7 February, 1983

FROB-26 SYSTEM

LISTINGS

Updated for Rel. 1.3

```
SOURCE FILE: EXPLORER
0000:           1 * EXPLORER -- A FROBCO PROGRAM TO EXPLORE THE OPERATION OF THE VCS GAME PLAYER
0000:           2 * COPYRIGHT 1982 BY FROBCO -- ALL RIGHTS RESERVED
0000:           3 *
0000:           4 * VERSION 1.3 -- LAST MODIFIED  2/5/83
0000:           5 *
0000:           6 *
0000:           7 * SYSTEM DEFINITIONS
0000:           8 *
0000:           9 VSYNCH  EQU  0        ; WRITE A 2 HERE TO CAUSE VERTICAL SYNCH PULSE (ONCE PER
0001:          10 VRESET  EQU  1        ; RESET VIDEO ONCE PER SCREEN
0002:          11 LWAIT   EQU  2        ; WRITE ANYTHING HERE TO WAIT TO END OF CURRENT LINE
0003:          12 VID03   EQU  3        ; NOT USED WITH NORMAL NTSC HORIZONTAL TIMING
0004:          13 P1VMOD  EQU  4        ; PLAYER1 VIDEO MODE REGISTER
0005:          14 P2VMOD  EQU  5        ; PLAYER2 VIDEO MODE REGISTER
0006:          15 P1COLOR EQU  6        ; PLAYER 1 COLOR REGISTER
0007:          16 P2COLOR EQU  7        ; PLAYER 2 COLOR REGISTER
0008:          17 OCOLOR  EQU  8        ; OBJECT COLOR REGISTER
0009:          18 BCOLOR  EQU  9        ; BACKGROUND COLOR REGISTER
000A:          19 FVMODE  EQU  $0A      ; FIELD VIDEO MODE REGISTER
000B:          20 P1FLIP  EQU  $0B      ; PLAYER1 FLIP REGISTER & READ FIRE BUTTON
000C:          21 P2FLIP  EQU  $0C      ; PLAYER2 FLIP REGISTER & READ FIRE BUTTON
000D:          22 FLDAIM  EQU  $0D      ; OBJECT FIELD A IMAGE REGISTER
000E:          23 FLDBIM  EQU  $0E      ; OBJECT FIELD B IMAGE REGISTER
000F:          24 FLDCIM  EQU  $0F      ; OBJECT FIELD C IMAGE REGISTER
0010:          25 P1HRES  EQU  $10      ; WRITE HERE TO RESET PLAYER1 HORZ POS TO 0
0011:          26 P2HRES  EQU  $11      ; WRITE HERE TO RESET PLAYER2 HORZ POS TO 0
0012:          27 S1HRES  EQU  $12      ; WRITE HERE TO RESET PLAYER1 SHOT HORZ POS TO 0
0013:          28 S2HRES  EQU  $13      ; WRITE HERE TO RESET PALYER2 SHOT HORZ POS TO 0
0014:          29 S3HRES  EQU  $14      ; WRITE HERE TO RESET SHOT3 HORZ POS TO 0
0015:          30 SND1MD  EQU  $15      ; SOUND MODE REGISTER FOR PLAYER 1
0016:          31 SND2MD  EQU  $16      ; SOUND MODE REGISTER FOR PLAYER 2
0017:          32 SND1TN  EQU  $17      ; SOUND TONE REGISTER FOR PLAYER 1
0018:          33 SND2TN  EQU  $18      ; SOUND TONE REGISTER FOR PLAYER 2
0019:          34 SND1AM  EQU  $19      ; SOUND AMPLITURE REGISTER FOR PLAYER 1
001A:          35 SND2AM  EQU  $1A      ; SOUND AMPLITUDE REGISTER FOR PLAYER 2
001B:          36 P1IMAG  EQU  $1B      ; VIDEO IMAGE FOR PLAYER 1
001C:          37 P2IMAG  EQU  $1C      ; VIDEO IMAGE FOR PLAYER 2
001D:          38 P1SHOT  EQU  $1D      ; WRITE 2 HERE TO ENABLE SHOT1 IMAGE
001E:          39 P2SHOT  EQU  $1E      ; SAME AS ABOVE BUT FOR SHOT 2
001F:          40 S3SHOT  EQU  $1F      ; SAME AS ABOVE BUT FOR SHOT 3
0020:          41 P1HORZ  EQU  $20      ; HORIZONTAL FINE INCREMENT FOR PLAYER1
0021:          42 P2HORZ  EQU  $21      ; HORIZONTAL FINE INCREMENT FOR PLAYER2
0022:          43 S1HORZ  EQU  $22      ; HORIZONTAL FINE INCREMENT FOR SHOT1
0023:          44 S2HORZ  EQU  $23      ; HORIZONTAL FINE INCREMENT FOR SHOT2
0024:          45 S3HORZ  EQU  $24      ; HORIZONTAL FINE INCREMENT FOR SHOT3
0025:          46 P1DELAY EQU  $25      ; CONTROLS DELAY FROM WHEN P1 IMAGE IS WRITTEN TO WHEN SHOWN
0026:          47 P2DELAY EQU  $26      ; CONTROLS DELAY FROM WHEN P2 IMAGE IS WRITTEN TO WHEN SHOWN
0027:          48 VID27   EQU  $27
0028:          49 S1CONT  EQU  $28      ; SHOT 1 CONTROL REGISTER
0029:          50 S2CONT  EQU  $29      ; SHOT 2 CONTROL REGISTER
002A:          51 HZSCRL  EQU  $2A      ; WRITE HERE TO ADVANCE PLAYER AND SHOT HORZ SCROLE
002B:          52 NOINC   EQU  $2B      ; WRITING HERE STOPS HOIZONTAL INCREMENT OF ALL SPRITES
002C:          53 COLRES  EQU  $2C      ; WRITE HERE TO RESET COLLISION REGISTERS
002D:          54 VID2D   EQU  $2D
002E:          55 VID2E   EQU  $2E
002F:          56 VID2F   EQU  $2F
0030:          57 PS1COL  EQU  $30      ; BIT 7 SET IF S1+P2, BIT 0 SET IF S1+P1
0031:          58 PS2COL  EQU  $31      ; BIT 7 SET IF S2+P1, BIT 0 SET IF S2+P2
0032:          59 P1OCOL  EQU  $32      ; BIT 7 SET IF OBJ+P1, BIT 0 SET IF S3+P1
0033:          60 P2OCOL  EQU  $33      ; BIT 7 SET IF OBJ+P2, BIT 0 SET IF S3+P2
0034:          61 S1OCOL  EQU  $34      ; BIT 7 SET IF OBJ+S1, BIT 0 SET IF S3+S1
0035:          62 S2OCOL  EQU  $35      ; BIT 7 SET IF OBJ+S2, BIT 0 SET IF S3+S2
```

```
0036:          63 S3OCOL  EQU  $36       ; BIT 7 SET IF OBJ+S3
0037:          64 PPCOL   EQU  $37       ; BIT 7 COMES BACK SET IF PLAYER 1 COLLIDES WITH PLAYER 2
0038:          65 PDL5L   EQU  $38       ; BIT 7 IS PIN 5 ANALOG FROM LEFT PADDLES
0039:          66 PDL9L   EQU  $39       ; BIT 7 IS PIN 9 ANALOG FROM LEFT PADDLES
003A:          67 PDL5R   EQU  $3A       ; BIT 7 IS PIN 5 ANALOG FROM RIGHT PADDLES
003B:          68 PDL9R   EQU  $3B       ; BIT 7 IS PIN 9 ANALOG FROM RIGHT PADDLES
003C:          69 LFTFR   EQU  $3C       ; BIT 7 CLEAR IF LEFT FIRE BUTTON DOWN
003D:          70 RGHFR   EQU  $3D       ; BIT 7 CLEAR IF RIGHT FIRE BUTTON DOWN
003E:          71 VID3E   EQU  $3E
003F:          72 VID3F   EQU  $3F
0000:          73 *
0000:          74 *
0000:          75 * NOW THE 6532 REGISTERS
0000:          76 *
0280:          77 CTROLS  EQU  $280      ; READ THIS FOR THE JOY STICK BITS
0281:          78 DDRA    EQU  $281      ; DATA DIRECTION REGISTER FOR THE ABOVE PORT
0282:          79 SWTCHS  EQU  $282      ; READ THIS FOR THE CONSOL SWITCH BITS
0283:          80 DDRB    EQU  $283      ; DATA DIRECTION REGISTER FOR THE ABOVE PORT
0284:          81 RTIME   EQU  $284      ; READ THE TIMER
0000:          82 *
0294:          83 TIM1    EQU  $294      ; WRITE HERE TO SET TIMER FOR 1/1
0295:          84 TIM8    EQU  $295      ; WRITE HERE TO SET TIMER FOR 1/8
0296:          85 TIM64   EQU  $296      ; WRITE HERE TO SET TIMER FOR 1/64
0297:          86 TIM1024 EQU  $297      ; WRITE HERE TO SET TIMER FOR 1/1024
0000:          87 *
0000:          88 *
0000:          89 * RAM ASSIGNMENTS
0000:          90 *
0080:          91 SCREENS EQU  $80       ; SCREEN COUNTER GOING FROM 0 TO 59
0081:          92 SECONDS EQU  SCREENS+1 ; SECOND COUNTER INCREMENTED EVERY 60 SCREENS
0082:          93 TARL    EQU  SECONDS+1 ; LOW ORDER INDIRECT USED BY FMON
0083:          94 TARH    EQU  TARL+1    ; HIGH ORDER INDERECT USED BY FMON
0084:          95 COMCOD  EQU  TARH+1    ; COMMAND CODE HOLDING REGISTER USED BY FMON
0085:          96 VLNCT   EQU  COMCOD+1  ; VERTICAL LINE COUNTER
0086:          97 SPREG   EQU  VLNCT+1   ; TEMP STORE USED FOR THE STACK POINTER
0087:          98 DIGCON  EQU  SPREG+1   ; DIGIT CONTROL REGISTER
0088:          99 TOPCON  EQU  DIGCON+1  ; CONTROL REG FOR THE TOP OF SCREEN
0089:         100 P1HS    EQU  TOPCON+1  ; HORIZONTAL POSITON OF PLAYER1
008A:         101 P2HS    EQU  P1HS+1    ; HORIZONTAL POSITION OF PLAYER2
008B:         102 S1HS    EQU  P2HS+1    ; HORIZONTAL POSITION OF PLAYER 1 SHOT
008C:         103 S2HS    EQU  S1HS+1    ; HORIZONTAL POSITION OF PLAYER 2 SHOT
008D:         104 TREG1   EQU  S2HS+1    ; THE TREG GROUP IS USED TO HOLD IMAGES TO BE DISPLAYED
0093:         105 TREG2   EQU  TREG1+6   ; AT THE TOP OF THE SCREEN.
0099:         106 TREG3   EQU  TREG2+6
009F:         107 TREG4   EQU  TREG3+6
00A5:         108 SMODE   EQU  TREG4+6   ; SCREEN MODE REGISTER
00A6:         109 MODE1   EQU  SMODE+1   ; VID MODE 1 REGISTER
00A7:         110 MODE2   EQU  MODE1+1   ; VID MODE 2 REGISTER
00A8:         111 EBITS   EQU  MODE2+1   ; TEMP STORAGE FOR THE ENABLE BITS
00A9:         112 TMCNT   EQU  EBITS+1   ; IN BETWEEN SCREEN TIME COUNT
00AA:         113 SPTS    EQU  TMCNT+1   ; PARTS OF THE SCREEN FUNCTIONS TO DO
00AB:         114 P1CR    EQU  SPTS+1    ; PLAYER1 COLOR (TOP 4 BITS CHROMA LOWER 4 BITS LUMA)
00AC:         115 P2CR    EQU  P1CR+1    ; PLAYER2 COLOR
00AD:         116 BKCR    EQU  P2CR+1    ; BACKGROUND COLOR
00AE:         117 OBJCR   EQU  BKCR+1    ; OBJECT COLOR
00AF:         118 P1VERT  EQU  OBJCR+1   ; PLAYER1 VERTICAL POSITION REGISTER
00B0:         119 P2VERT  EQU  P1VERT+1  ; PLAYER2 VERTICAL POSITION REGISTER
00B1:         120 S1VERT  EQU  P2VERT+1  ; SHOT 1 VERTICAL REGISTER
00B2:         121 S2VERT  EQU  S1VERT+1  ; SHOT 2 VERTICAL POSITION REGISTER
00B3:         122 P1IMG   EQU  S2VERT+1  ; PLAYER1 IMAGE ARRAY
00BB:         123 P2IMG   EQU  P1IMG+8   ; PLAYER2 IMAGE ARRAY
00C3:         124 OBJAVT  EQU  P2IMG+8   ; OBJECT A VERTICAL POSITION
00C4:         125 OBJAIM  EQU  OBJAVT+1  ; OBJECT A IMAGE ARRAY
00CC:         126 OBJBVT  EQU  OBJAIM+8  ; OBJECT B VERTICAL POSITION
00CD:         127 OBJBIM  EQU  OBJBVT+1  ; OBJECT B IMAGE ARRAY
00D5:         128 OBJCVT  EQU  OBJBIM+8  ; OBJECT C VERTICAL POSITION
```

```
00D6:          129 OBJCIM  EQU  OBJCVT+1  ; OBJECT C IMAGE ARRAY
00DE:          130 OBJXIM  EQU  OBJCIM+8  ; ARRAY USED IN 48 BIT DISPLAY
00E6:          131 TP1     EQU  OBJXIM+8  ; TIMING PARAMETER 1
00E7:          132 TP25    EQU  TP1+1     ; TIMING PARAMETER 25
00E8:          133 TP26    EQU  TP25+1    ; TIMING PARAMETER26
00E9:          134 TEMP1   EQU  TP26+1    ; TEMPORIES
00EA:          135 TEMP2   EQU  TEMP1+1
0000:          136 *
0000:          137 *
0000:          138 *
0000:          139 * START OF PROGRAM - LOAD AT $F000
----- NEXT OBJECT FILE NAME IS EXPLORER.OBJ0
F000:          140         ORG  $F000
F000:          141 *
F000:A2 FF     142 START   LDX  #$FF      ; HERE ON RESET SO SET UP THE STACK POINTER
F002:9A        143         TXS
F003:D8        144         CLD              ; CLEAR DECIMAL MODE
F004:A2 00     145         LDX  #0        ; DO A MASSIVE MEMORY CLEAR
F006:A9 00     146         LDA  #0
F008:95 00     147 CLOOP   STA  0,X
F00A:CA        148         DEX
F00B:D0 FB     149         BNE  CLOOP
F00D:          150 *
F00D:          151 *
F00D:          152 *     MAIN RAM INITIALIZATION
F00D:          153 *
F00D:A9 03     154         LDA  #3        ; SET UP FOR  PLAYER1 AND PLAYER2 DISPLAY AT FIRST
F00F:85 AA     155         STA  SPTS
F011:A9 02     156         LDA  #2        ; SET UP TOPCON TO DO THE DISPLAY OF TOP SECTION
F013:85 88     157         STA  TOPCON
F015:A9 00     158         LDA  #0        ; INIT THE HORZ POSITION SHADOW REGISTER FOR P1, P2, AND SHOTS
F017:85 89     159         STA  P1HS
F019:85 8A     160         STA  P2HS
F01B:85 8B     161         STA  S1HS
F01D:85 8C     162         STA  S2HS
F01F:          163 *
F01F:A9 14     164         LDA  #20       ; SET UP PLAYER1  VERTICAL
F021:85 AF     165         STA  P1VERT
F023:A9 1E     166         LDA  #30       ; SET UP PLAYER2 VERTICAL
F025:85 B0     167         STA  P2VERT
F027:A9 0E     168         LDA  #14       ; SET UP PLAYER1 SHOT VERTICAL
F029:85 B1     169         STA  S1VERT
F02B:A9 12     170         LDA  #18       ; SET UP PLAYER2 SHOT VERTICAL
F02D:85 B2     171         STA  S2VERT
F02F:A9 2D     172         LDA  #45       ; SET UP OBJECT A VERITCAL
F031:85 C3     173         STA  OBJAVT
F033:A9 37     174         LDA  #55       ; SET UP OBJECT B VERTICAL
F035:85 CC     175         STA  OBJBVT
F037:A9 41     176         LDA  #65       ; SET UP OBJECT C VERTICAL
F039:85 D5     177         STA  OBJCVT
F03B:          178 *
F03B:A9 FE     179         LDA  #$FE      ; MAKE PLAYER IMAGES
F03D:85 B3     180         STA  P1IMG
F03F:85 BB     181         STA  P2IMG
F041:A9 7F     182         LDA  #$7F
F043:85 B4     183         STA  P1IMG+1
F045:85 BC     184         STA  P2IMG+1
F047:A9 63     185         LDA  #$63
F049:85 B5     186         STA  P1IMG+2
F04B:85 BD     187         STA  P2IMG+2
F04D:A9 62     188         LDA  #$62
F04F:85 B6     189         STA  P1IMG+3
F051:85 BE     190         STA  P2IMG+3
F053:A9 7C     191         LDA  #$7C
F055:85 B7     192         STA  P1IMG+4
F057:85 BF     193         STA  P2IMG+4
```

C

```
F059:A9 78     194         LDA  #$78
F05B:85 B8     195         STA  P1IMG+5
F05D:85 C0     196         STA  P2IMG+5
F05F:A9 60     197         LDA  #$60
F061:85 B9     198         STA  P1IMG+6
F063:85 C1     199         STA  P2IMG+6
F065:A9 F2     200         LDA  #$F2
F067:85 BA     201         STA  P1IMG+7
F069:A9 F6     202         LDA  #$F6
F06B:85 C2     203         STA  P2IMG+7
F06D:          204 *
F06D:          205 *
F06D:4C A0 F0  206         JMP  TINIT     ; JUMP AROUND THIS DATA
F070:          207 *
F070:FE 62 F8  208 LETF    DFB  $FE,$62,$F8,$60,$60,$F0
F073:60 60 F0
F076:3F 66 66  209 LETR    DFB  $3F,$66,$66,$3E,$26,$4F
F079:3E 26 4F
F07C:7E FF C3  210 LETO    DFB  $7E,$FF,$C3,$C3,$FF,$7E
F07F:C3 FF 7E
F082:3F 26 7E  211 LETB    DFB  $3F,$26,$7E,$7E,$26,$3F
F085:7E 26 3F
F088:          212 *
F088:18 24 42  213 LETA    DFB  $18,$24,$42,$42,$7E,$42,$42,$CE
F08B:42 7E 42
F08E:42 CE
F090:FE 61 61  214 LETB8   DFB  $FE,$61,$61,$7E,$7E,$61,$61,$FE
F093:7E 7E 61
F096:61 FE
F098:7E FF FE  215 LETC    DFB  $7E,$FF,$FE,$F0,$F0,$FE,$FF,$7E
F09B:F0 F0 FE
F09E:FF 7E
F0A0:          216 *
F0A0:          217 *   LOOP TO LOAD IMAGES AT TOP OF SCREEN
F0A0:A2 00     218 TINIT   LDX  #0
F0A2:BD 70 F0  219 TLP     LDA  LETF,X
F0A5:95 8D     220         STA  TREG1,X
F0A7:E8        221         INX
F0A8:E0 18     222         CPX  #$18
F0AA:D0 F6     223         BNE  TLP
F0AC:          224 *
F0AC:A2 00     225         LDX  #0        ; INITIALIZE OBJECT A IMAGE
F0AE:BD 88 F0  226 AIMLP   LDA  LETA,X
F0B1:95 C4     227         STA  OBJAIM,X
F0B3:E8        228         INX
F0B4:E0 08     229         CPX  #8
F0B6:D0 F6     230         BNE  AIMLP
F0B8:          231 *
F0B8:A2 00     232         LDX  #0        ; INITIALIZE OBJECT B IMAGE
F0BA:BD 90 F0  233 BIMLP   LDA  LETB8,X
F0BD:95 CD     234         STA  OBJBIM,X
F0BF:E8        235         INX
F0C0:E0 08     236         CPX  #8
F0C2:D0 F6     237         BNE  BIMLP
F0C4:          238 *
F0C4:A2 00     239         LDX  #0        ; INITIALIXE OBJECT C IMAGE
F0C6:BD 98 F0  240 CIMLP   LDA  LETC,X
F0C9:95 D6     241         STA  OBJCIM,X
F0CB:E8        242         INX
F0CC:E0 08     243         CPX  #8
F0CE:D0 F6     244         BNE  CIMLP
F0D0:          245 *
F0D0:A9 82     246         LDA  #$82      ; SET UP PLAYER1 COLOR
F0D2:85 AB     247         STA  P1CR
F0D4:A9 DA     248         LDA  #$DA      ; SET UP PLAYER2 COLOR
F0D6:85 AC     249         STA  P2CR
```

D

```
F0D8:A9 EA     250         LDA  #$EA      ; SET UP BACKGROUND COLOR
F0DA:85 AD     251         STA  BKCR
F0DC:A9 64     252         LDA  #$64      ; SET UP COLOR FOR OBJECTS
F0DE:85 AE     253         STA  OBJCR
F0E0:A9 05     254         LDA  #5        ; SET UP THE SCREEN MODE PARAMETER
F0E2:85 A5     255         STA  SMODE
F0E4:A9 02     256         LDA  #2        ; SET UP DIGCON
F0E6:85 87     257         STA  DIGCON
F0E8:A9 2B     258         LDA  #$2B      ; NUMBER OF TIMER CLICKS BETWEEN SCREENS
F0EA:85 A9     259         STA  TMCNT     ; ($2B GIVES TIME FOR 36 LINES)
F0EC:          260 *
F0EC:          261 *     MAIN PROGRAM LOOP
F0EC:          262 *
F0EC:          263 *
F0EC:          264 *
F0EC:20 4B F2  265 MAIN    JSR  SYNCH     ; DO THE VERTICAL SYNCH
F0EF:20 75 F2  266         JSR  FMON      ; GO SEE IF USER HAS SOMETHING TO DO
F0F2:20 FB F0  267         JSR  SSETUP    ; SET UP THE ONCE/SCREEN REGISTERS
F0F5:20 14 F1  268         JSR  DISPLAY   ; DO THE LINE BY LINE DISPLAY
F0F8:4C EC F0  269         JMP  MAIN      ; LOOP ALWAYS
F0FB:          270 *
F0FB:          271 *
F0FB:          272 SSETUP  EQU  *
F0FB:A5 89     273         LDA  P1HS      ; PUT HORIZONTAL SHADOW REGISTERS INTO
F0FD:85 20     274         STA  P1HORZ    ; THE VIDEO REGISTERS
F0FF:A5 8A     275         LDA  P2HS
F101:85 21     276         STA  P2HORZ
F103:A5 8B     277         LDA  S1HS
F105:85 22     278         STA  S1HORZ
F107:A5 8C     279         LDA  S2HS
F109:85 23     280         STA  S2HORZ
F10B:          281 *
F10B:A5 AD     282         LDA  BKCR      ; GET THE BACKGROUND COLOR
F10D:85 09     283         STA  BCOLOR
F10F:A5 AE     284         LDA  OBJCR     ; UPDATE THE OBJECT COLOR
F111:85 08     285         STA  OCOLOR
F113:          286 *
F113:          287 *       DONE WITH THE ONCE PER SCREEN STUFF
F113:60        288         RTS
F114:          289 *
F114:          290 *
F114:          291 *       DISPLAY
F114:          292 *
F114:          293 * THE DISPLAY ROUTINE IS CALLED ONCE PER SCREEN TO DO A LINE BY LINE UPDATE
F114:          294 * OF THE VIDEO REGISTERS.
F114:          295 *
F114:          296 *
F114:          297 * NOTE: THIS LOOP LETS TWO HORIZONTAL SYNCH PULSES GO BEFORE
F114:          298 * INCREMENT OF VERTICAL LINE COUNT.
F114:          299 * THIS CAUSES ALL SCREEEN PATTERNS TO BE TWO LINES HIGH FOR EACH LOOP
F114:          300 * THROUGH DISPLAY.
F114:          301 *
F114:          302 *
F114:          303 *
F114:A9 00     304 DISPLAY LDA  #0        ; RESET THE LINE COUNTER
F116:85 85     305         STA  VLNCT
F118:          306 * HERE IS A TEST TO SEE IF WE GO TO THE SINGLE LINE RESOLUTION DISPLAY
F118:A5 AA     307         LDA  SPTS      ; GET THE CONTROL BITS
F11A:29 40     308         AND  #$40      ; SEE IF WE GO TO THE OTHER DISPLAY SECTION
F11C:F0 03     309         BEQ  DOUBLE    ; IF NOT CONTINUE WITH DOUBLE LINE ROUTINE
F11E:4C 08 F3  310         JMP  SINGLE
F121:85 02     311 DOUBLE  STA  LWAIT     ; WAIT FOR END OF LINE
F123:85 2A     312         STA  HZSCRL    ; DONE AT END OF LINE THIS CAUSES THE
F125:          313 *                      HORIZONTAL REGISTERS TO BE UPDATED FOR MOTION
F125:          314 *
F125:AD 84 02  315 TWAIT   LDA  RTIME     ; WAIT FOR THE TIMER TO GET TO ZERO
```

