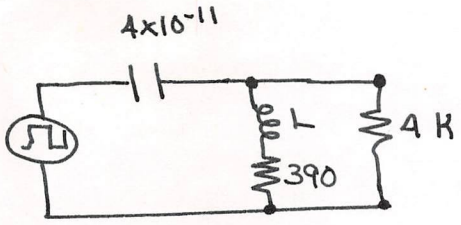
```
      ARHTST(IST)=ARHAT                                                  LUPC
      ADHTST(IST)=ADHAT                                                  LUPC
   48 XR=AR+BR=ARHAT  $  XD=AD+BD=ADHAT
      XRST(IST)=XR  $  XDST(IST)=XD                                      LUPC
      IF (RADL(I),LT,ABSF(XR),OR,RADW(I),LT,ABSF(XD))  GO TO 50
      IF (PKH,LT,RANF(-1)) GO TO 50
      ANPK(I)=ANPK(I)+1,/NMC
      GO TO 100
   50 I=I+1
      IF (I,GT,N) GO TO 100
      AR=RHOR*AR+QR*ZAPGAUSS(1)
      AD=RHOD*AD+QD*ZAPGAUSS(1)
      GO TO 40
  100 CONTINUE
      PRINT 184,RHOR,RHOD,FPSHT,RADL(1),RADW(1),RANGE,XR,XD
  184 FORMAT(4(2X,E11,4)/4(2X,E11,4))
      DO 110 I=2,N
  110 ANPK(I)=ANPK(I)+ANPK(I-1)
      DO 120 I=1,N
      PRINT 115, I,ANPK(I)
  115 FORMAT (5X,I4,2X,F5,3)
  120 CONTINUE
  222 CONTINUE
      GO TO 8
      END
```



1888
15
1873

$$0 = \frac{E_o}{4\times10^3} + \frac{E_o}{3.9\times10^2 + sL} + (E_o - 1)\, s\, 4\times10^{-11}$$

$$E_o = \frac{s\,4\times10^{-11}}{(s4\times10^{-11}) + \frac{1}{4\times10^3} + \frac{1}{3.9\times10^2+sL}}$$

$E^2C = EI\,SEC$

$$= \frac{(s4\times10^{-11})(4\times10^3)(3.9\times10^2+sL)}{(s4\times10^{-11})(4\times10^3)(3.9\times10^2+sL) + (3.9\times10^2+sL) + (4\times10^3)}$$

$$= \frac{6.24\times10^{-5}\,s + 1.6\times10^{-7}\,s^2L}{(6.24\times10^{-5}+L)s + 1.6\times10^{-7}\,s^2L + 4.39\times10^3}$$

16
3.9
234
39
62.4

Zeros at $s = 0$; $s = -\frac{390}{L}$

$\times10^{5-11}$
$\times10^{-6}$
6.24×10^{-5}

$C = \frac{AMP\,SEC}{V}$
$= \frac{SEC}{OHM}$

$L =$ OHMSEC.

Poles at

$$\frac{-(6.24\times10^{-5}+L) \pm \sqrt{(6.24\times10^{-5}+L)^2 - (4\times4.39\times10^3\times1.6\times10^{-7}L)}}{2\,(1.6\times10^{-7}L)}$$

$L^2 + (1.248\times10^{-4} \quad -2.8096\times10^{-2})L + 3.89376\times10^{-9}$

125
7971

$$L^2 - 2.7971\times10^{-2}L + 3.89\times10^{-9} = 0$$

$$2.797\times10^{-2} \pm \sqrt{\quad}$$

```
      PROGRAM SHPTOAIR
      DIMENSION Q(20)
      DIMENSION ANPK(5000), RADL(5000),RADW(5000)
      DIMENSION ARHTST(200), ADHTST(200),XRST(200),XDST(200)                LUPC
      EQUIVALENCE (TAUR,Q(1)),(TAUD,Q(2)),(SIGAR,Q(3)),(SIGAD,Q(4)),
     1(SIGBR,Q(5)),(SIGBD,Q(6)),(R,Q(7)),(SPM,Q(8)),(V,Q(9)),
     2(B,Q(12)),(PKH,Q(13)),(BLEN,Q(14)),(WID,Q(15)),(RHOR,Q(17)),
     3(RHOD,Q(18))
    8 CALL DATA (Q,IND)
      IF (IND) RETURN
      N=Q(10)  $  NMC=Q(11)  $  IP = Q(16)
      DO 10 I=1,N
   10 ANPK(I)=0.
      QR=SIGAR $ QD=SIGAD
      IF(Q(19)) GO TO 435
      RHOR=0.  $  RHOD=0.
  435 CALL RANFSET(B)
      T=60./SPM
      IF(SIGBR.EQ.0.)  SIGBR = 0.000001*SIGAR
      IF (SIGBD.EQ.0.) SIGBD = 0.000001*SIGAD
      IF (Q(19)) GO TO 18
      IF (TAUR.EQ.0.) GO TO 20
      RHOR=EXPF(-T/TAUR)
   18 QR=QR*SQRTF(1.-RHOR**2)
   20 IF (Q(19)) GO TO 24
      IF (TAUD.EQ.0.) GO TO 25
      RHOD=EXPF(-T/TAUD)
   24 QD=QD*SQRTF(1.-RHOR**2)
   25 ALEN=500.*BLEN  $  AWID=500.*WID
      FPSHT=V*T*1.6877
      DO 30 I=1,N
      RANGE=R-FPSHT*(I-1)
      RADL(I)=ALEN/RANGE
   30 RADW(I)=AWID/RANGE
C     MONTE CARLO DO LOOP
      DO 100 J=1,NMC
      I=1
      ROLDHAT=0.  $  DOLDHAT=0.                                             LUPC
      ARHAT=0.  $  ADHAT=0.                                                 LUPC
      VARR=SIGAR**2  $  VARD=SIGAD**2                                       LUPC
      DO 26 JINX=1,IP                                                       LUPC
   26 ADHTST(JINX) = ARHTST(JINX) = 0.                                      LUPC
      AR=SIGAR*ZAPGAUSS(1)  $  AD=SIGAD*ZAPGAUSS(1)
   40 BR=SIGBR*ZAPGAUSS(1)  $  BD=SIGBD*ZAPGAUSS(1)
      IST=I-(I/IP)*IP+1                                                     LUPC
      IF (I-IP) 48,48,41                                                    LUPC
   41 VARR=(QR**2+RHOR**2*VARR)*SIGBR**2/(QR**2+                            LUPC
     1SIGBR**2+RHOR**2*VARR)                                                LUPC
      VARD=(QD**2+RHOD**2*VARD)*SIGBD**2/(QD**2+                            LUPC
     1SIGBD**2+RHOD**2*VARD)                                                LUPC
      ROLDHAT=(ARHTST(IST)+XRST(IST))*VARR/SIGBR**2+                        LUPC
     1RHOR*(1.-VARR/SIGBR**2)*ROLDHAT                                       LUPC
      DOLDHAT=(ADHTST(IST)+XDST(IST))*VARD/SIGBD**2+                        LUPC
     1RHOD*(1.-VARD/SIGBD**2)*DOLDHAT                                       LUPC
      ARHAT=RHOR**IP*ROLDHAT                                                LUPC
      ADHAT=RHOD**IP*DOLDHAT                                                LUPC
```

```
      ARHTST(IST)=ARHAT                                                  LUP
      ADHTST(IST)=ADHAT                                                  LUP
   48 XR=AR*BR=ARHAT  $  XD=AD*BD=ADHAT
      XRST(IST)=XR  $  XDST(IST)=XD                                      LUP
      IF (RADL(I),LT,ABSF(XR),OR,RADW(I),LT,ABSF(XD))  GO TO 50
      IF (PKH,LT,RANF(-1)) GO TO 50
      ANPK(I)=ANPK(I)+1,/NMC
      GO TO 100
   50 I=I+1
      IF (I,GT,N) GO TO 100
      AR=RHOR*AR+QR*ZAPGAUSS(1)
      AD=RHOD*AD+QD*ZAPGAUSS(1)
      GO TO 40
  100 CONTINUE
      PRINT 184,RHOR,RHOD,FPSHT,RADL(1),RADW(1),RANGE,XR,XD
  184 FORMAT(4(2X,E11,4)/4(2X,E11,4))
      DO 110 I=2,N
  110 ANPK(I)=ANPK(I)+ANPK(I-1)
      DO 120 I=1,N
      PRINT 115, I,ANPK(I)
  115 FORMAT (5X,I4,2X,F5,3)
  120 CONTINUE
  222 CONTINUE
      GO TO 8
      END
```

```
      FUNCTION ZAPGAUSS (Y)
      DIMENSION TN(42)
      DATA (TN=-3,2905,-1,96,-1,6449,-1,4395,-1,2816,-1,1503,-1,0364,
     1-,9346,-,8416,-,7554,-,6745,-,5978,-,5244,-,4538,-,3853,-,3186,
     2-,2533,-,1891,-,1257,-,0627,0.,,0627,,1257,,1891,,2533,,3186,
     3,3853,,4538,,5244,,5978,,6745,,7554,,8416,,9346,1,0364,1,1503,
     41,2816,1,4395,1,6449,1,96,3,2905,3,2905)
      DATA (T=40,),(D=0,5)
      R=RANF(-1)
   10 M=T*R   $  N=M+1
      D=T*R-M
      ZAPGAUSS=TN(N)+D*(TN(N+1)-TN(N))
      RETURN
      END
```

```
      PROGRAM SHPTOAIR
      DIMENSION Q(20)
      DIMENSION ANPK(5000), RADL(5000),RADW(5000)
      DIMENSION ARHTST(200), ADHTST(200),XRST(200),XDST(200)                LUP
      EQUIVALENCE (TAUR,Q(1)),(TAUD,Q(2)),(SIGAR,Q(3)),(SIGAD,Q(4)),
     1(SIGBR,Q(5)),(SIGBD,Q(6)),(R,Q(7)),(SPM,Q(8)),(V,Q(9)),
     2(B,Q(12)),(PKH,Q(13)),(BLEN,Q(14)),(WID,Q(15)),(RHOR,Q(17)),
     3(RHOD,Q(18))
    8 CALL DATA (Q,IND)
      IF (IND) RETURN
      N=Q(10)  $  NMC=Q(11)  $  IP = Q(16)
      DO 10 I=1,N
   10 ANPK(I)=0,
      QR=SIGAR $ QD=SIGAD
      IF(Q(19)) GO TO 435
      RHOR=0,  $  RHOD=0,
  435 CALL RANFSET(8)
      T=60,/SPM
      IF(SIGBR,EQ,0,)  SIGBR = 0,000001*SIGAR
      IF (SIGBD,EQ,0,) SIGBD = 0,000001*SIGAD
      IF (Q(19)) GO TO 18
      IF (TAUR,EQ,0,) GO TO 20
      RHOR=EXPF(-T/TAUR)
   18 QR=QR*SQRTF(1,-RHOR**2)
   20 IF (Q(19)) GO TO 24
      IF (TAUD,EQ,0,) GO TO 25
      RHOD=EXPF(-T/TAUD)
   24 QD=QD*SQRTF(1,-RHOR**2)
   25 ALEN=500,*BLEN  $  AWID=500,*WID
      FPSHT=V*T*1,6877
      DO 30 I=1,N
      RANGE=R+FPSHT*(I-1)
      RADL(I)=ALEN/RANGE
   30 RADW(I)=AWID/RANGE
C     MONTE CARLO DO LOOP
      DO 100 J=1,NMC
      I=1
      ROLDHAT=0,  $  DOLDHAT=0,                                             LUP
      ARHAT=0,  $  ADHAT=0,
      VARR=SIGAR**2  $  VARD=SIGAD**2                                       LUP
      DO 26 JINX=1,IP                                                       LUP
   26 ADHTST(JINX) = ARHTST(JINX) = 0,                                      LUP
      AR=SIGAR*ZAPGAUSS(1)  $  AD=SIGAD*ZAPGAUSS(1)
   40 BR=SIGBR*ZAPGAUSS(1)  $  BD=SIGBD*ZAPGAUSS(1)
      IST=I-(I/IP)*IP+1                                                     LUP
      IF (I-IP) 48,48,41                                                    LUP
   41 VARR=(QR**2*RHOR**2*VARR)*SIGBR**2/(QR**2*                            LUP
     1SIGBR**2*RHOR**2*VARR)                                                LUP
      VARD=(QD**2*RHOD**2*VARD)*SIGBD**2/(QD**2*                            LUP
     1SIGBD**2*RHOD**2*VARD)                                                LUP
      ROLDHAT=(ARHTST(IST)*XRST(IST))*VARR/SIGBR**2*                        LUP
     1RHOR*(1,-VARR/SIGBR**2)*ROLDHAT                                       LUP
      DOLDHAT=(ADHTST(IST)*XDST(IST))*VARD/SIGBD**2*                        LUP
     1RHOD*(1,-VARD/SIGBD**2)*DOLDHAT                                       LUP
      ARHAT=RHOR**IP*ROLDHAT                                                LUP
      ADHAT=RHOD**IP*DOLDHAT                                                LUP
```

```
                                        IDENT     SHPTOAIR

PROGRAM LENGTH                 37442
ENTRY POINTS    SHPTOAIR       36740
EXTERNAL SYMBOLS
                Q8QENTRY
                Q1Q10100
                Q2Q07110
                Q1Q05100
                THEND,
                Q8QDICT,
                DATA
                RANFSET
                ZAPGAUSS
                RANF
                SQRTF
                EXPF
                STH,
                QNSINGL,
   00202 SYMBOLS
```

```
5,4DS  ZAPGAUSS                                        04/01/71        ED

                                          IDENT     ZAPGAUSS
     PROGRAM LENGTH                00133
     ENTRY POINTS    ZAPGAUSS      00057
     EXTERNAL SYMBOLS
                     Q1Q10100
                     Q1Q03100
                     Q8QDICT,
                     RANF
        00026 SYMBOLS
LOAD
RUN,50,500
```

```
PROGRAM NAMES
1 40335  SHPTOAIR  37442      1 40202  ZAPGAUSS  00133      1 37746  ALLOC,    002
1 35433  TSH,      00045      1 35400  Q8QIFIOC  00033      1 35340  Q8QPAUSE  000
1 35101  Q8QERROR  00205      1 35067  Q2QLOADA  00012      1 34704  POWRF     001
1 33302  IOP,      01360      1 33244  STH,      00036      1 33172  EXPF      000
1 33072  RANF      00041      1 32140  DATA      00732      1 31515  IOS,      004
1 31205  XTOI      00130      1 31021  Q1QSTORE  00164      1 30743  Q8QENTRY  000

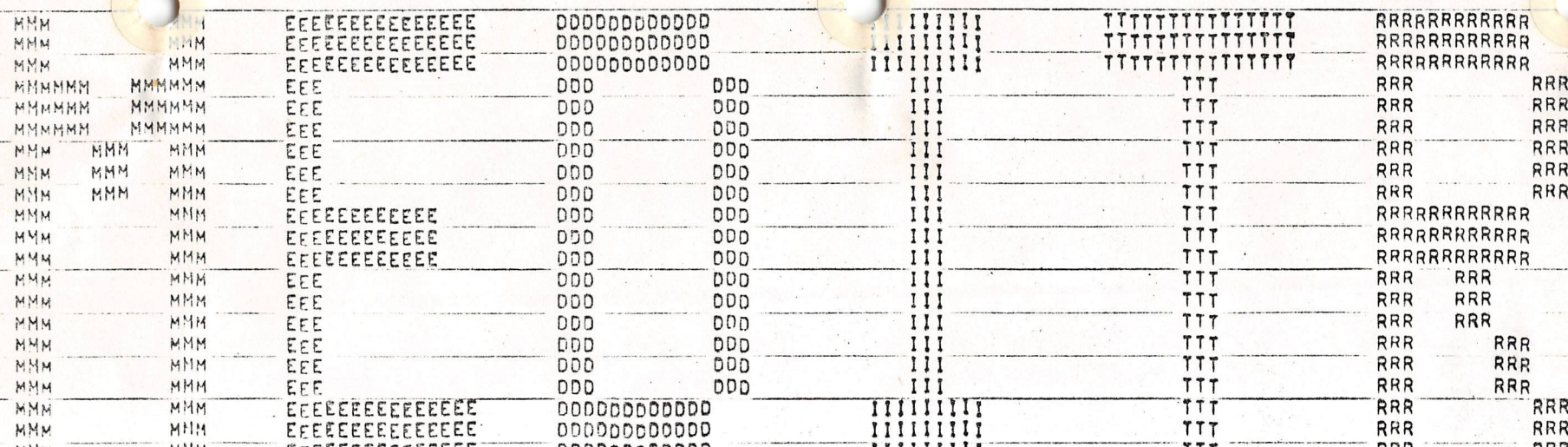
PROGRAM EXTENS,
NONE

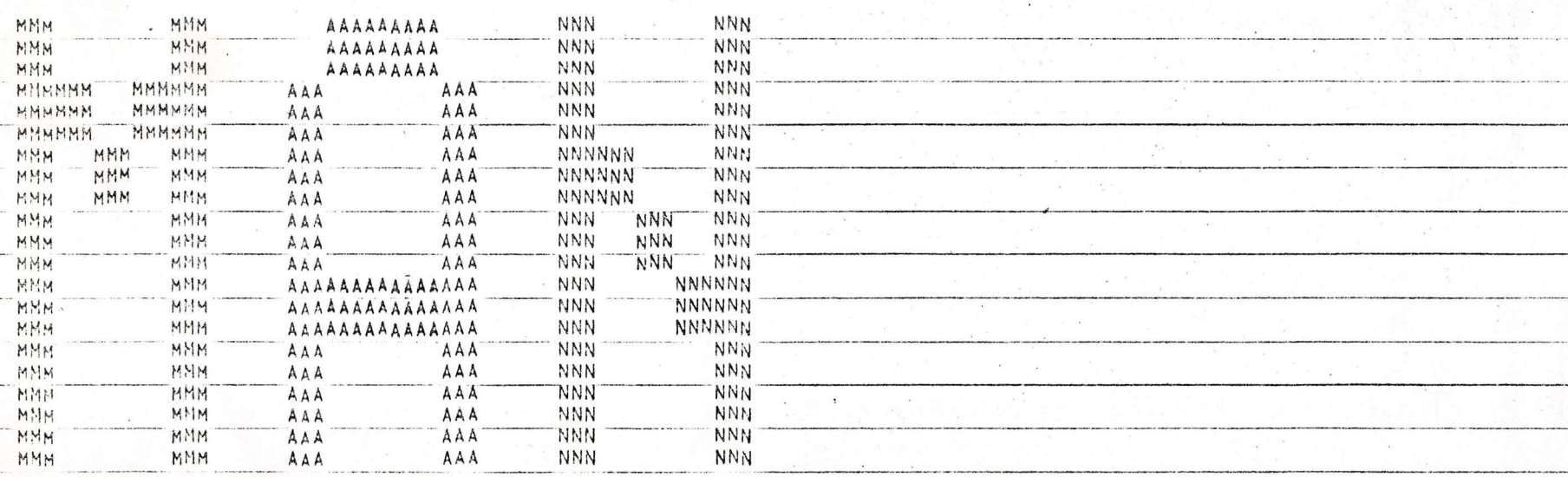
LABELED COMMON
NONE

NUMBERED COMMON
NONE

ENTRY POINTS
0 77777  SENTRY              1 77275  SHPTOAIR            1 30747  Q8QENTRY
1 31210  Q2Q07110            1 31370  Q1Q005100           1 32034  THEND,
1 32311  DATA                1 33072  RANFSET             1 40261  ZAPGAUSS
1 33142  SQRTF               1 33201  EXPF                1 33250  STH,
1 31360  Q1Q03100            1 33305  IOP,                1 31570  Q8QHIST,
1 34720  POWRF               1 35067  Q2QLOADA            1 35102  Q8QERROR
1 35340  Q8QSTOPS            1 34707  Q2Q07111            1 35403  Q8QIFIOC
1 34703  Q8QLDCON            1 34700  Q8QLODA             1 35524  IOH,
1 32040  IOR,                1 35500  BCDBUF,             1 40104  ALLOC,
1 40060  BUSY,               1 40067  IRETURN,            1 30744  EXIT
1 37621  .TSERR,             1 32031  QNDOUBL,            1 40150  ALLOCIN,
1 37600  .REPCNT,            1 35407  Q8QIFEOF            1 35346  Q8QPAUSE
1 35211  Q8QERSET            1 34714  Q8QPOWRF            1 34327  ETAB,
1 33136  Q8QSQRTF            1 33100  Q8QRANF             1 33117  RANFGET
1 31335  Q1Q000100           1 31335  Q1Q001100           1 31354  Q1Q002100
1 31406  Q3Q000040           1 31340  Q3Q000140           1 31415  Q3Q001040
1 31454  Q3Q002040           1 31424  Q3Q002140           1 31463  Q3Q003040
1 31472  Q3Q004040           1 31440  Q3Q004140           1 31501  Q3Q005040
1 31215  Q8QXTOI             1 31221  XTOI                1 31107  Q1Q10010
1 31055  Q1Q10030            1 31102  Q1Q10120            1 31061  Q1Q10130
1 31064  Q1Q10210            1 31064  Q1Q10230            1 31130  Q1Q10300
1 31102  Q1Q10320            1 31123  Q1Q10400            1 31116  Q1Q10410
1 31075  Q1Q10430            1 31021  Q3Q100400           1 31021  Q3Q10140
1 31021  Q3Q10340            1 31021  Q3Q104400

EXECUTION STARTED AT   1856 =09
```





LPTSPL VERSION 6(344) RUNNING ON LPT000
START USER 6502 GROUP [7066,322] JOB FOCDOC SEQ. 3247 DATE 03-SEP-77 20:13:30 MONITOR CSM DECSYSTEM-10 603V1 *START*
REQUEST CREATED: 03-SEP-77 19:45:19
FILE: DSKB1:MEDITR.MAN[7066,322,MARSH] CREATED: 29-AUG-77 23:35:00 <177> PRINTED: 03-SEP-77 20:18:53
QUEUE SWITCHES: /PRINT:ARROW /FILE:ASCII /COPIES:1 /SPACING:1 /LIMIT:155 /FORMS:MINE

C - CLEAR

A - APPEND FROM END OF TEXT

L - LIST BUFFER ON PRINTER

I - Initialize printer

FOR EDITOR AS SAVED
ON TAPE 30 MARTY
"EDITOR/ASSEMBLER
W/ CASSETTE I/O"

CASSETTE WRITE TERMINATES ITSELF
WITH AN 'ESC' CODE.
- READ OF CASSETTE TERMINATES ON
FINDING AN 'ESC' CODE.

FOR EDITOR AS SAVED
ON TAPE 30 MAR 79
"EDITOR/ASSEMBLER
W/ CASSETTE I/O"

EDITOR COMMANDS:

COMMAND MODE:
A - APPEND TO END OF TEXT BUFFER
C - CLEAR BUFFER & BEGIN EDIT
E - DON'T CLEAR; BEGIN EDIT
I - INITIALIZE LINE PRINTER
L - LIST BUFFER ON LINE PRINTER
M - RETURN TO KIM MONITOR
P - PRINT BUFFER ON CRT CONSOLE
R - READ A FILE FROM CASSETTE
S - APPEND A FILE FROM CASSETTE
W - WRITE THE BUFFER TO CASSETTE

EDIT MODE:
↑A - GO TO BUFFER START
↑D - GO TO BUFFER END
↑F - FIND FOLLOWING STRING
↑P - PRINT A LINE ON CONSOLE
↑Q - DELETE FROM CURRENT POS. TO
END OF LINE
↑R - REPRINT LINE TO CURR. POS.
↑S - MOVE CURSOR TO BEGINNING OF
LINE/PREV. LINE
LINE FEED - ↑P W/LINE FEED ADDED
RUBOUT - DELETE ONE CHAR BACK
ESCAPE - RETURN TO COMMAND MODE

NOTES:
- BE SURE TO HAVE A CTRL-Z IN
TEXT OR LINE PRINTER WILL GO
FOREVER.
- CAN TERMINATE LINE PRINTER BY
TAKING IT OFF-LINE AND TURNING
OFF ITS POWER.
- CASSETTE WRITE TERMINATES ITSELF
WITH AN 'ESC' CODE.
- READ OF CASSETTE TERMINATES ON
FINDING AN 'ESC' CODE.

THE MINI-EDITOR WAS DESIGNED AS A GENERAL PURPOSE EDITOR FOR SMALL TO MEDIUM SOURCE FILES AND DOCUMENTS. IT WAS DESIGNED FOR COMPACTNESS, SO THAT PROGRAMS COULD GENERATED AND ASSEMBLED WITH A MINIMUM OF MEMORY AND WITHOUT ELABORATE MASS STORAGE DEVICES.

CREATING AND EDITING FILES

THE EDITOR HAS TWO DISTINCT COMMAND MODES: OUTER AND INNER. IN THE INNER MODE, THE EDITOR DIRECTLY MANIPULATES THE TEXT. IN OUTER MODE THE EDITOR DEALS WITH EXTERNAL DEVICES.

OUTER MODE

UPON ENTRY, THE EDITOR WILL BE IN OUTER MODE. IN OUTER MODE THE EDITOR PROMPTS WITH A POUND SIGN (#). THE FOLLOWING COMMANDS AVAILABLE IN OUTER MODE:

R READ — THIS COMMAND CLEARS THE EDITING BUFFER AND READS A FILE FROM THE MASS STORAGE DEVICE. GENERALLY THIS WILL BE THE PAPER TAPE READER ON THE TELETYPE OR A CASSETTE INTERFACE ATTACHED TO THE CONSOLE DEVICE. IF THE MASS STORAGE DEVICE IS SEPERATE FROM THE CONSOLE, ADDRESSES OF DIFFERENT HANDLERS MAY BE PATCHED INTO THE READ AND WRITE ROUTINES. AFTER THE FILE HAS BEEN LOADED, SENDING AN ALTMODE WILL RETURN THE EDITOR TO OUTER MODE.

W WRITE — THIS COMMAND WRITES A FILE TO THE MASS STORAGE DEVICE. IT DOES NOT CLEAR THE BUFFER, AFTER THE THE COMMAND IS TYPED, THE EDITOR WAITS FOR A CARRIAGE RETURN BEFORE WRITING THE FILE. THE EDITOR RETURNS TO OUTER MODE AS SOON AS THE FILE IS WRITTEN.*

P PRINT — THIS COMMAND PRINTS THE FILE IN FULLY EXPANDED FORM WITH TAB SIMULATION AND CONTROL CHARACTER REPRESENTATION.*

M MONITOR — RETURNS THE USER TO THE SYSTEM MONITOR, SUCH AS TIM OR KIM MONITORS.

L CLEAR LOAD FROM KEYBOARD — CLEARS THE EDITING BUFFER AND ENTERS INNER MODE. TO START NEW EDIT

E EDIT — DOES NOT CLEAR THE EDITING BUFFER, AND ENTERS INNER MODE AT THE BEGINING OF THE TEXT.

C APPEND CONTINUE — DOES NOT CLEAR THE BUFFER AND ENTERS INNER MODE AT THE END OF THE TEXT.

L - LIST ON PRINTER
I - INITIALIZE PRINTER

*NOTE: AFTER "W" AND "P" COMMANDS THE EDITOR WAITS FOR THE INPUT OF TWO NON-NULL CHARACTERS BEFORE ENTERING NORMAL OUTER MODE. THIS ALLOWS CLEAN END OF LISTING OUTPUT AND END OF TAPE FORMATTING. THE INPUT ROUTINE ALSO IGNORES ALL NULL CHARACTER INPUT. THIS ALLOWS GENERATION OF NULL LEADER AND TRAILER FOR PAPER TAPES AND ALSO IGNORES LEADER AND TRAILER NULLS WHEN INPUTTING PAPER TAPES.

THE EDITOR SEES THE TEXT AS A SERIES LINES, EACH LINE BOUNDED BY A CARRIAGE RETURN, FOR SIMPLICITY, THE EDITOR DOES NOT UNDERSTAND LINE FEEDS, IT STRIPS THEM OUT OF THE TEXT WHEN IT READS AND ADDS THEM BACK WHEN IT WRITES.

IN INNER MODE, ANY CHARACTER TYPED ON THE KEYBOARD GOES DIRECTLY INTO THE TEXT BUFFER UNLESS ITS A COMMAND CHARCTER, COMMAND CHARACTERS ALLOW YOU MOVE THROUGH MOVE THROUGH THE TEXT, FIND STRINGS OF CHARACTERS, DELETE LINES AND CHARACTERS AND GENERALY EDIT THE TEXT. ALL COMMAND CHARACTERS ARE CONTROL CONTROL CHARACTERS.

THE EDITOR PRINTS MOST CONTROL CHARACTERS AS AN UP ARROW PLUS THE BASE CHARACTER, FOR EXAMPLE, CONTROL-L IS REPRESENTED "↑L". HORIZONAL TABS (CONTROL-I), ARE SIMULATED WITH STOPS AT 9,18,27,36,45, ETC.

THE FOLLOWING COMMANDS ARE AVAILABLE IN INNER MODE:

↑A	CONTROL-A MOVES THE EDITOR POINTER TO THE BEGINING OF THE TEXT. WHILE EDITING, THE BEGINING OF THE TEXT IS ALWAYS INDICATED BY LEFT ANGLE BRACKET "<".
↑D	CONTROL-D MOVES THE POINTER TO THE END OF THE TEXT. THE END OF TEXT IS ALWAYS INDICATED BY A RIGHT ANGLE BRACKET ">".
↑S	CONTROL-S MOVES THE POINTER TO THE BEGINING OF THE CURRENT LINE OR BACK ONE LINE IF ALREADY AT THE BEGINING OF A LINE. NO TEXT IS PRINTED
LINE FEED	MOVES THE POINTER TO THE BEGINING OF THE NEXT LINE AND PRINTS ALL OF THE INTERVENING TEXT.
↑R	CONTROL-R REPRINTS THE CURRENT LINE UP TO THE POINTER, IF THE POINTER IS AT THE BEGINING OF A LINE, A CARRIAGE RETURN-LINE FEED IS PRINTED.
↑Q	CONTROL-Q DELETES ONE LINE FORWARD FROM THE POINTER.
ALTMODE (ESCAPE)	RETURNS THE EDITOR OUTER MODE.
↑F	CONTROL-F ALLOWS THE USERS TO FIND A STRING OF CHARACTERS ANYWHERE WITHIN BUFFER. AFTER CONTROL-F IS TYPED THE USER MAY TYPE A SEARCH STRING OF UP TO TEN CHARACTERS THE STRING IS TERMINATED WITH A CARRIAGE RETURN. AS SOON AS THE CARRIAGE RETURN IS TYPED, THE EDITOR SEARCHES FOWARD FROM THE POINTER UNTIL EITHER A MATCH IS FOUND OR THE END OF TEXT IS REACHED. IF A MATCH IS FOUND, THE POINTER IS LEFT AT THE ENC OF THE LAST MATCHING CHARACTER, AND THE LINE UP TO THE POINTER IS PRINTED. IF NO MATCH IS FOUND THE END OF TEXT SIGN IS PRINTED ">". IF THE MATCHING STRING'S LAST CHARACTER IS THE END OF THE TEXT, THE END OF TEXT SIGN IS INDICATED AND NO TEXT PRINTED.
RUBOUT	DELETES ONE CHARACTER BACK FROM THE POINTER ← echos as \
↑P	CAUSES EDITOR TO PROCEED TO THE END OF THE CURRENT LINE INDICATED BY THE CURRENT POINTER. IT HAS THE SAME EFFECT AS THE LINE FEED BUT DOES NOT INTRODUCE AN EXTRA LINE FEED IN THE OUTPUT. THIS SAVES PAPER AND SHOWS ACTUAL FORMAT OF THE TEXT.

THE GENERAL EDITING PROCESS IS FAIRLY STRAIGHT FORWARD.
IF YOU ARE CREATING A NEW FILE, ENTER THE INNER MODE USING THE "L" COMMAND. THEN JUST TYPE THE TEXT INTO THE BUFFER.
IF YOU MAKE AN ERROR AS YOU TYPE YOU MAY DELETE ONE OR MORE CHARACTERS USING THE RUBOUT COMMAND. AFTER YOU HAVE DELETED THE ERRORS, REPRINT THE LINE USING THE "R" AND CONTINUE TYPING. IF YOU WISH TO REVIEW THE LAST FEW LINES ENTERED, TYPE SEVERAL "↑S" COMMANDS AND THEN STEP FORWARD USING THE LINE FEED COMMAND. A RIGHT ANGLE BRACKET WILL APPEAR WHEN YOU HAVE REACHED THE END OF THE TEXT AND YOU CAN CONTINUE ENTERING NEW TEXT AT THE END OF THE FILE.
WHEN YOU HAVE FINISHED ENTERING TEXT OR IF YOU WISH TO MAKE A TEMPORARY BACK UP TAPE, ENTER OUTER MODE USING AN ALTMODE, AND AND WRITE A TAPE.

IF YOU WISH TO MAKE CORRECTIONS IN AN EXISING TEXT, LOAD THE TEXT USING THE "R" COMMAND AND ENTER THE INNER MODE USING THE "E" COMMAND. IF YOU HAVE A GOOD IDEA WHERE YOU ARE GOING TO MAKE CHANGES, USE THE "↑F" TO FIND THE TEXT THAT NEEDS CHANGE. MATERIAL MAY BE INSERTED OR DELETED FROM THE EDITING POINTER AT ANY PLACE WITHIN THE TEXT.
USING THE LINE DELETE COMMAND CAN BE TRICKY. IF YOU AREN'T SURE WHERE THE POINTER IS LOCATED, YOU MAY DELETE TEXT THAT WANT TO SAVE. REMEMBER THAT THE LINE FEED COMMAND PUTS YOU AT THE BEGINING OF THE LINE FOLLOWING THE LINE JUST PRINTED. IF YOU WISH TO DELETE THE LINE JUST PRINTED, TYPE "↑S" TO BACK ONE LINE, THEN "↑Q" TO DELETE THE LINE.

NOTE: THE EDITOR ACTUALLY LOOKS FOR "ESCAPE" ($1B) INSTEAD OF "ALTMODE" ($7D) SINCE THIS SEEMS TO BE MORE STANDARD. IF YOU HAVE AN ALTMODE KEY CHANGE LOCATIONS BASE+$00CB, BASE+$0119, AND BASE+$02C1 FROM$1B TO $7D.

OPERATOR INPUT OF CONTROL CHARACTERS INDICATED BY (↑A), (↑P), (LF), etc.

```
.LH    (MINI-EDITOR LOADED VIA HIGH SPEED OPTICAL READER)

.R  3AC0 00 00 00 00 FF
.:  2600  (SET EXECUTION ADDRESS. KIM WOULD BE 3600).
.G ØE
<       NØTE: A CØLD START WITH AN "E" LEADS TØ SELF-DESTRUCTIØN
              IF THE EDITØR BUFFERS ARE INITIALLY ZERØ.
              NØW I WILL DØ A PRØPER CØLD STAPT. (ESCAPE)
#L
> (↑I)  ANØTHER PRØBLEM IS USING "L" IN THE ØUTER MØDE WHEN YØU
   "    ALREADY HAVE GENERATED A FAIR AMØUNT ØF TEXT. THE PØINTERS
   "    ARE RE-INITIALIZED BY BY "L", BUT YØU MAY GET A CLUE AS
   "    TØ WHERE YØUR END ØF TEXT IS BY ESCAPING TØ THE MØNITØR
(RUBOUT) A\(↑I)  AND EXAMINGING THE ZERØ PAGE PØINTERS. LETS EDIT SØME RIGHT
              NØW.
↑A<↑F    (SET TO START OF TEXT, THEN FIND:)
MING(R)
              (RUBOUT) (↑P)
              AND EXAMINGING THE ZERØ PAGE PØINTERS. LETS EDIT SØME RIGHT
↑S(↑P)         AND EXAMINING THE ZERØ PAGE PØINTERS. LETS EDIT SØME RIGHT
              NØW.

>#P
(ESCAPE)      ANØTHER PRØBLEM IS USING "L" IN THE ØUTER MØDE WHEN YØU
              ALREADY HAVE GENERATED A FAIR AMØUNT ØF TEXT. THE PØINTERS
              ARE RE-INITIALIZED BY BY "L", BUT YØU MAY GET A CLUE AS
              TØ WHERE YØUR END ØF TEXT IS BY ESCAPING TØ THE MØNITØR
              AND EXAMINING THE ZERØ PAGE PØINTERS. LETS EDIT SØME RIGHT
              NØW.
(W)(R) #

#M

.M  0000 8B 29 B5 2A B5 2A B5 2A
.R  2600 00 00 00 00 FF                (ILLUSTRATES RECOVERY FROM
.G #L                                   L IN OUTER MODE)
>#M

.M  0000 8E 29 8C 29 B5 2A 8C 29
.G #M                                  <-- INFORMATION REQUIRED FOR
                                           RECOVERY.
.M  0000 8B 29 8C 29 B5 2A 8C 29
.:  0000       B5 2A       B5 2A
.G ØE
<↑F                          (RUB)(RUB)(RUB)
BY
              ARE RE-INITIALIZED BY (LF)
                                    BY "L", BUT YØU MAY GET A CLUE AS
↑S↑S(↑P)       ALREADY HAVE GENERATED A FAIR AMØUNT ØF TEXT. THE PØINTERS
      "       ARE RE-INITIALIZED BY "L", BUT YØU MAY GET A CLUE AS
      "       TØ WHERE YØUR END ØF TEXT IS BY ESCAPING TØ THE MØNITØR
      "       AND EXAMINING THE ZERØ PAGE PØINTERS. LETS EDIT SØME RIGHT
      "       NØW. (↑P)
>
```

(KIM USERS WILL HAVE TO RESET EXECUTION ADDRESS. AFTER MONITOR ENTRY, TIM'S EXECUTION POINTER IS NOT MODIFIED BY THE MINI-EDITOR.)

```
>                SAMPLE ØF TEXT CREATIØN AND EDITING

LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TXT
LINE NØMBRE QUATØR ØF TEXT                    (R) = "RETURN"

↑A<(↑P)

                SAMPLE ØF TEXT CREATIØN AND EDITING

(↑P)LINE NUMBER 1 ØF TEXT
(R)
(↑P)LINE NUMBER 2 ØF TEXT          (Introduction of extra blank lines)
(R)
(ESCAPE) #P

                SAMPLE ØF TEXT CREATIØN AND EDITING

LINE NUMBER 1 ØF TEXT

LINE NUMEER 2 ØF TEXT

LINE NUMBER 3 ØF TEXT
LINE NUMEER 4 ØF TXT
LINE NØMBRE QUATØR ØF TEXT

LE#T ←(UNRECOGNIZED BY OUTER MODE - DOES "CRLF #" PROMPT).
      2 THROWAWAY CHARACTERS AFTER P(RINT)
#E
<#C  (ENTER AT END OF TEXT  C(ONTINUE)]
>          LET'S ILLUSTRATE REMØVAL ØF EXTRA LINE FEEDS
↑A<↑F
LINE (↑P)
LINE NUMBER 1 ØF TEXT

L\\#P                    (RUBOUT) (THIS RUBOUT DELETED THE LF)
(ERROR) (RUBOUT)
                SAMPLE ØF TEXT CREATIØN AND EDITING

LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT

LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TXT
LINE NØMBRE QUATØR ØF TEXT

          LET'S ILLUSTRATE REMØVAL ØF EXTRA LINE FEEDS
#E
<↑F
2
LINE NUMBER 2(LF)
                ØF TEXT
    (↑P)
(RUBOUT)\#P  (ESCAPE)
REMOVED
EXTRA BLANK
LINE
                SAMPLE ØF TEXT CREATIØN AND EDITING

LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TXT
LINE NØMBRE QUATØR ØF TEXT

          LET'S ILLUSTRATE REMØVAL ØF EXTRA LINE FEEDS
```

```
↑S↑P   LET'S ILLUSTRATE REMØVAL ØF EXTRA LINE FEEDS
>      NØW LET'S DELETE A LINE
↑A<↑F
LET'S(R)                    SET TO START OF LINE
       LET'S↑S↑Q#P          DELETED FIRST LINE FORWARD.

                SAMPLE ØF TEXT CREATIØN AND EDITING

LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TXT
LINE NØMBRE QUATØR ØF TEXT

       NØW LET'S DELETE A LINE
  #E
<↑F
NØW(R)
       NØW(LF)
           LET'S DELETE A LINE
>      LET'S REMØVE A TAB (CNTRL I)
↑A<↑F                  (RUBOUT) DELETED L  (RUBOUT) DELETED (↑I)
(↑I) L
(SEARCH STRING)        L\\L(LF)
                           ET'S REMØVE A TAB (CNTRL I)
>#P

                SAMPLE ØF TEXT CREATIØN AND EDITING

LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TXT
LINE NØMBRE QUATØR ØF TEXT

       NØW LET'S DELETE A LINE
LET'S REMØVE A TAB (CNTRL I)
  #E
<      DØ SØME EDITING (R)
(↑P)
                SAMPLE ØF TEXT CREATIØN AND EDITING
                                (SET BACK TO START OF LINE, DELETE IT, FIND:)
↑S↑Q↑F
4 ØF T(R)                (↑P)
LINE NUMBER 4 ØF TEXT
↑F
NØ(R)          (RUBOUT)
LINE NØ\U↑F
RE
LINE NUMBRE\\ER↑F
ØR
LINE NUMBER QUATØR\\RØ ØF TEXT
EP\\#P        (ESCAPE)
ERRORS  DØ SØME EDITING
  CORRECTIONS
LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TEXT
LINE NUMBER QUATRØ ØF TEXT

       NØW LET'S DELETE A LINE
LET'S REMØVE A TAB (CNTRL I)
```

```
↑E
<↑F
QUATRØ
LINE NUMBER QUATRØ\\\\\\\5 ØF TEXT
↑SLINE NUMBER 5 ØF TEXT
↑F
DELETE
        NØW LET'S DELETE\\\\\\CHANGE↑R   (RETYPE FROM START OF LINE
                                          TO POINTER).
        NØW LET'S CHANGE A LINE
↑S↑F
A(R)
        NØW LET'S CHA↑F
A(R)                          (RUBOUT)
        NØW LET'S CHANGE A\SØME(LF)
                             LINE
↑S↑F
LINE(R)
        NØW LET'S CHANGE SØME LINES  (ESCAPE)
#W                          (TURN PUNCH ON) (R)
DØ SØME EDITING


LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT                (↑I's ARE NON-PRINTING CHARACTERS)
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TEXT
LINE NUMBER 5 ØF TEXT

NØW LET'S CHANGE SØME LINES

LET'S REMØVE A TAB (CNTRL I)   (ESCAPE) (HEREIS) (R)  (END OF TAPE FORMAT)
#R                                                    (READ TAPE BACK IN)
DØ SØME EDITING


LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TEXT
LINE NUMBER 5 ØF TEXT

NØW LET'S CHANGE SØME LINES

LET'S REMØVE A TAB (CNTRL I)
#

#P

        DØ SØME EDITING


LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TEXT
LINE NUMBER 5 ØF TEXT

        NØW LET'S CHANGE SØME LINES

LET'S REMØVE A TAB (CNTRL I)
```

```
#E
<
\       DØ SØME EDITING                                          V
↑S<↑Q@@@@@@@@@↑S<@@@@@@@@
```

(ERROR: ↑Q TURNED AUTOMATIC TAPE READER ON.
RECOVERY: (↑S), (↑P) TO SEE WHAT HAPPENED
(↑A), (↑Q) SET TO START, DELETE ERROR LINE.

```
↑A<↑Q
\LINE NUMBER 1 ØF TEXT
 LINE NUMBER 2 ØF TEXT
 LINE NUMBER 3 ØF TEXT
 LINE NUMBER 4 ØF TEXT
 LINE NUMBER 5 ØF TEXT
```

(↑P) 's TO LOOK AT TEXT.

DELETE BLANK LINE.

NØW LET'S CHANGE SØME LINES — CLEAN UP TEXT

```
↑S↑Q
\LET'S REMØVE A TAB (CNTRL I)
>↑S↑Q>#P
LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TEXT
LINE NUMBER 5 ØF TEXT

#W
LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TEXT
LINE NUMBER 5 ØF TEXT
```

(WRITE TAPE)

(END FORMAT: NO (ESCAPE) - WANT TO MERGE TAPES.)

```
#E
<LINE NUMBER 6 ØF TEXT
 LINE NUMBER 7 ØF TEXT
```

ERROR: SHOULD HAVE DONE L.

```
#W
LINE NUMBER 6 ØF TEXT
LINE NUMBER 7 ØF TEXT

LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TEXT
LINE NUMBER 5 ØF TEXT
```

← DIDN'T MEAN TO KEEP THIS.

```
#L
>       NØW MERGE TWØ TAPES (THE FIRST MUST NØT END WITH "ESCAPE")
#R
LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TEXT
LINE NUMBER 5 ØF TEXT

LINE NUMBER 6 ØF TEXT
LINE NUMBER 7 ØF TEXT

LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TEXT
LINE NUMBER 5 ØF TEXT
```

↑THIS LINE DESTROYED BY #R

← FIRST TAPE

← SECOND TAPE

```
#P                                   ENTERED MANUALLY TO CLOSE FILE.
LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TEXT
LINE NUMBER 5 ØF TEXT


LINE NUMBER 6 ØF TEXT
LINE NUMBER 7 ØF TEXT

LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TEXT
LINE NUMBER 5 ØF TEXT

   #E
<(↑P)     FINAL EDITING EXAMPLE
↑F
5 (↵)
LINE NUMBER 5(LF)
              ØF TEXT
(↑P)          (RUBOUT) (↑P) (RUBOUT)   (REMOVED 2 BLANK LINES)
\LINE NUMBER 6 ØF TEXT
LINE NUMBER 7 ØF TEXT
                                  (RUBOUT) (REMOVED BLANK LINE)
\LINE NUMBER 1 ØF TEXT            (↑P) (LOOK AT NEXT LINE)
↑S↑Q↑Q↑Q↑Q↑Q↑Q>#P
        FINAL EDITING EXAMPLE     DELETE TO END OF FILE
                                  (ESC) PRINT FINAL VERSION

LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TEXT
LINE NUMBER 5 ØF TEXT
LINE NUMBER 6 ØF TEXT
LINE NUMBER 7 ØF TEXT
#C ——— (CONTINUE)
>
        THIS MINI-EDITØR IS EXTREMELY PØTENT. IT WLL \\\ILL TAKE
        SØME PRACTIVE\\CE TØ MASTER THE ART ØF KNØWING WHERE
        THE CURRENT PØINTER IS LØCATED.  ØNCE YØU HAVED\ ATTAINED
        THE SKILL REQUIRED YØU WILL APPRECIATE THE EFFICIENCY
        AND CØMPACTNESS ØF THIS MINI-EDITØR.

        ØNE ADDTIØNAL PØINT IS IMPØRTANT.  NØ WARNING IS PRØVIDED
        IF YØU TRY TØ STØRE TEXT IN NØN-EXISTENT MEME\ØRY. YØU
        SHØULD ØCCASIØNALLY ESCAPE TØ YØUR MØNITØR AND LØØK AT
        $0001 AND $0003.↑S↑F
1
        $0001\2↑R
              (↑P)
        $0002 AND $0003.> ANØTHER WAY TØ CHECK IS TØ DØ A ↑S AND
        ↑P ØR "LF", TØ SEE IF YØUR TEXT IS ACTUALLY BEING STØRED.
        #M

.M  0002 5A 2C E9 2B 5A 2C 20 2A          END OF TEXT
.G#C
>       WHILE A WARNING IS NICE, THE PØWER ØF INSERTING ØR
        DELETING TEXT AT ANY PØINT, WØULD REQUIRE A GREAT
        DEAL ØF ADDITIØNAL EDITØR CØDE, TØ CHECK FØR END ØF
MEMØRY↑S MEMØRY>
```

↑P

FINAL EDITING EXAMPLE

LINE NUMBER 1 ØF TEXT
LINE NUMBER 2 ØF TEXT
LINE NUMBER 3 ØF TEXT
LINE NUMBER 4 ØF TEXT
LINE NUMBER 5 ØF TEXT
LINE NUMBER 6 ØF TEXT
LINE NUMBER 7 ØF TEXT

THIS MINI-EDITØR IS EXTREMELY PØTENT. IT WILL TAKE SØME PRACTICE TØ MASTER THE ART ØF KNØWING WHERE THE CURRENT PØINTER IS LØCATED. ØNCE YØU HAVE ATTAINED THE SKILL REQUIRED YØU WILL APPRECIATE THE EFFICIENCY AND CØMPACTNESS ØF THIS MINI-EDITØR.

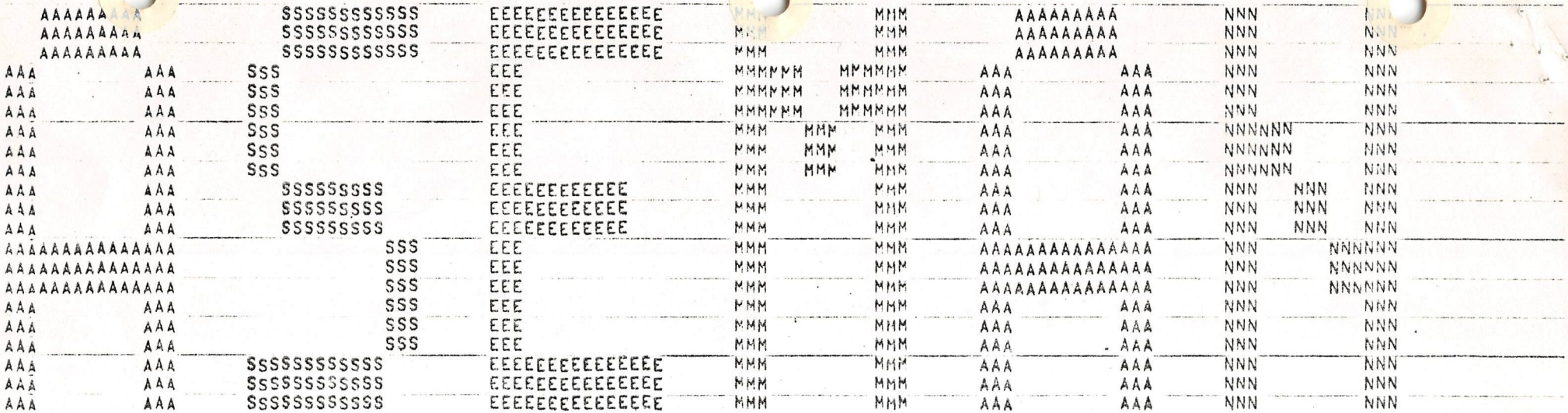
ØNE ADDTIØNAL PØINT IS IMPØRTANT. NØ WARNING IS PRØVIDED IF YØU TRY TØ STØRE TEXT IN NØN-EXISTENT MEMØRY. YØU SHØULD ØCCASIØNALLY ESCAPE TØ YØUR MØNITØR AND LØØK AT $0002 AND $0003. ANØTHER WAY TØ CHECK IS TØ DØ A ↑S AND ↑P ØR "LF", TØ SEE IF YØUR TEXT IS ACTUALLY BEING STØRED.

