Scan of manuals given by Anton Muller, KIM Gebruikers Club

KOIA

65Ø2

MACRO ASSEMBLER

AND

TEXT EDITOR

FOR PET, APPLE, SYM and OTHERS

```
>ASSEMBLE LIST

                0100 ;MOVE FROM TABLE1 TO TABLE2
                0110          .BA $400
0400- A0 00     0120 START    LDY #00
0402- B9 0B 04  0130 LOOP     LDA TABLE1,Y
0405- 99 0B 05  0140          STA TABLE2,Y
0408- C8        0150          INY
0409- D0 F7     0160          BNE LOOP
                0165
                0170 ;
040B-           0180 TABLE1   .DS 256     ;STORAGE
050B-           0190 TABLE2   .DS 256     ;   *
                0200 ;
                0210          .EN

LABEL FILE [/ = EXTERNAL]

START= 0400       LOOP=0402     TABLE1=040B
TABLE2=050B


//0000,060B,060B
>
```

This ASSEMBLER and TEXT EDITOR was written in machine language-not BASIC

K022

65Ø2

MACRO ASSEMBLER

AND

TEXT EDITOR

FOR PET, APPLE, SYM and OTHERS

```
>ASSEMBLE LIST

                    0100 ;MOVE FROM TABLE1 TO TABLE2
                    0110          .BA $400
0400- A0 00         0120 START    LDY #00
0402- B9 0B 04      0130 LOOP     LDA TABLE1,Y
0405- 99 0B 05      0140          STA TABLE2,Y
0408- C8            0150          INY
0409- D0 F7         0160          BNE LOOP
                    0165
                    0170 ;
040B-               0180 TABLE1   .DS 256     ;STORAGE
050B-               0190 TABLE2   .DS 256     ;   *
                    0200 ;
                    0210          .EN

LABEL FILE [/ = EXTERNAL]

START= 0400       LOOP=0402     TABLE1=040B
TABLE2=050B


//0000,060B,060B
>
```

This ASSEMBLER and TEXT EDITOR was written in machine language-not BASIC

COPYRIGHT NOTES

This manual and all object medias (Cassettes, Floppy Disks, etc.) is serial numbered and protected by a legitimate copyright. No part of this manual may be copied or reproduced without the express written permission of the copyright owner, Carl Moser. You may make a backup copy of the cassette or floppy disk to protect your copy of this software. It is though a Federal crime to make a copy of the manual, cassette, or floppy disk for use by anyone other than the individual who purchased this software or the individual a company purchased the software for.

Thus, you are in violation of Federal Copyright Laws if you do one of the following:

- Make a copy of the manual.
- If you allow someone else to use your copy (or backups) of the object media (Cassettes, Floppy Disks, etc.) while you retain a copy or are using a copy.
- If your Company or others purchase one or more copies and more individuals use this software than the number purchased.
- If you allow someone else to do the copying of this material, you will be considered as a party to the infringement.

A reward will be provided for anyone who supplies information which leads to the prosecution of parties who violate this copyright.

We do not presume that you are or will violate copyright laws. Most users do not. Some though do, and may not realize the consequences for violation of this Federal Law. Penalities and fines can be quite severe for both individuals and companies who infringe on this copyright.

Most importantly, software houses like the one which wrote this software have a tremendous investment in this software that can not be fully recovered if current illegal copying continues. Also, updates and program maintenance will have to be terminated if the return on investment is not sufficient.

Finally, an expressed appreciation is given to the purchaser of this software. We hope that you find it a valuable and worthwhile investment.

Serial Number: K022
Computer: Kim

6502 RELOCATING MACRO ASSEMBLER/TEXT EDITOR
1.0

Contents

EASTERN HOUSE SOFTWARE
CARL W. MOSER
3239 LINDA DRIVE
WINSTON SALEM, N.C. 27106

Serial Number: K01A
Computer: KIM

6502 RELOCATING MACRO ASSEMBLER/TEXT EDITOR
1.0

Contents

EASTERN HOUSE SOFTWARE
CARL W. MOSER
3239 LINDA DRIVE
WINSTON SALEM, N.C. 27106

13. SPECIFIC APPLICATION NOTES

A. PET

The default file boundaries for PET are: text file =0770-17FC, label file=1800-1EFC, and relocatable buffer start=1F00. When entering the upper file boundary via the SET command, enter the end address minus 3 (example: If the end =1EFF, then enter 1EFC).

The PET does not treat the ascii character set in the traditional manner. Thus part 11 dealing with ascii control codes should be ignored.

PET has a very nice cursor controlled screen editing feature which ASSM/TED takes full advantage of. Thus references to ^H (backspace), ^X (delete line), rubout (delete character), and EDIT form 2 should be ignored.

The command syntax for the >GET, >PUT, and >OUTPUT commands is expanded as follows:

GET
PUT
OUTPUT } Tn Fm "filename"

Where Tn - n=1 or 2 representing the selected tape drive (ex: T2). Default=1.

Fm - m=user assigned file number. Default=0

"filename" - filename is a name which must be enclosed in quotes. Default=null

Example: GET T2 F6 "MEMORY TEST"

When assembling source from tape, ASSM/TED assumes source input is on deck 1 and any relocatable output to be directed to deck 2.

When ASSM/TED is outputting, the user can temporarily stop the printout by pressing the STOP key, or suppress the printing but continuing processing via the OFF key, or terminating both processing and printing and immediately returning to command mode via DEL. To continue outputting after depressing STOP, press any key except DEL or OFF.

Page 1 variables referenced in the manual were relocated to 3F00 since PET uses most of the stack for load functions and subroutine linkage. All page 1 variables are offset by 3E00. For example, a reference in the manual to location 0123 is actually 0123+3E00=3F23.

Also, zero page references were relocated from BB-F8 to 1B-58 to avoid conflicts with the PET operating system and the Commodore monitor program. Thus, all zero page variables are offset by 60. For example, a reference in the manual to location 00DD is actually 00DD+60=003D.

B. PET Users Notes for the ASSM/TED

The following notes are provided to help you use the ASSM/TED on the PET.

- One command not previously discussed is SH G (shift graphics) and SH L (shift lower case). The SH G allows the entry of graphics characters when the SHIFT key is depressed. The SH L allows the entry of lower case alphabetical characters when the SHIFT key is depressed.

- When using the ASSM/TED to save source programs on tape (PUT command), it writes a separate file header for its own internal control. As a result, the PET screen will display WRITING name of program twice before the program is actually saved. Thus, this is normal for the ASSM/TED.

- Always use the PET monitor with the ASSM/TED. This will allow the user to BREAK from the ASSM/TED to display memory locations, etc. In addition, if a source program has been assembled and stored in memory, the PET monitor is used to save the final program on tape.

- Once the PET monitor has been loaded, the ASSM/TED is started by typing G 2000 (cold start). If the user has exited the ASSM/TED with a BREAK command and wishes to re-enter without losing any data, a warm start can be executed by typing G 2090.

- The ASSM/TED makes very good use of the PET editing ability. If the user wishes to make changes to a line of text, use the PRINT command to display the desired line and then cusor up and over to make the desired changes. (A line of text may be copied by using the method described above to change the line number. The COPY command may be used to copy multiple lines.)

  The ASSM/TED uses a DELETE command to delete a line or range of lines. The user can, if desired, delete a single line by typing the line number and hitting RETURN.

Due to a problem in the PET editor ROM (at least on our PET), there are a couple of strange things to watch out for.

(a) Normally, the ASSM/TED will allow up to 80 characters per line (that is, two full lines on the screen). However, the last line (line 24) on the screen will allow only 40 characters per line (that is, one line on the screen) without giving an error message or giving a double digit line number. The basic cure for this is to avoid entering more than 40 characters when on the last line. If the user wishes to enter more than 40 characters, simply hit CLEAR SCREEN key and then the RETURN key.

(b) When entering a line of text that contains more than 70 characters and the RETURN key is hit, the cusor will move to a character position on the same line. Hit the RETURN key a second time to exit the line. Although this is bothersome, it doesn't affect what has been entered on the line.

At present, the ASSM/TED does not contain a printer subroutine to interface with a printer. If the user wishes to add his own subroutine, you may add a JSR instruction at 37E2, 37E3, 37E4. Don't attempt to use memory locations $3F00 to 3FFF for any reason. This area is used by the ASSM/TED.

