C170	YGETFMT	C1D5
YGOMINI	C1E1	YHOME	C17C	YIOPORT	C18A	YPICKFIX	C1E8	YRDKEY	C179
YREG	0047	YREGTBLX	C243	YREGTBLY	C24F	YRESET	C176	YSAV	0034
YSAV1	0035	YSCROLL	C165	YSETWND	C173	ZAPMEM	FEAE	ZAPMEM2	FF05
ZERO	0000	ZEROFAC	E842	ZEROFAC2	E844	ZEROFERR	EA2D	ZMODE	FFC7
ZPCDLEN	001C								

Symbols numerically sorted:

ZERO	0000	ROL	0000	LOC0	0000	GOWARM	0000	ROH	0001
M.MOUSE	0001	LOC1	0001	HLINMOD	0001	FPCDMOD	0001	M.PAS1.0	0002
M.1	0002	LOC2	0002	IVARLEN	0002	MOVEMASK	0003	MAXPDL	0003
GOSTROUT	0003	AVARLEN	0003	PLOTMASK	0004	MANTLEN	0004	M.VMODE	0004
M.2	0004	REGTBLEN	0005	PWRCNLEN	0005	FACLEN	0005	AHDRLEN	0005
SWTBLEN	0006	GOODF8	0006	FACSIZE	0006	DISKIDLN	0006	SVARLEN	0007

SCMDMASK	0007	PXLSBYTE	0007	M.GOXY	0008	M.CTL	0008	INITBLEN	0008
BYTEBITS	0008	TITLEN	000A	GOUSR	000A	DFLTDIM	000B	CHARAC	000D
BYTVALUE	000D	OFFTITLE	000E	ENDCHR	000E	TOKNCNTR	000F	SROTMASK	000F
NUMDIM	000F	EOLPTR	000F	M.CURSOR	0010	M.4	0010	DIMFLG	0010
VALTYP	0011	GARFLG	0013	DATAFLG	0013	SUBFLG	0014	FRAMSIZE	0014
INPUTFLG	0015	TOGLFLG	0016	CPRMASK	0016	R12L	0018	R12H	0019
SHAPE	001A	ZPCDLEN	001C	R14L	001C	DELTITLE	001C	COLBITS	001C
R14H	001D	COLCOUNT	001D	R15L	001E	R15H	001F	WNDLFT	0020
MANTBITS	0020	M.CTL2	0020	WNDWDTH	0021	WNDTOP	0022	WNCBTM	0023
DECTBLEN	0024	CH	0024	CV	0025	GBASL	0026	GBASH	0027
BASL	0028	BASH	0029	BAS2L	002A	BAS2H	002B	LMNEM	002C
H2	002C	V2	002D	RMNEM	002D	MASK	002E	FORMAT	002E
CHKSUM	002E	SIGN	002F	LENGTH	002F	LASTIN	002F	HMASK	0030
COLOR	0030	MODE	0031	INVFLG	0032	PROMPT	0033	YSAV	0034
YSAV1	0035	CSWL	0036	CSWH	0037	KSWL	0038	KSWH	0039
PCL	003A	PCH	003B	A1L	003C	A1H	003D	A2L	003E
INVERSE	003F	A2H	003F	M.6	0040	A3L	0040	A3H	0041
A4L	0042	A4H	0043	OPRND	0044	MACSTAT	0044	A5L	0044
T2GUARD	0045	AREG	0045	XREG	0046	OTV.OR	0046	YREG	0047
PREG	0048	SPNT	0049	RNDL	004E	RNDH	004F	OTV.AND	0050
LINNUM	0050	ACL	0050	ACH	0051	TEMPPT	0052	LASTPT	0053
TEMPST	0055	INDEX	005E	DEST	0060	OFFSET	0061	MULMANT	0062
OTV.REL	0064	MULGUARD	0066	PRGTAB	0067	VARTAB	0069	ARYTAB	006B
STREND	006D	FRETOP	006F	FRESPC	0071	MEMSIZE	0073	CURLIN	0075