WHILE A WARNING IS NICE, THE PØWER ØF INSERTING ØR DELETING TEXT AT ANY PØINT, WØULD REQUIRE A GREAT DEAL ØF ADDITIØNAL EDITØR CØDE, TØ CHECK FØR END ØF MEMØRY

MM/M
(2 THROWAWAY CHARACTERS AFTER P.)
.R 2600 00 00 00 00 FF
.

(TO SAVE A MEMORY IMAGE OF YOUR TEXT,
LOOK AT END OF TEXT POINTER AT $0002, $0003)
THEN SAVE $0000 - $0012
AND TBEG - END OF TEXT.

TIM = $2988
KIM = $3995


AAAAAAAAA
AAAAAAAAA
AAAAAAAAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAAAAAAAAAAAAAA
AAAAAAAAAAAAAAA
AAAAAAAAAAAAAAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
SSSSSSSSSSS
SSSSSSSSSSS
SSSSSSSSSSS
SSS
SSS
SSS
SSS
SSS
SSS
SSS
SSS
SSS
SSS
SSS
SSS
SSS
SSS
SSS
EEEEEEEEEEEEEEE
EEEEEEEEEEEEEEE
EEEEEEEEEEEEEEE
EEE
EEE
EEE
EEE
EEE
EEE
EEE
EEE
EEE
EEE
EEE
EEE
EEE
EEE
EEE
MMM MMM
MMM MMM
MMM MMM
MMMMMMM MMMMMMM
MMMMMMM MMMMMMM
MMMMMMM MMMMMMM
MMM MMM MMM
MMM MMM MMM
MMM MMM MMM
MMM MMM
MMM MMM
MMM MMM
MMM MMM
MMM MMM
MMM MMM
MMM MMM
MMM MMM
MMM MMM
MMM MMM
MMM MMM
MMM MMM
MMM MMM
MMM MMM
AAAAAAAAA
AAAAAAAAA
AAAAAAAAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
NNN NNN
NNN NNN
NNN NNN
NNN NNN
NNN NNN
NNN NNN
NNNNNN NNN
NNNNNN NNN
NNNNNN NNN
NNN NNN
NNN NNN
NNN NNN
NNN NNN
NNN NNN
N

**********ASM65 USERS MANUAL**********

ASM65 IS A TWO PASS RESIDENT ASSEMBLER FOR 650X MACHINE CODE. THE ASSEMBLER TAKES SPECIALLY FORMATTED TEXT CALLED "SOURCE" AND AND CONVERTS IT INTO MACHINE LOADABLE BINARY TEXT. IT ALSO CREATES A COMBINED TEXT CONTAINING HEX ADDRESSES, OPCODES, SOURCE AND SYMBOL TABLE. THIS TEXT IS CALLED A "LISTING".

CREATING A SOURCE FILE

A SOURCE FILE CONSISTS OF SERIES OF FORMATTED COMMANDS TO THE ASSEMBLER. SOURCE FILES ARE GENERATED USING AN EDITOR PROGRAM AND ARE STORED ON CASSETTE TAPE, PAPER TAPE, FLOPPY DISK OR ANY SIMILIAR MASS STORAGE DEVICE. HERE IS A SAMPLE SOURCE FILE FOR A SHORT PROGRAM:

```
        ;SHORT PROGRAM
        .LOC    $1000
START:  LDA#    'A
        JSR     WRT     ;OUT PUT A CHARACTER
        JMP     START   ;START OVER
        .END
```

THE SOURCE TEXT IS DIVIDED INTO FOUR FIELDS. EACH FIELD IS SEPARATED BY A TAB CHARACTER (CONTROL I). STARTING FROM LEFT TO RIGHT THE FIELDS ARE: LABEL FIELD, OPCODE FIELD, OPERAND FIELD, AND COMMENT FIELD. EACH LINE OF THE SOURCE TEXT MAY HAVE ITEMS IN ONE OR MORE FIELDS. FOR EXAMPLE, THE THIRD LINE OF THE SAMPLE PROGRAM HAS A LABEL, AN OPCODE AND AN OPERAND, BUT NOTHING IN THE COMMENT FIELD.

LABELS

LABELS ARE USED TO DESIGNATE A CERTAIN LINE OF CODE SYMBOLICALLY. FOR EXAMPLE, THE JUMP INSTRUCTION IN OUR SAMPLE MUST JUMP SOMEWHERE, BUT INSTEAD OF JUMPING TO AN EXACT ADDRESS THAT MAY CHANGE IF WE ALTER THE PROGRAM, THE JUMP INSTRUCTION JUMPS TO A SYMBOLIC LOCATION CALLED "START". THE EXACT ADDRESS OF "START" MAYBE UNKNOWN UNTIL FINAL ASSEMBLY.

LABELS CAN CONTAIN AS MANY AS 10 CHARACTERS INCLUDING NUMBERS; EXCEPT THAT THE FIRST CHARACTER CANNOT BE A NUMBER, THE LABEL MUST BE FOLLOWED BY A COLON AND A TAB (CONTROL I). LABEL NAMES ARE USUALLY CHOOSEN SO THAT THEY DESCRIBE THE FUNCTION OF THE LINE DESIGNATED. FOR EXAMPLE, THE LABEL "START" INDICATES THAT THIS LINE IS THE START OF THE PROGRAM.

Source code of ASM65 indicates that the label can legally be followed by either a : or a tab - - -

LABEL SHOULD START WITH ALPHA, BUT MAY CONTAIN ANYTHING EXCEPT =, +, -, *, Tab, CR, space, ;

~~Apparently may contain spaces, which are ignored. (GO TO same as GOTO same as G OTO etc)~~

OPCODES

THE SECOND FIELD CONTAINS EITHER AN OPCODE OR A PSEUDO-OPCODE. AN OPCODE IS MNEMONIC REPRESENTATION OF A MACHINE LANGUAGE OPERATION. FOR EXAMPLE, LDA REPRESENTS THE MACHINE LANGUAGE OPERATION OF LOADING THE ACCUMULATOR, EACH OPCODE AND ITS MNEMONIC IS DESCRIBED IN DETAIL IN THE MOS TECHNOLOGY PROGRAM MANUAL. SINCE THERE ARE THIRTEEN ADDRESSING MODES AVAILABLE IN THE 6502 PROCESSOR, THE MNEMONIC MUST ALSO INDICATE ADDRESSING MODES. ADDRESSING MODES ARE REPRESENTED AS FOLLOWS:

LDA#	IMMEDIATE MODE
LDA	ABSOLUTE MODE
LDA	ZERO PAGE MODE
ROLA	ACCUMULATOR MODE
LDAX@	INDEXED INDIRECT X MODE
LDA@Y	INDIRECT INDEXED Y MODE
LDAX	ZERO PAGE INDEXED X MODE
LDAX	ABSOLUTE INDEXED X MODE
LDAY	ABSOLUTE INDEXED Y MODE
BCC	RELATIVE MODE
JMP@	INDIRECT MODE
LDXY	ZERO PAGE INDEXED Y MODE

NOTICE THAT THE FORMAT IS SLIGHTLY DIFFERENT FROM THE MOS TECHNOLOGY ASSEMBLER. IN THIS FORMAT THE OPCODE CARRIES THE ADDRESSING MODE INSTEAD OF THE OPERAND. THIS MAKES SOURCE TEXT MUCH CLEANER AND SIMPLIER. ALSO NOTICE THAT THE ASSEMBLER AUTOMATICALLY CHOOSES ZERO PAGE ADDRESSING WHEN THE OPCODE IS ON ZERO PAGE. NO SPECIAL SYMBOL IS REQUIRE FOR ZERO PAGE ADDRESSING.

PSEUDO-OPS

PSUEDO-OPS ARE DIRECT COMMANDS TO THE ASSEMBLER. THEY ARE PLACED IN THE OPCODE FIELD. ALL PSUEDO-OPS BEGIN WITH A DOT ".". THE FOLLOWING PSEUDO-OPS ARE AVAILABLE IN ASM65.

.BYTE

INSTRUCTS THE ASSEMBLER TO SET ASIDE ONE BYTE AT THIS LOCATION. THE VALUE OF THE BYTE CAN BE SET BY AN ARGUMENT IN THE OPERAND FIELD. IF NO ARGUMENT IS USED THE BYTE IS SET TO A DEFAULT VALUE OF ZERO.

```
.BYTE   $45
.BYTE   %012
.BYTE   START
```

.HBYTE

INSTRUCTS THE ASSEMBLER TO SET THE VALUE TO THE MOST SIGNIFICANT BYTE OF A TWO BYTE ARGUMENT. THIS MAY ALSO BE ACCOMPLISHED BY PLACING A ">" CHARACTER IN FRONT OF THE OPERAND.

```
.HBYTE  START
.HBYTE  $4567
.BYTE   >START
```

.WORD

INSTRUCTS THE ASSEMBLER TO SET ASIDE TWO BYTES AT THIS LOCATION. THE VALUE MAYBE SET BYAN ARGUMENT IN THE OPERAND FIELD.

```
.WORD   $7066
.WORD   START
```

.PAGE

INSTRUCTS THE ASSEMBLER TO PRINT A FORM FEED AND MOVE THE LISTING TO THE TOP OF THE NEXT PAGE. NO ARGUMENT IS REQUIRED.

.ASCII

INSTRUCTS THE ASSEMBLER TO CONVERT THE CHARACTERS IN THE ARGUMENT TO ASCII. THE CHARACTERS IN THE ARGUMENT ARE ENCLOSED IN SINGLE QUOTES.

```
        .ASCII  'SAMPLE LINE '
```

.LOC

INSTRUCTS THE ASSEMBLER TO RESET THE ADDRESS COUNTER TO A NEW LOCATION SPECIFIED BY THE ARGUMENT. IF NO LOCATION IS AT THE BEGINING OF ASSEMBLY, THE ASSEMBLER STARTS THE ADDRESS COUNTER AT PAGE ZERO.

```
        .LOC    $1000
        .LOC    BEGIN
```

.DEF

IS USED TO SET THE VALUE OF A LABEL OR TO SET ONE LABEL EQUAL TO ANOTHER.

```
        .DEF    WRT=$72C6
        .DEF    LAND=WRT
```

.END

INSTRUCTS THE ASSEMBLER TO FINISH THE ASSEMBLY PROCESS. THIS INSTRUCTION MUST BE PLACED AT THE END OF THE SOURCE OR THE ASSEMBLER WILL NOT RETURN TO THE CONSOLE.

```
        .PAGE
;Patch for ASM65 (V1C)  13-JUN-77
;T. Bruhns   - - -   23 JAN 82
;This patch is intended to fix a problem with the way ASM65
;deals with "<" and ">".  The above version of ASM65 assumes
;that the < or > operates on only the atom immediately following.
;I find it much more useful to have the < or > (generally only
;matters with the >) operate on the result of summing all the
;atoms in a chain separated by +'s and -'s.  For example:
;
;Before patch:
;
;                       .LOC    $200
;                       .DEF    Atom=$3800
;  0200 A9 34           LDA#    >Atom-4 ;Note -  ()Atom)-4
;  0202 A9 FC           LDA#    <Atom-4 ;Note -  (<Atom)-4
;                       .END
;
;After patch:
;                       .LOC    $200
;                       .DEF    Atom=$3800
;  0200 A9 37           LDA#    >Atom-4 ;Note -  >(Atom-4)
;  0202 A9 FC           LDA#    <Atom-4 ;Note -  <(Atom-4)
;                       .END
;
;At first glance, there is no difference for the < operator.
;However, before the patch, a string of atoms separated by
; +'s and -'s, preceeded by a <, could return a non-zero high
;byte; there are cases, such as in .DEF's, where this could
;be bad.
;
;Implementation:  Key in this patch  **AND** change the branch
;offset at $260C (in the KIM version noted above) from $68
;to $5D.  This patch takes somewhat less space than what it
;replaces; you don't need to change anything but what is here
;and loc. $260C.
```

```
*** ASM65 ***  (V1C)  13-JUN-77

                        .PAGE
                        .LOC  $266A       ;Location of AILBYT in original
                        .DEF  GEXPR1=$2291        ;Subr in oris. ASM65 - see below
                        .DEF  NUMBH=$2C   ;Atom value save area
                        .DEF  NUMBL=$2D   ;  (High and low bytes)
                        .DEF  ZERNUM=$291C        ;Subr to zero atom value save area
                    ;
266A 48         AIHLBT: PHA               ;Save the operator
266B 20 91 22           JSR   GEXPR1      ;Go skip ( or ), get atom, check for following
                                          ; + or -, and add terms till not + or -
266E 68                 PLA               ;Get back original ( or )
266F C9 3E              CMP#  '>          ;Assume low byte if not high byte
2671 D0 04              BNE   AILBYT
2673 A5 2C              LDA   NUMBH       ;Explicit high byte: fetch it
2675 85 2D              STA   NUMBL       ;Put it in low byte
2677 A9 00      AILBYT: LDA#  0           ;Low is OK; now zero high
2679 85 2C              STA   NUMBH
267B A5 2D              LDA   NUMBL       ;Antibussing: exit here same as original,
267D 60                 RTS               ;  with NUMBL in accumulator.
                        .END

  267E .
  2291 GEXPR1
  002C NUMBH
  002D NUMBL
  291C ZERNUM
  266A AIHLBT
  2677 AILBYT

? *** ERRORS WERE DETECTED! ***
```

OPERANDS

OPERANDS CAN CONSIST OF LABELS, ADDRESSES, NUMBERS OR EQUATIONS. NUMERICAL VALUES CAN BE REPRESENTED IN HEX, OCTAL OR DECIMAL FORM. SEVERAL SPECIAL CHARACTERS ARE USED WITHIN THE OPERAND FIELD.

"." (DOT)

REPRESENTS THE CURRENT VALUE OF THE ADDRESS COUNTER. IT CAN BE USED TO ADVANCE THE ADDRESS COUNTER.

```
.LOC    .+8
```

IT ALSO CAN BE USED TO SET A LABEL EQUAL TO THE ADDRESS COUNTER.

```
.DEF    FRON=.
```

"%" (PERCENT)

INDICATES THAT THE FOLLOWING NUMERICAL VALUE IS AN OCTAL REPRESENTATION.

```
.DEF    GOL=%021
```

"$" (DOLLAR SIGN)

INDICATES THAT THE FOLLOWING NUMERICAL VALUE IS A HEXADECIMAL REPRESENTATION.

```
.LOC    $4500
```

CODE INDICATES THAT < & > OPERATE ON ONLY THE NEXT ATOM.

"'" (SINGLE QUOTES)

INDICATES THAT THE FOLLOWING CHARACTER IS TO BE COVERTED TO AN ASCII VALUE.

```
LDA#    'B'
```

THE ASSEMBLER ALSO ALLOW ADDITION AND SUBTRACTION TO BE EVALUATED WITHIN THE OPERAND FIELD.

```
.DEF    JAR=WRT-%123
```

← APPARENTLY ONLY CONSTANTS ARE ACCEPTED IN SECOND ARG. ...

"<" ">" (LEFT AND RIGHT ANGLE BRACKETS)

LEFT ANGLE BRACKET TAKES LOW BYTE OF A VALUE AND RIGHT ANGLE BRACKET TAKE THE HIGH BYTE.

```
LDA#    <WRT
LDA#    >WRT
```

COMMENTS

COMMENTS ALLOW THE USER TO LEAVE NOTES THROUGHOUT THE SOURCE TEXT. ALL COMMENTS MUST BE PROCEEDED BY A SEMICOLON. COMMENTS COMMENTS MAY APPEAR IN ANY FIELD AND ARE IGNORED BY THE ASSEMBLER.

COMMENTS ARE ONE OF THE MOST IMPORTANT PARTS OF THE SOURCE TEXT. MANY ROUTINES MAKE ABSOLUTELY NO SENSE WITHOUT COMMENTS EACH STEP OF THE WAY.

ASSEMBLING A PROGRAM

ONCE YOU HAVE CREATED A SOURCE FILE, THE TEXT CAN BE FED TO THE ASSEMBLER. WHEN THE ASSEMBLER IS STARTED IT WILL ASK THE USER TO INDICATE THE WHICH PASS WILL BE EXECUTED:

```
WHICH PASS? (1 OR 2)
```

AT THE BEGINING OF EACH RUN THE USER MAY CHOOSE TO DO PASS ONE OR PASS TWO. DURING PASS ONE, THE ASSEMBLER COUNTS THROUGH ALL OF THE CODE IN THE SOURCE AND RETAINS THE ADDRESS OF EACH SYMBOL. THIS PROCESS IS CALLED BUILDING A SYMBOL TABLE. THE COMPLETE SYMBOL TABLE MUST BE BUILT BEFORE PASS TWO IS STARTED. DURING THE SECOND PASS, THE ASSEMBLER GENERATES MACHINE LOADABLE BINARY AND A LISTING. THE USER MAY DO SEVERAL PASS ONES, INORDER TO ASSEMBLE SEVERAL FILES INTO A SINGLE PROGRAM. GENERALLY, WHEN A NUMBER OF FILES ARE ASSEMBLED INTO A SINGLE PROGRAM, EACH INDIVIDUAL FILE IS TERMINATED WITH A CONTROL-Z. WHEN THE ASSEMBLER SEES THE CONTROL-Z AT THE END OF THE FILE IT WILL ASK FOR A NEW DEVICE NUMBER FOR THE NEXT FILE.

IF THE USER ELECTS TO DO PASS ONE, ASSEMBLER WILL ASK TWO MORE QUESTIONS:

```
FLUSH SYMBOL TABLE?

DEVICE # FOR SOURCE INPUT:
```

THE FIRST QUESTION ENABLES THE USER THE ERASE ENTRIES IN THE SYMBOL TABLE LEFT FROM PREVIOUS ASSEMBLIES. IT ALSO ENABLES THE USER TO SAVE THE SYMBOL TABLE SO THAT SEVERAL FILES CAN BE ASSEMBLED INTO ONE PROGRAM. THE SECOND QUESTION ENABLES THE ASSEMBLER TO READ SOURCE FROM SEVERAL DIFFERENT MASS STORAGE DEVICES. IF YOU ARE USING THE MINI-EDITOR, DEVICE NUMBER 3 WILL READ SOURCE DIRECTLY FROM THE TEXT EDITOR BUFFER.

DURING PASS ONE THE ASSEMBLER CHECKS FOR ERRORS. IF ERRORS ARE ENCOUNTERED, YOU MAY WISH TO RETURN TO THE EDITOR TO MAKE CHANGES OR GO THROUGH PASS TWO TO PRINT A LISTING SO THAT THE ERROR CAN BE ISOLATED. IF YOU MAKE ANY CHANGES TO THE SOURCE FILE, PASS ONE MUST BE REPEATED AND THE OLD SYMBOL TABLE MUST BE FLUSHED.

WHEN PASS ONE HAS BEEN COMPLETED, THE ASSEMBLER WILL AGAIN ASK WHICH PASS IS TO BE EXECUTED. IF THE USER CHOOSES PASS TWO, THE ASSEMBLER WILL ASK SEVERAL MORE QUESTIONS:

```
DEVICE # FOR BINARY OUTPUT:

DEVICE # FOR LISTING OUTPUT:

WIDTH (127 MAX) OF LISTING DEVICE:

DEVICE # FOR SOURCE INPUT:
```

SINCE THE ASSEMBLER GENERATES BOTH BINARY AND LISTING DURING PASS TWO, IT MAY OUTPUT BOTH TO DIFFERENT DEVICES SIMULTANEOUSLY. FOR EXAMPLE, THE LISTING CAN BE PRINTED ON A TELETYPE, WHILE THE BINARY IS BEING SAVED ON A CASSETTE TAPE. IF THE USER HAS ONLY ONE I/O DEVICE THE LISTING CAN BE SENT TO THE DEVICE DURING ONE RUN OF PASS TWO AND THE BINARY DURING ANOTHER. THE UNWANTED OUTPUT IS SIMPLY SENT TO THE "DUMMY" DEVICE, DEVICE NUMBER 4. FOR EXAMPLE, IF YOUR I/O DEVICE IS A TELETYPE, THE LISTING CAN BE PRINTED D RING ONE RUN OF PASS TWO AND THE BINARY PUNCHED ON TAPE DURING A SECOND RUN OF PASS TWO.

THE ASSEMBLER ALLOWS THE USER TO SET THE WIDTH OF THE LISTNG DEVICE TO ANY VALUE BETWEEN Ø-127. IF A LINE EXCEEDS THE PRESET VALUE, THE REMAINDER OF THE LINE IS DISCARDERD.

THE BINARY GENERATED BY THE ASSEMBLER IS DIRECTLY MACHINE LOADABLE IN THE TIM/KIM HEX FORMAT. IF YOUR SYSTEM IS A TIM, KIM OR JOLT, THE BINARY CAN BE LOADED BY EXECUTING THE PROPER KIM/TIM READ COMMANDS AND FEEDING A COPY OF THE BINARY TO THE SYSTEM. OTHER SYSTEMS CAN USE "LOADER-65" TO LOAD TIM/KIM FORMATTED BINARY INTO MEMORY.

USING THE ASSEMBLER WITH MINI-EDITOR

MINI-EDITOR WAS DESIGNED TO ALLOW THE USER TO ASSEMBLE PROGRAMS DIRECTLY IN MEMORY. THIS IS ACCOMPLISHED BY HAVING THE ASSEMBLER READ SOURCE DIRECTLY FROM THE EDITORS BUFFER. THE HANDLER THAT READS FROM THE BUFFER IS INPUT DEVICE NUMBER 3. ONE OF THE TRADE OFFS INHERENT IN THIS APPROACH IS THAT SOURCE CODE TAKES UP MEMORY SPACE. IT TAKES ABOUT 4K OF UNCOMMENTED SOURCE CODE TO MAKE 1K OF PROGRAM. ONE WAY AROUND THIS IS TO WRITE PROGRAMS IN SMALL MODULES AND ASSEMBLE THEM INTO A SINGLE LARGE PROGRAM.

THE CURRENT EDITOR/ASSEMBLER COMBINATIONS WAS ASSEMBLED SO THAT 8K SYSTEMS WOULD HAVE SEVEN PAGES OF SYMBOL TABLE AND SEVEN PAGES OF EDITING BUFFER. MEMORY BEYOND 8K BECOMES A PART OF THE EDITING BUFFER. THE PROBLEM HERE IS THAT IT IS DIFFICULT TO EXPAND BOTH THE SYMBOL TABLE AND THE EDITING BUFFER AS THE THE USER EXPANDS MEMORY. SEVEN PAGES OF SYMBOL TABLE WILL STORE APPROXIMATELY 300 THREE CHARACTER SYMBOLS. WHICH IS SUFFICIENT SPACE FOR PROGRAMS ONE TO THREE KILOBYTES IN LENGTH.

WHEN ASSEMBLING PROGRAMS DIRECTLY FROM MEMORY, BESURE THAT THAT EACH FILE IN THE EDITING BUFFER IS TERMINATED EITHER WITH A CONTROL-Z OR ,END PSEUDO OP. OTHERWISE, THE ASSEMBLER WILL CONTINUE TO READ SOURCE FROM THE EDITING BUFFER AND WILL NOT COME BACK TO CONSOLE.

ADDING ADDITIONAL DEVICES

ASM65 IS CONFIGURED SO THAT THE USER CAN EASILY ADD NEW DEVICE DRIVERS FOR A VARIETY OF PERIPHERAL DEVICES. THIS IS ACCOMPLISHED THROUGH AN I/O DISPATCH TABLE. TO ADD ADDITIONAL DEVICE DRIVERS, SIMPLY INSERT THE ADDRESS OF THE HANDLER INTO THE DISPATCH TABLE. IN THE CURRENT VERSION OF THE ASSEMBLER, FIVE DEVICE SLOTS ARE USED. DEVICES ONE AND TWO ARE DESCRIBED AS CASSETTE HANDLERS IN THE LISTING. SINCE NO HANDLERS ARE PROVIDED FOR CASSETTE, THE USER MAY EITHER ADD HIS OWN OR USE THE SLOTS FOR DIFFERENT DEVICES.
IF YOU WISH TO USE MORE THAN THE FIVE DEVICE SLOTS, TWO ADDITIONAL BYTES HAVE BEEN PROVIDED. IF YOU USE THESE SLOTS, BE SURE THAT ALL REFERENCES TO THE SYMBOL "DEVMAX" REFLECT THE NEW MAXIUM NUMBER OF DEVICES. RULES FOR WRITING CUSTOM DEVICE HANDLERS ARE PROVIDED IN THE I/O SECTION OF THE LISTING.

DEVICE NUMBER ASSIGNMENT

INPUT	OUTPUT
0-KEYBOARD	0-CRT OR TTY
1-UNASSIGNED	1-UNASSIGNED
2-UNASSIGNED	2-UNASSIGNED
3-EDITOR BUFFER	3-EDITOR HAS NO OUTPUT ROUTINE
4-DUMMY DEVICE	4-DUMMY DEVICE
5-UNASSIGNED	5-LINE PRINTER SIMULATOR (CAUSES AUTO PAGING)

ERROR CODES

L	LABEL ERROR
M	MULTIPLY DEFINED ERROR
U	UNDEFINED ERROR
P	PHASE ERROR
I	INTERNAL ERROR
O	OPCODE ERROR
A	ADDRESSING ERROR
S	SYNTAX ERROR

(NOTE: PHASE ERRORS OCCUR WHERE A SYMBOL IS DEFINED IN DIFFERENT LOCATIONS FROM PASS ONE TO PASS TWO. IT USUALLY OCCURS WHEN SYMBOL IS REFERRENCED FIRST AND THEN LATER DEFINED AS A ZERO PAGE LOCATION.)

THE FOLLOWING IS A TRANSCRIPTION OF A COMPLETE ASSEMBLY USING ASM65. THE FIRST PART IS A LISTING OF THE SOURCE CODE PRINTED FROM THE EDITOR BUFFER. THE REMAINING TEXT WAS GENERATED BY ASM65. THE SYSTEM IS 'TIM' SYSTEM WITH 10K OF MEMORY, TVTII AND KANSAS CITY CASSETTE I/O.

P

```
        ;MEMORY TEST
        ;GALLOPING ZEROS

        .DEF    WRT=$72C6
        .LOC    $0000

LOC:    .WORD                   ;TEST POINTER
LOW:    .WORD                   ;LOWER LIMIT
HIGH:   .WORD                   ;UPPER LIMIT
PTR:    .WORD                   ;TEST POINTER
BTEST:  .BYTE                   ;STORE TEST BYTE
        .LOC    $0010
MAD:    LDA#    $0D             ;PRINT CRLF
        JSR     WRT             ;
        LDA#    $0A             ;
        JSR     WRT             ;
        JSR     RSTLOC          ;SET TEST POINTERS TO LOW LIMIT
        JSR     RSTPTR          ;
        LDX#    $00             ;ZERO X REG

        ;CLEAR MEMORY FIELD

ML1:    LDA#    $00             ;GET A ZERO
        STAX@   PTR             ;PUT IN FIELD
        JSR     INCPTR          ;INCREMENT POINTER
        BNE     ML1             ;LOOP UNTIL DONE

        ;STORE FF IN A CELL

TEST:   LDA#    $FF             ;GET FF
        STAX@   LOC             ;PLACE IN THE FIELD
        JSR     RSTPTR          ;RESET POINTER

        ;TEST THAT ALL CELLS ARE ZERO EXCEPT THE TEST
VLOOP:  LDAX@   PTR     ;GET CELL
        BEQ     NEXTC           ;OK IF ZERO
        LDY     PTR             ;NOT ZERO, IS THE TEST LOCATION
        CPY     LOC             ;CHECK LOW BYTE
        BEQ     OK1             ;BRANCH IF LOW BYTE OK
        BRK                     ;BREAK IF BAD
OK1:    LDY     PTR+1           ;TEST HIGH ORDER
        CPY     LOC+1           ;
        BEQ     OK2             ;BRANCH IF IT IS THE TEST LOC
        BRK                     ;FF NOT IN PROPER LOC
OK2:    CMP#    $FF             ;IS THE DATA IN THE CELL STILL FF?
        BEQ     OK3             ;BRANCH IF DATA GOOD
        BRK                     ;WRONG DATA
OK3:    LDA#    $00             ;CLEAR THE TEST CELL
        STAX@   LOC             ;
```

```
NEXTC:  JSR     INCPTR          ;ADVANCE TO POINT AT NEXT TEST CELL
        BNE     VLOOP           ;TEST FOR CHANGED ZEROS UNTIL UPPER LIMIT
        LDA     LOC             ;HAS OUR GALLOPING FF CROSSED A PAGE?
        BNE     NOSTAR          ;BRANCH IF NOT
        LDA#    '*              ;PRINT STAR ON PAGE BOUNDARY
        JSR     WRT
        LDX#    $00             ;FIX X
NOSTAR: JSR     INCLOC          ;ADVANCE THE FF TEST CELL
        BNE     TEST            ;BRANCH UNTIL ALL LOCATION HAVE HAD FF
        JSR     RSTLOC          ;CLEAR FOR NEXT PASS
        JMP     MAD             ;START AGAIN

        .PAGE
        ;RESET LOC TO LOWER LIMIT

RSTLOC: LDA     LOW             ;GET LOW ORDER VALUE
        STA     LOC             ;PUT IT IN LOC
        LDA     LOW+1           ;GET HIGH ORDER VALUE
        STA     LOC+1           ;
        RTS

        ;RESET PTR TO LOWER LIMIT

RSTPTR: LDA     LOW             ;GET LOW ORDER BYTE
        STA     PTR             ;PLACE IN PTR
        LDA     LOW+1           ;GET HIGH ORDER
        STA     PTR+1           ;
        RTS

        ;INCREMENT PTR & TEST FOR UPPER LIMIT

INCPTR: INC     PTR             ;INCREMENT LOW ORDER
        BNE     INC1            ;BRANCH IF NO WRAP AROUND
        INC     PTR+1           ;BUMP HIGH ORDER
INC1:   LDA     HIGH            ;COMPARE WITH LOW BYTE OF LIMIT
        CMP     PTR             ;
        BNE     IPRET           ;BRANCH IF NOT AT LIMIT
        LDA     HIGH+1          ;TEST HIGH BYTE
        CMP     PTR+1           ;Z BIT WILL BE 1 IF LIMIT IS REACHED
IPRET:  RTS

        ;INCREMENT LOC AND CHECK FOR LIMIT
INCLOC: INC     LOC             ;INCREMENT LOW ORDER BYTE
        BNE     INC2            ;BRANCH IF WRAP AROUND
        INC     LOC+1           ;BUMP HIGH ORDER BYTE
INC2:   LDA     HIGH            ;TEST LOW BYTE AGAINST LIMIT
        CMP     LOC             ;
        BNE     ILRET           ;BRANCH IF NOT AT LIMIT
        LDA     HIGH+1          ;TEST HIGH BYTE
        CMP     LOC+1           ;Z=1 IF AT LIMIT
ILRET:  RTS

        ;BIT TEST
BISTR:  LDY#    $00             ;CLEAR Y
        LDA#    $01             ;GET STARTING BIT
        STA     BTEST           ;SAVE IT
NEWBIT: JSR     RSTLOC          ;ZERO POINTER
        JSR     RSTPTR          ;ZERO POINTER
DEPLOC: LDA     BTEST           ;GET TEST VALUE
```

```
          STA@Y   LOC             ;STORE IN FIELD
          JSR     INCLOC          ;INCREMENT POINTER
          BNE     DEPLOC          ;BRANCH UNTIL FIELD FILLED
BCHECK:   LDA     BTEST           ;GET TEST VALUE
          CMP@Y   PTR             ;COMPARE WITH FIELD
          BEQ     BIOK            ;BRANCH IF OK
          BRK                     ;BREAK IF ERROR
BIOK:     JSR     INCPTR          ;INC POINTER
          BNE     BCHECK          ;BRANCH UNTIL FIELD TESTED
          ASL     BTEST           ;MOVE BIT
          BCC     NEWBIT          ;BRANCH TIL ALL BITS CHECKED
          LDA#    '*              ;INDICATE PASS
          JSR     WRT             ;
          JMP     BISTR           ;AGAIN

          ;TIME TEST

TISTR:    LDY#    $00             ;CLEAR Y
          LDA#    $FF             ;GET TEST VALUE
          STA     BTEST           ;SAVE IT
          JSR     RSTLOC          ;CLEAR POINTER
          JSR     RSTPTR          ;
TILOOP:   LDA     BTEST           ;GET TEST VALUE
          STA@Y   LOC             ;STORE VALUE IN FIELD
          JSR     INCLOC          ;ADVANCE POINTER
          BNE     TILOOP
TIWRAP:   LDA     BTEST           ;GET VALUE
          CMP@Y   PTR             ;HAS IT CHANGED
          BEQ     TIOK            ;BRANCH IF OK
          BRK                     ;BREAK IF CHANGED
TIOK:     JSR     INCPTR          ;ADVANCE POINTER
          BNE     TIWRAP          ;LOOP TILL UPPER LIMIT
          LDA#    '*              ;INDICATE PASS
          JSR     WRT             ;PRINT IT
          JSR     RSTPTR          ;START AGAIN
          JMP     TIWRAP          ;
          .END
#M

.R  2600 31 93 FF 01 FF
.:  1000
.G*** ASM65 ***  (V1C)  13-JUN-77


WHICH PASS (1 OR 2)? 1

FLUSH USER SYMBOL TABLE? Y
DEVICE # FOR SOURCE INPUT:3
READY SOURCE DEVICE
THEN HIT "RETURN" KEY



WHICH PASS (1 OR 2)? 2
DEVICE # FOR BINARY OUTPUT:4
DEVICE # FOR LISTING OUTPUT:0
WIDTH (127 MAX) OF
```

```
LISTING DEVICE:90
DEVICE # FOR SOURCE INPUT:3
READY SOURCE DEVICE
THEN HIT "RETURN" KEY


                        ;MEMORY TEST
                        ;GALLOPING ZEROS

                        .DEF   WRT=$72C6
                        .LOC   $0000

0000 00 00      LOC:    .WORD                   ;TEST POINTER
0002 00 00      LOW:    .WORD                   ;LOWER LIMIT
0004 00 00      HIGH:   .WORD                   ;UPPER LIMIT
0006 00 00      PTR:    .WORD                   ;TEST POINTER
0008 00         BTEST:  .BYTE                   ;STORE TEST BYTE
                        .LOC   $0010
0010 A9 0D      MAD:    LDA#   $0D              ;PRINT CRLF
0012 20 C6 72           JSR    WRT              ;
0015 A9 0A              LDA#   $0A              ;
0017 20 C6 72           JSR    WRT              ;
001A 20 68 00           JSR    RSTLOC           ;SET TEST POINTERS TO LOW LIMIT
001D 20 71 00           JSR    RSTPTR           ;
0020 A2 00              LDX#   $00              ;ZERO X REG

                        ;CLEAR MEMORY FIELD

0022 A9 00      ML1:    LDA#   $00              ;GET A ZERO
0024 81 06              STAX@  PTR              ;PUT IN FIELD
0026 20 7A 00           JSR    INCPTR           ;INCREMENT POINTER
0029 D0 F7              BNE    ML1              ;LOOP UNTIL DONE

                        ;STORE FF IN A CELL

002B A9 FF      TEST:   LDA#   $FF              ;GET FF
002D 81 00              STAX@  LOC              ;PLACE IN THE FIELD
002F 20 71 00           JSR    RSTPTR           ;RESET POINTER

                        ;TEST THAT ALL CELLS ARE ZERO EXCEPT THE TEST
0032 A1 06      VLOOP:  LDAX@  PTR              ;GET CELL
0034 F0 17              BEQ    NEXTC            ;OK IF ZERO
0036 A4 06              LDY    PTR              ;NOT ZERO, IS THE TEST LOCATION
0038 C4 00              CPY    LOC              ;CHECK LOW BYTE
003A F0 01              BEQ    OK1              ;BRANCH IF LOW BYTE OK
003C 00                 BRK                     ;BREAK IF BAD
003D A4 07      OK1:    LDY    PTR+1            ;TEST HIGH ORDER
003F C4 01              CPY    LOC+1            ;
0041 F0 01              BEQ    OK2              ;BRANCH IF IT IS THE TEST LOC
0043 00                 BRK                     ;FF NOT IN PROPER LOC
0044 C9 FF      OK2:    CMP#   $FF              ;IS THE DATA IN THE CELL STILL FF?
0046 F0 01              BEQ    OK3              ;BRANCH IF DATA GOOD
0048 00                 BRK                     ;WRONG DATA
0049 A9 00      OK3:    LDA#   $00              ;CLEAR THE TEST CELL
004B 81 00              STAX@  LOC              ;
004D 20 7A 00   NEXTC:  JSR    INCPTR           ;ADVANCE TO POINT AT NEXT TEST CELL
0050 D0 E0              BNE    VLOOP            ;TEST FOR CHANGED ZEROS UNTIL UPPER LIMIT
0052 A5 00              LDA    LOC              ;HAS OUR GALLOPING FF CROSSED A PAGE?
0054 D0 07              BNE    NOSTAR           ;BRANCH IF NOT
0056 A9 2A              LDA#   '*               ;PRINT STAR ON PAGE BOUNDARY
0058 20 C6 72           JSR    WRT
005B A2 00              LDX#   $00              ;FIX X
005D 20 8B 00   NOSTAR: JSR    INCLOC           ;ADVANCE THE FF TEST CELL
0060 D0 C9              BNE    TEST             ;BRANCH UNTIL ALL LOCATION HAVE HAD FF
0062 20 68 00           JSR    RSTLOC           ;CLEAR FOR NEXT PASS
0065 4C 10 00           JMP    MAD              ;START AGAIN
```

```
*** ASM65 ***  (V1C)  13-JUN-77

                        .PAGE
                        ;RESET LOC TO LOWER LIMIT

0068 A5 02      RSTLOC: LDA   LOW               ;GET LOW ORDER VALUE
006A 85 00              STA   LOC               ;PUT IT IN LOC
006C A5 03              LDA   LOW+1             ;GET HIGH ORDER VALUE
006E 85 01              STA   LOC+1             ;
0070 60                 RTS

                        ;RESET PTR TO LOWER LIMIT

0071 A5 02      RSTPTR: LDA   LOW               ;GET LOW ORDER BYTE
0073 85 06              STA   PTR               ;PLACE IN PTR
0075 A5 03              LDA   LOW+1             ;GET HIGH ORDER
0077 85 07              STA   PTR+1             ;
0079 60                 RTS

                        ;INCREMENT PTR & TEST FOR UPPER LIMIT

007A E6 06      INCPTR: INC   PTR               ;INCREMENT LOW ORDER
007C D0 02              BNE   INC1              ;BRANCH IF NO WRAP AROUND
007E E6 07              INC   PTR+1             ;BUMP HIGH ORDER
0080 A5 04      INC1:   LDA   HIGH              ;COMPARE WITH LOW BYTE OF LIMIT
0082 C5 06              CMP   PTR               ;
0084 D0 04              BNE   IPRET             ;BRANCH IF NOT AT LIMIT
0086 A5 05              LDA   HIGH+1            ;TEST HIGH BYTE
0088 C5 07              CMP   PTR+1             ;Z BIT WILL BE 1 IF LIMIT IS REACHED
008A 60         IPRET:  RTS

                        ;INCREMENT LOC AND CHECK FOR LIMIT
008B E6 00      INCLOC: INC   LOC               ;INCREMENT LOW ORDER BYTE
008D D0 02              BNE   INC2              ;BRANCH IF WRAP AROUND
008F E6 01              INC   LOC+1             ;BUMP HIGH ORDER BYTE
0091 A5 04      INC2:   LDA   HIGH              ;TEST LOW BYTE AGAINST LIMIT
0093 C5 00              CMP   LOC               ;
0095 D0 04              BNE   ILRET             ;BRANCH IF NOT AT LIMIT
0097 A5 05              LDA   HIGH+1            ;TEST HIGH BYTE
0099 C5 01              CMP   LOC+1             ;Z=1 IF AT LIMIT
009B 60         ILRET:  RTS

                        ;BIT TEST
009C A0 00      BISTR:  LDY#  $00               ;CLEAR Y
009E A9 01              LDA#  $01               ;GET STARTING BIT
00A0 85 08              STA   BTEST             ;SAVE IT
00A2 20 68 00   NEWBIT: JSR   RSTLOC            ;ZERO POINTER
00A5 20 71 00           JSR   RSTPTR            ;ZERO POINTER
00A8 A5 08      DEPLOC: LDA   BTEST             ;GET TEST VALUE
00AA 91 00              STA@Y LOC               ;STORE IN FIELD
00AC 20 8B 00           JSR   INCLOC            ;INCREMENT POINTER
00AF D0 F7              BNE   DEPLOC            ;BRANCH UNTIL FIELD FILLED
00B1 A5 08      BCHECK: LDA   BTEST             ;GET TEST VALUE
00B3 D1 06              CMP@Y PTR               ;COMPARE WITH FIELD
00B5 F0 01              BEQ   BIOK              ;BRANCH IF OK
00B7 00                 BRK                     ;BREAK IF ERROR
00B8 20 7A 00   BIOK:   JSR   INCPTR            ;INC POINTER
00BB D0 F4              BNE   BCHECK            ;BRANCH UNTIL FIELD TESTED
00BD 06 08              ASL   BTEST             ;MOVE BIT
```

```
00BF 90 E1          BCC   NEWBIT          ;BRANCH TIL ALL BITS CHECKED
00C1 A9 2A          LDA#  '*              ;INDICATE PASS
00C3 20 C6 72       JSR   WRT             ;
00C6 4C 9C 00       JMP   BISTR           ;AGAIN


                    ;TIME TEST

00C9 A0 00    TISTR: LDY#  $00            ;CLEAR Y
00CB A9 FF           LDA#  $FF            ;GET TEST VALUE
00CD 85 08           STA   BTEST          ;SAVE IT
00CF 20 68 00        JSR   RSTLOC         ;CLEAR POINTER
00D2 20 71 00        JSR   RSTPTR         ;
00D5 A5 08    TILOOP: LDA   BTEST         ;GET TEST VALUE
00D7 91 00           STA@Y LOC            ;STORE VALUE IN FIELD
00D9 20 8B 00        JSR   INCLOC         ;ADVANCE POINTER
00DC D0 F7           BNE   TILOOP
00DE A5 08    TIWRAP: LDA   BTEST         ;GET VALUE
00E0 D1 06           CMP@Y PTR            ;HAS IT CHANGED
00E2 F0 01           BEQ   TIOK           ;BRANCH IF OK
00E4 00              BRK                  ;BREAK IF CHANGED
00E5 20 7A 00 TIOK:  JSR   INCPTR         ;ADVANCE POINTER
00E8 D0 F4           BNE   TIWRAP         ;LOOP TILL UPPER LIMIT
00EA A9 2A           LDA#  '*             ;INDICATE PASS
00EC 20 C6 72        JSR   WRT            ;PRINT IT
00EF 20 71 00        JSR   RSTPTR         ;START AGAIN
00F2 4C DE 00        JMP   TIWRAP         ;
                     .END
```

```
00F5 .
72C6 WRT
0000 LOC
0002 LOW
0004 HIGH
0006 PTR
0008 BTEST
0010 MAD
0068 RSTLOC
0071 RSTPTR
0022 ML1
007A INCPTR
0028 TEST
0032 VLOOP
004D NEXTC
003D OK1
0044 OK2
0049 OK3
005D NOSTAR
008B INCLOC
0080 INC1
008A IPRET
0091 INC2
009B ILRET
009C BISTR
00A2 NEWBIT
00A8 DEPLOC
00B1 BCHECK
00B8 BIOK
00C9 TISTR
00D5 TILOOP
```

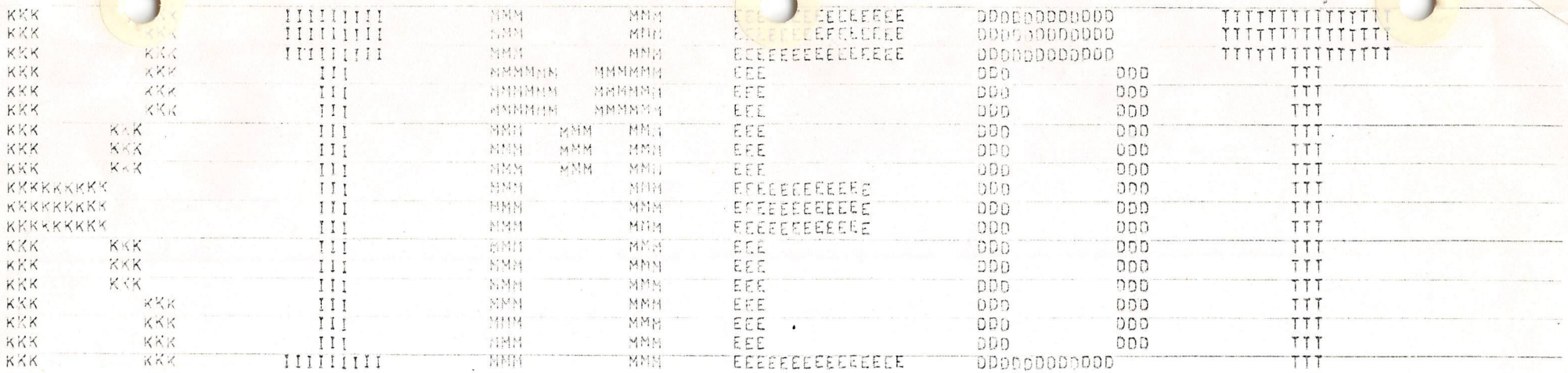
```
  00DE TIWRAP
  00E5 TIOK

NO ERRORS DETECTED

WHICH PASS (1 OR 2)? 2
DEVICE # FOR BINARY OUTPUT:0
DEVICE # FOR LISTING OUTPUT:4
WIDTH (127 MAX) OF
LISTING DEVICE:0
DEVICE # FOR SOURCE INPUT:3
READY SOURCE DEVICE
THEN HIT "RETURN" KEY  CR      (NOTE: TO GENERATE TAPE LEADER, YOU MAY
                                HIT "HEREIS" AS MANY TIMES AS
                                DESIRED AND THEN HIT "RETURN").
:00000000000000000000000000000009
:0B0010A900200C672A900A200C672200454
:0B001B6800207100A200A900810600F1
:0B0026207A00D0F7A9FFB10020710540
:0B003100A106F017A406C400F0010449
:0B003C00A407C401F00100C9FFF00560
:0B0047010A9D08100207A00D0E003C7
:0B0052A500D007A92A20C672A20004A6
:0B005D208802D0C9206800C4C10000390
:0B0068A5028500A503850160A50203D4
:0B00738506A503850760E606D0020458
:0B007EE607A504C506D004A505C5052D
:0B00890760E600D002E601A504C50508
:0B009400D004A505C50160A000A9048C
:0B009F01850820680020710050802FE
:0B00AA9100208B00D0F7A508D106053C
:0B00B5F00100207A00D0F400689004AD
:0B00C0E1A92A20C6724C9C00A000055F
:0B00CBA9FF850820680020710050C9
:0B00D6089100208B00D0F7A508D1056A
:0B00E106F00100207A00D0F4A92A0514
:0900EC20C67220071004CDE000408

NO ERRORS DETECTED
WHICH PASS (1 OR 2)?
```

(IF YOU WISH TO TERMINATE THE HEXTAPE ENTER ";00 HEREIS" AND TURN OFF THE PUNCH. OR FOR AUTOMATIC READERS: ";" "0" "0" "X-OFF" "HEREIS")



LPTSPL VERSION 6(344) RUNNING ON LPT000
START USER 6502 GROUP [7366,322] JOB TFOCAL SEQ. 4612 DATE 13-OCT-77 22:06:21 MONITOR CSM DECSYSTEM-10 603V1 *START*
REQUEST CREATED: 13-OCT-77 21:07:50
FILE: DSK3:KINEDT.PRI[7366,322,MARSH] CREATED: 13-OCT-77 20:54:00 <177> PRINTED: 13-OCT-77 22:27:50
QUEUE SWITCHES: /PRINT:ARROW /FILE:ASCII /COPIES:1 /SPACING:1 /LIMIT:343 /FORMS:NINE

```
PAL65   V002    13-OCT-77       20:52   PAGE 1

                        ;MINI EDITOR FOR 650X
                        ;COPYRIGHT:LARRY FISH 1977

                3600            .DEF    MSTAR=$3600     ;DEFINE THE STARTING ADDRESS
                                ;ZERO PAGE LOCATIONS
                0000            .LOC    $00
                0018            .DEF    ALTC=$18
0000  00 00       PA:           .WORD                   ;ASSEMBLERS TEXT POINTER
0002  00 00       PT:           .WORD                   ;POINTER TO THE END OF TEXT
0004  00 00       PD:           .WORD                   ;POINTER USED BY PRINT ROUTINE:ALSO TEMPORARY
0006  00 00       PE:           .WORD                   ;EDITOR'S POINTER
0008  00 00       PF:           .WORD                   ;TEMPORARY
000A  00          FLAG:         .BYTE                   ;TOGGLES FOR WRITE VS. PRINT ROUTINE
000B  95 39       PS:           .WORD   TBEG    ;SETS THE BEGINING OF TEXT AREA
000D  60          MASK1:        .BYTE   $60     ;MASK TO FIND CONTROL CHARACTERS
000E  00 00       CTP:          .WORD           ;TEMPORARY
0010  09          TAB:          .BYTE   STAB            ;TAB SIMULATOR
0011  09          TAB:          .BYTE   STAB    ;HOLDS NUMBER OF SPACES IN A TAB
0012  00          BF:           .BYTE           ;START OF 'FIND' COMMAND BUFFER AREA
                0009            .DEF    STAB=$09        ;DEFINES THE NUMBER OF SPACE FOR TABS
                3600            .DEF    MON=BUF ;MONITOR ENTRY POINT ****USER MAY SET TO JUMP TO ANY ROU
                                ;FOR KIM SET MON=$1C4F
```