Listing 2 (PET)

```
>PASS
0000              .LS
0001              .CT
0010 ;***RELOCATING LOADER FOR THE PET ASSM/TED***
0020 ;
0030 ;
0040 ;
0050              .OS
0060 ;
0070 ;*****COPYRIGHT 1979 BY CARL MOSER.*****
0080 ;*****     ALL RIGHTS RESERVED.      *****
0090 ;
0100 ;
0110 ;
0120 ;
0130 ;+++++++  USER INPUTTED VARIABLES BEFORE EXECUTION +++++++
0140 FILE/NO      .DE $3F10      ;FILE NUMBER (0-99)
0150 OFFSET       .DE $40        ;RELOCATOR OFFSET (2 BYTES)
0160 BUFFER       .DE $28        ;ADDRS. OF R.L. BUFFER
0170 ;
0180 ;
0190 ;
0200 ;              RELOCATOR DIRECTIVES
0210 ;
0220 ;  DIRECTIVE                    DESCRIPTION
0230 ;  ---------                    -----------
0240 ;     0F              EXTERNAL 2 BYTE ADDRS. PRECEEDS,
0250 ;                     DON'T RELOCATE.  OTHERWISE RELOCATE.
0260 ;
0270 ;     1F              #L,  DATA PRECEEDS.
0280 ;
0290 ;     2F              #H,  DATA PRECEEDS, LO PART FOLLOWS.
0300 ;
0310 ;     3F              .AS OR .HS BYTE FOLLOWS.
0320 ;
0330 ;     4F              .SE OR .SI 2 BYTE ADDRS. FOLLOWS.
0340 ;
0350 ;     5F              TURN RELOCATOR ON (VIA .RS).
0360 ;                     (RESOLVE ADDRESSES AND RELOCATE
0370 ;                      CODE.)
0380 ;
0390 ;     6F              TURN RELOCATOR OFF (VIA .RC).
0400 ;                     (RESOLVE ADDRESSES BUT DO NOT
0410 ;                      RELOCATE CODE.)
0420 ;
0430 ;     7F              .DS - 2 BYTE BLOCK VALUE FOLLOWS.
0440 ;
0450 ;
0460              .BA $0800
0470 ;
0480 ;TAPE INPUT PARMS
0490 LOAD/NO      .DE $3F23 0: NO STORE; 1: STORE
0500 TSTART       .DE $3F24 LOAD BEGINNING AT TSTART
0510 TEND         .DE $3F26 STOP LOADING AT TEND
0520 ;
0530 ;
0540 ;HEADER INPUT DATA
```

```
                    0550 HFILE/NO    .DE $3F7A HEADER FILE NUMBER
                    0560 HSTART      .DE $3F7B HEADER START
                    0570 HEND        .DE $3F7D HEADER END
                    0580 ;
                    0590 ;
                    0600 ;VARIABLES
                    0610 SCRAT       .DE $3F1E SCRATCH AREA
                    0620 TEMP1       .DE $3F1F SCRATCH AREA
                    0630 TEMP2       .DE $3F20 SCRATCH AREA
                    0640 SAVE        .DE $3F21 SCRATCH AREA
                    0650 ADDRS       .DE $3C 4 BYTES OF ADDRESS INFO.
                    0660 BUFF.END    .DE $3F23 END OF 256 BYTE BUFFER
                    0670 BUFF.INDEX  .DE $3F24 PRESENT ACCESSED DATA FROM BUFFER
                    0680 ;
                    0690 ;
                    0700 ;R(X)=00:  RELOCATOR ON
                    0710 ;R(X)=02:  RELOCATOR OFF
                    0720 ;
                    0730 ;BEGIN EXECUTION AT LABEL START
                    0740 ;
0800- A2 FF         0750 START       LDX #$FF
0802- 9A            0760             TXS INITIALIZE STACK
0803- E8            0770             INX R(X)=00: SET RELOCATOR INITIALLY TO ON
0804- D8            0780             CLD
0805- 8E 21 3F      0790             STX SAVE R(X)=00
0808- 20 E3 08      0800             JSR LOAD←BUFF
080B- 4C 11 08      0810             JMP ENTY
080E- 20 71 09      0820 LOOP1       JSR GET←DATA
                    0830 ;
0811- C9 7F         0840 ENTY        CMP #$7F      ;CKG. FOR .DS
0813- D0 03         0850             BNE PRO.3F
0815- 4C A7 09      0860             JMP PRO.7F    ;JUMP TO PROCESS DIR. 7F
0818- C9 3F         0870 PRO.3F      CMP #$3F CKG. FOR RELOCATOR DIRECTIVE
081A- D0 0B         0880             BNE OP←CKG
081C- 20 71 09      0890             JSR GET←DATA
081F- 81 3C         0900             STA (ADDRS,X)
0821- 20 85 09      0910             JSR INC←ADDRS
0824- 4C 0E 08      0920             JMP LOOP1
0827- C9 4F         0930 OP←CKG      CMP #$4F CKG. FOR .SE, .SI
0829- D0 03         0940             BNE W:
082B- 4C AA 08      0950             JMP TWO←BYT←AD
082E- C9 5F         0960 W:          CMP #$5F CKG. FOR RELOCATOR ON
0830- D0 04         0970             BNE CKNX
0832- A2 00         0980             LDX #$00
0834- F0 D8         0990             BEQ LOOP1
                    1000 ;
0836- C9 6F         1010 CKNX        CMP #$6F CKG. FOR RELOCATOR OFF
0838- D0 04         1020             BNE NO←REL
083A- A2 02         1030             LDX #$02
083C- D0 D0         1040             BNE LOOP1
083E- 81 3C         1050 NO←REL      STA (ADDRS,X) STORE OP CODE
0840- 20 85 09      1060             JSR INC←ADDRS
0843- C9 00         1070             CMP #$00 CKG. FOR BRK INSTR.
0845- F0 C7         1080             BEQ LOOP1
0847- C9 20         1090             CMP #$20 CKG. FOR JSR INSTR.
0849- F0 5F         1100             BEQ TWO←BYT←AD
084B- 8D 21 3F      1110             STA SAVE SAVE R(A), IT CONTAINS OP CODE
084E- 29 9F         1120             AND #$9F
```

```
0850- F0 BC      1130            BEQ LOOP1
0852- AD 21 3F   1140            LDA SAVE RESTORE OP CODE
0855- 29 1D      1150            AND #$1D
0857- C9 08      1160            CMP #$08 ↑↑KG. FOR ONE BYTE INSTR.
0859- F0 B3      1170            BEQ LOOP1
085B- C9 18      1180            CMP #$18 CKG. FOR ONE BYTE INSTR.
085D- F0 AF      1190            BEQ LOOP1
                 1200 ;
                 1210 ;NOW, TEST FOR INSTR. CONTAINING 2 BYTES
                 1220 ;OF ADDRESS INFORMATION
                 1230 ;
085F- AD 21 3F   1240            LDA SAVE RESTORE OP CODE
0862- 29 1C      1250            AND #$1C
0864- C9 1C      1260            CMP #$1C
0866- F0 42      1270            BEQ TWO←BYT←AD
0868- C9 18      1280            CMP #$18
086A- F0 3E      1290            BEQ TWO←BYT←AD
086C- C9 0C      1300            CMP #$0C
086E- F0 3A      1310            BEQ TWO←BYT←AD
                 1320 ;
                 1330 ;THE REMAINING CONTAIN ONE BYTE OF
                 1340 ;ADDRESS INFORMATION
                 1350 ;
                 1360 ;PROCSSING OF ON BYTE ADDRESSES AND IMMEDIATE DATA
0870- 20 71 09   1370 ONE←BYT←AD JSR GET←DATA
0873- 81 3C      1380            STA (ADDRS,X)
0875- 20 85 09   1390            JSR INC←ADDRS
0878- 20 71 09   1400            JSR GET←DATA
087B- C9 2F      1410            CMP #$2F CKG. FOR RELOCATOR DIRECTIVE
087D- F0 14      1420            BEQ IMM←HI CKG. FOR #H,
087F- C9 1F      1430            CMP #$1F CKG. FOR RELOCATOR DIRECTIVE
0881- D0 8E      1440            BNE ENTY
                 1450 ;
                 1460 ;PROCESS #L,  DATA FOR RELOCATION
0883- 20 92 09   1470 IMM←LO     JSR DEC←ADDRS
0886- 18         1480            CLC
0887- A1 3C      1490            LDA (ADDRS,X)
0889- 65 40      1500            ADC ←OFFSET+$00 ADD OFFSET LOW PART FOR #L,
088B- 81 3C      1510            STA (ADDRS,X)
088D- 20 85 09   1520            JSR INC←ADDRS
0890- 4C 0E 08   1530 BACK←TO←L1 JMP LOOP1
                 1540 ;PROCESS #H,  DATA FOR RELOCATION
0893- 20 71 09   1550 IMM←HI     JSR GET←DATA LOW BYTE FOLLOWS REL. DIR.
0896- 18         1560            CLC
0897- 65 40      1570            ADC ←OFFSET FORM THE LO ADDRS. PART
0899- 08         1580            PHP
089A- 20 92 09   1590            JSR DEC←ADDRS
089D- 28         1600            PLP
089E- A1 3C      1610            LDA (ADDRS,X)
08A0- 65 41      1620            ADC ←OFFSET+$1 NOW FORM THE EFFECTIVE #H,
08A2- 81 3C      1630            STA (ADDRS,X)
08A4- 20 85 09   1640            JSR INC←ADDRS
08A7- 4C 0E 08   1650            JMP LOOP1
                 1660 ;
                 1670 ;PROCESSING OF TWO BYTE ADDRESSES
08AA- A0 02      1680 TWO←BYT←AD LDY #$02
08AC- 98         1690 XX         TYA
08AD- 48         1700            PHA SAVE R(Y)
```

```
08AE- 20 71 09  1710            JSR GET←DATA
08B1- 81 3C     1720            STA (ADDRS,X)
08B3- 20 85 09  1730            JSR INC←ADDRS
08B6- 68        1740            PLA
08B7- A8        1750            TAY RESTORE R(Y)
08B8- 88        1760            DEY
08B9- D0 F1     1770            BNE XX
08BB- 20 71 09  1780            JSR GET←DATA
08BE- C9 0F     1790            CMP #$0F CKG. FOR RELOCATOR DIRECTIVE
08C0- D0 03     1800            BNE XY
08C2- 4C 0E 08  1810            JMP LOOP1
08C5- 48        1820 XY         PHA
08C6- 20 92 09  1830            JSR DEC←ADDRS
08C9- 20 92 09  1840            JSR DEC←ADDRS
                1850 ;DECREMENT BACK TO ADDRESS START
                1860 ;
08CC- A1 3C     1870            LDA (ADDRS,X)
08CE- 18        1880            CLC
08CF- 65 40     1890            ADC ←OFFSET ADD OFFSET LO
08D1- 81 3C     1900            STA (ADDRS,X)
08D3- 20 85 09  1910            JSR INC←ADDRS
08D6- A1 3C     1920            LDA (ADDRS,X)
08D8- 65 41     1930            ADC ←OFFSET+$1 ADD OFFSET HI
08DA- 81 3C     1940            STA (ADDRS,X)
08DC- 20 85 09  1950            JSR INC←ADDRS
08DF- 68        1960            PLA
08E0- 4C 11 08  1970            JMP ENTY
                1980 ;
                1990 ;SUBROUTINE LOAD BUFFER WITH DATA FROM TAPE
                2000 ;
08E3- A9 7A     2010 LOAD←BUFF  LDA #$7A ADDLO OF START OF HEADER
08E5- 8D 24 3F  2020            STA TSTART+$00
08E8- A9 7F     2030            LDA #$7F ADDLO OF END OF HEADER
08EA- 8D 26 3F  2040            STA TEND+$00
08ED- A9 01     2050            LDA #$01 HI ADDRS
08EF- 8D 25 3F  2060            STA TSTART+$01
08F2- 8D 27 3F  2070            STA TEND+$01
08F5- 8D 23 3F  2080            STA LOAD/NO 01: INDICATE TO LOAD
08F8- 20 D2 09  2090            JSR USER/LOAD USER LOA←BD FROM TAPE ROUTINE
                2100 ;
                2110 ;THE ABOVE SETS UP AND LOADS HEADER INFORMATION
                2120 ;FROM TAPE.  THE HEADER CONTAINS THE MODULE FILE
                2130 ;NUMBER, AND STARTING AND ENDING ADDRESS OF FOLLOWING
                2140 ;DATA.
                2150 ;
                2160 ;
08FB- D0 4D     2170            BNE ERROR IF Z-BIT FALSE, THEN ERROR IN LOADING
08FD- A2 00     2180            LDX #$00
                2190 ;
08FF- AD 7D 3F  2200            LDA HEND+$00
0902- 38        2210            SEC
0903- ED 7B 3F  2220            SBC HSTART+$00
                2230 ;CALCULATE NUMBER OF BYTES IN FOLLOWING DATA
                2240 ;
0906- 8D 23 3F  2250            STA BUFF.END INITIALIZE BUFFER END POINTER
0909- AD 7E 3F  2260            LDA HEND+$01
090C- ED 7C 3F  2270            SBC HSTART+$01
090F- D0 39     2280            BNE ERROR ONLY 256 BYTE BUFFER ALLOWED
```

```
0911- A5 28     2290            LDA *BUFFER
0913- 8D 24 3F  2300            STA TSTART
0916- 18        2310            CLC
0917- 6D 23 3F  2320            ADC BUFF.END # BYTES
091A- 8D 26 3F  2330            STA TEND
091D- A5 29     2340            LDA *BUFFER+$01
091F- 8D 25 3F  2350            STA TSTART+$01
0922- 69 00     2360            ADC #$00
0924- 8D 27 3F  2370            STA TEND+$01
                2380 ;NOW THE START AND END ADDRESS PARMS HAVE BEEN
                2390 ;SET UP TO LOAD FROM TAPE INTO THE BUFFER.
                2400 ;
0927- AD 10 3F  2410            LDA FILE/NO USER ENTERED FILE NUMBER
092A- F0 08     2420            BEQ STORE.DATA IF F# = 00, LOAD ANYWAY
092C- CD 7A 3F  2430            CMP HFILE/NO CMP WITH USER VERSUS THAT ON TAPE
092F- F0 03     2440            BEQ STORE.DATA
0931- 8E 23 3F  2450            STX LOAD/NO R(X)=0; NO STORE
0934- 20 D2 09  2460 STORE.DATA JSR USER/LOAD
                2470 ;
                2480 ;THE ABOVE LOADS IN DATA INTO BUFFER DEPENDING
                2490 ;ON THE STATE OF LOAD/NO
                2500 ;
0937- D0 11     2510            BNE ERROR Z-BIT = FALSE THEN ERROR
0939- A2 00     2520            LDX #$00
093B- AD 7A 3F  2530            LDA HFILE/NO
093E- C9 EE     2540            CMP #$EE COMPARE IF END OF FILE
0940- D0 0C     2550            BNE BUFFLOADED
0942- A9 00     2560            LDA #$00 INDICATE GOOD LOAD
0944- 00        2570 B          BRK
0945- EA        2580            NOP
0946- EA        2590            NOP
0947- 4C 00 08  2600            JMP START
094A- A9 EE     2610 ERROR      LDA #$EE INDICATE ERROR IN LOAD
094C- D0 F6     2620            BNE B
                2630 ;
                2640 ;
                2650 ;NOW GET ADDRS. INFO. AND PUT IN ADDRS+$2, +$3
                2660 ;ADDRS INFO. IS IN FIRST TWO BYTES OF BUFFER
                2670 ;
094E- AD 23 3F  2680 BUFFLOADED LDA LOAD/NO CKG. IF PROPER DATA
0951- F0 90     2690            BEQ LOAD+BUFF
0953- AE 21 3F  2700            LDX SAVE RESTORE R(X)
0956- A0 00     2710            LDY #$00
0958- B1 28     2720            LDA (BUFFER),Y
095A- 85 3E     2730            STA *ADDRS+$2
095C- C8        2740            INY
095D- B1 28     2750            LDA (BUFFER),Y
095F- 85 3F     2760            STA *ADDRS+$3
0961- 8C 24 3F  2770            STY BUFF.INDEX SET BUFFER DATA POINTER
                2780 ;
                2790 ;SET RELOCATION ADDRS. IN ADDRS+$0, +$1
0964- A5 3E     2800            LDA *ADDRS+$2
0966- 18        2810            CLC
0967- 65 40     2820            ADC *OFFSET
0969- 85 3C     2830            STA *ADDRS
096B- A5 41     2840            LDA *OFFSET+$1
096D- 65 3F     2850            ADC *ADDRS+$3
096F- 85 3D     2860            STA *ADDRS+$1
```

```
                        2870 ;
0971- 8E 21 3F          2880 GET←DATA   STX SAVE SAVE X IN CASE WE BR. TO LOAD←BUFF
0974- EE 24 3F          2890            INC BUFF.INDEX INC. 256 BYTE BUFFER POINTER
0977- AC 24 3F          2900            LDY BUFF.INDEX
097A- CC 23 3F          2910            CPY BUFF.END
097D- 90 03             2920            BCC WX BR. IF NOT AT END OF DATA IN BUFFER
097F- 4C E3 08          2930            JMP LOAD←BUFF RELOAD BUFFER
0982- B1 28             2940 WX         LDA (BUFFER),Y
0984- 60                2950            RTS
                        2960 ;
                        2970 ;
                        2980 ;INCREMENT ADDRS+$0, +$1 AND ADDRS+$2, +$3
                        2990 ;
0985- E6 3C             3000 INC←ADDRS  INC ←ADDRS
0987- D0 02             3010            BNE SKIP←INC1
0989- E6 3D             3020            INC ←ADDRS+$1
098B- E6 3E             3030 SKIP←INC1  INC ←ADDRS+$2
098D- D0 02             3040            BNE SKIP←INC2
098F- E6 3F             3050            INC ←ADDRS+$3
0991- 60                3060 SKIP←INC2  RTS
                        3070 ;
                        3080 ;
                        3090 ;DECREMENT ADDRS+$0, +1 AND ADDRS+$2, +$3
                        3100 ;
0992- C6 3C             3110 DEC←ADDRS  DEC ←ADDRS
0994- A5 3C             3120            LDA ←ADDRS
0996- C9 FF             3130            CMP #$FF
0998- D0 02             3140            BNE SKIP←DEC1
099A- C6 3D             3150            DEC ←ADDRS+$1
099C- C6 3E             3160 SKIP←DEC1  DEC ←ADDRS+$2
099E- A5 3E             3170            LDA ←ADDRS+$2
09A0- C9 FF             3180            CMP #$FF
09A2- D0 02             3190            BNE SKIP←DEC2
09A4- C6 3F             3200            DEC ←ADDRS+$3
09A6- 60                3210 SKIP←DEC2  RTS
                        3220 ;
                        3230 ;
                        3240 ;7F LO HI  --  PCL PCH 7F LO HI
                        3250 ;
09A7- 20 71 09          3260 PRO.7F     JSR GET←DATA
09AA- 48                3270            PHA  ;SAVE LO
09AB- 20 71 09          3280            JSR GET←DATA
09AE- A8                3290            TAY  ;SAVE HI IN R(Y)
09AF- AD 24 3F          3300            LDA BUFF.INDEX
09B2- C9 05             3310            CMP #$05      ;NO PROC. IF <= 4
09B4- 90 18             3320            BCC NO.PROC
09B6- 18                3330 PROC.DS    CLC
09B7- 68                3340            PLA  ;GET LO
09B8- 48                3350            PHA
09B9- 65 3C             3360            ADC ←ADDRS
09BB- 85 3C             3370            STA ←ADDRS
09BD- 98                3380            TYA  ;GET HI
09BE- 65 3D             3390            ADC ←ADDRS+1
09C0- 85 3D             3400            STA ←ADDRS+1
09C2- 68                3410            PLA
09C3- 48                3420            PHA  ;GET LO
09C4- 18                3430            CLC
09C5- 65 3E             3440            ADC ←ADDRS+2
```

```
                    3445            .BA $9C7
                    3446            .LS
09C7- 85 3E         3450            STA *ADDRS+2
09C9- 98            3460            TYA  ;GET HI
09CA- 65 3F         3470            ADC *ADDRS+3
09CC- 85 3F         3480            STA *ADDRS+3
09CE- 68            3490 NO.PROC    PLA
09CF- 4C 0E 08      3500            JMP LOOP1
                    3510 ;
                    3520 ;
                    3530 ;
                    3540 ;      *** PET CASSETTE INTERFACE PATCH ***
                    3550 ;
                    3560 ;
                    3570 ;PET DEFINITIONS:
                    3580 VERCK      .DE $020B     ;=1 THEN VERIFY
                    3590 ZZZZ       .DE $F667     ;SET UP BUFFER ADDRS POINTERS
                    3600 CSTE1      .DE $F83B     ;START TAPE MESS.
                    3610 LD300      .DE $F3FF     ;PRINT FILE NAME
                    3620 FNLEN      .DE $EE       ;LENGTH OF FILE NAME
                    3630 FAF        .DE $F495     ;READ HEADER BY NAME
                    3640 FAH        .DE $F5AE     ;READ ANY HEADER
                    3650 LDAD2      .DE $F64D     ;COPYSTART,END ADDRS
                    3660 LD400      .DE $F422     ;LOADING MESS.
                    3670 TRD        .DE $F88A     ;READ DATA
                    3680 TWAIT      .DE $F913     ;WAIT FOR KEY.IRQ
                    3690 STATUS     .DE $020C     ;TAPE ERROR STATUS
                    3700 ;
                    3710 STAL       .DE $F7
                    3720 SAL        .DE $E3
                    3730 STAH       .DE $F8
                    3740 SAH        .DE $E4
                    3750 EAL        .DE $E5
                    3760 EAH        .DE $E6
                    3770 ;
                    3780 ;
09D2- A2 00         3790 USER/LOAD  LDX #$00
09D4- 8E 0B 02      3800            STX VERCK
09D7- 86 EE         3810            STX *FNLEN    ;FILE NAME LENGTH
09D9- 20 67 F6      3820            JSR ZZZZ      ;SET UP CASS. BUFFER POINTERS
09DC- 20 3B F8      3830            JSR CSTE1     ;START TAPE MESS
09DF- 20 FF F3      3840            JSR LD300     ;PRINT FILE NAME
09E2- 20 AE F5      3850            JSR FAH       ;SEARCH FOR ANY HEADER
09E5- D0 03         3860            BNE SK.LOAD2
09E7- A9 EE         3870 EXIT.ERRL1 LDA #$EE
09E9- 60            3880            RTS  ;Z=F THEN ERROR
                    3890 ;
09EA- 20 4D F6      3900 SK.LOAD2   JSR LDAD2     ;COPY START,END ADDRS
09ED- 20 01 0A      3910            JSR OFFSET.ADD
09F0- 20 22 F4      3920            JSR LD400     ;LOADING MESSAGE
09F3- 20 8A F8      3930            JSR TRD       ;READ DATA
09F6- 20 13 F9      3940            JSR TWAIT     ;WAIT FOR KEY.IRQ
09F9- AD 0C 02      3950            LDA STATUS
09FC- 29 10         3960            AND #%00010000        ;TAPE ERROR TEST
09FE- D0 E7         3970            BNE EXIT.ERRL1
0A00- 60            3980            RTS  ;Z=T THEN LOAD OK
```

```
                    3990 ;
0A01- AD 24 3F      4000 OFFSET.ADD LDA TSTART
0A04- 85 F7         4010            STA *STAL
0A06- 85 E3         4020            STA *SAL
0A08- AD 25 3F      4030            LDA TSTART+1
0A0B- 85 F8         4040            STA *STAH
0A0D- 85 E4         4050            STA *SAH
0A0F- AD 26 3F      4060            LDA TEND
0A12- 85 E5         4070            STA *EAL
0A14- AD 27 3F      4080            LDA TEND+1
0A17- 85 E6         4090            STA *EAH
0A19- 60            4100            RTS
                    4110 ;
                    4120 ;
                    4130 END.PGM    .EN
```

