Appendix 1

The program listing of the EPROM

The monitor, the editor and the assembler

The next ten pages contain a survey of the EPROM contents. It is the 'expanded' version of Appendix 3 in Book 1, which showed the contents in the form of a hex dump (in bytes only).
The listing includes the following:

1. A survey of all the RAM memory locations in page ØØ (temporaries) and in page 1A (PIA addressing and locations for the NMI and IRQ jump vector).
2. The main monitor routine (1CØØ . . . 1CB4).
3. The main editor routine (1CB5 . . . 1D4C).
4. Subroutines as part of:
   a. the editor: SCAN (1D5C . . . 1D6E);
   b. the editor and the BRANCH routine: GETBYT (1D6F . . . 1D87);
   c. the editor and the monitor: SCAND(S) (1D88 . . . 1E1F, including SHOW, CONVD and GETKEY);
   d. the editor: RDINST (1E2Ø . . . 1E46) and FILLWS (1E47 . . . 1E5B);
   e. the editor and the assembler: OPLEN/LENACC (1E5C . . . 1E82);
   f. the editor and the assembler: UP (1E83 . . . 1EA5);
   g. the editor: ADCEND (1EDC . . . 1ED2);
   h. the editor and the assembler: BEGIN;
   i. the editor: ADCEND (1EDC . . . 1EE9);
   j. the editor and the assembler: RECEND (1EEA . . . 1EF7) and NEXT (1EF8 . . . 1FØE).
5. Look-up table LOOK, used by the monitor and the editor (subroutine CONVD) (1FØF . . . 1F1E).
6. Look-up table LEN, used by the editor and the assembler (subroutine OPLEN/LENACC (1F1F . . . 1F2E).
7. Subroutine GETLBL, used in the assembler (1F35 . . . 1F5Ø).
8. The main assembler routine (1F51 . . . 1FD2).
9. The displacement calculation routine BRANCH (1FD5 . . . 1FF7).
10. Six EPROM locations to establish the NMI, RES and IRQ vectors (1FFA . . . 1FFF) and six locations for the two JMP-IND instructions involved (see chapter 3 in Book 1) (1F2F . . . 1F34).
    N.B. Locations 1FF8 and 1FF9 are not used. When the EPROM is programmed they are filled with FF.
11. All the labels and names of memory locations that are used in the monitor, the editor and the assembler, in alphabetical order, plus the corresponding address.

```
0000: 1C00            LOYS   ORG   $1C00  VERSION D
0010:
0020:
0030:
0040:
0050:                 SOURCE LISTING OF ELEKTOR'S JUNIOR COMPUTER
0060:
0070:                 WRITTEN BY A. NACHTMANN
0080:
0090:                 DATE: 7 FEB. 1980
0100:
0110:                 THE FEATURES OF JUNIOR'S MONITOR ARE:
0120:
0130:                 HEX ADDRESS DATA DISPLAY (ENTRY VIA RST)
0140:                 HEX EDITOR (START ADDRESS $1CB5)
0150                  HEX ASSEMBLER (START ADDRESS $1F51)
0160:
0170:                 EDITOR'S POINTERS AND TEMPS IN PAGE ZERO
0180:
0190: 1C00            KEY    *      $00E1
0200: 1C00            BEGADL *      $00E2  BEGIN ADDRESS POINTER
0210: 1C00            BEGADH *      $00E3
0220: 1C00            ENDADL *      $00E4  END ADDRESS POINTER
0230: 1C00            ENDADH *      $00E5
0240: 1C00            CURADL *      $00E6  CURRENT ADDRESS POINTER
0250: 1C00            CURADH *      $00E7
0260: 1C00            CENDL  *      $00E8  CURRENT ADDRESS POINTER
0270: 1C00            CENDH  *      $00E9
0280: 1C00            MOVADL *      $00EA
0290: 1C00            MOVADH *      $00EB
0300: 1C00            TABLEL *      $00EC
0310: 1C00            TABLEH *      $00ED
0320: 1C00            LABELS *      $00EE
0330: 1C00            BYTES  *      $00F6  NUMBER OF BYTES TO BE DISPLAYED
0340: 1C00            COUNT  *      $00F7
0350:
0360:                 MPU REGISTERS IN PAGE ZERO
0370:
0380: 1C00            PCL    *      $00EF
0390: 1C00            PCH    *      $00F0
0400: 1C00            PREG   *      $00F1 = FLAGS
0410: 1C00            SPUSER *      $00F2
0420: 1C00            ACC    *      $00F3
0430: 1C00            YREG   *      $00F4
0440: 1C00            XREG   *      $00F5
0450
0460:                 HEX DISPLAY BUFFERS IN PAGE ZERO
0470:
0480: 1C00            INL    *      $00F8
0490: 1C00            INH    *      $00F9
0500: 1C00            POINTL *      $00FA
0510: 1C00            POINTH *      $00FB
0520:
0530:                 TEMPORARY DATA BUFFERS IN PAGE ZERO
0540:
0550: 1C00            TEMP   *      $00FC
0560: 1C00            TEMPX  *      $00FD
0570: 1C00            NIBBLE *      $00FE
0580: 1C00            MODE   *      $00FF  (0 = DA MODE, ≠0 = AD MODE)
0590:
0600:                 MEMORY LOCATIONS IN THE 6532-IC
0610:
0620: 1C00            PAD    *      $1A80  DATA REGISTER OF PORT A
0630: 1C00            PADD   *      $1A81  DATA DIRECTION REGISTER OF PORT A
0640: 1C00            PBD    *      $1A82  DATA REGISTER OF PORT B
0650: 1C00            PBDD   *      $1A83  DATA DIRECTION REGISTER OF PORT B
0660:
0670:                 WRITE EDGE DETECT CONTROL
0680:
0690: 1C00            EDETA  *      $1AE4  NEG EDET DISABLE PA7-IRQ
0700: 1C00            EDETB  *      $1AE5  POS EDET DISABLE PA7-IRQ
0710: 1C00            WDETC  *      $1AE6  NEG EDET ENABLE PA7-IRQ
0720: 1C00            EDETD  *      $1AE7  POS EDET ENABLE PA7-IRQ
0730:
0740:                 READ FLAG REGISTER AND CLEAR TIMER & IRQ FLAG
0750:
0760: 1C00            RDFLAG *      $1AD5  BIT6=PA7-FLAG; BIT7=TIMER-FLAG
0770:
0780:                 WRITE COUNT INTO TIMER, DISABLE TIMER-IRQ
0790:
0800: 1C00            CNTA   *      $1AF4  CLK1T
0810: 1C00            CNTB   *      $1AF5  CLK8T
0820: 1C00            CNTC   *      $1AF6  CLK64T
0830: 1C00            CNTD   *      $1AF7  CLK1KT
0840:
0850:                 WRITE COUNT INTO TIMER, ENABLE TIMER-IRQ
0860:
0870: 1C00            CNTE   *      $1AFC  CLK1T
0880: 1C00            CNTF   *      $1AFD  CLK8T
0890: 1C00            CNTG   *      $1AFE  CLK64T
0900: 1C00            CNTH   *      $1AFF  CLK1KT
0910:
0920:                 INTERRUPT VECTORS: IRQ & NMI VECTORS SHOULD BE
0930:                 LOADED IN THE FOLLOWING MEMORY LOCATIONS FOR
0940:                 PROPER SYSTEM OPERATION.