OLDLIN	0077	TEXTPTR	0079	OTV.ADD	0079	OTV.MUL	007B	DATLIN	007B
OTV.PWR	007D	DATPTR	007D	SRCPTR	007F	OTV.NEQ	007F	MSBCLR	007F
INVRSE80	007F	FLASH	007F	MSBSET	0080	M.PASCAL	0080	EXPBIAS	0080
VARNAM	0081	CTRLB	0082	VARPNT	0083	CTRLC	0083	FORPNT	0085
CTRLE	0085	TXPTRSAV	0087	LASTOP	0087	ASCIBELL	0087	LARROW	0088
CTRLH	0088	CPRTYPE	0089	TEMP3	008A	FUNCNAM	008A	DARROW	008A
CTRLJ	008A	UARROW	008B	CTRLK	008B	DSCPTR	008C	RETURN	008D
T3GUARD	008F	JMPADRS	0090	CTRLP	0090	RTNADR	0091	ARGGUARD	0092
TEMP1	0093	CTRLS	0093	LEN	0094	HIGHDS	0094	ARYPNT	0094
RARROW	0095	PROCESS	0095	CTRLU	0095	HIGHTR	0096	TEMP2	0098
CTRLX	0098	CTRLY	0099	COUNTER	0099	EXPCOUNT	009A	LOWTR	009B
ESCAPE	009B	DPFLAG	009B	EXPSIGN	009C	FACEXP	009D	DSCTMP	009D
FACMANT	009E	VARPTR	00A0	SPACE	00A0	FACSIGN	00A2	MINUSLOC	00A3
COEFNUM	00A3	EXTSIGN	00A4	PWRUPBYT	00A5	ARGEXP	00A5	ARGMANT	00A6
ARGSIGN	00AA	XORSIGN	00AB	STRING1	00AB	FACGUARD	00AC	STRING2	00AD
SAVY	00AD	COEFPTR	00AD	PRGEND	00AF	CHRGADR	00B1	CHRGET	00B1
CHRGOT	00B7	TXTPTR	00B8	TK.TAB	00C0	TK.TO	00C1	TK.FN	00C2
TK.SPC	00C3	TK.THEN	00C4	TK.AT	00C5	TK.NOT	00C6	TK.STEP	00C7
TK.PLUS	00C8	TK.MINUS	00C9	IRAND	00C9	FPRAND	00C9	SPCLFLAG	00CD
SIGNFLG	00CD	TK.GRTR	00CF	TK.EQUAL	00D0	SHPVAL	00D0	HRXDELTA	00D0
ROTQVAL	00D1	TK.SGN	00D2	ROTHVAL	00D2	HRYDELTA	00D2	ROTVVAL	00D3
HRFLAG	00D3	ROTHSUM	00D4	HRWORK	00D4	ROTVSUM	00D5	RUNFLAG	00D6
TK.SCRN	00D7	SHPOLD	00D7	ERRFLG	00D8	ERRLIN	00DA	ERRPOS	00DC
ERRNUM	00DE	LWRMASK	00DF	ERRSTK	00DF	HRXCOOR	00E0	HRYCOOR	00E2
HRCOLOR	00E4	HRHORZ	00E5	HRPAG	00E6	HRSCALE	00E7	HRSHPTBL	00E8
HRCOLCNT	00EA	MAXINPUT	00EF	FIRST	00F0	SPEEDBYT	00F1	TRACEFLG	00F2
FLASHBYT	00F3	TXTPTRSV	00F4	CURLINSV	00F6	REMSTK	00F8	HRROT	00F9
NEGTWO	00FE	ERROR.1	00FE	NEGONE	00FF	ERROR.2	00FF	STACK	0100
PAGESIZE	0100	INPUT	0200	XFERADR	03ED	AUTOBRK	03EF	AUTORSET	03F2
PWRSTATE	03F4	USRAHAND	03F5	USRYHAND	03F8	NMASKIRQ	03FB	MASKIRQ	03FE
TEXTPG1	0400	OLDCH	047B	XMODE	04FB	OURCH	057B	PG1TXLOC	05B0
OURCV	05FB	XCHAR	067B	XCOORD	06FB	XTEMP1	077B	OLDBASL	077B
MSLOT	07F8	XTEMP2	07FB	OLDBASH	07FB	PAGE08	0800	PAGE0C	0C00