E

```
F128:D0 FB     316         BNE  TWAIT
F12A:          317 *
F12A:85 02     318         STA  LWAIT
F12C:85 01     319         STA  VRESET    ; TURN SCREEN ON
F12E:          320 *
F12E:BA        321         TSX            ; SAVE STACK POINTER SO WE CAN USE A PHP INSTRUCTION TO GET THE
F12F:          322 *               FLAG REGISTER BELOW
F12F:86 86     323         STX  SPREG
F131:A5 87     324         LDA  DIGCON    ; GET THE DIGIT CONTROL BYTE
F133:85 0A     325         STA  FVMODE    ; SEND IT TO THE VIDEO
F135:A6 88     326         LDX  TOPCON    ; GET THE CONTROL BYTE FOR THE TOP OF SCREEN
F137:85 02     327 TOPWT   STA  LWAIT     ; WAIT THAT MANY LINES
F139:CA        328         DEX
F13A:D0 FB     329         BNE  TOPWT
F13C:          330 *
F13C:          331 * DISPLAY TOP PART OF SCREEN
F13C:          332 *
F13C:A2 00     333         LDX  #0        ; DO A SIX LINE LOOP
F13E:          334 *
F13E:85 02     335 TOPLP   STA  LWAIT     ; WAIT FOR EOL
F140:B5 8D     336         LDA  TREG1,X   ; FIRST IMAGE BLOCK
F142:85 0E     337         STA  FLDBIM    ; PUT ON SCREEN
F144:B5 93     338         LDA  TREG2,X   ; NEXT FIELD
F146:85 0F     339         STA  FLDCIM
F148:EA        340         NOP            ; WAIT A WHILE
F149:EA        341         NOP
F14A:EA        342         NOP
F14B:EA        343         NOP
F14C:EA        344         NOP
F14D:EA        345         NOP
F14E:EA        346         NOP
F14F:EA        347         NOP
F150:EA        348         NOP
F151:B5 99     349         LDA  TREG3,X
F153:85 0E     350         STA  FLDBIM    ; PUT UP NEXT FIELD
F155:B5 9F     351         LDA  TREG4,X
F157:85 0F     352         STA  FLDCIM    ; PUT IN 4TH FIELD
F159:          353 *
F159:85 02     354         STA  LWAIT     ; WAIT FOR THE NEXT LINE
F15B:B5 8D     355         LDA  TREG1,X   ; GET NEXT IMAGE
F15D:85 0E     356         STA  FLDBIM    ; PUT IT UP
F15F:B5 93     357         LDA  TREG2,X   ; NOW SECOND FIELD AGAIN
F161:85 0F     358         STA  FLDCIM
F163:EA        359         NOP            ; WAIT THE SAME WHILE
F164:EA        360         NOP
F165:EA        361         NOP
F166:EA        362         NOP
F167:EA        363         NOP
F168:EA        364         NOP
F169:EA        365         NOP
F16A:EA        366         NOP
F16B:EA        367         NOP
F16C:B5 99     368         LDA  TREG3,X
F16E:85 0E     369         STA  FLDBIM    ; PART 3
F170:B5 9F     370         LDA  TREG4,X   ; GET PART 4
F172:85 0F     371         STA  FLDCIM    ; PUT UP LAST PART
F174:          372 *
F174:E8        373         INX            ; MOVE TO NEXT GROUP
F175:E0 06     374         CPX  #6
F177:D0 C5     375         BNE  TOPLP
F179:          376 *
F179:A9 00     377 PART2   LDA  #0        ; CLEAR OUT VID REGISTER
F17B:85 02     378         STA  LWAIT     ; WAIT FOR LINE TO END
F17D:85 0E     379         STA  FLDBIM
F17F:85 0F     380         STA  FLDCIM
F181:          381 *
```

F

```
F1EF:A9 00     448 NOA     LDA  #0        ; NOT AT OBJECT A
F1F1:F0 03     449         BEQ  OADSP     ; GO FILL WITH 0
F1F3:AA        450 INOA    TAX            ; GO GET LINE OF IMAGE
F1F4:B5 C4     451         LDA  OBJAIM,X
F1F6:85 0D     452 OADSP   STA  FLDAIM    ; SEND TO OBJECT A
F1F8:          453 *
F1F8:A5 A8     454 OBTST   LDA  EBITS     ; SEE IF WE DO 4
F1FA:29 08     455         AND  #8
F1FC:F0 14     456         BEQ  OCTST
F1FE:38        457         SEC               ; TEST FOR VERT CLOSE TO OBJECT B
F1FF:A5 85     458         LDA  VLNCT
F201:E5 CC     459         SBC  OBJBVT    ; SUB OBJ B VERTICAL
F203:90 04     460         BCC  NOB       ; IF CLEAR, NOT THERE YET
F205:C9 08     461         CMP  #8        ; IF > 7, GONE TOO FAR
F207:90 04     462         BCC  INOB
F209:A9 00     463 NOB     LDA  #0        ; NOT AT OBJECT B
F20B:F0 03     464         BEQ  OBDSP     ; GO FILL WITH 0
F20D:AA        465 INOB    TAX            ; GO GET LINE OF IMAGE
F20E:B5 CD     466         LDA  OBJBIM,X
F210:85 0E     467 OBDSP   STA  FLDBIM    ; SEND TO OBJECT B
F212:          468 *
F212:A5 A8     469 OCTST   LDA  EBITS     ; SEE IF WE DO 5
F214:29 10     470         AND  #16
F216:F0 14     471         BEQ  ENDLP
F218:38        472         SEC               ; TEST FOR VERT CLOSE TO OBJECT C
F219:A5 85     473         LDA  VLNCT
F21B:E5 D5     474         SBC  OBJCVT    ; SUB OBJ C VERTICAL
F21D:90 04     475         BCC  NOC       ; IF CLEAR, NOT THERE YET
F21F:C9 08     476         CMP  #8        ; IF > 7, GONE TOO FAR
F221:90 04     477         BCC  INOC
F223:A9 00     478 NOC     LDA  #0        ; NOT AT OBJECT C
F225:F0 03     479         BEQ  OCDSP     ; GO FILL WITH 0
F227:AA        480 INOC    TAX            ; GO GET LINE OF IMAGE
F228:B5 D6     481         LDA  OBJCIM,X
F22A:85 0F     482 OCDSP   STA  FLDCIM    ; SEND TO OBJECT C
F22C:          483 *
F22C:E6 85     484 ENDLP   INC  VLNCT     ; MOVE TO NEXT LINE
F22E:A5 85     485         LDA  VLNCT     ; CHECK TO SEE IF DONE
F230:C9 66     486         CMP  #$66
F232:F0 03     487         BEQ  LPDONE    ; WITH THE MAIN SCREEN LOOP
F234:4C 9B F1  488         JMP  SLOOP     ; IF NOT GO BACK FOR ANOTHER LINE
F237:          489 *
F237:          490 *
F237:A6 86     491 LPDONE  LDX  SPREG     ; DONE WITH SCREEN SO PUT THE STACK POINTER BACK
F239:9A        492         TXS
F23A:          493 *    NOW SOME END OF PAGE REGISTER STUFFING
F23A:A9 00     494         LDA  #0
F23C:85 1D     495         STA  P1SHOT
F23E:85 1E     496         STA  P2SHOT
F240:85 1B     497         STA  P1IMAG
F242:85 1C     498         STA  P2IMAG
F244:85 0D     499         STA  FLDAIM
F246:85 0E     500         STA  FLDBIM
F248:85 0F     501         STA  FLDCIM
F24A:          502 *            DONE WITH DISPLAY
F24A:60        503         RTS
F24B:          504 *
F24B:          505 *
F24B:          506 *
F24B:          507 *
F24B:          508 *
F24B:          509 *
F24B:          510 *
F24B:E6 80     511 SYNCH   INC  SCREENS   ; UPDATE SCREEN COUNT
F24D:A9 3C     512         LDA  #60       ; TEST FOR OVERFLOW INTO SECONDS
F24F:45 80     513         EOR  SCREENS
```

```
F181:A5 A5     382          LDA  SMODE     ; GET THE SCREEN MODE PARAMETER
F183:85 0A     383          STA  FVMODE    ; WHAT?
F185:A5 AB     384          LDA  P1CR      ; GET THE COLOR FOR PLAYER 1
F187:85 06     385          STA  P1COLOR
F189:A5 AC     386          LDA  P2CR      ; GET THE COLOR FOR PLAYER 2
F18B:85 07     387          STA  P2COLOR
F18D:A5 AA     388          LDA  SPTS      ; GET ENABLE BITS
F18F:85 A8     389          STA  EBITS     ; FEED AS PARAMETERS TO NEXT SECTION
F191:10 08     390          BPL  SLOOP     ; SHOW EACH SCREEN
F193:A5 B0     391          LDA  SCREENS   ; ELSE ONLY ENABLE EVERY OTHER SCREEN
F195:29 01     392          AND  #1
F197:D0 02     393          BNE  SLOOP
F199:85 A8     394          STA  EBITS     ; CLEAR ENABLE BITS
F19B:          395 *
F19B:          396 *
F19B:A2 1E     397 SLOOP    LDX  #P2SHOT   ; USE THE STACK MECH TO SEND OUT SHOTS
F19D:9A        398          TXS
F19E:          399 *
F19E:85 02     400          STA  LWAIT     ; THIS IS A LINE BY LINE LOOP
F1A0:          401 *
F1A0:A5 B2     402          LDA  S2VERT    ; SEE IF TIME TO DISPLAY SHOT
F1A2:45 85     403          EOR  VLNCT
F1A4:08        404          PHP             ; THIS WILL DISPLAY IF Z FLAG SET
F1A5:A5 B1     405          LDA  S1VERT
F1A7:45 85     406          EOR  VLNCT
F1A9:08        407          PHP
F1AA:          408 *
F1AA:A5 A8     409          LDA  EBITS     ; SEE IF WE SHOULD DO THIS SECTION
F1AC:29 01     410          AND  #1
F1AE:F0 14     411          BEQ  P2TST     ; NOT IF THE BIT IS ZERO
F1B0:38        412          SEC                ; TEST FOR VERT CLOSE TO PLAYER 1
F1B1:A5 85     413          LDA  VLNCT     ; GET THE VERTICAL POSITION AT THIS LINE
F1B3:E5 AF     414          SBC  P1VERT    ; SUBTRACT THE PLAYER1 VERTICAL POSITION
F1B5:90 04     415          BCC  NOP1      ; IF CARRY CLEAR THEN WE ARE NOT THERE YET
F1B7:C9 08     416          CMP  #8        ; A NOW HOLDS THE INDEX TO THE IMAGE ARRAY
F1B9:90 04     417          BCC  INP1      ; IF CARRY CLEAR WE ARE IN RANGE TO SHOW A LINE OF P1 IMAGE
F1BB:A9 00     418 NOP1     LDA  #0        ; IF HERE, NO SHOW
F1BD:F0 03     419          BEQ  P1DSP
F1BF:AA        420 INP1     TAX            ; GO GET LINE OF IMAGE
F1C0:B5 B3     421          LDA  P1IMG,X
F1C2:85 1B     422 P1DSP    STA  P1IMAG    ; DISPLAY PLAYER1 IMAGE
F1C4:          423 *
F1C4:A5 A8     424 P2TST    LDA  EBITS     ; SEE IF WE SHOULD DO SECTION 2
F1C6:29 02     425          AND  #2
F1C8:F0 14     426          BEQ  OATST
F1CA:38        427          SEC              ; GET READY FOR SUBTRACT
F1CB:A5 85     428          LDA  VLNCT     ; GET THE VERTICAL POSITION AT THIS LINE
F1CD:E5 B0     429          SBC  P2VERT    ; SUBTRACT THE PLAYER2 VERTICAL POSITION
F1CF:90 04     430          BCC  NOP2      ; IF CARRY CLEAR THEN WE ARE NOT THERE YET
F1D1:C9 08     431          CMP  #8        ; A NOW HOLDS THE INDEX TO THE IMAGE ARRAY
F1D3:90 04     432          BCC  IMP2      ; IF CARRY CLEAR WE ARE IN RANGE TO SHOW A LINE OF P2 IMAGE
F1D5:A9 00     433 NOP2     LDA  #0        ; IF HERE, NO P2 SHOW
F1D7:F0 03     434          BEQ  P2DSP
F1D9:AA        435 IMP2     TAX                ; GO GET LINE OF IMAGE
F1DA:B5 BB     436          LDA  P2IMG,X
F1DC:85 1C     437 P2DSP    STA  P2IMAG    ; DISPLAY PLAYER2 IMAGE
F1DE:          438 *
F1DE:A5 A8     439 OATST    LDA  EBITS     ; SEE IF WE DO SECTION 3
F1E0:29 04     440          AND  #4        .
F1E2:F0 14     441          BEQ  OBTST
F1E4:38        442          SEC              ; TEST FOR VERT CLOSE TO OBJECT A
F1E5:A5 85     443          LDA  VLNCT
F1E7:E5 C3     444          SBC  OBJAVT    ; SUBTRACT THE OBJECT A VERTICAL POSITION
F1E9:90 04     445          BCC  NOA         ; IF CLEAR THEN NOT THERE
F1EB:C9 08     446          CMP  #8        ; IF INDEX PAST 8 THEN
F1ED:90 04     447          BCC  INOA      ; KEEP GOING, ELSE DISPLAY
```

```
F251:D0 04     514         BNE  SYN1      ; BRANCH IF NO OVERFLOW
F253:85 80     515         STA  SCREENS   ; ELSE ZERO SCREENS AND
F255:E6 81     516         INC  SECONDS   ; UPDATE THE SECONDS COUNT
F257:          517 *
F257:A9 02     518 SYN1    LDA  #2        ; GET READY FOR VERTICAL SYNCH PULSE GENERATION
F259:85 02     519         STA  LWAIT     ; WAIT FOR END OF LINE
F25B:85 01     520         STA  VRESET    ; RESET VIDEO
F25D:85 02     521         STA  LWAIT     ; WAIT THREE LINES
F25F:85 02     522         STA  LWAIT
F261:85 02     523         STA  LWAIT
F263:85 00     524         STA  VSYNCH    ; START SYNCH PULSE
F265:A9 00     525         LDA  #0        ; WRITE 0 TO END PULSE
F267:85 02     526         STA  LWAIT     ; BUT FIRST WAIT THREE LINES
F269:85 02     527         STA  LWAIT
F26B:85 02     528         STA  LWAIT
F26D:85 00     529         STA  VSYNCH    ; END PULSE
F26F:A5 A9     530         LDA  TMCNT     ; HAVE TIMER COUNT THIS DOWN
F271:8D 96 02  531         STA  TIM64     ; IN DIVIDE BY 64 MODE TO GIVE US
F274:          532 *   TIME TO DO SOME STUFF BEFORE WE NEED TO DISPLAY SCREEN.
F274:          533 *   WHEN THAT TIME COMES WE WILL BE DISPLAYING LINE BY LINE
F274:          534 *   AND TIME WILL BE VERY SHORT.
F274:          535 *
F274:60        536         RTS            ; DONE WITH VERTICAL SYNCH
F275:          537 *
F275:          538 *
F275:          539 *
F275:          540 *
F275:          541 * FMON
F275:          542 *  COMMANDS
F275:          543 *
F275:          544 *    10 - SET ADDRESS (2 BYTE)
F275:          545 *    20 - READ BYTE
F275:          546 *    40 - WRITE BYTE
F275:          547 *
F275:          548 *  SUB COMMANDS
F275:          549 *
F275:          550 *    X0 - DO NOTHING
F275:          551 *    X1 - POST INCREMENT
F275:          552 *    X2 - POST DECREMENT
F275:          553 *
F275:          554 *  MEMORY USAGE
F275:          555 *
F275:          556 *   TARL - TARGET ADDRESS LOW
F275:          557 *   TARH - TARGET ADDRESS HIGH
F275:          558 *   COMCOD - COMMAND CODE STORE
F275:          559 * DECLARATIONS
FFF0:          560 AR1     EQU  $FFF0     ; FMON WRITE REG
FFF1:          561 AR2     EQU  $FFF1     ; FMON STATUS
FFF2:          562 AR3     EQU  $FFF2     ; FMON READ REG
F275:          563 *
F275:          564 *
F275:AD F1 FF  565 FMON    LDA  AR2       ; LOOK TO SEE IF COMMAND WAITING
F278:29 40     566         AND  #$40
F27A:F0 0F     567         BEQ  NOCOM     ; IF NOT JUST EXIT
F27C:AD F2 FF  568         LDA  AR3       ; GET THE COMMAND
F27F:85 84     569         STA  COMCOD
F281:29 10     570         AND  #$10
F283:F0 07     571         BEQ  L1
F285:20 B6 F2  572         JSR  SETADDR
F288:4C 75 F2  573         JMP  FMON
F28B:60        574 NOCOM   RTS              ; IF NO COMMAND JUST RETURN
F28C:A5 84     575 L1      LDA  COMCOD
F28E:29 20     576         AND  #$20
F290:F0 06     577         BEQ  L2
F292:20 C7 F2  578         JSR  READ
F295:4C 75 F2  579         JMP  FMON
```

I

```
F298:A5 84     580 L2      LDA  COMCOD
F29A:29 40     581         AND  #$40
F29C:F0 03     582         BEQ  L3
F29E:20 E9 F2  583         JSR  WRITE
F2A1:4C 75 F2  584 L3      JMP  FMON
F2A4:          585 *
F2A4:          586 *  READOK
F2A4:          587 *
F2A4:          588 * CHECK IF IT IS OK TO READ APPLE
F2A4:          589 *
F2A4:AD F1 FF  590 READOK  LDA  AR2       ; GET STATUS BYTE
F2A7:29 40     591         AND  #$40      ; OK TO READ?
F2A9:F0 F9     592         BEQ  READOK    ; NO KEEP LOOKING
F2AB:60        593         RTS             ; YES
F2AC:68        594         PLA
F2AD:60        595 DOREAD  RTS
F2AE:          596 *
F2AE:          597 *
F2AE:AD F1 FF  598 WRITEOK LDA  AR2
F2B1:29 80     599         AND  #$80
F2B3:F0 F9     600         BEQ  WRITEOK
F2B5:60        601         RTS
F2B6:          602 *
F2B6:          603 * SETADDR
F2B6:          604 *
F2B6:          605 *
F2B6:20 A4 F2  606 SETADDR JSR  READOK
F2B9:AD F2 FF  607         LDA  AR3
F2BC:85 83     608         STA  TARH
F2BE:20 A4 F2  609         JSR  READOK
F2C1:AD F2 FF  610         LDA  AR3
F2C4:85 82     611         STA  TARL
F2C6:60        612         RTS
F2C7:          613 *
F2C7:          614 * READ
F2C7:          615 *
F2C7:          616 *
F2C7:20 AE F2  617 READ    JSR  WRITEOK
F2CA:A0 00     618         LDY  #0
F2CC:B1 82     619         LDA  (TARL),Y
F2CE:8D F0 FF  620         STA  AR1
F2D1:A5 84     621         LDA  COMCOD
F2D3:29 07     622         AND  #$07
F2D5:F0 09     623         BEQ  RL1
F2D7:29 01     624         AND  #$01
F2D9:F0 08     625         BEQ  RL2
F2DB:C6 82     626         DEC  TARL
F2DD:4C E8 F2  627         JMP  RXT
F2E0:4C E8 F2  628 RL1     JMP  RXT
F2E3:E6 82     629 RL2     INC  TARL
F2E5:4C E8 F2  630         JMP  RXT
F2E8:60        631 RXT     RTS
F2E9:          632 *
F2E9:          633 * WRITE
F2E9:          634 *
F2E9:20 A4 F2  635 WRITE   JSR  READOK
F2EC:AD F2 FF  636         LDA  AR3
F2EF:A0 00     637         LDY  #0
F2F1:91 82     638         STA  (TARL),Y
F2F3:A5 84     639         LDA  COMCOD
F2F5:29 07     640         AND  #$07
F2F7:F0 09     641         BEQ  WL1
F2F9:29 01     642         AND  #$01
F2FB:F0 08     643         BEQ  WL2
F2FD:C6 82     644         DEC  TARL
F2FF:4C 0A F3  645         JMP  WXT
```

```
F302:4C 0A F3  646 WL1     JMP  WXT
F305:E6 82     647 WL2     INC  TARL
F307:4C 0A F3  648         JMP  WXT
F30A:60        649 WXT     RTS
F30B:          650 *
F30B:          651 *
F30B:          652 *
F30B:          653 * HERE IS SINGLE HORIZONTAL LINE RESOLUTION DISPLAY ROUTINE.
F30B:          654 * THIS STUFF WAS ADDED IN 1/83 TO DEMONSTRATE HORIZONTAL POSITIONING
F30B:          655 *   WE WILL BE DOING A SEQUENCE OF GRAPHIC EXPERIMENTS AS WE GO DOWN THE
F30B:          656 * THE SCREEN.
F30B:          657 *
F30B:          658 *
F30B:          659 *
F30B:          660 * FIRST SOME INITIALIZATION
F30B:          661 SINGLE  EQU  *         ; HERE WHEN STARTING SINGLE LINE MODE
F30B:A2 07     662         LDX  #7        ; COPY FROBCO LETTERS INTO IMAGE BUFFERS
F30D:BD 1E F5  663 CPLP    LDA  FROB1,X   ; GET THE F
F310:95 B3     664         STA  P1IMG,X   ; PUT IN BUFFER
F312:BD 26 F5  665         LDA  FROB2,X   ; GET THE R
F315:95 BB     666         STA  P2IMG,X   ; PUT IN BUFFER
F317:BD 2E F5  667         LDA  FROB3,X   ; GET THE O
F31A:95 C4     668         STA  OBJAIM,X  ; BORROW THIS BUFFER
F31C:BD 36 F5  669         LDA  FROB4,X   ; AND THE OTHER OBJ BUFFERS
F31F:95 CD     670         STA  OBJBIM,X  ; FROM THE OTHER SCREEN
F321:BD 3E F5  671         LDA  FROB5,X   ; THE C
F324:95 D6     672         STA  OBJCIM,X
F326:BD 46 F5  673         LDA  FROB6,X   ; THE ROUND O
F329:95 DE     674         STA  OBJXIM,X  ; WE ADDED THIS BUFFER
F32B:CA        675         DEX
F32C:10 DF     676         BPL  CPLP      ; DO THE ABOVE FOR 8 LINES OF DATA
F32E:A9 05     677         LDA  #5        ; SET UP TAB STOP FOR 6 IMAGE EXPERIMENT
F330:85 E6     678         STA  TP1
F332:A9 30     679         LDA  #$30      ; SET TP25 TO NOTHING
F334:85 E7     680         STA  TP25
F336:          681 *
F336:          682 *
F336:          683 * END OF SINGLE LINE MODE INIT
F336:          684 *
F336:          685 * NOW THE ONCE PER SCREEN STUFF
F336:          686 DISPLAY2 EQU *
F336:20 75 F2  687         JSR  FMON      ; LET THE USER ADJUST PARAMETERS
F339:A5 AA     688         LDA  SPTS      ; GET THE ENABLE BITS
F33B:29 40     689         AND  #$40      ; SEE IF WE ARE STILL IN SINGLE LINE MODE
F33D:D0 03     690         BNE  HANGIN    ; IF SO HANG IN THERE
F33F:4C 00 F0  691         JMP  START     ; ELSE START OVER
F342:          692 HANGIN  EQU  *
F342:          693 *
F342:A5 A5     694         LDA  SMODE     ; SHADOW OF THE FIELD MODE REGISTER
F344:85 0A     695         STA  FVMODE
F346:A5 AB     696         LDA  P1CR      ; GET THE COLOR FOR PLAYER 1
F348:85 06     697         STA  P1COLOR
F34A:A5 AC     698         LDA  P2CR      ; GET THE COLOR FOR PLAYER 2
F34C:85 07     699         STA  P2COLOR
F34E:A5 AD     700         LDA  BKCR      ; GET THE BACKGROUND COLOR
F350:85 09     701         STA  BCOLOR
F352:A5 AE     702         LDA  OBJCR     ; UPDATE THE OBJECT COLOR
F354:85 08     703         STA  OCOLOR
F356:          704 *
F356:AD 84 02  705 STWAIT  LDA  RTIME     ; WAIT FOR TIMER TO GET TO 0
F359:D0 FB     706         BNE  STWAIT
F35B:          707 *
F35B:85 02     708         STA  LWAIT
F35D:85 01     709         STA  VRESET    ; TURN SCREEN ON
F35F:          710 *
F35F:          711 *
```