```

```
0950:
0960: 1C00            NMIL    *       $1A7A   NMI LOWER BYTE
0970: 1C00            NMIH    *       $1A7B   NMI HIGHER BYTE
0980: 1C00            IRQL    *       $1A7E   IRQ LOWER BYTE
0990: 1C00            IRQH    *       $1A7F   IRQ HIGHER BYTE
1000:
1010:                 BEGINNERS MAY LOAD INTO THESE LOCATIONS
1020:                 $1C00 FOR STEP BY STEP MODUS AND BRK COMMAND
1030:
1040:
1050:
1060:                 JUNIOR'S MAINROUTINES
1070:
1080: 1C00 85 F3      SAVE    STAZ  ACC     SAVE ACCU
1090: 1C02 68                 PLA           GET CURRENT P-REGISTER
1100: 1C03 85 F1              STAZ  PREG    SAVE P-REGISTER
1110: 1C05 68         SAVEA   PLA           GET CURRENT PCL
1120: 1C06 85 EF              STAZ  PCL     SAVE CURRENT PCL
1130: 1C08 85 FA              STAZ  POINTL  PCL TO DISPLAY BUFFER
1140: 1C0A 68                 PLA           GET CURRENT PCH
1150: 1C0B 85 F0              STAZ  PCH     SAVE CURRENT PCH
1160: 1C0D 85 FB              STAZ  POINTH  PCH TO DISPLAY BUFFER
1170: 1C0F 84 F4      SAVEB   STYZ  YREG    SAVE CURRENT Y-REGISTER
1180: 1C11 86 F5              STXZ  XREG    SAVE CURRENT X-REGISTER
1190: 1C13 BA                 TSX           GET CURRENT SP
1200: 1C14 86 F2              STX   SPUSER  SAVE CURRENT SP
1210: 1C16 A2 01              LDXIM $01     SET AD-MODE
1220: 1C18 86 FF              STXZ  MODE
1230: 1C1A 4C 33 1C           JMP   START
1240:
1250: 1C1D A9 1E      RESET   LDAIM $1E     PB1---PB4
1260: 1C1F 8D 83 1A           STA   PBDD     IS OUTPUT
1270: 1C22 A9 04              LDAIM $04     RESET P-REGISTER
1280: 1C24 85 F1              STAZ  PREG
1290: 1C26 A9 03              LDAIM $03
1300: 1C28 85 FF              STAZ  MODE     SET AD-MODE
1310: 1C2A 85 F6              STAZ  BYTES   DISPLAY POINTH,POINTL,INH
1320: 1C2C A2 FF              LDXIM $FF     ADJUST THE STACKPOINTER
1330: 1C2E 9A                 TXS
1340: 1C2F 86 F2              STXZ  SPUSER
1350: 1C31 D8                 CLD
1360: 1C32 78                 SEI
1370:
1380: 1C33 20 88 1D   START   JSR   SCAND   DISPLAY DATA SPECIFIED BY POINTH,POINTL
1390: 1C36 D0 FB              BNE   START   WAIT UNTIL KEY IS RELEASED
1400: 1C38 20 88 1D   STARA   JSR   SCAND   DISPLAY DATA SPECIFIED BY POINT
1410: 1C3B F0 FB              BEQ   STARA   ANY KEY DEPRESSED
1420: 1C3D 20 88 1D           JSR   SCAND   DEBOUNCE KEY
1430: 1C40 F0 F6              BEQ   STARA   ANY KEY STILL DEPRESSED
1440: 1C42 20 F9 1D           JSR   GETKEY  IF YES , DECODE KEY,RETURN WITH KEY IN ACCU
1450:
1460: 1C45 C9 13      GOEXEC  CMPIM $13     GO-KEY?
1470: 1C47 D0 13              BNE   ADMODE
1480: 1C49 A6 F2              LDXZ  SPUSER  GET CURRENT SP
1490: 1C4B 9A                 TXS
1500: 1C4C A5 FB              LDAZ  POINTH  START EXECUTION AT POINTH,POINTL
1510: 1C4E 48                 PHA
1520: 1C4F A5 FA              LDAZ  POINTL
1530: 1C51 48                 PHA
1540: 1C52 A5 F1              LDAZ  PREG    RESTORE CURRENT P REGISTER
1550: 1C54 48                 PHA
1560: 1C55 A6 F5              LDXZ  XREG
1570: 1C57 A4 F4              LDYZ  YREG
1580: 1C59 A5 F3              LDAZ  ACC
1590: 1C5B 40                 RTI           EXECUTE PROGRAM
1600: 1C5C C9 10      ADMODE  CMPIM $10     AD-KEY?
1610: 1C5E D0 06              BNE   DAMODE
1620: 1C60 A9 03              LDAIM $03     SET AD-MODE
1630: 1C62 85 FF              STAZ  MODE
1640: 1C64 D0 14              BNE   STEPA   always
1650:
1660: 1C66 C9 11      DAMODE  CMPIM $11     DA-KEY?
1670: 1C68 D0 06              BNE   STEP
1680: 1C6A A9 00              LDAIM $00     SET DA-MODE
1690: 1C6C 85 FF              STAZ  MODE
1700: 1C6E F0 0A              BEQ   STEPA   always
1710:
1720: 1C70 C9 12      STEP    CMPIM $12     PLUS-KEY?
1730: 1C72 D0 09              BNE   PCKEY
1740: 1C74 E6 FA              INCZ  POINTL
1750: 1C76 D0 02              BNE   STEPA
1760: 1C78 E6 FB              INCZ  POINTH
1770: 1C7A 4C 33 1C   STEPA   JMP   START
1780:
1790: 1C7D C9 14      PCKEY   CMPIM $14     PC-KEY?
1800: 1C7F D0 0B              BNE   ILLKEY
1810: 1C81 A5 EF              LDAZ  PCL
1820: 1C83 85 FA              STAZ  POINTL  LAST PC TO DISPLAY BUFFER
1830: 1C85 A5 F0              LDAZ  PCH
1840: 1C87 85 FB              STAZ  POINTH
1850: 1C89 4C 7A 1C           JMP   STEPA
1860:
1870: 1C8C C9 15      ILLKEY  CMPIM $15     ILLEGAL KEY?