PAGE10	1000	PAGE20	2000	PAGE40	4000	PAGEBD	BD00	PAGEBF	BF00
STR80OFF	C000	PAGEC0	C000	KEY	C000	IOSPACE	C000	STR80ON	C001
RAMRDOFF	C002	RAMRDON	C003	RAMWROFF	C004	RAMWRON	C005	CXROMOFF	C006
CXROMON	C007	AUXZPOFF	C008	AUXZPON	C009	C3ROMOFF	C00A	C3ROMON	C00B

VID80OFF	C00C	VID80ON	C00D	ALTCHOFF	C00E	ALTCHON	C00F	CLRKEY	C010
RDBANK2	C011	RDLCRAM	C012	RDRAMRD	C013	RDRAMWR	C014	RDCXROM	C015
RDAUXZP	C016	RDC3ROM	C017	RDSTR80	C018	RDVRTBLK	C019	RDTEXT	C01A
RDMIXED	C01B	RDPAGE2	C01C	RDHIRES	C01D	RDALTCH	C01E	RDVID80	C01F
TAPEOUT	C020	SPKRTOGL	C030	TEXTOFF	C050	TEXTON	C051	MIXEDOFF	C052
MIXEDON	C053	PAGE1ON	C054	PAGE2ON	C055	HIRESOFF	C056	HIRESON	C057
ANN1OFF	C058	ANN2OFF	C05A	ANN3ON	C05D	ANN4ON	C05F	TAPEIN	C060
PB1IN	C061	PB2IN	C062	GC1IN	C064	GCTOGL	C070	RAM2WP	C080
ROM2WE	C081	ROM2WP	C082	RAM2WE	C083	RAM1WP	C088	RAM1WE	C08B
PAGEC1	C100	DOCXCMD	C100	XCLREOP	C103	XHOME	C119	XSCROLL	C123
XSETWND	C152	XCLREOL2	C160	YSCROLL	C165	YCLREOL	C168	YCLREOL2	C16B
YCLREOP	C170	YSETWND	C173	YRESET	C176	XRESET	C176	YRDKEY	C179
YHOME	C17C	YIOPORT	C18A	XIOPORT	C195	ISO	C1A0	XRDKEY	C1A9
XBASCLC	C1B6	XRDESC	C1BD	XNEWVW	C1CF	YGETFMT	C1D5	XGETFMT	C1D5
YGOMINI	C1E1	XGOMINI	C1E1	YPICKFIX	C1E8	XPICKFIX	C1E8	XCLREOL	C1F1
XCLREOL1	C1F3	XVTAB	C203	CXEXIT	C208	XDOCMD	C211	YREGTBLX	C243
YREGTBLY	C24F	XKEYIN	C25B	XSETWNDX	C2A5	XRESETX	C2B5	CXRESET	C2DD
KBDTBL	C2EF	KBDOUT	C2F3	C3SPACE	C300	BASICINT	C300	BASICIN	C305
BASICOUT	C307	AUXMOVE	C311	XFER	C314	BASICENT	C317	JC8	C344
JBASINIT	C347	JPINIT	C34A	JPREAD	C350	JPWRITE	C356	JPSTAT	C35C
SETC8	C36D	DOMOVE	C376	DOXFER	C3C3	IRQDONE	C3F4	IRQRTN	C3FA
NEWIRQ	C47C	RAMSWTBL	C4C1	FORM	C4C8	GETNSP	C500	CXSTEP	C508
XJMPX	C575	XRTI	C584	XRTS	C588	PCINC2	C58C	PCINC3	C58E
XJSR	C598	XJMP	C5A3	XJMPAT	C5A4	BRANCH	C5B9	BRANCH2	C5C7
INITBL	C5CD	XGETFMT2	C5D5	PROCVAR	C600	PROCSPL	C64E	SW16	C670
SW16B	C679	SW16C	C67F	TOBR	C6A2	SETCMD2	C6B2	RSCMD2	C6C6
BSNSCMD2	C6D7	BRTBL	C6E2	OPTBL	C6E3	RTNCMD	C701	SETCMD	C709
RSCMD	C70B	BSNSCMD	C70D	LDCMD	C70F	STCMD	C718	LD@CMD	C721
ST@CMD	C72B	ST@CMD2	C72D	INRCMD	C733	LDD@CMD	C73A	STD@CMD	C743