MUST BE INITIALIZED

READ-ONLY (Never written)

```
                        .LOC  0

                        ;NOTE:  MACHINE LOADABLE OBJECT CODE IS PROVIDED
                        ;       WITH POINTERS SET AS BELOW.  THE PROGRAM
                        ;     WILL SELF DESTRUCT IF THE FIRST COMMAND
                        ;     AFTER A COLD START IS AN "E" WITH THE
                        ;     POINTERS SET TO ZERO.

0000 95 39      PA:     .WORD TBEG
0002 96 39      PT:     .WORD TBEG+1
0004 00 00      PD:     .WORD
0006 96 39      PE:     .WORD TBEG+1
0008 00 00      PF:     .WORD
000A 00         FLAG:   .BYTE
000B 95 39      PS:     .WORD TBEG
```

```
3987-   F0 08       BEQ   $3991
3989-   D0 03       BNE   $398E
398B-   20 0B 39    JSR   $390B
398E-   CA          DEX
398F-   D0 FA       BNE   $398B
3991-   4C 67 38    JMP   $3867
3994-   A9 1B       LDA   ##1B
3996-   20 80 17    JSR   $1780
3999-   4C 93 17    JMP   $1793
399C-   0D 0D 4E    ORA   $4E0D
399F-   4F          ???
39A0-   54          ???
39A1-   45 53       EOR   $53
39A3-   20 4F 4E    JSR   $4E4F
39A6-   20 50 52    JSR   $5250
39A9-   4F          ???
39AA-   44          ???
39AB-   55 43       EOR   $43,X
39AD-   49 4E       EOR   ##$4E
35F0-   20 DF 02    JSR   $02DF
35F3-   90 03       BCC   $35F8
35F5-   4C 00 36    JMP   $3600
35F8-   4C C2 36    JMP   $36C2
35FB-   F3          ???
35FC-   89          ???
35FD-   0D 08 41    ORA   $4108
3600-   A0 00       LDY   #$00
3602-   A9 23       LDA   ##$23
```

Patch for FIND cmd.

return

Patch for tape write close

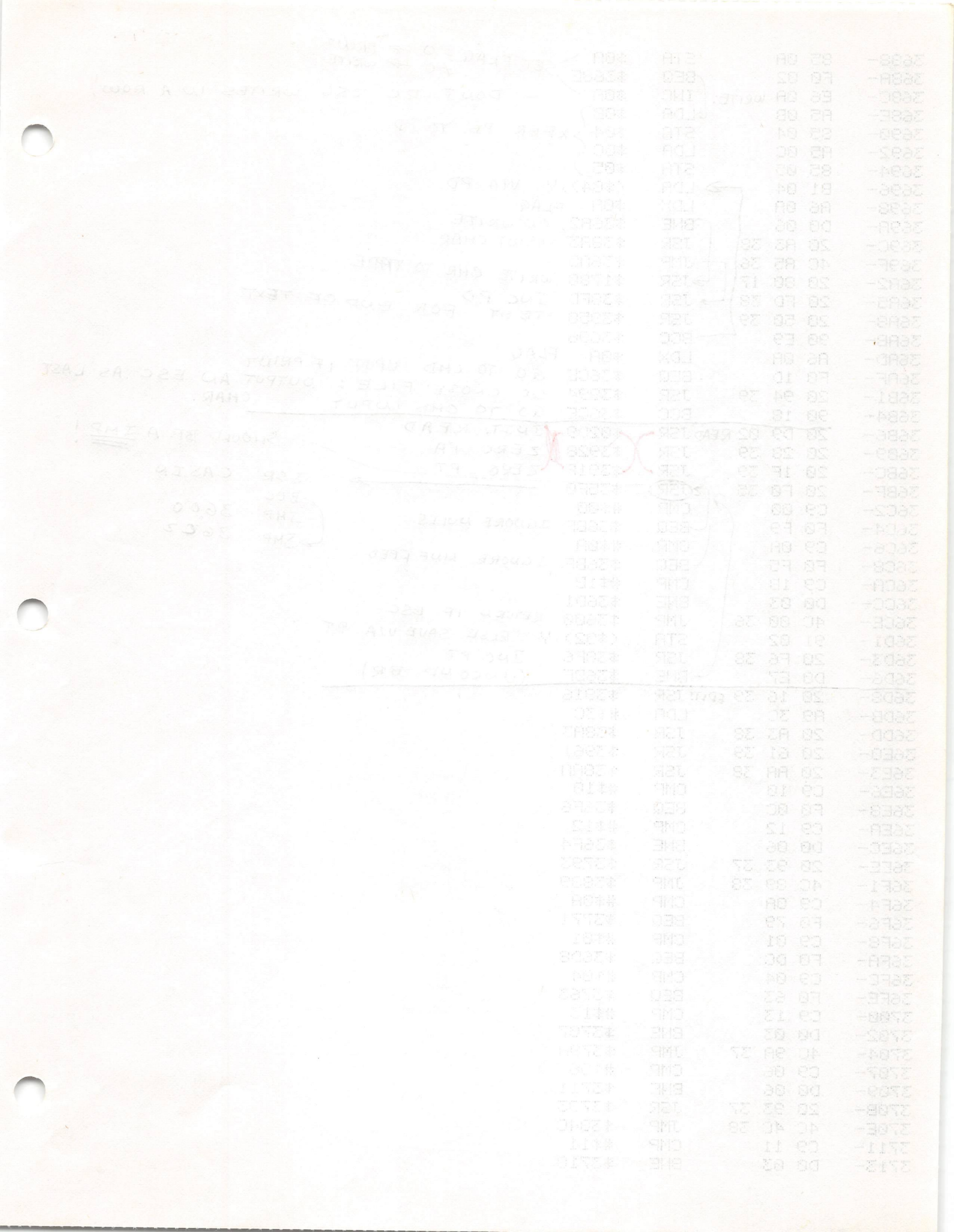
```
3918-   85 06        STA   $06
391A-   A5 0C        LDA   $0C
391C-   85 07        STA   $07
391E-   60           RTS
391F-   A5 0B        LDA   $0B
3921-   85 02        STA   $02
3923-   A5 0C        LDA   $0C
3925-   85 03        STA   $03
3927-   60           RTS
3928-   A5 0B        LDA   $0B
392A-   85 00        STA   $00
392C-   A5 0C        LDA   $0C
392E-   85 01        STA   $01
3930-   60           RTS
3931-   A5 07        LDA   $07
3933-   C5 0C        CMP   $0C
3935-   90 25        BCC   $395C
3937-   D0 26        BNE   $395F
3939-   A5 06        LDA   $06
393B-   C5 0B        CMP   $0B
393D-   90 1D        BCC   $395C
393F-   D0 1E        BNE   $395F
3941-   A5 02        LDA   $02
3943-   C5 06        CMP   $06
3945-   D0 18        BNE   $395F
3947-   A5 03        LDA   $03
3949-   C5 07        CMP   $07
394B-   D0 12        BNE   $395F
394D-   38           SEC
394E-   B0 10        BCS   $3960
3950-   A5 02        LDA   $02
3952-   C5 04        CMP   $04
3954-   D0 09        BNE   $395F
3956-   A5 03        LDA   $03
3958-   C5 05        CMP   $05
395A-   D0 03        BNE   $395F
395C-   38           SEC
395D-   B0 01        BCS   $3960
395F-   18           CLC
3960-   60           RTS
3961-   84 0E        STY   $0E
3963-   20 5A 1E     JSR   $1E5A
3966-   F0 FB        BEQ   $3963
3968-   A4 0E        LDY   $0E
396A-   60           RTS
396B-   00           BRK
396C-   00           BRK
396D-   84 0E        STY   $0E
396F-   20 A0 1E     JSR   $1EA0
3972-   A4 0E        LDY   $0E
3974-   60           RTS
3975-   48           PHA
3976-   A9 0D        LDA   #$0D
3978-   20 6D 39     JSR   $396D
397B-   A9 0A        LDA   #$0A
397D-   20 6D 39     JSR   $396D
3980-   A9 00        LDA   #$00
3982-   20 6D 39     JSR   $396D
3985-   68           PLA
3986-   60           RTS
```

```
38A6-   20 6D 39     JSR   $396D
38A9-   60           RTS
38AA-   48           PHA
38AB-   C9 10        CMP   #$10
38AD-   F0 2F        BEQ   $38DE
38AF-   C9 0A        CMP   #$0A
38B1-   F0 2B        BEQ   $38DE
38B3-   C6 10        DEC   $10
38B5-   F0 02        BEQ   $38B9
38B7-   10 03        BPL   $38BC
38B9-   20 EF 38     JSR   $38EF
38BC-   C9 09        CMP   #$09
38BE-   F0 20        BEQ   $38E0
38C0-   C9 1B        CMP   #$1B
38C2-   F0 1A        BEQ   $38DE
38C4-   C9 0D        CMP   #$0D
38C6-   F0 13        BEQ   $38DB
38C8-   24 0D        BIT   $0D
38CA-   D0 12        BNE   $38DE
38CC-   09 40        ORA   #$40
38CE-   48           PHA
38CF-   A9 5E        LDA   #$5E
38D1-   20 6D 39     JSR   $396D
38D4-   68           PLA
38D5-   20 6D 39     JSR   $396D
38D8-   4C DE 38     JMP   $38DE
38DB-   20 93 37     JSR   $3793
38DE-   68           PLA
38DF-   60           RTS
38E0-   A9 20        LDA   #$20
38E2-   20 6D 39     JSR   $396D
38E5-   C6 10        DEC   $10
38E7-   D0 F7        BNE   $38E0
38E9-   20 EF 38     JSR   $38EF
38EC-   4C DE 38     JMP   $38DE
38EF-   48           PHA
38F0-   A5 11        LDA   $11
38F2-   85 10        STA   $10
38F4-   68           PLA
38F5-   60           RTS
38F6-   E6 02        INC   $02
38F8-   D0 02        BNE   $38FC
38FA-   E6 03        INC   $03
38FC-   60           RTS
38FD-   E6 04        INC   $04
38FF-   D0 02        BNE   $3903
3901-   E6 05        INC   $05
3903-   60           RTS
3904-   E6 06        INC   $06
3906-   D0 02        BNE   $390A
3908-   E6 07        INC   $07
390A-   60           RTS
390B-   C6 06        DEC   $06
390D-   A5 06        LDA   $06
390F-   C9 FF        CMP   #$FF
3911-   D0 02        BNE   $3915
3913-   C6 07        DEC   $07
3915-   60           RTS
3915-   60           RTS
3916-   A5 0B        LDA   $0B
```

```
3820-   20 FD 38     JSR   $38FD
3823-   20 50 39     JSR   $3950
3826-   90 F1        BCC   $3819
3828-   20 2E 38     JSR   $382E
382B-   4C E0 36     JMP   $36E0
382E-   A9 5C        LDA   #$5C
3830-   20 A3 38     JSR   $38A3
3833-   C6 02        DEC   $02
3835-   A5 02        LDA   $02
3837-   C9 FF        CMP   #$FF
3839-   D0 02        BNE   $383D
383B-   C6 03        DEC   $03
383D-   60           RTS
383E-   20 41 39     JSR   $3941
3841-   90 C6        BCC   $3809
3843-   20 2E 38     JSR   $382E
3846-   20 0B 39     JSR   $390B
3849-   4C E0 36     JMP   $36E0
384C-   20 41 39     JSR   $3941
384F-   B0 2F        BCS   $3880
3851-   A2 00        LDX   #$00
3853-   20 61 39     JSR   $3961
3856-   C9 0D        CMP   #$0D
3858-   F0 07        BEQ   $3861
385A-   95 12        STA   $12,X
385C-   E8           INX
385D-   E0 0A        CPX   #$0A
385F-   D0 F2        BNE   $3853
3861-   A9 0D        LDA   #$0D
3863-   95 12        STA   $12,X
3865-   A2 00        LDX   #$00
3867-   B1 06        LDA   ($06),Y
3869-   85 08        STA   $08
386B-   20 04 39     JSR   $3904
386E-   20 41 39     JSR   $3941
3871-   B0 0D        BCS   $3880
3873-   B5 12        LDA   $12,X
3875-   E8           INX
3876-   C9 0D        CMP   #$0D
3878-   F0 0C        BEQ   $3886
387A-   C5 08        CMP   $08
387C-   4C 87 39     JMP   $3987
387F-   E5 20        SBC   $20
3881-   60           RTS
3882-   36 4C        ROL   $4C,X
3884-   E0 36        CPX   #$36
3886-   20 0B 39     JSR   $390B
3889-   A2 00        LDX   #$00
388B-   20 0B 39     JSR   $390B
388E-   E8           INX
388F-   B1 06        LDA   ($06),Y
3891-   C9 0D        CMP   #$0D
3893-   D0 F6        BNE   $388B
3895-   B1 06        LDA   ($06),Y
3897-   20 A3 38     JSR   $38A3
389A-   20 04 39     JSR   $3904
389D-   CA           DEX
389E-   D0 F5        BNE   $3895
38A0-   4C E0 36     JMP   $36E0
38A3-   20 AA 38     JSR   $38AA
```

```
3796-   20 EF 38     JSR   $38EF
3796-   20 EF 38     JSR   $38EF
3799-   60           RTS
379A-   20 0B 39     JSR   $390B
379D-   20 0B 39     JSR   $390B
37A0-   20 31 39     JSR   $3931
37A3-   B0 0C        BCS   $37B1
37A5-   B1 06        LDA   (#$06),Y
37A7-   C9 0D        CMP   ##$0D
37A9-   D0 F2        BNE   $379D
37AB-   20 04 39     JSR   $3904
37AE-   4C E0 36     JMP   $36E0
37B1-   4C D8 36     JMP   $36D8
37B4-   20 41 39     JSR   $3941
37B7-   90 06        BCC   $37BF
37B9-   20 63 36     JSR   $3663
37BC-   4C E0 36     JMP   $36E0
37BF-   A5 06        LDA   $06
37C1-   85 08        STA   $08
37C3-   85 04        STA   $04
37C5-   A5 07        LDA   $07
37C7-   85 09        STA   $09
37C9-   85 05        STA   $05
37CB-   20 FD 38     JSR   $38FD
37CE-   20 50 39     JSR   $3950
37D1-   B0 0E        BCS   $37E1
37D3-   B1 04        LDA   (#$04),Y
37D5-   C9 0D        CMP   ##$0D
37D7-   D0 F2        BNE   $37CB
37D9-   20 FD 38     JSR   $38FD
37DC-   20 50 39     JSR   $3950
37DF-   90 06        BCC   $37E7
37E1-   20 63 36     JSR   $3663
37E4-   4C F6 37     JMP   $37F6
37E7-   B1 04        LDA   (#$04),Y
37E9-   91 06        STA   (#$06),Y
37EB-   20 FD 38     JSR   $38FD
37EE-   20 04 39     JSR   $3904
37F1-   20 50 39     JSR   $3950
37F4-   90 F1        BCC   $37E7
37F6-   A5 06        LDA   $06
37F8-   85 02        STA   $02
37FA-   A5 07        LDA   $07
37FC-   85 03        STA   $03
37FE-   A5 08        LDA   $08
3800-   85 06        STA   $06
3802-   A5 09        LDA   $09
3804-   85 07        STA   $07
3806-   4C E0 36     JMP   $36E0
3809-   20 0B 39     JSR   $390B
380C-   20 31 39     JSR   $3931
380F-   B0 A0        BCS   $37B1
3811-   A5 06        LDA   $06
3813-   85 04        STA   $04
3815-   A5 07        LDA   $07
3817-   85 05        STA   $05
3819-   A0 01        LDY   ##$01
381B-   B1 04        LDA   (#$04),Y
381D-   88           DEY
381E-   91 04        STA   (#$04),Y
```

```
3715-   4C B4 37    JMP   $37B4
3718-   C9 1B       CMP   #$1B
371A-   D0 03       BNE   $371F
371C-   4C 00 36    JMP   $3600
371F-   C9 7F       CMP   #$7F
3721-   D0 03       BNE   $3726
3723-   4C 3E 38    JMP   $383E
3726-   48          PHA
3727-   20 41 39    JSR   $3941
372A-   90 04       BCC   $3730
372C-   68          PLA
372D-   4C 55 36    JMP   $3655
3730-   A5 02       LDA   $02
3732-   85 04       STA   $04
3734-   A5 03       LDA   $03
3736-   85 05       STA   $05
3738-   C6 04       DEC   $04
373A-   A5 04       LDA   $04
373C-   C9 FF       CMP   #$FF
373E-   D0 02       BNE   $3742
3740-   C6 05       DEC   $05
3742-   A0 00       LDY   #$00
3744-   B1 04       LDA   ($04),Y
3746-   C8          INY
3747-   91 04       STA   ($04),Y
3749-   A5 04       LDA   $04
374B-   C5 06       CMP   $06
374D-   D0 E9       BNE   $3738
374F-   A5 05       LDA   $05
3751-   C5 07       CMP   $07
3753-   D0 E3       BNE   $3738
3755-   20 F6 38    JSR   $38F6
3758-   A0 00       LDY   #$00
375A-   68          PLA
375B-   91 06       STA   ($06),Y
375D-   20 04 39    JSR   $3904
3760-   4C E0 36    JMP   $36E0
3763-   A5 02       LDA   $02
3765-   85 06       STA   $06
3767-   A5 03       LDA   $03
3769-   85 07       STA   $07
376B-   20 63 36    JSR   $3663
376E-   4C E0 36    JMP   $36E0
3771-   20 41 39    JSR   $3941
3774-   B0 17       BCS   $378D
3776-   B1 06       LDA   ($06),Y
3778-   20 04 39    JSR   $3904
377B-   C9 0D       CMP   #$0D
377D-   F0 06       BEQ   $3785
377F-   20 A3 38    JSR   $38A3
3782-   4C 71 37    JMP   $3771
3785-   20 93 37    JSR   $3793
3788-   20 41 39    JSR   $3941
378B-   90 03       BCC   $3790
378D-   20 63 36    JSR   $3663
3790-   4C E0 36    JMP   $36E0
3793-   20 75 39    JSR   $3975
```



```
3688-  85 0A        STA   $0A      \ SET FLAG = 0 => PRINT
368A-  F0 02        BEQ   $368E    /            ≠0 => WRITE
368C-  E6 0A WRITE: INC   $0A      - DON'T DO 256 WRITES IN A ROW!
368E-  A5 0B        LDA   $0B      \
3690-  85 04        STA   $04      | XFER PS TO PD
3692-  A5 0C        LDA   $0C      |
3694-  85 05        STA   $05      /
3696-  B1 04        LDA   ($04),Y  VIA PD
3698-  A6 0A        LDX   $0A      FLAG
369A-  D0 06        BNE   $36A2    ON WRITE
369C-  20 A3 38     JSR   $38A3    PRINT CHAR
369F-  4C A5 36     JMP   $36A5
36A2-  20 80 17     JSR   $1780    WRITE CHR TO TAPE
36A5-  20 FD 38     JSR   $38FD    INC PD
36A8-  20 50 39     JSR   $3950    TEST FOR END OF TEXT
36AB-  90 E9        BCC   $3696
36AD-  A6 0A        LDX   $0A      FLAG
36AF-  F0 1D        BEQ   $36CE    GO TO CMD INPUT IF PRINT
36B1-  20 94 39     JSR   $3994    GO CLOSE FILE: OUTPUT AN ESC AS LAST
36B4-  90 18        BCC   $36CE    GO TO CMD INPUT          CHAR.
36B6-  20 D9 02 READ JSR  $02D9    INIT. READ
36B9-  20 28 39     JSR   $3928    ZERO PA
36BC-  20 1F 39     JSR   $391F    ZERO PT
36BF-  20 F0 35     JSR   $35F0    SHOULD BE A JMP!
36C2-  C9 00        CMP   #$00
36C4-  F0 F9        BEQ   $36BF    IGNORE NULLS
36C6-  C9 0A        CMP   #$0A
36C8-  F0 F5        BEQ   $36BF    IGNORE LINE FEED
36CA-  C9 1B        CMP   #$1B
36CC-  D0 03        BNE   $36D1
36CE-  4C 00 36     JMP   $3600    RETURN IF ESC
36D1-  91 02        STA   ($02),Y  ELSE SAVE VIA PT
36D3-  20 F6 38     JSR   $38F6    INC PT
36D6-  D0 E7        BNE   $36BF    (UNCOND. BR)
36D8-  20 16 39 EDIT: JSR $3916
36DB-  A9 3C        LDA   #$3C
36DD-  20 A3 38     JSR   $38A3
36E0-  20 61 39     JSR   $3961
36E3-  20 AA 38     JSR   $38AA
36E6-  C9 10        CMP   #$10
36E8-  F0 0C        BEQ   $36F6
36EA-  C9 12        CMP   #$12
36EC-  D0 06        BNE   $36F4
36EE-  20 93 37     JSR   $3793
36F1-  4C 89 38     JMP   $3889
36F4-  C9 0A        CMP   #$0A
36F6-  F0 79        BEQ   $3771
36F8-  C9 01        CMP   #$01
36FA-  F0 DC        BEQ   $36D8
36FC-  C9 04        CMP   #$04
36FE-  F0 63        BEQ   $3763
3700-  C9 13        CMP   #$13
3702-  D0 03        BNE   $3707
3704-  4C 9A 37     JMP   $379A
3707-  C9 06        CMP   #$06
3709-  D0 06        BNE   $3711
370B-  20 93 37     JSR   $3793
370E-  4C 4C 38     JMP   $384C
3711-  C9 11        CMP   #$11
3713-  D0 03        BNE   $3718
```

JSR CASIN
BCC
JMP 3600
JMP 3622

TO APPEND A LINE

```
                                    6100
APNDLINE:  JSR   CASIN             6100  20 00 DF 02
           BCS   RTRN              0003  B0 00
           CMP#  0                 0005  C9 00
           BEQ   APNDLINE          0007  F0 F7
           CMP#  $0A               0009  C9 0A
           BEQ   APNDLINE          000B  F0 F3
           CMP#  $1B               000D  C9 1B
           BNE   SAVE              000F  D0 03
RTRN!      JMP   EDITOR            0011  4C 00 00 36
SAVE!      STA@Y $02               0014  91 02
           JSR   INCPT             0016  20 F6 38
           CMP#  $0D               0019  C9 0D
           BNE   APNDLINE          001B  D0 E3
           BEQ   RTRN              001D  F0 F2
```

1D
E3

```
3677-   D0 0B        BNE   $3684
3679-   A6 03        LDX   $03
367B-   E4 01        CPX   $01
367D-   D0 05        BNE   $3684
367F-   48           PHA
3680-   20 28 39     JSR   $3928
3683-   68           PLA
3684-   18           CLC
3685-   60           RTS


3686-   A9 00        LDA   #$00
3688-   85 0A        STA   $0A
368A-   F0 02        BEQ   $368E
368C-   E6 0A        INC   $0A
```

```
35FØ   2Ø DF Ø2     JSR   CASINT
35F3   9Ø Ø3        BCC
35F5   4C ØØ 36     JMP   $3600 - EDITOR
35F8   4C C2 36     JMP   $36C2 - ~~WRITE OUT~~ READIN

3600-  A0 00        LDY   #$00
3602-  A9 23        LDA   #$23
3604-  20 A3 38     JSR   $38A3
3607-  20 61 39     JSR   $3961
360A-  20 93 37     JSR   $3793
360D-  C9 4C        CMP   #$4C
360F-  D0 03        BNE   $3614
3611-  4C 47 36     JMP   $3647
3614-  C9 45        CMP   #$45
3616-  D0 03        BNE   $361B
3618-  4C D8 36     JMP   $36D8
361B-  C9 57        CMP   #$57
361D-  D0 06        BNE   $3625
361F-  20 61 39     JSR   $3961
3622-  4C 8C 36     JMP   $368C
3625-  C9 52        CMP   #$52
3627-  D0 03        BNE   $362C
3629-  4C B6 36     JMP   $36B6
362C-  C9 43        CMP   #$43
362E-  D0 03        BNE   $3633
3630-  4C 63 37     JMP   $3763
3633-  C9 50        CMP   #$50
3635-  D0 03        BNE   $363A
3637-  4C 86 36     JMP   $3686
363A-  C9 4D        CMP   #$4D
363C-  D0 03        BNE   $3641
363E-  4C 4F 1C     JMP   $1C4F
3641-  20 93 37     JSR   $3793
3644-  4C 00 36     JMP   $3600
3647-  20 28 39     JSR   $3928
364A-  20 1F 39     JSR   $391F
364D-  20 16 39     JSR   $3916
364D-  20 16 39     JSR   $3916
3650-  20 63 36     JSR   $3663
3653-  A9 0D        LDA   #$0D
3655-  91 02        STA   ($02),Y
3657-  20 F6 38     JSR   $38F6
365A-  20 04 39     JSR   $3904
365D-  4C E0 36     JMP   $36E0
3660-  20 93 37     JSR   $3793
3663-  A9 3E        LDA   #$3E
3665-  20 A3 38     JSR   $38A3
3668-  60           RTS
3669-  A0 00        LDY   #$00
366B-  B1 00        LDA   ($00),Y
366D-  E6 00        INC   $00
366F-  D0 02        BNE   $3673
3671-  E6 01        INC   $01
3673-  A6 02        LDX   $02
3675-  E4 00        CPX   $00
```

EDITOR MODS

```
35CØ:  C9 4C      CMDPTCH: CMP#   'L
  C2:  DØ Ø3               BNE    TI
  C4:  4C DC 35            JMP    PRINT
  C7:  C9 49       TI:     CMP#   'I
  C9:  DØ Ø5 78            BNE    TS
  CB:  2Ø AS 5E            JSR    PR.INI
  CE:  9Ø 3Ø               BCC    BUF
  DØ:  C9 53       TS:     CMP#   'S
  D2:  DØ 6D               BNE    TZ        ($3641)
  D4:  4C BC 36            JMP    RD.APND
  D7:      BF for          NOP
  D8:      no reinit       NOP
  D9:                      NOP
  DA:                      NOP
  DB:                      NOP
  DC:  2Ø 69 36    PRINT:  JSR    READC
  DF:  C9 1A               CMP#   'AZ
  E1:  FØ 1D               BEQ    BUF
  E3:  C9 ØD               CMP#   $ØD
  E5:  DØ ØZ               BNE    PR.
  E7:  A9 ØA 78            LDA#   $ØA
  E9:  2Ø B1 5E    PR.     JSR    PRINT
  EC:  9Ø EE               BCC    PRINT
  EE:  BØ 1Ø               BCS    BUF
  ---------------------------------------
  F0:  2Ø DF ØZ            JSR    CASIO
  F3:  BØ 9Ø Ø3            BCS    BUF
  F5:  4C ØØ 36            JMP    BUF
  F8:  4C C2 36    CONT:   JMP    READIN
  FB:
  FC:
  FD:
  FE:
  FF:
```

Patch in additional commands

print buffer to CTRL-Z — change CR to LINEFEED

DH 6D 41

28 -16 12

42-381 50 SHEETS 5 SQUARE
42-382 100 SHEETS 5 SQUARE
42-389 200 SHEETS 5 SQUARE
Made in U.S.A.
NATIONAL

Editor looks OK for PROM! 8 Sept '80


```
                                .PAGE
                        3600    .LOC    MSTAR   ;START OF EDITOR
                                ;PROCESS NON-EDITTING COMMANDS?
3600  A0 00    BUF:    LDY#    $00     ;CLEAR Y REGISTER FOR ALL EDITING OPERATIONS
3602  A9 23           LDA#    '#      ;PROMPT
3604  20 A338         JSR     ECHOAW  ;PRINT PROMPT CHARACTER
3607  20 6139         JSR     RSY     ;GET COMMAND
360A  20 9337         JSR     CRTAB   ;PRINT CARRIAGE RETURN AFTER COMMAND IS ENTERED
360D  C9 4C           CMP#    'L      ;IS IT A LOAD KEYBOARD COMMAND?
360F  D0 03           BNE     TE      ;BRANCH IF NOT
3611  4C 4736         JMP     LDKY    ;JUMP TO KEYBOARD INPUT ROUTINE
3614  C9 45    TE:     CMP#    'E      ;IS IT A GO TO EDITOR COMMAND?
3616  D0 03           BNE     TW      ;BRANCH IF NOT
3618  4C D836         JMP     EDIT    ;JUMP TO THE EDITOR ENTRY POINT
361B  C9 57    TW:     CMP#    'W      ;IS IT A WRITE TAPE COMMAND?
361D  D0 06           BNE     TR      ;BRANCH IF NOT
361F  20 6139         JSR     RSY     ;READ A CHARATER BEFORE WE WRITE
3622  4C 8C36         JMP     WRITE   ;JUMP TO WRITE ROUTINE
3625  C9 52    TR:     CMP#    'R      ;IS IT A READ TAPE COMMAND?
3627  D0 03           BNE     TC      ;BRANCH IF NOT
3629  4C B936         JMP     READ    ;JUMP TO THE READ ROUTINE
362C  C9 41    TC:     CMP#    'A      ;IS IT A CONTINUE FROM END OF TEXT COMM.?
362E  D0 03           BNE     TP      ;BRANCH IF NOT
3630  4C 6337         JMP     ENT     ;JUMP TO END OF TEXT ENTRY POINT
3633  C9 50    TP:     CMP#    'P      ;IS IT A PRINT TEXT COMMAND
3635  D0 03           BNE     TM      ;BRANCH IF NOT
3637  4C 8636         JMP     PRINT   ;GO TO PRINT ROUTINE
363A  C9 4D    TM:     CMP#    'M      ;IS IT A GO TO MONITOR COMMAND?
363C  D0 03           BNE     TZ      ;BRANCH IF NOT
363E  (4C 0036)->     JMP     MON     ;GO TO THE MONITOR
3641  20 9337 TZ:     JSR     CRTAB   ;PRINT A CR AND LF
3644  4C 0036         JMP     BUF     ;START AGAIN IF ITS NOT A COMMAND
```



```
                                .PAGE
                                ;MAIN EDITING ROUTINES
3647  20 2839   LDKY:   JSR     ZPA     ;ENTRY POINT THAT CLEARS BUFFER
364A  20 1F39           JSR     ZPT     ;ZERO POINTERS
364D  20 1639           JSR     ZPE     ;
3650  20 6336           JSR     ENDP    ;TELL USER WE ARE AT THE END OF TEXT
3653  A9 0D             LDA#    $0D     ;FIRST CHARACTER OF TEXT WILL ALWAYS BE A CR
3655  91 06     STAE:   STA@Y   PE      ;STORE A CHARACTER IN THE TEXT
3657  20 F638           JSR     INCPT   ;MOVE POINTERS
365A  20 0439           JSR     INCPE   ;
365D  4C E036           JMP     CE01    ;JUMP TO MAIN ENTRY POINT TO EDITOR
                        ;ROUTINE TO PROMPT FOR END OF TEXT
3660  20 9337 CENDP:   JSR     CRTAB   ;PRINT CRLF
3663  A9 3E    ENDP:   LDA#    '>      ;GET A '>'
3665  20 A338           JSR     ECHOAW  ;PRINT IT
3668  60                RTS             ;RETURN
```

```
                                .PAGE
                                ;ROUTINE FOR ASSEMBLER TO READ A CHARACTER
3669  A0 00     AREAD:  LDY#    00          ;ZERO Y REGISTER FOR INDEXED FETCH OF CHARACTERS
366B  B1 00             LDA@Y   PA          ;GET CHARACTER
366D  E6 00             INC     PA          ;ADVANCE POINTERS
366F  D0 02             BNE     NIPA        ;
3671  E6 01             INC     PA+1        ;
3673  A6 02     NIPA:   LDX     PT          ;ARE WE AT END OF TEXT?
3675  E4 00             CPX     PA          ;COMPARE HIGH BYTE
3677  D0 0B             BNE     CLEAR       ;BRANCH IF NOT END OF TEXT
3679  A6 03             LDX     PT+1        ;COMPARE LOW ORDER BYTE
367B  E4 01             CPX     PA+1        ;
367D  D0 05             BNE     CLEAR       ;BRANCH IF NOT THE END OF TEXT
367F  48                PHA                 ;SAVE CHARACTER
3680  20 2839           JSR     ZPA         ;RESET POINTER IF END OF TEXT
3683  68                PLA                 ;GET BACK CHARACTER
3684  18      CLEAR:  CLC                 ;CLEAR CARRY BIT TO MAKE ASSEMBLER HAPPY
3685  60              RTS                 ;RETURN
```



```
                                .PAGE
                                ;ROUNTINES TO PRINT TEXT OR TO WRITE TAPE
3686  A9 00     PRINT:  LDA#    $00         ;SET FLAG FOR PRINT ROUTINE
3688  85 0A             STA     FLAG        ;ZERO FLAG
368A  F0 02             BEQ     WPS         ;UNCONDITIONAL BRANCH
368C  E6 0A     WRITE:  INC     FLAG        ;SET FLAG FOR WRITE TAPE ROUTINE
368E  A5 03     WPS:    LDA     PS          ;ZERO POINTERS
3690  85 04             STA     PD          ;
3692  A5 0C             LDA     PS+1        ;
3694  85 05             STA     PD+1        ;
3696  B1 04     PNC:    LDA@Y   PD          ;GET A CHARACTER
3698  A6 0A             LDX     FLAG        ;TEST IF THIS A PRINT OR A WRITE ROUTINE
369A  D0 05             BNE     WTO         ;BRANCH IF IT IS WRITE
369C  20 A338   WECH:   JSR     ECHOAW      ;PRINT IT (WITH ECHOED CONTROL CHARACTERS ETC.)
369F  4C AE36           JMP     NWTJ        ;JUMP PAST WRITE ROUTINE
36A2  C9 0D     WTO:    CMP#    $0D         ;CR ARE ECHOED AS CRLF
36A4  D0 05             BNE     NWCR        ;BRANCH IF NOT CR
36A6  20 7539           JSR     ACRLF       ;PRINT CRLF
36A9  D0 03             BNE     NWTJ        ;UNCONDITAL BRANCH
36AB  20 6D39   NWCR:   JSR     SWRT        ;WRITE IT TO CASSETTE
36AE  20 FD38   NWTJ:   JSR     INCPD       ;ADVANCE POINTERS
36B1  20 5039           JSR     TSTPT       ;TEST FOR END OF TEXT
36B4  90 E0             BCC     PNC         ;BRANCH IF NOT
36B6  4C 8B 39          JMP     WRTEOT      ;ALLOW INPUT FOR END OF TAPE
                        ;ROUTINE READS FROM TAPE AND STORES IN TEXT BUFFER
36B9  20 2339   READ:   JSR     ZPA         ;ZERO ASSEMBLERS READ POINTER
36BC  20 1F39           JSR     ZPT         ;ZERO END OF TEXT POINTER
36BF  20 6139   RRDT:   JSR     RSY         ;READ A CHARACTER FROM TAPE
36C2  C9 00             CMP#    $00         ;STRIP OUT NULLS FOR PAPER TAPE LEADER
36C4  F0 F9             BEQ     RRDT        ;BRANCH IF A NULL
36C6  C9 0A             CMP#    $0A         ;STRIP LINE FEEDS
36C8  F0 F5             BEQ     RRDT        ;BRANCH IF LINE FEED
36CA  C9 1B             CMP#    ALTC        ;IS IT AN ALTMODE CHAR.; ALTMODE TERMINATES TAPE READ
36CC  D0 03             BNE     RNES        ;CONTINUE THE READ IF NO ALTMODE
36CE  4C 0036           JMP     BUF         ;GO BACK FOR NEW COMMAND IF ALTMODE
36D1  91 02     RNES:   STA@Y   PT          ;STORE CHARACTER IN TEXT AREA
36D3  20 F638           JSR     INCPT       ;ADVANCE POINTER
36D6  D0 E7             BNE     RRDT        ;UNCONDITIONAL
```

```
;USE ↑X as prefix to insert any character,
;including ruboot:
CED1:   JSR     RSY
        CMPA    '↑X
        BNE     F.1
        JSR     RSY
        JSR     ECHO
        JMP     ADDC
F.1:    JSR     ECHO
```

Adds 13 bytes

```
                                .PAGE
                                ;EDITOR COMMAND DECODER
3608  20 1639  EDIT:  JSR     ZPE     ;ZERO EDITOR POINTER
360B  A9 3C    EDPR1: LDA#    $3C     ;EDITOR ENTRY POINT STARTING AT BEGINING OF TEXT
360D  20 A338  Unused JSR     ECHOAW  ;PROMPT BEGINING OF TEXT
36E0  20 6139  CED1:  JSR     RSY     ;READ A CHARACTER FROM THE KEYBOARD
36E3  20 AA38         JSR     ECHO    ;ECHO CONTROL CHARACTERS ETC.
36E6  C9 10           CMP#    $10     ;IS IT A CONTROL 'P'
36E8  F0 0C           BEQ     SLINE1  ;BRANCH TO ADVANCE A LINE ROUTINE
36EA  C9 12           CMP#    $12     ;IS IT A CONTROL R?
36EC  D0 06           BNE     NCR     ;BRANCH IF NOT
36EE  20 9337         JSR     CRTAB   ;PRINT CRLF
36F1  4C 8938         JMP     CONR    ;REPRINT FROM BEGINING OF LINE
36F4  C9 0A    NCR:   CMP#    $0A     ;IS IT A LINE FEED; LF ADVANCES EDITOR ONE LINE
36F6  F0 79    SLINE1: BEQ    LINE    ;BRANCH TO ADVANCE A LINE ROUTINE
36F8  C9 01           CMP#    $01     ;IS IT A 'A'
36FA  F0 DC           BEQ     EDIT    ;BRANCH TO PLACE EDITOR AT BEGINING OF TEXT
36FC  C9 04           CMP#    $04     ;IS IT A CONTROL 'D'?
36FE  F0 63           BEQ     ENT     ;BRANCH TO PLACE EDITOR AT END OF TEXT
3700  C9 13           CMP#    $13     ;IS IT A CONTROL 'S'?
3702  D0 03           BNE     NCS     ;BRANCH IF NOT
3704  4C 9A37         JMP     BACK    ;JUMP TO BACK UP A LINE
3707  C9 06    NCS:   CMP#    $06     ;IS IT A CONTROL 'F'?
3709  D0 06           BNE     NCF     ;BRANCH IF NOT
370B  20 9337         JSR     CRTAB   ;PRINT CRLF BEFORE SEARCH STRING
370E  4C 4C38         JMP     FIND    ;JUMP TO FIND A CHARACTER STRING
3711  C9 11    NCF:   CMP#    $11     ;IS IT A CONTROL 'Q'
3713  D0 03           BNE     NCQ     ;BRANCH IF NOT
3715  4C B437         JMP     DLIN    ;JUMP TO DELETE A LINE FORWARD
3718  C9 1B    NCQ:   CMP#    ALTC    ;IS IT AN ALTMODE?
371A  D0 03           BNE     NESC    ;BRANCH IT NOT
371C  4C 0036         JMP     BUF     ;JUMP BACK TO NON-EDITOR COMMAND DECODER
371F  C9 7F    NESC:  CMP#    $7F     ;IS IT A RUBOUT?
3721  D0 03           BNE     ADDC    ;ALL COMMAND TESTS HAVE FAILED,MUST BE A TEXT CHARACTER
3723  4C 3E38         JMP     CDEL    ;JUMP TO DELETE A CHARACTER
```



```
                                .PAGE
                                ;ROUTINE INSERTS CHARACTERS IN MIDDLE OR AT THE END OF THE TEXT
3726  48       ADDC:  PHA     ;SAVE CHARACTER TO BE INSERTED
3727  20 4139         JSR     CPTPE   ;ARE WE AT THE END OF TEXT?
372A  90 04           BCC     NET     ;BRANCH IF WE ARE IN THE MIDDLE OF TEXT
372C  68              PLA             ;GET BACK CHARACTER
372D  4C 5536         JMP     STAE    ;JUMP TO STORE AT THE END OF TEXT
3730  A5 02    NET:   LDA     PT      ;ROUTINE INSERTS CHARACTER IN MIDDLE OF TEXT
3732  85 04           STA     PD      ;POINTERS PD PE AND PT ARE USED TO MOVE ALL TEXT
3734  A5 03           LDA     PT+1    ;DOWN. SET PD=PT
3736  85 05           STA     PD+1    ;
3738  C6 04    SAC:   DEC     PD      ;DECREMENT PD TO POINT TO CHARACTER TO BE MOVED
373A  A5 04    LOOP   LDA     PD      ;HAS THE DECREMENT WRAPPED AROUND?
373C  C9 FF    TIME:  CMP#    $FF     ;FF SAYS DECREMENT HAS WRAPPED AROUND
373E  D0 02    ~37 CYCLES BNE NDE     ;BRANCH IF NO WRAP
3740  C6 05    PER LOOP DEC   PD+1    ;DECREMENT HIGH ORDER IF WE HAVE WRAPPED
3742  A0 00    NDE:   LDY#    00      ;ZERO INDEX; INDEX IS USED TO SHIFT CHARACTERS
3744  B1 04    ONLY   LDA@Y   PD      ;PICK UP FIRST CHARACTER
3746  C8              INY             ;INCREMENT Y REGISTER TO MOVE CHARACTER
3747  91 04           STA@Y   PD      ;STORE CHARACTER VIA SHIFTED INDEX
3749  A5 04           LDA     PD      ;COMPARE PE AND PD
374B  C5 06           CMP     PE      ;IF PE=PD THEN WE HAVE SHIFTED ALL THE CHARACTERS
374D  D0 E9           BNE     SAC     ;LOOP UNTIL ALL CHARACTER AHEAD OF PE ARE SHIFTED
374F  A5 05           LDA     PD+1    ;COMPARE HIGH BYTES
3751  C5 07           CMP     PE+1    ;
3753  D0 E3           BNE     SAC     ;BRANCH IF NOT DONE
3755  20 F638         JSR     INCPT   ;CLEAN UP PT
3758  A0 00           LDY#    00      ;ZERO Y REGISTER
375A  68              PLA             ;GET BACK CHARACTER TO BE INSERTED
375B  91 06           STA@Y   PE      ;STORE IT
375D  20 3439         JSR     INCPE   ;CLEAN UP PE
3760  4C E236         JMP     CED1    ;JUMP TO GET NEW CHARACTER
```

```
                                ,PAGE
                                ;MOVE EDITOR TO THE END OF TEXT
3763  A5 02   ENT:   LDA   PT     ;MAKE PE EQUAL TO PT
3765  85 06          STA   PE     ;
3767  A5 03          LDA   PT+1   ;DO HIGH ORDER BYTES
3769  85 07          STA   PE+1   ;
376B  20 6336        JSR   ENDP   ;PROMPT FOR END OF TEXT
376E  4C E036        JMP   CED1   ;GO BACK FOR NEW CHARACTER OR COMMAND
                                ;MOVE FORWARD A LINE
3771  20 4139 LINE:  JSR   CPTPE  ;ARE WE AT THE END OF TEXT?
3774  B0 17          BCS   EOT    ;BRANCH IF YES
3776  B1 06          LDA@Y PE     ;GET A CHARACTER FROM THE TEXT
3778  20 0439        JSR   INCPE  ;ADVANCE POINTER
377B  C9 0D          CMP#  $0D    ;ARE WE AT THE END OF THE LINE?
377D  F0 06          BEQ   LEOL   ;BRANCH IF YES
377F  20 A338        JSR   ECHOAW ;PRINT EACH CHARACTER IN THE LINE AS WE PASS
3782  4C 7137        JMP   LINE   ;LOOP UNTIL WE ARE AT THE BEGIN OF A NEW LINE
3785  20 9337 LEOL:  JSR   CRTAB  ;PRINT A CARRIAGE RETURN LINE FEED
3788  20 4139        JSR   CPTPE  ;ARE WE AT THE END OF TEXT?
378B  90 03          BCC   MNET   ;BRANCH IF NOT
378D  20 6336 EOT:   JSR   ENDP   ;PROMPT FOR END OF TEXT
3790  4C E036 MNET:  JMP   CED1   ;BACK FOR NEW CHARACTER OR COMMAND
                                ;PRINTS CRLF AND ZEROS TAB COUNTER
3793  20 7539 CRTAB: JSR   ACRLF  ;PRINT CRLF
3796  20 EF38        JSR   TAZR   ;RESTART TAB COUNTER
3799  60             RTS          ;
```



```
                                ,PAGE
                                ;BACK UP A LINE
379A  20 0839 BACK:  JSR   DECPE  ;BACK UP ONE CHARACTER
379D  20 0839 SDEC:  JSR   DECPE  ;BACK UP ONE CHARACTER
37A0  20 3139        JSR   TEPREG ;PAST BEGINING?
37A3  B0 0C          BCS   JED    ;BRANCH IF WE ARE
37A5  B1 06          LDA@Y PE     ;GET THE CHARACTER THERE
37A7  C9 0D          CMP#  $0D    ;IS IT A CARRIAGE?
37A9  D0 F2          BNE   SDEC   ;BRANCH IF NOT
37AB  20 0439 JOT:   JSR   INCPE  ;CLEAN UP PE
37AE  4C E036        JMP   CED1   ;BACK FOR NEW CHARACTER OR COMMAND
37B1  4C D836 JED:   JMP   EDIT   ;RESET POINTERS TO BEGINING OF TEXT
```

[This is for BS on rubout] 28Feb82

ALGORITHMIC DESCRIPTION FOR DELETE OPERATION:

```
IF (char.to.delete = c.r)
     reprint line to this point
  else{if (char.to.delete = control.char)
     backspace - space- backspace}
  else
     backspace - space- backspace
```

Delete character at PE@ -1

BUT BECAUSE OF THE EXISTING FRAMEWORK, THIS WILL BE DONE IN REVERSE ORDER - - -

```
SAVE CHAR TO BE DELETED
IF AT BEGINNING OF TEXT, PROMT IT
DELETE CHAR

IF DELETED CHAR = CR or TAB
     REPRINT CURRENT LINE TO THIS POINT
ELSE
   {IF DELETED CHAR = CONTROL CHAR; BS/SP/BS}

   BS/SP/BS
```

Note: ECHOAW should add to tab ctr. for BS I guess-

28Feb82

PD
−PT
no borrow if PD ≥ PT

```
3          CDEL:   JSR   DECPE     ; POINT TO CHAR TO DEL
                   LDA@Y PE        ; GET IT
                   PHA             ; SAVE IT
3                  JSR   TEPBEG    ; PAST BEGINNING?
2                  BCS   JED       ; PROMPT & QUIT IF SO
2                  LDA@Y PE        ; GET CHAR TO DELETE
1                  PHA             ; SAVE IT
           ; CLOSE UP TEXT:
2                  LDA   PE        ; XFER TEXT PTR
2                  STA   PD        ; TO TEMP (NOW
2                  LDA   PE+1      ; PTS TO CHAR TO
2                  STA   PD+1      ; DELETE)
2          DLTLUP: LDY#  $01
2                  LDA@Y PD        ; MOVE ONE CHAR.
2                  DEY
2                  STA@Y PD
6                  JSR   INCPD     ; INC MOVE POINTER
2                  LDA   PD        ; CK FOR END OF TEXT
2                  CMP   PT        ; GO TILL PD>=PT
                   [illegible]
2                  LDA   PD+1
2                  SBC   PT+1
2                  BBS   DLTLUP
10         ; CHECK DELETED CHAR:
1                  PLA
2                  CMP#  $0D       ; CR?
2                  BEQ   CONRX     ; REPRINT LINE IF SO
                   [illegible] CMP# $09 ; TAB?
                               BEQ CONRX
2                  BIT   MASK1     ; CONTROL CHAR?
2                  BNE   F.3
3                  JSR   BSSPBS    ; BACK UP 1
3          F.3:    JSR   BSSPBS    ; BACK UP 1
3                  JMP   CED1      ; GO EDIT MORE
2          BSSPBS: LDA#  $08
3                  JSR   ECHOAW
2                  LDA#  $20
3                  JSR   ECHOAW
2                  LDA#  $08
3                  JMP   ECHOAW
```

```
INC  PD
BNE  F.1
INC  PD+1
```

```
LDX  PT
PHP
DEX
STX  PT
PLP
BNE  F.2
DEC  PT+1
```

11 bytes

```
DEC  PT
LDA  PT
CMP# $FF
BNE  F.2
DEC  PT+1
```

86₁₀ BYTES = $56

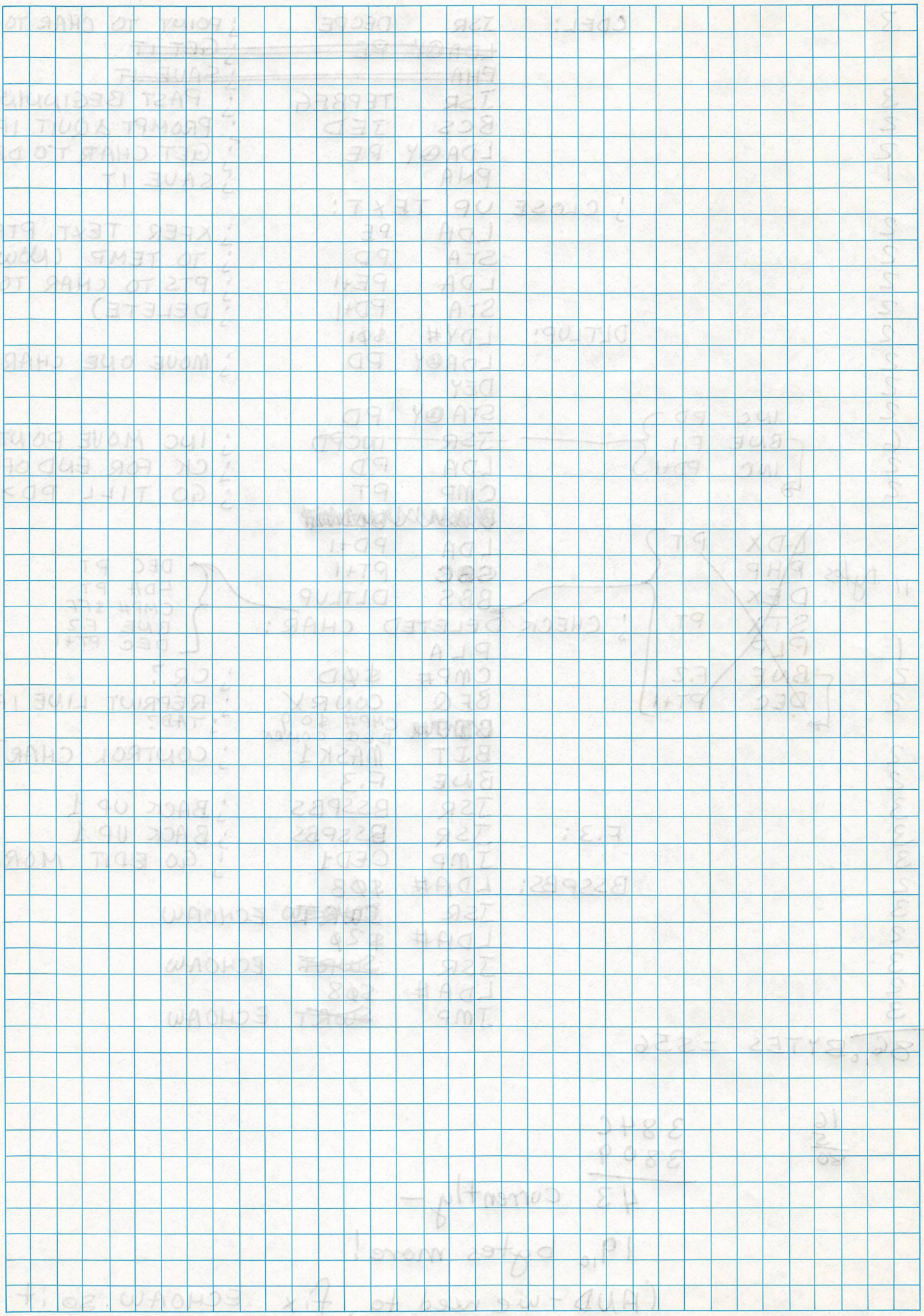
16
5
80

3840
3809

43 currently −

19₁₀ bytes more!