1880: 1C8E 10 EA              BPL   STEPA   IF YES,IGNORE IT
1890:
```

```
1900: 1C90 85 E1     DATA   STAZ  KEY      SAVE KEY
1910: 1C92 A4 FF            LDYZ  MODE     Y=0 IS DATA MODE,ELSE ADDRESS MODE
1920: 1C94 D0 0D            BNE   ADDRES
1930: 1C96 B1 FA            LDAIY POINTL GET DATA SPECIFIED
1940: 1C98 0A               ASLA         BY POINT
1950: 1C99 0A               ASLA         SHIFT LOW ORDER
1960: 1C9A 0A               ASLA         NIBBLE INTO HIGH ORDER NIBBLE
1970: 1C9B 0A               ASLA
1980: 1C9C 05 E1            ORAZ  KEY      DATA WITH KEY
1990: 1C9E 91 FA            STAIY POINTL RESTORE DATA
2000: 1CA0 4C 7A 1C         JMP   STEPA
2010:
2020: 1CA3 A2 04     ADDRES LDXIM $04      4 SHIFTS
2030: 1CA5 06 FA     ADLOOP ASLZ  POINTL   POINTH,POINTL 4 POSITIONS TO LEFT
2040: 1CA7 26 FB            ROLZ  POINTH
2050: 1CA9 CA               DEX
2060: 1CAA D0 F9            BNE   ADLOOP
2070: 1CAC A5 FA            LDAZ  POINTL
2080: 1CAE 05 E1            ORAZ  KEY      RESTORE ADDRESS
2090: 1CB0 85 FA            STAZ  POINTL
2100: 1CB2 4C 7A 1C         JMP   STEPA
2110:
2120:
2130:
2140:
2150:
2160:                JUNIOR'S HEX EDITOR
2170:
2180:                FOLLOWING COMMANDS ARE VALID:
2190:
2200:                "INSERT": INSERT A NEW LINE JUST BEFORE DISPLAYED LINE
2210:
2220:                "INPUT": INSERT A NEW LINE JUST BEHIND THE
2230:                          DISPLAYED LINE
2240:
2250:                "SEARCH": SEARCH IN WORKSPACE FOR A GIVEN 2BYTE PATTERN
2260:
2270:                "SKIP": SKIP TO NEXT INSTRUCTION
2280:
2290:                "DELETE": DELETE CURRENT DISPLAYED INSTRUCTION
2300:
2310:                AN ERROR IS INDICATED, IF THE INSTRUCTION POINTER
2320:                CURAD IS OUT OF RANGE
2330:
2340: 1CB5 20 D3 1E  EDITOR JSR   BEGIN  CURAD:=BEGAD
2350: 1CB8 A4 E3            LDYZ  BEGADH
2360: 1CBA A6 E2            LDXZ  BEGADL
2370: 1CBC E8               INX
2380: 1CBD D0 01            BNE   EDIT
2390: 1CBF C8               INY
2400: 1CC0 86 E8     EDIT   STXZ  CENDL  CEND:=BEGAD+1
2410: 1CC2 84 E9            STYZ  CENDH
2420: 1CC4 A9 77            LDAIM $77     DISPLAY "77"
2430: 1CC6 A0 00            LDYIM $00
2440: 1CC8 91 E6            STAIY CURADL
2450:
2460: 1CCA 20 4D 1D  CMND   JSR   SCAN   DISPLAY CURRENT INSTRUCTION,WAIT FOR A KEY
2470:
2480: 1CCD C9 14     SEARCH CMPIM $14     SEARCH COMMAND?
2490: 1CCF D0 2A            BNE   INSERT
2500: 1CD1 20 6F 1D         JSR   GETBYT READ 1ST BYTE
2510: 1CD4 10 F7            BPL   SEARCH COM. KEY?
2520: 1CD6 85 FB            STAZ  POINTH DISCARD DATA
2530: 1CD8 20 6F 1D         JSR   GETBYT READ 2ND BYTE
2540: 1CDB 10 F0            BPL   SEARCH COM. KEY?
2550: 1CDD 85 FA            STAZ  POINTL DISCARD DATA
2560: 1CDF 20 D3 1E         JSR   BEGIN  CURAD:=BEGAD
2570: 1CE2 A0 00     SELOOP LDYIM $00
2580: 1CE4 B1 E6            LDAIY CURADL COMPARE INSTRUCTION
2590: 1CE6 C5 FB            CMPZ  POINTH AGAINST DATA TO BE SEARCHED
2600: 1CE8 D0 07            BNE   SEARA  SKIP TO NEXT INSTRUCTION, IF NOT EQUAL
2610: 1CEA C8               INY
2620: 1CEB B1 E6            LDAIY CURADL
2630: 1CED C5 FA            CMPZ  POINTL
2640: 1CEF F0 D9            BEQ   CMND   RETURN, IF 2BYTE PATTERN IS FOUND
2650: 1CF1 20 5C 1E  SEARA  JSR   OPLEN  GET LENGTH OF THE CURRENT INSTRUCTION
2660: 1CF4 20 F8 1E         JSR   NEXT   SKIP TO NEXT INSTRUCTION
2670: 1CF7 30 E9            BMI   SELOOP SEARCH AGAIN, IF CURAD IS LESS THAN CEND
2680: 1CF9 10 3E            BPL   ERRA
2690:
2700: 1CFB C9 10     INSERT CMPIM $10     INSERT COMMAND?
2710: 1CFD D0 0A            BNE   INPUT
2720: 1CFF 20 20 1E         JSR   RDINST READ INSTRUCTION AND COMPUTE LENGTH
2730: 1D02 10 C9            BPL   SEARCH COM. KEY?
2740: 1D04 20 47 1E         JSR   FILLWS MOVE DATA IN WS DOWNWARD BY THE AM. IN BYTES
2750: 1D07 F0 C1            BEQ   CMND   RETURN TO DISPLAY THE INSERTED INSTR.
2760:
2770: 1D09 C9 13     INPUT  CMPIM $13     INPUT COMMAND?
2780: 1D0B D0 14            BNE   SKIP
2790: 1D0D 20 20 1E         JSR   RDINST READ INSTRUCTION AND COMPUTE LENGTH
2800: 1D10 10 BB            BPL   SEARCH COM. KEY?
2810: 1D12 20 5C 1E         JSR   OPLEN  LENGTH OF THE CURRENT INSTR.
2820: 1D15 20 F8 1E         JSR   NEXT   RETURN WITH N=1, IF CURAD IS LESS THAN CEND
2830: 1D18 A5 FD            LDAZ  TEMPX  LENGTH OF INSTR. TO BE INSERTED
2840: 1D1A 85 F6            STAZ  BYTES
```

```
2850: 1D1C 20 47 1E          JSR   FILLWS MOVE DATA IN WS DOWNWARD BY THE AM. IN BYTES
2860: 1D1F F0 A9             BEQ   CMND   RETURN TO DISPLAY THE INSERTED DATA
2870:
2880: 1D21 C9 12     SKIP    CMPIM $12    SKIP COMMAND?
2890: 1D23 D0 07             BNE   DELETE
2900: 1D25 20 F8 1E          JSR   NEXT   SKIP TO NEXT INSTRUCTION. CURAD LESS THAN CEND?
2910: 1D28 30 A0             BMI   CMND
2920: 1D2A 10 0D             BPL   ERRA
2930:
2940: 1D2C C9 11     DELETE  CMPIM $11    DELETE COMMAND?
2950: 1D2E D0 09             BNE   ERRA
2960: 1D30 20 B3 1E          JSR   UP     DELETE CURRENT INSTR. BY MOVING UP THE WS
2970: 1D33 20 EA 1E          JSR   RECEND ADJUST CURRENT END ADDRESS
2980: 1D36 4C CA 1C          JMP   CMND
2990:
3000: 1D39 A9 EE     ERRA    LDAIM $EE
3010: 1D3B 85 FB             STAZ  POINTH
3020: 1D3D 85 FA             STAZ  POINTL
3030: 1D3F 85 F9             STAZ  INH
3040: 1D41 A9 03             LDAIM $03
3050: 1D43 85 F6             STAZ  BYTES
3060: 1D45 20 8E 1D  ERRB    JSR   SCANDS DISPLAY EEEEEE UNTIL KEY IS RELEASED
3070: 1D48 D0 FB             BNE   ERRB
3080: 1D4A 4C CA 1C          JMP   CMND
3090:
3100:
3110:
3120:
3130:                        EDITOR'S SUBROUTINES
3140:
3150:                        SCAN IS A SUBROUTINE, FILLING UP
3160:                        THE DISPLAY BUFFER DETERMINED BY
3170:                        CURAD. THEN THE DISPLAY IS SCANNED
3180:                        DEPENDING OF THE LENGTH OF THE INSTRUCTION
3190:                        POINTED BY CURAD
3200:                        IF A DEPRESSED KEY IS DETECTED
3210:
3220:
3230:
3240:
3250:                        SCAN RETURNS WITH VALUE IN ACCU
3260:
3270: 1D4D A2 02     SCAN    LDXIM $02    FILL UP THE DISPLAY BUFFER
3280: 1D4F A0 00             LDYIM $00
3290: 1D51 B1 E6     FILBUF  LDAIY CURADL START FILLING AT OP CODE
3300: 1D53 95 F9             STAX  INH
3310: 1D55 C8                INY
3320: 1D56 CA                DEX
3330: 1D57 10 F8             BPL   FILBUF
3340: 1D59 20 5C 1E          JSR   OPLEN  STORE INSTRUCTION LENGTH IN BYTES
3350: 1D5C 20 8E 1D  SCANA   JSR   SCANDS DISPLAY CURRENT INSTRUCTION
3360: 1D5F D0 FB             BNE   SCANA  KEY RELEASED?