POPD@CMD	C74C	POP@CMD	C753	STP@CMD	C763	ADDCMD	C76C	SUBCMD	C77A
CPRCMD	C77C	BSCMD	C790	BRCMD	C79C	BNCCMD	C79D	BCCMD	C7AE
BPCMD	C7B1	BMCMD	C7B6	BZCMD	C7BB	BNZCMD	C7C2	BM1CMD	C7C9
BNM1CMD	C7D2	SOUTCMD	C7DB	RSNSCMD	C7E0	SJMPCMD	C7E9	DCRCMD	C7F7
PXINIT	C800	PAGEC8	C800	BASCINIT	C803	C3HOOKS	C82A	C3IN	C832
PXREAD	C84D	CSETUP	C850	CXNEWVW	C870	CXNEWVW2	C874	CXVIDCK3	C87C
CXVIDCK4	C87E	CTRLON	C8BD	BIORET	C8C5	XINPUT	C8E6	ESCCHAR	C96B
ESCTABL	C97C	CXKEYIN	C98D	PXWRITE	C9AA	PXINIT2	C9B0	PPINIT	C9B4
PPREAD	C9D6	PPWRITE	C9F0	DOBASL	CA1F	PWRITER	CA29	BITBYT60	CA2E
OPTBLC	CA71	OPTBL	CA7D	TSTROMCD	CA89	TESTCARD	CA90	XBASCALC	CABA
CTRLCHR2	CAD2	CTRLCHR	CAD6	CTRLXFER	CB07	PCURON	CB0D	PCUROFF	CB18
XBS	CB40	XCR	CB51	XEM	CB5F	XFS	CB6B	XUS	CB79
XSO	CB84	XSI	CB8F	CTRLADRL	CB9E	CTRLADRH	CBB9	SCROLLDN	CBD4
XLF	CBD8	SCROLLUP	CBEB	XFF	CC74	XVT	CC77	XSUB	CC93
XGS	CC97	XGSEOLZ	CC9A	CLR40	CCA5	CLRHALF	CCAD	CLR80	CCBA
CLR2	CCD2	XDC1	CCE7	XDC1.2	CCEC	XDC2	CCF9	DO40	CD2B
SETTOP	CD2E	MOUSEOFF	CD3A	MOUSEON	CD41	XNAK	CD4A	SETKEYIN	CD58
SETCOUT2	CD61	FULL40	CD6A	FULL80	CD6E	QUIT	CD7D	SCRN84	CD8E
SCRN48	CDC1	SCRNRET	CDF5	XVTAB2	CDFB	XVTAB3	CE00	XRDKEY2	CE11
PASINV	CE1F	INVERT	CE26	STORCHAR	CE38	PICK	CE44	STORIT	CE70
ESCON	CEB1	ESCOFF	CEC4	ESCRTN	CECD	PSETUP	CED4	COPYROM	CEF4
REL	CF3A	TRYNEXT	CF6B	NXTLINE2	CF9C	CLRROM	CFFF	PAGEDO	D000
BASADDR	D000	BSFOR	D002	BSDATA	D006	BSPOP	D042	BSGOTO	D056
BSGOSUB	D060	BSREM	D064	BSPRINT	D074	FS1SGN	D080	FN1ADDR	D080
FS1SCRN	D08A	FS2LEFT	D0AC	FS3PI	D0B2	TAGADDR	D0B6	TAG.NEG	D0CB
TAG.EQU	D0CE	TAG.REL	D0D1	BASNAME	D0D4	MESGS	D25B	MESG01	D25B
MESG02	D26B	MESG03	D271	MESG04	D285	MESG05	D290	MESG06	D2A0
MESG07	D2A8	MESG08	D2B5	MESG09	D2C8	MESG10	D2D5	MESG11	D2E4
MESG12	D2F4	MESG13	D302	MESG14	D30F	MESG15	D31E	MESG16	D331
MESG17	D340	MESG18	D352	MESG20	D35A	GTFORPNT	D362	BLTU	D393
CKSTKSIZ	D3D6	CKSTRSIZ	D3E3	OM.ERR	D410	PRTERR	D412	PRLINUM	D431
RESTART	D43C	ASROMWRM	D43C	ASROMRST	D4F2	ASENTER	D4F2	INLIN	D52C

INLIN2	D52E	PARSINPT	D559	PARSE	D56C	PARSE2	D56D	PARSE3	D5A8
PARSE4	D5CD	PARSE5	D5F2	PARSE6	D5F9	PARSE7	D610	FNDLIN	D61A