(AND − we need to fix ECHOAW so it does BS right!)



```
                                        ,PAGE
                                        ;ROUTINE TO DELETE THE NEXT LINE OF TEXT
3784  20 4139  DLIN:   JSR     CPTPE   ;ARE WE AT THE END OF TEXT
3787  90 06            BCC     NDEL    ;BRANCH IF NOT
3789  20 6336           JSR     ENDP    ;PROMPT END OF TEXT
378C  4C E036           JMP     CED1    ;BACK FOR CHARACTER OR COMMAND
378F  A5 06     NDEL:   LDA     PE      ;SET TWO POINTER EQUAL TO PE
3791  85 08            STA     PF      ;PF SAVES EDITORS LOCATION
3793  85 04            STA     PD      ;PD IS USED TO FIND THE BEGINING OF THE NEXT LINE
3795  A5 07            LDA     PE+1    ;THEN THE CHARACTERS ARE PASSED FROM PD TO PE
3797  85 09            STA     PF+1    ;UNTIL ALL THE TEXT IS MOVED IN OVER THE DELETED LINE
3799  85 05            STA     PD+1    ;
379B  20 FD38   NCRA:   JSR     INCPD   ;USE PD TO SEARCH FOR A CARRIAGE RETURN
379E  20 5039           JSR     TEPDPT  ;TEST FOR END OF TEXT
37A1  B0 0E             BCS     DLOUT   ;BRANCH IF END
37A3  B1 04             LDA@Y   PD      ;GET A CHARACTER
37A5  C9 0D             CMP#    $0D     ;IS IT A CR?
37A7  D0 F2             BNE     NCRA    ;BRANCH IF NOT
37A9  20 FD38           JSR     INCPD   ;MOVE PAST THE CR
37AC  20 5039           JSR     TEPDPT  ;END OF TEXT?
37AF  90 06             BCC     STO     ;BRANCH IF NOT
37E1  20 6336   DLOUT   JSR     ENDP    ;PROMPT END OF TEXT
37E4  4C F637           JMP     ADONE   ;JUMP TO CLEAN UP POINTERS
37E7  B1 04     STO:    LDA@Y   PD      ;GET CHARACTER
37E9  91 06             STA@Y   PE      ;STORE IT ACROSS THE GAP
37EB  20 FD38           JSR     INCPD   ;INCREMENT THE GAP BY INREM. PD AND PE
37EE  20 E439           JSR     INCPE   ;
37F1  20 5039           JSR     TEPDPT  ;IF PD EQUALS PT WE ARE DONE
37F4  90 F1             BCC     STO     ;BRANCH IF WE ARE NOT DONE
37F6  A5 06     ADONE:  LDA     PE      ;CLEAN UP POINTERS
37F8  85 02             STA     PT      ;MAKE PT=PE; THEN PE=PF
37FA  A5 07             LDA     PE+1    ;
37FC  85 03             STA     PT+1    ;
37FE  A5 08             LDA     PF      ;
3800  85 06             STA     PE      ;
3802  A5 09             LDA     PF+1    ;
3804  85 07             STA     PE+1    ;
3806  4C E036           JMP     CED1    ;
```



```
                                        ,PAGE
                                        ;DELETE A CHARACTER WHILE EDITING
3809  20 C839   EDEL:   JSR     DECPE   ;ENTRY FOR DELETE FROM MIDDLE OF TEXT
380C  20 3139   from CDEL  JSR  TEPBEG  ;PAST THE BEGINING?
380F  B0 A0     only    BCS     JED     ;BRANCH IF PAST   (PE<PS)
3811  A5 06             LDA     PE      ;MAKE PE=PE
3813  85 04             STA     PD      ;PD WILL BE USED TO MOVE TEXT OVER DELETED CHAR.
3815  A5 07             LDA     PE+1    ;
3817  85 05             STA     PD+1    ;
3819  A0 01     SDAG:   LDY#    $01     ;USE Y REGISTER TO PROVIDE THE OFFSET TO SHUTTLE CHARS.
381B  B1 04             LDA@Y   PD      ;GET CHAR TO BE SHIFTED
381D  88                DEY             ;SET NEW OFFSET FOR THE SHIFT
381E  91 04             STA@Y   PD      ;STORE IN NEW POSITION
3820  20 FD38           JSR     INCPD   ;ADVANCE POINTER
3823  20 5039           JSR     TEPDPT  ;TEXT FOR END OF TEXT
3826  90 F1             BCC     SDAG    ;BRANCH IF NOT.   CY CLR #EANS ≠ ONLY
3828  20 2E38   [local]JSR     DEL     ;FINISH THE PROCESS
382B  4C E036           JMP     CED1    ;GO BACK FOR NEW CHARACTER OR COMMAND
382E  A9 5C     DEL:    LDA#    '\      ;ENTRY FOR DELETE FROM END OF TEXT
3830  20 A338   6D39    JSR     ~~ECHOAW~~      ;INDICATE A DELETE; DONT COUNT TAB SPACE
3833  C6 02             DEC     PT  SWRT;MOVE TEXT POINTER BACK
3835  A5 02             LDA     PT      ;
3837  C9 FF             CMP#    $FF     ;HAS DECREMENT WRAPPED AROUND?
3839  D0 02             BNE     NDDEC   ;BRANCH IF NOT
383B  C6 03             DEC     PT+1    ;DEC HIGH BYTE
383D  60        NDDEC:  RTS             ;RETURN
383E  20 4139   CDEL:   JSR     CPTPE   ;ARE WE DELETING CENTER OR END OF TEXT
3841  90 C6             BCC     EDEL    ;BRANCH IF FROM THE CENTER
3843  20 2E38           JSR     DEL     ;JUMP IF FROM THE END
3846  20 3839           JSR     DECPE   ;PE MUST EQUAL PT AT THIS POINT
3849  4C E036           JMP     CED1    ;GO BACK FOR NEW CHARACTER OR COMMAND
```

A B C D A B C D

A B C D A B C

```
                                        ,PAGE
                                        ;ROUTINE TO FIND A CHARACTER STRING
384C  20 4139 FIND:   JSR     CPTPE     ;ARE WE AT THE END OF TEXT
384F  B0 2F           BCS     ENOUT     ;BRANCH IF WE ARE
3851  A2 00           LDX#    00        ;LOAD STRING WE ARE SEARCHING FOR
3853  20 6139 FGC:    JSR     RSY       ;GET A CHARACTER
3856  C9 0D           CMP#    $0D       ;CARRIAGE RETURN INDICATES THAT STRING IS LOADED
3858  F0 07           BEQ     STFIN     ;BRANCH IF IT IS A CARRIAGE RET.
385A  95 12           STAX    BF        ;STORE CHARACTER IN BUFFER VIA INDEX
385C  E8              INX               ;ADVANCE INDEX
385D  E0 0A           CPX#    $0A       ;ALLOW A STRING OF NO MORE THAN TEN CHARACTERS
385F  D0 F2           BNE     FGC       ;BRANCH IF STRING IS NOT LOADED
3861  A9 0D STFIN:    LDA#    $0D       ;PUT CR IN BUFFER IF STRING IS OVER TEN
3863  95 12           STAX    BF        ;STORE IN BUFFER
3865  A2 00 FIZX:     LDX#    00        ;COMPARE STRING AGAINST TEXT
3867  B1 06 NFZ:      LDA@Y   PE        ;GET CHARACTER FROM TEXT
3869  85 08           STA     PF        ;SAVE CHARACTER
386B  20 0439         JSR     INCPE     ;ADVANCE POINTER
386E  20 4139         JSR     CPTPE     ;ARE WE AT THE END OF TEXT
3871  B0 0D           BCS     ENOUT     ;BRANCH IF YES
3873  B5 12           LDAX    BF        ;GET STRING CHARACTER
3875  E8              INX               ;ADVANCE BUFFER POINTER
3876  C9 0D           CMP#    $0D       ;IF THE STRING CHAR IS A CR. WE HAVE A SUCESSFULL COMPA
3878  F0 0C           BEQ     FOUT      ;BRANCH IF SUCESSFULL
387A  C5 08           CMP     PF        ;DO CHARACTERS MATCH
387C  F0 E9 JMP PATCH BEQ     NFZ       ;IF YES THEN DON'T ZERO X REG.
387E  D0 E5           BNE     FIZX      ;NO, THEN ZERO X REG.
3880  20 6036 ENOUT:  JSR     CENDP     ;PROMPT END OF TEXT
3883  4C E036         JMP     CED1      ;GO BACK AND GET NEW CHARACTER OR COMMAND
3886  20 0B39 FOUT:   JSR     DECPE     ;PRINT TEXT FROM BEGIN OF LINE THROUGH SEARCH STRING
3889  A2 00 CONR:     LDX#    $00       ;BACK UP TO EVEN THINGS OUT
388B  20 0B39 NFCR:   JSR     DECPE     ;BACK UP TO BEGINING OF LINE
388E  E8              INX               ;COUNT HOW FAR WE BACK UP
388F  B1 06           LDA@Y   PE        ;GET A CHARACTER
3891  C9 0D           CMP#    $0D       ;IS IT A CARRIAGE RET.
3893  D0 F6           BNE     NFCR      ;BACK UP TILL WE FIND A CAR.RET.
3895  B1 06 FSTP:     LDA@Y   PE        ;PRINT THE LINE
3897  20 A338         JSR     ECHOAW    ;PRINT
389A  20 0439         JSR     INCPE     ;ADVANCE POINTER
389D  CA              DEX               ;COUNT BACK
389E  D0 F5           BNE     FSTP      ;BRANCH TILL DONE
38A0  4C E036         JMP     CED1      ;BACK FOR CHARACTER OR COMMAND
```

PATCH: ~~BEQ NOFIND RTN~~
~~JMP NFZ~~

NOFIND: DEX
BEQ
~~STX PF~~ JSR DECPE
~~LDA PE~~ DEX
~~SEC~~ BNE
RTN JMP NFZ
~~SBC PF~~
~~STA PE~~
~~LDA PE+1~~
~~SBC# 0~~
~~STA PE+1~~
~~JMP FIZX~~

```
                                ;PAGE
                                ;ECHO ROUTINES
38A3  20 AA38 ECHOAN: JSR     ECHO    ;ROUTINE ECHOS AND PRINTS
38A6  20 6D39         JSR     SWRT    ;PRINT
38A9  60              RTS     ;
                                ;THIS ROUTINE ECHOS CONTROL CHARACTERS
38AA  48        ECHO: PHA             ;SAVE ACTUAL CHARACTER
38AB  C9 10           CMP#    $10     ;DON'T ECHO OR COUNT TABS FOR CONTROL-P
38AD  F0 2F           BEQ     EPLA    ;BRANCH IF CONTROL-P
38AF  C9 0A           CMP#    $0A     ;DON'T ECHO OR COUNT TABS FOR LINE FEED
38B1  F0 2B           BEQ     EPLA    ;BRANCH IF LINE FEED
38B3  C6 10           DEC     TBB     ;COUNT DOWN TAB SPACES
38B5  F0 02           BEQ     TEC2    ;BRANCH IF ZERO
38B7  10 03     ONLY  BPL     TEC1    ;MAKE SURE
38B9  20 EF38 TEC2:   JSR     TAZR    ;START TAB AGAIN
38BC  C9 09   TEC1:   CMP#    $09     ;IS IT A TAB
38BE  F0 20           BEQ     TABR    ;BRANCH IF YES
38C0  C9 1B           CMP#    ALTC    ;DON'T ECHO ALTMODE
38C2  F0 1A           BEQ     EPLA    ;BRANCH IF ALTMODE
38C4  C9 0D           CMP#    $0D     ;IS IT A CARRIAGE RETURN
38C6  F0 13           BEQ     ENCC    ;BRANCH IF CARRIAGE RETURN
38C8  24 2D           BIT     MASK1   ;MASK TEST FOR CONTROL CHARACTER
38CA  D0 12           BNE     EPLA    ;BRANCH IF IT IS NOT A CONTROL CHARACTER
38CC  29 40           ORA#    $40     ;CONVERT IT TO AN ALPHA
38CE  48              PHA             ;SAVE THE ALPHA
38CF  A9 5E           LDA#    '^      ;PRINT AN UPARROW
38D1  20 6D39         JSR     SWRT    ;PRINT IT
38D4  68              PLA             ;GET BACK ALPHA
38D5  20 6D39         JSR     SWRT    ;PRINT THE ALPHA
38D8  4C DE38         JMP     EPLA    ;SKIP TO END
38DB  20 9337 ENCC:   JSR     CRTAB   ;;PRINT CRLF AND ZERO TAB COUNTER
38DE  68      EPLA:   PLA             ;GET ORIGINAL CHARACTER
38DF  60              RTS     ;
38E0  A9 20   TABR:   LDA#    $20     ;GET A SPACE
38E2  20 6D39         JSR     SWRT    ;PRINT IT
38E5  C6 10           DEC     TBB     ;COUNT
38E7  D0 F7           BNE     TABR    ;LOOP TILL ALL ARE PRINTED
38E9  20 EF38         JSR     TAZR    ;START TAB COUNTAGAIN
38EC  4C DE38         JMP     EPLA    ;EXIT
38EF  48      TAZR:   PHA             ;RESTARTS TAB COUNT
38F0  A5 11           LDA     TAB     ;GET TAB NUMBER
38F2  85 10           STA     TBB     ;PLACE IN TAB COUNTER
38F4  68              PLA             ;GET ACCUMULATOR
38F5  60              RTS             ;
```

CMP # $09 "BS?
BEQ SKIP
INC TBB
BNE EPLA

shouldn't this be after tabbing??

(0110 0000)

~~68 68 68~~
~~EA EA EA~~

~~38BC: C9 11~~

```
                                .PAGE
                                ;INCREMENTING AND DECREMENTING ROUTINES
38F6  E6 32   INCPT:  INC   PT      ;ROUTINE TO INCREMENT TC PT POINTER
38F8  D0 02           BNE   NINPT   ;BRANCH IF WRAPPED ARCUND
38FA  E6 33           INC   PT+1    ;HIGH BYTE
38FC  60      NINPT:  RTS
38FD  E6 34   INCPD:  INC   PD      ;ROUTINE TO INCREMENT PC POINTER
38FF  D0 02           BNE   NINPD   ;BRANCH IF WRAPPED ARCUND
3901  E6 35           INC   PD+1    ;HIGH BYTE
3903  60      NINPD:  RTS
3904  E6 36   INCPE:  INC   PE      ;INCREMENT TO INCREMENT PE POINTER
3906  D0 02           BNE   NINPE   ;BRANCH IF WRAPPED ARCUND
3908  E6 37           INC   PE+1    ;HIGH BYTE
390A  60      NINPE:  RTS
390B  C6 36   DECPE:  DEC   PE      ;ROUTINE TO DECREMENT PE
390D  A5 36           LDA   PE
390F  C9 FF           CMP#  $FF
3911  D0 02           BNE   DEDE    ;BRANCH IF WRAPPED ARCUND
3913  C6 37           DEC   PE+1    ;HIGH BYTE
3915  60      DEDE:   RTS
                                ;ROUTINES TO RESET POINTER
3916  A5 3B   ZPE:   LDA   PS      ;RESET PE,
3918  85 36           STA   PE      ;PS CONTAINS RESET VALUE
391A  A5 3C           LDA   PS+1    ;HIGH BYTES
391C  85 37           STA   PE+1    ;
391E  60              RTS
391F  A5 3B   ZPT:    LDA   PS      ;RESET PT
3921  85 32           STA   PT
3923  A5 3C           LDA   PS+1    ;HIGH BYTE
3925  85 33           STA   PT+1
3927  60              RTS
3928  A5 3B   ZPA:    LDA   PS      ;RESETS PA
392A  85 20           STA   PA
392C  A5 3C           LDA   PS+1    ;HIGH BYTE
392E  85 21           STA   PA+1
3930  60              RTS
```



```
                                .PAGE
                                ;ROUTINE TEST IF PE HAS PAST THE BEGINING OF THE TEXT
                                ;CARRY BIT INDICATES PAST
3931  A5 37   TEPBEG: LDA   PE+1    ;COMPARE POINTERS
3933  C5 3C           CMP   PS+1    ;TEST HIGH ORDER IFRST
3935  90 25           BCC   TESC    ;BRANCH IF OBVICUSLY PAST
3937  D0 26           BNE   TEO     ;BRANCH IF OBVICUSLY STILL IN THE TEXT
3939  A5 36           LDA   PE      ;TEST LOW ORDER BYTE
393B  C5 3B           CMP   PS      ;
393D  90 1D           BCC   TESC    ;BRANCH TO SET CARRY
393F  D0 1E           BNE   TEO     ;DON'T SET CARRY
                      •
                                ;ROUTINES TO COMPARE POINTERS, A SET CARRY BIT INDICATES A MATCH
3941  A5 32   CPTPE:  LDA   PT      ;ROUTINE COMPARS PT AND PE
3943  C5 36           CMP   PE      ;COMPARES LOW BTYES
3945  D0 18           BNE   TEO     ;BRANCH IF NO MATCH
3947  A5 33           LDA   PT+1    ;COMPARE HIGH BTYE
3949  C5 37           CMP   PE+1    ;
394B  D0 12           BNE   TEO     ;BRANCH IF NO MATCH
394D  38              SEC           ;INDICATE A MATCH
394E  B0 10           BCS   TOEN    ;UNCONDITIONAL BRANCH
3950  A5 32   TEPDPT: LDA   PT      ;COMPARE POINTERS PC AND PT
3952  C5 34           CMP   PD      ;LOW BYTE
3954  D0 09           BNE   TEO     ;BRANCH NO MATCH
3956  A5 33           LDA   PT+1    ;COMPARE HIGH BYTES
3958  C5 35           CMP   PD+1    ;
395A  D0 03           BNE   TEO     ;BRANCH IF NO MATCH
395C  38      TESC:   SEC           ;INDICATE MATCH
395D  B0 01           BCS   TOEN    ;UNCONDITIONAL BRANCH
395F  18      TEO:    CLC           ;INDICATE NO MATCH
3960  60      TOEN:   RTS
```

don't confuse this w/ READ at line 116!


```
                                .PAGE
                                ;I/O DEFINITIONS
        1EA0                    .DEF    OUTCH=$1EA0     ;KIM ROUTINE TO PRINT A CHARACTER
        1E5A                    .DEF    GETCH=$1E5A     ;KIM INPUT ROUTINE
                                ;ROUTINE TO READ A CHARACTER WITH ECHO, SAVING X AND Y REGISTERS
3961 84 0E      RSY:    READ:   STY     CTP
3963 20 5A 1E                   JSR     GETCH
3966 F0 FB                      BEQ     READ            ;IGNORES ALL NULL INPUT
3968 A4 0E                      LDY     CTP
396A 60                         RTS
396B 00 00                      .WORD                   ;TTY PATCH
                                ;ROUTINE PRINTS SAVING X AND Y
396D 84 0E      SWRT:           STY     CTP
396F 20 A0 1E                   JSR     OUTCH
3972 A4 0E                      LDY     CTP
3974 60                         RTS
                                ;ROUTINE PRINTS CR AND LF SAVING X,Y,A
3975 48         ACRLF:          PHA
3976 84 0E                      STY     CTP
3978 86 0F                      STX     CTP+1
397A A2 03                      LDX#    $03
397C BE D9 1F   LIMES:          LDAX    TOP+4
397F 20 A0 1E                   JSR     OUTCH
3982 CA                         DEX
3983 D0 F7                      BNE     LIMES
3985 A6 0F                      LDX     CTP+1
3987 A4 0E                      LDY     CTP
3989 68                         PLA
398A 60                         RTS
                                ;ROUTINE TO WRT CLEAN END OF TEXT TAPE
398B 20 61 39   WRTEOT: JSR     RSY             ;CHARACTER 1 FOR EOT
398E 20 61 39                   JSR     RSY             ;CHARACTER 2 FOR EOT
3991 4C 00 36                   JMP     BUF             ;RETURN TO START
3994 0D                         .BYTE $0D               ;PLACE A PERMANENT CR AT START 0
                                .DEF    TBEG=.          ;DEFINES BEGINNING OF TEXT AREA
```

17C8: C9 09 CMP# $09 TAB?
F0 03 BEQ NOOUT
20 A0 1E JSR OUTCH
60 NOOUT RTS

↑ patch to prevent echoing TAB.

JSR CRLF

ACRLF REFERRED TO AT LINES 108 AND 219
TBEG — REF. TO AT 16 & "493" ONLY

ACRLF	3975	NCF	3711	TESC	395C
ADDC	3726	NCQ	3718	TM	363A
ADONE	37F6	NCR	36F4	TOEN	3960
ALTC	0018	NCRA	37CB	TP	3633
AREAD	3669	NCS	3707	TR	3625
BACK	379A	NDDEC	383D	TW	3618
BF	0012	NDE	3742	TZ	3641
BUF	3600	NDEL	378F	WECH	369C
CDEL	383E	NESC	371F	WPS	368E
CED1	36E0	NET	3730	WRITE	360C
CENDP	3660	NFCR	388B	WTC	36A2
CLEAR	3684	NFZ	3867	ZPA	3928
CONR	3889	NINPD	3903	ZPE	3916
CPTPE	3941	NINPE	390A	ZPT	391F
CRLF	3987	NINPT	38FC		
CRTAB	3793	NIPA	3673		
CTP	000E	NWCR	36AB		
DECPE	390B	NWTJ	36AE		
DEDE	3915	OUTCH	1EA0		
DEL	382E	PA	0000		
DLIN	37B4	PD	0004		
DLOUT	37E1	PE	0006		
ECHO	38AA	PF	0008		
ECHOAW	38A3	PNC	3696		
EDEL	3809	PRINT	3686		
EDIT	36D8	PS	000B		
EDPR	36DB	PT	0002		
ENCC	38DB	READ	36B9		
ENDP	3663	RNES	36D1		
ENOUT	3880	RRDT	36BF		
ENT	3763	RSY	3961		
EOT	378D	SAC	3738		
EPLA	38DE	SDAG	3619		
FGC	3853	SDEC	379D		
FIND	384C	SLINE1	36F6		
FIZX	3865	STAB	0009		
FLAG	000A	STAE	3655		
FOUT	3886	STFIN	3861		
FSTP	3895	STO	37E7		
GETCH	1E5A	SWRT	396D		
INCPD	38FD	TAB	0011		
INCPE	3904	TABR	38E0		
INCPT	38F6	TAZR	38EF		
JED	3781	TBB	0010		
JOT	37AB	TBEG	3998		
LOKY	3647	TC	362C		
LEOL	3785	TE	3614		
LINE	3771	TEC1	38BC		
MASK1	000D	TEC2	38B9		
MNET	3798	TED	395F		
MON	3600	TEPBEG	3931		
MSTAR	3600	TEPOPT	3950		

397C LDMES
3963 READ
1FD5 TØP
398B WRTEØT

READ ALSO 3963!

ACRLF 108 219 474
ADDC 162 166
ADONE 256 263
ALTC 9 123 158 362
AREAD 75
BACK 150 224
BF 21 311 316 323
BUF 23 28 56 114 125 160
CDEL 163 298
CED1 67 134 195 203 217 232 239 271 289 302 331 344
CENDP 69 330
CLEAR 82 85 89
CONR 141 333
CPTPE 167 235 214 236 298 305 321 437
CRLF 477 485
CRTAB 32 55 69 140 153 213 219 375
CTP 18 460 461 463 464 467 468 470 471 475 476 478 479
DECPE 224 225 274 301 332 334 403
DEDE 436 438
DEL 288 290 300
DLIN 157 236
DLOUT 248 255
ECHO 135 347 351
ECHOAW 32 71 104 133 211 291 340 347
EDEL 274 299
EDIT 38 131 145 233
EDPR 132
ENCC 365 375
ENDP 62 70 202 216 238 255
ENOUT 306 322 332
ENT 48 147 198
EOT 216 216
EPLA 353 355 363 367 374 376 383
FGC 308 314
FIND 154 325
FIZX 317 329
FLAG 15 94 96 102
FOUT 326 332
FSTP 339 343
GETCH 458 462
INCPD 111 246 252 259 285 395
INCPE 66 194 208 231 260 320 341 399
INCPT 65 127 190 391
JED 227 233 276
JOT 231
LOKY 35 59
LEOL 210 213
LINE 143 235 212
MASK1 17 366
MNET 215 217
MON 23 54
MSTAR 6 26
NCF 152 155

NCQ	156	158												
NCR	139	142												
NCRA	246	251												
NCS	149	151												
NDDEC	295	297												
NDE	178	180												
NDEL	237	240												
NESC	159	161												
NET	166	171												
NFCR	334	338												
NFZ	318	328												
NINPD	396	398												
NINPE	400	402												
NINPT	392	394												
NIPA	78	80												
NWCR	107	110												
NWTJ	105	109	111											
OUTCH	457	469	486	488	490									
PA	10	76	77	79	81	84	421	423						
PD	12	98	100	101	172	174	175	176	179	181	183	184	187	242
	245	249	257	278	280	282	284	395	397	446	449			
PE	13	185	188	193	199	201	207	228	240	243	258	263	265	268
	270	277	279	318	336	339	399	401	403	404	407	411	413	428
	432	438	441											
PF	14	241	244	267	269	319	327							
PNC	101	113												
PRINT	51	93												
PS	16	97	99	410	412	415	417	420	422	429	433			
PT	11	64	80	83	126	171	173	198	200	264	266	292	293	296
	391	393	416	418	437	440	445	448						
READ	45	116												
RNES	124	126												
RRDT	118	120	122	128										
RSY	31	41	118	134	308	460								
SAC	175	186	189											
SDAG	281	287												
SDEC	225	230												
SLINE1	137	143												
STAB	19	20	22											
STAE	64	170												
STFIN	310	315												
STO	254	257	262											
SWRT	110	348	371	373	379	467								
TAB	20	385												
TABR	361	378	381											
TAZR	220	359	382	384										
TBB	19	356	380	386										
TBEG	16	493												
TC	44	46												
TE	34	36												
TEC1	358	360												
TEC2	357	359												
TEO	431	435	439	442	447	450	453							

Another READ 461/462! 2x lines

TEPBEG	226	275	428			
TEPDPT	112	247	253	261	286	445
TESC	430	434	451			
TH	50	52				
TOEN	444	452	454			
TP	47	49				
TR	40	43				
TW	37	39				
TZ	53	55				
WECH	124					
WPS	95	97				
WRITE	42	96				
WTO	103	106				
ZPA	59	87	116	420		
ZPE	61	131	410			
ZPT	60	117	415			

Required RAM Loc's for ASM65

2FBD (512)

2FBC (533) (~~683~~) (922)

2FBB (534) (686) (919)

Note: Instr's call routines thru ~~their SYM TAB def's~~ DSPCH TBL @ 2BE3

Beware line 612 } --
635
1018 }
1023 } @SYM ADR PTR
1059
1056
1088
1101 }
1258 — @TXTADR PTR

Routine converts Binary to KIM format

1693 - STAX ADBUFF ($28C3)
1701 - STAX DATBUF ($28C5)

1738 - INC ADBUFF

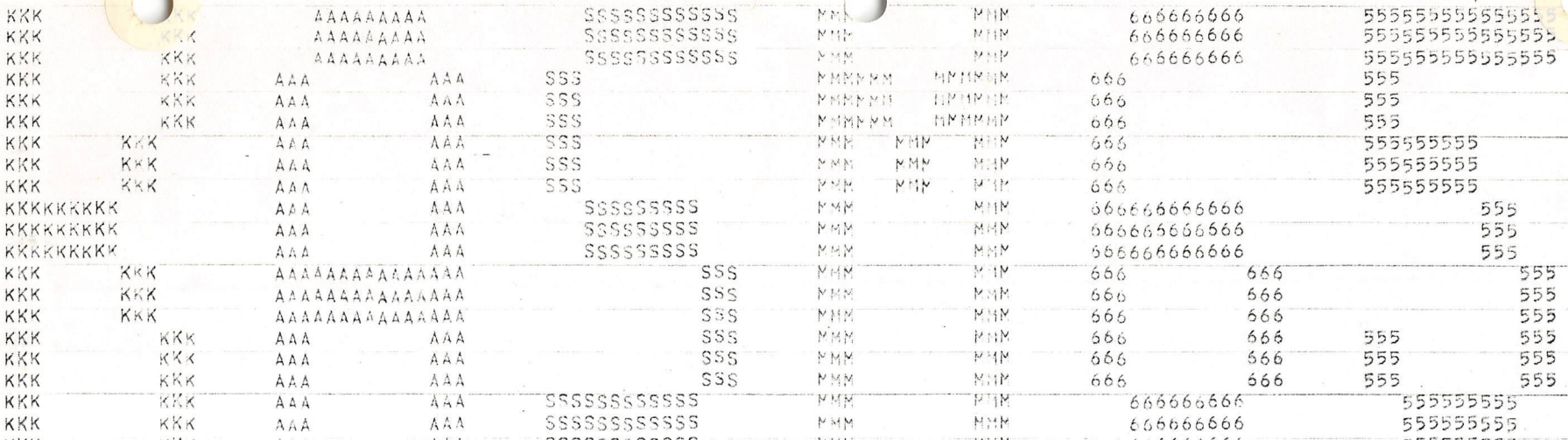
1736 - INC ADBUFF+1
1820 STA ADBUFF
1821 STA ADBUFF+1

MUST RAM! 28C3/C4/C5 ... /DØ
(or more if longer binary is desired)

ALSO: 2C97 - "COUNT" (LINES 2884 THRU 2916 HAVE REF'S)

Should not need this - - -

Believe these are the only ones —



LPTSPL VERSION 6(344) RUNNING ON LPT000
START USER 6502 GROUP [7066,322] JOB TFOCAL SEQ. 4612 DATE 13-OCT-77 22:06:21 MONITOR CSM DECSYSTEM-10 603V1 *START*
REQUEST CREATED: 13-OCT-77 21:07:50
FILE: DSKB1:KASM65.PRI[7066,322,MARSH] CREATED: 13-OCT-77 15:49:00 <177> PRINTED: 13-OCT-77 22:23:15
QUEUE SWITCHES: /PRINT:ARROW /FILE:ASCII /COPIES:1 /SPACING:1 /LIMIT:950 /FORMS:MINE

```
                ;
                ;       *** ASM65 - RESIDENT ASSEMBLER FOR THE MOS 650X ***
                ;
                ;               AUTHOR: WAYNE WALL
                ;                       CSM COMPUTING CENTER
                ;                       GOLDEN, COLO.  80401
                ;
                ;                       (303)-279-0300
                ;
                ;       COPYRIGHT:      WAYNE WALL, CSM COMPUTING CENTER, 1977
                ;
                ;
                ;          THIS PROGRAM IS A RESIDENT ASSEMBLER FOR THE MOS-650X
                ;       FAMILY OF MICROPROCESSORS.  IT IS DESIGNED TO BE A SIMPLISTIC,
                ;       BUT USEFUL, ASSEMBLER.  IT IS A TWO PASS ASSEMBLER WHICH WILL
                ;       OUTPUT THE BINARY AND LISTING DURING THE SECOND PASS.  THE
                ;       BINARY PRODUCED IS NORMALLY LOADED USING THE 'LINK-65' LOADER.
                ;       THE ASSEMBLER IS WRITTEN IN A DEVICE INDEPENDENT FASHION AND
                ;       IS LINKED TO THE APPROPRIATE DEVICE DRIVERS THROUGH DISPATCH
                ;       TABLES.
                ;
                ;          THIS PROGRAM IS WRITTEN IN THE "ASM65" ASSEMBLY LANGUAGE
                ;       AND, THEREFORE, MAY BE ASSEMBLED WITH THE "ASM65" RESIDENT
                ;       ASSEMBLER (I.E. THE ASSEMBLER WILL ASSEMBLE ITSELF).
                ;
```

```
                        .PAGE                   ;'SYMBOLIC DEFINITIONS NEEDED BY THIS PROGRAM'

                        ;DEFINE FIRST ADDR USED ON PAGE ZERO.

0020                    .DEF    PG0BEG=$0020

                        ;DEFINE STARTING LOCATION IN MEMORY.

2000                    .DEF    START=$2000     ;STARTING ADDR

                        ;DEFINE SOME SPECIAL CHARACTERS USED BY THE ASSEMBLER

001A                    .DEF    EOFCHR=$1A      ;END-OF-FILE CHARACTER (CTRL-Z)
000D                    .DEF    EOLCHR=$0D      ;END-OF-LINE CHARACTER (CARRIAGE RETURN)
000D                    .DEF    CRCHR=$0D       ;CARRIAGE RETURN
000A                    .DEF    LFCHR=$0A       ;LINE FEED
000C                    .DEF    FFCHR=$0C       ;'FORM-FEED' CHARACTER (CTRL-L)
0009                    .DEF    TABCHR=$09      ;'TABULATION' CHARACTER (TAB)
007F                    .DEF    DELCHR=$7F      ;'DELETE' BACK ONE CHAR (RUBOUT)

                        ;DEFINITIONS FOR INPUT LINE BUFFER AREA

0100                    .DEF    LINBUF=$0100    ;ADDR OF THE LINE BUFFER
007F                    .DEF    LINEL=$7F       ;MAX SIZE OF LINE BUFFER (127 CHARACTERS)

                        ;GLOBAL STATUS FLAGS

007F                    .DEF    STSP1=$7F       ;'AND' MASK TO SET BIT 7 = 0 = PASS1
0080                    .DEF    STSP2=$80       ;'OR'  MASK TO SET BIT 7 = 1 = PASS2
                                                ;*** THE 'PASS' FLAG MUST BE BIT 7! ***

0040                    .DEF    STSEOS=$40      ;BIT 6 = 1 TO INDICATE '.END' SEEN
00BF                    .DEF    STCEOS=$BF      ;COMPLEMENT OF 'STSEOS'

0020                    .DEF    STSDBS=$20      ;BIT 5 = 1 TO DUMP BAD SYMBOLS
00DF                    .DEF    STCDBS=$DF      ;COMPLEMENT OF 'STSDBS'

0100                    .DEF    STACK=$0100     ;DEFINES BASE ADDR OF HARDWARE STACK

0035                    .DEF    PAGMAX=$35      ;DEFINES MAX PAGE THE ASSEMBLER WILL
                                                ;WRITE SYMBOL TABLE INTO.  IT WILL
                                                ;WRITE IN THIS PAGE!
```

```
                        .PAGE                   ;'FORMAT OF A SYMBOL TABLE ENTRY'

                        ;       *** FORMAT OF A SYMBOL TABLE ENTRY ***
                        ;
                        ;       ØNNNNNNN        ;*** THE CHARACTERS COMPOSING SYMBOL'S NAME.
                        ;       ØNNNNNNN        ;THEY ALL HAVE BIT 7=Ø AND ARE ASCII.
                        ;          .
                        ;          .            ;THERE ARE AN ARBITRARY NUMBER OF THEM
                        ;          .            ;CURRENT MAXIMUM OF 10 CHARS IN NAME.
                        ;          .
                        ;       1SOLUMXX        ;*** THE STATUS BYTE
                        ;                       ;       BIT 7 = 1 ALWAYS TO FLAG END OF NAME
                        ;                       ;'S'= BIT 6 = Ø IF THE VALUE NEEDS 2 BYTES
                        ;                       ;             1 IF THE VALUE NEEDS 1 BYTE
                        ;                       ;'O'= BIT 5 = 1 IF THE SYMBOL REPRESENTS AN OPCO
                        ;                       ;'L'= BIT 4 = 1 IF THE SYMBOL REPRESENTS A LABEL
                        ;                       ;'U'= BIT 3 = 1 IF THE SYMBOL'S VALUE IS UNDEFIN
                        ;                       ;'M'= BIT 2 = 1 IF THE SYMBOL'S VALUE IS
                        ;                       ;               MULTIPLY-DEFINED.
                        ;                       ;'X'= BIT 1 = Ø CURRENTLY UNUSED
                        ;                       ;'X'= BIT Ø = Ø CURRENTLY UNUSED
                        ;                       ;-----------------------------------------
                        ;                       ;NOTE:  IF 'O' = 1 (AN OPCODE), THEN THE
                        ;                       ;       STATUS BITS HAVE A DIFFERENT MEANING,
                        ;                       ;       THE STATUS BYTE THEN BECOMES:
                        ;                       ;
                        ;                       ;     '1SCXXCCC'
                        ;                       ;
                        ;                       ;WHERE:
                        ;                       ;
                        ;                       ;'1','S','C' ARE THE SAME AS ABOVE, BUT
                        ;                       ;'X'= BIT 4 = Ø CURRENTLY UNUSED
                        ;                       ;'X'= BIT 3 = Ø CURRENTLY UNUSED
                        ;                       ;'CCC'= BITS 2,1,Ø = THE CLASS CODE
                        ;                       ;-----------------------------------------
                        ;
                        ;       VVVVVVVV        ;*** SYMBOL'S VALUE STORED AS 1 BYTE OR
                        ;       VVVVVVVV        ;2 BYTES AS INDICATED BY THE 'S' BIT IN
                        ;                       ;THE STATUS BYTE (SEE ABOVE).
                        ;                       ;IF 2 BYTES ARE USED, THEN THE FIRST ONE
                        ;                       ;IS THE HIGH ORDER 8 BITS AND THE SECOND
                        ;                       ;IS THE LOW ORDER 8 BITS.
```

```
                        .PAGE                   ;'SYMBOL TABLE ENTRY DEFINITIONS'

        0080            .DEF    SYMFLG=$80      ;BIT 7 - FLAGS THE END OF SYMBOL NAME
                                                ;        *** MUST ALWAYS BE BIT 7 !! ***
        0040            .DEF    SYMSIZ=$40      ;BIT 6 - SIZE INDICATOR FOR SYMBOL'S VALUE
        0020            .DEF    SYMOPC=$20      ;BIT 5 - INDICATES SYMBOL IS AN OPCODE
        0010            .DEF    SYMLAB=$10      ;BIT 4 - INDICATES SYMBOL IS A LABEL
        0008            .DEF    SYMUDF=$08      ;BIT 3 - INDICATES SYMBOL IS UNDEFINED
        00F7            .DEF    SYCUDF=$F7      ;THE COMPLEMENT OF 'SYMUDF'.
        0004            .DEF    SYMMDF=$04      ;BIT 2 - INDICATES SYMBOL IS MULTIPLY-DEFINED
                        ;IF THE SYMBOL IS AN OPCODE (BIT 5=1), THEN SOME OTHER
                        ;BITS HAVE DIFFERENT MEANINGS.
        0007            .DEF    SYMCLS=$07      ;BIT 2,1,0 - CONTAIN THE OPCODE CLASS
        00FF            .DEF    SYMEOT=$FF      ;A STATUS BYTE OF ALL ONES FLAGS THE
                                                ;END OF THE SYMBOL TABLE.
```

```
                        .PAGE                   ;'DEFINITION OF OPCODE CLASSES'


                        ;ALL OPCODES THAT BELONG TO A GIVEN CLASS HAVE
                        ;PROPERTIES SIMILAR ENOUGH THAT ONE ROUTINE IN THE ASSEMBLER
                        ;CAN HANDLE THE ASSEMBLING OF THE INSTUCTION.  CURRENTLY THERE
                        ;IS SPACE FOR 8 POSSIBLE CLASSES OF OPCODES.  THE CURRENTLY
                        ;RECOGNIZED CLASSES ARE LISTED BELOW.

0000                    .DEF    CL0=0           ;*** CLASS 0 ***
                                                ;THIS CLASS CONTAINS ALL OF THE 'IMMEDIATE'
                                                ;MODE INSTRUCTIONS.  THE LOW ORDER BYTE OF
                                                ;THE OPERAND IS TAKEN WITHOUT REGARD TO WHAT
                                                ;IS STORED IN THE HIGH ORDER BYTE.

0001                    .DEF    CL1=1           ;*** CLASS 1 ***
                                                ;THIS CLASS CONTAINS ALL INSTRUCTIONS THAT
                                                ;HAVE DIFFERENT OPCODES IF THE MEMORY REFERENCE
                                                ;IS LOCATED ON PAGE ZERO AS OPPOSED TO SOME
                                                ;OTHER PAGE.  THE ADDRESS IS ONE BYTE IF
                                                ;PAGE ZERO ADDRESSING IS USED, TWO BYTES
                                                ;IF ABSOLUTE ADDRESSING IS USED.

0002                    .DEF    CL2=2           ;*** CLASS 2 ***
                                                ;THIS CLASS CONTAINS ALL INSTRUCTIONS WHICH
                                                ;USE ONLY ABSOLUTE ADDRESSING (2-BYTE ADDR).

0003                    .DEF    CL3=3           ;*** CLASS 3 ***
                                                ;THIS CLASS CONTAINS ALL INSTRUCTIONS IN
                                                ;WHICH IT IS MANDATORY THAT A PAGE ZERO
                                                ;REFERENCE BE USED.

0004                    .DEF    CL4=4           ;*** CLASS 4 ***
                                                ;THIS CLASS CONTAINS ALL INSTRUCTIONS WHICH
                                                ;USE RELATIVE ADDRESSING ONLY.

0005                    .DEF    CL5=5           ;*** CLASS 5 ***
                                                ;THIS CLASS CONTAINS ALL INSTUCTIONS WHICH
                                                ;USE IMPLIED ADDRESSING.  THEY ARE ALL
                                                ;ONE BYTE INSTRUCTIONS.

0006                    .DEF    CL6=6           ;*** CLASS 6 ***
                                                ;THIS CLASS CONTAINS ALL PSUEDO-OPCODES
                                                ;THAT DO SPECIAL FUNCTIONS AT ASSEMBLY
                                                ;TIME.  THE VALUE OF THE SYMBOL IS THE
                                                ;ADDRESS OF THE ROUTINE THAT PERFORMS THE
                                                ;FUNCTION FOR THE ASSEMBLER.

0007                    .DEF    CL7=7           ;*** CLASS 7 ***
                                                ;CURRENTLY UNUSED.
```

```
                        .PAGE                   ;'MORE SYMECL TABLE DEFINITIONS'


                        ;DEFINE STATUS BYTES FOR EACH CLASS OF OPCODE.

        00E0            .DEF    OPCCL0=SYMFLG+SYMSIZ+SYMOPC+CL0 ;ONE-BYTE OPCODE VALUE
        00A1            .DEF    OPCCL1=SYMFLG+SYMOPC+CL1        ;TWO-BYTE VALUE TO CONTAIN
                                                                ;BOTH POSSIBLE OPCODES.
        00E2            .DEF    OPCCL2=SYMFLG+SYMSIZ+SYMOPC+CL2 ;ONE BYTE OPCODE VALUE.
        00E3            .DEF    OPCCL3=SYMFLG+SYMSIZ+SYMOPC+CL3 ;ONE BYTE OPCODE VALUE.
        00E4            .DEF    OPCCL4=SYMFLG+SYMSIZ+SYMOPC+CL4 ;ONE BYTE OPCODE VALUE.
        00E5            .DEF    OPCCL5=SYMFLG+SYMSIZ+SYMOPC+CL5 ;ONE BYTE OPCODE VALUE.
        00A6            .DEF    OPCCL6=SYMFLG+SYMOPC+CL6        ;TWO BYTE ADDR OF ROUTINE.

        0080            .DEF    USYM=SYMFLG     ;DEFINE SKELETON STATUS FOR USER DEFINED
                                                ;SYMBOL.

                        ;BIT DEFINITIONS FOR ERROR TYPES

        0080            .DEF    LERR=$80        ;BIT 7 = 1 INDICATES 'LABEL ERROR'
        0040            .DEF    MERR=$40        ;BIT 6 = 1 INDICATES 'MULTIPLY-DEFINED' ERROR
        0020            .DEF    UERR=$20        ;BIT 5 = 1 INDICATES 'UNDEFINED' ERROR
        0010            .DEF    IERR=$10        ;BIT 4 = 1 INDICATES 'INTERNAL' ERROR
        0008            .DEF    PERR=$08        ;BIT 3 = 1 INDICATES 'PHASE' ERROR
        0004            .DEF    OERR=$04        ;BIT 2 = 1 INDICATES 'OPCODE' ERROR
        0002            .DEF    AERR=$02        ;BIT 1 = 1 INDICATES 'ADDRESSING' ERROR
        0001            .DEF    SERR=$01        ;BIT 0 = 1 INDICATES 'SYNTAX' ERROR
```

```
                                .PAGE                   ;'PAGE ZERO STUFF'

                        0020    .LOC    PG0BEG

                                ;LOCATIONS TO HOLD THE DEVICE NUMBERS OF ACTIVE I-O DEVICES

0020    00      SRCDEV: .BYTE   0       ;HOLDS DEVICE NUMBER WHERE ASCII SOURCE IS INPUT
0021    00      LSTDEV: .BYTE   0       ;HOLDS DEVICE NUMBER WHERE ASCII LISTING IS PROD
0022    00      BINDEV: .BYTE   0       ;HOLDS DEVICE NUMBER TO OUTPUT BINARY ONTO
0023    00      TLDEV:  .BYTE   0       ;HOLDS DEVICE NUMBER CURRENTLY LISTING TO

0024    00      INPSW:  .BYTE   0       ;INPUT FLAG, IF = 0, THEN INPUT FROM DEVICE
                                        ;IF NOT, THEN INPUT FROM LINE BUFFER

0025    06      MAXDEV: .BYTE   DEVMAX  ;THIS LOC HOLDS THE LARGEST ALLOWED DEVICE
                                        ;NUMBER + 1.
0026    00      DEVI:   .BYTE   0       ;HOLDS CURRENTLY ACTIVE INPUT DEVICE NUMBER
0027    00      DEVO:   .BYTE   0       ;HOLDS CURRENTLY ACTIVE OUTPUT DEVICE NUMBER

0028    00      ERRDEV: .BYTE   0       ;=$FF IF NO I-O ERRORS, OTHERWISE CONTAINS
                                        ;THE DEVICE NUMBER OF DEVICE THAT GOT THE ERROR

0029    35      MAXPAG: .BYTE   PAGMAX  ;HOLDS 'PAGMAX', MAY BE PATCHED FOR
                                        ;SYSTEMS WITH MORE (OR LESS) MEMORY.

002A    86 70   MONADR:  .WORD  $7086   ;ADDRESS TO ENTER MONITOR, CHANGE FOR KIM.
                                        ;THIS IS THE ADDRESS FOR 'TIM' DEBUG MONITOR
                                        ;THIS IF YOUR ROUTINE IS ELSEWHERE.

002C    00      NUMBH:  .BYTE   0       ;HIGH ORDER BYTE OF NUMBER INPUT
002D    00      NUMBL:  .BYTE   0       ;LOW ORDER BYTE OF NUMBER INPUT

                002E            .DEF    TCNT=.          ;DOUBLES AS A TEMPORARY COUNTER.
002E    00      NTEMPH: .BYTE   0       ;TEMPORARY USED FOR NUMERICAL OPERATIONS
002F    00      NTEMPL: .BYTE   0       ;TEMPORARY USED FOR NUMERICAL OPERATIONS

0030    00 00   STRADR: .WORD   0       ;HOLDS ADDR OF OUTPUT STRING FOR THE
                                        ;'OUTSTR' SUBROUTINE.

0032    00      CHAR:   .BYTE   0       ;HOLDS CURRENT INPUT CHARACTER

0033    4C      JSRIND: .BYTE   $4C     ;OPCODE FOR 'JMP'.  THIS MUST PRECEED 'JTEMP'
0034    00 00   JTEMP:  .WORD   0       ;HOLDS TEMP ADDR FORMED FOR DISPATCH VIA
                                        ;'JSR JSRIND'.

0036    00 00   TXTADR: .WORD   0       ;HOLDS BASE ADDR OF TEXT BUFFER
0038    00      TEXTP:  .BYTE   0       ;HOLDS OFFSET POINTER INTO BUFFER

0039    00 01   LINADR: .WORD   LINBUF  ;HOLDS BASE ADDR OF THE LINE BUFFER

0038    00      FLAGS:  .BYTE   0       ;GLOBAL STATUS FLAGS
```

```
                                .PAGE                   ;'PAGE ZERO STUFF'

003C  00      NBYTES: .BYTE   0                 ;HOLDS NUMBER OF BYTES GENERATED BY CURRENT LIN
003D  00      BYTE1:  .BYTE   0                 ;THESE 3 LOCS HOLD VALUE GENERATED
003E  00      BYTE2:  .BYTE   0
003F  00      BYTE3:  .BYTE   0                 ;NOTE:  *** ALL 3 MUST BE SEQUENTIAL ***

0040  00      ERRFLG: .BYTE   0                 ;HOLDS ERROR TYPE INDENTIFIERS
0041  00 00   ERRCNT: .WORD   0                 ;HOLDS COUNT OF ERRORS
                                                ;DURING EACH PASS
0043  00      LWIDTH: .BYTE   0                 ;USER SUPPLIED WIDTH OF LISTING DEVICE
0044  00      HPOS:   .BYTE   0                 ;COUNTER USED BY LISTING ROUTINE
0045  00      LTEMP:  .BYTE   0                 ;TEMPORARY USED BY LISTING ROUTINE

0046  00      OPCVL:  .BYTE   0                 ;HOLDS SYMBOL VALUE EXTRACTED FROM
0047  00      OPCVH:  .BYTE   0                 ;OPCODE TABLE.

0048  00 00   SYMADR: .WORD   0                 ;POINTS TO SYMBOL TABLE DURING
                                                ;SYMBOL TABLE SEARCHES.

                                ;SOME BIT MASKS USED IN 'BIT' INSTRUCTIONS

004A  20      BMOPC:  .BYTE   SYMOPC            ;TESTS FOR OPCODE BIT IN SYMBOL STATUS
004B  40      BMSIZ:  .BYTE   SYMSIZ            ;TESTS FOR SIZE BIT IN SYMBOL STATUS
004C  10      BMLAB:  .BYTE   SYMLAB            ;TESTS FOR 'LABEL' BIT IN SYMBOL STATUS
004D  08      BMUDF:  .BYTE   SYMUDF            ;TESTS FOR 'UNDEFINED' BIT IN SYMBOL STATUS
004E  04      BMMDF:  .BYTE   SYMMDF            ;TESTS FOR 'MULTIPLY-DEFINED' BIT

                                ;SCRATCH AREA TO HOLD TEMPORARY SYMBOL NAME

004F  20      SYMTM:  .ASCII  ' '               ;TO LIST SYMBOL NAME

0050  00      SYMTMP: .BYTE   0
0051  00              .BYTE   0
0052  00              .BYTE   0
0053  00              .BYTE   0
0054  00              .BYTE   0
0055  00              .BYTE   0
0056  00              .BYTE   0
0057  00              .BYTE   0
0058  00              .BYTE   0
0059  00              .BYTE   0
005A  00              .BYTE   0

      0009      .DEF    SYMNMX=.-SYMTMP-2       ;DEFINES MAXIMUM NUMBER OF BYTES ALLOWE
                                                ;IN A SYMBOL NAME.
```

```
                        .PAGE                   ;'ADDRESSES OF OUTPUT MESSAGE STRINGS'

                        ;THESE ARE ON PAGE ZERO TO SAVE BYTES, THEY COULD BE
                        ;MOVED, BUT SUBROUTINE 'OUTSTR' WOULD THEN HAVE TO WORK
                        ;DIFFERENTLY.

025B  17 2A   MIOERR: .WORD   IOERRM
025D  5A 2A   MSRCN:  .WORD   SRCNM
025F  75 2A   MBINN:  .WORD   BINNM
0261  91 2A   MLSTN:  .WORD   LSTNM
0263  AE 2A   MPAGE:  .WORD   PAGEM
0265  AF 2A   MTITLE: .WORD   TITLEM
0267  D5 2A   MWIDTH: .WORD   WIDTHM
0269  B7 28   MERROR: .WORD   ERRORM
026B  6E 28   MREADY: .WORD   READYM
026D  F9 2A   MEOF:   .WORD   EOFM
026F  42 2A   MPASS:  .WORD   PASSM
0271  9B 28   MOVFL:  .WORD   OVFLM
0273  26 2A   MFLSYM: .WORD   FLSYMM
0275  4F 00   MSYMT:  .WORD   SYMTM
0277  33 28   MESGER: .WORD   GERMES
0279  57 28   MESNER: .WORD   NERMES

                        ;USED BY BINARY OUTPUT ROUTINES

027B  00      BFLAG:  .BYTE                   ;FLAG FOR ADDRESS VS DATA
027C  00      ADDCNT: .BYTE                   ;HOLDS COUNT OF ADDRESS BYTES LOADED
027D  00      DATCNT: .BYTE                   ;HOLDS COUNT OF DATA BYTES
027E  00 00   CHKBUF: .WORD                   ;HOLD CHECKSUM FORMING
0280  00      BITEMP: .BYTE                   ;TEMPORARY


0081                  .DEF    PGZERO=.        ;LET OTHER'S KNOW WHERE OUR PAGE ZERO
                                              ;USAGE ENDS
                              .LOC  $0081     ;FIX FOR ASM65 BRANCH
                                              ;CALCULATION CHECK ERROR
0081  20 23 29  BFIX:   JSR   ISUB            ;CALCULATE START OF
                                              ;BRANCH ADDRESS.
0084  24 2C             BIT   NUMEH           ;LOOK AT MSB OF DOUBLE
0086  30 07             BMI   BACK            ;PRECISION SUBTRACTION
0088  24 2D   FORWRD:  BIT   NUMEL            ;LOOK AT MSB OF BRANCH
008A  30 07             BMI   BERR            ;OFFSET
008C  4C 3A 23  BOK:    JMP   BASE+$033A      ;OFFSET OK
008F  24 2D     BACK:   BIT   NUMEL           ;CHECK BRANCH OFFSET
0091  30 F9             BMI   BOK
0093  4C 18 23  BERR:   JMP   CAERR           ;FAILED OFFSET TEST
```