3370: 1D61 20 8E 1D  SCANB   JSR   SCANDS DISPLAY CURRENT INSTRUCTION
3380: 1D64 F0 FB             BEQ   SCANB  ANY KEY DEPRESSED?
3390: 1D66 20 8E 1D          JSR   SCANDS DISPLAY CURRENT INSTRUCTION
3400: 1D69 F0 F6             BEQ   SCANB  ANY KEY STILL DEPRESSED?
3410: 1D6B 20 F9 1D          JSR   GETKEY IF YES, RETURN WITH KEY IN ACCU
3420: 1D6E 60                RTS
3430:
3440:                        GETBYT READS 2 HEXKEYS AND COMPOSES
3450:                        THEIR VALUES IN THE A REGISTER. IF ONLY
3460:                        HEXKEYS WERE DEPRESSED, IT RETURNS WITH
3470:                        N=1. IF A COMMAND KEY WAS DEPRESSED, IT
3480:                        RETURNES WITH N=0.
3490:
3500: 1D6F 20 5C 1D  GETBYT  JSR   SCANA  READ HIGH ORDER NIBBLE
3510: 1D72 C9 10             CMPIM $10
3520: 1D74 10 11             BPL   BYTEND COMMAND KEY?
3530: 1D76 0A                ASLA
3540: 1D77 0A                ASLA         IF NOT, SAVE HIGH ORDER NIBBLE
3550: 1D78 0A                ASLA
3560: 1D79 0A                ASLA
3570: 1D7A 85 FE             STA   NIBBLE
3580: 1D7C 20 5C 1D          JSR   SCANA  READ LOW ORDER NIBBLE
3590: 1D7F C9 10             CMPIM $10
3600: 1D81 10 04             BPL   BYTEND COMMAND KEY?
3610: 1D83 05 FE             ORA   NIBBLE IF  NOT, COMPOSE BYTE
3620: 1D85 A2 FF             LDXIM $FF    SET N=1
3630: 1D87 60        BYTEND  RTS
3640:
3650:                        SCAND IS A SUBROUTINE SHOWING DATA SPECIFIED BY
3660:                        POINT.
3670:                        SCANDS IS A SUBROUTINE SHOWING THE CONTENTS OF
3680:                        DISPLAY BUFFER AS A FUNCTION OF BYTES.
3690:                        THE FOLLOWING SUBROUTINE AK SCANS THE KEYBOARD.
3700:                        IT RETURNS WITH A=0, IF NO KEY IS DEPRESSED AND
3710:                        WITH A≠0 IF A KEY IS DEPRESSED.
3720:                        WHEN SCAND OR SCANDS ARE LEFT, PA0...PA7 IS INPUT.
3730:
3740:
3750:
3760:
3770: 1D88 A0 00     SCAND   LDYIM $00
3780: 1D8A B1 FA             LDAIY POINTL GET DATA SPECIFIED BY POINT
3790: 1D8C 85 F9             STAZ  INH
```

```
3800: 1D8E A9 7F     SCANDS LDAIM $7F
3810: 1D90 8D 81 1A         STA   PADD   PA0...PA6 IS OUTPUT
3820: 1D93 A2 08            LDXIM $08    ENABLE DISPLAY
3830: 1D95 A4 F6            LDYZ  BYTES  FETCH LENGTH FROM BYTES
3840: 1D97 A5 FB     SCDSA  LDAZ  POINTH OUTPUT 1ST BYTE
3850: 1D99 20 CC 1D         JSR   SHOW
3860: 1D9C 88               DEY
3870: 1D9D F0 0D            BEQ   SCDSB  MORE BYTES?
3880: 1D9F A5 FA            LDAZ  POINTL
3890: 1DA1 20 CC 1D         JSR   SHOW   IF YES, OUTPUT 2ND BYTE
3900: 1DA4 88               DEY
3910: 1DA5 F0 05            BEQ   SCDSB  MORE BYTES?
3920: 1DA7 A5 F9            LDAZ  INH
3930: 1DA9 20 CC 1D         JSR   SHOW   IF YES, OUTPUT 3RD BYTE
3940: 1DAC A9 00     SCDSB  LDAIM $00
3950: 1DAE 8D 81 1A         STA   PADD   PA0...PA7 IS INPUT
3960:
3970: 1DB1 A0 03     AK     LDYIM $03    SCAN 3 ROWS
3980: 1DB3 A2 00            LDXIM $00    RESET ROW COUNTER
3990:
4000: 1DB5 A9 FF     ONEKEY LDAIM $FF
4010: 1DB7 8E 82 1A  AKA    STX   PBD    OUTPUT ROW NUMBER
4020: 1DBA E8               INX          ENABLE FOLLOWING ROW
4030: 1DBB E8               INX
4040: 1DBC 2D 80 1A         AND   PAD    INPUT ROW PATTERN
4050: 1DBF 88               DEY          ALL ROWS SCANNED?
4060: 1DC0 D0 F5            BNE   AKA
4070: 1DC2 A0 06            LDYIM $06    TURN DISPLAY OFF
4080: 1DC4 8C 82 1A         STY   PBD
4090: 1DC7 09 80            ORAIM $80    SET BIT7=1
4100: 1DC9 49 FF            EORIM $FF    INVERT KEY PATTERN
4110: 1DCB 60               RTS
4120:
4130:                THE SUBROUTINE SHOW TRANSPORTS THE
4140:                CONTENTS OF ANY DISPLAY BUFFER TO THE
4150:                DISPLAY. THE X REGISTER IS USED AS A
4160:                SCAN COUNTER. IT DETERMINES, IF POINTH,
4170:                POINTL OR INH IS TRANSPORTED TO THE
4180:                DISPLAY.
4190: 1DCC 48        SHOW   PHA           SAVE DISPLAY
4200: 1DCD 84 FC            STYZ  TEMP   SAVE Y REGISTER
4210: 1DCF 4A               LSRA
4220: 1DD0 4A               LSRA         GET HIGH ORDER NIBBLE
4230: 1DD1 4A               LSRA
4240: 1DD2 4A               LSRA
4250: 1DD3 20 DF 1D         JSR   CONVD  OUTPUT HIGH ORDER NIBBLE
4260: 1DD6 68               PLA          GET DISPLAY AGAIN
4270: 1DD7 29 0F            ANDIM $0F    MASK OFF HIGH ORDER NIBBLE
4280: 1DD9 20 DF 1D         JSR   CONVD  OUTPUT LOW ORDER NIBBLE
4290: 1DDC A4 FC            LDYZ  TEMP   RESTORE Y REGISTER
4300: 1DDE 60               RTS
4310:
4320:
4330:
4340:
4350:                THE SUBROUTINE CONVD CONTROLS THE DISPLAY SCAN.