LABEL FILE: [/ = EXTERNAL]

```
/FILE/NO=3F10
/LOAD/NO=3F23
/HFILE/NO=3F7A
/SCRAT=3F1E
/SAVE=3F21
/BUFF.INDEX=3F24
ENTY=0811
W:=082E
ONE←BYT←AD=0870
IMM←HI=0893
XY=08C5
B=0944
GET←DATA=0971
SKIP←INC1=098B
SKIP←DEC1=099C
PROC.DS=09B6
/ZZZZ=F667
/FNLEN=00EE
/LOAD2=F64D
/TWAIT=F913
/SAL=00E3
/EAL=00E5
EXIT.ERRL1=09E7
END.PGM=0A1A
//0000,0A1A,0A1A
>
```

```
/OFFSET=0040
/TSTART=3F24
/HSTART=3F7B
/TEMP1=3F1F
/ADDRS=003C
START=0800
PRO.3F=0818
CKNX=0836
IMM←LO=0883
TWO←BYT←AD=08AA
LOAD←BUFF=08E3
ERROR=094A
WX=0982
SKIP←INC2=0991
SKIP←DEC2=09A6
NO.PROC=09CE
/CSTE1=F83B
/FAF=F495
/LD400=F422
/STATUS=020C
/STAH=00F8
/EAH=00E6
SK.LOAD2=09EA
```

```
/BUFFER=0028
/TEND=3F26
/HEND=3F7D
/TEMP2=3F20
/BUFF.END=3F23
LOOP1=080E
OP←CKG=0827
NO←REL=083E
BACK←TO←L1=0890
XX=08AC
STORE.DATA=0934
BUFFLOADED=094E
INC←ADDRS=0985
DEC←ADDRS=0992
PRO.7F=09A7
/VERCK=020B
/LD300=F3FF
/FAH=F5AE
/TRD=F88A
/STAL=00F7
/SAH=00E4
USER/LOAD=09D2
OFFSET.ADD=0A01
```

B. APPLE

The default file boundaries for APPLE are: text file = 0800-17FC, label file = 1800-1EFC, and relocatable buffer start = 1F00. When entering the upper file boundary via the >SET command, enter the end address minus 3 (example: If the end = 17FF, then enter 17FC).