K

```
F35F:A2 17     712         LDX  #23       ; GO DOWN TO LINE 60
F361:85 02     713 LN60    STA  LWAIT
F363:CA        714         DEX
F364:D0 FB     715         BNE  LN60
F366:          716 *
F366:          717 *
F366:          718 *      EXEPERIMENT NUMBER 1, PLAYER HORIZONTAL POSITIONING
F366:          719 *
F366:          720 * IN THIS EXPERIMENT WE TAKE THE VALUE THE USER HAS SET FOR HORIZONTAL
F366:          721 * AND USE A POSITIONING ALGORITHM COMPOSED OF A COURSE PART AND
F366:          722 * A FINE RESOLUTION PART.  IT IS NOT TOTALY LINIER, BUT IT SHOWS THE WAY.
F366:          723 *
F366:A9 00     724         LDA  #0        ; SET PLAYER FOR SINGLE IMAGE MODE
F368:85 04     725         STA  P1VMOD
F36A:A5 89     726         LDA  P1HS      ; SET H SETTING
F36C:4A        727         LSR  A         ; GET COURSE BITS
F36D:4A        728         LSR  A
F36E:4A        729         LSR  A
F36F:4A        730         LSR  A
F370:18        731         CLC            ; COURSE POSITION STARTS AT 4
F371:69 04     732         ADC  #4        ; SO BACK IT UP
F373:AA        733         TAX            ; SAVE IN X
F374:A5 89     734         LDA  P1HS      ; GET HORIZONTAL SHADOW
F376:0A        735         ASL  A         ; MOVE TO HIGH NIBBLE
F377:0A        736         ASL  A
F378:0A        737         ASL  A
F379:0A        738         ASL  A
F37A:49 70     739         EOR  #$70      ; MAKE NUMBERS GO FROM LEFT TO RIGHT
F37C:85 20     740         STA  P1HORZ    ;  FINE OFFSET
F37E:85 02     741         STA  LWAIT     ; WAIT FOR END OF LINE
F380:CA        742 HFP1    DEX            ; SELECT CORSE POSITION BY THE TIME IN THIS LOOP
F381:10 FD     743         BPL  HFP1
F383:85 10     744         STA  P1HRES    ; THIS IS THE COURSE SETTING PROCESS
F385:          745 *
F385:          746 * NOW CAUSE THE HORZ INC TO BE ADDED TO THE ABSOLUTE POSITION
F385:85 02     747         STA  LWAIT     ; GET TO END OF LINE
F387:85 2A     748         STA  HZSCRL    ; ADD INCREMENT
F389:          749 *
F389:          750 *
F389:A2 00     751         LDX  #0        ; INDEX TO PLAYER IMAGE
F38B:85 02     752 IMLP1   STA  LWAIT     ; WAIT TILL NEXT LINE
F38D:B5 B3     753         LDA  P1IMG,X   ; GET LINE OF PLAYER IMAGE
F38F:85 1B     754         STA  P1IMAG    ; PUT OUT TO REGISTER
F391:E8        755         INX
F392:E0 08     756         CPX  #8        ; SEE IF DONE
F394:90 F5     757         BCC  IMLP1     ; IF NOT GO DO ANOTHER LINE
F396:85 02     758         STA  LWAIT     ; FINISH THE LINE
F398:A9 00     759         LDA  #0
F39A:85 1B     760         STA  P1IMAG    ; TURN OFF PLAYER
F39C:          761 *
F39C:          762 * WE ARE NOW AT LINE 71
F39C:          763 *
F39C:          764 * NOW DO THE SAME FOR PLAYER2
F39C:          765 *
F39C:A9 00     766         LDA  #0        ; SET PLAYER FOR SINGLE IMAGE MODE
F39E:85 05     767         STA  P2VMOD
F3A0:A5 8A     768         LDA  P2HS      ; SET H SETTING
F3A2:4A        769         LSR  A         ; GET COURSE BITS
F3A3:4A        770         LSR  A
F3A4:4A        771         LSR  A
F3A5:4A        772         LSR  A
F3A6:18        773         CLC            ; COURSE POSITION STARTS AT 4
F3A7:69 04     774         ADC  #4        ; SO BACK IT UP
F3A9:AA        775         TAX            ; SAVE IN X
F3AA:A5 8A     776         LDA  P2HS      ; GET HORIZONTAL SHADOW
F3AC:0A        777         ASL  A         ; MOVE TO HIGH NIBBLE
```

```
F3AD:0A       778         ASL  A
F3AE:0A       779         ASL  A
F3AF:0A       780         ASL  A
F3B0:49 70    781         EOR  #$70      ; MAKE NUMBERS GO FROM LEFT TO RIGHT
F3B2:85 21    782         STA  P2HORZ    ;  FINE OFFSET
F3B4:85 02    783         STA  LWAIT     ; WAIT FOR END OF LINE
F3B6:CA       784 HFP2    DEX            ; SELECT CORSE POSITION BY THE TIME IN THIS LOOP
F3B7:10 FD    785         BPL  HFP2
F3B9:85 11    786         STA  P2HRES    ; THIS IS THE COURSE SETTING PROCESS
F3BB:         787 *
F3BB:         788 * NOW CAUSE THE HORZ INC TO BE ADDED TO THE ABSOLUTE POSITION
F3BB:85 02    789         STA  LWAIT     ; GET TO END OF LINE
F3BD:85 2A    790         STA  HZSCRL    ; ADD INCREMENT
F3BF:         791 *
F3BF:         792 *
F3BF:A2 00    793         LDX  #0        ; INDEX TO PLAYER IMAGE
F3C1:85 02    794 IMLP2   STA  LWAIT     ; WAIT TILL NEXT LINE
F3C3:B5 BB    795         LDA  P2IMG,X   ; GET LINE OF PLAYER IMAGE
F3C5:85 1C    796         STA  P2IMAG    ; PUT OUT TO REGISTER
F3C7:E8       797         INX
F3C8:E0 08    798         CPX  #8        ; SEE IF DONE
F3CA:90 F5    799         BCC  IMLP2     ; IF NOT GO DO ANOTHER LINE
F3CC:85 02    800         STA  LWAIT     ; FINISH THE LINE
F3CE:A9 00    801         LDA  #0
F3D0:85 1C    802         STA  P2IMAG    ; TURN OFF PLAYER
F3D2:         803 *
F3D2:         804 * WE ARE NOW AT LINE 82
F3D2:         805 *
F3D2:         806 *
F3D2:         807 * EXPERIMENT NUMBER 2, DOUBLE IMAGES
F3D2:         808 *
F3D2:         809 *
F3D2:A9 01    810         LDA  #1        ; SET PLAYER FOR DOUBLE IMAGE MODE
F3D4:85 04    811         STA  P1VMOD
F3D6:         812 *
F3D6:A2 00    813         LDX  #0        ; INDEX TO PLAYER IMAGE
F3D8:85 02    814 DMLP1   STA  LWAIT     ; WAIT TILL NEXT LINE
F3DA:A4 E6    815         LDY  TP1       ; WAIT BEFORE LOADING DISPLAY
F3DC:88       816 YWAIT1  DEY            ; THIS WAITS 5*TP1 CYCLES
F3DD:D0 FD    817         BNE  YWAIT1
F3DF:B5 B3    818         LDA  P1IMG,X   ; GET LINE OF PLAYER IMAGE
F3E1:85 1B    819         STA  P1IMAG    ; PUT OUT TO REGISTER
F3E3:B5 C4    820         LDA  OBJAIM,X  ; GET ANOTHER IMAGE
F3E5:85 1B    821         STA  P1IMAG    ; AND SEND IT ALSO
F3E7:E8       822         INX
F3E8:E0 08    823         CPX  #8        ; SEE IF DONE
F3EA:90 EC    824         BCC  DMLP1     ; IF NOT GO DO ANOTHER LINE
F3EC:85 02    825         STA  LWAIT     ; FINISH THE LINE
F3EE:A9 00    826         LDA  #0
F3F0:85 1B    827         STA  P1IMAG    ; TURN OFF PLAYER
F3F2:         828 *
F3F2:         829 * WE ARE NOW AT LINE 92
F3F2:         830 *
F3F2:         831 * NOW DO THE SAME FOR PLAYER2
F3F2:         832 *
F3F2:A9 01    833         LDA  #1        ; SET PLAYER FOR DOUBLE IMAGE MODE
F3F4:85 05    834         STA  P2VMOD
F3F6:         835 *
F3F6:A2 00    836         LDX  #0        ; INDEX TO PLAYER IMAGE
F3F8:85 02    837 DMLP2   STA  LWAIT     ; WAIT TILL NEXT LINE
F3FA:A4 E6    838         LDY  TP1       ; WAIT BEFORE LOADING DISPLAY
F3FC:88       839 YWAIT2  DEY            ; THIS WAITS 5*TP1 CYCLES
F3FD:D0 FD    840         BNE  YWAIT2
F3FF:B5 BB    841         LDA  P2IMG,X   ; GET LINE OF PLAYER IMAGE
F401:85 1C    842         STA  P2IMAG    ; PUT OUT TO REGISTER
F403:B5 CD    843         LDA  OBJBIM,X  ; GET ANOTHER IMAGE
```

```
F405:85 1C    844         STA  P2IMAG    ; PUT IT UP
F407:E8       845         INX
F408:E0 08    846         CPX  #8        ; SEE IF DONE
F40A:90 EC    847         BCC  DMLP2     ; IF NOT GO DO ANOTHER LINE
F40C:85 02    848         STA  LWAIT     ; FINISH THE LINE
F40E:A9 00    849         LDA  #0
F410:85 1C    850         STA  P2IMAG    ; TURN OFF PLAYER
F412:         851 *
F412:         852 * WE ARE NOW AT LINE 102
F412:         853 *
F412:         854 * EXPERIMENT NUMBER 3, ABSOLUTE CYCLE POSITIONING
F412:         855 *
F412:         856 * TO SHOW THE RELATION OF TIME TO SCREEN POSITION, WE WILL TURN THE VIDEO
F412:         857 * ON AND OFF AT SET TIMES
F412:         858 *
F412:         859 *
F412:A9 00    860         LDA  #0        ; CODE TO TURN VIDEO ON
F414:A2 02    861         LDX  #2        ; CODE TO TURN VIDEO OFF
F416:A0 05    862         LDY  #5        ; HOW MANY TIMES TO DO IT
F418:         863 *
F418:85 02    864         STA  LWAIT     ; START LINE
F41A:EA       865         NOP            ; WAIT 20 CYCLES
F41B:EA       866         NOP
F41C:EA       867         NOP
F41D:EA       868         NOP
F41E:EA       869         NOP
F41F:         870 *
F41F:EA       871         NOP
F420:EA       872         NOP
F421:EA       873         NOP
F422:EA       874         NOP
F423:EA       875         NOP
F424:86 01    876 VLP     STX  VRESET    ; TURN OFF VIDEO
F426:85 01    877         STA  VRESET    ; TURN ON 3 CYCLES LATER
F428:88       878         DEY            ; LOOP COUNT
F429:D0 F9    879         BNE  VLP       ; LOOP TILL EXPERIMENT COMPLETE
F42B:85 02    880         STA  LWAIT     ; MOVE TO LINE 105
F42D:         881 *
F42D:         882 *
F42D:         883 *
F42D:         884 * EXPERIMENT 4, 6 DIGIT DISPLAY
F42D:         885 * NOW WE WILL USE BOTH THE AREAS OF MEMORY OF THE PLAYER IMAGES
F42D:         886 * AND THE OBJECT IMAGES TO PUT UP A HIGH RESOLUTION 6 DIGIT DISPLAY.
F42D:         887 *
F42D:         888 *
F42D:A2 0F    889         LDX  #15       ; GO DOWN SOME LINES
F42F:85 02    890 DL120   STA  LWAIT
F431:CA       891         DEX
F432:D0 FB    892         BNE  DL120
F434:         893 * WE ARE NOW ON LINE 120
F434:         894 *
F434:A9 06    895         LDA  #6        ; SET PALYERS FOR TRIPLE IMAGE MODE
F436:85 04    896         STA  P1VMOD
F438:85 05    897         STA  P2VMOD
F43A:         898 *
F43A:         899 *
F43A:         900 * ALL TIMING IN THE FOLLOWING IS CRITICAL
F43A:         901 *
F43A:A5 E6    902         LDA  TP1       ; GET HORIZONTAL TIME PARAMETER
F43C:25 0F    903         AND  $0F       ; LOOK AT LOWER 4 BITS
F43E:AA       904         TAX            ; SAVE IN X
F43F:85 02    905         STA  LWAIT     ; GET SET
F441:CA       906 HFLP3   DEX            ; USE UP TIME
F442:10 FD    907         BPL  HFLP3
F444:         908 *
F444:85 10    909         STA  P1HRES    ; SET STOP FOR PLAYER1
```

```
F446:85 11    910         STA  P2HRES    ; SET STOP FOR PLAYER2
F448:         911 *
F448:85 02    912         STA  LWAIT     ; GO TO NEXT LINE
F44A:85 2A    913         STA  HZSCRL    ; ADD THE FINE OFFSETS
F44C:         914 *
F44C:         915 *
F44C:         916 *
F44C:         917 *
F44C:A2 00    918         LDX  #0        ; START OF LOOP COUNT
F44E:         919 *
F44E:         920 *
F44E:         921 * HERE IS THE MAIN LOOP
F44E:         922 D48LP   EQU  *
F44E:85 02    923         STA  LWAIT     ; WAIT FOR START OF LINE
F450:         924 *
F450:B5 B3    925         LDA  P1IMG,X   ; GET FIRST PART       (4 CYCLES)
F452:85 1B    926         STA  P1IMAG    ; PUT IN SHIFTER       (3 CYCLES)
F454:85 30    927         STA  $30       ; BURN 13 CYCLES
F456:EA       928         NOP
F457:EA       929         NOP
F458:EA       930         NOP
F459:EA       931         NOP
F45A:EA       932         NOP
F45B:B5 BB    933         LDA  P2IMG,X   ; GET THE SECOND PART  (4 CYCLES)
F45D:85 1C    934         STA  P2IMAG    ; PUT IN THE OTHER SHIFTER (3 CYCLES)
F45F:         935 *
F45F:B5 C4    936         LDA  OBJAIM,X  ; GET THE THIRD PART   (4 CYCLES)
F461:         937 *
F461:85 1B    938         STA  P1IMAG    ; REUSE SHIFTER        (3 CYCLES)
F463:B5 CD    939         LDA  OBJBIM,X  ; THE NEXT PART        (4 CYCLES)
F465:85 1C    940         STA  P2IMAG    ; PUT IT UP            (3 CYCLES)
F467:B5 D6    941         LDA  OBJCIM,X  ; GET THE FIFTH PART   (4 CYCLES)
F469:85 1B    942         STA  P1IMAG    ; PUT IT UP            (3 CYCLES)
F46B:B5 DE    943         LDA  OBJXIM,X  ; GET LAST PART        (4 CYCLES)
F46D:         944 *
F46D:85 1C    945         STA  P2IMAG    ; PUT THAT UP          (3 CYCLES)
F46F:         946 *
F46F:E8       947         INX            ; INCREMENT COUNT      (2 CYCLES)
F470:         948 *
F470:         949 *
F470:         950 *
F470:E0 08    951         CPX  #8        ; SEE IF DONE          (2 CYCLES)
F472:90 DA    952         BCC  D48LP     ; LOOP TILL 8 LINES DONE (3 CYCLES IF BRANCH TAKEN)
F474:         953 *
F474:85 02    954         STA  LWAIT     ; GO TO NEXT LINE
F476:A9 00    955         LDA  #0        ; CLEAR OUT IMAGES
F478:85 1B    956         STA  P1IMAG
F47A:85 1C    957         STA  P2IMAG
F47C:         958 *
F47C:         959 * THIS PUTS US AT LINE 116
F47C:         960 *
F47C:         961 * WE NOW DO THE SAME THING BUT THIS TIME WE WILL NOT USE LWAIT AND
F47C:         962 * WE HAVE PLACED A GENERAL STORE INSTRUCTION IN THE FRONT OF THE LOOP
F47C:         963 * SO WE CAN USE XCONTROL TO CHOOSE REGISTERS AND VALUES TO STORE
F47C:         964 * WHILE THE LOOP IS RUNNING.
F47C:         965 *
F47C:         966 *
F47C:         967 *
F47C:         968 *
F47C:A4 E7    969         LDY  TP25      ; PAGE 0 ADDRESS OF WHERE TO DO A STORE
F47E:A2 00    970         LDX  #0        ; RESET INDEX
F480:85 02    971         STA  LWAIT
F482:         972 *
F482:         973 * NOW A LOOP JUST AS ABOVE, EXCEPT NO STORES TO LWAIT
F482:         974 *
F482:         975 *
```

```
F482:          976 * HERE IS THE MAIN LOOP
F482:          977 D48LP2  EQU  *
F482:B5 B3     978         LDA  P1IMG,X   ; GET FIRST PART       (4 CYCLES)
F484:85 1B     979         STA  P1IMAG    ; PUT IN SHIFTER       (3 CYCLES)
F486:A5 E8     980         LDA  TP26      ; GET WHAT TO STORE    (3 CYCLES)
F488:99 00 00  981         STA  $0000,Y   ; STORE IT             (5 CYCLES)
F48B:85 30     982         STA  $30       ; BURN 5 CYCLES
F48D:EA        983         NOP
F48E:B5 BB     984         LDA  P2IMG,X   ; GET THE SECOND PART  (4 CYCLES)
F490:85 1C     985         STA  P2IMAG    ; PUT IN THE OTHER SHIFTER (3 CYCLES)
F492:          986 *
F492:B5 C4     987         LDA  OBJAIM,X  ; GET THE THIRD PART   (4 CYCLES)
F494:          988 *
F494:85 1B     989         STA  P1IMAG    ; REUSE SHIFTER        (3 CYCLES)
F496:B5 CD     990         LDA  OBJBIM,X  ; THE NEXT PART        (4 CYCLES)
F498:85 1C     991         STA  P2IMAG    ; PUT IT UP            (3 CYCLES)
F49A:B5 D6     992         LDA  OBJCIM,X  ; GET THE FIFTH PART   (4 CYCLES)
F49C:85 1B     993         STA  P1IMAG    ; PUT IT UP            (3 CYCLES)
F49E:B5 DE     994         LDA  OBJXIM,X  ; GET LAST PART        (4 CYCLES)
F4A0:          995 *
F4A0:85 1C     996         STA  P2IMAG    ; PUT THAT UP          (3 CYCLES)
F4A2:          997 *
F4A2:EA        998         NOP            ; BURN 14 CYCLES
F4A3:EA        999         NOP
F4A4:EA       1000         NOP
F4A5:EA       1001         NOP
F4A6:EA       1002         NOP
F4A7:EA       1003         NOP
F4A8:EA       1004         NOP
F4A9:E8       1005         INX             ; INCREMENT COUNT      (2 CYCLES)
F4AA:         1006 *
F4AA:         1007 *
F4AA:         1008 *
F4AA:E0 08    1009         CPX  #8        ; SEE IF DONE          (2 CYCLES)
F4AC:90 D4    1010         BCC  D48LP2    ; LOOP TILL 8 LINES DONE (3 CYCLES IF BRANCH TAKEN)
F4AE:         1011 *
F4AE:A9 00    1012         LDA  #0        ; CLEAR OUT IMAGES
F4B0:85 1B    1013         STA  P1IMAG
F4B2:85 1C    1014         STA  P2IMAG
F4B4:85 02    1015         STA  LWAIT
F4B6:         1016 *
F4B6:         1017 *
F4B6:         1018 *
F4B6:         1019 *
F4B6:         1020 * EXPERIMENT 5, HRES TIMING
F4B6:         1021 *
F4B6:         1022 * THIS IS JUST THE SAME AS EXPERIMENT 1, BUT WE HAVE PUT IN ONE NOP
F4B6:         1023 * THIS SHOWS THE EFFECT OF 2 CYCLES ON THE HRES TIMING
F4B6:         1024 *
F4B6:         1025 *
F4B6:A9 00    1026         LDA  #0        ; SET PLAYER FOR SINGLE IMAGE MODE
F4B8:85 04    1027         STA  P1VMOD
F4BA:A5 89    1028         LDA  P1HS      ; SET H SETTING
F4BC:4A       1029         LSR  A         ; GET COURSE BITS
F4BD:4A       1030         LSR  A
F4BE:4A       1031         LSR  A
F4BF:4A       1032         LSR  A
F4C0:18       1033         CLC            ; COURSE POSITION STARTS AT 4
F4C1:69 04    1034         ADC  #4        ; SO BACK IT UP
F4C3:AA       1035         TAX            ; SAVE IN X
F4C4:A5 89    1036         LDA  P1HS      ; GET HORIZONTAL SHADOW
F4C6:0A       1037         ASL  A         ; MOVE TO HIGH NIBBLE
F4C7:0A       1038         ASL  A
F4C8:0A       1039         ASL  A
F4C9:0A       1040         ASL  A
F4CA:49 70    1041         EOR  #$70      ; MAKE NUMBERS GO FROM LEFT TO RIGHT
```

P

```
F4CC:85 20    1042         STA  P1HORZ    ;  FINE OFFSET
F4CE:85 02    1043         STA  LWAIT     ; WAIT FOR END OF LINE
F4D0:EA       1044         NOP
F4D1:CA       1045 HFP3    DEX            ; SELECT CORSE POSITION BY THE TIME IN THIS LOOP
F4D2:10 FD    1046         BPL  HFP3
F4D4:85 10    1047         STA  P1HRES    ; THIS IS THE COURSE SETTING PROCESS
F4D6:         1048 *
F4D6:         1049 * NOW CAUSE THE HORZ INC TO BE ADDED TO THE ABSOLUTE POSITION
F4D6:85 02    1050         STA  LWAIT     ; GET TO END OF LINE
F4D8:85 2A    1051         STA  HZSCRL    ; ADD INCREMENT
F4DA:         1052 *
F4DA:         1053 *
F4DA:A2 00    1054         LDX  #0        ; INDEX TO PLAYER IMAGE
F4DC:85 02    1055 IMLP3   STA  LWAIT     ; WAIT TILL NEXT LINE
F4DE:B5 B3    1056         LDA  P1IMG,X   ; GET LINE OF PLAYER IMAGE
F4E0:85 1B    1057         STA  P1IMAG    ; PUT OUT TO REGISTER
F4E2:E8       1058         INX
F4E3:E0 08    1059         CPX  #8        ; SEE IF DONE
F4E5:90 F5    1060         BCC  IMLP3     ; IF NOT GO DO ANOTHER LINE
F4E7:85 02    1061         STA  LWAIT     ; FINISH THE LINE
F4E9:A9 00    1062         LDA  #0
F4EB:85 1B    1063         STA  P1IMAG    ; TURN OFF PLAYER
F4ED:         1064 *
F4ED:         1065 *
F4ED:         1066 *
F4ED:         1067 *
F4ED:         1068 *
F4ED:         1069 * THE LAST EXPERIMENT SHOWS THE RESULT OF COMBINING STORES TO HRES AND
F4ED:         1070 * THE IMAGE REGISTER  (I.E. NOTHING)
F4ED:         1071 *
F4ED:         1072 *
F4ED:A2 00    1073         LDX  #0        ; INDEX TO PLAYER IMAGE
F4EF:85 02    1074 IMLP4   STA  LWAIT     ; WAIT TILL NEXT LINE
F4F1:B5 B3    1075         LDA  P1IMG,X   ; GET LINE OF PLAYER IMAGE
F4F3:85 1B    1076         STA  P1IMAG    ; PUT OUT TO REGISTER
F4F5:EA       1077         NOP            ; BURN 14 CYCLES
F4F6:EA       1078         NOP
F4F7:EA       1079         NOP
F4F8:EA       1080         NOP
F4F9:EA       1081         NOP
F4FA:EA       1082         NOP
F4FB:EA       1083         NOP
F4FC:85 10    1084         STA  P1HRES    ; REPOSITION PLAYER
F4FE:EA       1085         NOP            ; WAIT 4 CYCLES
F4FF:EA       1086         NOP
F500:85 10    1087         STA  P1HRES    ; REPOSITION
F502:85 1B    1088         STA  P1IMAG    ; TRY TO DISPLAY
F504:85 10    1089         STA  P1HRES    ; REPOSITION
F506:E8       1090         INX
F507:E0 08    1091         CPX  #8        ; SEE IF DONE
F509:90 E4    1092         BCC  IMLP4     ; IF NOT GO DO ANOTHER LINE
F50B:85 02    1093         STA  LWAIT     ; FINISH THE LINE
F50D:A9 00    1094         LDA  #0
F50F:85 1B    1095         STA  P1IMAG    ; TURN OFF PLAYER
F511:         1096 *
F511:         1097 *
F511:         1098 *
F511:         1099 * DONE SO WAIT THE REST OF THE SCREEN
F511:A2 74    1100         LDX  #116      ; WAIT THE REST OF THE SCREEN
F513:85 02    1101 DWAIT   STA  LWAIT
F515:CA       1102         DEX
F516:D0 FB    1103         BNE  DWAIT
F518:         1104 *
F518:         1105 *
F518:         1106 *
F518:         1107 * NOW AT LINE 255 SO CALL HSYNCH
```

```
F518:20 4B F2 1108         JSR  SYNCH
F51B:4C 36 F3 1109         JMP  DISPLAY2  ; LOOP BACK
F51E:         1110 *
F51E:         1111 *
F51E:         1112 * NOW DATA FOR THE IMAGES
F51E:         1113 *
F51E:FF 63 F8 1114 FROB1   DFB  $FF,$63,$F8,$F8,$60,$60,$F0,$F0
F521:F8 60 60
F524:F0 F0
F526:FC 62 62 1115 FROB2   DFB            $FC,$62,$62,$7C,$78,$6C,$66,$E3
F529:7C 78 6C
F52C:66 E3
F52E:3C 7E C3 1116 FROB3   DFB  $3C,$7E,$C3,$C3,$C3,$C3,$7E,$3C
F531:C3 C3 C3
F534:7E 3C
F536:FC 62 62 1117 FROB4   DFB  $FC,$62,$62,$7C,$7C,$62,$62,$FC
F539:7C 7C 62
F53C:62 FC
F53E:FF FF 00 1118 FROB5   DFB  $FF,$FF,$00,$1C,$3C,$20,$3C,$1C
F541:1C 3C 20
F544:3C 1C
F546:FF FF 00 1119 FROB6   DFB  $FF,$FF,$00,$18,$3C,$24,$3C,$18
F549:18 3C 24
F54C:3C 18
F54E:         1120 *
F54E:         1121 *
F54E:         1122 *

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

R

```
 F0AE AIMLP       FFF0 AR1         FFF1 AR2         FFF2 AR3
   09 BCOLOR      F0BA BIMLP         AD BKCR        F0C6 CIMLP
 F008 CLOOP     ?  2C COLRES        84 COMCOD       F30D CPLP
?0280 CTRDLS     F44E D48LP       F482 D48LP2     ?0281 DDRA
?0283 DDRB         87 DIGCON      F114 DISPLAY      F336 DISPLAY2
 F42F DL120      F3D8 DMLP1       F3F8 DMLP2      ?F2AD DOREAD
 F121 DOUBLE     F513 DWAIT         A8 EBITS        F22C ENDLP
   0D FLDAIM       0E FLDBIM        0F FLDCIM       F275 FMON
 F51E FROB1      F526 FROB2       F52E FROB3        F536 FROB4
 F53E FROB5      F546 FROB6         0A FVMODE       F342 HANGIN
 F441 HFLP3      F380 HFP1        F3B6 HFP2         F4D1 HFP3
   2A HZSCRL     F38B IMLP1       F3C1 IMLP2        F4DC IMLP3
 F4EF IMLP4      F1D9 IMP2        F1F3 INOA         F20D INOB
 F227 INOC       F1BF INP1        F28C L1           F298 L2
 F2A1 L3         F088 LETA       ?F082 LETB         F090 LETB8
 F098 LETC       F070 LETF       ?F07C LETO        ?F076 LETR
?  3C LFTFR      F361 LN60        F237 LPDONE        02 LWAIT
 F0EC MAIN         A6 MODE1         A7 MODE2        F1EF NOA
 F209 NOB        F28B NOCOM       F223 NOC        ?  2B NOINC
 F1BB NOP1       F1D5 NOP2        F1F6 OADSP        F1DE OATST
 F210 OBDSP        C4 OBJAIM        C3 OBJAVT         CD OBJBIM
   CC OBJBVT       D6 OBJCIM        AE OBJCR          D5 OBJCVT
   DE OBJXIM     F1F8 OBTST       F22A OCDSP          08 OCOLOR
 F212 OCTST        06 P1COLOR       AB P1CR         ?  25 P1DELAY
 F1C2 P1DSP     ?  0B P1FLIP        20 P1HORZ         10 P1HRES
   89 P1HS         1B P1IMAG        B3 P1IMG        ?  32 P10COL
   1D P1SHOT       AF P1VERT        04 P1VMOD         07 P2COLOR
   AC P2CR      ?  26 P2DELAY    F1DC P2DSP        ?  0C P2FLIP
   21 P2HORZ       11 P2HRES        8A P2HS           1C P2IMAG
   BB P2IMG     ?  33 P20COL        1E P2SHOT       F1C4 P2TST
   B0 P2VERT       05 P2VMOD      ?F179 PART2       ?  38 PDL5L
?  3A PDL5R     ?  39 PDL9L      ?  3B PDL9R       ?  37 PPCOL
?  30 PS1COL    ?  31 PS2COL       F2A4 READOK       F2C7 READ
?  3D RGHFR       F2E0 RL1         F2E3 RL2          0284 RTIME
 F2E8 RXT       ?  28 S1CONT        22 S1HORZ      ?  12 S1HRES
   8B S1HS      ?  34 S10COL        B1 S1VERT      ?  29 S2CONT
   23 S2HORZ    ?  13 S2HRES        8C S2HS        ?  35 S20COL
   B2 S2VERT    ?  24 S3HORZ     ?  14 S3HRES      ?  36 S30COL
?  1F S3SHOT       80 SCREENS       81 SECONDS        F2B6 SETADDR
 F30B SINGLE     F19B SLOOP         A5 SMODE        ?  19 SND1AM
?  15 SND1MD    ?  17 SND1TN     ?  1A SND2AM      ?  16 SND2MD
?  18 SND2TN       86 SPREG         AA SPTS           F0FB SSETUP
 F000 START      F356 STWAIT     ?0282 SWTCHS        F257 SYN1
 F24B SYNCH        83 TARH          82 TARL           E9 TEMP1
?  EA TEMP2     ?0294 TIM1       ?0297 TIM1024       0296 TIM64
?0295 TIM8       F0A0 TINIT       F0A2 TLP            A9 TMCNT
   88 TOPCON     F13E TOPLP       F137 TOPWT          E6 TP1
   E7 TP25         E8 TP26          8D TREG1          93 TREG2
   99 TREG3        9F TREG4       F125 TWAIT      ?  03 VID03
?  27 VID27     ?  2D VID2D      ?  2E VID2E       ?  2F VID2F
?  3E VID3E     ?  3F VID3F        85 VLNCT        F424 VLP
   01 VRESET       00 VSYNCH      F302 WL1          F305 WL2
 F2AE WRITEOK    F2E9 WRITE       F30A WXT          F3DC YWAIT1
 F3FC YWAIT2
```