4360:                IT CONVERTS THE CONTENTS OF THE DISPLAY BUFFER
4370:                TO BE DISPLAYED INTO A SEGMENT PATTERN.
4380:
4390: 1DDF A8        CONVD  TAY           USE NIBBLE AS INDEX
4400: 1DE0 B9 0F 1F         LDAY  LOOK   FETCH SEGMENT PATTERN
4410: 1DE3 8D 80 1A         STA   PAD    OUTPUT SEGMENT PATTERN
4420: 1DE6 8E 82 1A         STX   PBD    OUTPUT DIGIT ENABLE
4430: 1DE9 A0 7F            LDYIM $7F
4440: 1DEB 88        DELAY  DEY          DELAY 500 US APPROX.
4450: 1DEC 10 FD            BPL   DELAY
4460: 1DEE 8C 80 1A         STY   PAD    TURN SEGMENTS OFF
4470: 1DF1 A0 06            LDYIM $06
4480: 1DF3 8C 82 1A         STY   PBD    TURN DISPLAY OFF
4490: 1DF6 E8               INX          ENABLE NEXT DIGIT
4500: 1DF7 E8               INX
4510: 1DF8 60               RTS
4520:
4530:                GETKEY CONVERTS A DEPRESSED KEY INTO A
4540:                HEX NUMBER. IT RETURNS WITH THE KEY VALUE
4550:                IN ACCU. IF AN INVALID KEY WAS DEPRESSED,
4560:
4570: 1DF9 A2 21     GETKEY LDXIM $21    START AT ROW 0
4580: 1DFB A0 01     GETKEA LDYIM $01    GET ONE ROW
4590: 1DFD 20 B5 1D         JSR   ONEKEY A=0, NO KEY DEPRESSED
4600: 1E00 D0 07            BNE   KEYIN
4610: 1E02 E0 27            CPXIM $27
4620: 1E04 D0 F5            BNE   GETKEA EACH ROW SCANNED?
4630: 1E06 A9 15            LDAIM $15    RETURN IF INVALID KEY
4640: 1E08 60               RTS
4650: 1E09 A0 FF     KEYIN  LDYIM $FF
4660: 1E0B 0A        KEYINA ASLA         SHIFT LEFT UNTIL Y=KEY NUMBER
4670: 1E0C B0 03            BCS   KEYINB
4680: 1E0E C8               INY
4690: 1E0F 10 FA            BPL   KEYINA
4700: 1E11 8A        KEYINB TXA
4710: 1E12 29 0F            ANDIM $0F    MASK MSD
4720: 1E14 4A               LSRA         DEVIDE BY 2
4730: 1E15 AA               TAX
4740: 1E16 98               TYA
```

```
4750: 1E17 10 03           BPL   KEYIND
4760: 1E19 18        KEYINC CLC
4770: 1E1A 69 07           ADCIM $07    ADD ROW OFFSET
4780: 1E1C CA        KEYIND DEX
4790: 1E1D D0 FA           BNE   KEYINC
4800: 1E1F 60              RTS
4810:
4820:                      RDINST TRANSFERS AN INSTRUCTION FROM KEYBOARD
4830:                      TO THE DISPLAY BUFFER. IT RETURNS WITH N=0 IF
4840:                      A COMMAND KEY WAS DEPRESSED. ONCE THE ENTIRE
4850:                      INSTRUCTION  IS READ, RDINST RETURNS WITH N=1.
4860:
4870:
4880: 1E20 20 6F 1D  RDINST JSR   GETBYT READ OP CODE
4890: 1E23 10 21           BPL   RDB    RETURN, IF COMMAND KEY
4900: 1E25 85 FB           STAZ  POINTH STORE OP CODE IN DISPLAY BUFFER
4910: 1E27 20 60 1E        JSR   LENACC COMPUTE INSTRUCTION LENGTH
4920: 1E2A 84 F7           STYZ  COUNT
4930: 1E2C 84 FD           STYZ  TEMPX
4940: 1E2E C6 F7           DECZ  COUNT
4950: 1E30 F0 12           BEQ   RDA    1 BYTE INSTRUCTION?
4960: 1E32 20 6F 1D        JSR   GETBYT IF NOT, READ FIRST OPERAND
4970: 1E35 10 0F           BPL   RDB    RETURN, IF COMMAND KEY
4980: 1E37 85 FA           STAZ  POINTL STORE 1ST OPERAND IN DISPLAY BUFFER
4990: 1E39 C6 F7           DECZ  COUNT
5000: 1E3B F0 07           BEQ   RDA    2 BYTE INSTRUCTION?
5010: 1E3D 20 6F 1D        JSR   GETBYT IF NOT, READ 2ND OPERAND
5020: 1E40 10 04           BPL   RDB    RETURN IF COMMAND KEY
5030: 1E42 85 F9           STAZ  INH    STORE 2ND OPERAND IN DISPLAY BUFFER
5040: 1E44 A2 FF     RDA   LDXIM $FF    N=1
5050: 1E46 60        RDB   RTS
5060:
5070:                      FILLWS TRANSFERS THE DATA FROM DISPLAY TO
5080:                      WORKSPACE. IT'S ALWAYS LEFT WITH Z=1
5090:
5100: 1E47 20 A6 1E  FILLWS JSR   DOWN   MOVE DATA DOWN BY THE AMOUNT IN BYTES
5110: 1E4A 20 DC 1E        JSR   ADCEND ADJUST CURRENT END ADDRESS
5120: 1E4D A2 02           LDXIM $02
5130: 1E4F A0 00           LDYIM $00
5140: 1E51 B5 F9     WS    LDAZX INH    FETCH DATA FROM DISPLAY BUFFER
5150: 1E53 91 E6           STAIY CURADL INSERT DATA INTO DATA FIELD
5160: 1E55 CA              DEX
5170: 1E56 C8              INY
5180: 1E57 C4 F6           CPYZ  BYTES  ALL INSERTED?
5190: 1E59 D0 F6           BNE   WS     IF NOT, CONTINUE
5200: 1E5B 60              RTS
5210:
5220:                      OPLEN COMPUTES THE LENGTH OF ANY 6502 INSTR.
5230:                      THE INSTR. LENGTH IS SAVED IN BYTES.
5240:
5250: 1E5C A0 00     OPLEN LDYIM $00
5260: 1E5E B1 E6           LDAIY CURADL FETCH OP CODE FROM WS
5270: 1E60 A0 01     LENACC LDYIM $01    LENGTH OF OP CODE IS 1 BYTE
5280: 1E62 C9 00           CMPIM $00
5290: 1E64 F0 1A           BEQ   LENEND BRK INSTRUCTION?
5300: 1E66 C9 40           CMPIM $40
5310: 1E68 F0 16           BEQ   LENEND RTI INSTRUCTION?
5320: 1E6A C9 60           CMPIM $60
5330: 1E6C F0 12           BEQ   LENEND RTS INSTRUCTION?
5340: 1E6E A0 03           LDYIM $03
5350: 1E70 C9 20           CMPIM $20
5360: 1E72 F0 0C           BEQ   LENEND JSR INSTRUCTION?