FNDLIN2	D61E	RTN.D6.4	D648	BNEW	D649	SCRTCH	D64B	SETPTRS	D665
ASROMCLR	D665	BCLEAR	D66A	CLEARC	D66C	STKINIT	D683	RTN.D6.9	D696
STXTPTR	D697	RTN.D6.A	D6A4	BLIST	D6A5	LIST2	D6DA	LIST3	D6FC
LIST4	D700	LIST5	D721	LIST6	D727	GETCHR	D758	BFOR	D766
STEP	D7AF	NEWSTT	D7D2	ASROMNEW	D7D2	DOTRACE	D805	GOEND3	D826
DOSTAMT	D828	DOSTAMT2	D82A	DOSTAMT3	D842	BRESTORE	D849	SETDAPTR	D853
RTN.D8.5	D857	ISCNTLC	D858	DOCTRL.C	D863	ASROMERR	D865	BSTOP	D86E
BEND	D870	END2	D871	END3	D88A	BCONT	D896	RTN.D8.A	D8AF
DOHANDLR	D8B0	PULL3A	D8B3	RDBYTE	D8BB	BLOAD	D8C9	RD2BIT	D8FF
RDBIT	D902	BRUN	D912	RUN2	D91E	BGOSUB	D921	GOSUB2	D935
BGOTO	D93E	ASROMSET	D955	RTN.D9.6	D96A	BRETURN	D96B	BPOP	D96B
US.ERR	D97C	PULL3	D981	BDATA	D995	DATA2	D998	RTN.D9.A	D9A2
DATSCAN	D9A3	DATSCAN2	D9A6	SY.ERR2	D9C5	BIF	D9C9	BREM	D9DC
REM2	D9E1	BON	D9EC	RTN.DA.0	DA0B	LINGET	DA0C	LINGET2	DA12
BLET	DA46	LET2	DA63	PUTSTR	DA7B	COPYSTR	DAB7	RTN.DA.C	DACE
PRSTRING	DACF	BPRINT	DAD5	PRINT2	DAD7	UNARY2	DB32	LINEOUT	DB38
STROUT	DB3B	STRPRT	DB3E	PRTCR	DB50	OUTSPC	DB53	OUTPROMT	DB56
OUTCHR	DB58	INPUTERR	DB6F	READERR	DB79	ERRLINN	DB7D	SY.ERR3	DB81
INPERR	DB86	RESPERR	DB87	BGET	DBA0	BINPUT	DBB2	HEXTIN	DBDC
BREAD	DBE2	READ2	DBE9	INPTLIST	DBEB	INPTITEM	DBF1	INSTART	DC2B
INSTART2	DC69	INSTART3	DC72	INPTFLG	DC99	FINDATA	DCA0	INPTDONE	DCC7
MESG21	DCDF	MESG22	DCEF	BNEXT	DCF9	NEXT2	DCFB	RTN.DD.4	DD4C
FPCOMPT3	DD4D	FRMNUM	DD64	CHKNUM	DD67	CHKSTR	DD69	CHKVAL	DD6A
TM.ERR	DD73	NF.ERR	DD76	FRMEVAL	DD7B	FRMEVAL2	DD86	FRMEVAL3	DD95
FRMEVAL4	DD98	SAVOP	DDD7	FRMRECUR	DDFD	SY.ERR4	DE0D	FRMSTAK	DE10
FRMSTAK2	DE15	FRMSTAK3	DE23	NOTMATH	DE32	NOTMATH2	DE35	NOTMATH3	DE37
NOTMATH4	DE40	FRMELMNT	DE60	STRTXT	DE81	NOTFUNC	DE90	OEQUAL	DE98
FNFUNC	DEA7	SGNFUNC	DEAE	PARENCHK	DEB2	CHKCLSP	DEB8	CHKOPNP	DEBB
CHKCOM	DEBE	SYNTAXCHK	DEC0	SY.ERR	DEC9	MINUFUNC	DECE	EQUFUNC	DED0
GETIVAL	DED5	FSCREEN	DEF9	UNARY	DF09	UNARY3	DF3F	OR	DF4F
OAND	DF55	FALSE	DF5D	TRUE	DF60	OLT	DF65	STRCMP	DF7D
NUMCMP	DFB0	FPDL	DFCD	BDIM	DFD9	PTRGET	DFE3	PTRGET2	DFE8
PTRGET3	DFEA	SY.ERR5	DFE4	BASIC	E000	BASIC2	E003	PTRGET4	E006