```
                00 VSYNCH       01 VRESET       02 LWAIT      ? 03 VID03
   04 P1VMOD    05 P2VMOD       06 P1COLOR      07 P2COLOR
   08 OCOLOR    09 BCOLOR       0A FVMODE     ? 0B P1FLIP
?  0C P2FLIP    0D FLDAIM       0E FLDBIM       0F FLDCIM
   10 P1HRES    11 P2HRES     ? 12 S1HRES     ? 13 S2HRES
?  14 S3HRES  ? 15 SND1MD     ? 16 SND2MD     ? 17 SND1TN
?  18 SND2TN  ? 19 SND1AM     ? 1A SND2AM       1B P1IMAG
   1C P2IMAG    1D P1SHOT       1E P2SHOT     ? 1F S3SHOT
   20 P1HORZ    21 P2HORZ       22 S1HORZ       23 S2HORZ
?  24 S3HORZ  ? 25 P1DELAY    ? 26 P2DELAY    ? 27 VID27
?  28 S1CONT  ? 29 S2CONT       2A HZSCRL     ? 2B NOINC
?  2C COLRES  ? 2D VID2D      ? 2E VID2E      ? 2F VID2F
?  30 PS1COL  ? 31 PS2COL     ? 32 P1OCOL     ? 33 P2OCOL
?  34 S1OCOL  ? 35 S2OCOL     ? 36 S3OCOL     ? 37 PPCOL
?  38 PDL5L   ? 39 PDL9L      ? 3A PDL5R      ? 3B PDL9R
?  3C LFTFR   ? 3D RGHFR      ? 3E VID3E      ? 3F VID3F
   80 SCREENS      81 SECONDS      82 TARL       83 TARH
   84 COMCOD       85 VLNCT        86 SPREG      87 DIGCON
   88 TOPCON       89 P1HS         8A P2HS       8B S1HS
   8C S2HS         8D TREG1        93 TREG2      99 TREG3
   9F TREG4        A5 SMODE        A6 MODE1      A7 MODE2
   A8 EBITS        A9 TMCNT        AA SPTS       AB P1CR
   AC P2CR         AD BKCR         AE OBJCR      AF P1VERT
   B0 P2VERT       B1 S1VERT       B2 S2VERT     B3 P1IMG
   BB P2IMG        C3 OBJAVT       C4 OBJAIM     CC OBJBVT
   CD OBJBIM       D5 OBJCVT       D6 OBJCIM     DE OBJXIM
   E6 TP1          E7 TP25         E8 TP26       E9 TEMP1
?  EA TEMP2     ?0280 CTROLS    ?0281 DDRA    ?0282 SWTCHS
?0283 DDRB       0284 RTIME      ?0294 TIM1    ?0295 TIM8
 0296 TIM64     ?0297 TIM1024     F000 START    F008 CLOOP
 F070 LETF       F076 LETR       F07C LETO     F082 LETB
 F088 LETA       F090 LETB8      F098 LETC     F0A0 TINIT
 F0A2 TLP        F0AE AIMLP      F0BA BIMLP    F0C6 CIMLP
 F0EC MAIN       F0FB SSETUP     F114 DISPLAY  F121 DOUBLE
 F125 TWAIT      F137 TOPWT      F13E TOPLP    F179 PART2
 F19B SLOOP      F1BB NOP1       F1BF INP1     F1C2 P1DSP
 F1C4 P2TST      F1D5 NOP2       F1D9 IMP2     F1DC P2DSP
 F1DE OATST      F1EF NOA        F1F3 INOA     F1F6 OADSP
 F1F8 OBTST      F209 NOB        F20D INOB     F210 OBDSP
 F212 OCTST      F223 NOC        F227 INOC     F22A OCDSP
 F22C ENDLP      F237 LPDONE     F248 SYNCH    F257 SYN1
 F275 FMON       F28B NOCOM      F28C L1       F298 L2
 F2A1 L3         F2A4 READOK   ?F2AD DOREAD    F2AE WRITEOK
 F2B6 SETADDR    F2C7 READ       F2E0 RL1      F2E3 RL2
 F2E8 RXT        F2E9 WRITE      F302 WL1      F305 WL2
 F30A WXT        F30B SINGLE     F30D CPLP     F336 DISPLAY2
 F342 HANGIN     F356 STWAIT     F361 LN60     F380 HFP1
 F38B IMLP1      F3B6 HFP2       F3C1 IMLP2    F3D8 DMLP1
 F3DC YWAIT1     F3F8 DMLP2      F3FC YWAIT2   F424 VLP
 F42F DL120      F441 HFLP3      F44E D48LP    F482 D48LP2
 F4D1 HFP3       F4DC IMLP3      F4EF IMLP4    F513 DWAIT
 F51E FROB1      F526 FROB2      F52E FROB3    F536 FROB4
 F53E FROB5      F546 FROB6    FFF0 AR1      FFF1 AR2
 FFF2 AR3
```

```
SOURCE FILE: XCONTROL
0000:           1 * EXPLORER CONTROL PROGRAM
0000:           2 * COPYRIGHT 1982 BY FROBCO -- ALL RIGHTS RESERVED
0000:           3 *
0000:           4 * VERSION  1.3 -- LAST MODIFIED 2/2/83
0000:           5 *
0000:           6 *
0000:           7 * SYSTEM DEFINITIONS
C000:           8 KBD     EQU  $C000     ; APPLE KEYBOARD REGISTER
C010:           9 KSTRB   EQU  $C010     ; APPLE KEYBOARD STROBE LOCATION
0010:          10 SACMD   EQU  $10       ; SET ADDRESS COMMAND FOR FROB
0020:          11 RDCMD   EQU  $20       ; READ REGISTER COMMAND
0040:          12 WRCMD   EQU  $40       ; WRITE REGISTER COMMAND
0000:          13 *
0000:          14 *
0000:          15 *
0000:          16 * PAGE 0 RAM ALLOCATION
0000:          17 *
0085:          18 RLOC    EQU  $85       ; READ REGISTER ADDRESS
0086:          19 FSTAT   EQU  RLOC+1    ; POINTER TO FROB STATUS REGISTER
0088:          20 FDATA   EQU  FSTAT+2   ; POINTER TO FROB DATA REGISTER
008A:          21 RBASE   EQU  FDATA+2   ; SCREEN ROW BASE POINTER FOR READ REGISTERS
008C:          22 CRBASE  EQU  RBASE+2   ; SCREEN ROW BASE POINTER USED BY CURSOR
008E:          23 FBASE   EQU  CRBASE+2  ; FIELD ARRAY ROW BASE POINTER
0090:          24 CROW    EQU  FBASE+2   ; CURSOR ROW
0091:          25 CCOL    EQU  CROW+1    ; CURSOR COLUMN
0092:          26 RRCOL   EQU  CCOL+1    ; READ REG COLUMN
0093:          27 OLDCR   EQU  RRCOL+1   ; OLD CURSOR ROW
0094:          28 OLDCCL  EQU  OLDCR+1   ; OLD CURSOR COLUMN
0095:          29 ROW     EQU  OLDCCL+1  ; PARAMETER TO ROW BASE LOOKUP ROUTINE
0096:          30 TMP1    EQU  ROW+1     ; RANDOM TEMP STORE
0097:          31 RDHGH   EQU  TMP1+1    ; READ ADDRESS HIGH POINTER
0098:          32 RDLOW   EQU  RDHGH+1   ; LOW PART OF ABOVE
0099:          33 LSTKEY  EQU  RDLOW+1   ; LAST KEY PRESSED
009A:          34 LSTRD   EQU  LSTKEY+1  ; LAST READ REGISTER VALUE
009B:          35 MASK    EQU  LSTRD+1   ; MASK TO USE ON THE READBACK LOCATIONS
0000:          36 *
0000:          37 *
0000:          38 * START OF PROGRAM -- LOAD AT $8000
0000:          39 *
----- NEXT OBJECT FILE NAME IS XCONTROL.OBJ0
8000:          40         ORG  $8000
8000:          41 *
8000:20 15 83  42 START   JSR  SAVER     ; THIS PROGRAM IS CALLED FROM BASIC
8003:20 B7 80  43 MLOOP   JSR  INCRR     ; GO INCREMENT THE READ REGISTER
8006:90 03     44         BCC  MLP1      ; CHECK FOR FULL SET DONE
8008:20 74 80  45         JSR  SPECR     ; IF SO GO DO SPECIAL READ
800B:20 E7 80  46 MLP1    JSR  RDREG     ; GO DO STANDARD READ
800E:85 9A     47         STA  LSTRD     ; SAVE RESULT
8010:20 FB 80  48         JSR  ADRCNV    ; DO AN ADDRESS CONVERSION SO WE KNOW WHERE THIS REGISTER
8013:          49 *        SHOULD GO ON THE SCREEN.
8013:A5 9A     50         LDA  LSTRD     ; GET VALUE BACK
8015:25 9B     51         AND  MASK      ; MASK WITH USER SELECTED MASK
8017:20 34 81  52         JSR  DSPBYT    ; PUT IT UP
801A:20 58 81  53         JSR  CHKKEY    ; CHECK FOR KEY PRESSED (GET IT IF SO)
801D:90 E4     54         BCC  MLOOP     ; END OF MAIN LOOP IF NO KEY
801F:C9 83     55         CMP  #$83      ; SEE IF CONTROL C
8021:D0 03     56         BNE  MLP2      ; IF SO EXIT TO BASIC
8023:4C 22 83  57         JMP  RESTORER
8026:85 99     58 MLP2    STA  LSTKEY    ; SAVE KEY
8028:C9 BB     59         CMP  #$BB      ; IS IT THE ";" (UP)?
802A:D0 06     60         BNE  NTUP      ; IF NOT GO ON
802C:20 67 81  61         JSR  MOVUP     ; ELSE MOVE CURSOR UP
```

```
802F:4C 03 80   62          JMP  MLOOP
8032:C9 AF      63 NTUP     CMP  #$AF      ; IS IT THE "/" (DOWN)?
8034:D0 06      64          BNE  NTDWN     ; IF NOT GO ON
8036:20 7C 81   65          JSR  MOVDWN    ; ELSE MOVE CURSOR DOWN
8039:4C 03 80   66          JMP  MLOOP
803C:C9 88      67 NTDWN    CMP  #$88      ; IS IT THE LEFT ARROW?
803E:D0 06      68          BNE  NTLFT     ; IF NOT GO ON
8040:20 A8 81   69          JSR  MOVLFT    ; ELSE MOVE CURSOR LEFT
8043:4C 03 80   70          JMP  MLOOP
8046:C9 95      71 NTLFT    CMP  #$95      ; IS IT THE RIGHT ARROW?
8048:D0 06      72          BNE  NTRGH     ; IF NOT GO ON
804A:20 90 81   73          JSR  MOVRGH    ; ELSE MOVE CURSOR RIGHT
804D:4C 03 80   74          JMP  MLOOP
8050:           75 *
8050:20 EF 81   76 NTRGH    JSR  NUMFLD    ; CHECK TO SEE IF CURSOR IN NUMBER FIELD
8053:90 AE      77          BCC  MLOOP     ; IF NOT THE KEY IS MEANINGLESS
8055:A5 99      78          LDA  LSTKEY    ; GET KEY BACK
8057:20 1F 81   79          JSR  HEXKEY    ; IS IT A HEXIDECIMAL DIGIT?
805A:90 A7      80          BCC  MLOOP     ; IF NOT THE KEY IS MEANINGLESS
805C:A5 99      81          LDA  LSTKEY    ; GET KEY BACK AGAIN
805E:20 2C 82   82          JSR  DSPCHR    ; GO PUT IT AT CURSOR AND ADVANCE CURSOR
8061:20 90 81   83          JSR  MOVRGH
8064:20 EF 81   84          JSR  NUMFLD    ; SEE IF STILL IN NUMBER FIELD
8067:B0 9A      85          BCS  MLOOP     ; IF SO WE ARE DONE FOR NOW
8069:20 04 82   86          JSR  SPECCK    ; ELSE TIME TO DO SOMETHING SO CHECK FOR SPECIAL CASE
806C:B0 95      87          BCS  MLOOP     ; IF TRUE THEN THE SPECIAL CASE HAS BEEN DONE BEFORE RETURN
806E:20 3A 82   88          JSR  DOWRT     ; GO DO THE VCS REGISTER WRITE
8071:4C 03 80   89          JMP  MLOOP     ; END OF TOP LEVEL
8074:           90 *
8074:           91 *
8074:           92 * ROUTINE TO DO A SPECIAL READ
8074:           93 *
8074:A5 93      94 SPECR    LDA  OLDCR     ; WE WILL NEED TO USE
8076:48         95          PHA             ; OLDCR AND OLDCCL AS PARAMETERS
8077:A5 94      96          LDA  OLDCCL    ; TO GETVAL
8079:48         97          PHA
807A:A9 12      98          LDA  #$12      ; ROW OF SPECIAL OPERATIONS
807C:85 93      99          STA  OLDCR
807E:A9 13     100          LDA  #$13      ; COLUMN OF SECOND DIGIT OF THE HIGH ORDER ADDRESS
8080:85 94     101          STA  OLDCCL
8082:20 5E 82  102          JSR  GETVAL    ; GET HIGH ADDRESS PART
8085:85 97     103          STA  RDHGH     ; PUT IN HIGH ADDR POINTER
8087:A9 15     104          LDA  #$15      ; COLUMN OF SECOND DIGIT OF THE LOW ORDER ADDRESS
8089:85 94     105          STA  OLDCCL    ; PASS ON TO GETVAL
808B:20 5E 82  106          JSR  GETVAL
808E:85 98     107          STA  RDLOW     ; NOW THE ADDRESS IS SET UP
8090:68        108          PLA             ; GET OLDCCL BACK
8091:85 94     109          STA  OLDCCL
8093:68        110          PLA             ; GET OLDCR BACK
8094:85 93     111          STA  OLDCR
8096:20 E7 80  112          JSR  RDREG     ; GO DO THE VCS READ
8099:48        113          PHA             ; SAVE THE RESULT
809A:A9 24     114          LDA  #$24      ; COLUMN OF THE RESULT
809C:85 92     115          STA  RRCOL
809E:A2 12     116          LDX  #$12      ; ROW OF THE RESULT
80A0:BD 2F 83  117          LDA  CHLIST,X  ; GET HIGH PART OF SCREE ROW
80A3:85 8B     118          STA  RBASE+1   ; PUT IN HIGH PART
80A5:BD 47 83  119          LDA  CLLIST,X  ; GET LOW PART
80A8:85 8A     120          STA  RBASE
80AA:68        121          PLA             ; GET VALUE TO BE DISPLAYED
80AB:20 34 81  122          JSR  DSPBYT    PUT IT UP
80AE:A9 00     123          LDA  #0        ; CLEAR RDHGH
80B0:85 97     124          STA  RDHGH
80B2:A5 85     125          LDA  RLOC      ; RESTORE RDLOW
80B4:85 98     126          STA  RDLOW
80B6:60        127          RTS
```

```
80B7:          128 *
80B7:          129 *
80B7:          130 * ROUTINE TO INCREMENT READ REGISTER NUMBER AND RETURN CARRY SET
80B7:          131 * WHEN THROUGH THE ENTIRE LIST
80B7:          132 *
80B7:E6 85     133 INCRR   INC  RLOC
80B9:A5 85     134         LDA  RLOC      ; GET THE NEW VALUE
80BB:C9 40     135         CMP  #$40      ; GONE TOO FAR?
80BD:90 04     136         BCC  INCR1     ; BRANCH IF NOT AT END
80BF:A9 30     137         LDA  #$30      ; ELSE RESET COUNTER
80C1:85 85     138         STA  RLOC
80C3:A9 00     139 INCR1   LDA  #0        ; UPDATE THE READ ADDRESS POINTERS
80C5:85 97     140         STA  RDHGH     ; HIGH PART
80C7:A5 85     141         LDA  RLOC
80C9:85 98     142         STA  RDLOW     ; LOW PART
80CB:60        143         RTS                 ; DONE
80CC:          144 *
80CC:          145 *
80CC:          146 * ROUTINE TO DO A SET READ/WRITE ADDRESS OVER IN THE VCS
80CC:          147 *
80CC:A0 00     148 SETADR  LDY  #0        ; SO WE CAN LOAD INDIRECT
80CE:B1 86     149 SA1     LDA  (FSTAT),Y ; GO GET FROB STATUS REGISTER
80D0:10 FC     150         BPL  SA1       ; WAIT FOR OK TO WRITE
80D2:A9 10     151         LDA  #SACMD    ; GET THE SET ADDRESS COMMAND CODE
80D4:91 88     152         STA  (FDATA),Y ; SEND IT TO THE FROB EXPLORER
80D6:B1 86     153 SA2     LDA  (FSTAT),Y ; GO GET FROB STATUS REGISTER
80D8:10 FC     154         BPL  SA2       ; WAIT FOR OK FOR NEXT WRITE
80DA:A5 97     155         LDA  RDHGH     ; GET THE HIGH ORDER PART
80DC:91 88     156         STA  (FDATA),Y ; SEND IT
80DE:B1 86     157 SA3     LDA  (FSTAT),Y ; SAME AS ABOVE
80E0:10 FC     158         BPL  SA3
80E2:A5 98     159         LDA  RDLOW     ; GET THE LOW ORDER PART
80E4:91 88     160         STA  (FDATA),Y ; THAT FINISHES THE JOB.
80E6:60        161         RTS
80E7:          162 *
80E7:          163 *
80E7:          164 * ROUTINE TO READ A VCS REGISTER
80E7:          165 *
80E7:20 CC 80  166 RDREG   JSR  SETADR    ; SEND THE VCS THE ADDRESS OF THE REGISTER WE WANT
80EA:B1 86     167 RDR0    LDA  (FSTAT),Y ; GET FROB STATUS
80EC:10 FC     168         BPL  RDR0      ; WAIT TIL OK TO WRITE
80EE:A9 20     169         LDA  #RDCMD    ; SEND IT A READ REGISTER COMMAND
80F0:91 88     170         STA  (FDATA),Y
80F2:B1 86     171 RDR1    LDA  (FSTAT),Y ; GET FROB STATUS REGISTER
80F4:29 40     172         AND  #$40      ; LOOK AT DATA COMING BACK BIT
80F6:F0 FA     173         BEQ  RDR1      ; WAIT FOR DATA TO COME BACK
80F8:B1 88     174         LDA  (FDATA),Y ; NOW READ DATA
80FA:60        175         RTS
80FB:          176 *
80FB:          177 *
80FB:          178 * ROUTINE TO CONVERT READ REGISTER NUMBER INTO SCREEN ADDRESS
80FB:          179 *
80FB:A5 85     180 ADRCNV  LDA  RLOC      ; GET READ REGISTER NUMBER
80FD:29 0F     181         AND  #$0F      ; LOOK AT LOWER BITS
80FF:4A        182         LSR  A          ; DIVIDE BY 4
8100:4A        183         LSR  A
8101:09 14     184         ORA  #$14      ; PUT IN ROW OFFSET
8103:AA        185         TAX            ; SAVE IN X
8104:BD 2F 83  186         LDA  CHLIST,X  ; LOOKUP THE ROW BASE ADDRESS HIGH
8107:85 8B     187         STA  RBASE+1   ; SAVE IT IN POINTER
8109:BD 47 83  188         LDA  CLLIST,X  ; LOOKUP THE ROW BASE ADRESS LOW
810C:85 8A     189         STA  RBASE     ; SAVE IT IN POINTER
810E:A5 85     190         LDA  RLOC      ; GET REGISTER NUMBER AGAIN
8110:29 03     191         AND  #$3       ; LOOK AT LAST TWO BITS
8112:85 96     192         STA  TMP1      ; MULTIPLY BY 9
8114:0A        193         ASL  A
```

```
8115:0A       194         ASL  A
8116:0A       195         ASL  A
8117:05 96    196         ORA  TMP1
8119:18       197         CLC              ; NOW ADD COLUMN OFFSET
811A:69 06    198         ADC  #6
811C:85 92    199         STA  RRCOL
811E:60       200         RTS
811F:         201 *
811F:         202 *
811F:         203 * ROUTINE TO CHECK KEY NUMBER FOR LEGAL HEX DIGIT
811F:         204 *
811F:C9 BA    205 HEXKEY  CMP  #$BA      ; SEE IF IT IS IN ALPHA RANGE
8121:B0 05    206         BCS  HK1
8123:C9 B0    207         CMP  #$B0
8125:90 0B    208         BCC  NOHEX     ; ALSO FAIL IF TOO LOW
8127:60       209         RTS              ; ELSE OK
8128:         210 *
8128:C9 C1    211 HK1     CMP  #$C1      ; SEE IF OUT OF BA>C0 RANGE
812A:90 06    212         BCC  NOHEX
812C:C9 C7    213         CMP  #$C7      ; SEE IF OFF THE TOP
812E:B0 02    214         BCS  NOHEX
8130:38       215         SEC              ; ELSE IT IS OK
8131:60       216         RTS
8132:         217 *
8132:18       218 NOHEX   CLC            ; RETURN CARRY CLEAR ON FAIL
8133:60       219         RTS
8134:         220 *
8134:         221 *
8134:         222 * ROUTINE TO PUT UP HEX BYTE AT RBASE AND RRCOL
8134:         223 *
8134:85 96    224 DSPBYT  STA  TMP1      ; SAVE BYTE TO BE DISPALYED
8136:4A       225         LSR  A          ; GET UPPER 4 BITS
8137:4A       226         LSR  A
8138:4A       227         LSR  A
8139:4A       228         LSR  A
813A:20 4C 81 229         JSR  TOHEX     ; CONVERT NIBBLE TO ALPHANUMERIC FOR THE HEX DIGIT
813D:A4 92    230         LDY  RRCOL     ; GET THE COLUMN ADDRESS
813F:91 8A    231         STA  (RBASE),Y ; PUT THE CHAR ON THE SCREEN
8141:A5 96    232         LDA  TMP1      ; GET BYTE BACK
8143:20 4C 81 233         JSR  TOHEX     ; CONVERT LOWER NIBBLE
8146:A4 92    234         LDY  RRCOL     ; GET THE COLUMN ADDRESS
8148:C8       235         INY              ; MOVE TO NEXT COLUMN
8149:91 8A    236         STA  (RBASE),Y ; PUT UP CHAR
814B:60       237         RTS
814C:         238 *
814C:         239 *
814C:         240 * ROUTINE TO RETURN HEX ALPHA CHAR FOR LOW NIBBLE IN A
814C:         241 *
814C:29 0F    242 TOHEX   AND  #$0F      ; GET LOW NIBBLE
814E:C9 0A    243         CMP  #$0A      ; SEE IF IN THE ALPHA GROUP
8150:90 02    244         BCC  TH1       ; BRANCH IF NOT ALPHA
8152:69 06    245         ADC  #6        ; IF SO ADD 7 (CARRY + 6)
8154:18       246 TH1     CLC            ; PUT IN UPPER BITS
8155:69 B0    247         ADC  #$B0
8157:60       248         RTS
8158:         249 *
8158:         250 * ROUTINE TO SEE IF KEY PRESSED (RETURNS VALUE AND CARRY SET IF SO)
8158:         251 *
8158:2C 00 C0 252 CHKKEY  BIT  KBD       ; LOOK AT KBD STROBE BIT
815B:30 02    253         BMI  CKY1      ; BRANCH IF KEY PRESSED
815D:18       254         CLC              ; ELSE CLEAR CARRY AND EXIT
815E:60       255         RTS
815F:         256 *
815F:AD 00 C0 257 CKY1    LDA  KBD       ; GET KEY CODE
8162:2C 10 C0 258         BIT  KSTRB     ; RESET THE STROBE BIT
8165:38       259         SEC              ; SHOW WE GOT IT
```