See 25C4 for 12 patch ---

FOR HALT ON PASS ~~Ø, 1~~ ≠ 1 or 2 :

2035:	A5	2D		LDA	NUMBL
2037:	C9	Ø2		CMP#	Ø2
2039:	FØ	11		BEQ	DOP2 (204C)
203B:	C9	01		CMP#	Ø1
203D:	FØ	Ø4		BEØ	DOP1 (2043)
203F:	6C	2A	ØØ	~~JMP@~~	~~MONADR~~
	~~[illegible]~~	~~[illegible]~~		~~JSR~~	~~SAV1~~
				~~NOP~~	

```
                        .PAGE                   ;'ASMINT - MAIN BODY OF ASSEMBLER'

                        ;THE MAIN BODY OF THE PROGRAMMING CODE FOR THE ASSEMBLER
                        ;EXISTS IN THIS MODULE.  THIS MODULE IS DEVICE INDEPENDENT AND
                        ;DOES THE SOURCE CODE INTERPRETATION, BINARY OUTPUT AND LISTING
                        ;OUTPUT GENERATION.


        2000            .LOC    START           ;WHERE TO START ASSEMBLING THE ASSEMBLER!

                        ;INITIALIZATION AND STARTUP


2000    78      ASM65:  SEI                     ;BE VERY CAUTIOUS, DISABLE ALL IRQ'S
2001    D8              CLD                     ;CLEAR DECIMAL MODE
2002    A2 FF           LDX#    $FF             ;INIT STACK POINTER
2004    9A              TXS
2005    86 28           STX     ERRDEV          ;INDICATE NO I-O ERRORS YET
2007    E8              INX                     ;FORM A ZERO
2008    86 23           STX     SRCDEV          ;CLEAR OUT SOURCE DEVICE NUMBER
200A    86 22           STX     BINDEV          ;AND BINARY DEVICE NUMBER
200C    86 21           STX     LSTDEV          ;AND LISTING DEVICE NUMBER
200E    86 24           STX     INPSW           ;INDICATE INPUT TO COME FROM BUFFER
2010    86 38           STX     FLAGS           ;CLEAR OUT GLOBAL FLAGS
2012    20 8C25         JSR     TXTINI          ;INIT POINTERS TO TEXT BUFFER
2015    A2 00           LDX#    CONODV          ;SET CURRENT OUTPUT DEVICE TO CONSOLE
2017    86 27           STX     DEVO
2019    20 C729         JSR     INODEV          ;AND 'INITIALIZE' IT
201C    A2 00           LDX#    CONIDV          ;SET CURRENT INPUT DEVICE TO CONSOLE INPUT
201E    86 26           STX     DEVI
2020    20 B229         JSR     INIDEV          ;AND 'INITIALIZE' IT
                        ;CONTINUED ON NEXT PAGE.
```



```
                        .PAGE                   ;'ASMINT - MAIN BODY OF ASSEMBLER'


2023    A2 65           LDX#    MTITLE          ;OUTPUT TITLE FOR IDENTIFICATION
2025    20 EF29         JSR     OUTSTR          ;OF VERSION.
2028    A9 00   ASKPAS: LDA#    CONODV          ;SET CONSOLE AS OUTPUT
202A    85 27           STA     DEVO
202C    A9 00           LDA#    CONIDV          ;AND INPUT DEVICE
202E    85 26           STA     DEVI
2030    A2 6F           LDX#    MPASS           ;ASK USER WHICH PASS HE WANTS TO DO
2032    20 AD20         JSR     ASKDEC          ;AND ACCEPT A NUMBER (1-2).
2035    A5 2C           LDA     NUMBH           ;GET HIGH ORDER
2037    D0 EF           BNE     ASKPAS          ;BRANCH IF OUT OF RANGE
2039    A5 2D           LDA     NUMBL
203B    C9 02           CMP#    2               ;WHICH PASS?
203D    F0 0D           BEQ     DOP2            ;BRANCH IF PASS 2
203F    C9 01           CMP#    1
2041    D0 E5           BNE     ASKPAS          ;BRANCH IF BAD INPUT
2043    20 C020         JSR     PASS1           ;DO PASS 1
2046    20 8526         JSR     DMPSYB          ;DUMP ANY BAD SYMBOLS THAT MAY EXIST
2049    4C 2820         JMP     ASKPAS          ;AND GO PROMPT USER AGAIN
204C    20 ED20 DOP2:   JSR     PASS2           ;DO PASS 2
204F    20 9D26         JSR     DMPSCL          ;DUMP THE SYMBOL TABLE TO LISTING DEVICE
                                                ;AND TO THE CONSOLE DEVICE
2052    A2 79           LDX#    MESNER          ;ASSUME WE GOT NO ERRORS
2054    A5 41           LDA     ERRCNT          ;DID WE GET ANY?
2056    05 42           ORA     ERRCNT+1
2058    F0 02           BEQ     ERRMC           ;BRANCH IF NO ERRORS DETECTED
205A    A2 77           LDX#    MESGER          ;WE GOT ERRORS, LOAD ADDR OF NEW MESSAGE
205C    8A      ERRMC:  TXA                     ;SAVE MESSAGE ADDR
205D    48              PHA                     ;ON STACK
205E    20 EF29         JSR     OUTSTR          ;OUTPUT APPROPRIATE MESSAGE TO CONSOLE
2061    68              PLA                     ;RESTORE MESSAGE ADDR
2062    AA              TAX
2063    A5 21           LDA     LSTDEV          ;IS THE LISTING DEVICE,
2065    C9 00           CMP#    CONODV          ;THE CONSOLE?
2067    F0 03           BEQ     ERRME           ;BRANCH IF YES, DON'T OUTPUT TWICE
2069    20 3327         JSR     LOUTST          ;NO, THEN OUTPUT TO LISTING DEVICE ALSO
206C    20 3D2A ERRME:  JSR     CLOSEO          ;CLOSE OUTPUT FILES
206F    4C 2820         JMP     ASKPAS          ;AND PROMPT USER AGAIN
```

```
                                .PAGE                   ;'ROUTINE TO PICK UP DEVICE NUMBERS FROM USER'


                        GETODV:
2072  A2 5F     GETBDV: LDX#    MBINN                   ;ASK USER TO INPUT BINARY DEVICE NUMBER
2074  20 9C20           JSR     GDEVNO                  ;AND PICK UP NUMBER
2077  B0 F9             BCS     GETBDV                  ;ASK HIM AGAIN IF NUMBER WAS BAD.
2079  85 22             STA     BINDEV                  ;STORE THE BINARY DEVICE NUMBER.
207B  AA                TAX                             ;LOAD X REGISTER WITH DEVICE NUMBER
207C  20 C729           JSR     INODEV                  ;AND 'INITIALIZE' IT
207F  A2 61     GETLDV: LDX#    MLSTN                   ;ASK USER TO INPUT LISTING DEVICE NUMBER
2081  20 9C20           JSR     GDEVNO                  ;AND PICK UP THE NUMBER
2084  B0 F9             BCS     GETLDV                  ;ASK HIM AGAIN IF NUMBER WAS BAD
2086  85 21             STA     LSTDEV                  ;STORE THE LISTING DEVICE NUMBER
2088  AA                TAX                             ;LOAD X REGISTER WITH DEVICE NUMBER
2089  20 C729           JSR     INODEV                  ;'INITIALIZE' IT
208C  A2 67     ASKWID: LDX#    MWIDTH                  ;ASK USER TO TELL US THE WIDTH OF
208E  20 AD20           JSR     ASKDEC                  ;HIS LISTING DEVICE
2091  A5 2C             LDA     NUMBH                   ;GET HIGH ORDER OF ANSWER
2093  D0 F7             BNE     ASKWID                  ;DON'T ACCEPT ANYTHING GREATER THAN 127
2095  A5 2D             LDA     NUMBL
2097  30 F3             BMI     ASKWID
2099  85 43             STA     LWIDTH                  ;STORE FOR LATER
209B  60                RTS                             ;AND RETURN

                                ;ROUTINE TO INPUT A DEVICE NUMBER WITH PROMPT.

209C  20 AD20 GDEVNO: JSR       ASKDEC                  ;PROMPT AND ACCEPT DECIMAL INPUT
209F  A5 2C             LDA     NUMBH                   ;GET HIGH ORDER 8 BITS
20A1  D0 08             BNE     BADEVN                  ;BRANCH IF NOT ZERO.  NUMBER IS BAD
20A3  A5 2D             LDA     NUMBL                   ;GET LOW ORDER 8 BITS
20A5  C5 25             CMP     MAXDEV                  ;IS IT WITHIN RANGE?
20A7  10 02             BPL     BADEVN                  ;BRANCH IF NOT
20A9  18                CLC                             ;INDICATE ITS A GOOD ONE
20AA  60                RTS                             ;AND RETURN
20AB  38        BADEVN: SEC                             ;INDICATE DEVICE NUMBER WAS BAD
20AC  60                RTS                             ;AND RETURN

                                ;ROUTINE TO PROMPT FOR A DECIMAL NUMBER

20AD  20 EF29 ASKDEC: JSR       OUTSTR                  ;ASK FOR INPUT
20B0  20 CD25         JSR       GETLIN                  ;GET THE INPUT
20B3  4C D128         JMP       INPDEC                  ;* PJMP * EXTRACT DECIMAL NUMBER FROM INPUT

                                ;ROUTINE TO ASK FOR A SOURCE DEVICE NUMBER

20B6  A2 5D     GETSRC: LDX#    MSRCN                   ;PROMPT THE USER
20B8  20 9C20           JSR     GDEVNO                  ;FOR A DEVICE NUMBER
20BB  B0 F9             BCS     GETSRC                  ;BRANCH IF BAD DEVICE NUMBER
20BD  85 20             STA     SRCDEV                  ;STORE IT AWAY
20BF  60                RTS                             ;AND RETURN
```

```
                                .PAGE                   ;'TOP LEVEL ROUTINES FOR EACH PASS'


20C0  A5 3B   PASS1:  LDA     FLAGS             ;INDICATE THIS IS PASS ONE
20C2  29 7F           AND#    STSP1
20C4  29 BF           AND#    STCEOS            ;AND END OF SOURCE NOT SEEN YET
20C6  85 3B           STA     FLAGS             ;SAVE FOR OTHERS
20C8  A2 73   ASKSYM: LDX#    MFLSYM            ;ASK USER IF HE WANTS US TO
20CA  20 EF29         JSR     OUTSTR            ;FLUSH THE SYMBOL TABLE
20CD  20 CD25         JSR     GETLIN            ;GET A LINE OF INPUT
20D0  C9 4E           CMP#    'N                ;WAS ANSWER 'NO'?
20D2  F0 09           BEQ     PASSC             ;BRANCH IF YES, KEEP SYMBOL TABLE
20D4  C9 59           CMP#    'Y                ;WAS ANSWER 'YES'?
20D6  D0 F3           BNE     ASKSYM            ;BRANCH IF BAD INPUT
20D8  A9 FF           LDA#    SYMEOT            ;NOW MARK SYMBOL TABLE
20DA  8D B02F         STA     SYMCLR            ;AS EMPTY
20DD  20 FD20 PASSC:  JSR     PINIT             ;INIT SOME STUFF
20E0  20 CD25 NEXTP:  JSR     GETLIN            ;INPUT A LINE OF SOURCE INTO LINE BUFFER
20E3  20 1C21         JSR     PROCLN            ;NOW PROCESS IT
20E6  A9 40           LDA#    STSEOS            ;END OF SOURCE REACHED?
20E8  24 3B           BIT     FLAGS
20EA  F0 F4           BEQ     NEXTP             ;BRANCH IF NOT
20EC  60              RTS                       ;YES THEN RETURN

20ED  20 A428 PASS2:  JSR     MARESE            ;RESET BINARY OUTPUT ROUTINES
20F0  20 7220         JSR     GETODV            ;GET DEVICE SPECS FOR OUTPUT DEVICES
20F3  A5 3B           LDA     FLAGS             ;INDICATE THIS IS PASS 2
20F5  29 BF           AND#    STCEOS            ;AND END OF SOURCE NOT SEEN
20F7  09 80           ORA#    STSP2
20F9  85 3B           STA     FLAGS
20FB  D0 E0           BNE     PASSC             ;* PBNE * AND ENTER COMMON CODE

                      ;SOME COMMON INITIALIZATION BEFORE A PASS STARTS

20FD  20 842A PINIT:  JSR     SETCON            ;SET I-O DEVICES TO CONSOLE
2100  A9 00           LDA#    0                 ;GET A ZERO
2102  8D B02F         STA     LOCL              ;RESET LOCATION COUNTER
2105  8D B82F         STA     LOCH
2108  85 41           STA     ERRCNT            ;RESET ERROR COUNTER
210A  85 42           STA     ERRCNT+1
210C  20 B620 NEWSRC: JSR     GETSRC            ;GET DEVICE SPECS FOR SOURCE DEVICE
210F  A2 68           LDX#    MREADY            ;ASK USER TO READY SOURCE DEVICE
2111  20 9C20         JSR     GDEVNO            ;AND ACCEPT A NUMBER AS A DUMMY INPUT
2114  A5 23           LDA     SRCDEV            ;NOW MAKE SOURCE DEVICE THE INPUT DEVICE
2116  85 26           STA     DEVI
2118  AA              TAX                       ;MOVE INTO X REGISTER
2119  4C B229         JMP     INIDEV            ;* PJMP * AND INITIALIZE THE DEVICE
```

```
                                ,PAGE                   ;'MAIN LOOP TO PROCESS A LINE OF SOURCE'

211C A9 00  PROCLN: LDA#    0           ;GET A ZERO
211E 85 3C          STA     NBYTES      ;INDICATE NO BYTES GENERATED YET
2120 85 40          STA     ERRFLG      ;AND NO ERRORS ON THIS LINE YET
2122 20 EA24        JSR     GETSYM      ;TRY TO GET A SYMBOL AT FIRST OF LINE
2125 E0 00          CPX#    0           ;DID WE GET ANYTHING?
2127 F0 4B          BEQ     PNOSYM      ;BRANCH IF NO SYMBOL AT FIRST OF LINE
2129 A5 32          LDA     CHAR        ;GET CHAR WHICH TERMINATED THE SYMBOL
212B C9 3A          CMP#    ':
212D F0 58          BEQ     LABEL       ;TERMINATOR OF ':' INDICATES A LABEL
212F C9 09          CMP#    TABCHR      ;TABULATION?
2131 F0 54          BEQ     LABEL       ;YES, THEN DON'T REQUIRE COLON,
2133 20 CE24 LABERR: JSR    SLERR       ;INDICATE ERROR IS IN LABEL
2136 4C E921 PENDL: JMP     ENDL        ;* PJMP *  HANDLE END-OF-LINE AND RETURN
                    ;
2139 20 EA24 PROPC: JSR     GETSYM      ;TRY TO COLLECT A SYMBOL NAME
213C E0 00          CPX#    0           ;DID WE GET ANYTHING?
213E F0 05          BEQ     PROPCE      ;ERROR IF WE DIDN'T GET ANYTHING
2140 20 2425        JSR     OPTSRC      ;TRY TO LOCATE THIS SYMBOL IN THE OPCODE TABLE
2143 30 03          BMI     PROPC1      ;BRANCH IF WE FOUND IT
2145 A5 32  PROPCE: LDA     CHAR        ;GET TERMINATOR
2147 C9 3B          CMP#    ';          ;COMMENT?
2149 F0 EB          BEQ     PENDL       ;YES, THEN END OF LINE, OTHERWISE
214B 20 D624        JSR     SOERR       ;INDICATE OPCODE IS IN ERROR
214E D0 E6          BNE     PENDL       ;UNCONDITIONAL BRANCH
2150 A9 00  PROPC1: LDA#    0           ;INDICATE NO BYTES GENERATED YET
2152 85 3C          STA     NBYTES
2154 20 4324        JSR     GETSV       ;EXTRACT THE OPCODE'S VALUE BYTES
2157 A5 2D          LDA     NUMBL       ;STORE THEM IN SPECIAL PLACE
2159 85 46          STA     OPCVL
215B A5 2C          LDA     NUMBH
215D 85 47          STA     OPCVH
215F B1 48          LDA@Y   SYMADR      ;GET THE STATUS BYTE
2161 29 07          AND#    SYMCLS      ;KEEP ONLY THE CLASS CODE
2163 AA             TAX                 ;INTO X REGISTER
2164 BD EA2B        LDAX    CDSPL       ;GET ADDR OF ROUTINE TO HANDLE THE CLASS
2167 85 34          STA     JTEMP       ;STORE IT
2169 BD E32B        LDAX    CDSPH       ;HIGH ORDER ALSO
216C 85 35          STA     JTEMP+1
216E 20 3300        JSR     JSRIND      ;CALL THE ROUTINE
2171 4C 3621 JPENDL: JMP    PENDL       ;AND ASSUME END-OF-LINE
                    ;
```



```
                                ,PAGE                   ;'HERE TO FIND SOMETHING INTERESTING'

2174 A5 32  PNOSYM: LDA     CHAR        ;GET CHAR
2176 C9 3B  PNOSY1: CMP#    ';          ;COMMENT?
2178 F0 BC          BEQ     PENDL       ;YES, THEN END OF LINE
217A C9 0D          CMP#    EOLCHR      ;BLANK LINE?
217C F0 B8          BEQ     PENDL       ;YES, THEN END OF LINE
217E C9 09          CMP#    TABCHR      ;TABULATION?
2180 D0 B7          BNE     PROPC       ;BRANCH IF NOT, TRY TO FIND AN OPCODE
2182 20 6225 PEATC: JSR     GETC        ;YES, EAT IT
2185 10 EF          BPL     PNOSY1      ;AND UNCONDITIONALLY CHECK FOR SPECIAL CHARACTE
```

```
600                             .PAGE                   ;'ROUTINE TO PROCESS LABELS'
601
602
603
604  2187  20 2C25 LABEL:  JSR     USYSRC          ;TRY TO FIND THE LABEL IN USER SYMBOL TABLE
605  218A  10 46           BPL     LNOTF           ;BRANCH IF NOT FOUND
606  218C  A6 3B           LDX     FLAGS           ;GET GLOBAL FLAGS
607  218E  30 1A           BMI     LPASS2          ;BRANCH IF PASS 2.
608  2190  24 4D           BIT     BMUDF           ;PASS 1, IS THE SYMBOL REFERENCED, BUT UNDEFINE
609  2192  F0 35           BEQ     LMDEF           ;NO, THEN THIS IS A MULTIPLE DEFINITION
610  2194  29 F7           AND#    SYCUDF          ;OK MAKE IT DEFINED
611  2196  09 10           ORA#    SYMLAB          ;AS A LABEL
612  2198  91 48           STA@Y   SYMADR          ;PUT STATUS BACK INTO TABLE
613  219A  AD BC2F         LDA     LOCL            ;NOW GIVE IT THE VALUE OF THE LOCATION
614  219D  85 2D           STA     NUMBL           ;COUNTER
615  219F  AD BB2F         LDA     LOCH
616  21A2  85 2C           STA     NUMBH
617  21A4  20 6324         JSR     SETSV
618  21A7  4C B221 ENDLAB: JMP     PEATC           ;EAT THE LABEL TERMINATOR, THEN CONTINUE
619                        ;
620  21AA  24 4C   LPASS2: BIT     BMLAB           ;DEFINED DURING PASS ONE AS A LABEL?
621  21AC  F0 16           BEQ     LIERR           ;BRANCH IF NOT, ITS AN ERROR
622  21AE  20 4B24         JSR     GETSV           ;YES, GET IT'S VALUE
623  21B1  A5 2D           LDA     NUMBL           ;IS IT EQUAL TO CURRENT LOCATION COUNTER?
624  21B3  CD BC2F         CMP     LOCL
625  21B6  D0 07           BNE     LPERR           ;NO, THEN WE ARE OUT OF PHASE
626  21B8  A5 2C           LDA     NUMBH
627  21BA  CD BB2F         CMP     LOCH            ;HIGH ORDERS ALSO?
628  21BD  F0 E8           BEQ     ENDLAB          ;BRANCH IF THEY ARE EQUAL, ALL IS OK
629  21BF  20 CA24 LPERR:  JSR     SPERR           ;INDICATE PHASE ERROR
630  21C2  D0 AD           BNE     JPENDL          ;UNCONDITIONALLY TREAT AS END-OF-LINE.
631                        ;
632  21C4  20 D224 LIERR:  JSR     SIERR           ;INDICATE INTERNAL ERROR
633  21C7  D0 A8           BNE     JPENDL          ;UNCONDITIONAL BRANCH
634  21C9  09 04   LMDEF:  ORA#    SYMMDF          ;FLAG SYMBOL AS 'MULTIPLY-DEFINED'
635  21CB  91 48           STA@Y   SYMADR          ;IN SYMBOL TABLE
636  21CD  20 DA24         JSR     SMERR           ;INDICATE 'MULTIPLY-DEFINED' ERROR
637  21D0  D0 9F           BNE     JPENDL          ;UNCONDITIONAL BRANCH
```



```
638                             .PAGE                   ;'ROUTINE TO PROCESS LABELS'
639
640
641                             ;HERE IF THIS IS THE FIRST TIME WE HAVE SEEN THIS
642                             ;SYMBOL.
643
644
645  21D2  A6 3B   LNOTF:  LDX     FLAGS           ;PASS 2?
646  21D4  30 EE           BMI     LIERR           ;YES, THEN WHY HAVEN'T WE SEEN THIS LABEL?
647  21D6  A9 80           LDA#    USYM            ;GET SKELETON USER SYMBOL STATUS
648  21D8  09 10           ORA#    SYMLAB          ;INDICATE IT IS NOW DEFINED AS A LABEL
649  21DA  AE BC2F         LDX     LOCL            ;GET LOW ORDER LOCATION COUNTER
650  21DD  86 2D           STX     NUMBL           ;SAVE AS LOW ORDER VALUE FOR LABEL
651  21DF  AE BB2F         LDX     LOCH            ;GET HIGH ORDER LOCATION COUNTER
652  21E2  86 2C           STX     NUMBH           ;SAVE AS HIGH ORDER VALUE FOR LABEL
653  21E4  20 7824         JSR     ADDSYM          ;ADD THE SYMBOL TO THE SYMBOL TABLE
654  21E7  D0 BE           BNE     ENDLAB          ;UNCONDITIONALLY END OF LABEL
655
```

```
                                .PAGE                   ;'ROUTINE TO HANDLE END-OF-LINE'

                                ;HERE WHEN ALL PROCESSING IS THROUGH ON A SOURCE LINE

21E9  A5 3B   ENDL1: LDA    FLAGS               ;WHICH PASS?
21EB  10 19          BPL    EADDL               ;PASS 1, JUST UPDATE LOCATION COUNTER
21ED  20 B327        JSR    OUTBIN              ;OUTPUT BINARY (IF THERE IS ANY).
21F0  A5 21          LDA    LSTDEV              ;GET LISTING DEVICE NUMBER
21F2  85 23          STA    TLDEV               ;TELL LISTING OUTPUT ROUTINE TO OUTPUT THERE
21F4  20 1822        JSR    LISTLN              ;LIST THE SOURCE LINE WITH ALL THE GOODIES
21F7  A5 40          LDA    ERRFLG              ;DID THIS LINE HAVE ANY ERRORS?
21F9  F0 08          BEQ    EADDL               ;BRANCH IF NO ERRORS
21FB  A9 00          LDA#   CONODV              ;GET CONSOLE DEVICE NUMBER
21FD  C5 21          CMP    LSTDEV              ;IS THE CONSOLE ALREADY THE LISTING DEVICE?
21FF  F0 05          BEQ    EADDL               ;BRANCH IF YES, DON'T OUTPUT AGAIN
2201  85 23          STA    TLDEV               ;AS LISTING DEVICE
2203  20 1822        JSR    LISTLN
                     EADDL:                     ;* PFALL * UPDATE LOCATION COUNTER AND RETURN

                                ;SUBROUTINE TO UPDATE THE LOCATION COUNTER BY 'NBYTES'

2206  18       ADDLOC: CLC                       ;ADD 'NBYTES'
2207  AD BC2F         LDA    LOCL                ;TO LOCATION COUNTER
220A  65 3C           ADC    NBYTES
220C  8D BC2F         STA    LOCL
220F  AD BB2F         LDA    LOCH
2212  69 00           ADC#   0                   ;16 BIT ADD
2214  8D BB2F         STA    LOCH
2217  60              RTS                        ;AND RETURN
```

Problem for pseudo-paging (device #5)

```
                                .PAGE                   ;'ROUTINE TO LIST THE LINE'

2216  20 8122 LISTLN: JSR   RESETH          ;RESET TO FIRST OF LINE
2218  A5 3C           LDA   NBYTES          ;GET NUMBER OF BYTES TO GENERATE
221D  D0 05           BNE   LADDR           ;BRANCH IF WE OUTPUT SOME
221F  20 6127         JSR   LSP4            ;NO, THEN FOUR SPACES
2222  10 10           BPL   LADDRC          ;UNCONDITIONAL BRANCH
2224  AD BC2F LADDR:  LDA   LOCH            ;GET HIGH ORDER ADDRESS
2227  85 2D           STA   NUMBL           ;SAVE IN LOW ORDER VALUE
2229  20 4E27         JSR   LBYTO           ;OUTPUT IT
222C  AD BC2F         LDA   LOCL            ;GET LOW ORDER BYTE
222F  85 2D           STA   NUMBL
2231  20 4E27         JSR   LBYTO           ;OUTPUT IT
2234  20 6727 LADDRC: JSR   LSPACE          ;SPACE FOR SEPARATION
2237  A5 3C           LDA   NBYTES          ;GET NUMBER OF BYTES TO GENERATE
2239  F0 12           BEQ   LENDB           ;BRANCH IF NONE TO DO
223B  A0 00           LDY#  0               ;POINT TO FIRST BYTE
223D  B9 3D00 LBLOOP: LDAY  BYTE1           ;GET A BYTE
2240  85 2D           STA   NUMBL           ;SAVE IT
2242  20 4E27         JSR   LBYTO           ;OUTPUT IT
2245  20 6727         JSR   LSPACE          ;SPACE FOR SEPARATION
2248  C8              INY                   ;POINT TO NEXT BYTE
2249  C4 3C           CPY   NBYTES          ;DONE YET?
224B  D0 F0           BNE   LBLOOP          ;BRANCH IF MORE TO OUTPUT
224D  20 6225 LENDB:  JSR   GETC            ;LOOK AHEAD FOR END OF LINE CHARACTER
2250  C9 0D           CMP#  $0D             ;IS IT AN END OF LINE CHARACTER
2252  F0 0C           BEQ   LCPPY1          ;NO TAB IF IT IS
2254  48              PHA                   ;SAVE THE CHARACTER
2255  A9 09           LDA#  TABCHR          ;PRINT A TAB
2257  20 6927         JSR   LOUTC           ;PRINT IT
225A  68              PLA                   ;GET CHARACTER BACK
225B  D0 03           BNE   LCPPY1          ;UNCONDITIONAL BRANCH
225D  20 6225 LCOPYL: JSR   GETC            ;GET A CHAR FROM LINE BUFFER
2260  20 6927 LCPPY1: JSR   LOUTC           ;LIST IT
2263  C9 0D           CMP#  EOLCHR          ;END OF LINE?
2265  D0 F6           BNE   LCOPYL          ;NO, CONTINUE TO COPY LINE TO LISTING DEVICE
2267  A5 40           LDA   ERRFLG          ;DOES THIS LINE HAVE ANY ERRORS?
2269  F0 15           BEQ   LISTRT          ;BRANCH IF NOT
226B  A0 FF           LDY#  $FF             ;INIT ERROR NAME TABLE POINTER
226D  C8        SHERR:  INY                 ;POINT TO NEXT ERROR NAME
226E  4A              LSRA                  ;SHIFT NEXT POSSIBLE ERROR BIT INTO 'C'
226F  90 FC           BCC   SHERR           ;BRANCH IF NOT THIS ERROR
2271  48              PHA                   ;GOT AN ERROR, SAVE REST OF ERROR BITS
2272  B9 DB2B         LDAY  ERRTAB          ;GET THE CHAR TO OUTPUT FOR THIS ERROR
2275  20 6927         JSR   LOUTC           ;LIST IT
2278  68              PLA                   ;RESTORE REMAINING ERROR BITS
2279  D0 F2           BNE   SHERR           ;BRANCH IF MORE ERRORS
227B  A2 69           LDX#  MERROR          ;ALL ERRORS LISTED, OUTPUT ATTENTION
227D  4C 3027         JMP   LOUTST          ;* PJMP * GETTING MESSAGE AND RETURN
2280  60        LISTRT: RTS                 ;RETURN

                                ;ROUTINE TO RESET TO FIRST OF LINE BUFFER

2281  A9 00    RESETH: LDA#  0               ;INDICATE HORIZONTAL POSITION
2283  85 44           STA   HPOS            ;ON LISTING DEVICE IS ZERO
2285  4C 8C25         JMP   TXTINI          ;* PJMP * RESET LINE BUFFER POINTERS AND RETURN
```

```
                        .PAGE                   ;'EXPRESSION EVALUATOR'


                        ;LEFT TO RIGHT EXPRESSION EVALUATOR

2288  A5 32   GEXPR:   LDA     CHAR          ;GET CHARACTER
228A  C9 0D             CMP#    EOLCHR        ;END OF LINE?
228C  D0 03             BNE     GEXPR1        ;BRANCH IF NOT
228E  4C 1C29           JMP     ZERNUM        ;* JUMP * YES, RETURN ZERO
2291  20 6225 GEXPR1:  JSR     GETC          ;GET NEXT CHARACTER
2294  20 F725 EXPR:    JSR     ATOM          ;GET FIRST TERM
2297  A5 32   EXPNXT:  LDA     CHAR          ;GET TERMINATOR OF TERM
2299  C9 0D             CMP#    EOLCHR        ;END OF LINE?
229B  F0 34             BEQ     EXPRET        ;BRANCH IF YES
229D  C9 2B             CMP#    '+            ;OPERATOR?
229F  F0 04             BEQ     OPROK         ;BRANCH IF YES
22A1  C9 2D             CMP#    '-
22A3  D0 2C             BNE     EXPRET        ;NO, THEN RETURN
22A5  48        OPROK:  PHA                   ;SAVE THE OPERATOR
22A6  A5 2D             LDA     NUMBL         ;SAVE 'NUMB'
22A8  48                PHA
22A9  A5 2C             LDA     NUMBH
22AB  48                PHA
22AC  20 6225           JSR     GETC          ;MOVE PAST OPERATOR
22AF  20 F725           JSR     ATOM          ;PICK UP NEXT TERM
22B2  A5 2D             LDA     NUMBL         ;PLACE IT IN 'NTEMP'
22B4  85 2F             STA     NTEMPL
22B6  A5 2C             LDA     NUMBH
22B8  85 2E             STA     NTEMPH
22BA  68                PLA                   ;RESTORE VALUE THUS FAR
22BB  85 2C             STA     NUMBH
22BD  68                PLA
22BE  85 2D             STA     NUMBL
22C0  68                PLA                   ;GET THE OPERATOR
22C1  C9 2B             CMP#    '+            ;ADDITION?
22C3  F0 06             BEQ     EXADD         ;BRANCH IF YES
22C5  20 2329           JSR     ISUB          ;NO, THEN MUST BE SUBTRACTION
22C8  4C 9722           JMP     EXPNXT        ;FORM NEW VALUE, LOOK FOR NEXT TERM
22CB  20 8E29 EXADD:    JSR     IADD          ;FORM NEW VALUE
22CE  4C 9722           JMP     EXPNXT        ;AND LOOK FOR NEXT TERM
22D1  60        EXPRET: RTS                   ;RETURN
```

only

from lines 760, 764 only

```
                                .PAGE                   ;'ROUTINES TO HANDLE OPCODE CLASSES'

                                ;CLASS 0.  (IMMEDIATE MODE ADDRESSING)

22D2  20 8822  CLASS0:  JSR     GEXPR           ;GET AN OPERAND VALUE
22D5  A5 46             LDA     OPCVL           ;GET THE OPCODE
22D7  85 3D             STA     BYTE1           ;AS FIRST BYTE TO OUTPUT
22D9  A5 2D             LDA     NUMBL           ;GET LOW ORDER VALUE
22DB  85 3E             STA     BYTE2           ;SAVE AS SECOND BYTE
22DD  A9 02             LDA#    2               ;INDICATE TWO BYTES ARE TO BE OUTPUT
22DF  85 3C             STA     NBYTES
22E1  60                RTS                     ;AND RETURN

                                ;CLASS 1.  (DIRECT MEMORY REFERENCE)

22E2  20 8822  CLASS1:  JSR     GEXPR           ;GET AN OPERAND VALUE
22E5  A5 2C             LDA     NUMBH           ;LOOK AT HIGH ORDER ADDR
22E7  D0 0D             BNE     CABS            ;BRANCH IF WE MUST USE ABSOLUTE ADDRESSING
22E9  A5 47             LDA     OPCVH           ;GET OPCODE FOR PAGE ZERO ADDRESSING
22EB  85 3D   CPZERO:   STA     BYTE1           ;SAVE OPCODE AS FIRST BYTE TO OUTPUT
22ED  A5 2D             LDA     NUMBL           ;GET LOW ORDER VALUE
22EF  85 3E             STA     BYTE2           ;SAVE AS SECOND BYTE TO OUTPUT
22F1  A9 02             LDA#    2               ;INDICATE 2 BYTES TO OUTPUT
22F3  85 3C             STA     NBYTES
22F5  60                RTS                     ;AND RETURN
22F6  A5 46   CABS:     LDA     OPCVL           ;GET OPCODE FOR ABSOLUTE ADDRESSING
22F8  85 3D             STA     BYTE1           ;STORE AS FIRST BYTE TO OUTPUT
22FA  A5 2D             LDA     NUMBL           ;GET LOW ORDER ADDR
22FC  85 3E             STA     BYTE2           ;SAVE AS SECOND BYTE TO OUTPUT
22FE  A5 2C             LDA     NUMBH           ;GET HIGH ORDER
2300  85 3F             STA     BYTE3           ;SAVE AS THIRD BYTE TO OUTPUT
2302  A9 03             LDA#    3               ;INDICATE 3 BYTES ARE TO BE OUTPUT
2304  85 3C             STA     NBYTES
2306  60                RTS                     ;AND RETURN

                                ;CLASS2.  (ABSOLUTE ADDRESSING ALWAYS)

2307  20 8822  CLASS2:  JSR     GEXPR           ;GET OPERAND
230A  4C F622           JMP     CABS            ;* JUMP * STORE 3 BYTES AND RETURN
```

```
                        .PAGE                   ;'ROUTINES TO HANDLE OPCODE CLASSES'


                        ;CLASS3.  (MANDATORY ZERO PAGE ADDRESSING)

230D  20 8822 CLASS3: JSR     GEXPR             ;GET OPERAND
2310  A9 02           LDA#    2                 ;INDICATE 2 BYTES ARE TO BE STORED
2312  85 3C           STA     NBYTES            ;IN CASE WE GET AN ERROR
2314  A5 2C           LDA     NUMBH             ;GET HIGH ORDER ADDR
2316  F0 28           BEQ     CONE              ;IT'S ZERO, SO USE ONE-BYTE ADDR
2318  4C C224 CAERR:  JMP     SAERR             ;* PJMP * INDICATE ADDRESSING ERROR, THEN RETUR

                        ;CLASS4.  (RELATIVE ADDRESSING ALWAYS)

231B  20 8822 CLASS4: JSR     GEXPR             ;GET OPERAND
231E  A9 02           LDA#    2                 ;INDICATE TWO BYTES ARE TO BE STORED
2320  85 3C           STA     NBYTES            ;IN CASE WE GET AN ERROR
2322  AD BC2F         LDA     LOCL              ;GET THE CURRENT LOCATION COUNTER.
2325  85 2F           STA     NTEMPL            ;SAVE IT IN 'NTEMP'
2327  AD BB2F         LDA     LOCH
232A  85 2E           STA     NTEMPH
232C  20 2329         JSR     ISUB              ;FORM ADDR-.
232F  A9 00           LDA#    0
2331  85 2E           STA     NTEMPH
2333  A9 02           LDA#    2
2335  85 2F           STA     NTEMPL
2337  4C 81 00         JMP     BFIX              ;FIX FOR FAILURE TO  TEST BRANCH OFFSET
233A  A5 2C           LDA     NUMBH             ;GET HIGH ORDER ADDR OF OFFSET
233C  30 07           BMI     RELBAK            ;BRANCH IF RELATIVE ADDR IS NEGATIVE
233E  D0 D8           BNE     CAERR             ;POSITIVE,  ERROR IF > 127.
2340  A5 46    CONE:   LDA     OPCVL             ;GET OPCODE VALUE
2342  4C EB22         JMP     CPZERO            ;* PJMP * STORE AND RETURN
2345  C9 FF    RELBAK: CMP#    $FF               ;HIGH ORDER MUST BE ALL ONES
2347  F0 F7           BEQ     CONE              ;OK IF IT IS
2349  D0 CD           BNE     CAERR             ;OTHERWISE IT'S OUT OF RANGE.

                        ;CLASS5.  (ONE BYTE OF OPCODE ONLY,  NO ADDRESS,)

234B  A5 46   CLASS5: LDA     OPCVL             ;GET OPCODE VALUE
234D  85 3D           STA     BYTE1             ;STORE AS FIRST BYTE TO OUTPUT
234F  A9 01           LDA#    1                 ;INDICATE ONE BYTE TO BE OUTPUT
2351  85 3C           STA     NBYTES
2353  60             RTS                       ;AND RETURN

                        ;CLASS6.  (PSUEDO-OPS)

2354  A5 2D   CLASS6: LDA     NUMBL             ;GET LOW ORDER ADDR OF ROUTINE TO
2356  85 34           STA     JTEMP             ;HANDLE THE PSUEDO-OP
2358  A5 2C           LDA     NUMBH
235A  85 35           STA     JTEMP+1
235C  6C 3400         JMP@    JTEMP             ;* PJMP * CALL ROUTINE, THEN RETURN
```

```
880                             .PAGE                   ;'ROUTINES TO HANDLE PSLEDO-OPS'
881
882
883                             ;'.HBYTE' RETURNS HIGH ORDER BYTE OF EXPRESSION
884                             ;'.BYTE' RETURNS LOW ORDER BYTE OF EXPRESSION
885
886  235F  20 8822 PHBYTE:  JSR     GEXPR           ;GET OPERAND
887  2362  A5 2C            LDA     NUMBH           ;GET HIGH ORDER BYTE
888  2364  4C 6C23          JMP     PBYTEC          ;AND ENTER COMMON CODE
889  2367  20 8822 PBYTE:   JSR     GEXPR           ;GET OPERAND
890  236A  A5 2D            LDA     NUMBL           ;GET LOW ORDER BYTE
891  236C  85 3D    PBYTEC: STA     BYTE1           ;STORE AS FIRST BYTE TO OUTPUT
892  236E  A9 01            LDA#    1               ;INDICATE ONE BYTE TO OUTPUT
893  2370  85 3C            STA     NBYTES
894  2372  60              RTS
895
896                             ;'.WORD' RETURNS TWO BYTES (LOW ORDER, HIGH ORDER).
897
898  2373  20 8822 PWORD:   JSR     GEXPR           ;GET OPERAND
899  2376  A5 2D            LDA     NUMBL           ;GET LOW ORDER BYTE
900  2378  85 3D            STA     BYTE1           ;STORE AS FIRST BYTE TO OUTPUT
901  237A  A5 2C            LDA     NUMBH           ;GET HIGH ORDER
902  237C  85 3E            STA     BYTE2           ;STORE AS SECOND BYTE TO OUTPUT
903  237E  A9 02            LDA#    2               ;INDICATE TWO BYTES TO OUTPUT
904  2380  85 3C            STA     NBYTES
905  2382  60              RTS                     ;AND RETURN
```



```
906                             .PAGE                   ;'ROUTINES TO HANDLE PSLEDO-OPS'
907
908
909                             ;'.LOC' SETS A NEW LOCATION COUNTER
910
911  2383  20 8822 PLOC:   JSR     GEXPR           ;GET OPERAND
912  2386  A5 2D            LDA     NUMBL           ;COPY VALUE INTO LOCATION COUNTER
913  2388  8D BC2F          STA     LOCL
914  238B  A5 2C            LDA     NUMBH
915  238D  8D BB2F          STA     LOCH
916  2390  A5 3B            LDA     FLAGS           ;WHICH PASS?
917  2392  10 2C            BPL     PRET            ;DON'T DO ANYMORE IF PASS 1.
918  2394  20 2628          JSR     HSTAR           ;INITIALIZE ROUTINE TO RECIEVE ADDRESS
919  2397  AD BB2F          LDA     LOCH            ;AND ALSO OUPUT THE NEW LOCATION COUNTER
920  239A  85 2D            STA     NUMBL
921  239C  20 F927          JSR     BBYTO           ;SO LOADER WILL KNOW WHERE IT GOES.
922  239F  AD BC2F          LDA     LOCL            ;LOW ORDER ALSO
923  23A2  85 2D            STA     NUMBL
924  23A4  20 F927          JSR     BBYTO
925  23A7  60              RTS                     ;RETURN
926
927                             ;'.PAGE' FORCES LISTING DEVICE TO THE TOP OF A NEW PAGE
928
929  23A8  A5 3B    PPAGE:  LDA     FLAGS           ;WHICH PASS?
930  23AA  10 14            BPL     PRET            ;DON'T DO ANYTHING IF PASS 1
931  23AC  A5 21            LDA     LSTDEV          ;SET LISTING DEVICE
932  23AE  85 23            STA     TLDEV           ;AS CURRENT DEVICE TO LIST TO
933  23B0  A2 63            LDX#    MPAGE           ;AND OUTPUT THE TOP-OF-PAGE STRING
934  23B2  4C 3027          JMP     LOUTST          ;* PJMP * AND RETURN
935
936                             ;'.END' SIGNALS END OF THE SOURCE INPUT
937
938  23B5  A5 3B    PEND:   LDA     FLAGS           ;GET GLOBAL FLAGS
939  23B7  09 40            ORA#    STSEOS          ;FLAG END OF SOURCE
940  23B9  85 3B            STA     FLAGS
941  23BB  10 03            BPL     PRET            ;RETURN IF PASS ONE
942  23BD  20 1F28          JSR     HCTZ            ;RESET BINARY OUTPUT ROUTINE
943  23C0  60       PRET:   RTS
```

```
                        .PAGE                   ;'ROUTINES TO HANDLE PSEUDO-OPS'

                        ;'.ASCII' STORES A STRING OF ASCII CHARACTERS WHICH ARE
                        ;ENCLOSED BY A PAIR OF DELIMITERS.

23C1  A5 32   PASCII:   LDA     CHAR            ;GET CHARACTER WHICH TERMINATED '.ASCII'
23C3  C9 0D             CMP#    EOLCHR          ;END OF LINE?
23C5  F0 3E             BEQ     PASERR          ;BRANCH IF YES.  BAD SYNTAX
23C7  A5 38             LDA     TEXTP           ;REMEMBER WHERE WE ARE ON THE LINE
23C9  48                PHA
23CA  20 E921           JSR     ENDL            ;OUTPUT THE LINE TO LISTING WITH NO BINARY
23CD  20 8C25           JSR     TXTINI          ;RESET POINTERS TO START OF LINE
23D0  A9 0D             LDA#    EOLCHR          ;GET END OF LINE CHARACTER
23D2  20 7525           JSR     PACKC           ;FAKE A BLANK LINE BY OVERWRITTING FIRST CHAR
23D5  68                PLA                     ;RESTORE TEXT POINTER TO START OF STRING
23D6  85 38             STA     TEXTP
23D8  20 9925           JSR     GSPNOR          ;GET FIRST NON-BLANK
23DB  C9 0D             CMP#    EOLCHR          ;END OF LINE?
23DD  F0 21             BEQ     PASEND          ;BRANCH IF YES.  NO TEXT TO OUTPUT
23DF  48                PHA                     ;YES, SAVE THE STRING DELIMITER.
23E0  20 6225 PASNXT:   JSR     GETC            ;GET NEXT CHAR FROM STRING
23E3  68                PLA                     ;RESTORE DELIMITER
23E4  C5 32             CMP     CHAR            ;IS THIS CHARACTER THE DELIMITER?
23E6  F0 18             BEQ     PASEND          ;BRANCH IF YES.  END OF STRING
23E8  A6 32             LDX     CHAR            ;NO. GET STRING CHARACTER
23EA  E0 0D             CPX#    EOLCHR          ;IS IT END OF LINE?
23EC  F0 12             BEQ     PASEND          ;BRANCH IF YES.  ALSO END OF STRING
23EE  48                PHA                     ;SAVE DELIMITER AGAIN
23EF  86 3D             STX     BYTE1           ;STORE STRING CHARACTER AS BYTE TO OUTPUT
23F1  A9 01             LDA#    1               ;INDICATE ONE BYTE TO OUTPUT
23F3  85 3C             STA     NBYTES
23F5  A5 38             LDA     TEXTP           ;SAVE OUR CURRENT POSITION IN THE LINE
23F7  48                PHA                     ;ACCROSS CALL
23F8  20 E921           JSR     ENDL            ;OUTPUT THE BYTE
23FB  68                PLA                     ;RESTORE OUT POSITION ON THE LINE
23FC  85 38             STA     TEXTP
23FE  10 E0             BPL     PASNXT          ;AND PROCESS NEXT CHAR IN STRING
2400  A9 00   PASEND:   LDA#    0               ;INDICATE NO BYTES ARE TO BE GENERATED
2402  85 3C             STA     NBYTES
2404  60                RTS                     ;RETURN
2405  4C BE24 PASERR:   JMP     SSERR           ;* PJMP * INDICATE SYNTAX ERROR AND RETURN
```

```
                        .PAGE                   ;'ROUTINES TO HANDLE PSEUDO-OPS'

                        ;'.DEF' DEFINES A SYMBOL'S VALUE
                        ;       .DEF    SYMBOL=EXPRESSION

2438  A5 32   PDEF:    LDA     CHAR            ;GET OPCODE TERMINATOR
243A  C9 0D             CMP#    EOLCHR          ;END OF LINE?
243C  F0 0A             BEQ     PDSERR          ;SYNTAX ERROR IF IT IS
243E  A5 38             LDA     TEXTP           ;SAVE LINE BUFFER POINTER
2440  48                PHA                     ;SO WE KNOW WHERE SYMBOL STARTS
2441  20 E724           JSR     GGTSYM          ;NOW TRY TO EXTRACT A SYMBOL NAME
2444  E0 00             CPX#    0               ;DID WE GET ANYTHING?
2446  D0 04             BNE     CHKARG          ;BRANCH IF WE DID
2448  68      PDSERR:   PLA                     ;ADJUST STACK
2449  4C BE24           JMP     SSERR           ;* PUMP * INDICATE SYNTAX ERROR AND RETURN
244C  A5 32   CHKARG:   LDA     CHAR            ;GET THE TERMINATOR
244E  C9 3D             CMP#    '=              ;EQUALS?
2450  D0 F6             BNE     PDSERR          ;SYNTAX ERROR IF NOT
2452  20 8822           JSR     GEXPR           ;NOW GET THE EXPRESSION ON RIGHT
2455  68                PLA                     ;RESTORE LINE BUFFER POINTER
2456  85 38             STA     TEXTP           ;TO POINT TO SYMBOL NAME ON LEFT SIDE
2458  20 E724           JSR     GGTSYM          ;NOW EXTRACT IT AGAIN
245B  20 2C25           JSR     USYSRC          ;TRY TO LOCATE THE SYMBOL IN SYMBOL TABLE
245E  10 0F             BPL     PDNOTF          ;BRANCH IF NOT THERE
2460  24 4C             BIT     BMLAB           ;IS IT ALREADY DEFINED AS A LABEL?
2462  D0 10             BNE     PDMERR          ;BRANCH IF YES
2464  24 4E             BIT     BMMDF           ;NO, BUT IS IT ALREADY MULTIPLY DEFINED?
2466  D0 0C             BNE     PDMERR          ;BRANCH IF YES
2468  29 F7             AND#    SYCUDF          ;MAKE SURE ITS DEFINED
246A  91 48             STA@Y   SYMADR          ;STORE UPDATED STATUS BACK
246C  4C 6324           JMP     SETSV           ;* PUMP * NOW SET ITS NEW VALUE AND RETURN
246F  A9 80   PDNOTF:   LDA#    USYM            ;GET SKELETON STATUS BYTE
2471  4C 7824           JMP     ADDSYM          ;* PUMP * ADD THE SYMBOL TO THE TABLE AND RETURN
2474  09 04   PDMERR:   ORA#    SYMMDF          ;INDICATE SYMBOL IS 'MULTIPLY-DEFINED'
2476  91 48             STA@Y   SYMADR          ;STORE UPDATED STATUS
2478  4C DA24           JMP     SMERR           ;* PUMP * SET ERROR FLAG AND RETURN
```

```
1025                            .PAGE                   ;'ROUTINES TO GET AND SET A SYMBOL'S VALUE'
1026
1027
1028                            ;RETURN THE VALUE FROM THE SYMBOL TABLE INTO 'NUMB'
1029
1030
1031  244B  20 1C29 GETSV:      JSR     ZERNUM          ;ASSUME A ZERO VALUE
1032  244E  98                  TYA                     ;REMEMBER POINTER
1033  244F  48                  PHA
1034  2450  B1 48               LDA@Y   SYMADR          ;GET STATUS BYTE
1035  2452  24 4B               BIT     BMSIZ           ;CHECK SIZE FLAG
1036  2454  D0 05               BNE     GETSV1          ;BRANCH IF ONLY ONE BYTE OF VALUE
1037  2456  C8                  INY                     ;POINT TO FIRST OF TWO BYTES
1038  2457  B1 48               LDA@Y   SYMADR          ;GET HIGH ORDER VALUE
1039  2459  85 2C               STA     NUMBH           ;SAVE IT
1040  245B  C8         GETSV1: INY                     ;POINT TO LOW ORDER VALUE
1041  245C  B1 48               LDA@Y   SYMADR          ;GET IT
1042  245E  85 2D               STA     NUMBL           ;SAVE IT
1043  2460  68                  PLA                     ;RESTORE POINTER TO STATUS BYTE
1044  2461  A8                  TAY
1045  2462  60                  RTS                     ;AND RETURN
1046
1047                            ;ROUTINE TO PLACE 'NUMB' INTO SYMBOL TABLE ENTRY.
1048
1049  2463  98         SETSV:  TYA                     ;SAVE POINTER TO STATUS BYTE
1050  2464  48                  PHA
1051  2465  B1 48               LDA@Y   SYMADR          ;GET STATUS BYTE
1052  2467  24 4B               BIT     BMSIZ           ;HOW MANY BYTES OF VALUE?
1053  2469  D0 05               BNE     SETSV1          ;BRANCH IF ONLY ONE
1054  246B  C8                  INY                     ;POINT TO HIGH ORDER
1055  246C  A5 2C               LDA     NUMBH
1056  246E  91 48               STA@Y   SYMADR          ;STORE HIGH ORDER OF 'NUMB' THERE
1057  2470  C8         SETSV1: INY                     ;POINT TO LOW ORDER
1058  2471  A5 2D               LDA     NUMBL           ;GET VALUE
1059  2473  91 48               STA@Y   SYMADR          ;PUT IN TABLE
1060  2475  68                  PLA                     ;RESTORE POINTER TO STATUS BYTE
1061  2476  A8                  TAY
1062  2477  60                  RTS                     ;AND RETURN
```