The APPLE II computer does not have tape motor control support. Thus the >ON, >OFF, and ^T functions are not implemented.

Since the APPLE II is deficient in a cassette record start sequence, the user is required to position the tape at the recorded leader tone before executing the cassette interface software. Thus, APPLE II users may experience difficulty in using ASSM/TED to assemble program modules from tape, and in using the relocationg loader.

ASSM/TED for the APPLE uses BB-F8 of zero page and most of the bottom of the stack (0100 up).

DISK INFORMATIION FOR APPLE II VERSIONS

The disk patches for Apple are provided courtesy of R.F. Suitor, 166 Tremont Street, Newton, Mass. 02158.

NOTE: Do not use >PUT X (end of file mark) if you are using the Apple disk with this ASSM/TED.

DISK EXAMPLES:

#1 Put contents of text buffer to disk with name TEST:

```
>ENTER TEST              ← Open output channel for TEST
>PUT                     ← Put contents to disk
>ENTER                   ← Close output channel
```

#2 Get contents from disk file JUNK and put in text buffer:

```
>LOOKUP JUNK             ← Open input channel for JUNK
>GET                     ← Get and enter in text file
>LOOKUP                  ← Close input channel
```

#3 Put current contents of text buffer to disk with name CONTROL, then type in some more text and put it to CONTROL also:

```
>ENTER CONTROL           ← Open
>PUT F1                  ← Put current contents with file #1
>CLEAR                   ← Clear text buffer
  .                      ← Type in some more text
  .
>PUT F2                  ← Put again with file #2
>ENTER                   ← Close
```

#4 Get the second module (file #2) that was previously put in example #3:

```
>LOOKUP CONTROL
>GET F2                  ← Reads passed file #1 until #2,
                           then puts #2 in text file
```

#5 Assemble file CONTROL that was created in example #3:

```
>CLEAR                   ← Clear text file
>LOOKUP CONTROL          ← Open input channel for CONTROL
>0   .CT                 ← Indicate to continue with
                           subsequent files
>ASSEMBLE                ← Assemble
   .                     ← Some messages may be output
   .
READY FOR PASS 2         ← End of pass message
>LOOKUP                  ← Close channel
>CLEAR                   ← Clear text file
>LOOKUP CONTROL          ← Open for second pass
>0   .CT                 ← Indicate to continue for pass 2
>PASS                    ← Do second pass
  .                      ← Some messages may be output
  .
```

```
.   //xxxx,yyyy,zzzz          ← End of assembly output
    >LOOKUP                   ← Close channel when done
```

#6 Illustrate some APPLE DOS operations:

```
>DC  CATALOG
>DC DELETE TEST
>DC BLOAD PRINTDVR
>DC BSAVE TEST, A$500, L$2A0
```

NOTE: Practically any Apple disk I/O operation may be be performed via the >DC command.

DISK NOTES:

#1 The disk patches will generate two or more entries in the disk catalog for each >ENTER command operation. For each file created, the disk patches will generate a header file in the catalog as S.name and, for each subsequent >PUT, an additional entry as Sxx.name where xx is the file number specified in the >PUT command. For example, the file created in example #3 will appear in the catalog as:

```
T xxx S.CONTROL
B xxx S01.CONTROL
B xxx S02.CONTROL
```

#2 When editing a multi-file number segmented file similar to example #3, consider using the >DUPLICATE command to copy down to a specific numbered segment.

#3 If you press RESET, you should reneter ASSM/TED at 1BC0 so the disk patches can be properly initialized.

Listing 2 (APPLE II)