5370: 1E74 29 1F           ANDIM $1F    STRIP TO 5 BITS
5380: 1E76 C9 19           CMPIM $19
5390: 1E78 F0 06           BEQ   LENEND ANY ABS,Y INSTRUCTION?
5400: 1E7A 29 0F           ANDIM $0F    STRIP TO 4 BITS
5410: 1E7C AA              TAX          USE NIBBLE AS INDEX
5420: 1E7D BC 1F 1F        LDYX  LEN    FETCH LENGTH FROM LEN
5430: 1E80 84 F6     LENEND STYZ  BYTES  DISCARD LENGTH IN BYTES
5440: 1E82 60              RTS
5450:
5460:                      UP MOVES A DATA FIELD BETWEEN CURAD AND CEND
5470:                      UPWARD BY THE AMOUNT IN BYTES
5480:
5490: 1E83 A5 E6     UP    LDAZ  CURADL
5500: 1E85 85 EA           STAZ  MOVADL
5510: 1E87 A5 E7           LDAZ  CURADH MOVAD:=CURAD
5520: 1E89 85 EB           STAZ  MOVADH
5530: 1E8B A4 F6     UPLOOP LDYZ  BYTES
5540: 1E8D B1 EA           LDAIY MOVADL MOVE UPWARD BY THE AMOUNT IN BYTES
5550: 1E8F A0 00           LDYIM $00
5560: 1E91 91 EA           STAIY MOVADL
5570: 1E93 E6 EA           INCZ  MOVADL
5580: 1E95 D0 02           BNE   UPA
5590: 1E97 E6 EB           INCZ  MOVADH MOVADH:=MOVADH+1
5600: 1E99 A5 EA     UPA   LDAZ  MOVADL
5610: 1E9B C5 E8           CMPZ  CENDL
5620: 1E9D D0 EC           BNE   UPLOOP ALL DATA MOVED?
5630: 1E9F A5 EB           LDAZ  MOVADH IF NOT, CONTINUE
5640: 1EA1 C5 E9           CMPZ  CENDH
5650: 1EA3 D0 E6           BNE   UPLOOP
5660: 1EA5 60              RTS
5670:
5680:                      DOWN MOVES A DATA FIELD BETWEEN CURAD
5690:                      AND ENDAD DOWNWARD BY THE AMOUNT IN BYTES
```

```
5700:
5710: 1EA6 A5 E8      DOWN    LDAZ  CENDL
5720: 1EA8 85 EA              STAZ  MOVADL MOVAD:=CEND
5730: 1EAA A5 E9              LDAZ  CENDH
5740: 1EAC 85 EB              STAZ  MOVADH
5750: 1EAE A0 00      DNLOOP  LDYIM $00
5760: 1EB0 B1 EA              LDAIY MOVADL MOVE DOWNWARD BY THE AMOUNT IN BYTES
5770: 1EB2 A4 F6              LDYZ  BYTES
5780: 1EB4 91 EA              STAIY MOVADL
5790: 1EB6 A5 EA              LDAZ  MOVADL
5800: 1EB8 C5 E6              CMPZ  CURADL
5810: 1EBA D0 06              BNE   DNA     ALL DATA MOVED?
5820: 1EBC A5 EB              LDAZ  MOVADH IF NOT, CONTINUE
5830: 1EBE C5 E7              CMPZ  CURADH
5840: 1EC0 F0 10              BEQ   DNEND
5850: 1EC2 38         DNA     SEC
5860: 1EC3 A5 EA              LDAZ  MOVADL
5870: 1EC5 E9 01              SBCIM $01
5880: 1EC7 85 EA              STAZ  MOVADL
5890: 1EC9 A5 EB              LDAZ  MOVADH MOVAD:=MOVAD-1
5900: 1ECB E9 00              SBCIM $00
5910: 1ECD 85 EB              STAZ  MOVADH
5920: 1ECF 4C AE 1E           JMP   DNLOOP
5930: 1ED2 60         DNEND   RTS
5940:
5950:
5960:                 BEGIN SETS CURAD EQUAL TO BEGAD
5970:
5980: 1ED3 A5 E2      BEGIN   LDAZ  BEGADL
5990: 1ED5 85 E6              STAZ  CURADL
6000: 1ED7 A5 E3              LDAZ  BEGADH CURAD:=BEGAD
6010: 1ED9 85 E7              STAZ  CURADH
6020: 1EDB 60                 RTS
6030:
6040:                 ADCEND ADVANCES CURRENT END ADDRESS
6050:                 DOWNWARD BY THE AMOUNT IN BYTES
6060:
6070: 1EDC 18         ADCEND  CLC
6080: 1EDD A5 E8              LDAZ  CENDL
6090: 1EDF 65 F6              ADCZ  BYTES  CEND:=CEND+BYTES
6100: 1EE1 85 E8              STAZ  CENDL
6110: 1EE3 A5 E9              LDAZ  CENDH
6120: 1EE5 69 00              ADCIM $00
6130: 1EE7 85 E9              STAZ  CENDH
6140: 1EE9 60                 RTS
6150:
6160:                 RECEND REDUCES THE CURRENT END ADDRESS
6170:                 BY THE AMOUNT IN BYTES
6180:
6190: 1EEA 38         RECEND  SEC
6200: 1EEB A5 E8              LDAZ  CENDL
6210: 1EED E5 F6              SBCZ  BYTES  CEND:=CEND-BYTES
6220: 1EEF 85 E8              STAZ  CENDL
6230: 1EF1 A5 E9              LDAZ  CENDH
6240: 1EF3 E9 00              SBCIM $00
6250: 1EF5 85 E9              STAZ  CENDH
6260: 1EF7 60                 RTS
6270:
6280:                 NEXT ADVANCES THE CURRENT DISPLAYED ADDRES
6290:                 DOWNWARD BY THE AMOUNT IN BYTES
6300:
6310: 1EF8 18         NEXT    CLC
6320: 1EF9 A5 E6              LDAZ  CURADL
6330: 1EFB 65 F6              ADCZ  BYTES  CURAD:=CURAD+BYTES
6340: 1EFD 85 E6              STAZ  CURADL
6350: 1EFF A5 E7              LDAZ  CURADH
6360: 1F01 69 00              ADCIM $00
6370: 1F03 85 E7              STAZ  CURADH
6380: 1F05 38                 SEC
6390: 1F06 A5 E6              LDAZ  CURADL
6400: 1F08 E5 E8              SBCZ  CENDL
6410: 1F0A A5 E7              LDAZ  CURADH
6420: 1F0C E5 E9              SBCZ  CENDH
6430: 1F0E 60                 RTS
6440:
6450:                 THE LOOKUP TABLE "LOOK" IS USED, TO CONVERT
6460:                 A HEX NUMBER INTO A 7 SEGMENT PATTERN.
6470:                 THE LOOKUP TABLE "LEN" IS USED, TO CONVERT AN
6480:                 INSTRUCTION INTO AN INSTRUCTION LENGTH.