ADDSYM CALLED FROM LINES: 653 1021 1408

```
1063                            .PAGE                   ;'ROUTINE TO ADD A SYMBOL TO TABLE'
1064
1065
1066
1067                            ;ROUTINE TO ADD SYMBOL IN 'SYMTMP' TO END OF SYMBOL TABLE
1068                            ;ROUTINE IS CALLED WITH STATUS BYTE IN THE ACCUMULATOR.
1069
1070  2478  48          ADDSYM: PHA                     ;SAVE STATUS BYTE
1071  2479  18                  CLC                     ;GET READY TO ADD
1072  247A  98                  TYA                     ;OFFSET TO
1073  247B  65 48               ADC     SYMADR          ;BASE ADDR
1074  247D  A5 49               LDA     SYMADR+1
1075  247F  69 00               ADC#    0
1076  2481  C5 29               CMP     MAXPAG          ;DO WE STILL HAVE ROOM?
1077  2483  D0 0F               BNE     ADDS1           ;BRANCH IF THERE IS STILL ROOM
1078  2485  A9 00               LDA#    CONODV          ;OUT OF ROOM.  MAKE CONSOLE THE
1079  2487  85 27               STA     DEVO            ;CURRENT OUTPUT DEVICE
1080  2489  A2 71               LDX#    MOVFL           ;TELL USER THE SYMBOL TABLE OVERFLOWED
1081  248B  20 EF29             JSR     OUTSTR
1082  248E  A2 FF               LDX#    $FF             ;RESET STACK POINTER
1083  2490  9A                  TXS
1084  2491  4C 2820             JMP     ASKPAS          ;AND PROMPT FOR FURTHER INSTRUCTIONS
1085  2494  A2 00       ADDS1:  LDX#    0               ;SET OFFSET TO 'SYMTMP' TO ZERO
1086  2496  B5 50       INNAME: LDAX    SYMTMP          ;GET A CHAR FROM 'SYMTMP'
1087  2498  F0 06               BEQ     INDONE          ;BRANCH IF END OF NAME
1088  249A  91 48               STA@Y   SYMADR          ;STORE NAME CHAR IN SYMBOL TABLE
1089  249C  E8                  INX                     ;POINT TO NEXT BYTE IN 'SYMTMP'
1090  249D  C8                  INY                     ;POINT TO NEXT BYTE IN SYMBOL TABLE
1091  249E  D0 F6               BNE     INNAME          ;UNCONDITIONALLY BRANCH FOR NEXT CHAR
1092  24A0  68          INDONE: PLA                     ;GET STATUS BYTE BACK
1093  24A1  AA                  TAX                     ;SAVE IN X REGISTER
1094  24A2  98                  TYA                     ;SAVE POINTER TO STATUS BYTE
1095  24A3  48                  PHA                     ;ON STACK
1096  24A4  8A                  TXA                     ;GET STATUS BYTE BACK INTO ACCUMULATOR
1097  24A5  A6 2C               LDX     NUMBH           ;CHECK HIGH ORDER BYTE OF VALUE
1098  24A7  08                  PHP                     ;SAVE FLAGS FOR LATER
1099  24A8  D0 02               BNE     ADDSY1          ;BRANCH IF WE NEED 2 BYTES TO STORE VALUE
1100  24AA  09 40               ORA#    SYMSIZ          ;INDICATE WE ONLY NEED ONE BYTE
1101  24AC  91 48       ADDSY1: STA@Y   SYMADR          ;STORE STATUS IN SYMBOL TABLE
1102  24AE  20 6324             JSR     SETSV           ;AND STORE THE VALUE AWAY
1103  24B1  28                  PLP                     ;GET FLAGS BACK
1104  24B2  F0 01               BEQ     IONEB           ;BRANCH IF ONLY ONE BYTE OF VALUE
1105  24B4  C8                  INY                     ;POINT TO HIGH ORDER BYTE
1106  24B5  C8          IONEB:  INY                     ;POINT TO LOW ORDER BYTE
1107  24B6  C8                  INY                     ;POINT TO NEXT BYTE
1108  24B7  A9 FF               LDA#    SYMEOT          ;GET END-OF-TABLE FLAG
1109  24B9  91 48               STA@Y   SYMADR          ;STORE IT AT END OF TABLE
1110  24BB  68                  PLA                     ;RESTORE POINTER TO POINT AT
1111  24BC  A8                  TAY                     ;TO STATUS BYTE
1112  24BD  60                  RTS                     ;AND RETURN
1113
```

```
1114                                .PAGE                   ;'ROUTINE TO SET ERROR STATUS BITS'
1115
1116
1117
1118  24BE  A9 01   SSERR:  LDA#    SERR                    ;SYNTAX ERROR
1119  24C0  D0 1A           BNE     SERRT
1120  24C2  A9 02   SAERR:  LDA#    AERR                    ;ADDRESSING ERROR
1121  24C4  D0 16           BNE     SERRT
1122  24C6  A9 20   SUERR:  LDA#    UERR                    ;'UNDEFINED SYMBOL' ERROR
1123  24C8  D0 12           BNE     SERRT
1124  24CA  A9 08   SPERR:  LDA#    PERR                    ;PHASE ERROR
1125  24CC  D0 0E           BNE     SERRT
1126  24CE  A9 80   SLERR:  LDA#    LERR                    ;LABEL ERROR
1127  24D0  D0 0A           BNE     SERRT
1128  24D2  A9 10   SIERR:  LDA#    IERR                    ;INTERNAL ERROR
1129  24D4  D0 06           BNE     SERRT
1130  24D6  A9 04   SOERR:  LDA#    OERR                    ;OPCODE ERROR
1131  24D8  D0 02           BNE     SERRT
1132  24DA  A9 40   SMERR:  LDA#    MERR                    ;'MULTIPLY-DEFINED' ERROR
1133  24DC  E6 41   SERRT:  INC     ERRCNT                  ;COUNT THIS ERROR
1134  24DE  D0 02           BNE     SERRT1                  ;BRANCH IF HIGH ORDER OK
1135  24E0  E6 42           INC     ERRCNT+1
1136  24E2  05 40   SERRT1: ORA     ERRFLG                  ;'OR' IN REST OF FLAGS
1137  24E4  85 40           STA     ERRFLG                  ;SAVE UPDATED FLAGS BACK
1138  24E6  60              RTS                             ;RETURN
1139
1140
```



```
1141                                .PAGE                   ;'ROUTINE TO EXTRACT SYMBOL NAMES FROM SOURCE'
1142
1143
1144
1145  24E7  20 6225 GGTSYM: JSR     GETC                    ;GET NEXT CHARACTER
1146  24EA  A2 00   GETSYM: LDX#    0                       ;SET POINTER TO TEMP STORAGE TO ZERO
1147  24EC  20 9C25 GETSY1: JSR     SPNOR                   ;IGNORE BLANKS
1148  24EF  20 0525         JSR     SYMTRM                  ;IS CHARACTER A TERMINATOR FOR A SYMBOL?
1149  24F2  F0 0C           BEQ     GETSRT                  ;BRANCH IF YES
1150  24F4  E0 09           CPX#    SYMNMX                  ;REACHED MAX ALLOWED LENGTH OF SYMBOL NAME?
1151  24F6  10 03           BPL     NSYMC                   ;BRANCH IF YES, IGNORE THIS CHAR
1152  24F8  95 50           STAX    SYMTMP                  ;NO THEN STORE CHAR IN TEMP STORAGE
1153  24FA  E8              INX                             ;POINT TO NEXT TEMP STORAGE BYTE
1154  24FB  20 6225 NSYMC:  JSR     GETC                    ;GET NEXT CHARACTER FROM LINE
1155  24FE  10 EC           BPL     GETSY1                  ;UNCONDITIONALLY BRANCH
1156  2500  A9 00   GETSRT: LDA#    0                       ;GET A ZERO
1157  2502  95 50           STAX    SYMTMP                  ;FLAG THE END OF THE SYMBOL NAME
1158  2504  60              RTS                             ;AND RETURN
1159
1160                                ;ROUTINE TO SEE IF THE CHARACTER IN THE ACCUMULATOR
1161                                ;IS A TERMINATOR FOR A SYMBOL NAME.
1162                                ;RETURNS WITH Z=1 IF CHARACTER IS A TERMINATOR, Z=0 IF NOT.
1163
1164
1165  2505  C9 3A   SYMTRM: CMP#    ':                      ;COLON?
1166  2507  F0 1A           BEQ     SYMTRT                  ;YES
1167  2509  C9 20           CMP#    '                       ;SPACE?
1168  250B  F0 16           BEQ     SYMTRT                  ;YES
1169  250D  C9 09           CMP#    TABCHR                  ;TABULATION CHARACTER?
1170  250F  F0 12           BEQ     SYMTRT                  ;YES
1171  2511  C9 3D           CMP#    '=                      ;EQUALS (FOR '.DEF' PSUEDO-OP)
1172  2513  F0 0E           BEQ     SYMTRT                  ;YES
1173  2515  C9 2B           CMP#    '+                      ;ADDITION OPERATOR?
1174  2517  F0 0A           BEQ     SYMTRT                  ;YES
1175  2519  C9 2D           CMP#    '-                      ;SUBTRACTION OPERATOR?
1176  251B  F0 06           BEQ     SYMTRT                  ;YES
1177  251D  C9 0D           CMP#    EOLCHR                  ;END OF LINE DELIMITER?
1178  251F  F0 02           BEQ     SYMTRT                  ;YES
1179  2521  C9 3B           CMP#    ';                      ;COMMENT?
1180  2523  60      SYMTRT: RTS                             ;RETURN
1181
```

```
1182                                .PAGE                   ;'SYMBOL TABLE SEARCH ROUTINE'
1183
1184                                ;THIS ROUTINE WILL SEARCH THE USER-DEFINED SYMBOL TABLE
1185                                ;TRYING TO FIND A SYMBOL OF THE SAME NAME AS THE ONE
1186                                ;STORED IN THE 'SYMTMP' STRING.  IT RETURNS WITH:
1187                                ;
1188                                ;N BIT = 0 IF NOT FOUND
1189                                ;N BIT = 1 IF IT WAS FOUND, WITH 'SYMADR' INDEXED BY 'Y' POINTING TO
1190                                ;THE STATUS BYTE.  THE STATUS BYTE IS ALSO IN THE ACCUMULATOR.
1191                                ;ENTER AT 'OPTSRC' TO SEARCH THE OPCODE TABLE.
1192                                ;ENTER AT 'USYSRC' TO SEARCH THE USER-DEFINED TABLE.
1193
1194  2524  A9 E3   OPTSRC: LDA#    OPCTAB          ;SET UP BASE ADDR POINTER TO START
1195  2526  85 48          STA     SYMADR          ;OF OPCODE SYMBOL TABLE
1196  2528  A9 2C          LDA#    >OPCTAB         ;HIGH ORDER ALSO
1197  252A  D0 06          BNE     USYSR1          ;UNCONDITIONALLY ENTER MAIN CODE
1198  252C  A9 B9   USYSRC: LDA#    SYMTAB          ;SET UP BASE ADDR POINTER TO START
1199  252E  85 48          STA     SYMADR          ;OF USER-DEFINED SYMBOL TABLE
1200  2530  A9 2F          LDA#    >SYMTAB         ;HIGH ORDER ALSO
1201  2532  85 49   USYSR1: STA     SYMADR+1
1202  2534  A0 00   UNEXT0: LDY#    0               ;SET OFFSET POINTER TO SYMBOL TABLE TO ZERO
1203  2536  A2 00   UNEXT1: LDX#    0               ;SET OFFSET TO TEMP NAME TO ZERO,
1204  2538  F0 02          BEQ     USLOOP          ;ENTER MAIN LOOP
1205  253A  E8      UCHROK: INX                     ;POINT TO NEXT CHAR IN TEMP STRING
1206  253B  C8              INY                     ;POINT TO NEXT CHAR IN SYMBOL TABLE NAME
1207  253C  B5 50   USLOOP: LDAX    SYMTMP          ;GET CHAR FROM TEMP STORAGE
1208  253E  F0 1D          BEQ     UCHKDN          ;BRANCH IF AT END OF TEMP STRING
1209  2540  D1 48          CMP@Y   SYMADR          ;COMPARE TO SYMBOL TABLE ENTRY CHARACTER
1210  2542  F0 F6          BEQ     UCHROK          ;BRANCH IF THE ARE THE SAME (SO FAR)
1211  2544  B1 48   USKIPN: LDA@Y   SYMADR          ;AT END OF SYMBOL TABLE NAME YET?
1212  2546  30 03          BMI     UENDNM          ;BRANCH IF YES
1213  2548  C8              INY                     ;NO, POINT TO NEXT CHAR IN THE NAME
1214  2549  D0 F9          BNE     USKIPN          ;UNCONDITIONALLY EAT THE NAME
1215  254B  C9 FF   UENDNM: CMP#    SYMEOT          ;ARE WE AT THE END OF THE TABLE?
1216  254D  F0 12          BEQ     UNOTF           ;YES, THEN WE DID NOT FIND THE SYMBOL
1217  254F  29 40          AND#    SYMSIZ          ;NO, BUT HOW MANY DATA BYTES FOLLOW?
1218  2551  D0 01          BNE     USHORT          ;BRANCH IF ONLY ONE BYTE
1219  2553  C8              INY                     ;TWO BYTE VALUE
1220  2554  C8      USHORT: INY                     ;SKIP OVER THE SYMBOL VALUE BYTES
1221  2555  C8              INY                     ;POINT TO START OF NEXT SYMBOL'S NAME
1222  2556  10 DE          BPL     UNEXT1          ;GO TRY NEXT TABLE ENTRY
1223  2558  20 AF25        JSR     UPDSYM          ;UPDATE BASE ADDR POINTER > 127.
1224  255B  D0 D7          BNE     UNEXT0          ;UNCONDITIONALLY TRY NEXT ENTRY
1225  255D  B1 48   UCHKDN: LDA@Y   SYMADR          ;GET BYTE FROM SYMBOL TABLE ENTRY
1226  255F  10 E3          BPL     USKIPN          ;BRANCH IF THE NAMES ARE NOT EQUAL
1227                                ;OTHERWISE RETURN SINCE THEY ARE EQUAL
1228                                ;N BIT = 1
1229  2561  60      UNOTF:  RTS
```

```
1230                                .PAGE                   ;'ROUTINES TO MANIPULATE THE TEXT BUFFER'
1231
1232
1233
1234                                ;ROUTINE TO GET A CHARACTER FROM EITHER THE TEXT BUFFER OR THE
1235                                ;DEVICE, DEPENDING ON THE SETTING OF 'INPSW'.
1236
1237
1238  2562  A5 24   GETC:  LDA     INPSW           ;WHERE DOES INPUT COME FROM?
1239  2564  F0 06           BEQ     GETCC           ;FROM TEXT AREA
1240  2566  20 7429         JSR     INPB            ;FROM DEVICE
1241  2569  85 32           STA     CHAR            ;SAVE FOR LATER ALSO
1242  256B  60              RTS                     ;AND RETURN
1243  256C  A4 38   GETCC: LDY     TEXTP           ;GET POINTER TO TEXT AREA
1244  256E  E6 38           INC     TEXTP           ;POINT TO NEXT CHAR FOR NEXT TIME
1245  2570  B1 36           LDA@Y   TXTADR          ;GET THE CHARACTER FROM THE TEXT AREA
1246  2572  85 32           STA     CHAR            ;SAVE FOR LATER
1247  2574  60              RTS                     ;AND RETURN
1248
1249
1250                                ;ROUTINE TO PACK A CHARACTER INTO THE TEXT AREA WITH
1251                                ;DELETION PROCESSING.
1252
1253
1254  2575  A5 32   PACKC:  LDA     CHAR            ;GET THE CHARACTER
1255  2577  A4 38   PACKC1: LDY     TEXTP           ;GET CURRENT POINTER INTO TEXT AREA
1256  2579  C9 7F           CMP#    DELCHR          ;IS THIS THE DELETION CHARACTER?
1257  257B  F0 0A           BEQ     DEL1            ;YES, THEN DELETE ONE CHARACTER BACK
1258  257D  91 36           STA@Y   TXTADR          ;NO, STORE THIS CHAR INTO TEXT BUFFER
1259  257F  C8      PCKDEL: INY                     ;POINT TO NEXT CHAR POSITION
1260  2580  C0 7F           CPY#    LINEL           ;END OF BUFFER?
1261  2582  10 02           BPL     RTS1            ;YES, THEN DON'T UPDATE POINTER AS NO MORE ROOM
1262  2584  84 38   PCKRET: STY     TEXTP           ;STORE UPDATED POINTER
1263  2586  60      RTS1:   RTS                     ;AND RETURN
1264                        !
1265  2587  88      DEL1:   DEY                     ;POINT BACK ONE CHARACTER
1266  2588  10 FA           BPL     PCKRET          ;BRANCH IF STILL SOMEWHERE IN BUFFER
1267  258A  30 F3           BMI     PCKDEL          ;DON'T DELETE BEYOND FIRST OF BUFFER
1268
```

```
1269                            .PAGE                   ;'MORE TEXT AND CHARACTER MANIPULATION ROUTI
1270
1271
1272                            ;ROUTINE TO INITIALIZE THE TEXT POINTERS TO THE BEGINNING
1273                            ;OF THE TEXT BUFFER.
1274
1275
1276  258C  A5 39   TXTINI: LDA     LINADR          ;COPY THE BASE ADDR OF THE BUFFER
1277  258E  85 36           STA     TXTADR          ;INTO THE POINTER BASE ADDR
1278  2590  A5 3A           LDA     LINADR+1
1279  2592  85 37           STA     TXTADR+1
1280  2594  A9 00   TPINI:  LDA#    0               ;AND RESET OFFSET TO START OF BUFFER
1281  2596  85 38           STA     TEXTP
1282  2598  60             RTS                     ;AND RETURN
1283
1284                            ;ROUTINE TO IGNORE BLANKS
1285
1286  2599  20 6225 GSPNOR: JSR     GETC            ;GET NEXT INPUT CHARACTER
1287  259C  A5 32   SPNOR:  LDA     CHAR            ;GET CURRENT INPUT CHARACTER
1288  259E  C9 20   SPNOR1: CMP#    ' '             ;BLANK?
1289  25A0  F0 F7           BEQ     GSPNOR          ;YES, IGNORE, GET NEXT CHAR
1290  25A2  60              RTS                     ;NO, THEN RETURN
1291
1292                            ;ROUTINE TO TEST IF THE CHARACTER IS A NUMBER
1293
1294  25A3  20 9925 TESTNS: JSR     GSPNOR          ;GET NEXT NON-BLANK
1295  25A6  A5 32   TESTN:  LDA     CHAR            ;GET THE CHARACTER
1296  25A8  48      TESTN1: PHA                     ;SAVE CHAR SINCE WE DESTROY IT
1297  25A9  49 30           EOR#    $30             ;CONVERT TO A BCD NUMBER (IF 0-9)
1298  25AB  C9 0A           CMP#    $0A             ;SET 'C' BIT = 0 IF A NUMBER
1299  25AD  68              PLA                     ;RESTORE THE CHARACTER
1300  25AE  60              RTS                     ;RETURN
1301
1302                            ;ROUTINE TO UPDATE BASE ADDR OF POINTER TO SYMBOL TABLE.
1303
1304  25AF  18      UPDSYM: CLC                     ;UNLESS OFFSET IS > 127
1305  25B0  98              TYA                     ;GET OFFSET
1306  25B1  65 48           ADC     SYMADR          ;ADD OFFSET TO BASE ADDR
1307  25B3  85 48           STA     SYMADR          ;TO GET NEW BASE ADDR
1308  25B5  A5 49           LDA     SYMADR+1
1309  25B7  69 00           ADC#    0               ;HIGH ORDER ALSO
1310  25B9  85 49           STA     SYMADR+1
1311  25BB  60              RTS                     ;AND RETURN
```

```
1312                            .PAGE                   ;'MORE TEXT MANIPULATION ROUTINES'
1313
1314
1315
1316                            ;ROUTINE TO INPUT A LINE FROM THE INPUT DEVICE AND
1317                            ;STORE IT IN THE TEXT BUFFER.
1318                            ;ENTRY POINT IS 'GETLIN'.
1319
1320
1321  25BC  A5 27   GOTEOF: LDA     DEVO              ;SAVE CURRENT OUTPUT DEVICE NUMBER
1322  25BE  48              PHA
1323  25BF  20 042A         JSR     SETCON            ;SET INPUT AND OUTPUT TO CONSOLE DEVICE
1324  25C2  A2 6D           LDX#    MEOF              ;TELL USER WE ENCOUNTERED EOF
1325  25C4  20 EF29         JSR     OUTSTR  <- Patch here for return to monitor on ↑Z
1326  25C7  20 7C21         JSR     NEWSRC            ;GET A NEW SOURCE DEVICE SPEC
1327  25CA  68              PLA                       ;RESTORE OUTPUT DEVICE
1328  25CB  85 27           STA     DEVO
1329  25CD  20 8C25 GETLIN: JSR     TXTINI            ;INIT POINTERS TO START OF BUFFER
1330  25D0  E6 24   GETL1:  INC     INPSW             ;INDICATE INPUT IS TO COME FROM DEVICE
1331  25D2  20 6225         JSR     GETC              ;GET A CHARACTER
1332  25D5  C6 24           DEC     INPSW             ;SET BACK TO WHAT IT WAS BEFORE
1333  25D7  C9 0A           CMP#    LFCHR             ;LINE FEED?
1334  25D9  F0 F5           BEQ     GETL1             ;YES, THEN IGNORE IT
1335  25DB  C9 1A           CMP#    EOFCHR            ;END OF FILE?
1336  25DD  F0 DD           BEQ     GOTEOF            ;YES, GO HANDLE
1337  25DF  20 7725         JSR     PACKC1            ;PACK THE CHARACTER INTO THE BUFFER
1338  25E2  C9 0D           CMP#    EOLCHR            ;END OF LINE DELIMITER?
1339  25E4  D0 EA           BNE     GETL1             ;BRANCH IF NOT, GET ANOTHER CHARACTER
1340  25E6  A5 26           LDA     DEVI              ;GET NUMBER OF CURRENT INPUT DEVICE
1341  25E8  C9 00           CMP#    CONIDV            ;ARE WE INPUTTING FROM THE CONSOLE?
1342  25EA  D0 05           BNE     GETLC             ;BRANCH IF NOT, PRESS ON
1343  25EC  A9 0A           LDA#    LFCHR             ;YES, THEN FOLLOW THE CARRIAGE RETURN
1344  25EE  20 9229         JSR     OUTB              ;WITH A LINE FEED
1345  25F1  20 8C25 GETLC:  JSR     TXTINI            ;RESET POINTERS TO START OF BUFFER
1346  25F4  4C 9925         JMP     GSPNOR            ;* PJMP * POINT TO FIRST NON-BLANK THEN RETURN
1347
```



```
1348                            .PAGE                   ;'ROUTINE TO EXTRACT AN ATOM FROM EXPRESSION'
1349
1350
1351
1352  25F7  20 1C29 ATOM:   JSR     ZERNUM            ;ASSUME A VALUE OF ZERO
1353  25FA  20 9C25         JSR     SPNOR             ;IGNORE LEADING BLANKS
1354  25FD  C9 24           CMP#    '$                ;HEX CONSTANT?
1355  25FF  F0 3B           BEQ     AIHEX             ;BRANCH IF YES
1356  2601  C9 25           CMP#    '%                ;OCTAL CONSTANT?
1357  2603  F0 3D           BEQ     AIOCT             ;BRANCH IF YES
1358  2605  C9 27           CMP#    ''                ;QUOTED ASCII CHARACTER?
1359  2607  F0 3F           BEQ     AIASC             ;BRANCH IF YES
1360  2609  C9 3E           CMP#    '>                ;ARE WE TO KEEP HIGH ORDER ONLY?
1361  260B  F0 68           BEQ     AIHBYT            ;BRANCH IF YES
1362  260D  C9 3C           CMP#    '<                ;ARE WE TO KEEP LOW ORDER ONLY?
1363  260F  F0 59           BEQ     AILBYT            ;BRANCH IF YES
1364  2611  20 A825         JSR     TESTN1            ;IS IT A NUMBER?
1365  2614  B0 03           BCS     ASYM              ;BRANCH IF NOT
1366  2616  4C D128         JMP     INPDEC            ;* PJMP * YES, THEN GET A DECIMAL CONSTANT
1367  2619  20 EA24 ASYM:   JSR     GETSYM            ;TRY TO EXTRACT A SYMBOL
1368  261C  E0 00           CPX#    0                 ;DID WE GET ANYTHING?
1369  261E  F0 64           BEQ     ATOMRT            ;BRANCH IF NOT
1370  2620  20 2C25         JSR     USYSRC            ;TRY TO FIND THE SYMBOL IN SYMBOL TABLE
1371  2623  10 2F           BPL     ASYUDF            ;BRANCH IF IT'S NOT THERE
1372  2625  A6 3B           LDX     FLAGS             ;IT'S THERE, PASS 2?
1373  2627  10 10           BPL     ASYMC             ;NO, JUST USE IT'S VALUE (EVEN IF UNDEFINED)
1374  2629  24 4E           BIT     BMMDF             ;YES, IS SYMBOL MULTIPLY-DEFINED?
1375  262B  D0 09           BNE     AMERR             ;BRANCH IF YES
1376  262D  24 4D           BIT     BMUDF             ;IS IT UNDEFINED?
1377  262F  F0 08           BEQ     ASYMC             ;NO, THEN USE IT'S VALUE
1378  2631  20 C624         JSR     SUERR             ;UNDEFINED AT PASS 2, THEN FLAG ERROR
1379  2634  D0 03           BNE     ASYMC             ;BUT USE THE CRUD VALUE ANYWAY
1380  2636  20 DA24 AMERR:  JSR     SMERR             ;MULTIPLY-DEFINED, FLAG ERROR
1381                                                  ;AND USE CRUD VALUE
```

```
*** ASM65 ***  (V1C)  13-JUN-77

                        .PAGE
                        .LOC  $266A       ;Location of AILBYT in original
                        .DEF  GEXPR1=$2291        ;Subr in oris. ASM65 - see below
                        .DEF  NUMBH=$2C   ;Atom value save area
                        .DEF  NUMBL=$2D   ;  (High and low bytes)
                ;
266A 48         AIHLBT: PHA               ;Save the operator
266B 20 91 22           JSR   GEXPR1      ;Go skip < or >, get atom, check for followins
                                          ; + or -, and add terms till not + or -
266E 68                 PLA               ;Get back original < or >
266F C9 3E              CMP#  '>          ;Assume low byte if not high byte
2671 D0 04              BNE   AILBYT
2673 A5 2C              LDA   NUMBH       ;Explicit high byte: fetch it
2675 85 2D              STA   NUMBL       ;Put it in low byte
2677 A9 00      AILBYT: LDA#  0           ;Low is OK; now zero high
2679 85 2C              STA   NUMBH
267B A5 2D              LDA   NUMBL       ;Antibussing: exit here same as original,
267D 60                 RTS               ;  with NUMBL in accumulator.
                ; Don't forget:  $260C: $5D <-- $68
                        .END

  267E .
  2291 GEXPR1
  002C NUMBH
  002D NUMBL
  266A AIHLBT
  2677 AILBYT

NO ERRORS DETECTED
```

```
AILBT:  PHA
        JSR     EXPR1       ;SAVE THE OPERATOR
                            ;GO SKIP <OR>, GET ATOM,
                            ; CHECK FOR + OR - AND
                            ;ADD TERMS TILL NOT + OR -
        PLA                 ;GET OPERATOR BACK
        CMP#    '>          ;ASSUME LOW BYTE IF NOT
        BNE     AILBYT      ;  HIGH BYTE
        LDA     NUMBH       ;EXPLICIT HIGH BYTE:
        JMP     AIBYTC      ;  FETCH IT
AILBYT: LDA     NUMBL       ;ASSUMED LOW: FETCH IT
AIBYTC: PHA                 ;SAVE HIGH OR LOW
        JSR     ZERNUM
        PLA
        STA     NUMBL
        RTS
```

```
        BNE     AILBYT
        LDA     NUMBH
        STA     NUMBL
AILBYT: LDA#    Ø
        STA     NUMBH
        RDA     NUMBL
```

1B
75
18

42-381 50 SHEETS 5 SQUARE
42-382 100 SHEETS 5 SQUARE
42-389 200 SHEETS 5 SQUARE
Made in U.S.A.
NATIONAL

```
*** ASM65 ***  (VIC)  13-JUN-77
```

AFTER PATCH

```
                        .PAGE
                    ;Here is a little demonstration program for
                    ;the problems with "<" and ">" - - -
                    ;T. Bruhns   - - -   23 Jan 1982

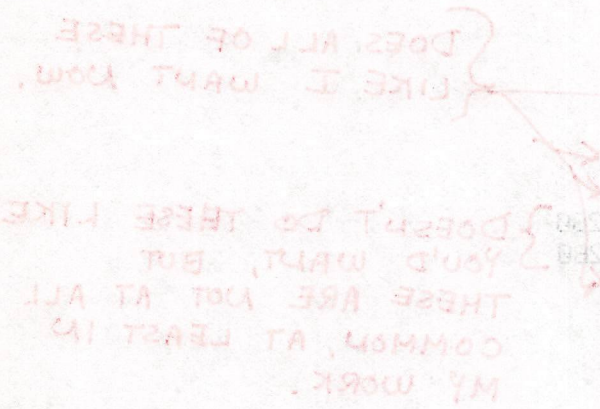
                        .LOC  $200
                        .DEF  Atom.1=<$100-$200
                        .DEF  Atom=$3800
0200 A9 37              LDA#  >Atom-4
0202 A9 FC              LDA#  <Atom-4
                        .DEF  Atom.2=>Atom+$300+$200-$100
                        .DEF  Atom.3=>$490+$5A0+$3C0+$2B0
                        .DEF  Atom.4=>$490+>$5A0+>$3C0+>$2B0
                        .DEF  Atom.5=<$490+<$5A0+<$3C0+<$2B0
                        .DEF  Atom.6=<$490+$5A0+$3C0+$2B0
                        .DEF  Atom.7=Atom.4+Atom.5
                        .DEF  Atom.8=$490+$5A0+$3C0+$2B0
                        .DEF  Atom.9=Atom.3+Atom.6
                        .END

  0204 .
  0000 Atom.1
  3800 Atom
  003C Atom.2
  0010 Atom.3
  0004 Atom.4
  00A0 Atom.5
  00A0 Atom.6
  00A4 Atom.7
  10A0 Atom.8
  00B0 Atom.9


NO ERRORS DETECTED
```

DOES ALL OF THESE LIKE I WANT NOW.

DOESN'T DO THESE LIKE YOU'D WANT, BUT THESE ARE NOT AT ALL COMMON, AT LEAST IN MY WORK.



*** ASM65 *** (V1C) 13-JUN-77

BEFORE PATCH

```
                        .PAGE
                    ;Here is a little demonstration program for
                    ;the problems with "<" and ">" - - -
                    ;T. Bruhns   - - -   23 Jan 1982

                        .LOC  $200
                        .DEF  Atom.1=<$100-$200
                        .DEF  Atom=$3800
0200 A9 34              LDA#  >Atom-4
0202 A9 FC              LDA#  <Atom-4
                        .DEF  Atom.2=>Atom+$300+$200-$100
                        .DEF  Atom.3=>$490+$5A0+$3C0+$2B0
                        .DEF  Atom.4=>$490+>$5A0+>$3C0+>$2B0
                        .DEF  Atom.5=<$490+<$5A0+<$3C0+<$2B0
                        .DEF  Atom.6=<$490+$5A0+$3C0+$2B0
                        .DEF  Atom.7=Atom.4+Atom.5
                        .DEF  Atom.8=$490+$5A0+$3C0+$2B0
                        .DEF  Atom.9=Atom.3+Atom.6
                        .END

  0204 .
  FE00 Atom.1
  3800 Atom
  0438 Atom.2
  0C14 Atom.3
  000E Atom.4
  02A0 Atom.5
  0CA0 Atom.6
  02AE Atom.7
  10A0 Atom.8
  18B4 Atom.9


NO ERRORS DETECTED
```

```
1382                            ,PAGE                   ;'ROUTINE TO EXTRACT AN ATOM FROM EXPRESSION'
1383
1384
1385
1386  2639   4C 4B24 ASYMC:  JMP     GETSV           ;* PJMP * YES, GET IT'S VALUE AND RETURN
1387                    ;
1388  263C   20 6225 AIHEX:  JSR     GETC            ;MOVE PAST '$'
1389  263F   4C 3129         JMP     INPHEX          ;* PJMP * AND PICK UP THE HEX CONSTANT
1390                    ;
1391  2642   20 6225 AIOCT:  JSR     GETC            ;MOVE PAST '%'
1392  2645   4C 5529         JMP     INPOCT          ;* PJMP * AND PICK UP THE OCTAL CONSTANT
1393                    ;
1394  2648   20 6225 AIASC:  JSR     GETC            ;MOVE PAST THE "'"
1395  264B   C9 0D           CMP#    EOLCHR          ;IS THE CHAR END-OF-LINE?
1396  264D   F0 35           BEQ     ATOMRT          ;BRANCH IF YES
1397  264F   85 2D           STA     NUMBL           ;SAVE IN LOW ORDER VALUE
1398  2651   4C 6225         JMP     GETC            ;* PJMP * MOVE PAST CHAR AND RETURN
1399                    ;
1400  2654   A6 3B   ASYUDF: LDX     FLAGS           ;PASS 2?
1401  2656   30 0F           BMI     AIERR           ;YES, THEN WHY IS THIS SYMBOL NOT IN TABLE?
1402  2658   A9 80           LDA#    $80             ;SET CRUD VALUE ($8000) FOR LACK OF ANYTHING
1403  265A   85 2C           STA     NUMBH           ;ELSE.
1404  265C   A9 00           LDA#    0
1405  265E   85 2D           STA     NUMBL
1406  2660   A9 80           LDA#    USYM            ;GET SKELETON STATUS BYTE
1407  2662   09 08           ORA#    SYMUDF          ;INDICATE THIS SYMBOL IS UNDEFINED
1408  2664   4C 7824         JMP     ADDSYM          ;* PJMP * ADD SYMBOL TO TABLE AND RETURN
1409  2667   4C D224 AIERR:  JMP     SIERR           ;* PJMP * INDICATE INTERNAL ERROR AND RETURN
1410                    ;
1411  266A   20 6225 AILBYT: JSR     GETC            ;MOVE PAST '<'
1412  266D   20 F725         JSR     ATOM            ;CALL 'ATOM' AGAIN TO GET NEXT ATOM
1413  2670   A5 2D           LDA     NUMBL           ;GET LOW ORDER OF VALUE
1414  2672   4C 7026         JMP     AIBYTC          ;AND ENTER COMMON CODE
1415  2675   20 6225 AIHBYT: JSR     GETC            ;MOVE PAST '>'
1416  2678   20 F725         JSR     ATOM            ;CALL 'ATOM' AGAIN TO GET NEXT ATOM
1417  267B   A5 2C           LDA     NUMBH           ;GET HIGH ORDER OF VALUE
1418  267D   48      AIBYTC: PHA                     ;SAVE VALUE
1419  267E   20 1C29         JSR     ZERNUM          ;ZERO VALUE
1420  2681   68              PLA                     ;GET VALUE BACK
1421  2682   85 2D           STA     NUMBL           ;SAVE AS LOW ORDER
1422  2684   60      ATOMRT: RTS                     ;AND RETURN
```

REF's

AIHBYT: 1361, 1415 (OK)

AILBYT: 1363, 1411 (OK)

AIBYTC: 1414, 1418 (OK)

ATOMRT: 1369, 1396, 1422 (SAVE!)

```
1423                            .PAGE                   ;'SYMBOL TABLE LISTING ROUTINES'
1424
1425                    FROM LINE 422 ONLY?
1426                            ;THIS ROUTINE SCANS THE SYMBOL TABLE AND LISTS OUT ANY
1427                            ;SYMBOLS THAT ARE EITHER 'UNDEFINED' OR 'MULTIPLY-DEFINED'.
1428
1429  2685  A5 38   DMPSYB:  LDA     FLAGS           ;INDICATE WE DUMP ONLY THE 'BAD' ONES
1430  2687  09 20            ORA#    ST.DBS
1431  2689  85 38            STA     FLAGS
1432  268B  A9 00   DMPSC:   LDA#    CONODV          ;MAKE CONSOLE DEVICE THE LISTING DEVICE
1433  268D  85 23            STA     TLDEV
1434  268F  A9 7F            LDA#    127             ;SET WIDTH TO MAX SO NO TRUNCATION POSSIBLE
1435  2691  85 43            STA     LWIDTH
1436  2693  20 B426          JSR     DMPSYM          ;NOW DUMP THEM
1437  2696  A5 38            LDA     FLAGS           ;NOW CLEAR THE FLAG
1438  2698  29 DF            AND#    ST.DBS
1439  269A  85 38            STA     FLAGS
1440  269C  60      DMPRET:  RTS                     ;AND RETURN
1441
1442                            ;THIS ROUTINE DUMPS THE ENTIRE SYMBOL TABLE.
1443
1444  269D  A5 22   DMPSCL:  LDA     BINDEV          ;GET BINARY DEVICE NUMBER
1445  269F  C9 00            CMP#    CONODV          ;IS IT THE CONSOLE DEVICE
1446  26A1  F0 F9            BEQ     DMPRET          ;DON'T LIST SYMBOL TABLE TO BINARY DEVICE
1447  26A3  A5 43            LDA     LWIDTH          ;SAVE THE WIDTH OF LISTING DEVICE
1448  26A5  48               PHA
1449  26A6  20 8B26          JSR     DMPSC           ;NOW DUMP TO CONSOLE
1450  26A9  68               PLA                     ;RESTORE WIDTH OF LISTING DEVICE
1451  26AA  85 43            STA     LWIDTH
1452  26AC  A5 21            LDA     LSTDEV          ;GET LISTING DEVICE NUMBER
1453  26AE  C9 00            CMP#    CONODV          ;IS IT THE CONSOLE DEVICE?
1454  26B0  F0 EA            BEQ     DMPRET          ;BRANCH IF YES, THEN DON'T LIST IT AGAIN
1455  26B2  85 23            STA     TLDEV           ;SET LISTING DEVICE AS CURRENT LISTER
1456                                                  ;* PFALL * INTO DUMP ROUTINE AND RETURN
```

```
1457                            .PAGE                   ;'SYMBOL TABLE LISTING ROUTINES'
1458
1459
1460
1461  26B4  A9 B9   DMPSYM: LDA#    SYMTAB              ;SETUP POINTER TO USER SYMBOL TABLE
1462  26B6  85 48           STA     SYMADR
1463  26B8  A9 2F           LDA#    >SYMTAB
1464  26BA  85 49           STA     SYMADR+1
1465  26BC  20 6D27         JSR     LCRLF               ;PRECEED LISTING WITH A BLANK LINE
1466  26BF  A0 00   DNEXT0: LDY#    0                   ;RESET OFFSET TO ZERO
1467  26C1  A2 00   DNEXT1: LDX#    0                   ;RESET POINTER TO 'SYMTMP' TO ZERO
1468  26C3  F0 06           BEQ     DSLOOP              ;AND ENTER LOOP
1469  26C5  E8      DCHROK: INX                         ;POINT TO NEXT CHAR IN 'SYMTMP'
1470  26C6  C8              INY                         ;POINT TO NEXT BYTE IN SYMBOL TABLE
1471  26C7  A9 00           LDA#    0                   ;PLACE A ZERO AT END OF 'SYMTMP'
1472  26C9  95 50           STAX    SYMTMP              ;IN CASE THIS IS THE LAST CHAR
1473  26CB  B1 48   DSLOOP: LDA@Y   SYMADR              ;GET A BYTE FROM SYMBOL TABLE
1474  26CD  30 04           BMI     DCHKU               ;BRANCH IF NEG, END OF NAME
1475  26CF  95 50           STAX    SYMTMP              ;STORE AWAY NEXT CHAR IN SYMBOL'S NAME
1476  26D1  10 F2           BPL     DCHROK              ;AND POINT TO NEXT BYTE
1477  26D3  C9 FF   DCHKU:  CMP#    SYMEOT              ;IS THIS THE END OF THE USER SYMBOL TABLE?
1478  26D5  F0 56           BEQ     DMPEND              ;BRANCH IF YES,
1479  26D7  24 4D           BIT     BMUDF               ;IS THE SYMBOL UNDEFINED?
1480  26D9  D0 1E           BNE     DSUDF               ;BRANCH IF UNDEFINED, LIST IT,
1481  26DB  24 4E           BIT     BMMDF               ;IS THE SYMBOL 'MULTIPLY-DEFINED'
1482  26DD  D0 1E           BNE     DSMDF               ;BRANCH IS IT IS,  LIST IT,
1483  26DF  A5 3B           LDA     FLAGS               ;GET FLAGS
1484  26E1  29 20           AND#    STSOBS              ;DO WE LIST ONLY THE 'BAD' ONES
1485  26E3  D0 04           BNE     DBONLY              ;BRANCH IF YES
1486  26E5  A9 20           LDA#    ' '                 ;LIST THEM ALL,  SPACE FOR ALIGNMENT
1487  26E7  D0 16           BNE     DSCNT               ;AND UNCONDITIONALLY LIST THIS ONE,
1488  26E9  B1 48   DBONLY: LDA@Y   SYMADR              ;GET SYMBOL STATUS BACK AGAIN
1489  26EB  24 4B           BIT     BMSIZ               ;DON'T LIST THIS ONE
1490  26ED  D0 01           BNE     DSHORT              ;BRANCH IF VALUE IS STORED AS ONLY ONE BYTE
1491  26EF  C8              INY                         ;SKIP OVER SYMBOL'S VALUE
1492  26F0  C8      DSHORT: INY
1493  26F1  C8      DSHC:   INY                         ;POINT TO NEXT SYMBOL'S NAME
1494  26F2  10 CD           BPL     DNEXT1              ;BRANCH IF POINTER STILL < 128
1495  26F4  20 AF25         JSR     UPDSYM              ;> 128,  UPDATE BASE ADDR,
1496  26F7  D0 C6           BNE     DNEXT0              ;AND GO LOOK AT NEXT SYMBOL,
```



```
1497                            .PAGE                   ;'SYMBOL TABLE LISTING ROUTINES'
1498
1499
1500
1501  26F9  A9 55   DSUDF:  LDA#    'U                  ;PRECEED UNDEFINED SYMBOLS WITH 'U'
1502  26FB  D0 02           BNE     DSCNT
1503  26FD  A9 4D   DSMDF:  LDA#    'M                  ;PRECEED 'MULTIPLY-DEFINED' SYMBOLS WITH 'M'
1504  26FF  20 6927 DSCNT:  JSR     LOUTC               ;OUTPUT INDICATOR WHY SYMBOL IS BAD
1505  2702  20 6727         JSR     LSPACE              ;SPACE FOR SEPARATION
1506  2705  A2 00           LDX#    0                   ;ASSUME A HIGH ORDER OF ZERO
1507  2707  B1 48           LDA@Y   SYMADR              ;GET STATUS BYTE FROM SYMBOL TABLE
1508  2709  24 4B           BIT     BMSIZ               ;IS VALUE ONLY STORED AS ONE BYTE?
1509  270B  D0 04           BNE     DSONE               ;BRANCH IF YES
1510  270D  C8              INY                         ;POINT TO HIGH ORDER VALUE
1511  270E  B1 48           LDA@Y   SYMADR              ;GET HIGH ORDER VALUE
1512  2710  AA              TAX                         ;INTO X REGISTER
1513  2711  86 2D   DSONE:  STX     NUMBL               ;STORE HIGH ORDER VALUE
1514  2713  20 4E27         JSR     LBYTO               ;OUTPUT IT
1515  2716  C8              INY                         ;POINT TO LOW ORDER VALUE
1516  2717  B1 48           LDA@Y   SYMADR              ;GET IT
1517  2719  85 2D           STA     NUMBL
1518  271B  20 4E27         JSR     LBYTO               ;OUTPUT IT
1519  271E  98              TYA                         ;SAVE POINTER TO SYMBOL TABLE
1520  271F  48              PHA
1521  2720  A2 75           LDX#    MSYMT               ;NOW OUTPUT THE SYMBOL NAME
1522  2722  20 3327         JSR     LOUTST
1523  2725  68              PLA                         ;RESTORE POINTER TO SYMBOL TABLE
1524  2726  A8              TAY
1525  2727  20 6D27         JSR     LCRLF               ;ADVANCE A LINE
1526  272A  4C F126         JMP     DSHC                ;AND POINT TO NEXT SYMBOL
1527  272D  4C 6D27 DMPEND: JMP     LCRLF               ;* PJMP * BLANK LINE FOR LOOKS, THEN RETURN
```

```
1528                                    .PAGE                   ;'I-O ROUTINES FOR LISTING DEVICE'
1529
1530                                    ;ROUTINE TO OUTPUT A STRING TO LISTING DEVICE
1531
1532  2730  A5 27   LOUTST: LDA     DEVO                    ;GET CURRENT OUTPUT DEVICE
1533  2732  48              PHA                             ;SAVE IT
1534  2733  A5 23           LDA     TLDEV                   ;SET TEMPORARY LISTING DEVICE
1535  2735  85 27           STA     DEVO
1536  2737  20 EF29         JSR     OUTSTR                  ;CALL NORMAL ROUTINE TO OUTPUT A STRING
1537  273A  68              PLA                             ;RESTORE DEVICE NUMBER
1538  273B  85 27           STA     DEVO
1539  273D  60              RTS                             ;AND RETURN
1540
1541                                    ;ROUTINE TO FIND THE NEXT TAB STOP IN TAB-STOP TABLE
1542
1543  273E  A5 44   FNDTAB: LDA     HPOS                    ;GET OUR CURRENT HORIZONTAL POSITION
1544  2740  A2 FF           LDX#    $FF                     ;INIT POINTER TO TAB-STOP TABLE
1545  2742  E8       FNDLOP: INX                             ;POINT TO NEXT ENTRY IN TAB-STOP TABLE
1546  2743  DD F12B         CMPX    TABS                    ;IS THE ENTRY BEYOND WHERE WE ARE?
1547  2746  10 FA           BPL     FNDLOP                  ;BRANCH IF NOT
1548  2748  BD F12B         LDAX    TABS                    ;YES, THEN GET IT'S POSITION
1549  274B  85 45           STA     LTEMP                   ;SAVE IT FOR LATER
1550  274D  60              RTS                             ;AND RETURN
1551
1552                                    ;ROUTINE TO OUTPUT THE BYTE IN 'NUMBL'.
1553
1554  274E  A5 2D   LBYTO:  LDA     NUMBL                   ;GET THE BYTE
1555  2750  4A              LSRA                            ;EXTRACT HIGH ORDER
1556  2751  4A              LSRA
1557  2752  4A              LSRA
1558  2753  4A              LSRA
1559  2754  20 5B27         JSR     LHEXO                   ;OUTPUT THE HEX DIGIT
1560  2757  A5 2D           LDA     NUMBL                   ;GET BYTE AGAIN
1561  2759  29 0F           AND#    %17                     ;KEEP ONLY LOW ORDER
1562                                                        ;* PFALL * OUTPUT HEX DIGIT AND RETURN
1563
1564                                    ;ROUTINE TO OUTPUT A HEX DIGIT
1565
1566  275B  AA       LHEXO: TAX                             ;MOVE DIGIT INTO X REGISTER
1567  275C  BD CB2B         LDAX    HEXTAB                  ;CONVERT TO ASCII
1568  275F  D0 08           BNE     LOUTC                   ;* PBNE * OUTPUT AND RETURN
1569
1570                                    ;ROUTINE TO OUTPUT SPACES ON LISTING DEVICE
1571
1572  2761  20 6427 LSP4:   JSR     LSP2                    ;OUTPUT FOUR SPACES
1573  2764  20 6727 LSP2:   JSR     LSPACE                  ;OUTPUT TWO SPACES
1574  2767  A9 20   LSPACE: LDA#    '                       ;GET A SPACE.
1575                                                        ;* PFALL * OUTPUT IT AND RETURN
```

```
1576                            .PAGE                   ;'I-O ROUTINES TO HANDLE LISTING DEVICE'
1577
1578
1579                            ;ROUTINE TO OUTPUT ONE CHARACTER ON THE LISTING DEVICE
1580
1581
1582  2769  C9 0D   LOUTC:  CMP#    EOLCHR              ;END OF LINE?
1583  276B  D0 11            BNE     LOUTC1              ;BRANCH IF NOT
1584  276D  48       LCRLF:  PHA                         ;SAVE THE END-OF-LINE CHAR
1585  276E  A9 0D            LDA#    CRCHR               ;OUTPUT A CARRIAGE RETURN
1586  2770  20 9D27          JSR     LOUTB
1587  2773  A9 0A            LDA#    LFCHR               ;AND A LINE FEED TO ADVANCE TO NEXT LINE
1588  2775  20 9D27          JSR     LOUTB
1589  2778  A9 00            LDA#    0                   ;RESET HORIZONTAL POSITION COUNTER
1590  277A  85 44            STA     HPOS
1591  277C  68               PLA                         ;RESTORE END-OF-LINE CHAR
1592  277D  60               RTS                         ;AND RETURN
1593  277E  48       LOUTC1: PHA                         ;SAVE CHAR TO OUTPUT
1594  277F  A5 44            LDA     HPOS                ;GET HORIZONTAL POSITION
1595  2781  C5 43            CMP     LWIDTH              ;ARE WE STILL LESS THAN MAX WIDTH?
1596  2783  10 14            BPL     LOUTCP              ;NO, THEN DON'T OUTPUT THIS CHARACTER
1597  2785  68               PLA                         ;GET THE CHARACTER BACK
1598  2786  C9 09            CMP#    TABCHR              ;TABULATION?
1599  2788  D0 13            BNE     LOUTB               ;* PBNE * NO, THEN OUTPUT AND RETURN
1600  278A  48               PHA                         ;SAVE THE CHARACTER
1601  278B  20 3E27          JSR     FNDTAB              ;FIND THE NEXT TAB STOP
1602  278E  20 6727 LOUTC2:  JSR     LSPACE              ;NOW SIMULATE THE TAB WITH SPACES
1603  2791  A5 44            LDA     HPOS                ;GET OUR HORIZONTAL POSITION
1604  2793  C5 45            CMP     LTEMP               ;HAVE WE REACHED TAB STOP YET?
1605  2795  D0 F7            BNE     LOUTC2              ;BRANCH IF NOT
1606  2797  68               PLA                         ;RESTORE CHAR
1607  2798  60               RTS                         ;AND RETURN
1608  2799  E6 44   LOUTCP:  INC     HPOS                ;COUNT THIS CHAR EVEN THOUGH IT WASN'T OUTPUT
1609  279B  68               PLA                         ;RESTORE THE CHARACTER
1610  279C  60               RTS
1611
1612                            ;ROUTINE TO OUTPUT ONE BYTE TO LISTING DEVICE
1613
1614  279D  48       LOUTB:  PHA                         ;SAVE THE BYTE TO OUTPUT
1615  279E  A5 27            LDA     DEVO                ;GET CURRENT OUTPUT DEVICE NUMBER
1616  27A0  48               PHA                         ;SAVE IT ON STACK
1617  27A1  A5 23            LDA     TLDEV               ;GET TEMPORARY LISTING DEVICE
1618  27A3  85 27            STA     DEVO                ;MAKE IT THE CURRENT ONE
1619  27A5  BA               TSX                         ;GET STACK POINTER
1620  27A6  BD 0201          LDAX    STACK+2             ;GET CHARACTER TO OUTPUT
1621  27A9  20 9229          JSR     OUTB                ;OUTPUT IT
1622  27AC  E6 44            INC     HPOS                ;BUMP THE HORIZONTAL POSITION
1623  27AE  68               PLA                         ;RESTORE ORIGINAL DEVICE NUMBER
1624  27AF  85 27            STA     DEVO
1625  27B1  68               PLA                         ;RESTORE BYTE INTO ACCUMULATOR
1626  27B2  60               RTS                         ;AND RETURN
```

```
1627                                .PAGE                   ;'ROUTINES TO HANDLE OUTPUT TO BINARY DEVICE'
1628
1629
1630                                ;'OUTBIN' DUMPS BYTES THAT WERE GENERATED TO THE 'BINARY' DEVICE.
1631
1632  27B3  A5 3C   OUTBIN: LDA     NBYTES          ;ANY BYTES TO OUTPUT?
1633  27B5  F0 41           BEQ     OUTRT           ;BRANCH IF NOTHING TO DO
1634  27B7  A0 00           LDY#    0               ;YES, POINT TO START OF BYTE LIST
1635  27B9  B9 3D00 OBLOOP: LDAY    BYTE1           ;GET A BYTE FROM THE BYTE LIST
1636  27BC  85 2D           STA     NUMBL           ;STORE IT
1637  27BE  20 F927         JSR     BBYTO           ;OUTPUT IT TO BINARY DEVICE.
1638  27C1  C8              INY                     ;POINT TO NEXT BYTE
1639  27C2  C4 3C           CPY     NBYTES          ;HAVE WE OUTPUT THEM ALL YET?
1640  27C4  D0 F3           BNE     OBLOOP          ;BRANCH IF MORE TO OUTPUT
1641  27C6  F0 30           BEQ     OUTRT           ;RETURN
1642
1643                                ;ROUTINE TO OUTPUT A BYTE AS 2 HEX CHARACTERS TO BINARY DEVICE.
1644                                ;CALLED FROM LINE 1829 ONLY
1645  27C8  A5 2D   WROB:   LDA     NUMBL           ;GET BYTE
1646  27CA  4A              LSRA                    ;GET HIGH ORDER
1647  27CB  4A              LSRA
1648  27CC  4A              LSRA
1649  27CD  4A              LSRA
1650  27CE  20 D527         JSR     BHEXO           ;OUTPUT HIGH ORDER HEX DIGIT
1651  27D1  A5 2D           LDA     NUMBL           ;GET LOW ORDER
1652  27D3  29 0F           AND#    %17             ;* PFALL * AND OUTPUT LOW ORDER HEX DIGIT
1653  27D5  AA      BHEXO:  TAX                     ;MOVE HEX INTO X REGISTER
1654  27D6  BD CB2B         LDAX    HEXTAB          ;CONVERT TO BINARY
1655  27D9  D0 0A           BNE     BINOUT          ;* PBNE * OUTPUT AND RETURN
1656
1657
1658
1659
```