PTRGET5	E076	PTRGET6	E089	PTRGET7	E0CB	CHKASCI	E0DA	PNTARVAL	E0E3
STRSETUP	E0FF	IVALZERO	E105	FP8000	E107	MAKINT	E10C	AYPOSINT	E112
AYINT	E116	IQ.ERR2	E123	ARRAY	E128	BS.ERR	E198	IQ.ERR	E19B
RA.ERR	E19E	OD.ERR	E1A1	FINDELE	E24B	BS.ERR2	E269	OM.ERR2	E26C
RTN.E2.A	E2AC	MULSUBS	E2AD	MULSUBS2	E2B6	RTN.E2.D	E2DD	FFRE	E2DE
GIVAYFP	E2F2	FPOS	E2FF	SNGFLT	E301	CHKIFDIR	E306	UF.ERR	E30E
BDEF	E313	GETFNC	E341	CALLFNC	E354	FNCDATA	E3AF	FSTR	E3C5
STRINI	E3D0	STRSPA	E3D8	STRLIT	E3E2	STRLIT1	E3E3	STRLIT2	E3E9
PUTNEW	E426	FC.ERR	E449	OM.ERR3	E44C	GETSSPC	E454	GARBAG	E484
CHKVARS	E48D	CHKVARS2	E4BA	CHKARRYS	E4C2	GARBEXIT	E501	MOVVARS	E50C
MOVVARS2	E518	NXTVAR	E567	COPYVAR	E573	DECPTR	E58C	CAT2STR	E597
SL.ERR	E5CF	MOVINS	E5D4	MOVSTR	E5E2	MOVSTR2	E5E6	FRESTR	E5FD
FREFAC	E600	FRETMP	E604	FRETMS	E635	FCHR	E646	FLEFT	E65A
LEFT2	E660	LEFT3	E667	LEFT4	E668	FRIGHT	E686	FMID	E691
STRSET2	E6BC	FLEN	E6D6	GETSTRLN	E6DC	FASC	E6E5	IQ.ERR3	E6F2
GETBYTC	E6F5	GETBYT	E6F8	CONVINT	E6FB	FVAL	E707	STRCOPY	E73D
GETASNUM	E746	COMBYTE	E74C	GETADDR	E752	FPEEK	E764	BPOKE	E77B
BWAIT	E784	RTN.E7.9	E79F	FPPI	E7A1	FPIGUARD	E7A6	FSUB	E7A7
SUB2	E7AA	OMINUS	E7AA	FADD	E7BE	OPLUS	E7C1	ADD2	E7C1
ADD3	E7CA	COMPFAC1	E81D	NORMFAC1	E822	ZEROFAC	E842	ZEROFAC2	E844
ADD4	E849	NORMFAC2	E868	NORMFAC3	E874	NORMFAC4	E883	RTN.E8.9	E891
COMPFAC2	E892	COMPMANT	E898	INCMANT	E8BA	RTN.E8.D	E8C8	OF.ERR	E8C9
SHFTBYT1	E8CE	SHFTBYT2	E8E5	SHFTBITS	E8FC	PDL2	E908	FPLOGE	E913
FPSQR0.5	E918	FPSQR2.0	E91D	FPN0.5	E922	FPLN2	E927	POLY.LOG	E92C
FLN	E941	FMULT	E97F	OMULT	E982	MULT2	E982	TSTMULT	E9AA
BYTMULT	E9AF	LOADARG	E9E3	PROCEXP	EA10	PROCEXP2	EA12	PROCEXP3	EA28

ZEROFERR	EA2D	OF.ERR2	EA31	DZ.ERR	EA34	MULFAC10	EA39	FP10.0	EA4F
DIVFAC10	EA54	FDIV	EA66	ODIVIDE	EA69	DIV2	EA69	FP.25	EAD7
FP1.0E9	EADC	COPYM2F	EAE6	LOADFAC	EAF9	COPYF2T2	EB1E	COPYF2T1	EB21
COPYF2FR	EB27	COPYFAC	EB2B	COPYFAC2	EB2E	COPYA2F	EB53	COPYA2F2	EB55
COPYF2A	EB63	RNDUP	EB70	SIGNCHK	EB82	SIGNCHK2	EB86	SIGNCHK3	EB88
RTN.EB.8	EB8F	FSGN	EB90	FLOAT	EB93	FLOAT2	EB9B	FLOAT3	EBA0
FABS	EBAF	FPCOMP0	EBB2	FPCOMP	EBB5	FPCOMP2	EBB7	FP2INT	EBF2