6490:
6500:
6510: 1F0F 40         LOOK    =     $40          "0"
6520: 1F10 79                 =     $79          "1"
6530: 1F11 24                 =     $24          "2"
6540: 1F12 30                 =     $30          "3"
6550: 1F13 19                 =     $19          "4"
6560: 1F14 12                 =     $12          "5"
6570: 1F15 02                 =     $02          "6"
6580: 1F16 78                 =     $78          "7"
6590: 1F17 00                 =     $00          "8"
6600: 1F18 10                 =     $10          "9"
6610: 1F19 08                 =     $08          "A"
6620: 1F1A 03                 =     $03          "B"
6630: 1F1B 46                 =     $46          "C"
6640: 1F1C 21                 =     $21          "D"
```

```
6650: 1F1D 06               =     $06         "E"
6660: 1F1E 0E               =     $0E         "F"
6670:
6680: 1F1F 02         LEN   =     $02
6690: 1F20 02               =     $02
6700: 1F21 02               =     $02
6710: 1F22 01               =     $01
6720: 1F23 02               =     $02
6730: 1F24 02               =     $02
6740: 1F25 02               =     $02
6750: 1F26 01               =     $01
6760: 1F27 01               =     $01
6770: 1F28 02               =     $02
6780: 1F29 01               =     $01
6790: 1F2A 01               =     $01
6800: 1F2B 03               =     $03
6810: 1F2C 03               =     $03
6820: 1F2D 03               =     $03
6830: 1F2E 03               =     $03
6840:
6850: 1F2F 6C 7A 1A         JMI   NMIL   JUMP TO A  USER SELECTABLE NMI VECTOR
6860: 1F32 6C 7E 1A         JMI   IRQL   JUMP TO A  USER SELECTABLE IRQ VECTOR
6870:
6880:                 GETLBL IS AN ASSEMBLER SUBROUTINE. IT SEARCHES FOR
6890:                 LABELS ON THE SYMBOL PSEUDO STACK. IF THIS STACK
6900:                 CONTAINS A VALID LABEL, IT RETURNS WITH THE
6910:                 HIGH ORDER LABEL ADDRESS IN X AND THE LOW ORDER LABEL
6920:                 ADDRESS IN A. IF NO VALID LABEL IS FOUND, IT RE-
6930:                 TURNES WITH Z=1.
6940:
6950: 1F35 B1 E6      GETLBL LDAIY CURADL FETCH CURRENT LABEL NUMBER FROM WS
6960: 1F37 A0 FF             LDYIM $FF    RESET PSEUDO STACK
6970: 1F39 C4 EE      SYMA   CPYZ  LABELS UPPER MOST SYMBOL TABLE ADDRESS?
6980: 1F3B F0 0D             BEQ   SYMB   IF YES, RETURN, NO LABEL ON PSEUDO STACK
6990: 1F3D D1 EC             CMPIY TABLEL LABEL NR. IN WS = LABEL NR. ON PSEUDO STACK?
7000: 1F3F D0 0A             BNE   SYMNXT
7010: 1F41 88                DEY          IF YES, GET HIGH ORDER ADD
7020: 1F42 B1 EC             LDAIY TABLEL
7030: 1F44 AA                TAX          DISCARD HIGH ORDER ADD IN X
7040: 1F45 88                DEY
7050: 1F46 B1 EC             LDAIY TABLEL GET LOW ORDER ADD
7060: 1F48 A0 01             LDYIM $01     PREPARE Y REGISTER
7070: 1F4A 60         SYMB   RTS
7080:
7090: 1F4B 88         SYMNXT DEY                  *********   *********
7100: 1F4C 88                DEY                  * X=ADH *   * A=ADL *
7110: 1F4D 88                DEY                  *********   *********
7120: 1F4E D0 E9             BNE   SYMA
7130: 1F50 60                RTS
7140:
7150:
7160:
7170:
7180:
7190:
7200:
7210:                 ASSEMBLER MAIN ROUTINE
7220:
7230:                 FOLLOWING INSTRUCTIONS ARE ASSEMBLED:
7240:
7250:                 JSR INSTRUCTION
7260:                 JMP INSTRUCTION
7270:                 BRANCH INSTRUCTIONS
7280:
7290:
7300:
7310:
7320:
7330:
7340: 1F51 38         ASSEMB SEC
7350: 1F52 A5 E4             LDAZ  ENDADL
7360: 1F54 E9 FF             SBCIM $FF
7370: 1F56 85 EC             STAZ  TABLEL TABLE:=ENDAD-$FF
7380: 1F58 A5 E5             LDAZ  ENDADH
7390: 1F5A E9 00             SBCIM $00
7400: 1F5C 85 ED             STAZ  TABLEH
7410: 1F5E A9 FF             LDAIM $FF
7420: 1F60 85 EE             STAZ  LABELS
7430: 1F62 20 D3 1E          JSR   BEGIN  CURAD:=BEGAD
7440:
7450: 1F65 20 5C 1E   PASSA  JSR   OPLEN  START PASS ONE, GET CURR. INSTR.
7460: 1F68 A0 00             LDYIM $00
7470: 1F6A B1 E6             LDAIY CURADL FETCH CURRENT INSTRUCTION
7480: 1F6C C9 FF             CMPIM $FF    IS THE CURRENT INSTR. A LABEL?
7490: 1F6E D0 1D             BNE   NXTINS
7500: 1F70 C8                INY
7510: 1F71 B1 E6             LDAIY CURADL IF YES, FETCH LABEL NR.
7520: 1F73 A4 EE             LDYZ  LABELS
7530: 1F75 91 EC             STAIY TABLEL DEPOSIT LABEL NR. ON SYMBOL STACK
7540: 1F77 88                DEY
7550: 1F78 A5 E7             LDAZ  CURADH GET HIGH ORDER ADD
7560: 1F7A 91 EC             STAIY TABLEL DEPOSIT ON SYMBOL STACK
7570: 1F7C 88                DEY
7580: 1F7D A5 E6             LDAZ  CURADL GET LOW ORDER ADD
7590: 1F7F 91 EC             STAIY TABLEL DEPOSIT ON SYMBOL STACK
```

```
7600: 1F81 88                DEY
7610: 1F82 84 EE             STYZ  LABELS ADJUST PSEUDO STACK POINTER
7620: 1F84 20 83 1E          JSR   UP     DELETE CURRENT LABEL IN WS
7630: 1F87 20 EA 1E          JSR   RECEND ADJUST CURRENT END ADD
7640: 1F8A 4C 65 1F          JMP   PASSA  LOOK FOR MORE LABELS
7650:
7660: 1F8D 20 F8 1E  NXTINS  JSR   NEXT   IF NO LABEL, SKIP TO NEXT INSTR.
7670: 1F90 30 D3             BMI   PASSA  ALL LABELS IN WS COLLECTED?
7680: 1F92 20 D3 1E          JSR   BEGIN  START PASS 2
7690: 1F95 20 5C 1E  PASSB   JSR   OPLEN  GET LENGTH OF THE CURRENT INSTR.
7700: 1F98 A0 00             LDYIM $00
7710: 1F9A B1 E6             LDAIY CURADL FETCH CURRENT INSTR.
7720: 1F9C C9 4C             CMPIM $4C    JMP INSTR.?
7730: 1F9E F0 16             BEQ   JUMPS
7740: 1FA0 C9 20             CMPIM $20    JSR INSTR.?
7750: 1FA2 F0 12             BEQ   JUMPS
7760: 1FA4 29 1F             ANDIM $1F    STRIP TO 5 BITS
7770: 1FA6 C9 10             CMPIM $10    ANY BRANCH INSTRUCTION?
7780: 1FA8 F0 1A             BEQ   BRINST
7790: 1FAA 20 F8 1E  PB      JSR   NEXT   IF NOT, RETURN
7800: 1FAD 30 E6             BMI   PASSB  ALL LABELS BETWEEN CURAD AND ENDAD ASSEMBLED?