ASSEMBLE LIST

```
0010 ;***RELOCATING LOADER FOR THE APPLE II ASSM/TED***
0020 ;
0030 ;
0040 ;
0050            .OS
0060 ;
0070 ;*****COPYRIGHT 1979 BY CARL MOSER.*****
0080 ;*****    ALL RIGHTS RESERVED.     *****
0090 ;
0100 ;
0110 ;
0120 ;
0130 ;+++++++  USER INPUTTED VARIABLES BEFORE EXECUTION +++++++
0140 FILE/NO    .DE $0110     ;FILE NUMBER (0-99)
0150 OFFSET     .DE $E0       ;RELOCATOR OFFSET (2 BYTES)
0160 BUFFER     .DE $C8       ;ADDRS. OF R.L. BUFFER
0170 ;
0180 ;
0190 ;
0200 ;            RELOCATOR DIRECTIVES
0210 ;
0220 ;  DIRECTIVE                DESCRIPTION
0230 ;  ---------                -----------
0240 ;    0F            EXTERNAL 2 BYTE ADDRS. PRECEEDS,
0250 ;                  DON'T RELOCATE.  OTHERWISE RELOCATE.
0260 ;
0270 ;    1F            #L,  DATA PRECEEDS.
0280 ;
0290 ;    2F            #H,  DATA PRECEEDS, LO PART FOLLOWS.
0300 ;
0310 ;    3F            .AS OR .HS BYTE FOLLOWS.
0320 ;
0330 ;    4F            .SE OR .SI 2 BYTE ADDRS. FOLLOWS.
0340 ;
0350 ;    5F            TURN RELOCATOR ON (VIA .RS).
0360 ;                  (RESOLVE ADDRESSES AND RELOCATE
0370 ;                   CODE.)
0380 ;
0390 ;    6F            TURN RELOCATOR OFF (VIA .RC).
0400 ;                  (RESOLVE ADDRESSES BUT DO NOT
0410 ;                   RELOCATE CODE.)
0420 ;
0430 ;    7F            .DS - 2 BYTE BLOCK VALUE FOLLOWS.
0440 ;
0450 ;
0460            .BA $0800
0470 ;
0480 ;TAPE INPUT PARMS
0490 LOAD/NO    .DE $0180 0: NO STORE; 1: STORE
0500 TSTART     .DE $3C LOAD BEGINNING AT TSTART
0510 TEND       .DE $3E STOP LOADING AT TEND
0520 ;
0530 ;
0540 ;HEADER INPUT DATA
0550 HFILE/NO   .DE $017A HEADER FILE NUMBER
```

```
                        0560 HSTART      .DE $017B HEADER START
                        0570 HEND        .DE $017D HEADER END
                        0580 ;
                        0590 ;
                        0600 ;VARIABLES
                        0610 SCRAT       .DE $11E SCRATCH AREA
                        0620 TEMP1       .DE $11F SCRATCH AREA
                        0630 TEMP2       .DE $120 SCRATCH AREA
                        0640 SAVE        .DE $121 SCRATCH AREA
                        0650 ADDRS       .DE $DC 4 BYTES OF ADDRESS INFO.
                        0660 BUFF.END    .DE $0123 END OF 256 BYTE BUFFER
                        0670 BUFF.INDEX  .DE $0124 PRESENT ACCESSED DATA FROM BUFFER
                        0680 ;
                        0690 ;
                        0700 ;R(X)=00:  RELOCATOR ON
                        0710 ;R(X)=02:  RELOCATOR OFF
                        0720 ;
                        0730 ;BEGIN EXECUTION AT LABEL START
                        0740 ;
0200- A2 FF             0750 START       LDX #$FF
0202- 9A                0760             TXS INITIALIZE STACK
0203- E8                0770             INX R(X)=00: SET RELOCATOR INITIALLY TO ON
0204- 20 86 8B          0780             JSR ACCESS
0207- D8                0790             CLD
0208- 8E 21 01          0800             STX SAVE R(X)=00
020B- 20 E6 02          0810             JSR LOAD+BUFF
020E- 4C 14 02          0820             JMP ENTY
0211- 20 74 03          0830 LOOP1       JSR GET+DATA
                        0840 ;
0214- C9 7F             0850 ENTY        CMP #$7F      ;CKG. FOR .DS
0216- D0 03             0860             BNE PRO.3F
0218- 4C AA 03          0870             JMP PRO.7F    ;JUMP TO PROCESS DIR. 7F
021B- C9 3F             0880 PRO.3F      CMP #$3F CKG. FOR RELOCATOR DIRECTIVE
021D- D0 0B             0890             BNE OP+CKG
021F- 20 74 03          0900             JSR GET+DATA
0222- 81 DC             0910             STA (ADDRS,X)
0224- 20 88 03          0920             JSR INC+ADDRS
0227- 4C 11 02          0930             JMP LOOP1
022A- C9 4F             0940 OP+CKG      CMP #$4F CKG. FOR .SE, .SI
022C- D0 03             0950             BNE W:
022E- 4C AD 02          0960             JMP TWO+BYT+AD
0231- C9 5F             0970 W:          CMP #$5F CKG. FOR RELOCATOR ON
0233- D0 04             0980             BNE CKNX
0235- A2 00             0990             LDX #$00
0237- F0 D8             1000             BEQ LOOP1
                        1010 ;
0239- C9 6F             1020 CKNX        CMP #$6F CKG. FOR RELOCATOR OFF
023B- D0 04             1030             BNE NO+REL
023D- A2 02             1040             LDX #$02
023F- D0 D0             1050             BNE LOOP1
0241- 81 DC             1060 NO+REL      STA (ADDRS,X) STORE OP CODE
0243- 20 88 03          1070             JSR INC+ADDRS
0246- C9 00             1080             CMP #$00 CKG. FOR BRK INSTR.
0248- F0 C7             1090             BEQ LOOP1
024A- C9 20             1100             CMP #$20 CKG. FOR JSR INSTR.
024C- F0 5F             1110             BEQ TWO+BYT+AD
024E- 8D 21 01          1120             STA SAVE SAVE R(A), IT CONTAINS OP CODE
0251- 29 9F             1130             AND #$9F
```

```
0852- AD 21 01  1140            LDA SAVE RESTORE OP CODE
0855- 29 1D     1150            AND #$1D
0857- C9 08     1160            CMP #$08 ↑↑KG. FOR ONE BYTE INSTR.
0859- F0 B3     1170            BEQ LOOP1
085B- C9 18     1180            CMP #$18 CKG. FOR ONE BYTE INSTR.
085D- F0 AF     1190            BEQ LOOP1
                1200 ;
                1210 ;NOW, TEST FOR INSTR. CONTAINING 2 BYTES
                1220 ;OF ADDRESS INFORMATION
                1230 ;
085F- AD 21 01  1240            LDA SAVE RESTORE OP CODE
0862- 29 1C     1250            AND #$1C
0864- C9 1C     1260            CMP #$1C
0866- F0 42     1270            BEQ TWO←BYT←AD
0868- C9 18     1280            CMP #$18
086A- F0 3E     1290            BEQ TWO←BYT←AD
086C- C9 0C     1300            CMP #$0C
086E- F0 3A     1310            BEQ TWO←BYT←AD
                1320 ;
                1330 ;THE REMAINING CONTAIN ONE BYTE OF
                1340 ;ADDRESS INFORMATION
                1350 ;
                1360 ;PROCSSING OF ON BYTE ADDRESSES AND IMMEDIATE DATA
0870- 20 71 09  1370 ONE←BYT←AD JSR GET←DATA
0873- 81 DC     1380            STA (ADDRS,X)
0875- 20 85 09  1390            JSR INC←ADDRS
0878- 20 71 09  1400            JSR GET←DATA
087B- C9 2F     1410            CMP #$2F CKG. FOR RELOCATOR DIRECTIVE
087D- F0 14     1420            BEQ IMM←HI CKG. FOR #H,
087F- C9 1F     1430            CMP #$1F CKG. FOR RELOCATOR DIRECTIVE
0881- D0 8E     1440            BNE ENTY
                1450 ;
                1460 ;PROCESS #L,  DATA FOR RELOCATION
0883- 20 92 09  1470 IMM←LO     JSR DEC←ADDRS
0886- 18        1480            CLC
0887- A1 DC     1490            LDA (ADDRS,X)
0889- 65 E0     1500            ADC ♦OFFSET+$00 ADD OFFSET LOW PART FOR #L,
088B- 81 DC     1510            STA (ADDRS,X)
088D- 20 85 09  1520            JSR INC←ADDRS
0890- 4C 0E 08  1530 BACK←TO←L1 JMP LOOP1
                1540 ;PROCESS #H,  DATA FOR RELOCATION
0893- 20 71 09  1550 IMM←HI     JSR GET←DATA LOW BYTE FOLLOWS REL. DIR.
0896- 18        1560            CLC
0897- 65 E0     1570            ADC ♦OFFSET FORM THE LO ADDRS. PART
0899- 08        1580            PHP
089A- 20 92 09  1590            JSR DEC←ADDRS
089D- 28        1600            PLP
089E- A1 DC     1610            LDA (ADDRS,X)
08A0- 65 E1     1620            ADC ♦OFFSET+$1 NOW FORM THE EFFECTIVE #H,
08A2- 81 DC     1630            STA (ADDRS,X)
08A4- 20 85 09  1640            JSR INC←ADDRS
08A7- 4C 0E 08  1650            JMP LOOP1
                1660 ;
                1670 ;PROCESSING OF TWO BYTE ADDRESSES
08AA- A0 02     1680 TWO←BYT←AD LDY #$02
08AC- 98        1690 XX         TYA
08AD- 48        1700            PHA SAVE R(Y)
08AE- 20 71 09  1710            JSR GET←DATA
```

```
08B1- 81 DC      1720              STA (ADDRS,X)
08B3- 20 85 09   1730              JSR INC←ADDRS
08B6- 68         1740              PLA
08B7- A8         1750              TAY RESTORE R(Y)
08B8- 88         1760              DEY
08B9- D0 F1      1770              BNE XX
08BB- 20 71 09   1780              JSR GET←DATA
08BE- C9 0F      1790              CMP #$0F CKG. FOR RELOCATOR DIRECTIVE
08C0- D0 03      1800              BNE XY
08C2- 4C 0E 08   1810              JMP LOOP1
08C5- 48         1820 XY           PHA
08C6- 20 92 09   1830              JSR DEC←ADDRS
08C9- 20 92 09   1840              JSR DEC←ADDRS
                 1850 ;DECREMENT BACK TO ADDRESS START
                 1860 ;
08CC- A1 DC      1870              LDA (ADDRS,X)
08CE- 18         1880              CLC
08CF- 65 E0      1890              ADC ←OFFSET ADD OFFSET LO
08D1- 81 DC      1900              STA (ADDRS,X)
08D3- 20 85 09   1910              JSR INC←ADDRS
08D6- A1 DC      1920              LDA (ADDRS,X)
08D8- 65 E1      1930              ADC ←OFFSET+$1 ADD OFFSET HI
08DA- 81 DC      1940              STA (ADDRS,X)
08DC- 20 85 09   1950              JSR INC←ADDRS
08DF- 68         1960              PLA
08E0- 4C 11 08   1970              JMP ENTY
                 1980 ;
                 1990 ;SUBROUTINE LOAD BUFFER WITH DATA FROM TAPE
                 2000 ;
08E3- A9 7A      2010 LOAD←BUFF    LDA #$7A ADDLO OF START OF HEADER
08E5- 8D 3C 00   2020              STA TSTART+$00
08E8- A9 7F      2030              LDA #$7F ADDLO OF END OF HEADER
08EA- 8D 3E 00   2040              STA TEND+$00
08ED- A9 01      2050              LDA #$01 HI ADDRS
08EF- 8D 3D 00   2060              STA TSTART+$01
08F2- 8D 3F 00   2070              STA TEND+$01
08F5- 8D 80 01   2080              STA LOAD/NO 01: INDICATE TO LOAD
08F8- 20 D2 09   2090              JSR USER/LOAD USER LOA←BD FROM TAPE ROUTINE
                 2100 ;
                 2110 ;THE ABOVE SETS UP AND LOADS HEADER INFORMATION
                 2120 ;FROM TAPE.  THE HEADER CONTAINS THE MODULE FILE
                 2130 ;NUMBER, AND STARTING AND ENDING ADDRESS OF FOLLOWING
                 2140 ;DATA.
                 2150 ;
                 2160 ;
08FB- D0 4D      2170              BNE ERROR IF Z-BIT FALSE, THEN ERROR IN LOADING
08FD- A2 00      2180              LDX #$00
                 2190 ;
08FF- AD 7D 01   2200              LDA HEND+$00
0902- 38         2210              SEC
0903- ED 7B 01   2220              SBC HSTART+$00
                 2230 ;CALCULATE NUMBER OF BYTES IN FOLLOWING DATA
                 2240 ;
0906- 8D 23 01   2250              STA BUFF.END INITIALIZE BUFFER END POINTER
0909- AD 7E 01   2260              LDA HEND+$01
090C- ED 7C 01   2270              SBC HSTART+$01
090F- D0 39      2280              BNE ERROR ONLY 256 BYTE BUFFER ALLOWED
0911- A5 C8      2290              LDA ←BUFFER
```

```
0913- 8D 3C 00  2300            STA TSTART
0916- 18        2310            CLC
0917- 6D 23 01  2320            ADC BUFF.END # BYTES
091A- 8D 3E 00  2330            STA TEND
091D- A5 C9     2340            LDA *BUFFER+$01
091F- 8D 3D 00  2350            STA TSTART+$01
0922- 69 00     2360            ADC #$00
0924- 8D 3F 00  2370            STA TEND+$01
                2380 ;NOW THE START AND END ADDRESS PARMS HAVE BEEN
                2390 ;SET UP TO LOAD FROM TAPE INTO THE BUFFER.
                2400 ;
0927- AD 10 01  2410            LDA FILE/NO USER ENTERED FILE NUMBER
092A- F0 08     2420            BEQ STORE.DATA IF F# = 00, LOAD ANYWAY
092C- CD 7A 01  2430            CMP HFILE/NO CMP WITH USER VERSUS THAT ON TAPE
092F- F0 03     2440            BEQ STORE.DATA
0931- 8E 80 01  2450            STX LOAD/NO R(X)=0; NO STORE
0934- 20 D2 09  2460 STORE.DATA JSR USER/LOAD
                2470 ;
                2480 ;THE ABOVE LOADS IN DATA INTO BUFFER DEPENDING
                2490 ;ON THE STATE OF LOAD/NO
                2500 ;
0937- D0 11     2510            BNE ERROR Z-BIT = FALSE THEN ERROR
0939- A2 00     2520            LDX #$00
093B- AD 7A 01  2530            LDA HFILE/NO
093E- C9 EE     2540            CMP #$EE COMPARE IF END OF FILE
0940- D0 0C     2550            BNE BUFFLOADED
0942- A9 00     2560            LDA #$00 INDICATE GOOD LOAD
0944- 00        2570 B          BRK
0945- EA        2580            NOP
0946- EA        2590            NOP
0947- 4C 00 08  2600            JMP START
094A- A9 EE     2610 ERROR      LDA #$EE INDICATE ERROR IN LOAD
094C- D0 F6     2620            BNE B
                2630 ;
                2640 ;
                2650 ;NOW GET ADDRS. INFO. AND PUT IN ADDRS+$2, +$3
                2660 ;ADDRS INFO. IS IN FIRST TWO BYTES OF BUFFER
                2670 ;
094E- AD 80 01  2680 BUFFLOADED LDA LOAD/NO CKG. IF PROPER DATA
0951- F0 90     2690            BEQ LOAD+BUFF
0953- AE 21 01  2700            LDX SAVE RESTORE R(X)
0956- A0 00     2710            LDY #$00
0958- B1 C8     2720            LDA (BUFFER),Y
095A- 85 DE     2730            STA *ADDRS+$2
095C- C8        2740            INY
095D- B1 C8     2750            LDA (BUFFER),Y
095F- 85 DF     2760            STA *ADDRS+$3
0961- 8C 24 01  2770            STY BUFF.INDEX SET BUFFER DATA POINTER
                2780 ;
                2790 ;SET RELOCATION ADDRS. IN ADDRS+$0, +$1
0964- A5 DE     2800            LDA *ADDRS+$2
0966- 18        2810            CLC
0967- 65 E0     2820            ADC *OFFSET
0969- 85 DC     2830            STA *ADDRS
096B- A5 E1     2840            LDA *OFFSET+$1
096D- 65 DF     2850            ADC *ADDRS+$3
096F- 85 DD     2860            STA *ADDRS+$1
                2870 ;
```

```
0971- 8E 21 01  2880 GET←DATA   STX SAVE SAVE X IN CASE WE BR. TO LOAD←BUFF
0974- EE 24 01  2890            INC BUFF.INDEX INC. 256 BYTE BUFFER POINTER
0977- AC 24 01  2900            LDY BUFF.INDEX
097A- CC 23 01  2910            CPY BUFF.END
097D- 90 03     2920            BCC WX BR. IF NOT AT END OF DATA IN BUFFER
097F- 4C E3 08  2930            JMP LOAD←BUFF RELOAD BUFFER
0982- B1 C8     2940 WX         LDA (BUFFER),Y
0984- 60        2950            RTS
                2960 ;
                2970 ;
                2980 ;INCREMENT ADDRS+$0, +$1 AND ADDRS+$2, +$3
                2990 ;
0985- E6 DC     3000 INC←ADDRS  INC ←ADDRS
0987- D0 02     3010            BNE SKIP←INC1
0989- E6 DD     3020            INC ←ADDRS+$1
098B- E6 DE     3030 SKIP←INC1  INC ←ADDRS+$2
098D- D0 02     3040            BNE SKIP←INC2
098F- E6 DF     3050            INC ←ADDRS+$3
0991- 60        3060 SKIP←INC2  RTS
                3070 ;
                3080 ;
                3090 ;DECREMENT ADDRS+$0, +1 AND ADDRS+$2, +$3
                3100 ;
0992- C6 DC     3110 DEC←ADDRS  DEC ←ADDRS
0994- A5 DC     3120            LDA ←ADDRS
0996- C9 FF     3130            CMP #$FF
0998- D0 02     3140            BNE SKIP←DEC1
099A- C6 DD     3150            DEC ←ADDRS+$1
099C- C6 DE     3160 SKIP←DEC1  DEC ←ADDRS+$2
099E- A5 DE     3170            LDA ←ADDRS+$2
09A0- C9 FF     3180            CMP #$FF
09A2- D0 02     3190            BNE SKIP←DEC2
09A4- C6 DF     3200            DEC ←ADDRS+$3
09A6- 60        3210 SKIP←DEC2  RTS
                3220 ;
                3230 ;
                3240 ;7F LO HI  --  PCL PCH 7F LO HI
                3250 ;
09A7- 20 71 09  3260 PRO.7F     JSR GET←DATA
09AA- 48        3270            PHA  ;SAVE LO
09AB- 20 71 09  3280            JSR GET←DATA
09AE- A8        3290            TAY  ;SAVE HI IN R(Y)
09AF- AD 24 01  3300            LDA BUFF.INDEX
09B2- C9 05     3310            CMP #$05      ;NO PROC. IF <= 4
09B4- 90 18     3320            BCC NO.PROC
09B6- 18        3330 PROC.DS    CLC
09B7- 68        3340            PLA  ;GET LO
09B8- 48        3350            PHA
09B9- 65 DC     3360            ADC ←ADDRS
09BB- 85 DC     3370            STA ←ADDRS
09BD- 98        3380            TYA  ;GET HI
09BE- 65 DD     3390            ADC ←ADDRS+1
09C0- 85 DD     3400            STA ←ADDRS+1
09C2- 68        3410            PLA
09C3- 48        3420            PHA  ;GET LO
09C4- 18        3430            CLC
09C5- 65 DE     3440            ADC ←ADDRS+2
09C7- 85 DE     3450            STA ←ADDRS+2
```

```
09C9- 98          3460              TYA  ;GET HI
09CA- 65 DF       3470              ADC *ADDRS+3
09CC- 85 DF       3480              STA *ADDRS+3
09CE- 68          3490 NO.PROC      PLA
09CF- 4C 0E 08    3500              JMP LOOP1
                  3510 ;
                  3520 ;
                  3530 ;
                  3540 ;      *** APPLE II CASSETTE INTERFACE PATCH ***
                  3550 ;
                  3560 ;
                  3570 ;APPLE DEFINITIONS:
                  3580 READ         .DE $FEFD     ;READ FROM TAPE
                  3590 ;
                  3600 ;
09D2- 20 FD FE    3610 USER/LOAD    JSR READ      ;READ FROM TAPE
09D5- A2 00       3620              LDX #00
09D7- 60          3630              RTS
                  3640 ;
                  3650 ;
                  3660 END.PGM      .EN
```