Looks like a good place to add rel. addressing


```
1660                                .PAGE                   ;'ROUTINES TO HANDLE OUTPUT TO BINARY DEVICE'
1661
1662
1663                                ;ELEMENTARY OUTPUT ROUTINES TO BINARY DEVICE.
1664                                ;CALLED ONLY @ 1747
1665
1666  27DB  A9 0D   BCRLF:  LDA#    CRCHR           ;OUTPUT CRLF
1667  27DD  20 E527         JSR     BINOUT          ;
1668  27E0  A9 0A           LDA#    LFCHR           ;
1669  27E2  20 E527         JSR     BINOUT          ;
1670  27E5  48      BINOUT: PHA                     ;SAVE CHAR TO OUTPUT
1671  27E6  A5 27           LDA     DEVO            ;SAVE CURRENT OUTPUT DEVICE
1672  27E8  48              PHA
1673  27E9  A5 22           LDA     BINDEV          ;REPLACE WITH BINARY OUTPUT DEVICE
1674  27EB  85 27           STA     DEVO
1675  27ED  BA              TSX                     ;GET STACK POINTER
1676  27EE  BD 0201         LDAX    STACK+2         ;GET CHARACTER TO OUTPUT
1677  27F1  20 9229         JSR     OUTB            ;OUTPUT IT
1678  27F4  68              PLA                     ;RESTORE OUTPUT DEVICE
1679  27F5  85 27           STA     DEVO
1680  27F7  68              PLA                     ;RESTORE CHARACTER
1681  27F8  60      OUTRT:  RTS                     ;AND RETURN
1682
1683
```

This is why 2 line-feeds!

BINOUT REFERRED TO AT LINES: 1655, (1667, 1669), 1798

FROM 1827 ONLY

1744, 1746, 1793 only

reg saving routine restoring called @ 1727 only

normal, except in setting new adr or for .end

CALLED FROM LINES 921, 924, 1637

```
1684                            .PAGE
1685
1686                            ;CONVERTS BINARY TO KIM/TIM FORMATT
1687
1688
1689  27F9  A5 2D   BBYTO:  LDA     NUMBL       ;GET BYTE
1690  27FB  A6 7B           LDX     BFLAG       ;TEST FOR ADDRESS VS DATA
1691  27FD  F0 11           BEQ     CHART       ;BRANCH IF NOT ADDRESS
1692  27FF  A6 7C           LDX     ADDCNT      ;GET INDEX
1693  2801  9D C328         STAX    ADBUFF      ;STORE IT
1694  2804  E6 7C           INC     ADDCNT      ;COUNT
1695  2806  A6 7C           LDX     ADDCNT      ;HAVE WE LOADED 2 BYTES?
1696  2808  E0 02           CPX#    $02         ;TWO?
1697  280A  D0 03           BNE     BIRTS       ;RETURN IF NOT ZERO
1698  280C  20 8F28         JSR     RSFLG       ;RESET FLAG
1699  280F  60              RTS                 ;
1700  2810  A6 7D   CHART:  LDX     DATCNT      ;GET INDEX
1701  2812  9D C528         STAX    DATBUF      ;STORE THE BINARY
1702  2815  E6 7D           INC     DATCNT      ;COUNT
1703  2817  E0 0A           CPX#    $0A         ;TEN?
1704  2819  D0 F4           BNE     BIRTS       ;NO;EXIT
1705  281B  20 3128         JSR     PRINT       ;PRINT BUFFER
1706  281E  60              RTS                 ;
1707  281F  20 3128 HCTZ:   JSR     PRINT       ;PRINT
1708  2822  20 A428         JSR     MARESET     ;MASTER RESET
1709  2825  60              RTS                 ;
1710  2826  20 3128 HSTAR:  JSR     PRINT       ;PRINT
1711  2829  A2 FF           LDX#    $FF         ;SET FLAG FOR AN ADDRESS READ
1712  282B  86 7B           STX     BFLAG       ;
1713  282D  E8              INX                 ;MAKE IT ZERO
1714  282E  86 7C           STX     ADDCNT      ;STORE IN COUNTER
1715  2830  60              RTS                 ;
1716
1717
```

only only

— CALLED @ 942 only

— CALLED @ 918 only

```
1718                            .PAGE
1719                            ;PRINT ROUTINE
1720
1721  2831  98        PRINT:  TYA                     ;SAVE Y REGISTER FOR CALLING ROUTINE
1722  2832  48                PHA                     ;
1723  2833  A4 7D            LDY     DATCNT          ;ANY DATA IN THE BUFFER?
1724  2835  F0 3D            BEQ     PROUT           ;DON'T ALLOW PRINT IF NO DATA HAS BEEN LOADED
1725  2837  A0 00            LDY#    $00             ;CLEAR X
1726  2839  A9 3B            LDA#    ';              ;PRINT A SEMI
1727  283B  20 9428           JSR     SWRT            ;
1728  283E  A5 7D            LDA     DATCNT          ;GET # OF DATA BYTES
1729  2840  20 8028           JSR     ACCHOU          ;ADD TO CHECKSUM CONVERT TO ASCII & OUT PUT
1730  2843  AD C328           LDA     ADBUFF  ;GET FIRST ADDRESS BYTE
1731  2846  20 8028           JSR     ACCHOU          ;OUTPUT IT
1732  2849  AD C428           LDA     ADBUFF+1                ;GET LOW ORDER BYTE
1733  284C  20 8028           JSR     ACCHOU          ;PRINT IT
1734  284F  B9 C528  BPRINT:  LDAY    DATBUF          ;GET BYTE
1735  2852  20 8028           JSR     ACCHOU          ;CHECKSUM,ASCII AND OUTPUT
1736  2855  EE C428           INC     ADBUFF+1                ;BUMP ADDRESS COUNTER
1737  2858  D0 03            BNE     NHBUA           ;WRAP AROUND?
1738  285A  EE C328           INC     ADBUFF  ;BUMP HIGH ORDER
1739  285D  C8          NHBUA:  INY                     ;COUNT
1740  285E  C4 7D            CPY     DATCNT          ;ALL DATA OUT?
1741  2860  F0 02            BEQ     PCHEK           ;BRANCH IF YES
1742  2862  D0 EB            BNE     BPRINT          ;LOOP TILL DONE
1743  2864  A5 7F    PCHEK:  LDA     CHKBUF+1        ;PRINT CHECKSUM
1744  2866  20 B928           JSR     ASCOUT          ;HIGH BYTE
1745  2869  A5 7E            LDA     CHKBUF          ;LOW BYTE
1746  286B  20 B928           JSR     ASCOUT          ;
1747  286E  20 DB27           JSR     BCRLF           ;PRINT A CRLF
1748  2871  20 7728           JSR     ZALL            ;CLEAR COUNTERS
1749  2874  68          PROUT:  PLA                     ;RESTORE Y REGISTER
1750  2875  A8                TAY                     ;
1751  2876  60                RTS                     ;
1752  2877  A9 00      ZALL:   LDA#    $00             ;CLEAR COUNTERS
1753  2879  85 7E            STA     CHKBUF          ;
1754  287B  85 7F            STA     CHKBUF+1        ;
1755  287D  85 7D            STA     DATCNT          ;
1756  287F  60                RTS                     ;
1757
1758
1759
1760
```



```
1761                            .PAGE
1762
1763                    CALLED ONLY FROM PRINT ROUTINE ABOVE
1764                            ;ADD BYTE IN ACCUMM. TO CHECK SUM, CONVERT TO ASCII AND PRINT
1765
1766  2880  18          ACCHOU: CLC                     ;CLEAR CARRY FOR ADD
1767  2881  48                PHA                     ;SAVE ACCUM.
1768  2882  65 7E            ADC     CHKBUF          ;ADD TO LOW BYTE
1769  2884  85 7E            STA     CHKBUF          ;STORE SUM
1770  2886  90 02            BCC     NCARY           ;BRANCH IF NO CARRY
1771  2888  E6 7F            INC     CHKBUF+1        ;BUMB HIGH BYTE
1772  288A  68          NCARY:  PLA                     ;GET ACCUM BACK
1773  288B  20 B928           JSR     ASCOUT          ;CONVERT TO ASCII AND PRINT
1774  288E  60                RTS                     ;
1775
1776
```

```
1777                            .PAGE
1778                            ;ROUTINE CLEARS ADDRESS FLAG, AND TWO BYTE COUNTER
1779
1780
1781  288F  A9 00   RSFLG:  LDA#    $00             ;GET ZERO
1782  2891  85 7B           STA     BFLAG           ;SET FLAG TO ZERO
1783  2893  60              RTS
1784
1785
1786
1787
1788                            ;PRINT CHARACTER SAVING REGISTERS
1789
1790                          CALLED @ 1729 ONLY
1791
1792  2894  85 80   SWRT:   STA     BITEMP          ;SAVE ACCUMULATOR
1793  2896  8A              TXA                     ;PUT X IN ACCUMULATOR
1794  2897  48              PHA                     ;SAVE IT ON STACK
1795  2898  98              TYA                     ;PUT Y IN ACCUMULATOR
1796  2899  48              PHA                     ;PUSH IT
1797  289A  A5 80           LDA     BITEMP          ;GET ACCUM BACK
1798  289C  20 E527         JSR     BINOUT          ;JUMP TO BIT OUTPUT ROUTINE
1799  289F  68              PLA                     ;PULL Y OFF STACK
1800  28A0  A8              TAY                     ;PUT IT BACK IN REGISTER
1801  28A1  68              PLA                     ;GET X REGISTER
1802  28A2  AA              TAX                     ;PUT IT BACK
1803  28A3  60              RTS                     ;
```



```
1804                            .PAGE
1805
1806
1807                            ;MASTER RESET
1808
1809
1810  28A4  20 8F28 MARESE: JSR     RSFLG           ;RESET FLAG
1811  28A7  85 7C           STA     ADDCNT          ;CLEAR ADDCNT WITH ZERO FROM LAST SUBROUTINE
1812  28A9  20 7728         JSR     ZALL            ;ZERO CHECK SUM AND DATA COUNT
1813  28AC  20 B028         JSR     ZADRES          ;ZERO ADBUFF
1814  28AF  60              RTS                     ;
1815
1816
1817                            ;ROUTINE ZEROS ADBUFF
1818
1819  28B0  A9 00   ZADRES: LDA#    $00             ;GET A ZERO
1820  28B2  8D C328         STA     ADBUFF          ;STORE IN FIRST LOC
1821  28B5  8D C428         STA     ADBUFF+1        ;STORE IN SECOND BYTE
1822  28B8  60              RTS                     ;
1823
1824
1825                            ;CONVERT A BYTE TO ASCII AND OUTPUT TO BINARY DEVICE
1826
1827  28B9  86 80   ASCOUT: STX     BITEMP          ;SAVE X REGISTER
1828  28BB  85 2D           STA     NUMBL           ;PASS BYTE
1829  28BD  20 C827         JSR     NROB            ;CONVERT TO HEX ASCII AND OUTPUT
1830  28C0  A6 80           LDX     BITEMP          ;GET X BACK
1831  28C2  60              RTS                     ;
1832          CALLED AT: 1744, 1746, 1773
1833
```



```
1834                            .PAGE
1835                            ;DEFINES BUFFERS
1836  28C3  00 00   ADBUFF: .WORD   ;2 BYTES USED TO STORE ADDRESS
1837  28C5  00      DATBUF: .BYTE
1838          28D1            .LOC    DATBUF+12       ;DEFINES SPACE FOR BINARY BUFFER
```

ADBUFF: RESET @ 1820/1821
USED FOR OUTPUT @ 1730/32/36/38
SET @ 1693

DATBUF: USED FOR OUTPUT @ 1734
SET @ 1701

```
1839                                .PAGE                   ;'NUMERIC INPUT UTILITY ROUTINES'
1840
1841                                ;ROUTINE TO INPUT A DECIMAL NUMBER
1842
1843  28D1  20 1C29 INPDEC: JSR     ZERNUM                  ;ASSUME ZERO
1844  28D4  20 A625 INPD1:  JSR     TESTN                   ;IS IT A NUMBER?
1845  28D7  B0 29           BCS     INPORT                  ;RETURN IF NOT A NUMBER.
1846  28D9  29 0F           AND#    $0F                     ;MAKE BINARY
1847  28DB  20 EC28         JSR     IMUL10                  ;MULTIPLY NUMBER BY 10
1848  28DE  85 2F           STA     NTEMPL                  ;SAVE DIGIT IN 'NTEMP'
1849  28E0  A9 00           LDA#    0                       ;HIGH ORDER OF ZERO
1850  28E2  85 2E           STA     NTEMPH
1851  28E4  20 0E29         JSR     IADD                    ;ADD IN THIS DIGIT
1852  28E7  20 6225         JSR     GETC                    ;GET NEXT CHARACTER
1853  28EA  10 E8           BPL     INPD1                   ;UNCONDITIONALLY LOOP FOR NEXT DIGIT
1854
1855                                ;ROUTINE TO MULTIPLY THE NUMBER IN 'NUMB' BY 10 DECIMAL
1856
1857  28EC  48        IMUL10: PHA                           ;PRESERVE ACCUMULATOR
1858  28ED  A5 2D           LDA     NUMBL                   ;COPY 'NUMB' INTO 'NTEMP'
1859  28EF  85 2F           STA     NTEMPL
1860  28F1  A5 2C           LDA     NUMBH
1861  28F3  85 2E           STA     NTEMPH
1862  28F5  20 0929         JSR     SLEFT1                  ;SHIFT 'NUMB' LEFT TO MULTIPLY TIMES 2
1863  28F8  20 0929         JSR     SLEFT1                  ;TIMES 2 AGAIN
1864  28FB  20 0E29         JSR     IADD                    ;ADD IN ORIGINAL NUMBER FOR 5*ORIGINAL
1865  28FE  20 0929         JSR     SLEFT1                  ;TIMES 2 MAKES 10.
1866  2901  68              PLA                             ;RESTORE ACCUMULATOR
1867  2902  60        INPDRT: RTS                           ;AND RETURN
```



```
1868                                .PAGE                   ;'NUMERIC INPUT UTILITY ROUTINES'
1869
1870
1871                                ;ROUTINES TO SHIFT 'NUMB' LEFT
1872
1873  2903  20 0629 SLEFT4: JSR     SLEFT2                  ;SHIFT LEFT FOUR
1874  2906  20 0929 SLEFT2: JSR     SLEFT1                  ;SHIFT LEFT TWO
1875  2909  06 2D   SLEFT1: ASL     NUMBL                   ;SHIFT LOW ORDER
1876  290B  26 2C           ROL     NUMBH                   ;NOW HIGH ORDER
1877  290D  60              RTS                             ;RETURN
1878
1879                                ;ROUTINE TO ADD 'NTEMP' TO 'NUMB' LEAVING RESULT IN 'NUMB'.
1880
1881  290E  18        IADD:   CLC                           ;CLEAR C
1882  290F  A5 2D           LDA     NUMBL                   ;GET LOW ORDERS
1883  2911  65 2F           ADC     NTEMPL                  ;ADD THEM
1884  2913  85 2D           STA     NUMBL                   ;SAVE RESULT
1885  2915  A5 2C           LDA     NUMBH                   ;GET HIGH ORDERS
1886  2917  65 2E           ADC     NTEMPH                  ;ADD THEM WITH ANY CARRY FROM LOW ORDERS
1887  2919  85 2C           STA     NUMBH                   ;SAVE RESULT
1888  291B  60              RTS                             ;RETURN
1889
1890                                ;ROUTINE TO SET 'NUMB' TO ZERO.
1891
1892  291C  A9 00   ZERNUM: LDA#    0                       ;GET A ZERO
1893  291E  85 2D           STA     NUMBL                   ;STORE IN LOW ORDER
1894  2920  85 2C           STA     NUMBH                   ;STORE IN HIGH ORDER
1895  2922  60              RTS                             ;RETURN
1896
1897                                ;ROUTINE TO SUBTRACT 'NTEMP' FROM 'NUMB' LEAVING RESULT IN 'NUMB'.
1898
1899  2923  38        ISUB:   SEC                           ;SET CARRY (BORROW)
1900  2924  A5 2D           LDA     NUMBL                   ;DO A DOUBLE PRECISION SUBTRACT
1901  2926  E5 2F           SBC     NTEMPL
1902  2928  85 2D           STA     NUMBL
1903  292A  A5 2C           LDA     NUMBH
1904  292C  E5 2E           SBC     NTEMPH
1905  292E  85 2C           STA     NUMBH
1906  2930  60              RTS                             ;AND RETURN
1907
```

```
1908                            .PAGE                   ;'NUMERIC INPUT UTILITY ROUTINES'
1909
1910
1911                            ;ROUTINE TO INPUT A HEX NUMBER
1912
1913  2931  20 1C29 INPHEX: JSR     ZERNUM          ;ASSUME ZERO
1914  2934  20 A625 INPH1:  JSR     TESTN           ;IS THE CHAR 0-9?
1915  2937  B0 04          BCS     NOT09           ;BRANCH IF NOT
1916  2939  29 0F          AND#    %17             ;YES, MAKE BINARY
1917  293B  90 0B          BCC     HEXDIG          ;UNCONDITIONALLY ENTER COMMON CODE
1918  293D  C9 41  NOT09:  CMP#    'A              ;IS IT 'A-F'?
1919  293F  30 13          BMI     HEXRET          ;BRANCH IF NOT
1920  2941  C9 47          CMP#    'F+1
1921  2943  10 0F          BPL     HEXRET          ;BRANCH IF NOT
1922  2945  38             SEC                     ;FORM BINARY
1923  2946  E9 37          SBC#    %67
1924  2948  20 0329 HEXDIG: JSR     SLEFT4          ;MAKE ROOM FOR THIS DIGIT
1925  294B  05 2D          ORA     NUMBL           ;ADD IT INTO FULL NUMBER
1926  294D  85 2D          STA     NUMBL           ;STORE NEW VALUE
1927  294F  20 6225        JSR     GETC            ;MOVE PAST THIS DIGIT
1928  2952  10 E0          BPL     INPH1           ;AND GO LOOK FOR ANOTHER
1929  2954  60       HEXRET: RTS                     ;RETURN
1930
1931                            ;ROUTINE TO INPUT AN OCTAL NUMBER
1932
1933  2955  20 1C29 INPOCT: JSR     ZERNUM          ;ASSUME ZERO
1934  2958  A5 32          LDA     CHAR            ;GET FIRST CHARACTER
1935  295A  C9 30  INPO1:  CMP#    '0              ;IS IT '0-7'?
1936  295C  30 15          BMI     OCTRET          ;BRANCH IF NOT
1937  295E  C9 38          CMP#    '7+1
1938  2960  10 11          BPL     OCTRET          ;BRANCH IF NOT
1939  2962  29 07          AND#    7               ;FORM THE BINARY
1940  2964  20 0629        JSR     SLEFT2          ;MAKE ROOM FOR THIS DIGIT
1941  2967  20 0929        JSR     SLEFT1
1942  296A  05 2D          ORA     NUMBL           ;ADD IN THIS DIGIT
1943  296C  85 2D          STA     NUMBL           ;STORE BACK UPDATED NUMBER
1944  296E  20 6225        JSR     GETC            ;MOVE PAST THIS DIGIT
1945  2971  10 E7          BPL     INPO1           ;UNCONDITIONALLY CHECK FOR ANOTHER DIGIT
1946  2973  60       OCTRET: RTS                     ;RETURN
```

```
1947                                .PAGE                   ;'LOWEST LEVEL I-O ROUTINES'
1948
1949
1950                                ;ALL I-O DONE BY THE ASSEMBLER IS THROUGH THESE SUBROUTINES.
1951                                ;THEY LINK TO THE ACTUAL DEVICE DRIVERS THROUGH ADDRESS DISPATCH
1952                                ;TABLES.
1953
1954                                ;       'INPB' - ROUTINE TO INPUT ONE BYTE INTO THE ACCUMULATOR
1955                                ;                ALSO STORES THE BYTE IN 'CHAR'.
1956
1957
1958  2974  8A        INPB:   TXA                             ;SAVE X REGISTER ON STACK
1959  2975  48                PHA
1960  2976  98                TYA                             ;SAVE Y REGISTER ON STACK
1961  2977  48                PHA
1962  2978  A6 26            LDX     DEVI                    ;GET CURRENT INPUT DEVICE'S NUMBER
1963  297A  BD 0A2C          LDAX    IDSPL                   ;GET LOW ORDER ADDR OF DEVICE HANDLER
1964  297D  85 34            STA     JTEMP                   ;STORE IT
1965  297F  BD 022C          LDAX    IDSPH                   ;GET HIGH ORDER ADDR OF DEVICE HANDLER
1966  2982  85 35            STA     JTEMP+1
1967  2984  20 3300          JSR     JSRIND                  ;CALL THE DRIVER FOR THE DEVICE
1968  2987  B0 52            BCS     DIERR                   ;BRANCH IF THE DEVICE GOT AN ERROR
1969  2989  85 32            STA     CHAR                    ;SAVE THE CHARACTER IN 'CHAR' FOR LATER
1970  298B  68                PLA                             ;RESTORE Y REGISTER
1971  298C  A8                TAY
1972  298D  68                PLA                             ;RESTORE X REGISTER
1973  298E  AA                TAX
1974  298F  A5 32            LDA     CHAR                    ;GET THE CHARACTER INTO THE ACCUMULATOR
1975  2991  60        IORET:  RTS                             ;AND RETURN
1976
1977                                ;       'OUTB' - OUTPUT THE BYTE IN THE ACCUMULATOR TO THE OUTPUT
1978                                ;                DEVICE.  THE BYTE IS PRESERVED.
1979
1980
1981  2992  48        OUTB:   PHA                             ;SAVE THE BYTE TO BE OUTPUT ON STACK
1982  2993  8A                TXA                             ;SAVE X REGISTER ON THE STACK
1983  2994  48                PHA
1984  2995  98                TYA                             ;SAVE Y REGISTER ON THE STACK
1985  2996  48                PHA
1986  2997  A6 27            LDX     DEVO                    ;GET DEVICE NUMBER OF CURRENT OUTPUT DEVICE
1987  2999  BD 1A2C          LDAX    ODSPL                   ;GET LOW ORDER ADDR OF DEVICE HANDLER
1988  299C  85 34            STA     JTEMP                   ;SAVE IT
1989  299E  BD 122C          LDAX    ODSPH                   ;GET HIGH ORDER ADDR OF DEVICE HANDLER
1990  29A1  85 35            STA     JTEMP+1
1991  29A3  BA                TSX                             ;GET STACK POINTER INTO X REGISTER
1992  29A4  BD 0301          LDAX    STACK+3                 ;GET THE BYTE TO OUTPUT INTO THE ACCUMULATOR
1993  29A7  20 3300          JSR     JSRIND                  ;CALL THE DEVICE HANDLER FOR THIS DEVICE
1994  29AA  B0 33            BCS     DOERR                   ;BRANCH IF THERE WAS AN ERROR
1995  29AC  68                PLA                             ;RESTORE Y REGISTER
1996  29AD  A8                TAY
1997  29AE  68                PLA                             ;RESTORE X REGISTER
1998  29AF  AA                TAX
1999  29B0  68                PLA                             ;GET BYTE TO OUTPUT BACK AGAIN
2000  29B1  60                RTS                             ;OTHERWISE RETURN
```

OUTB REFERRED TO AT LINES:
1344, 1621, 1677, 2069

```
2001                            ,PAGE                   ;'DEVICE OPEN AND CLOSE ROUTINES'
2002
2003
2004
2005                            ;HERE TO INITIALIZE AN INPUT DEVICE.  DEVICE NUMBER IS IN
2006                            ;X REGISTER
2007
2008
2009  2982  BD 2A2C INIDEV: LDAX    INIAL           ;GET ADDR OF DEVICE HANDLER
2010  2985  85 34          STA     JTEMP           ;TO DO THE INITIALIZE
2011  2987  BD 222C        LDAX    INIAH           ;GET HIGH ORDER ALSO
2012  298A  85 35   INCONT: STA     JTEMP+1         ;STORE IT
2013  298C  8A             TXA                     ;SAVE THE DEVICE NUMBER
2014  298D  48             PHA                     ;ON STACK
2015  298E  18             CLC                     ;ASSUME SUCCESS
2016  298F  20 3300        JSR     JSRIND          ;CALL THE DEVICE HANDLER
2017  29C2  68             PLA                     ;RESTORE DEVICE NUMBER
2018  29C3  90 CC          BCC     IORET           ;BRANCH IF NO ERRORS
2019  29C5  B0 1A          BCS     ERRC            ;BRANCH IF AN ERROR WAS ENCOUNTERED
2020
2021
2022                            ;HERE TO INITIALIZE AN OUTPUT DEVICE.  DEVICE NUMBER IS IN
2023                            ;X REGISTER
2024
2025
2026  29C7  BD 3A2C INODEV: LDAX    INOAL           ;GET LOW ORDER ADDR OF DEVICE HANDLER
2027  29CA  85 34          STA     JTEMP           ;THAT KNOWS HOW TO DO THE INITIALIZE
2028  29CC  BD 322C        LDAX    INOAH           ;GET THE HIGH ORDER
2029  29CF  D0 E9          BNE     INCONT          ;* PBNE * ENTER COMMON CODE AND THEN RETURN
2030
2031
2032                            ;HERE TO CLOSE AN OUTPUT DEVICE.  DEVICE NUMBER IS IN X REGISTER.
2033
2034  29D1  BD 5A2C CLODEV: LDAX    CLOAL           ;GET LOW ORDER ADDR OF DEVICE HANDLER
2035  29D4  85 34          STA     JTEMP           ;THAT KNOWS HOW TO CLOSE THE DEVICE
2036  29D6  BD 522C        LDAX    CLOAH           ;GET THE HIGH ORDER
2037  29D9  D0 DF          BNE     INCONT          ;* PBNE * ENTER COMMON CODE THEN RETURN
2038
2039                            ;HERE ON I-O ERRORS
2040
2041  29DB  A5 26   DIERR:  LDA     DEVI            ;GET CURRENT INPUT DEVICE'S NUMBER
2042  29DD  10 02           BPL     ERRC            ;UNCONDITIONALLY ENTER REST OF CODE
2043  29DF  A5 27   DOERR:  LDA     DEVO            ;GET CURRENT OUTPUT DEVICE'S NUMBER
2044  29E1  85 28   ERRC:   STA     ERRDEV          ;SAVE THE DEVICE'S NUMBER AS THE ONE THAT
2045                                                ;GOT THE ERROR
2046  29E3  A9 00           LDA#    CONODV          ;NOW MAKE CONSOLE OUTPUT DEVICE THE CURRENT ONE
2047  29E5  85 27           STA     DEVO
2048  29E7  A2 5B           LDX#    MIOERR          ;TELL USER THERE WAS AN ERROR
2049  29E9  20 3F29         JSR     OUTSTR          ;BY OUTPUTTING MESSAGE
2050  29EC  6C 2A00         JMP@    MONADR          ;AND CALL THE DEBUG MONITOR
2051
```

```
2052                            .PAGE                   ;'UTILITY ROUTINES'
2053
2054
2055
2056                            ;ROUTINE TO OUTPUT A STRING WHOSE ADDRESS IS PLACED SOMEWHERE
2057                            ;ON PAGE ZERO.  THE X REGISTER CONTAINS THE PAGE ZERO ADDR
2058                            ;OF WHERE THE TWO BYTE ADDR OF THE STRING IS STORED.  THIS
2059                            ;ROUTINE DESTROYS ALL REGISTERS.
2060
2061
2062  29EF  B5 00   OUTSTR: LDAX    $2000           ;PICK UP LOW ORDER ADDR OF STRING
2063  29F1  85 30           STA     STRADR          ;SAVE IT IN POINTER
2064  29F3  B5 01           LDAX    $2001           ;PICK UP HIGH ORDER ADDR OF STRING
2065  29F5  85 31           STA     STRADR+1        ;SAVE IT IN POINTER
2066  29F7  A0 00           LDY#    0               ;SET OFFSET TO ZERO
2067  29F9  B1 30   OUTSL:  LDA@Y   STRADR          ;GET A BYTE FROM STRING
2068  29FB  F0 06           BEQ     OUTRET          ;STRING IS ENDED WITH A ZERO BYTE
2069  29FD  20 9229         JSR     OUTB            ;OUTPUT THE BYTE TO DEVICE
2070  2A00  C8              INY                     ;POINT TO NEXT BYTE
2071  2A01  10 F6           BPL     OUTSL           ;AND LOOP FOR MORE
2072  2A03  60      OUTRET: RTS                     ;RETURN WHEN STRING IS EXHAUSTED.
2073
2074
2075                            ;ROUTINE TO SET THE INPUT AND OUTPUT DEVICES TO
2076                            ;THE CONSOLE DEVICE.
2077
2078
2079  2A04  A9 00   SETCON: LDA#    CONIDV          ;GET DEVICE NUMBER OF CONSOLE INPUT DEVICE
2080  2A06  85 26           STA     DEVI            ;MAKE IT CURRENT INPUT DEVICE
2081  2A08  A9 00           LDA#    CONODV          ;GET DEVICE NUMBER OF CONSOLE OUTPUT DEVICE
2082  2A0A  85 27           STA     DEVO            ;MAKE IT CURRENT OUTPUT DEVICE
2083  2A0C  60              RTS                     ;AND RETURN
2084
2085
2086                            ;ROUTINE TO CLOSE THE OUTPUT FILES.
2087
2088
2089  2A0D  A6 21   CLOSEO: LDX     LSTDEV          ;GET DEVICE NUMBER OF LISTING DEVICE
2090  2A0F  20 D129         JSR     CLODEV          ;CLOSE IT
2091  2A12  A6 22           LDX     BINDEV          ;GET DEVICE NUMBER OF BINARY DEVICE
2092  2A14  4C D129         JMP     CLODEV          ;* JMP * OUTPUT IT AND RETURN
```

OUTSTR CALLED FROM LINES: 407, 434, 485, 505, 1081, 1325, 1536, 2049
(ERR MSG)

Should have way to put ↑Z or esc at end of binary file:

```
LDA#  $1A       ;↑Z
JSR   BINOUT    ;OUTPUT BYTE TO BINARY DEVICE
```

```
                                .PAGE                   ;'MESSAGE STRINGS OUTPUT BY THE ASSEMBLER'

2A17  0D    IOERRM: .BYTE   CRCHR            ;CARRIAGE RETURN
2A18  0A            .BYTE   LFCHR            ;LINE FEED
2A19  3F            .ASCII  '?I-O ERROR'
2A1A  49
2A1B  2D
2A1C  4F
2A1D  20
2A1E  45
2A1F  52
2A20  52
2A21  4F
2A22  52

2A23  0D            .BYTE   CRCHR            ;ADVANCE ANOTHER LINE
2A24  0A            .BYTE   LFCHR
2A25  00            .BYTE   0                ;ENDS THE STRING

2A26  0D    FLSYMM: .BYTE   CRCHR
2A27  0A            .BYTE   LFCHR
2A28  46            .ASCII  'FLUSH USER SYMBOL TABLE? '
2A29  4C
2A2A  55
2A2B  53
2A2C  48
2A2D  20
2A2E  55
2A2F  53
2A30  45
2A31  52
2A32  20
2A33  53
2A34  59
2A35  4D
2A36  42
2A37  4F
2A38  4C
2A39  20
2A3A  54
2A3B  41
2A3C  42
2A3D  4C
2A3E  45
2A3F  3F
2A40  20

2A41  00            .BYTE   0

2A42  0D    PASSM:  .BYTE   CRCHR
2A43  0A            .BYTE   LFCHR
2A44  57            .ASCII  'WHICH PASS (1 OR 2)? '
2A45  48
```

```
2147  2A46  49
2148  2A47  43
2149  2A48  48
2150  2A49  20
2151  2A4A  50
2152  2A4B  41
2153  2A4C  53
2154  2A4D  53
2155  2A4E  20
2156  2A4F  28
2157  2A50  31
2158  2A51  20
2159  2A52  4F
2160  2A53  52
2161  2A54  20
2162  2A55  32
2163  2A56  29
2164  2A57  3F
2165  2A58  20
2166
2167  2A59  00              .BYTE   0
2168
2169  2A5A  44      SRCNM:  .ASCII  'DEVICE # FOR SOURCE INPUT:'
2170  2A5B  45
2171  2A5C  56
2172  2A5D  49
2173  2A5E  43
2174  2A5F  45
2175  2A60  20
2176  2A61  23
2177  2A62  20
2178  2A63  46
2179  2A64  4F
2180  2A65  52
2181  2A66  20
2182  2A67  53
2183  2A68  4F
2184  2A69  55
2185  2A6A  52
2186  2A6B  43
2187  2A6C  45
2188  2A6D  20
2189  2A6E  49
2190  2A6F  4E
2191  2A70  50
2192  2A71  55
2193  2A72  54
2194  2A73  3A
2195
2196  2A74  00              .BYTE   0
2197
2198  2A75  44      BINNM:  .ASCII  'DEVICE # FOR BINARY OUTPUT:'
2199  2A76  45
2200  2A77  56
```

```
2201  2A78  49
2202  2A79  43
2203  2A7A  45
2204  2A7B  20
2205  2A7C  23
2206  2A7D  20
2207  2A7E  46
2208  2A7F  4F
2209  2A80  52
2210  2A81  20
2211  2A82  42
2212  2A83  49
2213  2A84  4E
2214  2A85  41
2215  2A86  52
2216  2A87  59
2217  2A88  20
2218  2A89  4F
2219  2A8A  55
2220  2A8B  54
2221  2A8C  50
2222  2A8D  55
2223  2A8E  54
2224  2A8F  3A
2225
2226  2A90  00              .BYTE   0
2227
2228  2A91  44        LSTNM:  .ASCII  'DEVICE # FOR LISTING OUTPUT:'
2229  2A92  45
2230  2A93  56
2231  2A94  49
2232  2A95  43
2233  2A96  45
2234  2A97  20
2235  2A98  23
2236  2A99  20
2237  2A9A  46
2238  2A9B  4F
2239  2A9C  52
2240  2A9D  20
2241  2A9E  4C
2242  2A9F  49
2243  2AA0  53
2244  2AA1  54
2245  2AA2  49
2246  2AA3  4E
2247  2AA4  47
2248  2AA5  20
2249  2AA6  4F
2250  2AA7  55
2251  2AA8  54
2252  2AA9  50
2253  2AAA  55
2254  2AAB  54
```

```
2255  2AAC  3A
2256
2257  2AAD  00                .BYTE   0
2258
2259  2AAE  0C        PAGEM:  .BYTE   FFCHR           ;FORM-FEED TO NEW PAGE
2260  2AAF  2A        TITLEM: .ASCII  '*** ASM65 ***  (V1C)  13-OCT-77'
2261  2AB0  2A
2262  2AB1  2A
2263  2AB2  20
2264  2AB3  41
2265  2AB4  53
2266  2AB5  4D
2267  2AB6  36
2268  2AB7  35
2269  2AB8  20
2270  2AB9  2A
2271  2ABA  2A
2272  2ABB  2A
2273  2ABC  20
2274  2ABD  20
2275  2ABE  28
2276  2ABF  56
2277  2AC0  31
2278  2AC1  43
2279  2AC2  29
2280  2AC3  20
2281  2AC4  20
2282  2AC5  31
2283  2AC6  33
2284  2AC7  2D
2285  2AC8  4A
2286  2AC9  55
2287  2ACA  4E
2288  2ACB  2D
2289  2ACC  37
2290  2ACD  37
2291
2292  2ACE  0D                .BYTE   CRCHR
2293  2ACF  0A                .BYTE   LFCHR
2294  2AD0  0D                .BYTE   CRCHR
2295  2AD1  0A                .BYTE   LFCHR
2296  2AD2  0D                .BYTE   CRCHR
2297  2AD3  0A                .BYTE   LFCHR
2298  2AD4  00                .BYTE   0
2299
2300  2AD5  57        WIDTHM: .ASCII  'WIDTH (127 MAX) OF'
2301  2AD6  49
2302  2AD7  44
2303  2AD8  54
2304  2AD9  48
2305  2ADA  20
2306  2ADB  28
2307  2ADC  31
2308  2ADD  32
```

```
2309  2ADE  37
2310  2ADF  20
2311  2AE0  4D
2312  2AE1  41
2313  2AE2  58
2314  2AE3  29
2315  2AE4  20
2316  2AE5  4F
2317  2AE6  46
2318
2319  2AE7  0D              .BYTE   CRCHR
2320  2AE8  0A              .BYTE   LFCHR
2321  2AE9  4C              .ASCII  'LISTING DEVICE:'
2322  2AEA  49
2323  2AEB  53
2324  2AEC  54
2325  2AED  49
2326  2AEE  4E
2327  2AEF  47
2328  2AF0  20
2329  2AF1  44
2330  2AF2  45
2331  2AF3  56
2332  2AF4  49
2333  2AF5  43
2334  2AF6  45
2335  2AF7  3A
2336
2337  2AF8  00              .BYTE   0
2338
2339  2AF9  45      EOFMS:  .ASCII  'END-OF-FILE (CTRL-Z) SEEN ON'
2340  2AFA  4E
2341  2AFB  44
2342  2AFC  20
2343  2AFD  4F
2344  2AFE  46
2345  2AFF  2D
2346  2B00  46
2347  2B01  49
2348  2B02  4C
2349  2B03  45
2350  2B04  20
2351  2B05  28
2352  2B06  43
2353  2B07  54
2354  2B08  52
2355  2B09  4C
2356  2B0A  2D
2357  2B0B  5A
2358  2B0C  29
2359  2B0D  20
2360  2B0E  53
2361  2B0F  45
2362  2B10  45
```

```
2363  2B11  4E
2364  2B12  20
2365  2B13  4F
2366  2B14  4E
2367
2368  2B15  0D              .BYTE   CRCHR
2369  2B16  0A              .BYTE   LFCHR
2370  2B17  53              .ASCII  'SOURCE DEVICE.  ENTER NEW'
2371  2B18  4F
2372  2B19  55
2373  2B1A  52
2374  2B1B  43
2375  2B1C  45
2376  2B1D  20
2377  2B1E  44
2378  2B1F  45
2379  2B20  56
2380  2B21  49
2381  2B22  43
2382  2B23  45
2383  2B24  2E
2384  2B25  20
2385  2B26  20
2386  2B27  45
2387  2B28  4E
2388  2B29  54
2389  2B2A  45
2390  2B2B  52
2391  2B2C  20
2392  2B2D  4E
2393  2B2E  45
2394  2B2F  57
2395
2396  2B30  0D              .BYTE   CRCHR
2397  2B31  0A              .BYTE   LFCHR
2398  2B32  00              .BYTE   0
2399
2400  2B33  0D      GERMES: .BYTE   CRCHR
2401  2B34  0A              .BYTE   LFCHR
2402  2B35  3F              .ASCII  '? *** ERRORS WERE DETECTED! ***'
2403  2B36  20
2404  2B37  2A
2405  2B38  2A
2406  2B39  2A
2407  2B3A  20
2408  2B3B  45
2409  2B3C  52
2410  2B3D  52
2411  2B3E  4F
2412  2B3F  52
2413  2B40  53
2414  2B41  20
2415  2B42  57
2416  2B43  45
```

```
2417  2B44  52
2418  2B45  45
2419  2B46  20
2420  2B47  44
2421  2B48  45
2422  2B49  54
2423  2B4A  45
2424  2B4B  43
2425  2B4C  54
2426  2B4D  45
2427  2B4E  44
2428  2B4F  21
2429  2B50  20
2430  2B51  2A
2431  2B52  2A
2432  2B53  2A
2433
2434  2B54  0D              .BYTE   CRCHR
2435  2B55  0A              .BYTE   LFCHR
2436  2B56  00              .BYTE   0
2437
2438  2B57  0D      NERMES: .BYTE   CRCHR
2439  2B58  0A              .BYTE   LFCHR
2440  2B59  4E              .ASCII  'NO ERRORS DETECTED'
2441  2B5A  4F
2442  2B5B  20
2443  2B5C  45
2444  2B5D  52
2445  2B5E  52
2446  2B5F  4F
2447  2B60  52
2448  2B61  53
2449  2B62  20
2450  2B63  44
2451  2B64  45
2452  2B65  54
2453  2B66  45
2454  2B67  43
2455  2B68  54
2456  2B69  45
2457  2B6A  44
2458
2459  2B6B  0D              .BYTE   CRCHR
2460  2B6C  0A              .BYTE   LFCHR
2461  2B6D  00              .BYTE   0
2462
2463  2B6E  52      READYM: .ASCII  'READY SOURCE DEVICE'
2464  2B6F  45
2465  2B70  41
2466  2B71  44
2467  2B72  59
2468  2B73  20
2469  2B74  53
2470  2B75  4F
```

```
2471  2B76  55
2472  2B77  52
2473  2B78  43
2474  2B79  45
2475  2B7A  20
2476  2B7B  44
2477  2B7C  45
2478  2B7D  56
2479  2B7E  49
2480  2B7F  43
2481  2B80  45
2482
2483  2B81  0D              .BYTE   CRCHR
2484  2B82  0A              .BYTE   LFCHR
2485  2B83  54              .ASCII  'THEN HIT "RETURN" KEY'
2486  2B84  48
2487  2B85  45
2488  2B86  4E
2489  2B87  20
2490  2B88  48
2491  2B89  49
2492  2B8A  54
2493  2B8B  20
2494  2B8C  22
2495  2B8D  52
2496  2B8E  45
2497  2B8F  54
2498  2B90  55
2499  2B91  52
2500  2B92  4E
2501  2B93  22
2502  2B94  20
2503  2B95  4B
2504  2B96  45
2505  2B97  59
2506
2507  2B98  0D              .BYTE   CRCHR
2508  2B99  0A              .BYTE   LFCHR
2509  2B9A  00              .BYTE   0
2510
2511  2B9B  0D      OVFLM:  .BYTE   CRCHR
2512  2B9C  0A              .BYTE   LFCHR
2513  2B9D  3F              .ASCII  '?SYMBOL TABLE OVERFLOW.'
2514  2B9E  53
2515  2B9F  59
2516  2BA0  4D
2517  2BA1  42
2518  2BA2  4F
2519  2BA3  4C
2520  2BA4  20
2521  2BA5  54
2522  2BA6  41
2523  2BA7  42
2524  2BA8  4C
```

```
2525  2BA9  45
2526  2BAA  20
2527  2BAB  4F
2528  2BAC  56
2529  2BAD  45
2530  2BAE  52
2531  2BAF  46
2532  2BB0  4C
2533  2BB1  4F
2534  2BB2  57
2535  2BB3  2E
2536
2537  2BB4  0D              .BYTE   CRCHR
2538  2BB5  0A              .BYTE   LFCHR
2539  2BB6  00              .BYTE   0
2540
2541  2BB7  20      ERRORM: .ASCII  ' !!! ERROR(S) !!!'
2542  2BB8  21
2543  2BB9  21
2544  2BBA  21
2545  2BBB  20
2546  2BBC  45
2547  2BBD  52
2548  2BBE  52
2549  2BBF  4F
2550  2BC0  52
2551  2BC1  28
2552  2BC2  53
2553  2BC3  29
2554  2BC4  20
2555  2BC5  21
2556  2BC6  21
2557  2BC7  21
2558
2559  2BC8  0D              .BYTE   CRCHR
2560  2BC9  0A              .BYTE   LFCHR
2561  2BCA  00              .BYTE   0
2562
2563  2BCB  30      HEXTAB: .ASCII  '0123456789ABCDEF'
2564  2BCC  31
2565  2BCD  32
2566  2BCE  33
2567  2BCF  34
2568  2BD0  35
2569  2BD1  36
2570  2BD2  37
2571  2BD3  38
2572  2BD4  39
2573  2BD5  41
2574  2BD6  42
2575  2BD7  43
2576  2BD8  44
2577  2BD9  45
2578  2BDA  46
2579
2580
2581  2BDB  53      ERRTAB: .ASCII  'SAOPIUML'
2582  2BDC  41
2583  2BDD  4F
2584  2BDE  50
2585  2BDF  49
2586  2BE0  55
2587  2BE1  4D
2588  2BE2  4C
2589
```

```
2590                            .PAGE                   ;'DISPATCH TABLES FOR OPCODE CLASSES'
2591
2592
2593                            ;HIGH ORDER ADDRESSES
2594
2595
2596  2BE3  22      CDSPH:    .HBYTE  CLASS0
2597  2BE4  22                .HBYTE  CLASS1
2598  2BE5  23                .HBYTE  CLASS2
2599  2BE6  23                .HBYTE  CLASS3
2600  2BE7  23                .HBYTE  CLASS4
2601  2BE8  23                .HBYTE  CLASS5
2602  2BE9  23                .HBYTE  CLASS6
2603
2604
2605                            ;LOW ORDER ADDRESSES
2606
2607
2608  2BEA  D2      CDSPL:    .BYTE   CLASS0
2609  2BEB  E2                .BYTE   CLASS1
2610  2BEC  07                .BYTE   CLASS2
2611  2BED  0D                .BYTE   CLASS3
2612  2BEE  1B                .BYTE   CLASS4
2613  2BEF  4B                .BYTE   CLASS5
2614  2BF0  54                .BYTE   CLASS6
2615
2616
2617                            ;THIS TABLE DEFINES WHERE THE TAB STOPS ARE ON THE
2618                            ;LISTING DEVICE.  COLUMNS ARE NUMBERED 0-127.
2619
2620  2BF1  0F      TABS:     .BYTE   15
2621  2BF2  17                .BYTE   23
2622  2BF3  1D                .BYTE   29
2623  2BF4  00                .BYTE   0               ;ZEROES ARE HERE TO FILL TABLE OUT
2624  2BF5  00                .BYTE   0               ;IN CASE USER WANTS TO OVERWRITE WITH
2625  2BF6  28                .BYTE   40              ;STANDARD 8 CHARACTER TAB STOPS
2626  2BF7  30                .BYTE   48
2627  2BF8  38                .BYTE   56
2628  2BF9  40                .BYTE   64
2629  2BFA  48                .BYTE   72
2630  2BFB  50                .BYTE   80
2631  2BFC  58                .BYTE   88
2632  2BFD  60                .BYTE   96
2633  2BFE  68                .BYTE   104
2634  2BFF  70                .BYTE   112
2635  2C00  78                .BYTE   120
2636  2C01  7F                .BYTE   127             ;AT END OF LINE.
2637
```

```
2638                            .PAGE                   ;'I-O DISPATCH TABLES FOR ASSEMBLER'
2639
2640
2641
2642                            ;HERE ARE THE TABLES OF ADDRESSES USED BY THE ASSEMBLER
2643                            ;WHEN DOING BYTE I-O TO INPUT AND OUTPUT DEVICES.  EACH
2644                            ;DEVICE HAS A DEVICE NUMBER (0,1,2,...) THAT CORRESPONDS TO
2645                            ;IT'S ENTRY POSITION IN THE FOLLOWING TABLES.  THERE IS SOME
2646                            ;PATCH SPACE FOR USERS TO ADD ADDRESSES OF CUSTOMIZED DEVICE
2647                            ;HANDLERS.
2648
2649
2650                            ;*** RULES TO FOLLOW WHEN WRITING YOUR OWN DEVICE HANDLER ***
2651
2652
2653                            ;1.     THE DEVICE HANDLER IS ALWAYS CALLED WITH A 'JSR'.  THE
2654                            ;       HANDLER MAY DESTROY ANY REGISTERS AND SHOULD RETURN VIA
2655                            ;       AN 'RTS'.
2656                            ;
2657                            ;2.     WHEN THE HANDLER RETURNS, IT SHOULD LEAVE THE 'C' BIT=0
2658                            ;       IF THE I-O OPERATION WAS SUCCESSFUL, OR THE 'C' BIT=1
2659                            ;       IF THE OPERATION WAS NOT SUCCESSFUL.
2660                            ;
2661                            ;3.     'OUTPUT' HANDLERS WILL BE SUPPLIED THE BYTE TO OUTPUT
2662                            ;       IN THE ACCUMULATOR.
2663                            ;
2664                            ;4.     UPON A SUCCESSFUL RETURN FROM AN 'INPUT' HANDLER, THE
2665                            ;       NEXT BYTE FROM THE DEVICE SHALL BE RETURNED IN THE
2666                            ;       ACCUMULATOR.
2667                            ;
2668                            ;5.     ALL ERROR MESSAGES ARE OUTPUT TO THE DEVICE DESIGNATED
2669                            ;       AS THE CONSOLE OUTPUT DEVICE (SYMBOL 'CONODV').  IF A DEVICE
2670                            ;       GETS AN I-O ERROR, AN ERROR MESSAGE WILL BE OUTPUT TO THE
2671                            ;       CONSOLE OUTPUT DEVICE, THE DEVICE NUMBER OF THE DEVICE
2672                            ;       WHICH GOT THE ERROR WILL BE STORED IN 'ERRDEV' AND THE
2673                            ;       ASSEMBLER WILL RETURN TO THE DEBUG MONITOR.
2674                            ;
2675                            ;6.     IF AN INPUT DEVICE RETURNS CHARACTER DATA, IT MUST
2676                            ;       BE 7-BIT ASCII WITH BIT 7 (MSB) OF THE 8-BIT BYTE
2677                            ;       RETURNED = 0.  WHEN OUTPUTTING CHARACTER DATA, THE
2678                            ;       OUTPUT DEVICE HANDLER WILL BE GIVEN A 7-BIT ASCII
2679                            ;       CHARACTER (WITH MSB = 0) IN THE ACCUMULATOR.
2680                            ;
2681
```