```
8166:60        260         RTS
8167:          261 *
8167:          262 *
8167:          263 * ROUTINE TO MOVE THE CURSOR UP
8167:          264 *
8167:A5 90     265 MOVUP   LDA  CROW      ; GET THE CURSOR ROW POSITION
8169:85 93     266         STA  OLDCR     ; THIS IS NOW THE OLD POSITION
816B:C6 90     267         DEC  CROW      ; MOVE NEW POSITION UP
816D:A5 90     268         LDA  CROW      ; CHECK FOR WRAP OFF TOP
816F:10 04     269         BPL  MUP1      ; BRANCH IF OK
8171:A9 17     270         LDA  #$17      ; ELSE MOVE TO BOTTOM ROW
8173:85 90     271         STA  CROW
8175:A5 91     272 MUP1    LDA  CCOL      ; UPDATE OLD COLUMN
8177:85 94     273         STA  OLDCCL
8179:4C BA 81  274         JMP  CUPDTE    ; THEN GO UPDATE THE CURSOR
817C:          275 *
817C:          276 *
817C:          277 * ROUTINE TO MOVE CURSOR DOWN
817C:          278 *
817C:A5 90     279 MOVDWN  LDA  CROW      ; GET THE CURSOR ROW
817E:85 93     280         STA  OLDCR     ; UPDATE THE OLD POSITION
8180:C9 17     281         CMP  #$17      ; AT THE BOTTOM?
8182:F0 05     282         BEQ  MDN1      ; IF SO WRAP TO TOP
8184:E6 90     283         INC  CROW      ; ELSE MOVE DOWN
8186:4C 75 81  284         JMP  MUP1      ; AND HANDLE AS ABOVE
8189:          285 *
8189:A9 00     286 MDN1    LDA  #0        ; WRAP TO TOP OF SCREEN
818B:85 90     287         STA  CROW
818D:4C 75 81  288         JMP  MUP1
8190:          289 *
8190:          290 *
8190:          291 * ROUTINE TO MOVE THE CURSOR TO THE RIGHT
8190:          292 *
8190:A5 91     293 MOVRGH  LDA  CCOL      ; GET THE CURSOR COLUMN
8192:85 94     294         STA  OLDCCL    ; UPDATE THE OLD POSITION
8194:C9 27     295         CMP  #39       ; SEE IF AT RIGHT EDGE
8196:D0 07     296         BNE  MRT0      ; BRANCH IF SIMPLE MOVE
8198:A9 00     297         LDA  #0        ; ELSE RESET COLUMN
819A:85 91     298         STA  CCOL
819C:4C A1 81  299         JMP  MRT1
819F:          300 *
819F:E6 91     301 MRT0    INC  CCOL      ; ELSE MOVE TO THE RIGHT
81A1:A5 90     302 MRT1    LDA  CROW      ; GET THE ROW
81A3:85 93     303         STA  OLDCR      ; UPDATE OLD POSITION
81A5:4C BA 81  304         JMP  CUPDTE    ; GO UPDATE THE CURSOR
81A8:          305 *
81A8:          306 *
81A8:          307 * ROUTINE TO MOVE THE CURSOR LEFT
81A8:          308 *
81A8:A5 91     309 MOVLFT  LDA  CCOL      ; GET THE COLUMN
81AA:85 94     310         STA  OLDCCL    ; UPDATE OLD POSITION
81AC:D0 07     311         BNE  MLF1      ; BRANCH IF ROOM FOR SIMPLE MOVE
81AE:A9 27     312         LDA  #39       ; ELSE MOVE TO LAST COLUMN
81B0:85 91     313         STA  CCOL
81B2:4C A1 81  314         JMP  MRT1
81B5:          315 *
81B5:C6 91     316 MLF1    DEC  CCOL
81B7:4C A1 81  317         JMP  MRT1      ; TAKE CARE OF AS ABOVE
81BA:          318 *
81BA:          319 *
81BA:          320 * ROUTINE TO UPDATE CURSOR POSITION
81BA:          321 *
81BA:A4 94     322 CUPDTE  LDY  OLDCCL    ; GET OLD CURSOR COLUMN POSITION
81BC:A5 93     323         LDA  OLDCR     ; GET OLD CURSOR ROW
81BE:85 95     324         STA  ROW       ; SAVE AS PARAMETER TO CADDR
81C0:20 E2 81  325         JSR  CRADDR    ; LOOKUP CURSOR ROW BASE ADDRESS GIVEN "ROW"
```

```
81C3:B1 8C     326         LDA  (CRBASE),Y ; GET WHAT IS AT THE OLD CURSOR POSITION
81C5:09 80     327         ORA  #$80       ; TAKE OUT OF INVERSE VIDEO
81C7:85 96     328         STA  TMP1       ; HERE IS THE MAPPING
81C9:29 20     329         AND  #$20       ; ALGORITHM
81CB:0A        330         ASL  A
81CC:49 40     331         EOR  #$40
81CE:05 96     332         ORA  TMP1
81D0:91 8C     333         STA  (CRBASE),Y ; PUT IT BACK
81D2:A4 91     334         LDY  CCOL       ; GET THE NEW CURSOR COLUMN
81D4:A5 90     335         LDA  CROW       ; GET THE NEW CURSOR ROW
81D6:85 95     336         STA  ROW        ; SAVE AS PARAMETER TO CRADDR
81D8:20 E2 81  337         JSR  CRADDR     ; LOOKUP NEW CURSOR ROW BASE ADDRESS GIVEN "ROW"
81DB:B1 8C     338         LDA  (CRBASE),Y ; GET NEW CURSOR CHARACTOR
81DD:29 3F     339         AND  #$3F       ; MAKE INVERSE VIDEO
81DF:91 8C     340         STA  (CRBASE),Y ; PUT IT BACK
81E1:60        341         RTS
81E2:          342 *
81E2:          343 *
81E2:          344 * ROUTINE TO LOOKUP CURSOR ROW BASE ADDRESS FROM INDEX "ROW"
81E2:          345 *
81E2:A6 95     346 CRADDR  LDX  ROW        ; GET INDEX
81E4:BD 2F 83  347         LDA  CHLIST,X   ; LOOKUP HIGH PART
81E7:85 8D     348         STA  CRBASE+1   ; SAVE IN HIGH POINTER
81E9:BD 47 83  349         LDA  CLLIST,X   ; LOOKUP LOW PART
81EC:85 8C     350         STA  CRBASE     ; SAVE IN LOW POINTER
81EE:60        351         RTS
81EF:          352 *
81EF:          353 *
81EF:          354 * ROUTINE TO SEE IF CURSOR IS IN A NUMERITC FIELD
81EF:          355 *
81EF:A6 90     356 NUMFLD  LDX  CROW       ; LOOKUP ROW BASE ADDRESS
81F1:BD DF 8A  357         LDA  FHLIST,X   ; HIGH PART OF FIELD LIST
81F4:85 8F     358         STA  FBASE+1    ; SAVE IN HIGH POINTER
81F6:BD F7 8A  359         LDA  FLLIST,X   ; LOOKUP LOW PART
81F9:85 8E     360         STA  FBASE      ; SAVE IN LOW POINTER
81FB:A4 91     361         LDY  CCOL       ; GET THE COLUMN NUMBER
81FD:B1 8E     362         LDA  (FBASE),Y  ; GET THE FIELD FLAG
81FF:4C 1F 81  363         JMP  HEXKEY     ; GO SEE IF LEGAL HEX DIGIT
8202:          364 *
8202:18        365 NONUM   CLC             ; CLEAR CARRY TO RETURN "FALSE"
8203:60        366         RTS
8204:          367 *
8204:          368 *
8204:          369 * ROUTINE TO HANDLE SPECIAL CASE
8204:A5 93     370 SPECCK  LDA  OLDCR      ; SEE IF IN SPECIAL ROW
8206:C9 12     371         CMP  #$12
8208:D0 15     372         BNE  MTEST      ; CHECK FOR MASK UPDATE
820A:A5 94     373         LDA  OLDCCL     ; ELSE CHECK COLUMN FOR LAST ADDRESS DIGIT
820C:C9 15     374         CMP  #$15
820E:F0 0D     375         BEQ  SPDONE     ; IF SO WE HAVE DONE THE ADDRESS
8210:C9 1C     376         CMP  #$1C       ; THIS IS THE WRITE OPERATION COLUMN
8212:D0 16     377         BNE  NOSPEC     ; IF NOT THIS THEN NOT SPECIAL
8214:20 74 80  378         JSR  SPECR      ; DO A SPECIAL READ TO SET UP THE ADDRESS
8217:20 5E 82  379         JSR  GETVAL     ; GET THE VALUE TO WRITE
821A:20 8D 82  380         JSR  WTREG      ; DO THE WRITE
821D:38        381 SPDONE  SEC             ; DONE SO SET CARRY AND RETURN
821E:60        382         RTS
821F:          383 *
821F:C9 13     384 MTEST   CMP  #$13       ; SEE IF AT READ MASK ROW
8221:D0 07     385         BNE  NOSPEC     ; IF NOT THEN NOT SPECIAL
8223:20 5E 82  386         JSR  GETVAL     ; ELSE GET VALUE OF NEW MASK
8226:85 9B     387         STA  MASK       ; SAVE IT
8228:38        388         SEC             ; RETURN MARKING SPECIAL
8229:60        389         RTS
822A:          390 *
822A:18        391 NOSPEC  CLC             ; IF NOT SPECIAL THEN CLEAR CARRY AND RETURN
```

```
822B:60        392         RTS
822C:          393 *
822C:          394 * ROUTINE TO PUT CHARACTOR AT CURSOR
822C:          395 *
822C:A5 90     396 DSPCHR  LDA  CROW      ; GET THE CURSOR ROW
822E:85 95     397         STA  ROW       ; SETUP FOR ROW ADDRESS COMPUTATION
8230:20 E2 81  398         JSR  CRADDR    ; DO LOOKUP
8233:A4 91     399         LDY  CCOL      ; GET THE COLUMN
8235:A5 99     400         LDA  LSTKEY    ; GET THE CHARACTOR
8237:91 8C     401         STA  (CRBASE),Y ; PUT IT ON THE SCREEN
8239:60        402         RTS
823A:          403 *
823A:          404 *
823A:          405 * ROUTINE TO DO THE VCS REG WRITE
823A:          406 *
823A:20 4A 82  407 DOWRT   JSR  GETFLD    ; GET THE FIELD BYTE FOR THE OLD CURSOR
823D:20 57 82  408         JSR  SETREG    ; TURN THAT INTO A VCS REGISTER ADDRESS
8240:20 CC 80  409         JSR  SETADR    ; SET IT UP IN THE VCS
8243:20 5E 82  410         JSR  GETVAL    ; GO GET THE CURRENT VALUE OF THE OLD CURSOR FIELD
8246:20 8D 82  411         JSR  WTREG     ; GO SEND IT TO THE VCS
8249:60        412         RTS              ; DONE
824A:          413 *
824A:          414 *
824A:          415 * ROUTINE TO GET FIELD BYTE
824A:          416 *
824A:A6 93     417 GETFLD  LDX  OLDCR     ; GET THE OLD CURSOR ROW
824C:BD DF 8A  418         LDA  FHLIST,X  ; GET THE HIGH PART OF THE FIELD ROW BASE ADDRESS
824F:85 8D     419         STA  CRBASE+1  ; USE THIS POINTER SO WE CAN DROP IN BELOW
8251:BD F7 8A  420         LDA  FLLIST,X  ; NOW THE LOW PART
8254:4C 68 82  421         JMP  GVAL1       ; DO THE REST BELOW
8257:          422 *
8257:          423 *
8257:          424 * ROUTINE TO GO FROM FIELD BYTE TO VCS ADDRESS
8257:          425 *
8257:85 98     426 SETREG  STA  RDLOW     ; SAVE IN LOW ORDER REGISTER
8259:A9 00     427         LDA  #0        ; CLEAR OUT HIGH PART
825B:85 97     428         STA  RDHGH
825D:60        429         RTS
825E:          430 *
825E:          431 *
825E:          432 * ROUTINE TO GET VALUE OF THE FIELD AT OLD CURSOR
825E:          433 *
825E:A6 93     434 GETVAL  LDX  OLDCR     ; GET THE OLD CURSOR ROW
8260:BD 2F 83  435         LDA  CHLIST,X  ; GET HIGH PART OF ROW BASE ADDRESS
8263:85 8D     436         STA  CRBASE+1, ; SAVE IN HIGH PART OF POINTER
8265:BD 47 83  437         LDA  CLLIST,X  ; GET LOW PART OF ROW BASE ADDRESS
8268:85 8C     438 GVAL1   STA  CRBASE    ; SAVE IT
826A:A4 94     439         LDY  OLDCCL    ; GET THE OLD CURSOR COLUMN POSITION
826C:B1 8C     440         LDA  (CRBASE),Y ; THIS GETS THE LOW DIGIT OF THE VALUE
826E:48        441         PHA              ; SAVE IT
826F:88        442         DEY              ; BACK UP TO THE HIGH DIGIT
8270:B1 8C     443         LDA  (CRBASE),Y ; GET THAT
8272:20 82 82  444         JSR  TOBIN     ; CONVERT FROM HEX DIGIT TO BINARY
8275:0A        445         ASL  A         ; MAKE HIGH PART
8276:0A        446         ASL  A
8277:0A        447         ASL  A
8278:0A        448         ASL  A
8279:85 96     449         STA  TMP1      ; SAVE FOR LATER
827B:68        450         PLA            ; GET LOW ORDER HEX DIGIT BACK
827C:20 82 82  451         JSR  TOBIN     ; CONVERT TO BINARY
827F:05 96     452         ORA  TMP1      ; THIS PUTS IT TOGETHER
8281:60        453         RTS
8282:          454 *
8282:          455 *
8282:          456 * ROUTINE TO CONVERT ASCII FOR HEX DIGIT BACK TO BINARY
8282:          457 *
```

```
8282:29 7F     458 TOBIN   AND  #$7F      ; GET RID OF TOP BIT
8284:C9 3A     459         CMP  #$3A      ; SEE IF IN NUMERIC RANGE
8286:90 02     460         BCC  TOB1      ; BRANCH IF SO
8288:E9 07     461         SBC  #7        ; ELSE SUBTRACT 7 TO CORRECT ALPHA RANGE
828A:29 0F     462 TOB1    AND  #$0F      ; NOW WE JUST NEED THE LAST 4 BITS
828C:60        463         RTS
828D:          464 *
828D:          465 *
828D:          466 * ROUTINE TO DO A VCS REGISTER WRITE
828D:          467 *
828D:48        468 WTREG   PHA            ; SAVE VALUE
828E:A0 00     469         LDY  #0        ; SET UP TO INDIRECT
8290:B1 86     470 WTR1    LDA  (FSTAT),Y ; LOOK AT FROB STATUS REGISTER
8292:10 FC     471         BPL  WTR1      ; WAIT TIL OK TO WRITE
8294:A9 40     472         LDA  #WRCMD    ; SEND IT A FROB EXPLORER WRITE COMMAND
8296:91 88     473         STA  (FDATA),Y
8298:B1 86     474 WTR2    LDA  (FSTAT),Y ; WAIT FOR IT TO GET IT
829A:10 FC     475         BPL  WTR2
829C:68        476         PLA            ; GET VALUE BACK
829D:91 88     477         STA  (FDATA),Y SEND IT TO VCS
829F:60        478         RTS
82A0:          479 *
82A0:          480 *
82A0:          481 * THE INITILAIZATION ROUTINE
82A0:          482 *
82A0:20 E7 82  483 INIT    JSR  SLOAD     ; PUT UP THE SCREEN
82A3:A9 C0     484         LDA  #$C0      ; SET UP FSTAT AND FDATA
82A5:85 87     485         STA  FSTAT+1
82A7:85 89     486         STA  FDATA+1
82A9:A9 A0     487 PARAM1  LDA  #$A0
82AB:85 86     488         STA  FSTAT
82AD:A9 A1     489 PARAM2  LDA  #$A1
82AF:85 88     490         STA  FDATA
82B1:A9 00     491         LDA  #0        ; CLEAR SOME VARIABLES
82B3:85 94     492         STA  OLDCCL
82B5:85 93     493         STA  OLDCR
82B7:85 91     494         STA  CCOL
82B9:85 90     495         STA  CROW
82BB:85 99     496         STA  LSTKEY
82BD:A9 81     497         LDA  #$81      ; INIT THE MASK
82BF:85 9B     498         STA  MASK
82C1:          499 *
82C1:          500 * NOW A LOOP TO DO THE INITIAL REGISTER WRITES
82C1:          501 *
82C1:20 EF 81  502 ILP     JSR  NUMFLD    ; IS CURSOR AT A NUMBER FIELD?
82C4:B0 1A     503              BCS       INFLD ; IF YES, JUST PUT THE FLAG IN LSTKEY AND MOVE ON
82C6:24 99     504         BIT  LSTKEY    ; IF NOT, SEE IF WE JUST WERE
82C8:F0 07     505         BEQ  NOCHNG    ; IF NOT THEN THE CHANGE IS NOT IMPORTANT
82CA:A9 00     506         LDA  #0        ; CLEAR LSTKEY FLAG
82CC:85 99     507         STA  LSTKEY
82CE:20 3A 82  508         JSR  DOWRT     ; UPDATE THE REGISTER
82D1:          509 *
82D1:20 90 81  510 NOCHNG  JSR  MOVRGH    ; MOVE TO THE NEXT POSITION
82D4:A5 91     511         LDA  CCOL      ; GET NEW CURSOR COLUMN
82D6:D0 E9     512         BNE  ILP       ; LOOP IF NOT READY FOR NEW LINE
82D8:20 7C 81  513         JSR  MOVDWN    ; ELSE MOVE DOWN TO NEXT LINE
82DB:A5 90     514         LDA  CROW      ; SEE IF AT THE END OF THE DISPLAY
82DD:D0 E2     515         BNE  ILP       ; IF NOT KEEP GOING
82DF:60        516         RTS              ; BUT IF SO THEN WE ARE DONE
82E0:          517 *
82E0:A9 FF     518 INFLD   LDA  #$FF      ; SET LSTKEY TO INDICATE WE ARE IN A FIELD
82E2:85 99     519         STA  LSTKEY
82E4:4C D1 82  520         JMP  NOCHNG    ; GET BACK IN
82E7:          521 *
82E7:          522 *
82E7:          523 *
```

```
82E7:          524 * ROUTINE TO LOAD DISPLAY FROM ARRAY "SCREEN"
82E7:          525 *
82E7:A2 17     526 SLOAD   LDX  #$17      ; DO FOR 24 LINES
82E9:A9 86     527         LDA  #<SCREEN+920 ; SCREEN+(23*40) POINTS TO THE LAST LINE
82EB:85 8B     528         STA  RBASE+1   ; STORE THE HIGH PART
82ED:A9 F7     529         LDA  #>SCREEN+920 ; NOW THE LOW PART
82EF:85 8A     530         STA  RBASE
82F1:BD 2F 83  531 SL1     LDA  CHLIST,X  ; GET HIGH POINTER TO DISPLAY ROW
82F4:85 8D     532         STA  CRBASE+1  ; PUT IN POINTER
82F6:BD 47 83  533         LDA  CLLIST,X  ; GET LOW PART
82F9:85 8C     534         STA  CRBASE    ; PUT THAT IN POINTER
82FB:A0 27     535         LDY  #39       ; WORK BACK FROM END OF LINE
82FD:B1 8A     536 SL2     LDA  (RBASE),Y ; GET CHARACTOR FROM ARRAY
82FF:91 8C     537         STA  (CRBASE),Y ; PUT UP ON SCREEN
8301:88        538         DEY
8302:10 F9     539         BPL  SL2       ; DO FOR Y=39 TO 0
8304:38        540         SEC            ; SUBTRACT 40 FROM RBASE
8305:A5 8A     541         LDA  RBASE
8307:E9 28     542         SBC  #40
8309:85 8A     543         STA  RBASE
830B:A5 8B     544         LDA  RBASE+1   ; 16 BIT
830D:E9 00     545         SBC  #0
830F:85 8B     546         STA  RBASE+1
8311:CA        547         DEX              ; THIS IS THE MAJOR LOOP COUNT
8312:10 DD     548         BPL  SL1       ; WORK THROUGH 24 LINES
8314:60        549         RTS
8315:          550 *
8315:          551 *
8315:          552 SAVER   EQU  *         ; ROUTINE TO SAVE PAGE 0 LOCATIONS FOR BASIC
8315:A2 80     553         LDX  #$80      ; START AT LOC 80 HEX
8317:B5 00     554 SAVLP   LDA  0,X       ; GET BYTE
8319:9D 8F 8A  555         STA  SAVBUF-$80,X ; SAVE IT
831C:E8        556         INX            ; MOVE TO NEXT
831D:30 F8     557         BMI  SAVLP     ; KEEP GOING
831F:4C A0 82  558         JMP  INIT      ; THE REST START IN INIT
8322:          559 *
8322:          560 *
8322:          561 RESTORER EQU *         ; ROUTINE TO PUT BACK THE BASIC LOCATIONS TO PAGE 0
8322:A2 80     562         LDX  #$80      ; START AT LOC 80 HEX
8324:BD 8F 8A  563 RESLP   LDA  SAVBUF-$80,X ; GET BYTE
8327:95 00     564         STA  0,X       ; RESTORE IT
8329:E8        565         INX            ; MOVE TO NEXT
832A:30 F8     566         BMI  RESLP     ; DO FOR X=$80 TO $FF
832C:4C D0 03  567         JMP  $3D0      ; RETURN TO BASIC
832F:          568 *
832F:          569 *
832F:          570 *
832F:          571 * SCREEN ROW BASE LOOKUP TABLES
832F:          572 *
832F:04 04 05  573 CHLIST  DFB  4,4,5,5,6,6,7,7,4,4,5,5,6,6,7,7,4,4,5,5,6,6,7,7
8332:05 06 06
8335:07 07 04
8338:04 05 05
833B:06 06 07
833E:07 04 04
8341:05 05 06
8344:06 07 07
8347:00 80 00  574 CLLIST  DFB  0,$80,0,$80,0,$80,0,$80
834A:80 00 80
834D:00 80
834F:28 A8 28  575         DFB  $28,$A8,$28,$A8,$28,$A8,$28,$A8
8352:A8 28 A8
8355:28 A8
8357:50 D0 50  576         DFB  $50,$D0,$50,$D0,$50,$D0,$50,$D0
835A:D0 50 D0
835D:50 D0
```

```
835F:          577 *
835F:          578 *
835F:          579 * HERE IS THE PAGE SETUP
835F:          580 *
835F:          581         MSB  ON        ; DO STRINGS WITH MSB ON
835F:          582 *
835F:A0 A0 A0  583 SCREEN  ASC  /             FROB EXPLORER CONTROL SCREEN      /
8362:A0 A0 A0
8365:C6 D2 CF
8368:C2 A0 C5
836B:D8 D0 CC
836E:CF D2 C5
8371:D2 A0 C3
8374:CF CE D4
8377:D2 CF CC
837A:A0 D3 C3
837D:D2 C5 C5
8380:CE A0 A0
8383:A0 A0 A0
8386:A0
8387:C5 CE C1  584         ASC  /ENABLE:0C  (PUT 40 HERE FOR SCREEN2)    /
838A:C2 CC C5
838D:BA B0 C3
8390:A0 A0 A8
8393:D0 D5 D4
8396:A0 B4 B0
8399:A0 C8 C5
839C:D2 C5 A0
839F:C6 CF D2
83A2:A0 D3 C3
83A5:D2 C5 C5
83A8:CE B2 A9
83AB:A0 A0 A0
83AE:A0
83AF:D0 CC C1  585         ASC  /PLAYER1:COLOR 82 VERT 14 HORZ 30 FLP 00 /
83B2:D9 C5 D2
83B5:B1 BA C3
83B8:CF CC CF
83BB:D2 A0 B8
83BE:B2 A0 D6
83C1:C5 D2 D4
83C4:A0 B1 B4
83C7:A0 C8 CF
83CA:D2 DA A0
83CD:B3 B0 A0
83D0:C6 CC D0
83D3:A0 B0 B0
83D6:A0
83D7:A0 A0 A0  586         ASC  /           IMAGE: FE 7F 63 62 7C 78 60 F2      /
83DA:A0 C9 CD
83DD:C1 C7 C5
83E0:BA A0 C6
83E3:C5 A0 B7
83E6:C6 A0 B6
83E9:B3 A0 B6
83EC:B2 A0 B7
83EF:C3 A0 B7
83F2:B8 A0 B6
83F5:B0 A0 C6
83F8:B2 A0 A0
83FB:A0 A0 A0
83FE:A0
83FF:A0 A0 A0  587         ASC  /           SOUND: A:00 B:00 C:00             /
8402:A0 D3 CF
8405:D5 CE C4
8408:BA A0 C1
```

```
840B:BA B0 B0
840E:A0 C2 BA
8411:B0 B0 A0
8414:C3 BA B0
8417:B0 A0 A0
841A:A0 A0 A0
841D:A0 A0 A0
8420:A0 A0 A0
8423:A0 A0 A0
8426:A0
8427:D0 CC C1  588         ASC  /PLAYER2:COLOR 00 VERT 1E HORZ 30 FLP 00 /
842A:D9 C5 D2
842D:B2 BA C3
8430:CF CC CF
8433:D2 A0 B0
8436:B0 A0 D6
8439:C5 D2 D4
843C:A0 B1 C5
843F:A0 CB CF
8442:D2 DA A0
8445:B3 B0 A0
8448:C6 CC D0
844B:A0 B0 B0
844E:A0
844F:A0 A0 A0  589         ASC  /            IMAGE: FE 7F 63 62 7C 78 60 F6      /
8452:A0 C9 CD
8455:C1 C7 C5
8458:BA A0 C6
845B:C5 A0 B7
845E:C6 A0 B6
8461:B3 A0 B6
8464:B2 A0 B7
8467:C3 A0 B7
846A:B8 A0 B6
846D:B0 A0 C6
8470:B6 A0 A0
8473:A0 A0 A0
8476:A0
8477:A0 A0 A0  590         ASC  /            SOUND: A:00 B:00 C:00             /
847A:A0 D3 CF
847D:D5 CE C4
8480:BA A0 C1
8483:BA B0 B0
8486:A0 C2 BA
8489:B0 B0 A0
848C:C3 BA B0
848F:B0 A0 A0
8492:A0 A0 A0
8495:A0 A0 A0
8498:A0 A0 A0
849B:A0 A0 A0
849E:A0
849F:C2 C1 C3  591         ASC  /BACKGROUND COLOR EA  OBJECT COLOR 64    /
84A2:CB C7 D2
84A5:CF D5 CE
84A8:C4 A0 C3
84AB:CF CC CF
84AE:D2 A0 C5
84B1:C1 A0 A0
84B4:CF C2 CA
84B7:C5 C3 D4
84BA:A0 C3 CF
84BD:CC CF D2
84C0:A0 B6 B4
84C3:A0 A0 A0
84C6:A0
```

```
84C7:CF C2 CA  592          ASC  /OBJECT   A IMAGE: 18 24 42 42 7E 42 42 CE /
84CA:C5 C3 D4
84CD:A0 C1 A0
84D0:C9 CD C1
84D3:C7 C5 BA
84D6:A0 B1 B8
84D9:A0 B2 B4
84DC:A0 B4 B2
84DF:A0 B4 B2
84E2:A0 B7 C5
84E5:A0 B4 B2
84E8:A0 B4 B2
84EB:A0 C3 C5
84EE:A0
84EF:CF C2 CA  593          ASC  /OBJECT   B IMAGE: FE 61 61 7E 7E 61 61 FE /
84F2:C5 C3 D4
84F5:A0 C2 A0
84F8:C9 CD C1
84FB:C7 C5 BA
84FE:A0 C6 C5
8501:A0 B6 B1
8504:A0 B6 B1
8507:A0 B7 C5
850A:A0 B7 C5
850D:A0 B6 B1
8510:A0 B6 B1
8513:A0 C6 C5
8516:A0
8517:CF C2 CA  594          ASC  /OBJECT   C IMAGE: 7E FF FE F0 F0 FE FF 7E /
851A:C5 C3 D4
851D:A0 C3 A0
8520:C9 CD C1
8523:C7 C5 BA
8526:A0 B7 C5
8529:A0 C6 C6
852C:A0 C6 C5
852F:A0 C6 B0
8532:A0 C6 B0
8535:A0 C6 C5
8538:A0 C6 C6
853B:A0 B7 C5
853E:A0
853F:D2 C5 C7  595          ASC  /REG03:00 P1VMODE:00 P2VMODE:00 FVMODE:05/
8542:B0 B3 BA
8545:B0 B0 A0
8548:D0 B1 D6
854B:CD CF C4
854E:C5 BA B0
8551:B0 A0 D0
8554:B2 D6 CD
8557:CF C4 C5
855A:BA B0 B0
855D:A0 C6 D6
8560:CD CF C4
8563:C5 BA B0
8566:B5
8567:D0 B1 C8  596          ASC  /P1HRES:00 P2HRES:00 S1HRES:00 S2HRES:00 /
856A:D2 C5 D3
856D:BA B0 B0
8570:A0 D0 B2
8573:C8 D2 C5
8576:D3 BA B0
8579:B0 A0 D3
857C:B1 C8 D2
857F:C5 D3 BA
8582:B0 B0 A0
```