LABEL FILE: [/ = EXTERNAL]

```
/FILE/NO=0110           /OFFSET=00E0          /BUFFER=00C8
/LOAD/NO=0180           /TSTART=003C          /TEND=003E
/HFILE/NO=017A          /HSTART=017B          /HEND=017D
/SCRAT=011E             /TEMP1=011F           /TEMP2=0120
/SAVE=0121              /ADDRS=00DC           /BUFF.END=0123
/BUFF.INDEX=0124        START=0800            LOOP1=080E
ENTY=0811               PRO.3F=0818           OP+CKG=0827
W:=082E                 CKNX=0836             NO+REL=083E
ONE+BYT+AD=0870         IMM+LO=0883           BACK+TO+L1=0890
IMM+HI=0893             TWO+BYT+AD=08AA       XX=08AC
XY=08C5                 LOAD+BUFF=08E3        STORE.DATA=0934
B=0944                  ERROR=094A            BUFFLOADED=094E
GET+DATA=0971           WX=0982               INC+ADDRS=0985
SKIP+INC1=098B          SKIP+INC2=0991        DEC+ADDRS=0992
SKIP+DEC1=099C          SKIP+DEC2=09A6        PRO.7F=09A7
PROC.DS=09B6            NO.PROC=09CE          /READ=FEFD
USER/LOAD=09D2          END.PGM=09D8
//0000,09D8,09D8
>
```

14. MACRO ASSM/TED LEARNING AID

<Examples>

This section contains examples which illustrate the many powerful operations that can be performed using the PET, APPLE II, SYM, and KIM versions of Macro ASSM/TED.

We recommend that you first read this entire manual and become familiar with its contents. Then, read this learning aid carefully for a detailed understanding. When using this aid, make frequent references to other parts of the manual for futher understanding. Learning is an iterative process. You may find yourself advancing rapidly thru this material and later having to fall back and reread previously covered material. This may be because you forgot something or we did not properly cover the subject.

In some of the examples that follow, we recommend that you actually type in the commands and try them. You will find that learning from actually "doing" reduces the mental overhead caused by dealing with abstract concepts.

So, lets begin .....

Text Editor Examples <Try These!>

#1 Cold start the ASSM/TED so all parameters are properly initialized:

```
.G 2000                    (or  *2000G   if APPLE)
```

NOTE: Always cold start the ASSM/TED as the first entry after you load the ASSM/TED tape and when you think things are all messed up.

#2 Cause an error to occur so we can see what an error message is like:

```
>XX                    ← You type garbage such as XX and return
!ED AT LINE xxxxx      ← And this error message will appear
```

#3 Change the text file memory allocation to $0800 - $1FFC, the label file to $4000 - $46FC, and relocatable buffer to $4700.

```
>SET $800 $1FFC $4000 $46FC $4700
```

NOTE: The actual value you choose will vary depending on amount of memory you have available. Do not issue this command if you do not have memory in the indicated areas.

#4 Go to monitor and then reneter the ASSM/TED at warm start:

```
>BREAK
```

The monitor greeting message appears (Registers displayed or whatever

```
.G 2090 (for PET)    *2003G (for APPLE)    .G 2003 (for KIM, SYM)
```

NOTE: The warm start entry leaves the text file, label file, and associated parameters intact. This is the non-destructive entry.

#5 Enter a short subroutine in the text file using the auto line #-ing feature. Type this in as shown as later examples refer to it.

```
>AUTO 10
>1000;THIS PROGRAM ADDS 06 TO THE ACCUMULATOR
1010>;
1020> .BA $800 ;ASSMEMBLE MACHINE CODE (OR OBJECT) AT HEX 800
1030> .OS ;INDICATE TO STORE IN MEMORY
1040>;
1050>ADD CLC ;SUBROUTINE ADD
1060 ADC #06
1070ENDADD RTS
1080//                ← Use // to exit auto #-ing

>AUTO 0               ← Use to disable auto #-ing
```

#6 Now print out the program:

```
>PRINT

.
.           ← Program appears properly tabulated
.
```

#7 Now print out the program but unformatted (i.e. without tabulating):

```
>FORMAT CLEAR
>PRINT

.
.           ← Program appears but untabulated
```

#8 Now lets go back to automatic tabulating:

```
>FORMAT SET
```

#9 Lets output just the last line:

```
>PR /
```

#10 Renumber the text file by 10:

```
>NU 0 10
```

#11 Renumber the text file with line # 100 as the first # with increment of 10:

```
>NU 0 90
>NU 90 10
```

Note: The first >NU renumbers by 90 with the first line # = 90. The second >NU will add 10 to line 90 giving 100 as the first line #.

#12 Put the contents of the text file to tape with an identification as file #3:

First insert cassette, position, press play and record, then type:

```
>PUT F3          if PET then type >PUT T1 F3 "program name"
```

#13 Assemble with no listing:

```
>ASSEMBLE

.
.

//0000,0804,0804          ← This message appears when done
    ↑    ↑    ↑
    ↑    ↑    ------------ This is last object code relative to .MC.
    ↑    ↑                 If no .MC then it will be same as
    ↑    ↑                 relative to .BA.
    ↑    ↑
    ↑    ------------------ This is last object code relative to .BA.
    ↑                      (End of your machine language program.)
    ↑
    ----------------------- # of errors
```

#14 Assemble with listing:

```
>ASSEMBLE LIST

.
.     ← Listing appears
```

#15 Output the label file (or Symbol Table)

```
>LABELS
```

#16 Run the program entered in example #5

```
>RUN ADD          ← Note: There will be no indication that
                          the program did anything since
                          no ouputting was performed.
  or
>RUN $800         ← Start address in hex
```

or

```
>RUN 2048        ← Start address in decimal
```

#17 Now lets cause an error to occur and see what happens:

For APPLE, SYM, KIM:

Change the . in line 120 to * using >EDIT form 2

```
>EDIT 120
0120 .BR $800          ← Line appears

↑F  >  .  ↑H  *        ← You press underlined part: Press control F to
--  -  -  --  -           change to search prompter (>). Then enter
                          the search character (.), then control H
                          to backspace, then asterisk (*) which
                          replaces the period. Press return when done.
0120 *BR $800          ← The corrected line will be output.
```

Now this may seem confusing, but you should study it carefully as you'll find it an invaluable aid later on.

For PET:

Cursor up to the period in line 120, change to an asterisk (*), and then press return.

#18 Now lets assemble with the error and see what happens:

```
>ASSEMBLE

!02 AT LINE 0120        ← Illegal mnemonic error message
                            (A list of error messages is contained
                             in the manual.)
```

#19 Note that the assembly stopped at line 0120. Sometimes we would like for the assembly to continue on errors so all mistakes in the file will be identified. We can instruct the assembler to continue with errors using the .CE pseudo op. (Note: The assembler can not handle some types of errors and will stop, but for most it will continue if you specify a .CE.)

Insert the .CE pseudo op at line 10:

```
>10 .CE
```

#20 Now assemble again

>ASSEMBLE

Note that the assembler continues assembling with errors.

#21 Now lets do something else. Find and output all occurrances of the phrase THIS PROGRAM.