7810: 1FAF A9 03             LDAIM $03    ENABLE 3 DISPLAY BUFFERS
7820: 1FB1 85 F6             STAZ  BYTES
7830: 1FB3 4C 33 1C          JMP   START  EXIT HERE  ********
7840:
7850:
7860: 1FB6 C8        JUMPS   INY          SET POINTER TO LABEL NR.
7870: 1FB7 20 35 1F          JSR   GETLBL GET LABEL ADD
7880: 1FBA F0 EE             BEQ   PB     RETURN, IF NOT FOUND
7890: 1FBC 91 E6             STAIY CURADL STORE LOW ORDER ADD
7900: 1FBE 8A                TXA
7910: 1FBF C8                INY
7920: 1FC0 91 E6             STAIY CURADL STORE HIGH ORDER ADD
7930: 1FC2 D0 E6             BNE   PB
7940:
7950: 1FC4 C8        BRINST  INY          SET POINTER TO LABEL NR.
7960: 1FC5 20 35 1F          JSR   GETLBL GET LABEL ADD.
7970: 1FC8 F0 E0             BEQ   PB     RETURN, IF LABEL NOT FOUND
7980: 1FCA 38                SEC
7990: 1FCB E5 E6             SBCZ  CURADL COMPUTE BRANCH OFFSET
8000: 1FCD 38                SEC
8010: 1FCE E9 02             SBCIM $02    DESTINATION-SOURCE-2=OFFSET
8020: 1FD0 91 E6             STAIY CURADL INSERT BRANCH OFFSET IN WS
8030: 1FD2 4C AA 1F          JMP   PB
8040:
8050:
8060
8070
8080
8090:                THE SUBROUTINE BRANCH COMPUTES THE OFFSET OF BRANCH
8100:                INSTRUCTIONS. THE 2 RIGHT HAND DISPLAYS SHOW THE
8110:                COMPUTED OFFSET DEFINED BY THE 4 LEFT HAND DISPLAYS.
8120:                THE PROGRAM MUST BE STOPPED WITH THE RESET KEY.
8130:
8140: 1FD5 D8        BRANCH  CLD
8150: 1FD6 A9 00             LDAIM $00    RESET DISPLAY BUFFER
8160: 1FD8 85 FB             STAZ  POINTH
8170: 1FDA 85 FA             STAZ  POINTL
8180: 1FDC 85 F9             STAZ  INH
8190: 1FDE 20 6F 1D  BR      JSR   GETBYT READ SOURCE
8200: 1FE1 10 F2             BPL   BRANCH COMMAND KEY?
8210: 1FE3 85 FB             STAZ  POINTH SAVE SOURCE IN BUFFER
8220: 1FE5 20 6F 1D          JSR   GETBYT READ DESTINATION
8230: 1FE8 10 EB             BPL   BRANCH COMMAND KEY
8240: 1FEA 85 FA             STAZ  POINTL SAVE DESTINATION IN BUFFER
8250: 1FEC 18                CLC
8260: 1FED A5 FA             LDAZ  POINTL FETCH DESTINATION
8270: 1FEF E5 FB             SBCZ  POINTH SUBTRACT SOURCE
8280: 1FF1 85 F9             STAZ  INH
8290: 1FF3 C6 F9             DECZ  INH    EQUALIZE AND SAVE OFFSET IN BUFFER
8300: 1FF5 4C DE 1F          JMP   BR
8310:
8320:                VECTORS AT THE END OF THE MEMORY:
8330:
8340:                1FFA $2F   NMI VECTOR
8350:                1FFB $1F
8360:                1FFC $1D   RESET VECTOR
8370:                1FFD $1C
8380:                1FFE $32   IRQ OR BRK VECTOR
8390:                1FFF $1F
8400:
8410:                END OF JUNIOR'S MONITOR
8420:
8430:
8440:
8450:
8460:
8470:
8480:
8490:
8500:
8510:
8520:
8530:
```

```
8540:  ACC     00F3    ADCEND  1EDC    ADDRES  1CA3    ADLOOP  1CA5
8550:  ADMODE  1C5C    AK      1DB1    AKA     1DB7    ASSEMB  1F51
8560:  BEGADH  00E3    BEGADL  00E2    BEGIN   1ED3    BR      1FDE
8570:  BRANCH  1FD5    BRINST  1FC4    BYTEND  1D87    BYTES   00F6
8580:  CENDH   00E9    CENDL   00E8    CMND    1CCA    CNTA    1AF4
8590:  CNTB    1AF5    CNTC    1AF6    CNTD    1AF7    CNTE    1AFC
8600:  CNTF    1AFD    CNTG    1AFE    CNTH    1AFF    CONVD   1DDF
8610:  COUNT   00F7    CURADH  00E7    CURADL  00E6    DAMODE  1C66
8620:  DATA    1C90    DELAY   1DEB    DELETE  1D2C    DNA     1EC2
8630:  DNEND   1ED2    DNLOOP  1EAE    DOWN    1EA6    EDETA   1AE4
8640:  EDETB   1AE5    EDETD   1AE7    EDIT    1CC0    EDITOR  1CB5
8650:  ENDADH  00E5    ENDADL  00E4    ERRA    1D39    ERRB    1D45
8660:  FILBUF  1D51    FILLWS  1E47    GETBYT  1D6F    GETKEA  1DFB
8670:  GETKEY  1DF9    GETLBL  1F35    GOEXEC  1C45    ILLKEY  1C8C
8680:  INH     00F9    INL     00F8    INPUT   1D09    INSERT  1CFB
8690:  IRQH    1A7F    IRQL    1A7E    JUMPS   1FB6    KEYIN   1E09
8700:  KEYINA  1E0B    KEYINB  1E11    KEYINC  1E19    KEYIND  1E1C
8710:  KEY     00E1    LABELS  00EE    LENACC  1E60    LENEND  1E80
8720:  LEN     1F1F    LOOK    1F0F    LOYS    1C00    MODE    00FF
8730:  MOVADH  00EB    MOVADL  00EA    NEXT    1EF8    NIBBLE  00FE
8740:  NMIH    1A7B    NMIL    1A7A    NXTINS  1F8D    ONEKEY  1DB5
8750:  OPLEN   1E5C    PADD    1A81    PAD     1A80    PASSA   1F65
8760:  PASSB   1F95    PB      1FAA    PBDD    1A83    PBD     1A82
8770:  PCH     00F0    PCKEY   1C7D    PCL     00EF    POINTH  00FB
8780:  POINTL  00FA    PREG    00F1    RDA     1E44    RDB     1E46
8790:  RDFLAG  1AD5    RDINST  1E20    RECEND  1EEA    RESET   1C1D
8800:  SAVE    1C00    SAVEA   1C05    SAVEB   1C0F    SCAN    1D4D
8810:  SCANA   1D5C    SCANB   1D61    SCAND   1D88    SCANDS  1D8E
8820:  SCDSA   1D97    SCDSB   1DAC    SEARA   1CF1    SEARCH  1CCD
8830:  SELOOP  1CE2    SHOW    1DCC    SKIP    1D21    SPUSER  00F2
8840:  STARA   1C38    START   1C33    STEP    1C70    STEPA   1C7A
8850:  SYMA    1F39    SYMB    1F4A    SYMNXT  1F4B    TABLEH  00ED
8860:  TABLEL  00EC    TEMP    00FC    TEMPX   00FD    UP      1E83
8870:  UPA     1E99    UPLOOP  1E8B    WDETC   1AE6    WS      1E51
8880:  XREG    00F5    YREG    00F4
```