```
8585:D3 B2 C8
8588:D2 C5 D3
858B:BA B0 B0
858E:A0
858F:D3 B3 C8  597          ASC  /S3HRES:00 S3SHOT:00 S3HORZ:00 P1DELAY:00/
8592:D2 C5 D3
8595:BA B0 B0
8598:A0 D3 B3
859B:D3 C8 CF
859E:D4 BA B0
85A1:B0 A0 D3
85A4:B3 C8 CF
85A7:D2 DA BA
85AA:B0 B0 A0
85AD:D0 B1 C4
85B0:C5 CC C1
85B3:D9 BA B0
85B6:B0
85B7:D0 B2 C4  598          ASC  /P2DELAY:00 REG27:00 S1CONT:00 S2CONT:00 /
85BA:C5 CC C1
85BD:D9 BA B0
85C0:B0 A0 D2
85C3:C5 C7 B2
85C6:B7 BA B0
85C9:B0 A0 D3
85CC:B1 C3 CF
85CF:CE D4 BA
85D2:B0 B0 A0
85D5:D3 B2 C3
85D8:CF CE D4
85DB:BA B0 B0
85DE:A0
85DF:C8 DA D3  599          ASC  /HZSCRL:00 NOINC:00 COLRES:00 REG2D:00   /
85E2:C3 D2 CC
85E5:BA B0 B0
85E8:A0 CE CF
85EB:C9 CE C3
85EE:BA B0 B0
85F1:A0 C3 CF
85F4:CC D2 C5
85F7:D3 BA B0
85FA:B0 A0 D2
85FD:C5 C7 B2
8600:C4 BA B0
8603:B0 A0 A0
8606:A0
8607:D2 C5 C7  600          ASC  /REG2E:00 REG2F:00                    /
860A:B2 C5 BA
860D:B0 B0 A0
8610:D2 C5 C7
8613:B2 C6 BA
8616:B0 B0 A0
8619:A0 A0 A0
861C:A0 A0 A0
861F:A0 A0 A0
8622:A0 A0 A0
8625:A0 A0 A0
8628:A0 A0 A0
862B:A0 A0 A0
862E:A0
862F:C7 C5 CE  601          ASC  /GENERAL  WRITE LOC:0081 VAL:00  READ=00  /
8632:C5 D2 C1
8635:CC A0 D7
8638:D2 C9 D4
863B:C5 A0 CC
863E:CF C3 BA
```

```
8641:B0 B0 B8
8644:B1 A0 D6
8647:C1 CC BA
864A:B0 B0 A0
864D:A0 D2 C5
8650:C1 C4 BD
8653:B0 B0 A0
8656:A0
8657:D2 C5 C1  602          ASC  /READ     ONLY LOCATIONS WITH MASK:81        /
865A:C4 A0 CF
865D:CE CC D9
8660:A0 CC CF
8663:C3 C1 D4
8666:C9 CF CE
8669:D3 A0 D7
866C:C9 D4 C8
866F:A0 CD C1
8672:D3 CB BA
8675:B8 B1 A0
8678:A0 A0 A0
867B:A0 A0 A0
867E:A0
867F:D2 C5 C7  603          ASC  /REG30:00 REG31:00 REG32:00 REG33:00     /
8682:B3 B0 BA
8685:B0 B0 A0
8688:D2 C5 C7
868B:B3 B1 BA
868E:B0 B0 A0
8691:D2 C5 C7
8694:B3 B2 BA
8697:B0 B0 A0
869A:D2 C5 C7
869D:B3 B3 BA
86A0:B0 B0 A0
86A3:A0 A0 A0
86A6:A0
86A7:D2 C5 C7  604          ASC  /REG34:00 REG35:00 REG36:00 REG37:00     /
86AA:B3 B4 BA
86AD:B0 B0 A0
86B0:D2 C5 C7
86B3:B3 B5 BA
86B6:B0 B0 A0
86B9:D2 C5 C7
86BC:B3 B6 BA
86BF:B0 B0 A0
86C2:D2 C5 C7
86C5:B3 B7 BA
86C8:B0 B0 A0
86CB:A0 A0 A0
86CE:A0
86CF:D2 C5 C7  605          ASC  /REG38:00 REG39:00 REG3A:00 REG3B:00     /
86D2:B3 B8 BA
86D5:B0 B0 A0
86D8:D2 C5 C7
86DB:B3 B9 BA
86DE:B0 B0 A0
86E1:D2 C5 C7
86E4:B3 C1 BA
86E7:B0 B0 A0
86EA:D2 C5 C7
86ED:B3 C2 BA
86F0:B0 B0 A0
86F3:A0 A0 A0
86F6:A0
86F7:D2 C5 C7  606          ASC  /REG3C:00 REG3D:00 REG3E:00 REG3F:00     /
86FA:B3 C3 BA
```

```
86FD:B0 B0 A0
8700:D2 C5 C7
8703:B3 C4 BA
8706:B0 B0 A0
8709:D2 C5 C7
870C:B3 C5 BA
870F:B0 B0 A0
8712:D2 C5 C7
8715:B3 C6 BA
8718:B0 B0 A0
871B:A0 A0 A0
871E:A0
871F:          607 *
871F:          608 *
871F:A0 A0 A0  609 SR0     ASC  /              XXXX XXXXXXXX XXXXXXX XXXXXX      /
8722:A0 A0 A0
8725:D8 D8 D8
8728:D8 A0 D8
872B:D8 D8 D8
872E:D8 D8 D8
8731:D8 A0 D8
8734:D8 D8 D8
8737:D8 D8 D8
873A:A0 D8 D8
873D:D8 D8 D8
8740:D8 A0 A0
8743:A0 A0 A0
8746:A0
8747:D8 D8 D8  610 SR1     ASC  /XXXXXX:AA                          /
874A:D8 D8 D8
874D:BA C1 C1
8750:A0 A0 A0
8753:A0 A0 A0
8756:A0 A0 A0
8759:A0 A0 A0
875C:A0 A0 A0
875F:A0 A0 A0
8762:A0 A0 A0
8765:A0 A0 A0
8768:A0 A0 A0
876B:A0 A0 A0
876E:A0
876F:D8 D8 D8  611 SR2     ASC  /XXXXXXX:XXXXX AB XXXX AF XXXX B9 XXX 0B /
8772:D8 D8 D8
8775:D8 BA D8
8778:D8 D8 D8
877B:D8 A0 C1
877E:C2 A0 D8
8781:D8 D8 D8
8784:A0 C1 C6
8787:A0 D8 D8
878A:D8 D8 A0
878D:B8 B9 A0
8790:D8 D8 D8
8793:A0 B0 C2
8796:A0
8797:A0 A0 A0  612 SR3     ASC  /           XXXXX: B3 B4 B5 B6 B7 B8 B9 BA      /
879A:A0 D8 D8
879D:D8 D8 D8
87A0:BA A0 C2
87A3:B3 A0 C2
87A6:B4 A0 C2
87A9:B5 A0 C2
87AC:B6 A0 C2
87AF:B7 A0 C2
87B2:B8 A0 C2
```

```
87B5:B9 A0 C2
87B8:C1 A0 A0
87BB:A0 A0 A0
87BE:A0
87BF:A0 A0 A0  613 SR4     ASC  /            XXXXX: X:15 X:17 X:19              /
87C2:A0 D8 D8
87C5:D8 D8 D8
87C8:BA A0 D8
87CB:BA B1 B5
87CE:A0 D8 BA
87D1:B1 B7 A0
87D4:D8 BA B1
87D7:B9 A0 A0
87DA:A0 A0 A0
87DD:A0 A0 A0
87E0:A0 A0 A0
87E3:A0 A0 A0
87E6:A0
87E7:D8 D8 D8  614 SR5     ASC  /XXXXXXX:XXXXX AC XXXX B0 XXXX 8A XXX 0C /
87EA:D8 D8 D8
87ED:D8 BA D8
87F0:D8 D8 D8
87F3:D8 A0 C1
87F6:C3 A0 D8
87F9:D8 D8 D8
87FC:A0 C2 B0
87FF:A0 D8 D8
8802:D8 D8 A0
8805:B8 C1 A0
8808:D8 D8 D8
880B:A0 B0 C3
880E:A0
880F:A0 A0 A0  615 SR6     ASC  /            XXXXX: BB BC BD BE BF C0 C1 C2     /
8812:A0 D8 D8
8815:D8 D8 D8
8818:BA A0 C2
881B:C2 A0 C2
881E:C3 A0 C2
8821:C4 A0 C2
8824:C5 A0 C2
8827:C6 A0 C3
882A:B0 A0 C3
882D:B1 A0 C3
8830:B2 A0 A0
8833:A0 A0 A0
8836:A0
8837:A0 A0 A0  616 SR7     ASC  /            XXXXX: X:16 X:18 X:1A              /
883A:A0 D8 D8
883D:D8 D8 D8
8840:BA A0 D8
8843:BA B1 B6
8846:A0 D8 BA
8849:B1 B8 A0
884C:D8 BA B1
884F:C1 A0 A0
8852:A0 A0 A0
8855:A0 A0 A0
8858:A0 A0 A0
885B:A0 A0 A0
885E:A0
885F:D8 D8 D8  617 SR8     ASC  /XXXXXXXXXX XXXXX AD  XXXXXX XXXXX AE    /
8862:D8 D8 D8
8865:D8 D8 D8
8868:D8 A0 D8
886B:D8 D8 D8
886E:D8 A0 C1
```

```
8871:C4 A0 A0
8874:D8 D8 D8
8877:D8 D8 D8
887A:A0 D8 D8
887D:D8 D8 D8
8880:A0 C1 C5
8883:A0 A0 A0
8886:A0
8887:D8 D8 D8  618 SR9     ASC  /XXXXXX   X XXXXX: C4 C5 C6 C7 C8 C9 CA CB /
888A:D8 D8 D8
888D:A0 D8 A0
8890:D8 D8 D8
8893:D8 D8 BA
8896:A0 C3 B4
8899:A0 C3 B5
889C:A0 C3 B6
889F:A0 C3 B7
88A2:A0 C3 B8
88A5:A0 C3 B9
88A8:A0 C3 C1
88AB:A0 C3 C2
88AE:A0
88AF:D8 D8 D8  619 SRA     ASC  /XXXXXX   X XXXXX: CD CE CF D0 D1 D2 D3 D4 /
88B2:D8 D8 D8
88B5:A0 D8 A0
88B8:D8 D8 D8
88BB:D8 D8 BA
88BE:A0 C3 C4
88C1:A0 C3 C5
88C4:A0 C3 C6
88C7:A0 C4 B0
88CA:A0 C4 B1
88CD:A0 C4 B2
88D0:A0 C4 B3
88D3:A0 C4 B4
88D6:A0
88D7:D8 D8 D8  620 SRB     ASC  /XXXXXX   X XXXXX: D6 D7 D8 D9 DA DB DC DD /
88DA:D8 D8 D8
88DD:A0 D8 A0
88E0:D8 D8 D8
88E3:D8 D8 BA
88E6:A0 C4 B6
88E9:A0 C4 B7
88EC:A0 C4 B8
88EF:A0 C4 B9
88F2:A0 C4 C1
88F5:A0 C4 C2
88F8:A0 C4 C3
88FB:A0 C4 C4
88FE:A0
88FF:D8 D8 D8  621 SRC     ASC  /XXXXX:03 XXXXXXX:04 XXXXXXX:05 XXXXXX:A5/
8902:D8 D8 BA
8905:B0 B3 A0
8908:D8 D8 D8
890B:D8 D8 D8
890E:D8 BA B0
8911:B4 A0 D8
8914:D8 D8 D8
8917:D8 D8 D8
891A:BA B0 B5
891D:A0 D8 D8
8920:D8 D8 D8
8923:D8 BA C1
8926:B5
8927:D8 D8 D8  622 SRD     ASC  /XXXXXX:10 XXXXXX:11 XXXXXX:12 XXXXXX:13 /
892A:D8 D8 D8
```

```
892D:BA B1 B0
8930:A0 D8 D8
8933:D8 D8 D8
8936:D8 BA B1
8939:B1 A0 D8
893C:D8 D8 D8
893F:D8 D8 BA
8942:B1 B2 A0
8945:D8 D8 D8
8948:D8 D8 D8
894B:BA B1 B3
894E:A0
894F:D8 D8 D8  623 SRE     ASC  /XXXXXX:14 XXXXXX:1F XXXXXX:24 XXXXXXX:25/
8952:D8 D8 D8
8955:BA B1 B4
8958:A0 D8 D8
895B:D8 D8 D8
895E:D8 BA B1
8961:C6 A0 D8
8964:D8 D8 D8
8967:D8 D8 BA
896A:B2 B4 A0
896D:D8 D8 D8
8970:D8 D8 D8
8973:D8 BA B2
8976:B5
8977:D8 D8 D8  624 SRF     ASC  /XXXXXXX:26 XXXXX:27 XXXXXX:28 XXXXXX:29 /
897A:D8 D8 D8
897D:D8 BA B2
8980:B6 A0 D8
8983:D8 D8 D8
8986:D8 BA B2
8989:B7 A0 D8
898C:D8 D8 D8
898F:D8 D8 BA
8992:B2 B8 A0
8995:D8 D8 D8
8998:D8 D8 D8
899B:BA B2 B9
899E:A0
899F:D8 D8 D8  625 SR10    ASC  /XXXXXX:2A XXXXX:2B XXXXXX:2C XXXXX:2D   /
89A2:D8 D8 D8
89A5:BA B2 C1
89A8:A0 D8 D8
89AB:D8 D8 D8
89AE:BA B2 C2
89B1:A0 D8 D8
89B4:D8 D8 D8
89B7:D8 BA B2
89BA:C3 A0 D8
89BD:D8 D8 D8
89C0:D8 BA B2
89C3:C4 A0 A0
89C6:A0
89C7:D8 D8 D8  626 SR11    ASC  /XXXXX:2E XXXXX:2F                    /
89CA:D8 D8 BA
89CD:B2 C5 A0
89D0:D8 D8 D8
89D3:D8 D8 BA
89D6:B2 C6 A0
89D9:A0 A0 A0
89DC:A0 A0 A0
89DF:A0 A0 A0
89E2:A0 A0 A0
89E5:A0 A0 A0
89E8:A0 A0 A0
```

```
89EB:A0 A0 A0
89EE:A0
89EF:D8 D8 D8  627 SR12    ASC  /XXXXXXX  XXXXX XXX:0000 XXX:00  XXXX=XX  /
89F2:D8 D8 D8
89F5:D8 A0 D8
89F8:D8 D8 D8
89FB:D8 A0 D8
89FE:D8 D8 BA
8A01:B0 B0 B0
8A04:B0 A0 D8
8A07:D8 D8 BA
8A0A:B0 B0 A0
8A0D:A0 D8 D8
8A10:D8 D8 BD
8A13:D8 D8 A0
8A16:A0
8A17:D8 D8 D8  628 SR13    ASC  /XXXX     XXXX XXXXXXXXX XXXX XXXX:00        /
8A1A:D8 A0 D8
8A1D:D8 D8 D8
8A20:A0 D8 D8
8A23:D8 D8 D8
8A26:D8 D8 D8
8A29:D8 A0 D8
8A2C:D8 D8 D8
8A2F:A0 D8 D8
8A32:D8 D8 BA
8A35:B0 B0 A0
8A38:A0 A0 A0
8A3B:A0 A0 A0
8A3E:A0
8A3F:D8 D8 D8  629 SR14    ASC  /XXXXX:30 XXXXX:31 XXXXX:32 XXXXX:33     /
8A42:D8 D8 BA
8A45:B3 B0 A0
8A48:D8 D8 D8
8A4B:D8 D8 BA
8A4E:B3 B1 A0
8A51:D8 D8 D8
8A54:D8 D8 BA
8A57:B3 B2 A0
8A5A:D8 D8 D8
8A5D:D8 D8 BA
8A60:B3 B3 A0
8A63:A0 A0 A0
8A66:A0
8A67:D8 D8 D8  630 SR15    ASC  /XXXXX:34 XXXXX:35 XXXXX:36 XXXXX:37     /
8A6A:D8 D8 BA
8A6D:B3 B4 A0
8A70:D8 D8 D8
8A73:D8 D8 BA
8A76:B3 B5 A0
8A79:D8 D8 D8
8A7C:D8 D8 BA
8A7F:B3 B6 A0
8A82:D8 D8 D8
8A85:D8 D8 BA
8A88:B3 B7 A0
8A8B:A0 A0 A0
8A8E:A0
8A8F:D8 D8 D8  631 SR16    ASC  /XXXXX:38 XXXXX:39 XXXXX:3A XXXXX:3B     /
8A92:D8 D8 BA
8A95:B3 B8 A0
8A98:D8 D8 D8
8A9B:D8 D8 BA
8A9E:B3 B9 A0
8AA1:D8 D8 D8
8AA4:D8 D8 BA
```

```
8AA7:B3 C1 A0
8AAA:D8 D8 D8
8AAD:D8 D8 BA
8AB0:B3 C2 A0
8AB3:A0 A0 A0
8AB6:A0
8AB7:D8 D8 D8  632 SR17    ASC  /XXXXX:3C XXXXX:3D XXXXX:3E XXXXX:3F     /
8ABA:D8 D8 BA
8ABD:B3 C3 A0
8AC0:D8 D8 D8
8AC3:D8 D8 BA
8AC6:B3 C4 A0
8AC9:D8 D8 D8
8ACC:D8 D8 BA
8ACF:B3 C5 A0
8AD2:D8 D8 D8
8AD5:D8 D8 BA
8AD8:B3 C6 A0
8ADB:A0 A0 A0
8ADE:A0
8ADF:          633 *
8ADF:          634 *
8ADF:87 87 87  635 FHLIST  DFB  <SR0,<SR1,<SR2,<SR3,<SR4,<SR5
8AE2:87 87 87
8AE5:88 88 88  636         DFB  <SR6,<SR7,<SR8,<SR9,<SRA,<SRB
8AE8:88 88 88
8AEB:88 89 89  637         DFB  <SRC,<SRD,<SRE,<SRF,<SR10,<SR11
8AEE:89 89 89
8AF1:89 8A 8A  638         DFB  <SR12,<SR13,<SR14,<SR15,<SR16,<SR17
8AF4:8A 8A 8A
8AF7:          639 *
8AF7:1F 47 6F  640 FLLIST  DFB  >SR0,>SR1,>SR2,>SR3,>SR4,>SR5
8AFA:97 BF E7
8AFD:0F 37 5F  641         DFB  >SR6,>SR7,>SR8,>SR9,>SRA,>SRB
8B00:87 AF D7
8B03:FF 27 4F  642         DFB  >SRC,>SRD,>SRE,>SRF,>SR10,>SR11
8B06:77 9F C7
8B09:EF 17 3F  643         DFB  >SR12,>SR13,>SR14,>SR15,>SR16,>SR17
8B0C:67 8F B7
8B0F:          644 *
8B0F:          645 *
8B0F:          646 SAVBUF  DS   $80       ; SPACE FOR PAGE 0 STUFF WE NEED TO SAVE AND RESTORE
8B8F:          647 *
8B8F:          648 *
8B8F:          649 * END OF XCONRTOL

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

80FB ADRCNV
815F CKY1
90 CROW
822C DSPCHR
8AF7 FLLIST
8268 GVAL1
80C3 INCR1
C000 KBD
9B MASK
800B MLP1
8190 MOVRGH
821F MTEST
?8202 NONUM
8050 NTRGH
93 OLDCR
20 RDCMD
80F2 RDR1
85 RLOC
80D6 SA2
8315 SAVER
8257 SETREG
821D SPDONE
8747 SR1
8A17 SR13
8AB7 SR17
87E7 SR5
8887 SR9
8927 SRD
8154 TH1
814C TOHEX
828D WTREG

91 CCOL
8347 CLLIST
81BA CUPDTE
8E FBASE
86 FSTAT
811F HEXKEY
80B7 INCRR
C010 KSTRB
8189 MDN1
8026 MLP2
8167 MOVUP
8175 MUP1
822A NOSPEC
8032 NTUP
?82A9 PARAM1
97 RDHGH
80E7 RDREG
95 ROW
80DE SA3
8317 SAVLP
82F1 SL1
8204 SPECCK
899F SR10
8A3F SR14
876F SR2
880F SR6
88AF SRA
894F SRE
96 TMP1
40 WRCMD

8158 CHKKEY
81E2 CRADDR
823A DOWRT
88 FDATA
824A GETFLD
8128 HK1
82E0 INFLD
99 LSTKEY
9B MLF1
817C MOVDWN
819F MRT0
82D1 NOCHNG
803C NTDWN
81EF NUMFLD
?82AD PARAM2
98 RDLOW
8324 RESLP
92 RRCOL
10 SACMD
835F SCREEN
82FD SL2
8074 SPECR
89C7 SR11
8A67 SR15
8797 SR3
8837 SR7
88D7 SRB
8977 SRF
828A TOB1
8290 WTR1