>FIND /THIS PROGRAM/

The line with the phrase will be ouput followed with the # of occurrances.

#22 Find and ouput all occurrances of the letter A:

>FIND *A* ← Note that the terminator is *

The line with the phrase will be output followed with the # of occurrances.

#23 Find all occurrances of the letter A but just show count:

>FIND /A/#

#24 Replace the phrase THIS PROGRAM with THIS STUPID PROGRAM using edit form 1.

>EDIT /THIS PROGRAM/THIS STUPID PROGRAM/

search string ↑ ↑ replacement string

Note, after replacement the edited line will be output.

#25 Replace all occurrances of the character A with X between lines 100 and 150:

>EDIT /A/X/ 100 150

Note, lines will be output with the replaced string.

#26 Now lets change the text file back but show no output:

>EDIT /X/A/# 100 150

#27 This example does not apply to PET versions:
The phrase ADD appears in several places in the text file. Set up to search for ADD and conditionally replace with SUBTRACT:

```
>EDIT /ADD/SUBTRACT/*

xx 1000 ;THIS PROGRAM ADDS 06 TO THE ACCUMULATOR
↑
--- Count in hex to start of search string

*   <------ Experiment with the following one letter commands:
            A - Go ahead and alter string
            S - Skip over this line
            D - Delete this line
           ↑F - Enter edit form 2   (↑F is control F)
            M - Move to next field
            X - Exit this command operation
```

#28 Now lets clear the text file:

```
>CLEAR
```

#29 Lets reload the text file with that saved on tape in example # 5 via the following procedure:

```
Rewind cassette, press play
>GET F3       if PET then >GET T1 F03 "program name"

F03  xxxx  yyyy-zzzz    ← Status message output during loading
 ↑    ↑        ↑
 ↑    ↑     Range where loaded in memory
 ↑  Length of file
File loaded
```

#30 Copy lines 100 thru 120 to after line 900

```
>COPY 900 100 120
```

#31 Delete lines 140 thru 150

```
>DELETE 140 150
```

#32 Move lines 100 thru 120 to after line 160

```
>MOVE 160 100 120
```

Now that we have tried the basics, lets cover some more sophisticated operations.
Note: The following are not try by example:

a) Search for all occurrances of phrases LOOP01, LOOa01, LOOK01, or any phrase with LOO followed with some character we don't care about, then 01:

```
>FIND /LOOP%01/    ← % is the don't care character
```

b) Search for all occurrances of period, some pair of #'s, and then %. Here we have a delima - % is our don't care character. But wait, it can be changed as follows:

```
>FIND /.XX%/ %X
              ↑
        Here we change the don't care from % to X
```

c) Search for all slashes (/) in text file:

```
>FIND A/A       ← Note that A is the string terminator
```

d) Always put an end of file mark at the end of tape modules as various commands key on this mark to terminate operation at the logical end of tape. You can put an end of file mark as follows:

```
>PUT X
```

e) This example applies to all except the PET version: Assume you have a large program with for example 10 modules recorded on tape with file #'s 1 thru 10, and you have found an error in file # 6. How do you correct the error? Well you insert the tape containing the 10 modules in tape deck # 1, then insert a blank tape in deck # 0, and peform the following:

Press play on deck # 1
Press play and record on deck # 0

```
>DUPLICATE F6        ← This reads from deck 1 and writes to deck 0
                       all files down to but not including file # 6.
                       When done, file # 6 will be in the text file
                       ready for editing.
```

Correct the text and then type: >PUT F6

```
>DUPLICATE F99       ← This duplicates down to file 99 or end of
                       file mark - whichever comes first.
```

f) If you want to append say F12 to the contents of the text buffer, do the following:

```
>GET F12 APPEND      or for PET:  >GET T1 F12 "program name"
```

g) If you use ASSM/TED for generating text, letters, etc.,then you should set up as follows:

```
>FORMAT CLEAR     ← To prevent tabulating on fields
>MANUSCRIPT SET   ← To suppress output of line #'s
```

Assembler Examples

#1 Begin assembly at $1000 and store object code.

```
.BA  $1000
.OS
```

#2 Begin assembly at $1000 but store object code at $4000.

```
.BA  $1000
.MC  $4000
.OS
```

#3 Defind a ROM routine.

```
ROM1   .DE  $FFD2
```

#4 Assign an internal work location in zero page.

```
WORK  .DI  $0
```

#5 Allocate 6 bytes of storage.

```
TABLE  .DS  6
```

#6 Define label EOI as mask with bit 6 set and show use in AND statement.

```
EOI     .DE  %01000000
        AND  #EOI
```

#7 Load the low address part of the label VALUES in register X and high part in register Y.

```
LDX  #L,VALUES
LDY  #H,VALUES
```

#8 Give example of .BY pseudo op.

```
.BY  'ALARM CONDITION ON MOTOR 1'  $0D  $0A
```

#9 Store the address of the internal label TABLE and the external label OUT1.

```
.SI  TABLE
.SE  OUT1
```

#10 NOTE: This example does not apply to PET versions: Define the contents of the text file as Macro Global so its macro definitions can be used by subsequent files in the assembly.

```
.MG
```

NOTE: This locks the macro definitions in the text buffer. If you get a !0F error on subsequent loads, you should know that you have overflowed the text buffer. The solution is to allocate more memory (via)SET command) and then reassemble.

#11 Show example of a very long label.

```
        MEMORY.TEST.FOR.6502
                JMP MEMORY.TEST.FOR.6502
```

NOTE: Long labels (greater than that specified via JFO command) are allowed if defined on a line with no mnemonics.

#12 Define the 6502 reset vector so the relocating loader will not alter the address during loading.

```
    RESET  .DE $FFFC      ;6502 RESET VECTOR
             .
             .
           LDA RESET
```

-- OR --

```
           LDA $FFFC
```

The following examples can be entered and actually tried, but you will need to enter the following source sequence to be used with each example:

```
 10          .BA $800    ;ASSEMBLE OBJECT CODE AT $0800
0020         .OS         ;INDICATE TO ASSM TO STORE OBJECT CODE
0030LOC      .DS 1       ;RESERVE ONE BYTE OF STORAGE FOR LOC
0040;
0050WRT.     .DE see note  ;WRITE ASCII TO SCREEN ROUTINE
0060TBYT     .DE see note  ;WRITE BYTE AS 2 HEX DIGITS
0070RDT.     .DE see note  ;INPUT ASCII FROM KEYBOARD ROUTINE

             .
             .
    insert code from example here
             .
             .

9000;
9010         RTS         ;RTS WILL CAUSE A RETURN TO ASSM/TED
9020;
9030         .EN         ;INDICATE END OF ASSEMBLY
```

NOTE: Enter the following for the above 50-70 lines depending on your type of microcomputer.

MICROCOMPUTER	WRT.	TBYT.	RDT.
* PET (old)	$FFD2	$0613	$065E
PET (new)	$FFD2	$E775	$FFCF
** APPLE	$FDED	$FDDA	$FD0C
SYM	$A663	$82FA	$A660

KIM	$1EA0	$1E3B	$1E5A

* = Insure that you have loaded you machine language monitor.
** = This will display in inverse video. For normal video, provide an ORA #$80 before each JSR WRT. instruction.

After you have entered the source for the example you are trying, type >ASSEMBLE to assemble the program. If you get no errors, then valid object code is stored starting at $0800. If you get errors, correct them and then assemble again. Repeat this procedure until you have an error-free assembly.

When you get an error-free assembly, type >RUN TEST to execute the program example. Note that TEST is the label where we want the program to begin execution.

When you complete an example, use the >DELETE 80 8000 command to delete the lines associated with the example. This will leave the required setup intact for the next example.

If an example uses the BRK instruction, then you should use your monitor to examine the registers and/or memory to verify the operation. Then type G to your monitor to reenter ASSM/TED.

#13 Load R(A) with hex F3, R(X) with decimal 96, and R(Y) with binary 110110:

```
0080TEST        LDA #$F3            ;$ MEANS HEX
0090            LDX #96             ;NO SPECIAL SYMBOL = DECIMAL
0100            LDY #%110110        ;% = BINARY
0110            BRK             ;ENTER MONITOR TO EXAMINE REGISTERS
```

#14 Load R(A) with hex 69 and output to screen:

```
0080TEST        LDA #$69            ;LOAD R(A) WITH 69
0090            JSR TBYT            ;WRITE TO SCREEN
```

#15 Decimal mode arithmetic - Store 25 at location LOC and then add 06 to LOC:

```
0080TEST        SED                 ;SET DECIMAL MODE ARITHMETIC
0090            LDA #$25            ;STORE 25 AT LOC
0100            STA LOC             ;  *
0110            CLC                 ;CLEAR CARRY BEFORE ADDING
0120            ADC #06             ;ADD 06 TO R(A)
0130            STA LOC             ;AND PUT IN LOC
0140            BRK         ;BRK TO EXAMINE LOCATION LOC
0150;  OBSERVE THAT R(A) AND LOC CONTAIN 31
```

#16 Subtract 14 from LOC:

```
0080TEST        SED                 ;SET DECIMAL MODE ARITHMETIC
```

```
0090            SEC                 ;SET CARRY BEFORE SUBTRACTING
0100            LDA LOC
0110            SUB #$14
0120            STA LOC             ;RESULT IN R(A) AND LOC
0130            BRK
0140;  OBSERVE THAT R(A) AND LOC CONTAIN 17
```

#17 Hex mode arithmetic - Store hex 26 at location LOC and then add hex 06 to LOC:

```
0080TEST        CLD                 ;CLEAR DECIMAL MODE - ENTER HEX
0090            LDA #$26
0100            STA LOC
0110            CLC
0120            ADC #$06
0130            STA LOC             ;RESULT IN R(A) AND LOC
0140            BRK
```

#18 Hex mode arithmetic - Subtract hex 14 from LOC:

```
0080TEST        SED
0090            SEC
0100            LDA LOC
0110            SUB #$14
0120            STA LOC
0130            BRK
```

#19 Store $47 in location LOC and then increment:

```
0080TEST        LDA #$47
0090            STA LOC             ;LOC = 47
0100            INC LOC             ;NOW LOC = 48
0110            BRK                 ;INC INSTR. ALWAYS HEX MODE
```

#20 Store $47 in location LOC and then decrement:

```
0080TEST        LDA #$47
0090            STA LOC             ;LOC = 47
0100            DEC LOC             ;NOW LOC =46
0110            BRK                 ;DEC INSTR. ALWAYS HEX MODE
```

#21 Load R(A) with ascii X and then output to screen:

```
0080TEST        LDA #'X
0090            JSR WRT.
```

#22 Load R(X) with 8 and output the character A eight times:

```
0080TEST        LDX #8              ;8 TIMES
0090LOOP        LDA #'A             ;CHARACTER TO OUPUT IS A
0100            JSR WRT.            ;WRITE TO SCREEN
0110            DEX                 ;DECREMENT R(X)
0120            BNE LOOP            ;LOOP 8 TIMES (UNTIL R(X) = 0)
```