FINT	EC23	CLRMANT	EC40	RTN.EC.4	EC49	GETINT	EC4A	GETINT2	EC61
GETINT3	EC87	GETINT4	ECA0	GETINT5	ECC0	GETINT6	ECD4	ADD2FAC	ECF6
PRTMSG19	ED0A	LINEPRT	ED18	MESG19	ED25	FP9.9E7	ED2A	FP9.9E8	ED2F
FPOUT	ED34	FPOUT2	EE49	SAV2STK	EE54	FPDECTBL	EE5C	FSQR	EE8D
OPOWER	EE97	FEXP2	EECD	OGT	EED0	NEGFAC	EED0	RTN.EE.D	EEDA
FPINVLN2	EEDB	POLY.EXP	EEE0	FP1.0	EF04	FEXP	EF09	FLOG	EF3E
FPI	EF48	POLYSIN	EF57	POLYPROC	EF5B	POLYNOM	EF71	RTN.EF.A	EFA5
RANDVAL1	EFA6	RANDVAL2	EFAA	FRND	EFAE	FCOS	EFEA	FSIN	EFF1
PAGEF0	F000	SIN2	F025	FTAN	F03A	FPIDIV2	F05C	FP0.5	F061
POLY.SIN	F066	FPIMUL2	F099	FATAN	F09E	POLY.ATN	F0CC	PGZCODE	F109
COLDSTRT	F125	FRMSTAK4	F1B1	COPYF2T3	F1BA	INCCOEF	F1C5	CLEARMUL	F1CC
BCALL	F1D5	BIN	F1DE	BPR	F1E5	PLOTFNS	F1EC	IQ.ERR4	F206
LINCOOR	F209	BPLOT	F225	BHLIN	F232	BVLIN	F241	BCOLOR	F24F
BVTAB	F256	BSPEED	F262	BTRACE	F26D	BNOTRACE	F26F	BNORMAL	F273
BINVERSE	F277	INVERSE2	F279	INVERSE3	F27B	BFLASH	F280	BHIMEM	F286
OM.ERR4	F2A3	BLOMEM	F2A6	BONERR	F2CB	HANDLERR	F2E9	BRESUME	F318
SY.ERR6	F32E	BDEL	F331	BGR	F390	BTEXT	F399	CXREAD	F39C
BHGR2	F3D8	BHGR	F3E2	CLRHIRE	F3EC	SETHIRES	F3EE	HPOSN	F411
HRPLOT	F457	HRMOVLF	F465	HRMOVLF2	F46E	HRMOVLF3	F478	COLSHIFT	F47E
HRMOVRT	F48A	DRAWHDR	F49C	XDRAWIT	F4A6	DRAWIT	F4B8	HRMOVUP	F4D1
HRMOVDN	F501	RTN.F5.2	F52C	BITBYT03	F52D	BITBYT04	F52E	BITBYT1C	F52F
BITABLE	F530	HLIN	F53A	ROTATBL	F5B3	IQ.ERR5	F5C4	DRAWCMD	F5C7
SQR2	F666	COPYA2F3	F68E	COPYF2A2	F693	COPYT32A	F6A8	GETFNS	F6B9
IQ.ERR6	F6E6	BHCOLOR	F6E9	RTN.F6.F	F6F5	HRCOLTBL	F6F6	BHPLOT	F6FE
HPLLOT2	F708	BROT	F721	BSCALE	F727	RND2	F72D	BDRAW	F769
BXDRAW	F76F	RND3	F775	TITLE	F791	PARSIEX1	F79B	PARSIEX2	F7A0
PARSIEX3	F7AE	LISTEX1	F7BE	PRTCRESX1	F7C6	LISTEX2	F7C6	PRTCRESX2	F7D5
PRTCRESX3	F7DC	RTN.F7.E	F7E6	BHTAB	F7E7	PLOT	F800	F8SPACE	F800
RTMASK	F80C	PLOT1	F80E	HLINE	F819	HLINE1	F81C	VLINE	F828
CLRSCR	F832	CLRTOP	F836	GBASCALC	F847	NXTCOL	F85F	SETCOL	F864
SCRN	F871	SCRN2	F879	INSDS1	F882	INSDS2	F88E	GETFMT	F8A5
TESTROM	F8B6	INSTDSP	F8D0	PRNTOP	F8D4	PRNTBL	F8DB	PRNTYX	F940
PRNTAX	F941	PRNTX	F944	PRBLNK	F948	PRBL2	F94A	PCADJ	F953