832F CHLIST
8C CRBASE
8134 DSPBYT
8ADF FHLIST
825E GETVAL
82C1 ILP
82A0 INIT
9A LSTRD
8003 MLOOP
81A8 MOVLFT
81A1 MRT1
8132 NOHEX
8046 NTLFT
94 OLDCCL
8A RBASE
80EA RDR0
8322 RESTORER
80CE SA1
8B0F SAVBUF
80CC SETADR
82E7 SLOAD
871F SR0
89EF SR12
8A8F SR16
87BF SR4
885F SR8
88FF SRC
?8000 START
8282 TOBIN
8298 WTR2

```
                10 SACMD        20 RDCMD        40 WRCMD        85 RLOC
  86 FSTAT      88 FDATA        8A RBASE        8C CRBASE
  8E FBASE      90 CROW         91 CCOL         92 RRCOL
  93 OLDCR      94 OLDCCL       95 ROW          96 TMP1
  97 RDHGH      98 RDLOW        99 LSTKEY       9A LSTRD
  9B MASK     8000 START      8003 MLOOP      800B MLP1
8026 MLP2     8032 NTUP       803C NTDWN      8046 NTLFT
8050 NTRGH    8074 SPECR      80B7 INCRR      80C3 INCR1
80CC SETADR   80CE SA1        80D6 SA2        80DE SA3
80E7 RDREG    80EA RDR0       80F2 RDR1       80FB ADRCNV
811F HEXKEY   8128 HK1        8132 NOHEX      8134 DSPBYT
814C TOHEX    8154 TH1        8158 CHKKEY     815F CKY1
8167 MOVUP    8175 MUP1       817C MOVDWN     8189 MDN1
8190 MOVRGH   819F MRT0       81A1 MRT1       81A8 MOVLFT
81B5 MLF1     81BA CUPDTE     81E2 CRADDR     81EF NUMFLD
8202 NONUM    8204 SPECCK     821D SPDONE     821F MTEST
822A NOSPEC   822C DSPCHR     823A DOWRT      824A GETFLD
8257 SETREG   825E GETVAL     8268 GVAL1      8282 TOBIN
828A TOB1     828D WTREG      8290 WTR1       8298 WTR2
82A0 INIT     82A9 PARAM1     82AD PARAM2     82C1 ILP
82D1 NOCHNG   82E0 INFLD      82E7 SLOAD      82F1 SL1
82FD SL2      8315 SAVER      8317 SAVLP      8322 RESTORER
8324 RESLP    832F CHLIST     8347 CLLIST     835F SCREEN
871F SR0      8747 SR1        876F SR2        8797 SR3
87BF SR4      87E7 SR5        880F SR6        8837 SR7
885F SR8      8887 SR9        88AF SRA        88D7 SRB
88FF SRC      8927 SRD        894F SRE        8977 SRF
899F SR10     89C7 SR11       89EF SR12       8A17 SR13
8A3F SR14     8A67 SR15       8A8F SR16       8AB7 SR17
8ADF FHLIST   8AF7 FLLIST     8B0F SAVBUF     C000 KBD
C010 KSTRB
```

```
SOURCE FILE: FMON
0000:           1 * FMON -- THE VCS SIDE OF THE DEBUGGER SYSTEM
0000:           2 * COPYRIGHT 1982 BY FROBCO
0000:           3 * ALL RIGHTS RESERVED
0000:           4 *
0000:           5 * VERSION 1.3 -- LAST MODIFIED 1/26/83
0000:           6 *
0000:           7 *
0000:           8 * SYSTEM DEFINITIONS
0000:           9 *
FFF1:          10 PSTAT   EQU  $FFF1     ; STATUS PORT
FFF2:          11 RDATA   EQU  $FFF2     ; DATA PORT FROM APPLE
FFF0:          12 WDATA   EQU  $FFF0     ; DATA PORT TO APPLE
0000:          13 *
FFF4:          14 F1REST  EQU  $FFF4     ; FIRST PART OF FLAG RESTORE
FFF5:          15 F2REST  EQU  $FFF5     ; LAST PART OF FLAG RESTORE
FFF6:          16 AREST   EQU  $FFF6     ; RESTORE A FROM HERE
FFF7:          17 XREST   EQU  $FFF7     ; RESTORE X FROM HERE
FFF8:          18 YREST   EQU  $FFF8     ; RESTORE Y FORM HERE
FFF9:          19 SREST   EQU  $FFF9     ; RESTORE S FORM HERE
0000:          20 *
0000:          21 *
0000:          22 *
0000:          23 * RAM DEFINITIONS
0000:          24 *
00E0:          25 RSAVE   EQU  $E0       ; PLACE TO START SAVEING RAM LOCATIONS
00F8:          26 RBASE   EQU  $F8       ; GENERAL 16 BIT POINTER
0000:          27 *
0000:          28 *
----- NEXT OBJECT FILE NAME IS FMON.OBJ0
FF00:          29         ORG  $FF00     ; FMON ADDRESS SPACE
FF00:          30 *
FF00:          31 *
FF00:          32 * THIS IS THE ENTRY POINT UPON BREAK POINT
FF00:          33 * THE SYSTEM SAVE THE STATE BY PASSING INFORMATION
FF00:          34 * OVER TO THE APPLE.
FF00:          35 *
FF00:          36 * THE TRICK HERE IS TO SAVE THE STATE OF THE N AND Z
FF00:          37 * FLAGS WITHOUT USING THE STACK WHICH MAY OR MAY NOT
FF00:          38 * BE SET UP AS A STACK AT THE TIME OF THE BREAK POINT
FF00:          39 *
FF00:          40 *
FF00:8D F0 FF  41 BREAK   STA  WDATA     ; SEND THE A REGISTER TO THE APPLE
FF03:D0 09     42         BNE  NOTZ      ; BRANCH IF Z FLAG NOT SET
FF05:AD F1 FF  43 BRKW1   LDA  PSTAT     ; ELSE SEND A CODE THAT WILL
FF08:10 FB     44         BPL  BRKW1     ; RESTORE N=0 AND Z=1 WHEN WE GET BACK
FF0A:A9 FF     45         LDA  #$FF      ; AND DO AN INC MEM. (FF WILL INC TO 0)
FF0C:D0 12     46         BNE  BRK1      ; BRANCH ALWAYS
FF0E:          47 *
FF0E:30 09     48 NOTZ    BMI  FNEG      ; KEEP GOING IF N=1
FF10:AD F1 FF  49 BRKW2   LDA  PSTAT     ; ELSE CHOOSE CODE THAT WILL RESTORE
FF13:10 FB     50         BPL  BRKW2
FF15:A9 01     51         LDA  #$01      ; NON ZERO AND POSITIVE
FF17:D0 07     52         BNE  BRK1      ; BRANCH ALWAYS
FF19:          53 *
FF19:          54 *
FF19:AD F1 FF  55 FNEG    LDA  PSTAT     ; LOOK AT THE OK TO WRITE FLAG
FF1C:10 FB     56         BPL  FNEG       ; WAIT FOR IT TO GO HIGH
FF1E:A9 80     57         LDA  #$80      ; THIS WILL INC TO NEG
FF20:8D F0 FF  58 BRK1    STA  WDATA     ; SEND FLAG CODE
FF23:AD F1 FF  59 BRKW4   LDA  PSTAT     ; WAIT TO STORE X REG
FF26:10 FB     60         BPL  BRKW4
FF28:8E F0 FF  61         STX  WDATA     ; SEND THE X REGISTER
```

```
FF2B:AD F1 FF   62 BRK2    LDA  PSTAT     ; LOOK AT THE OK TO WRITE FLAG
FF2E:10 FB      63         BPL  BRK2      ; KEEP WAITING
FF30:8C F0 FF   64         STY  WDATA     ; SEND THE Y REGISTER
FF33:AD F1 FF   65 BRK3    LDA  PSTAT     ; CHECK FLAG AGAIN
FF36:10 FB      66         BPL  BRK3      ; AND WAIT FOR IT
FF38:BA         67         TSX            ; GET THE STACK POINTER
FF39:8E F0 FF   68         STX  WDATA     ; SEND IT OVER
FF3C:A2 E0      69         LDX  #RSAVE    ; SET UP LOOP TO SAVE RAM LOCATIONS
FF3E:AD F1 FF   70 BRK4    LDA  PSTAT     ; CHECK OK TO WRITE FLAG
FF41:10 FB      71         BPL  BRK4      ; LOOP ON IT
FF43:B5 00      72         LDA  0,X       ; GET RAM VALUE
FF45:8D F0 FF   73         STA  WDATA     ; SEND IT TO THE APPLE
FF48:E8         74         INX            ; MOVE TO NEXT
FF49:30 F3      75         BMI  BRK4
FF4B:           76 *
FF4B:CA         77         DEX            ; NOW RESET THE STACK POINTER
FF4C:9A         78         TXS
FF4D:08         79         PHP            ; PUSH THE FLAGS
FF4E:68         80         PLA            ; GET THEM IN A
FF4F:20 98 FF   81         JSR  WBA       ; GO SEND FLAGS
FF52:           82 *
FF52:           83 *
FF52:           84 * INITIALIZATION FROM RESTART
FF52:           85 *
FF52:           86 RESTART EQU  *
FF52:A2 FF      87         LDX  #$FF      ; SETUP STACK POINTER
FF54:9A         88         TXS
FF55:D8         89         CLD            ; CLEAR DECIMAL MODE
FF56:AD F2 FF   90         LDA  RDATA     ; CLEAR BUFFER
FF59:           91 *
FF59:           92 * FALL INTO COMMAND LOOP
FF59:           93 *
FF59:           94 *
FF59:           95 * THIS IS THE COMMAND LEVEL
FF59:           96 *
FF59:20 A1 FF   97 CI      JSR  RBA       ; GET A POSSIBLE COMMAND BYTE
FF5C:C9 10      98         CMP  #$10      ; DO WE READ MEM?
FF5E:F0 5B      99         BEQ  RM        ; IF SO DO IT
FF60:C9 20     100         CMP  #$20      ; DO WE WRITE MEM?
FF62:F0 66     101         BEQ  WM        ; IF SO DO IT
FF64:C9 30     102         CMP  #$30      ; DO WE GO FROM BREAK?
FF66:F0 07     103         BEQ  GO        ; IF SO DO IT
FF68:C9 40     104         CMP  #$40      ; DO WE JUMP INDIRECT?
FF6A:D0 ED     105         BNE  CI        ; IF NOT KEEP LOOPING
FF6C:4C D9 FF  106         JMP  GOI       ; ELSE DO THE JMP INDIRECT
FF6F:          107 *
FF6F:          108 * ROUTINE TO RESTORE STATE AFTER BREAK POINT AND JMP
FF6F:          109 * BACK INTO THE VCS PROGRAM
FF6F:          110 *
FF6F:          111 *
FF6F:AD F4 FF  112 GO      LDA  F1REST    ; GET PART 1 OF THE FLAG RESTORE PROCESS
FF72:48        113         PHA            ; PUT IN FLAGS BY WAY OF STACK
FF73:28        114         PLP
FF74:          115 *
FF74:A2 20     116         LDX  #256-RSAVE ; LOOP COUNT FOR RAM-STUFF BACK
FF76:AD F1 FF  117 GO1     LDA  PSTAT     ; LOOK AT STATUS
FF79:29 40     118         AND  #$40      ; SEE IF STUFF THERE
FF7B:F0 F9     119         BEQ  GO1       ; IF NOT THEN WAIT
FF7D:AD F2 FF  120         LDA  RDATA     ; THEN GET VALUE
FF80:95 DF     121         STA  RSAVE-1,X ; PUT BACK IN RAM
FF82:CA        122         DEX            ; MOVE TO NEXT
FF83:D0 F1     123         BNE  GO1       ; LOOP
FF85:          124 *
FF85:          125 *
FF85:AE F9 FF  126         LDX  SREST     ; GET VALUE IN X TO RESTORE STACK POINTER
FF88:9A        127         TXS            ; RESTORE STACK POINTER
```

```
FF89:AC F8 FF  128         LDY  YREST     ; GET VALUE TO RESTORE Y
FF8C:AE F7 FF  129         LDX  XREST     ; GET VALUE TO RESTORE X
FF8F:AD F6 FF  130         LDA  AREST     ; GET VALUE TO RESTORE A
FF92:EE F5 FF  131         INC  F2REST    ; RESTORE Z AND N FLAGS
FF95:          132 * NOW ALL IS RESTORED SO JMP BACK
FF95:4C 00 FF  133 BJUMP   JMP  BREAK
FF98:          134 *
FF98:          135 *
FF98:          136 *
FF98:          137 *
FF98:          138 * HERE IS THE COMM PACKAGE
FF98:          139 *
FF98:2C F1 FF  140 WBA     BIT  PSTAT     ; GET READY TO WRITE A BYTE TO THE APPLE
FF9B:10 FB     141         BPL  WBA       ; WAIT FOR WRITE OK FLAG
FF9D:8D F0 FF  142         STA  WDATA     ; THEN DO IT AND RETURN
FFA0:60        143         RTS
FFA1:          144 *
FFA1:          145 *
FFA1:2C F1 FF  146 RBA     BIT  PSTAT     ; GET READY TO READ SOMETHING
FFA4:50 FB     147         BVC  RBA       ; WAIT FOR READ READY BIT
FFA6:AD F2 FF  148         LDA  RDATA     ; READ THE DATA
FFA9:60        149         RTS              ; THEN DONE
FFAA:          150 *
FFAA:          151 *
FFAA:          152 *
FFAA:          153 * ROUTINE TO SET UP A 16 BIT ADDRESS AND 8 BIT COUNT
FFAA:          154 *
FFAA:20 A1 FF  155 GETADR  JSR  RBA       ; GO GET HIGH ADDRESS PART
FFAD:85 F9     156         STA  RBASE+1   ; PUT INTO BASE POINTER
FFAF:20 A1 FF  157         JSR  RBA       ; GET LOWER PART
FFB2:85 F8     158         STA  RBASE     ; PUT IN POINTER
FFB4:20 A1 FF  159         JSR  RBA       ; GET THE LENGTH
FFB7:AA        160         TAX             ; PUT IN X REG
FFB8:A0 00     161         LDY  #0        ; USE Y FOR INDEX
FFBA:60        162         RTS            ; NOW ADDRESS AND COUNT SETUP
FFBB:          163 *
FFBB:          164 *
FFBB:          165 *
FFBB:          166 * ROUTINE TO READ OUT MEMORY
FFBB:20 AA FF  167 RM      JSR  GETADR    ; SET UP ADDRESS AND COUNT
FFBE:B1 F8     168 RM1     LDA  (RBASE),Y ; GET VALUE
FFC0:20 98 FF  169         JSR  WBA
FFC3:C8        170         INY            ; MOVE INDEX
FFC4:CA        171         DEX            ; DECREMENT COUNT
FFC5:D0 F7     172         BNE  RM1       ; LOOP TILL DONE
FFC7:4C 59 FF  173         JMP  CI        ; DONE
FFCA:          174 *
FFCA:          175 *
FFCA:          176 *
FFCA:          177 * ROUTINE TO WRITE MEMORY
FFCA:          178 *
FFCA:20 AA FF  179 WM      JSR  GETADR    ; GO GET ADDRESS AND COUNT
FFCD:20 A1 FF  180 WM1     JSR  RBA       ; GO GET BYTE
FFD0:91 F8     181         STA  (RBASE),Y ; STORE IT
FFD2:C8        182         INY            ; MOVE INDEX
FFD3:CA        183         DEX            ; DECREMENT COUNT
FFD4:D0 F7     184         BNE  WM1       ; LOOP TILL DONE
FFD6:4C 59 FF  185         JMP  CI        ; DONE
FFD9:          186 *
FFD9:          187 *
FFD9:          188 *
FFD9:          189 * ROUTINE TO TAKE A JMP INDIRECT
FFD9:          190 *  (USED TO PUT THINGS TO SLEEP)
FFD9:20 A1 FF  191 GOI     JSR  RBA       ; GET HIGH ORDER JMP ADDR
FFDC:85 F9     192         STA  RBASE+1   ; PUT IN POINTER HIGH
FFDE:20 A1 FF  193         JSR  RBA       ; GET LOW ORDER JMP ADDR
```

```
FFE1:85 F8     194          STA  RBASE     ; PUT IN POINTER LOW
FFE3:6C F8 00  195          JMP  (RBASE)   ; GO THERE
FFE6:          196 *
FFE6:          197 * PATCH AREA
FFE6:00 00     198          DW   0
FFE8:00 00     199          DW   0
FFEA:00 00     200          DW   0
FFEC:00 00     201          DW   0
FFEE:00 00     202          DW   0
FFF0:          203 *
FFF0:          204 *
FFF0:          205 *
FFF0:          206 * TOP OF MEMORY STUFF -- THIS ADDRESS MUST BE $FF00
FFF0:          207 *
FFF0:          208 * THESE REGISERS ARE USED FOR
FFF0:          209 * COMMUNICATION TO AMON -- SO DONT REUSE THIS AREA
FFF0:          210 *
FFF0:00 00     211          DW   0
FFF2:00 00     212          DW   0
FFF4:00 00     213          DW   0
FFF6:00 00     214          DW   0
FFF8:00 00     215          DW   0
FFFA:59 FF     216          DW   CI        ; REENTRY VECTOR
FFFC:52 FF     217          DW   RESTART   ; RESET VECTOR
FFFE:00 00     218          DW   0         ; END OF ADDRESS SPACE (FFFE)
0000:          219 *
0000:          220 *
0000:          221 *
0000:          222 * END OF FMON

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

```
FFF6 AREST      ?FF95 BJUMP     FF00 BREAK      FF20 BRK1
FF2B BRK2        FF33 BRK3      FF3E BRK4       FF05 BRKW1
FF10 BRKW2       FF23 BRKW4     FF59 CI         FFF4 F1REST
FFF5 F2REST      FF19 FNEG      FFAA GETADR     FF6F GO
FF76 GO1         FFD9 GOI       FF0E NOTZ       FFF1 PSTAT
FFA1 RBA           F8 RBASE     FFF2 RDATA      FF52 RESTART
FFBE RM1         FFBB RM          E0 RSAVE      FFF9 SREST
FF98 WBA         FFF0 WDATA     FFCA WM         FFCD WM1
FFF7 XREST       FFF8 YREST
```

```
                                E0 RSAVE        F8 RBASE        FF00 BREAK      FF05 BRKW1
FF0E NOTZ       FF10 BRKW2      FF19 FNEG       FF20 BRK1
FF23 BRKW4      FF2B BRK2       FF33 BRK3       FF3E BRK4
FF52 RESTART    FF59 CI         FF6F GO         FF76 GO1
?FF95 BJUMP     FF98 WBA        FFA1 RBA        FFAA GETADR
FFBB RM         FFBE RM1        FFCA WM         FFCD WM1
FFD9 GOI        FFF0 WDATA      FFF1 PSTAT      FFF2 RDATA
FFF4 F1REST     FFF5 F2REST     FFF6 AREST      FFF7 XREST
FFF8 YREST      FFF9 SREST
```

```
SOURCE FILE: PMOVE
0000:            1 * ************************
0000:            2 * * COPYRIGHT 1982      *
0000:            3 * * FROBCO  ALL RIGHTS RESERVED *
0000:            4 * *                       *
0000:            5 * ************************
0000:            6 *
0000:            7 * AUTHOR: KEN CLEMENTS
0000:            8 * DATE: 10/13/82
0000:            9 * LAST MODIFIED:  1/27/83
0000:           10 * VERSION 1.3
0000:           11 * PAGE MOVE
0000:           12 *
----- NEXT OBJECT FILE NAME IS PMOVE.OBJ0
0300:           13         ORG  $300
0300:A2 00      14 PMOVE   LDX  #0
0302:BD 00 10   15 PM1     LDA  $1000,X
0305:9D 00 10   16 PM2     STA  $1000,X
0308:E8         17         INX
0309:D0 F7      18         BNE  PM1
0304:           19 PFROM   EQU  PM1+2
0307:           20 PTO     EQU  PM2+2
030B:8E 03 03   21         STX  PFROM-1
030E:8E 06 03   22         STX  PTO-1
0311:60         23         RTS
0312:           24 *
0312:           25 *
0312:           26 * HERE IS A ROUTINE TO LOAD OR
0312:           27 * UNLOAD THE FROB
0312:           28 *
0312:00         29 FSLOT   DFB  0          ; SLOT NUMBER FOR THE FROB
0313:00         30 LFROM   DFB  0          ; WHERE IN MEMORY TO LOAD TO/FROM
0314:00         31 LNPGS   DFB  0          ; HOW MANY PAGES TO TRANSFER
0315:00         32 STPAGE  DFB  0          ; WHAT FROB PAGE TO START WITH
0316:           33 *
0316:A9 03      34 DNLOAD  LDA  #<PTO      ; PICKUP HIGH ORDER ADDRESS PART
0318:8D 65 03   35              STA        TX+2 ; SETUP FOR DOWNLOAD
031B:A9 07      36              LDA        #>PTO ; PICKUP LOW ORDER ADDRESS PART
031D:8D 64 03   37              STA        TX+1
0320:A9 03      38              LDA        #<PFROM
0322:8D 77 03   39              STA        TX1+2
0325:8D 89 03   40              STA        TX2+2
0328:A9 04      41              LDA        #>PFROM
032A:8D 76 03   42              STA        TX1+1
032D:8D 88 03   43              STA        TX2+1
0330:4C 4D 03   44              JMP         FXFER
0333:           45 *
0333:A9 03      46 UPLOAD  LDA  #<PFROM
0335:8D 65 03   47         STA  TX+2
0338:A9 04      48         LDA  #>PFROM
033A:8D 64 03   49         STA  TX+1
033D:A9 03      50         LDA  #<PTO
033F:8D 77 03   51         STA  TX1+2
0342:8D 89 03   52         STA  TX2+2
0345:A9 07      53         LDA  #>PTO
0347:8D 76 03   54         STA  TX1+1
034A:8D 88 03   55         STA  TX2+1
034D:           56 *
034D:AD 14 03   57 FXFER   LDA  LNPGS
0350:F0 34      58         BEQ  EXIT       ; DON'T DO FOR 0 PAGES
0352:A9 00      59         LDA  #0
0354:8D 06 03   60         STA  PTO-1
0357:8D 03 03   61         STA  PFROM-1
```

```
035A:A9 C0      62          LDA  #$C0
035C:8D 7D 03   63          STA  PPOINT+2
035F:18         64          CLC
0360:6D 12 03   65          ADC  FSLOT
0363:8D 07 03   66 TX       STA  PTO
0366:29 07      67          AND  #7         ; GET SLOT NUMBER BACK
0368:18         68          CLC             ; ADD IN DEV SELECT OFFSET
0369:69 08      69          ADC  #8
036B:0A         70          ASL  A          ; SHIFT IT OVER TO
036C:0A         71          ASL  A          ; FORM LOWER BYTE OF
036D:0A         72          ASL  A          ; DEVSEL ADDRESS FOR SLOT
036E:0A         73          ASL  A
036F:8D 7C 03   74          STA  PPOINT+1
0372:AD 13 03   75          LDA  LFROM      ; MEM PAGE ADDRESS
0375:8D 04 03   76 TX1      STA  PFROM
0378:           77 *
0378:AD 15 03   78 LDLOOP   LDA  STPAGE     ; FROB START PG
037B:8D E8 03   79 PPOINT   STA  1000       ; FROB PAGE SELECT
037E:20 00 03   80          JSR  PMOVE
0381:CE 14 03   81          DEC  LNPGS
0384:D0 01      82          BNE  TX2          ; KEEP GOING UNTIL 0
0386:60         83 EXIT     RTS
0387:EE 04 03   84 TX2      INC  PFROM
038A:EE 15 03   85          INC  STPAGE
038D:4C 78 03   86          JMP  LDLOOP
0390:           87 *
0390:           88 * HERE IS A UTILITY LINKAGE TO SOME MONITOR ROUTINES:
0390:           89 *
0390:           90 * LINK TO THE PRINT CHAR ROUTINE
0390:00         91 LINK     DFB  0
0391:AD 90 03   92 PCHR     LDA  LINK       ; PUT CHAR IN A
0394:4C ED FD   93          JMP  $FDED      ; MONITOR ENTRY POINT
0397:           94 * LINK TO THE PRINT HEX ROUTINE
0397:AD 90 03   95 PHEX     LDA  LINK       ; PUT BYTE IN A
039A:4C DA FD   96          JMP  $FDDA      ; GO DO IN MONITOR
039D:           97 *
039D:           98 * ROUTINE TO PUT OUT ONE HEX DIGIT
039D:AD 90 03   99 PHEXD    LDA  LINK       ; GET DIGIT
03A0:29 0F     100          AND  #$0F       ; DUMP HIGH PART
03A2:C9 0A     101          CMP  #$0A       ; SEE IF IN NUMBERIC
03A4:90 05     102          BCC  NONALF     ; IF SO DONT OFFSET
03A6:69 B6     103          ADC  #$B6       ; SHIFT IT UP
03A8:4C ED FD  104          JMP  $FDED      ; GO PRINT IT
03AB:69 B0     105 NONALF   ADC  #$B0       ; OFFSET FOR NUMBERIC
03AD:4C ED FD  106          JMP  $FDED      ; GO PRINT IT
03B0:          107 *
03B0:          108 *