#23 Define two locations TEMP1 and TEMP2, preload with 4F and 03, then add in hex mode and display on screen:

```
0080TEMP1       .BY $4F
```

```
0090TEMP2       .BY $03
0100;
0110TEST        CLD
0120            CLC
0130            LDA TEMP1
0140            ADC TEMP2
0150            JSR TBYT            ;52 SHOULC BE DISPLAYED
```

#24 Store the message "THIS IS YOUR LIFE!" and then output to screen:

```
0080MESSAGE     .BY 'THIS IS YOUR LIFE!'  $0D $0A 0
0090;
0100TEST        LDY #00             ;INDEX TO START OF MESSAGE
0110LOOP        LDA MESSAGE,Y       ;GET CHARACTER
0120            BEQ DONE            ;IF 00 BYTE THEN DONE
0130            STY LOC             ;SAVE R(Y)
0140            JSR WRT.            ;WRITE TO SCREEN
0150            LDY LOC             ;RESTORE R(Y)
0160            INY                 ;INCREMENT FOR NEXT CHAR.
0170            BNE LOOP            ;LOOP FOR NEXT CHAR.
0180DONE
```

#25 Set up two messages and let user select message via keyboard input:

```
0080MESSAGES
0090A           .BY 'THE WORLD IS ROUND!' $0D $0A $00
0100B           .BY 'THE WORLD IS FLAT!' $0D $0A $00
0110INTRO       .BY 'INPUT EITHER A OR B? ' $0D $0A $00
0120;
0130SAVEX       .DS 1       ;SAVE AREA FOR R(X)
0140;
0150TEST        LDX #INTRO-MESSAGES ;INDEX FOR "INPUT.." MESS.
0160            JSR MESS.OUT        ;PRINT MESSAGE
0170            JSR RDT.            ;GET KEYBOARD INPUT
0180            AND %01111111       ;MASK OUT BIT 7
0190            CMP #'A             ;WAS THE INPUT A?
0200            BEQ DO←A            ;BRANCH IF IT WAS A
0210            CMP #'B             ;WAS THE INPUT B?
0220            BEQ DO←B            ;BRANCH IF IT WAS B
0230            BNE TEST            ;BRANCH IF NEITHER A OR B
0240;
0250DO←A        LDX #A-MESSAGES     ;SET INDEX FOR "...ROUND!"
0260            JSR MESS.OUT        ;PRINT MESSAGE
0270            RTS
0280;
0290DO←B        LDX #B-MESSAGES     ;SET INDEX FOR "...FLAT!"
0300            JSR MESS.OUT        ;PRINT MESSAGE
0310            RTS
0320;
0330;SUBOUTINE WHICH OUTPUTS A MESSAGE STRING TO CRT
0340;R(X) POINTS TO START OF TEXT
0350MESS.OUT    LDA MESSAGES,X      ;GET CHARACTER
0360            BEQ DONE            ;IF 00 THEN END OF MESSAGE
0370            STX SAVEX           ;PRESERVE R(X)
0380            JSR WRT.            ;PUT TO SCREEN
0390            LDX SAVEX           ;RESTORE R(X)
0400            INX                 ;INCREMENT FOR NEXT CHAR.
0410            BNE MESS.OUT        ;LOOP UNTIL DONE
```

SPECIFIC LOCATIONS IN MACRO ASSM/TED

APPLE/SYM/KIM = $38D4
PET = $37E2

At this location, three NOP instructions are provided where the user may insert a JSR to a printer driver routine which outputs the ASCII character in register A. Outputting to the printer is turned on via >HA SET, or turned off via >HA CLEAR.

APPLE/SYM/KIM = $362C
PET = $354F

10 byte table which contains the default values for the memory allocated for the text file, label file, and relocatable buffer. On cold start entry, Macro ASSM/TED assumes these values.

APPLE/SYM/KIM = $244B
PET = $23EB

A subroutine which outputs an error message. The error code to be output should be loaded in register X. This location may be userful especially if you want to add additional program modules to ASSM/TED.

SPECIAL NOTES FOR APPLE II VERSION

1) Macro ASSM/TED for Apple does not contain code to control the tape motors since there is no control hardware for this. If you add hardware and want to activate this software, you can insert code in ASSM/TED to control the motors as follows:

>ON, >OFF Command Motor Controls:

Turn off deck 0 (332F - 3336)	For each, example:
Turn off deck 1 (3338 - 333F)	LDA PORTADDRS
Turn on deck 0 (3341 - 3348)	ORA # or AND #
Turn on deck 1 (334A - 3351)	STA PORTADDRS

Toggle Motor Controls (!T):

Toggle motor control 0 (3C7E - 3C85)	For each, example:
Toggle motor control 1 (3C73 - 3C7A)	LDA PORTADDRS
	EOR #
	STA PORTADDRS

2) The APPLE II cassette recording format does not contain a record start sequence. Thus before loading a module, you should wait until you hear the leader tone before terminating the command with a return.

3) Macro ASSM/TED and Integer Basic use the ">" character as their prompter. If you want to change ASSM/TED's prompter to some other character, simply enter the ascii code for the desired character at $25AC.

MACRO ASSM/TED - Unconfigured Version and Versions for KIM

The information on these sheets describe how to load the data on the supplied cassette and configure this software for your system.

The supplied cassette is in a specially recorded format which can be read by any 6502 based system with a system clock of 1 mhz. The procedure for loading the data on this cassette is as follows:

Procedure for loading the data on the cassette tape

1) Read the description of the Fast Cassette Interface. This is the software which reads the cassette data.

2) Manually enter the object code contained on the Fast Cassette Interface listing, and construct the connection to your tape deck.

3) Configure this listing per your system. The required changes pertain to cassette input/output ports and are underlined in the listing.

4) Enter the following data:

Address	Data
0123	01
0124	00
0125	20
0126	FD
0127	3F

5) Insert cassette tape and position a few seconds before start of data, execute the Fast Cassette Interface software at 4141, and then press the Play switch on the tape deck.

6) After approximately one minute, the data should have been loaded. If the contents of the accumulator = 00 then you have a good load. If = EE then an error was detected, and you should try again. If it appears the program "hung up", then recheck connections and the modifications to the Fast Cassette Interface software. Also try via the invertor in the circuit.

The following is not required for the following versions: KIM
Configure ASSM/TED for your system requirements

1) Configure via table A by entering the address of appropriate routines or patches.

2) If you prefer, link ASSM/TED to the Fast Cassette Interface software as follows:

Address	Data	
3FA3	4C A5 40	links in load
3FD3	4C 00 40	links in record

EASTERN HOUSE SOFTWARE
CARL W. MOSER
3239 LINDA DRIVE
WINSTON SALEM, N.C. 27106

Table A - ASSM/TED Input/Output Requirements

- CRT output — Output to CRT the ASCII character in reg. A — 3C27/ 4C lo hi

- Keyboard input — Input from keyboard with ASCII character in reg. A (Do not echo to CRT) — 3C89/ 20 lo hi

- Break key — Tests if break key is depressed Carry bit in PSR = 1 if key down. (Two locations) — 3BCC/ 20 lo hi; 3BD2/ 20 lo hi

- Tape Motor Controls (ON and OFF Commands)

turn off deck 0	(332F - 3336)	For each, example:
turn off deck 1	(3338 - 333F)	LDA PORTADDRS
turn on deck 0	(3341 - 3348)	ORA # or AND #
turn on deck 1	(334A - 3351)	STA PORTADDRS
		RTS

- Toggle Motor Controls (^T)

toggle motor control 0	(3C7E - 3C85)	For each, example:
toggle motor control 1	(3C73 - 3C7A)	LDA PORTADDRS
		EOR #
		STA PORTADDRS

- Go to Basic (^B) — 3C53/ 4C lo hi

- Tape load/record requirements

LOAD (3FA3 - 3FC7) — Load from START ADDRESS to END ADDRESS but store in memory only if LOAD/NO is not = 0. If = 0 then load from tape or disc but do not store in memory. (LOAD/NO function is skip over non selected files.)

Note: The LOAD software should set the z-bit in the TSR if good load, else clear it if error in loading.

RECORD (3FD3 - 3FFD) — Record from START ADDRESS to END ADDRESS to tape or disc. (LOAD/NO has no meaning.)

Variables used by LOAD and RECORD:

Variable	Location
LOAD/NO	0123
START ADDRESS lo	0124
" " hi	0125
END ADDRESS lo	0126
" "	0127

Table of default file boundaries is located at: 362C/ 10 bytes

KIM ERRATA

EASTERN HOUSE SOFTWARE
3239 Linda Drive
Winston-Salem, N.C. 27106

The KIM version uses zero page from $AC to $EA, and the bottom of the stack from $0100 up for variable storage. Do not let other programs alter these locations.

COLD/WARM START ENTRY POINTS: The entry points for the KIM version are cold start = $2000, and warm start = $2003.

PRINTER DRIVER: Three NOP instructions are provided which the user may insert a JSR to a printer driver routine which outputs the ASCII character in register A. The address for this is at 38D4.

>FORMAT COMMAND: The user may specify maximum label size via the >FORMAT Command, where n=number of characters per label (default = 10, maximum = 31). Example: >FORMAT SET 8. This feature is useful for those who are very verbose (>FO SET 31), and for those who have low character density displays (>FO SET 4). You may for example enter >FO SET 6 and get 6 character labels which also automatically displays the mnemonic after the 7-th column. No matter which setting you use, a label up to 79 characters may be entered if no opcode or pseudo op is on the same line. An example follows:

```
MEMORY.TEST.FOR.6502
            LDA #$F7
            JMP MEMORY.TEST.FOR.6502
```

>PRINT /: Outputs the last line in the text file.

.MG PSEUDO OP: .MG declares the entire contents of the text file as Macro Global. When assembling from tape or disk, all following files will be loaded into the text file area following the file with the .MG. Thus, even though there can be many modules loaded and assembled, the macro solbal file is "locked" into the text file area providing its macro definitions for use by all subsequent files.

>AUTO Command: Not clearly mentioned in the manual was the method of exiting auto line numbering. To temporarily terminate prompting via line numbers, type // and return. The prompter (>) will appear. The next time you enter a line number, auto line numbering will again commence. To halt auto line numbering, type >AUTO 0, or just >AUTO.

MAKE THE FOLLOWING CORRECTIONS IN THE MANUAL:

PAGE #	WHERE	CHANGE	TO
25	at DISC1	$F0, $F1	$B0, $B1
25	at DISC2	$F2, $F3	$B2, $B3
25	at DISCI.VEC	$F6, $F7	$B6, $B7
26	at DISCO.VEC	$F4, $F5	$B4, $B5

ADD THE FOLLOWING AT THE END OF PART 6 ON PAGE 26:

DISCC.VEC $AE, $AF

Address vector to your DOS (or patch to DOS) which accepts user input of disk commands beginning at $0135,Y. The user provided patch should accept the command string until R(Y) = 50 hex which indicates end of the string.

ADD THE FOLLOWING COMMAND ON PAGE 8:

>DC command string

Pass disk commands to DOS or patch to DOS thru vector DISCC.VEC. Possible examples are:
>DC DELETE STARTREK , >DC DIRECTORY , etc.

//

]