PCADJ2	F954	PCADJ3	F956	RTS2	F961	FMT1	F962	MNEML	F9A6
MNEMR	F9EB	CHAR1	FA30	CHAR2	FA36	CXOFF	FA3C	CXRTN	FA3F
OLDIRQ	FA40	NEWBREAK	FA47	BREAK	FA4C	OLDBRK	FA59	RESET	FA62
SWEET16	FA72	SW16RTN	FA78	RESET2	FA7E	NEWMON	FA81	NEWMON2	FA9B
PWRUP	FAA6	REGDSP	FAD7	REGDSP2	FADA	PWRCON	FAFD	DISKID	FB02
RSETINIT	FB08	XLATBL	FB14	REGTBL	FB19	PREAD	FB1E	INIT	FB2F
INIT2	FB33	SETEXT	FB39	SETGR	FB40	SETWND	FB4B	TABV	FB5B
APPLE2	FB60	STITLE	FB65	SETPWRUP	FB6F	VIDWAIT	FB78	KBDWAIT	FB88
ESCOLD	FB97	ESCNEW	FBA5	SIGROM	FBB3	GOTOROM	FBB4	SIGBYTE	FBC0
BASCALC	FBC1	FMT2	FBC7	BELL2	FBD9	RINGBELL	FBDD	STORADV	FBF0
ADVANCE	FBF4	RTN.FB.F	FBFC	VIDOUT	FBFD	BS	FC10	UP	FC1A
VTAB	FC22	VTAB2	FC24	ESCOLD2	FC2C	CLREOP	FC42	NEWVW	FC46
HOME	FC58	GOTOROM2	FC5A	STEPRTN2	FC5F	CR	FC62	LF	FC66
SCROLL	FC70	GOTOIRQ	FC74	IRQDONE2	FC7A	CHKINV	FC95	CLREOL	FC9C
CLREOL2	FC9E	WAIT	FCA8	NEXTA4	FCB4	NEXTA1	FCBA	HEADR	FCC9
STEPRTN	FCCA	XERR	FCD2	FINDOP	FCE3	NXTLINE	FCF0	UPRMON	FCFD
UPRCASE	FD01	RDKEY	FD0C	RDKEY2	FD13	RDKEY3	FD18	KEYIN	FD1B
GOTOROM4	FD1D	RDESC	FD21	NEWRDKEY	FD28	ESC	FD2F	RDCHAR	FD35
PICKFIX	FD3D	NOTCR	FD47	CANCEL	FD62	GETLINE2	FD67	GETLINE	FD6A
BCKSPC	FD71	NXTCHAR	FD75	ADDINP	FD84	CROUT	FD8E	PRA1	FD92
PRXY	FD96	XAM8	FDA3	MOD8CHK	FDAD	XAM	FDB3	DATAOUT	FDB6

XAMPM	FDC6	ADD	FDD1	PRBYTE	FDDA	PRHEX	FDE3	PRHEX2	FDE5
COUT	FDED	COUT2	FDF0	COUTZ	FDF6	COUTZ2	FDF7	PAGEFE	FE00
BL1	FE00	BLANK	FE04	STOR	FE0B	STOR2	FE0F	NEXTA3	FE11
SETMODE	FE18	SETMDZ	FE1D	LT	FE20	MOVE	FE2C	VERIFY	FE36
LIST	FE5E	A1PC	FE75	A1PCLP	FE78	SETINV	FE80	SETNORM	FE84
SETKBD	FE89	INPORT	FE8B	INPORT2	FE8D	SETVID	FE93	OUTPORT	FE95
OUTPORT2	FE97	IOPORT	FE9B	ZAPMEM	FEAE	XBASIC	FEB0	BASCONT	FEB3
GO	FEB6	REGZ	FEBF	TRACE	FEC2	STEPZ	FEC4	USR	FECA
WRITE	FECD	CHRTBLX3	FECE	SEARCH	FED2	MINIASM	FEF1	CRMON	FEF6
CRMON2	FEF9	READ	FEFD	ZAPMEM2	FF05	RTN.FF.0	FF0E	CHRTBLX2	FF0F
LOOKASC	FF18	PRNTERR	FF2D	BELL	FF3A	RESTORE	FF3F	SAVE	FF4A
SAVE2	FF4C	IORTS	FF58	OLDRST	FF59	CHRTBLX1	FF5F	MON	FF65
MON2	FF69	NEXTITM	FF73	CHRSRCH	FF7A	DIG	FF8A	NEXTBIT	FF90
GETNUM	FFA7	NEXTCHR	FFAD	TOSUBR	FFBE	ZMODE	FFC7	CHRTBL	FFCC
SUBTBL	FFE3								