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

?0316 DNLOAD	0386 EXIT	0312 FSLOT	034D FXFER
0378 LDLOOP	0313 LFROM	0390 LINK	0314 LNPGS
03AB NONALF	?0391 PCHR	0304 PFROM	?0397 PHEX
?039D PHEXD	0302 PM1	0305 PM2	0300 PMOVE
037B PPOINT	0307 PTO	0315 STPAGE	0375 TX1
0363 TX	0387 TX2	?0333 UPLOAD	

```
                                0300 PMOVE      0302 PM1        0304 PFROM      0305 PM2

 0307 PTO        0312 FSLOT      0313 LFROM      0314 LNPGS
 0315 STPAGE    ?0316 DNLOAD    ?0333 UPLOAD     034D FXFER
 0363 TX         0375 TX1        0378 LDLOOP     037B PPOINT
 0386 EXIT       0387 TX2        0390 LINK      ?0391 PCHR
?0397 PHEX      ?039D PHEXD      03AB NONALF
```

```
T

1  HIMEM: 32767
2  HOME : PRINT "FROB MONITOR PROGRAM"
3  PRINT : REM  VERSION 1.3 1/24/83
4  DIM DIG%(4),S%(32),BP%(3),PG%(18),BIT%(8)
5 FM% = 0
6  DIM OCODE%(255),ACODE%(255)
7 PM% = 768:BUF$ = "8":BUF = 8 * 4096
8  PRINT  CHR$ (4);"BLOAD PMOVE.OBJ"
10  INPUT "WHAT IS THE FROB SLOT NUMBER (1,7)?";FSLOT%
12  IF FSLOT% < 1 OR FSLOT% > 7 THEN 10
14 DEV = 4096 * 12 + 16 * (FSLOT% + 8)
20  INPUT "DO YOU WISH TO LOAD A FILE? (Y/N)";A$
22  IF A$ = "N" THEN 30
24  IF A$ < > "Y" THEN 20
26  GOSUB 11000: REM  GO LOAD A FILE
30  INPUT "DO YOU WISH TO USE FMON? (Y/N)";A$
32  IF A$ = "N" THEN 40
34  IF A$ < > "Y" THEN 30
36  GOSUB 12000: REM  GO LOAD FMON
40 BP% = 0: REM    SET TO 1 ON BREAK POINT HIT
60  GOSUB 4000: REM  LOAD THE SHADOW
100  IF FM% = 0 THEN 200
110  INPUT "PLEASE TURN ON VCS AND HIT RETURN";A$
130  IF  PEEK (DEV) < 128 THEN 110
140  POKE DEV + 1,0
150  IF  PEEK (DEV) < 128 THEN 110
160  PRINT "SETTING UP...";
200 OPCODES$ = "***ADCANDASLBCCBCSBEQBITBMIBNEBPLBRKBVCBVSCLCCLDCLICLVCMPCPXCPY"
201 OPCODES$ = OPCODES$ + "DECDEXDEYEORINCINXINYJMPJSRLDALDXLDYLSRNOPORAPHAPHPPLA"
202 OPCODES$ = OPCODES$ + "PLPROLRORRTIRTSSBCSECSEDSEISTASTXSTYTAXTAYTSXTXATXSTYA"
210  FOR I = 0 TO 255: READ OCODE%(I): NEXT I
211  DATA  11,35,0,0,0,35,3,0
212  DATA  37,35,3,0,0,35,3,0
213  DATA  10,35,0,0,0,35,3,0
214  DATA  14,35,0,0,0,35,3,0
215  DATA  29,2,0,0,7,2,40,0
216  DATA  39,2,40,0,7,2,40,0
217  DATA  8,2,0,0,0,2,40,0
218  DATA  45,2,0,0,0,2,40,0
219  DATA  42,24,0,0,0,24,33,0
220  DATA  36,24,33,0,28,24,33,0
221  DATA  12,24,0,0,0,24,33,0
222  DATA  16,24,0,0,0,24,33,0
223  DATA  43,1,0,0,0,1,41,0
224  DATA  38,1,41,0,28,1,41,0
225  DATA  13,1,0,0,0,1,41,0
226  DATA  47,1,0,0,0,1,41,0
227  DATA  0,48,0,0,50,48,49,0
228  DATA  23,0,54,0,50,48,49,0
229  DATA  4,48,0,0,50,48,49,0
230  DATA  56,48,55,0,0,48,0,0
231  DATA  32,30,31,0,32,30,31,0
232  DATA  52,30,51,0,32,30,31,0
233  DATA  5,30,0,0,32,30,31,0
234  DATA  17,30,53,0,32,30,31,0
235  DATA  20,18,0,0,20,18,21,0
236  DATA  27,18,22,0,20,18,21,0
237  DATA  9,18,0,0,0,18,21,0
238  DATA  15,18,0,0,0,18,21,0
239  DATA  19,44,0,0,19,44,25,0
240  DATA  26,44,34,0,19,44,25,0
241  DATA  6,44,0,0,0,44,25,0
242  DATA  46,44,0,0,0,44,25,0
```

```
250  FOR I = 0 TO 255: READ ACODEZ(I): NEXT I
260  DATA  1,10,1,1,1,3,3,1
261  DATA  1,5,2,1,1,4,4,1
262  DATA  13,11,1,1,1,6,6,1
263  DATA  1,9,1,1,1,8,8,1
264  DATA  4,10,1,1,3,3,3,1
265  DATA  1,5,2,1,4,4,4,1
266  DATA  13,11,1,1,1,6,6,1
267  DATA  1,9,1,1,1,8,8,1
268  DATA  1,10,1,1,1,3,3,1
269  DATA  1,5,2,1,4,4,4,1
270  DATA  13,11,1,1,1,6,6,1
271  DATA  1,9,1,1,1,8,8,1
272  DATA  1,10,1,1,1,3,3,1
273  DATA  1,5,2,1,12,4,4,1
274  DATA  13,11,1,1,1,6,6,1
275  DATA  1,9,1,1,1,8,8,1
276  DATA  1,10,1,1,3,3,3,1
277  DATA  1,1,1,1,4,4,4,1
278  DATA  13,11,1,1,6,6,7,1
279  DATA  1,9,1,1,1,8,1,1
280  DATA  5,10,5,1,3,3,3,1
281  DATA  1,5,1,1,4,4,4,1
282  DATA  13,11,1,1,6,6,7,1
283  DATA  1,9,1,1,8,8,9,1
284  DATA  5,10,1,1,3,3,3,1
285  DATA  1,5,1,1,4,4,4,1
286  DATA  13,11,1,1,1,6,6,1
287  DATA  1,9,1,1,1,8,8,1
288  DATA  5,10,1,1,3,3,3,1
289  DATA  1,5,1,1,4,4,4,1
290  DATA  13,11,1,1,1,6,6,1
291  DATA  1,9,1,1,1,8,8,1
390  REM  HERE IS THE SLEEP ROUTINE
391  REM  TO BE RUN IN PAGE 0 OF VCS
400  FOR I = 1 TO 18: READ PGZ(I): NEXT I
401  DATA   198,241,208,252
402  DATA   198,240,208,248
403  DATA   198,239,208,244
404  DATA   108,250,255,2,0,0
405  REM  E0: DEC $F1
406  REM  E2: BNE $E0
407  REM  E4: DEC $F0
408  REM  E6: BNE $E0
409  REM  E8: DEC $EF
410  REM  EA: BNE $E0
411  REM   EC: JMP ($FFFA)
412  REM  EF: DFB 2,0,0
420  PRINT "READY"
490 PC = 1000
495 SFZ = 0
500 FAILZ = 0
510  GOSUB 1000
515  IF A$ = "RESTORE" THEN 800
517  IF  LEN (D$) = 0 THEN D$ = "1"
520  IF  LEN (B$) > 4 THEN 900
540  IF  LEN (C$) > 0 THEN 600
550  IF  LEN (B$) = 0 THEN 500
570  GOTO 20000: REM  THE READ MEM ROUTINE
600  REM  DELIM FIELD NOT BLANK
610  IF C$ = "." THEN 20000: REM  READ MEMORY
620  IF C$ = "<" THEN 21000: REM  WRITE MEMORY
630  IF C$ = "M" THEN 22000: REM  MODIFY REGISTER
640  IF C$ = "R" THEN 23000: REM  DISPLAY REGISTERS
650  IF C$ = "P" THEN 24000: REM  PLACE BREAKPOINT
660  IF C$ = "K" THEN 25000: REM  KILL BREAK POINT
```

```
670  IF C$ = "H" THEN 26000: REM  HALT THE VCS
675  IF C$ = "S" THEN 31000: REM  START VCS AT RESET
680  IF C$ = "G" AND  LEN (B$) = 0 THEN 27000: REM   GO FROM BREAK
690  IF C$ = "G" THEN 28000: REM  GOTO ADDR
700  IF C$ = "L" AND  LEN (B$) = 0 THEN 29000: REM  CONTINUE DISASM
710  IF C$ = "L" THEN 30000: REM  DISASM
720  IF C$ = "I" THEN 32000: REM  LOAD FILE
750  GOTO 950
760  GOTO 500
800  REM  TURN ON THE I/O PORTS
810  POKE DEV,48
820  PRINT "COMMUNICATION IS NOW ON"
830  IF FM% = 0 THEN 500
840  PRINT "TURN THE VCS OFF AND ON TO GET TO FMON"
850  GOTO 500
900  PRINT "ERROR"
910  GOTO 500
920  PRINT "ERROR -- FMON NOT LOADED"
930  GOTO 500
950  PRINT "NOT A VALID COMMAND"
960  GOTO 500
1000  REM  ROUTINE TO RETURN STRING
1005 B$ = "": C$ = "": D$ = ""
1010  INPUT ">";A$
1015 K% = 1
1020  FOR L = 1 TO  LEN (A$)
1025 X$ =  MID$ (A$,L,1)
1027  IF X$ = " " THEN 1100
1030  GOSUB 2000: REM  IS THIS A HEX DIG?
1031  IF FLAG% > 0 AND K% = 2 THEN K% = 3
1032  IF FLAG% = 0 THEN K% = K% + 1
1033  IF K% > 3 THEN 1100
1035  IF K% = 1 THEN B$ = B$ + X$
1040  IF K% = 2 THEN C$ = C$ + X$
1050  IF K% = 3 THEN D$ = D$ + X$
1100  NEXT L
1110  RETURN
2000  REM  RETURN FLAG% SET IF X$ IS HEX
2005 FLAG% = 1
2010  IF X$ < "0" OR X$ > "F" THEN FLAG% = 0
2020  IF X$ > "9" AND X$ < "A" THEN FLAG% = 0
2100  RETURN
2500  REM   CHANGE B$ TO HEX ADDR
2505 ADDR = 0
2510  FOR L = 1 TO  LEN (B$)
2520 X% =  ASC ( MID$ (B$,L,1)) - 48
2540  IF X% > 9 THEN X% = X% - 7
2550 ADDR = ADDR * 16 + X%
2560  NEXT L
2570  RETURN
2630  GOTO 500
3000  REM  ROUTINE TO TALK TO FMON
3100  IF  PEEK (DEV) < 128 THEN 3100
3110  POKE DEV + 1,WR%
3120  RETURN
3200  REM  READ VALUE FROM FROB
3202 X% =  PEEK (DEV): IF X% > 127 THEN X% = X% - 128
3205  IF X% < 64 THEN 3202
3210 X% =  PEEK (DEV + 1)
3220  RETURN
3300  REM  ROUTINE TO GET REGISTERS
3310  GOSUB 3200: POKE BUF + 4086,X%
3311  GOSUB 3200: POKE BUF + 4085,X%
3312  GOSUB 3200: POKE BUF + 4087,X%
3313  GOSUB 3200: POKE BUF + 4088,X%
3314  GOSUB 3200: POKE BUF + 4089,X%
```

```
3320  FOR I = 1 TO 32
3330  GOSUB 3200:S%(I) = X%
3340  NEXT I
3345  GOSUB 3200
3350 N% = 0: IF  PEEK (BUF + 4085) = 128 THEN N% = 1
3360 Z% = 1: IF N% = 1 THEN Z% = 0
3370  IF  PEEK (BUF + 4085) = 1 THEN Z% = 0
3380  GOSUB 13000: REM  TAKE FLAGS APART
3385  POKE BUF + 4084,N% * 128 + BIT%(2) * 64 + BIT%(3) * 32 + BIT%(4) * 16 + BIT%(5) * 8 + BIT%(6) * 4 + Z% * 2 + BIT%(8)
3390  RETURN
3400  REM  ROUTINE TO RESTORE REGISTERS
3410  FOR I = 32 TO 1 STEP  - 1
3420 WR% = S%(I): GOSUB 3000
3430  NEXT I
3440  RETURN
4000  REM  LOAD SHADOW TO FROB
4010  POKE PM% + 18,FSLOT%
4020  POKE PM% + 19, INT (BUF / 256)
4030  POKE PM% + 20,16
4040  POKE PM% + 21,32
4050  CALL PM% + 22
4060  POKE DEV,48
4100  RETURN
5000  REM  CONVERT D$ TO LENGTH FIELD
5005  IF D$ = "1000" THEN 5200
5010 NUMBYT% = 0
5020  IF  LEN (D$) = 0 THEN  RETURN
5030  IF  LEN (D$) > 3 THEN D$ =  RIGHT$ (D$,3)
5040  FOR L = 1 TO  LEN (D$)
5050 X% =  ASC ( MID$ (D$,L,1)) - 48
5060  IF X% > 9 THEN X% = X% - 7
5070 NUMBYT% = NUMBYT% * 16 + X%
5080  NEXT L
5100  RETURN
5200 NUMBYT% = 4096: RETURN
6000  REM    PRINT LOWER 3 HEX DIGITS OF A1
6010 A2 =  INT (A1 / 256):A3 = A1 - (A2 * 256)
6020  POKE 912,A2
6030  CALL 925: REM  PRINT HEX DIGIT
6040  POKE 912,A3
6050  CALL 919: REM  PRINT HEX BYTE
6100  RETURN
6200  RETURN
6500  REM   PRINT 2 HEX DIGITS FROM A1
6510  POKE 912,A1
6520  CALL 919: REM  PRINT HEX DIGITS
6530  RETURN
7000  REM  FMON READ
7010 WR% = 16: GOSUB 3000
7020 WR% = ADDR / 256: GOSUB 3000
7030 WR% = ADDR - (WR% * 256): GOSUB 3000
7040 WR% = 1: GOSUB 3000
7050  GOSUB 3200
7060 A1 = X%
7070  RETURN
8000  REM  DO AN FMON WRITE
8010 WR% = 32: GOSUB 3000
8020 WR% = ADDR / 256: GOSUB 3000
8030 WR% = ADDR - (WR% * 256): GOSUB 3000
8040 WR% = 1: GOSUB 3000
8050 WR% = A1: GOSUB 3000
8100  RETURN
9000  REM  ROUTINE TO PUT TO SLEEP
9010 WR% = 32: GOSUB 3000
9020 WR% = 0: GOSUB 3000
9030 WR% = 224: GOSUB 3000
```

```
9040 WRZ = 18: GOSUB 3000
9045  FOR I = 1 TO 18
9050 WRZ = PGZ(I): GOSUB 3000
9060  NEXT I
9062 WRZ = 64: GOSUB 3000
9064 WRZ = 0: GOSUB 3000
9066 WRZ = 224: GOSUB 3000
9080  GOSUB 4000: REM  LOAD SHADOW
9100  RETURN
11000  REM   LOAD A FILE
11010  INPUT "WHAT IS THE FILE NAME?";A$
11020  INPUT "LOAD TO WHAT VCS HEX ADDRESS?";B$
11030  IF  LEN (B$) <  > 4 THEN 11020
11031 X$ =  LEFT$ (B$,1)
11032  IF X$ = "2" OR X$ = "4" OR X$ = "6" OR X$ = "0" OR X$ = "8" OR X$ = "A" OR X$ = "C" OR X$ = "E" THEN 11020
11035 OKZ = 1
11040  FOR L = 1 TO 4
11050 X$ =  MID$ (B$,L,1)
11060  GOSUB 2000: REM  SEE IF A HEX DIGIT
11070  IF FLAGZ = 0 THEN OKZ = 0
11080  NEXT L
11090  IF OKZ = 0 THEN 11020
11100  PRINT  CHR$ (4);"BLOAD ";A$;",A$";BUF$; RIGHT$ (B$,3)
11110  RETURN
12000  REM : LOAD FMON
12010  PRINT  CHR$ (4);"BLOAD FMON.OBJ,A$";BUF$;"F00"
12020 FMZ = 1
12030  RETURN
13000  REM  CONVERT XZ TO BITS
13005 X1Z = XZ:XBZ = 256
13010  FOR I = 1 TO 8
13020 BITZ(I) = 0:XBZ = XBZ / 2
13030  IF X1Z < XBZ THEN 13060
13040 BITZ(I) = 1
13050 X1Z = X1Z - XBZ
13060  NEXT I
13070  RETURN
14000  REM  PRINT DADDR AND ":"
14010 X = DADDR: GOSUB 14300
14040  PRINT ":";
14050  RETURN
14100  REM  PRINT 2 HEX FROM XZ
14110 A1 = XZ
14120  GOTO 6500
14200  REM  PRINT OPCODE
14210 NZ = OCODEZ(OPZ) * 3 + 1
14220  PRINT  MID$ (OPCODES$,NZ,3);
14230  RETURN
14300  REM  PRINT DADDR RELATIVE
14310  PRINT FHR$;
14320 A1 = X: GOSUB 6000
14330  RETURN
20000  REM  READ MEMORY
20002 RDZ = 1: REM  FOLLOW THE READ PATH
20005  GOSUB 5000: REM  CONVERT D$ TO NUMBER
20010  IF  LEN (B$) = 4 THEN 20050
20020  FOR L = 1 TO 4 -  LEN (B$)
20030 B$ = "0" + B$
20040  NEXT L
20050 X$ =  LEFT$ (B$,1)
20055 FHR$ = X$
20060  IF X$ = "0" OR X$ = "2" OR X$ = "4" OR X$ = "6" OR X$ = "8" OR X$ = "A" OR X$ = "C" OR X$ = "E" THEN 20100
20070 B$ = BUF$ +  RIGHT$ (B$,3)
20080  GOTO 20200
20100 B$ = "0" +  RIGHT$ (B$,3)
20110  IF FMZ = 0 THEN 920
```

```
20200  GOSUB 2500: REM  CONVERT ADDRESS
20201  IF RD% = 0 THEN 21100: REM  DO WRITE
20203  PRINT
20205  PRINT FHR$;
20210 A1 = ADDR: GOSUB 6000
20212  PRINT ":";
20220  IF ADDR > 4095 THEN A1 =  PEEK (ADDR)
20225  IF ADDR < 256 THEN 20600
20230  IF ADDR < 4096 THEN  GOSUB 7000: REM     GO DO FMON READ
20240  GOSUB 6500
20250  PRINT " ";
20260 ADDR = ADDR + 1:NUMBYT% = NUMBYT% - 1
20270  IF NUMBYT% = 0 THEN 20500
20280 B% = ADDR - ( INT (ADDR / 8) * 8)
20290  IF B% = 0 THEN 20203
20300  GOTO 20220
20500  PRINT : PRINT
20510  GOTO 500
20600  IF ADDR < 224 THEN 20230
20610 A1 = S%(ADDR - 223)
20620  GOTO 20240
21000  REM  WRITE MEMORY
21010 RD% = 0: REM  CHOOSE WRITE PATH
21020  GOTO 20005
21100  IF NUMBYT% > 255 THEN 900
21110 A1 = NUMBYT%
21120  IF ADDR > 4095 THEN  POKE ADDR,A1
21125  IF ADDR < 256 THEN 21200
21130  IF ADDR < 4096 THEN  GOSUB 8000
21140 NUMBYT% = 1
21150  GOTO 20203
21200  IF ADDR < 224 THEN 21130
21210 S%(ADDR - 223) = A1
21220  GOTO 21140
22000  REM  MODIFY REGISTER
22005  IF FM% = 0 THEN 920
22010  INPUT "WHICH REGISTER? (A,X,Y,S,P)";A$
22020  IF A$ < > "A" AND A$ < > "X" AND A$ < > "Y" AND A$ < > "S" AND A$ < > "P" THEN 22010
22030  INPUT "NEW VALUE (HEX)=";D$
22040  IF  LEN (D$) > 2 THEN 22030
22050 A1 = 0:FLAG% = 1
22060  FOR I = 1 TO  LEN (D$)
22065  IF FLAG% = 0 THEN 22100
22070 X$ =  MID$ (D$,I,1): GOSUB 2000
22080 X% =  ASC (X$) - 48: IF X% > 9 THEN X% = X% - 7
22090 A1 = A1 * 16 + X%
22100  NEXT I
22110  IF FLAG% = 0 THEN 22030
22120  IF A$ < > "A" THEN 22130
22125  POKE BUF + 4086,A1: GOTO 500
22130  IF A$ < > "X" THEN 22140
22135  POKE BUF + 4087,A1: GOTO 500
22140  IF A$ < > "Y" THEN 22150
22145  POKE BUF + 4088,A1: GOTO 500
22150  IF A$ < > "S" THEN 22160
22152 SF% = 1: REM  SHOW THE STACK POINTER AS UPDATED
22155  POKE BUF + 4089,A1: GOTO 500
22160 F% = A1:F% = F% / 2
22165 Z% = F% - ( INT (F% / 2) * 2)
22170 N% =  INT (F% / 64)
22175  IF Z% = 0 OR N% = 0 THEN 22200
22180  PRINT "??? EITHER Z OR N OR BOTH MUST BE 0"
22190  GOTO 22030
22200  POKE BUF + 4084,A1
22210  IF Z% = 0 THEN 22250
22220  POKE BUF + 4085,255
```

```
22230  GOTO 500
22250  IF NZ THEN 22280
22260  POKE BUF + 4085,1
22270  GOTO 500
22280  POKE BUF + 4085,128
22290  GOTO 500
23000  REM  DISPLAY REGISTERS
23005  IF FMZ = 0 THEN 920
23010  PRINT
23020  PRINT "A=";:A1 =  PEEK (BUF + 4086): GOSUB 6500
23030  PRINT " X=";:A1 =  PEEK (BUF + 4087): GOSUB 6500
23040  PRINT " Y=";:A1 =  PEEK (BUF + 4088): GOSUB 6500
23050  PRINT " S=";:A1 =  PEEK (BUF + 4089): GOSUB 6500
23060  PRINT " P=";:A1 =  PEEK (BUF + 4084): GOSUB 6500
23070  PRINT : GOTO 500
24000  REM  PLACE BREAK POINT
24005  IF FMZ = 0 THEN 920
24010  IF  LEN (B$) < > 4 THEN 950
24020 B$ = BUF$ +  RIGHT$ (B$,3)
24030  GOSUB 2500
24040 BREAK = ADDR
24050  FOR I = 1 TO 3
24060 BPZ(I) =  PEEK (ADDR + I - 1)
24070  NEXT I
24075  REM  STICK IN JUMP TO FF00
24080  POKE ADDR,76: POKE ADDR + 1,00: POKE ADDR + 2,255
24090  GOTO 500
25000  REM  KILL BREAK POINT
25005  IF A$ < > "K" THEN 950
25010  IF BREAK = 0 THEN 500
25020  FOR I = 1 TO 3
25030  POKE BREAK + I - 1,BPZ(I)
25040  NEXT I
25050 BREAK = 0: GOTO 500
26000  REM  HALT THE VCS
26005  IF A$ < > "H" THEN 950
26010  POKE DEV,0
26020  PRINT "FROB MEMORY NOW UNDER APPLE CONTROL"
26030  GOTO 500
27000  REM   GO FROM BREAK
27002  IF FMZ = 0 THEN 920
27003  IF PC = 1000 THEN 27200
27005  IF PC = BREAK - BUF + 61440 THEN 27100
27006  INPUT "DO YOU WISH TO WAIT FOR A BREAK? (Y/N)";A$
27007  IF A$ < > "Y" AND A$ < > "N" THEN 27006
27010  POKE BUF + 3990,PC - ( INT (PC / 256) * 256)
27015  POKE BUF + 3991, INT (PC / 256)
27020  GOSUB 9000: REM  SLEEP AND SET UP
27030 WRZ = 48: GOSUB 3000
27040  GOSUB 3400: REM  RESTORE REGISTERS
27042  IF A$ = "N" THEN 500
27045  PRINT "WAITING FOR BREAK POINT"
27050  GOSUB 3300: REM   WAIT FOR BREAK
27052  PRINT "BREAK POINT ENCOUNTERED"
27053 SFZ = 1: REM  MARK STACK POINTER AS UPDATED
27055 PC = BREAK - BUF + 61440
27060  GOTO 25000
27100  PRINT "ERROR -- YOU MUST MOVE THE BREAKPOINT"
27110  PRINT "BEFORE YOU CONTINUE"
27120  GOTO 500
27200  PRINT "ERROR -- NO BREAK POINT FROM"
27210  PRINT "WHICH TO PROCEED"
27220  GOTO 500
28000  REM  GOTO ADDR
28005  IF SFZ > 0 THEN 28010
28007  PRINT "WARNING -- STACK POINTER SET AT ";
```

```
28008 A1 =  PEEK (BUF + 4089): GOSUB 6500: PRINT
28010  GOSUB 2500
28020 PC = ADDR
28030  GOTO 27006
29000  REM  CONTINUE DISASM
29005 DLIM = DADDR + 30
29010 OPZ =  PEEK (DADDR)
29015  GOSUB 14000: REM   PRINT DADDR:
29016 DADDR = DADDR + 1
29017 XZ = OPZ: GOSUB 14100: PRINT ' ';
29020  ON ACODEZ(OPZ) GOTO 29100,29150,29200,29250,29300,29350,29400,29450,29500,29550,29600,29650,29700
29100  REM  IMPLIED (NO ADDRESS)
29110  PRINT '      ';
29120  GOSUB 14200: REM  PRINT OP STRING
29130  GOTO 29900
29150  REM  ACCUMULATOR
29155  PRINT '      ';
29160  GOSUB 14200
29165  PRINT ' A';
29170  GOTO 29900
29200  REM  PAGE 0 ADDRESS
29205 X1Z =  PEEK (DADDR):DADDR = DADDR + 1
29210 XZ = X1Z: GOSUB 14100: PRINT ' ';
29215  PRINT '   ';
29220  GOSUB 14200: PRINT ' $';
29225 XZ = X1Z: GOSUB 14100
29230  GOTO 29900
29250  REM  16 BIT ABSOLUTE
29252 X1Z =  PEEK (DADDR):DADDR = DADDR + 1
29254 XZ = X1Z: GOSUB 14100: PRINT ' ';
29256 X2Z =  PEEK (DADDR):DADDR = DADDR + 1
29258 XZ = X2Z: GOSUB 14100: PRINT ' ';
29260  GOSUB 14200: PRINT ' $';
29262 XZ = X2Z: GOSUB 14100
29264 XZ = X1Z: GOSUB 14100
29270  GOTO 29900
29300  REM  IMMEDIATE
29305 X1Z =  PEEK (DADDR):DADDR = DADDR + 1
29310 XZ = X1Z: GOSUB 14100: PRINT ' ';
29315  PRINT '   ';
29320  GOSUB 14200: PRINT ' #$';
29325 XZ = X1Z: GOSUB 14100
29330  GOTO 29900
29350  REM  ZERO PAGE INDEXED X
29355 X1Z =  PEEK (DADDR):DADDR = DADDR + 1
29360 XZ = X1Z: GOSUB 14100: PRINT '    ';
29365  GOSUB 14200: PRINT ' $';
29370 XZ = X1Z: GOSUB 14100: PRINT ',X';
29375  GOTO 29900
29400  REM  ZERO PAGE INDEXED Y
29405 X1Z =  PEEK (DADDR):DADDR = DADDR + 1
29410 XZ = X1Z: GOSUB 14100: PRINT '    ';
29415  GOSUB 14200: PRINT ' $';
29420 XZ = X1Z: GOSUB 14100: PRINT ',Y';
29425  GOTO 29900
29450  REM  ABSOLUTE INDEXED X
29455 X1Z =  PEEK (DADDR):DADDR = DADDR + 1
29460 XZ = X1Z: GOSUB 14100: PRINT ' ';
29465 X2Z =  PEEK (DADDR):DADDR = DADDR + 1
29470 XZ = X2Z: GOSUB 14100: PRINT ' ';
29475  GOSUB 14200: PRINT ' $';
29480 XZ = X2Z: GOSUB 14100
29485 XZ = X1Z: GOSUB 14100
29490  PRINT ',X';: GOTO 29900
29500  REM  ABSOLUTE INDEXED Y
29505 X1Z =  PEEK (DADDR):DADDR = DADDR + 1
```

```
29510 X% = X1%: GOSUB 14100: PRINT " ";
29515 X2% =  PEEK (DADDR):DADDR = DADDR + 1
29520 X% = X2%: GOSUB 14100: PRINT " ";
29525  GOSUB 14200: PRINT " $";
29530 X% = X2%: GOSUB 14100
29535 X% = X1%: GOSUB 14100
29540  PRINT ",Y";: GOTO 29900
29550  REM  INDEXED INDIRECT X
29555 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29560 X% = X1%: GOSUB 14100: PRINT "    ";
29565  GOSUB 14200: PRINT " ($";
29570 X% = X1%: GOSUB 14100: PRINT ",X)";
29575  GOTO 29900
29600  REM  INDIRECT INDEXED Y
29605 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29610 X% = X1%: GOSUB 14100: PRINT "    ";
29615  GOSUB 14200: PRINT " ($";
29620 X% = X1%: GOSUB 14100: PRINT "),Y";
29625  GOTO 29900
29650  REM  JUMP INDIRECT
29655 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29660 X% = X1%: GOSUB 14100: PRINT " ";
29661 X2% =  PEEK (DADDR):DADDR = DADDR + 1
29662 X% = X2%: GOSUB 14100: PRINT " ";
29665  GOSUB 14200: PRINT " ($";
29670 X% = X2%: GOSUB 14100
29671 X% = X1%: GOSUB 14100: PRINT ")";
29675  GOTO 29900
29700  REM  RELATIVE BRANCH
29705 X1% =  PEEK (DADDR):DADDR = DADDR + 1
29710 X% = X1%: GOSUB 14100: PRINT "    ";
29715  GOSUB 14200: PRINT " $";
29720  IF X1% < 128 THEN 29750
29730 X1% =  - 1 * (256 - X1%)
29750 X = DADDR + X1%
29760  GOSUB 14300: REM  PRINT 16 BITS SPECIAL FIELD
29900  PRINT
29910  IF DADDR < DLIM THEN 29010
29920  GOTO 500
30000  REM  DISASSEMBLE
30010  IF  LEN (B$) = 4 THEN 30020
30012  PRINT "ERROR -- ADDRESS MUST POINT TO ROM"
30015  GOTO 500
30020  IF  LEN (D$) = 0 THEN D$ = "1"
30025  GOSUB 5000: REM  CONVERT D$
30030 X$ =  LEFT$ (B$,1)
30040  IF X$ = "0" OR X$ = "2" OR X$ = "4" OR X$ = "6" OR X$ = "8" OR X$ = "A" OR X$ = "C" OR X$ = "E" THEN 30012
30050 FHR$ = X$
30060 B$ = BUF$ +  RIGHT$ (B$,3)
30070  GOSUB 2500:DADDR = ADDR
30080 DLIM = DADDR + NUMBYT%
30090  GOTO 29010
31000  REM  START THE VCS
31003  IF A$ < > "S" THEN 950
31005  GOSUB 4000: REM  RELOAD SHADOW
31010  POKE DEV,16
31020  PRINT "FROB MEMORY NOW UNDER VCS CONTROL"
31025  IF FM% = 0 THEN 31030
31027  PRINT "(COMMUNICATION WITH FMON IS TURNED OFF)"
31030  PRINT "PLEASE POWER UP THE VCS TO START IT"
31040  GOTO 500
32000  REM  LOAD A FILE
32010  IF B$ = "F" AND D$ = "LE" THEN 32020
32015  GOTO 750
32020  GOSUB 11000
32030 FM% = 0
```

```
32040  INPUT "DO YOU WISH TO USE FMON? (Y/N)";A$
32050  IF A$ = 'N' THEN 500
32060  IF A$ < > 'Y' THEN 32030
32070  GOSUB 12000: REM  LOAD FMON
32080 BP% = 0
32090  GOSUB 4000: REM  LOAD SHADOW
32100  INPUT "PLEASE TURN ON THE VCS AND HIT RETURN";A$
32110  GOTO 500

]
```