1780 - CASSETTE OUT

02DF - CASSETTE IN

02D9 - INIT IN

178C - INIT OUT

02F3 - CLOSE IN

1793 - CLOSE OUT

```
                                ,PAGE                   ;'I-O DISPATCH TABLES FOR ASSEMBLER'

                                ;IF YOUR CUSTOMIZED DEVICE NEEDS NO 'INITIALIZATION' BEFORE IT
                                ;CAN BE USED, THEN YOU MAY PUT THE ADDRESS OF THE DUMMY
                                ;SUBROUTINE 'DUMMY' IN THE APPROPRIATE TABLE ENTRY.

                                ;DEFINE THE DEVICE NUMBERS WHICH CORRESPOND TO THE USER'S
                                ;COMMAND CONSOLE.

                0000            .DEF    CONIDV=0        ;DEVICE 0 - INPUT FROM KEYBOARD
                0000            .DEF    CONODV=0        ;DEVICE 0 - OUTPUT TO TV READOUT

                                                        INPUT DRIVERS
2C02  2C        IDSPH:  .HBYTE  KEYINE          ;DEVICE 0  - KEYBOARD INPUT ROUTINE
2C03  2C                .HBYTE  C0BYTI          ;DEVICE 1  - CASSETTE #0 INPUT ROUTINE
2C04  2C                .HBYTE  C1BYTI          ;DEVICE 2  - CASSETTE #1 INPUT ROUTINE
2C05  36                .HBYTE  EDBUFFRD        ;DEVICE 3 - READ TEXT FROM EDITOR BUFFER
2C06  2C                .HBYTE  NULLI           ;DEVICE 4  - NULL DEVICE
2C07  2C                .HBYTE  NULLI           ;DEVICE 5  - LINE PRINTER SIMULATION
                IDSPE:
                0006            .DEF    DEVMAX=IDSPE-IDSPH

2C08  00                .BYTE   0               ;SPACE FOR HACKERS
2C09  00                .BYTE   0

2C0A  79        IDSPL:  .BYTE   KEYINE
2C0B  7C                .BYTE   C0BYTI
2C0C  7C                .BYTE   C1BYTI
2C0D  69                .BYTE   EDBUFFRD
2C0E  62                .BYTE   NULLI
2C0F  62                .BYTE   NULLI
2C10  00                .BYTE   0
2C11  00                .BYTE   0

2C12  2C        ODSPH:  .HBYTE  TVOUT           ;DEVICE 0  - TV OUTPUT ROUTINES
2C13  2C-17             .HBYTE  C0BYTO          ;DEVICE 1  - CASSETTE #0 OUTPUT ROUTINE
2C14  2C                .HBYTE  C1BYTO          ;DEVICE 2  - CASSETTE #1 OUTPUT ROUTINE
2C15  2C-03  4E79       .HBYTE  DUMMY           ;DEVICE 3 - NO OUTPUT ROUTINE TO EDITOR BUFFER
2C16  2C                .HBYTE  DUMMY           ;DEVICE 4  - NULL DEVICE
2C17  2C-78             .HBYTE  LPTSIM          ;PRINTER SIMULATIONS ROUTINE
2C18  00                .BYTE   0               ;SPACE FOR HACKERS
2C19  00                .BYTE   0

2C1A  66        ODSPL:  .BYTE   TVOUT
2C1B  7C-80             .BYTE   C0BYTO
2C1C  7C                .BYTE   C1BYTO
2C1D  64-07  77         .BYTE   DUMMY
2C1E  64                .BYTE   DUMMY
2C1F  98-B1             .BYTE   LPTSIM
2C20  00                .BYTE   0
2C21  00                .BYTE   0

                                                        OUTPUT DRIVERS
```

```
2735                                    .PAGE                   ;'DISPATCH TABLE FOR I-C DEVICE NUMBERS'
2736
2737  2C22  25          INIAH:  .HBYTE  RTS1                    ;DON'T NEED TO INITIALIZE KEYBOARD
2738  2C23  2C-02               .HBYTE  C0OPNI                  ;CASSETTE #0 OPEN FOR INPUT ROUTINE
2739  2C24  2C                  .HBYTE  C1OPNI                  ;CASSETTE #1 OPEN FOR INPUT ROUTINE
2740  2C25  39                  .HBYTE  EDINITI                 ;INITIALIZE EDITING BUFFER
2741  2C26  25                  .HBYTE  RTS1                    ;NULL DEVICE NEEDS NO INITIALIZATION
2742  2C27  25                  .HBYTE  RTS1                    ;PRINTER SIMULATION IS AN INPUT DEVICE
2743  2C28  00                  .BYTE   0                       ;SPACE FOR HACKERS
2744  2C29  00                  .BYTE   0
2745
2746  2C2A  86          INIAL:  .BYTE   RTS1
2747  2C2B  7C-D9               .BYTE   C0OPNI
2748  2C2C  7C                  .BYTE   C1OPNI
2749  2C2D  28                  .BYTE   EDINITI
2750  2C2E  86                  .BYTE   RTS1
2751  2C2F  86                  .BYTE   RTS1
2752  2C30  00                  .BYTE   0
2753  2C31  00                  .BYTE   0
2754
2755  2C32  25          INOAH:  .HBYTE  CONINI                  ;INITIALIZE ROUTINE FOR TV
2756  2C33  2C-17               .HBYTE  C0OPNO                  ;CASSETTE #0 OPEN FOR OUTPUT ROUTINE
2757  2C34  2C                  .HBYTE  C1OPNO                  ;CASSETTE #1 OPEN FOR OUTPUT ROUTINE
2758  2C35  25-03 DR79          .HBYTE  RTS1                    ;NO EDITOR OUTPUT ROUTINE
2759  2C36  25                  .HBYTE  RTS1                    ;NULL DEVICE NEEDS NO INITIALIZATION
2760  2C37  2C-78               .HBYTE  LPTRES                  ;INITALIZES PRINTER ROUTINE
2761  2C38  00                  .BYTE   0
2762  2C39  00                  .BYTE   0
2763
2764  2C3A  86          INOAL:  .BYTE   CONINI
2765  2C3B  7C-8C               .BYTE   C0OPNO
2766  2C3C  7C                  .BYTE   C1OPNO
2767  2C3D  86-00 70            .BYTE   RTS1
2768  2C3E  86                  .BYTE   RTS1
2769  2C3F  D2-A5               .BYTE   LPTRES
2770  2C40  00                  .BYTE   0
2771  2C41  00                  .BYTE   0
```

INITIALIZE INPUT

Initialize output devices

```
2772                                    .PAGE                   ;'DISPATCH TABLE FOR I-C DEVICE HANDLERS'
2773
2774                                close input Devices
2775
2776  2C42  25          CLIAH:  .HBYTE  RTS1                    ;KEYBOARD DOESN'T NEED A CLOSE ROUTINE
2777  2C43  2C-02               .HBYTE  C0CLSI                  ;CASSETTE #0 INPUT CLOSE ROUTINE
2778  2C44  2C                  .HBYTE  C1CLSI                  ;CASSETTE #1 INPUT CLOSE ROUTINE
2779  2C45  25                  .HBYTE  RTS1                    ;EDITOR REQUIRES NO CLOSE ROUTINE
2780  2C46  25                  .HBYTE  RTS1                    ;NULL DEVICE NEEDS NO CLOSE ROUTINE
2781  2C47  25                  .HBYTE  RTS1                    ;NO CLOSE FOR PRINTER SIMULATON
2782  2C48  00                  .HBYTE  0                       ;SPACE FOR HACKERS
2783  2C49  00                  .HBYTE  0                       ;SPACE FOR HACKERS
2784
2785  2C4A  86          CLIAL:  .BYTE   RTS1
2786  2C4B  7C-F3               .BYTE   C0CLSI
2787  2C4C  7C                  .BYTE   C1CLSI
2788  2C4D  86                  .BYTE   RTS1
2789  2C4E  86                  .BYTE   RTS1
2790  2C4F  86                  .BYTE   RTS1
2791  2C50  00                  .BYTE   0
2792  2C51  00                  .BYTE   0
2793
2794  2C52  25          CLOAH:  .HBYTE  RTS1                    ;TV DOESN'T NEED A CLOSE ROUTINE
2795  2C53  2C-17               .HBYTE  C0CLSO                  ;CASSETTE #0 OUTPUT CLOSE ROUTINE
2796  2C54  2C                  .HBYTE  C1CLSO                  ;CASSETTE #1 OUTPUT CLOSE ROUTINE
2797  2C55  25                  .HBYTE  RTS1                    ;ASSEMBLER DOESN'T OUTPUT TO EDIT
2798  2C56  25                  .HBYTE  RTS1                    ;DUMMY DEVICE NEEDS NO CLOSE
2799  2C57  25                  .HBYTE  RTS1                    ;NO CLOSE NEEDED
2800  2C58  00                  .HBYTE  0                       ;SPACE FOR HACKERS
2801  2C59  00                  .HBYTE  0
2802                                                            close output devices
2803  2C5A  86          CLOAL:  .BYTE   RTS1
2804  2C5B  7C-93               .BYTE   C0CLSO
2805  2C5C  7C                  .BYTE   C1CLSO
2806  2C5D  86                  .BYTE   RTS1
2807  2C5E  86                  .BYTE   RTS1
2808  2C5F  86                  .BYTE   RTS1
2809  2C60  00                  .BYTE   0
2810  2C61  00                  .BYTE   0
2811
2812                                    ;DEVICE HANDLER FOR THE 'NULL' OR EMPTY DEVICE
2813
2814  2C62  A9 1A       NULLI:  LDA#    EOFCHR                  ;GET THE END OF FILE CHARACTER
2815  2C64  18          DUMMY:  CLC                             ;ALWAYS INDICATE SUCCESS
2816  2C65  60                  RTS                           ;AND RETURN
```

```
                        .PAGE
                        ;I-O ROUTINES FOR KIM BASED SYSTEMS
                        ;THESE ROUTINES DEFINE KIM MONITOR ENTRY POINTS

        1E5A            .DEF    GETCH=$1E5A     ;ADDRESS OF CHARACTER INPUT ROUTINE
        1EA0            .DEF    OUTCH=$1EA0     ;CHARACTER OUTPUT ROUTINE

                        ;ROUTINE TO OUTPUT ONE CHAR TO THE TERMINAL

2C66 C9 0D    TVOUT:    CMP#    $0D             ;IS IT A CR
2C68 F0 05              BEQ     FILOUT          ;BRANCH IF IT IS
2C6A 20 A01E            JSR     OUTCH           ;PRINT WITHOUT FILL
2C6D 18                 CLC                     ;INDICATE SUCCESS
2C6E 60                 RTS
2C6F 20 A01E  FILOUT:   JSR     OUTCH           ;PRINT CHARACTER
2C72 A9 7F              LDA#    $7F             ;GET FILL CHARACTER
2C74 20 A01E            JSR     OUTCH           ;PRINT IT
2C77 18                 CLC                     ;INDICATE SUCCESS
2C78 60                 RTS

                        ;ROUTINE TO INPUT ONE CHARACTER

2C79 20 5A1E  KEYIN:    JSR     GETCH           ;GET A CHARACTER FROM TTY
2C7C 18       TRTS:     CLC                     ;INICATE SUCCESS
2C7D 60                 RTS


2C7E 00       HAKSPA:   .BYTE                   ;TWENTY FIVE BYTES OF I/O HACK SPACE
        2C97            .LOC    HAKSPA+25       ;
                        ;CASSETTE I/O ROUTINES ARE LEFT TO USER SINCE THEY ARE
                        ;DEVICE DEPENDENT


        2C7C            .DEF    C0BYTI=TRTS     ;DEFINES INPUT ROUTINE FOR CASSETTE # 0
        2C7C            .DEF    C1BYTI=TRTS     ;DEFINES INPUT ROUTINE FOR CASSETTE #1
        2C7C            .DEF    C0BYTO=TRTS     ;DEFINES OUTPUT ROUTINE FOR CASSETTE #0
        2C7C            .DEF    C1BYTO=TRTS     ;DEFINES OUTPUT ROUTINE FOR CASSETTE #1
```



```
                        .PAGE


        2C7C            .DEF    C0OPNI=TRTS     ;DEFINES ROUTINE TO OPEN CASSETTE #0
        2C7C            .DEF    C1OPNI=TRTS     ;DEFINES ROUTINE TO OPEN CASSETTE #1
        2C7C            .DEF    C0OPNO=TRTS     ;DEFINES ROUTINE TO OPEN CASS #0 FOR OUTPUT
        2C7C            .DEF    C1OPNO=TRTS     ;DEFINE ROUTINE TO OPEN CASS #1 FOR OUTPUT
        2C7C            .DEF    C0CLSI=TRTS     ;DEFINE CLOSE FOR INPUT CASSETTE #0
        2C7C            .DEF    C1CLSI=TRTS     ;DEFINE CLOSE FOR INPUT CASSETTE #1
        2C7C            .DEF    C0CLSO=TRTS     ;DEFINE CLOSE FOR OUTPUT CASSETTE #0
        2C7C            .DEF    C1CLSO=TRTS     ;DEFINE CLOSE FOR OUTPUT CASSETTE #1
        2586            .DEF    CONINI=RTS1     ;TV NEEDS NO INITIALIZATION IN THIS SYSTEM


                        ;DEFINE ROUTINE TO READ FROM EDITING BUFFER

        3669            .DEF    EDBUFFRD=$3669
        3928            .DEF    EDINIT1=$3928   ;ROUTINE INITIALIZE EDITOR
```

```
2882                                    .PAGE
2883                                    ;HANDLER PRODUCES AUTO PAGING AND SIMULATES FORM FEEDS
2884  2C97  00        COUNT:   .BYTE                  ;COUNTER
2885
2886
2887  2C98  C9 0A    LPTSIM:  CMP#    $0A            ;IS THE CHARACTER A LINE FEED
2888  2C9A  D0 08             BNE     NOTLF          ;BRANCH IF NOT
2889  2C9C  CE 972C           DEC     COUNT          ;COUNT THE LINE FEED
2890  2C9F  D0 03             BNE     NOTLF          ;ARE WE AT THE BOTTOM OF THE PAGE?
2891  2CA1  20 BF2C  (local)  JSR     SIXLIN         ;PRINT SIX CRLF IF WE ARE
2892  2CA4  C9 0C    NOTLF:   CMP#    $0C            ;IS IT A FORM FEED
2893  2CA6  D0 12             BNE     PRIOUT         ;PRINT IF NOT
2894  2CA8  48                PHA                    ;SAVE AC
2895  2CA9  AD 972C           LDA     COUNT          ;CHECK THE COUNT
2896  2CAC  F0 0C    (local)  BEQ     PRIOUT1        ;EXIT IF COUNT IS ALREADY ZERO
2897  2CAE  20 D82C  LPLOP:   JSR     WCRLF          ;PRINT CRLF
2898  2CB1  CE 972C           DEC     COUNT          ;COUNT AND PRINT REMAINING SPACE
2899  2CB4  D0 F8             BNE     LPLOP          ;LOOP UNTIL COUNT ZERO
2900  2CB6  20 BF2C  (local)  JSR     SIXLIN         ;NOW PRINT THE SIX BETWEEN PAGES
2901  2CB9  68        PROUT1:  PLA                    ;RESTORE AC
2902  2CBA  20 662C  PRIOUT:  JSR     TVOUT          ;PRINT
2903  2CBD  18        (local) CLC                    ;INDICATE SUCCESS
2904  2CBE  60                RTS
2905  2CBF  48        SIXLIN:  PHA                    ;SAVE AC
2906  2CC0  A9 06              LDA#    06             ;GET COUNT
2907  2CC2  8D 972C           STA     COUNT          ;PLACE IN COUNT REGISTER
2908  2CC5  20 D82C  SIXLOP:  JSR     WCRLF          ;PRINT CRLF
2909  2CC8  CE 972C           DEC     COUNT          ;COUNT
2910  2CCB  D0 F8             BNE     SIXLOP         ;LOOP UNTIL SIX
2911  2CCD  20 D22C           JSR     LPTRES         ;RESET COUNTER
2912  2CD0  68                PLA                    ;RESTORE AC
2913  2CD1  60                RTS
2914                                   (ENTRY)
2915  2CD2  A9 3C    LPTRES:  LDA#    60             ;SIXTY LINE PER PAGE
2916  2CD4  8D 972C           STA     COUNT          ;STORE IN COUNTER
2917  2CD7  60                RTS
2918
2919  2CD8  A9 0D    WCRLF:   LDA#    $0D            ;GET CR
2920  2CDA  20 BA2C           JSR     PRIOUT         ;PRINT IT
2921  2CDD  A9 0A             LDA#    $0A            ;GET LF
2922  2CDF  20 BA2C           JSR     PRIOUT         ;PRINT IT
2923  2CE2  60                RTS
```

NAM

LPTRES: 2760/2769

LPTSIM: 2723/2732

This has been replaced with H-14 printer driver

~~PS: 16, 97, 99, 410, 412, 4~~

OPTEND [3902]

SYMTAB 1198[C] 1200[C] 1461[C] 1463[C] [3909]

LOCH: 534[C] 615[C] 627[C] 651[C] 684[C] 686[C]
696[C] 848 915[C] 919[C] [3912]

LOCL: 533[C] 613[C] 624[C] 649[C] 681[C] 683[C]
699[C] 846 913[C] 922[C] [3913]

SYMCLR 512[C] [3914]

42-381 50 SHEETS 5 SQUARE
42-382 100 SHEETS 5 SQUARE
42-389 200 SHEETS 5 SQUARE
Made in U.S.A.
NATIONAL

```
2924                            .PAGE                   ;'PERMANENT SYMBOL TABLE'
2925
2926
2927
2928                            ;HERE IS THE PERMANENT SYMBOL TABLE FOR THE ASSEMBLER.
2929                            ;NEW ENTRIES MAY BE ADDED EASILY ADDED AT THE END.
2930                            ;THIS TABLE IS NOT PURGED BETWEEN PASSES.
2931
2932
2933
2934                            ;HERE IS THE OPCODE TABLE.
2935
2936             2CE3           .DEF    OPCTAB=.
2937
2938
2939  2CE3       41            .ASCII  'ADC#'
2940  2CE4       44
2941  2CE5       43
2942  2CE6       23
2943
2944  2CE7       E0            .BYTE   OPCCL0
2945  2CE8.      69            .BYTE   $69             ;THE OPCODE FOR 'IMMEDIATE MODE'.
2946
2947  2CE9       41            .ASCII  'ADC'
2948  2CEA       44
2949  2CEB       43
2950
2951  2CEC       A1            .BYTE   OPCCL1
2952  2CED       65            .BYTE   $65             ;THE OPCODE IF 'PAGE-ZERO' ADDRESSING
2953  2CEE       6D            .BYTE   $6D             ;THE OPCODE IF 'ABSOLUTE' ADDRESSING
2954
2955  2CEF       41            .ASCII  'ADCX'
2956  2CF0       44
2957  2CF1       43
2958  2CF2       58
2959
2960  2CF3       A1            .BYTE   OPCCL1
2961  2CF4       75            .BYTE   $75             ;THE OPCODE IF 'PAGE-ZERO' ADDRESSING
2962  2CF5       7D            .BYTE   $7D             ;THE OPCODE IF 'ABSOLUTE' ADDRESSING
2963
2964  2CF6       41            .ASCII  'ADCY'
2965  2CF7       44
2966  2CF8       43
2967  2CF9       59
2968
2969  2CFA       E2            .BYTE   OPCCL2
2970  2CFB       79            .BYTE   $79
2971
2972  2CFC       41            .ASCII  'ADCX@'
2973  2CFD       44
2974  2CFE       43
2975  2CFF       58
2976  2D00       40
2977
```

BUT BESURE TO CHANGE ALL REF'S TO OPEND, SYMTAB, LOCH, LOCL & SYMCLR (SEE NOTE ON PG 66) (AFTER PG 65-17)

```
2978  2D01  E3              .BYTE   OPCCL3
2979  2D02  61              .BYTE   $61
2980
2981  2D03  41              .ASCII  'ADC@Y'
2982  2D04  44
2983  2D05  43
2984  2D06  40
2985  2D07  59
2986
2987  2D08  E3              .BYTE   OPCCL3
2988  2D09  71              .BYTE   $71
2989
2990  2D0A  41              .ASCII  'AND#'
2991  2D0B  4E
2992  2D0C  44
2993  2D0D  23
2994
2995  2D0E  E0              .BYTE   OPCCL0
2996  2D0F  29              .BYTE   $29
2997
2998  2D10  41              .ASCII  'AND'
2999  2D11  4E
3000  2D12  44
3001
3002  2D13  A1              .BYTE   OPCCL1
3003  2D14  25              .BYTE   $25
3004  2D15  2D              .BYTE   $2D
3005
3006  2D16  41              .ASCII  'ANDX'
3007  2D17  4E
3008  2D18  44
3009  2D19  58
3010
3011  2D1A  A1              .BYTE   OPCCL1
3012  2D1B  35              .BYTE   $35
3013  2D1C  3D              .BYTE   $3D
3014
3015  2D1D  41              .ASCII  'ANDY'
3016  2D1E  4E
3017  2D1F  44
3018  2D20  59
3019
3020  2D21  E2              .BYTE   OPCCL2
3021  2D22  39              .BYTE   $39
3022
3023  2D23  41              .ASCII  'ANDX@'
3024  2D24  4E
3025  2D25  44
3026  2D26  58
3027  2D27  40
3028
3029  2D28  E3              .BYTE   OPCCL3
3030  2D29  21              .BYTE   $21
3031
```

```
3032  2D2A  41          .ASCII  'AND@Y'
3033  2D2B  4E
3034  2D2C  44
3035  2D2D  40
3036  2D2E  59
3037
3038  2D2F  E3          .BYTE   OPCCL3
3039  2D30  31          .BYTE   $31
3040
3041  2D31  41          .ASCII  'ASLA'
3042  2D32  53
3043  2D33  4C
3044  2D34  41
3045
3046  2D35  E5          .BYTE   OPCCL5
3047  2D36  0A          .BYTE   $0A
3048
3049  2D37  41          .ASCII  'ASL'
3050  2D38  53
3051  2D39  4C
3052
3053  2D3A  A1          .BYTE   OPCCL1
3054  2D3B  06          .BYTE   $06
3055  2D3C  0E          .BYTE   $0E
3056
3057  2D3D  41          .ASCII  'ASLX'
3058  2D3E  53
3059  2D3F  4C
3060  2D40  58
3061
3062  2D41  A1          .BYTE   OPCCL1
3063  2D42  16          .BYTE   $16
3064  2D43  1E          .BYTE   $1E
3065
3066  2D44  42          .ASCII  'BCC'
3067  2D45  43
3068  2D46  43
3069
3070  2D47  E4          .BYTE   OPCCL4
3071  2D48  90          .BYTE   $90
3072
3073  2D49  42          .ASCII  'BCS'
3074  2D4A  43
3075  2D4B  53
3076
3077  2D4C  E4          .BYTE   OPCCL4
3078  2D4D  B0          .BYTE   $B0
3079
3080  2D4E  42          .ASCII  'BEQ'
3081  2D4F  45
3082  2D50  51
3083
3084  2D51  E4          .BYTE   OPCCL4
3085  2D52  F0          .BYTE   $F0
```

```
3086
3087  2D53  42              .ASCII  'BIT'
3088  2D54  49
3089  2D55  54
3090
3091  2D56  A1              .BYTE   OPCCL1
3092  2D57  24              .BYTE   $24
3093  2D58  2C              .BYTE   $2C
3094
3095  2D59  42              .ASCII  'BMI'
3096  2D5A  4D
3097  2D5B  49
3098
3099  2D5C  E4              .BYTE   OPCCL4
3100  2D5D  30              .BYTE   $30
3101
3102  2D5E  42              .ASCII  'BNE'
3103  2D5F  4E
3104  2D60  45
3105
3106  2D61  E4              .BYTE   OPCCL4
3107  2D62. D0              .BYTE   $D0
3108
3109  2D63  42              .ASCII  'BPL'
3110  2D64  50
3111  2D65  4C
3112
3113  2D66  E4              .BYTE   OPCCL4
3114  2D67  10              .BYTE   $10
3115
3116  2D68  42              .ASCII  'BRK'
3117  2D69  52
3118  2D6A  4B
3119
3120  2D6B  E5              .BYTE   OPCCL5
3121  2D6C  00              .BYTE   $00
3122
3123  2D6D  42              .ASCII  'BVC'
3124  2D6E  56
3125  2D6F  43
3126
3127  2D70  E4              .BYTE   OPCCL4
3128  2D71  50              .BYTE   $50
3129
3130  2D72  42              .ASCII  'BVS'
3131  2D73  56
3132  2D74  53
3133
3134  2D75  E4              .BYTE   OPCCL4
3135  2D76  70              .BYTE   $70
3136
3137  2D77  43              .ASCII  'CLC'
3138  2D78  4C
3139  2D79  43
```

```
3140
3141  2D7A  E5          .BYTE   OPCCL5
3142  2D7B  18          .BYTE   $18
3143
3144  2D7C  43          .ASCII  'CLD'
3145  2D7D  4C
3146  2D7E  44
3147
3148  2D7F  E5          .BYTE   OPCCL5
3149  2D80  D8          .BYTE   $D8
3150
3151  2D81  43          .ASCII  'CLI'
3152  2D82  4C
3153  2D83  49
3154
3155  2D84  E5          .BYTE   OPCCL5
3156  2D85  58          .BYTE   $58
3157
3158  2D86  43          .ASCII  'CLV'
3159  2D87  4C
3160  2D88  56
3161
3162  2D89  E5          .BYTE   OPCCL5
3163  2D8A  B8          .BYTE   $B8
3164
3165  2D8B  43          .ASCII  'CMP#'
3166  2D8C  4D
3167  2D8D  50
3168  2D8E  23
3169
3170  2D8F  E0          .BYTE   OPCCL0
3171  2D90  C9          .BYTE   $C9
3172
3173  2D91  43          .ASCII  'CMP'
3174  2D92  4D
3175  2D93  50
3176
3177  2D94  A1          .BYTE   OPCCL1
3178  2D95  C5          .BYTE   $C5
3179  2D96  CD          .BYTE   $CD
3180
3181  2D97  43          .ASCII  'CMPX'
3182  2D98  4D
3183  2D99  50
3184  2D9A  58
3185
3186  2D9B  A1          .BYTE   OPCCL1
3187  2D9C  D5          .BYTE   $D5
3188  2D9D  DD          .BYTE   $DD
3189
3190  2D9E  43          .ASCII  'CMPY'
3191  2D9F  4D
3192  2DA0  50
3193  2DA1  59
```

```
3194
3195  2DA2  E2          .BYTE   OPCCL2
3196  2DA3  D9          .BYTE   $D9
3197
3198  2DA4  43          .ASCII  'CMPX@'
3199  2DA5  4D
3200  2DA6  50
3201  2DA7  58
3202  2DA8  40
3203
3204  2DA9  E3          .BYTE   OPCCL3
3205  2DAA  C1          .BYTE   $C1
3206
3207  2DAB  43          .ASCII  'CMP@Y'
3208  2DAC  4D
3209  2DAD  50
3210  2DAE  40
3211  2DAF  59
3212
3213  2DB0  E3          .BYTE   OPCCL3
3214  2DB1  D1          .BYTE   $D1
3215
3216  2DB2  43          .ASCII  'CPX#'
3217  2DB3  50
3218  2DB4  58
3219  2DB5  23
3220
3221  2DB6  E0          .BYTE   OPCCL0
3222  2DB7  E0          .BYTE   $E0
3223
3224  2DB8  43          .ASCII  'CPX'
3225  2DB9  50
3226  2DBA  58
3227
3228  2DBB  A1          .BYTE   OPCCL1
3229  2DBC  E4          .BYTE   $E4
3230  2DBD  EC          .BYTE   $EC
3231
3232  2DBE  43          .ASCII  'CPY#'
3233  2DBF  50
3234  2DC0  59
3235  2DC1  23
3236
3237  2DC2  E0          .BYTE   OPCCL0
3238  2DC3  C0          .BYTE   $C0
3239
3240  2DC4  43          .ASCII  'CPY'
3241  2DC5  50
3242  2DC6  59
3243
3244  2DC7  A1          .BYTE   OPCCL1
3245  2DC8  C4          .BYTE   $C4
3246  2DC9  CC          .BYTE   $CC
3247
```

```
3248  2DCA  44          .ASCII  'DEC'
3249  2DCB  45
3250  2DCC  43
3251
3252  2DCD  A1          .BYTE   OPCCL1
3253  2DCE  C6          .BYTE   $C6
3254  2DCF  CE          .BYTE   $CE
3255
3256  2DD0  44          .ASCII  'DECX'
3257  2DD1  45
3258  2DD2  43
3259  2DD3  58
3260
3261  2DD4  A1          .BYTE   OPCCL1
3262  2DD5  D6          .BYTE   $D6
3263  2DD6  DE          .BYTE   $DE
3264
3265  2DD7  44          .ASCII  'DEX'
3266  2DD8  45
3267  2DD9  58
3268
3269  2DDA  E5          .BYTE   OPCCL5
3270  2DDB  CA          .BYTE   $CA
3271
3272  2DDC  44          .ASCII  'DEY'
3273  2DDD  45
3274  2DDE  59
3275
3276  2DDF  E5          .BYTE   OPCCL5
3277  2DE0  88          .BYTE   $88
3278
3279  2DE1  45          .ASCII  'EOR#'
3280  2DE2  4F
3281  2DE3  52
3282  2DE4  23
3283
3284  2DE5  E0          .BYTE   OPCCL0
3285  2DE6  49          .BYTE   $49
3286
3287  2DE7  45          .ASCII  'EOR'
3288  2DE8  4F
3289  2DE9  52
3290
3291  2DEA  A1          .BYTE   OPCCL1
3292  2DEB  45          .BYTE   $45
3293  2DEC  4D          .BYTE   $4D
3294
3295  2DED  45          .ASCII  'EORX'
3296  2DEE  4F.
3297  2DEF  52
3298  2DF0  58
3299
3300  2DF1  A1          .BYTE   OPCCL1
3301  2DF2  55          .BYTE   $55
```

```
3302  2DF3  5D            .BYTE   $5D
3303
3304  2DF4  45            .ASCII  'EORY'
3305  2DF5  4F
3306  2DF6  52
3307  2DF7  59
3308
3309  2DF8  E2            .BYTE   OPCCL2
3310  2DF9  59            .BYTE   $59
3311
3312  2DFA  45            .ASCII  'EORX@'
3313  2DFB  4F
3314  2DFC  52
3315  2DFD  58
3316  2DFE  40
3317
3318  2DFF  E3            .BYTE   OPCCL3
3319  2E00  41            .BYTE   $41
3320
3321  2E01  45            .ASCII  'EOR@Y'
3322  2E02  4F
3323  2E03  52
3324  2E04  40
3325  2E05  59
3326
3327  2E06  E3            .BYTE   OPCCL3
3328  2E07  51            .BYTE   $51
3329
3330  2E08  49            .ASCII  'INC'
3331  2E09  4E
3332  2E0A  43
3333
3334  2E0B  A1            .BYTE   OPCCL1
3335  2E0C  E6            .BYTE   $E6
3336  2E0D  EE            .BYTE   $EE
3337
3338  2E0E  49            .ASCII  'INCX'
3339  2E0F  4E
3340  2E10  43
3341  2E11  58
3342
3343  2E12  A1            .BYTE   OPCCL1
3344  2E13  F6            .BYTE   $F6
3345  2E14  FE            .BYTE   $FE
3346
3347  2E15  49            .ASCII  'INX'
3348  2E16  4E
3349  2E17  58
3350
3351  2E18  E5            .BYTE   OPCCL5
3352  2E19  E8            .BYTE   $E8
3353
3354  2E1A  49            .ASCII  'INY'
3355  2E1B  4E
```

```
3356  2E1C  59
3357
3358  2E1D  E5              .BYTE   OPCCL5
3359  2E1E  C8              .BYTE   $C8
3360
3361  2E1F  4A              .ASCII  'JMP'
3362  2E20  4D
3363  2E21  50
3364
3365  2E22  E2              .BYTE   OPCCL2
3366  2E23  4C              .BYTE   $4C
3367
3368  2E24  4A              .ASCII  'JMP@'
3369  2E25  4D
3370  2E26  50
3371  2E27  40
3372
3373  2E28  E2              .BYTE   OPCCL2
3374  2E29  6C              .BYTE   $6C
3375
3376  2E2A  4A              .ASCII  'JSR'
3377  2E2B  53
3378  2E2C  52
3379
3380  2E2D  E2              .BYTE   OPCCL2
3381  2E2E  20              .BYTE   $20
3382
3383  2E2F  4C              .ASCII  'LDA#'
3384  2E30  44
3385  2E31  41
3386  2E32  23
3387
3388  2E33  E0              .BYTE   OPCCL0
3389  2E34  A9              .BYTE   $A9
3390
3391  2E35  4C              .ASCII  'LDA'
3392  2E36  44
3393  2E37  41
3394
3395  2E38  A1              .BYTE   OPCCL1
3396  2E39  A5              .BYTE   $A5
3397  2E3A  AD              .BYTE   $AD
3398
3399  2E3B  4C              .ASCII  'LDAX'
3400  2E3C  44
3401  2E3D  41
3402  2E3E  58
3403
3404  2E3F  A1              .BYTE   OPCCL1
3405  2E40  B5              .BYTE   $B5
3406  2E41  BD              .BYTE   $BD
3407
3408  2E42  4C              .ASCII  'LDAY'
3409  2E43  44
```

```
3410  2E44  41
3411  2E45  59
3412
3413  2E46  E2              .BYTE   OPCCL2
3414  2E47  B9              .BYTE   $B9
3415
3416  2E48  4C              .ASCII  'LDAX@'
3417  2E49  44
3418  2E4A  41
3419  2E4B  58
3420  2E4C  40
3421
3422  2E4D  E3              .BYTE   OPCCL3
3423  2E4E  A1              .BYTE   $A1
3424
3425  2E4F  4C              .ASCII  'LDA@Y'
3426  2E50  44
3427  2E51  41
3428  2E52  40
3429  2E53  59
3430
3431  2E54  E3              .BYTE   OPCCL3
3432  2E55  B1              .BYTE   $B1
3433
3434  2E56  4C              .ASCII  'LDX#'
3435  2E57  44
3436  2E58  58
3437  2E59  23
3438
3439  2E5A  E0              .BYTE   OPCCL0
3440  2E5B  A2              .BYTE   $A2
3441
3442  2E5C  4C              .ASCII  'LDX'
3443  2E5D  44
3444  2E5E  58
3445
3446  2E5F  A1              .BYTE   OPCCL1
3447  2E60  A6              .BYTE   $A6
3448  2E61  AE              .BYTE   $AE
3449
3450  2E62  4C              .ASCII  'LDXY'
3451  2E63  44
3452  2E64  58
3453  2E65  59
3454
3455  2E66  A1              .BYTE   OPCCL1
3456  2E67  B6              .BYTE   $B6
3457  2E68  BE              .BYTE   $BE
3458
3459  2E69  4C              .ASCII  'LDY#'
3460  2E6A  44
3461  2E6B  59
3462  2E6C  23
3463
```

```
3464  2E6D  E0          .BYTE   OPCCL0
3465  2E6E  A0          .BYTE   $A0
3466
3467  2E6F  4C          .ASCII  'LDY'
3468  2E70  44
3469  2E71  59
3470
3471  2E72  A1          .BYTE   OPCCL1
3472  2E73  A4          .BYTE   $A4
3473  2E74  AC          .BYTE   $AC
3474
3475  2E75  4C          .ASCII  'LDYX'
3476  2E76  44
3477  2E77  59
3478  2E78  58
3479
3480  2E79  A1          .BYTE   OPCCL1
3481  2E7A  B4          .BYTE   $B4
3482  2E7B  BC          .BYTE   $BC
3483
3484  2E7C  4C          .ASCII  'LSRA'
3485  2E7D  53
3486  2E7E  52
3487  2E7F  41
3488
3489  2E80  E5          .BYTE   OPCCL5
3490  2E81  4A          .BYTE   $4A
3491
3492  2E82  4C          .ASCII  'LSR'
3493  2E83  53
3494  2E84  52
3495
3496  2E85  A1          .BYTE   OPCCL1
3497  2E86  46          .BYTE   $46
3498  2E87  4E          .BYTE   $4E
3499
3500  2E88  4C          .ASCII  'LSRX'
3501  2E89  53
3502  2E8A  52
3503  2E8B  58
3504
3505  2E8C  A1          .BYTE   OPCCL1
3506  2E8D  56          .BYTE   $56
3507  2E8E  5E          .BYTE   $5E
3508
3509  2E8F  4E          .ASCII  'NOP'
3510  2E90  4F
3511  2E91  50
3512
3513  2E92  E5          .BYTE   OPCCL5
3514  2E93  EA          .BYTE   $EA
3515
3516  2E94  4F          .ASCII  'ORA#'
3517  2E95  52
```

```
3518  2E96  41
3519  2E97  23
3520
3521  2E98  E0              .BYTE   OPCCL0
3522  2E99  09              .BYTE   $09
3523
3524  2E9A  4F              .ASCII  'ORA'
3525  2E9B  52
3526  2E9C  41
3527
3528  2E9D  A1              .BYTE   OPCCL1
3529  2E9E  05              .BYTE   $05
3530  2E9F  0D              .BYTE   $0D
3531
3532  2EA0  4F              .ASCII  'ORAX'
3533  2EA1  52
3534  2EA2  41
3535  2EA3  58
3536
3537  2EA4  A1              .BYTE   OPCCL1
3538  2EA5  15              .BYTE   $15
3539  2EA6  1D              .BYTE   $1D
3540
3541  2EA7  4F              .ASCII  'ORAY'
3542  2EA8  52
3543  2EA9  41
3544  2EAA  59
3545
3546  2EAB  E2              .BYTE   OPCCL2
3547  2EAC  19              .BYTE   $19
3548
3549  2EAD  4F              .ASCII  'ORAX@'
3550  2EAE  52
3551  2EAF  41
3552  2EB0  58
3553  2EB1  40
3554
3555  2EB2  E3              .BYTE   OPCCL3
3556  2EB3  01              .BYTE   $01
3557
3558  2EB4  4F              .ASCII  'ORA@Y'
3559  2EB5  52
3560  2EB6  41
3561  2EB7  40
3562  2EB8  59
3563
3564  2EB9  E3              .BYTE   OPCCL3
3565  2EBA  11              .BYTE   $11
3566
3567  2EBB  50              .ASCII  'PHA'
3568  2EBC  48
3569  2EBD  41
3570
3571  2EBE  E5              .BYTE   OPCCL5
```

```
3572  2EBF  48          .BYTE  $48
3573
3574  2EC0  50          .ASCII 'PHP'
3575  2EC1  48
3576  2EC2  50
3577
3578  2EC3  E5          .BYTE  OPCCL5
3579  2EC4  08          .BYTE  $08
3580
3581  2EC5  50          .ASCII 'PLA'
3582  2EC6  4C
3583  2EC7  41
3584
3585  2EC8  E5          .BYTE  OPCCL5
3586  2EC9  68          .BYTE  $68
3587
3588  2ECA  50          .ASCII 'PLP'
3589  2ECB  4C
3590  2ECC  50
3591
3592  2ECD  E5          .BYTE  OPCCL5
3593  2ECE  28          .BYTE  $28
3594
3595  2ECF  52          .ASCII 'ROLA'
3596  2ED0  4F
3597  2ED1  4C
3598  2ED2  41
3599
3600  2ED3  E5          .BYTE  OPCCL5
3601  2ED4  2A          .BYTE  $2A
3602
3603  2ED5  52          .ASCII 'ROL'
3604  2ED6  4F
3605  2ED7  4C
3606
3607  2ED8  A1          .BYTE  OPCCL1
3608  2ED9  26          .BYTE  $26
3609  2EDA  2E          .BYTE  $2E
3610
3611  2EDB  52          .ASCII 'ROLX'
3612  2EDC  4F
3613  2EDD  4C
3614  2EDE  58
3615
3616  2EDF  A1          .BYTE  OPCCL1
3617  2EE0  36          .BYTE  $36
3618  2EE1  3E          .BYTE  $3E
3619
3620  2EE2  52          .ASCII 'RTI'
3621  2EE3  54
3622  2EE4  49
3623
3624  2EE5  E5          .BYTE  OPCCL5
3625  2EE6  40          .BYTE  $40
```

```
3626
3627  2EE7  52              .ASCII  'RTS'
3628  2EE8  54
3629  2EE9  53
3630
3631  2EEA  E5              .BYTE   OPCCL5
3632  2EEB  60              .BYTE   $60
3633
3634  2EEC  53              .ASCII  'SBC#'
3635  2EED  42
3636  2EEE  43
3637  2EEF  23
3638
3639  2EF0  E0              .BYTE   OPCCL0
3640  2EF1  E9              .BYTE   $E9
3641
3642  2EF2  53              .ASCII  'SBC'
3643  2EF3  42
3644  2EF4  43
3645
3646  2EF5  A1              .BYTE   OPCCL1
3647  2EF6  E5              .BYTE   $E5
3648  2EF7  ED              .BYTE   $ED
3649
3650  2EF8  53              .ASCII  'SBCX'
3651  2EF9  42
3652  2EFA  43
3653  2EFB  58
3654
3655  2EFC  A1              .BYTE   OPCCL1
3656  2EFD  F5              .BYTE   $F5
3657  2EFE  FD              .BYTE   $FD
3658
3659  2EFF  53              .ASCII  'SBCY'
3660  2F00  42
3661  2F01  43
3662  2F02  59
3663
3664  2F03  E2              .BYTE   OPCCL2
3665  2F04  F9              .BYTE   $F9
3666
3667  2F05  53              .ASCII  'SBCX@'
3668  2F06  42
3669  2F07  43
3670  2F08  58
3671  2F09  40
3672
3673  2F0A  E3              .BYTE   OPCCL3
3674  2F0B  E1              .BYTE   $E1
3675
3676  2F0C  53              .ASCII  'SBC@Y'
3677  2F0D  42
3678  2F0E  43
3679  2F0F  40
```

```
3680  2F10  59
3681
3682  2F11  E3              .BYTE   OPCCL3
3683  2F12  F1              .BYTE   $F1
3684
3685  2F13  53              .ASCII  'SEC'
3686  2F14  45
3687  2F15  43
3688
3689  2F16  E5              .BYTE   OPCCL5
3690  2F17  38              .BYTE   $38
3691
3692  2F18  53              .ASCII  'SED'
3693  2F19  45
3694  2F1A  44
3695
3696  2F1B  E5              .BYTE   OPCCL5
3697  2F1C  F8              .BYTE   $F8
3698
3699  2F1D  53              .ASCII  'SEI'
3700  2F1E  45
3701  2F1F  49
3702
3703  2F20  E5              .BYTE   OPCCL5
3704  2F21  78              .BYTE   $78
3705
3706  2F22  53              .ASCII  'STA'
3707  2F23  54
3708  2F24  41
3709
3710  2F25  A1              .BYTE   OPCCL1
3711  2F26  85              .BYTE   $85
3712  2F27  8D              .BYTE   $8D
3713
3714  2F28  53              .ASCII  'STAX'
3715  2F29  54
3716  2F2A  41
3717  2F2B  58
3718
3719  2F2C  A1              .BYTE   OPCCL1
3720  2F2D  95              .BYTE   $95
3721  2F2E  9D              .BYTE   $9D
3722
3723  2F2F  53              .ASCII  'STAY'
3724  2F30  54
3725  2F31  41
3726  2F32  59
3727
3728  2F33  E2              .BYTE   OPCCL2
3729  2F34  99              .BYTE   $99
3730
3731  2F35  53              .ASCII  'STAX@'
3732  2F36  54
3733  2F37  41
```

```
3734  2F38  58
3735  2F39  40
3736
3737  2F3A  E3              .BYTE   OPCCL3
3738  2F3B  81              .BYTE   $81
3739
3740  2F3C  53              .ASCII  'STA@Y'
3741  2F3D  54
3742  2F3E  41
3743  2F3F  40
3744  2F40  59
3745
3746  2F41  E3              .BYTE   OPCCL3
3747  2F42  91              .BYTE   $91
3748
3749  2F43  53              .ASCII  'STX'
3750  2F44  54
3751  2F45  58
3752
3753  2F46  A1              .BYTE   OPCCL1
3754  2F47  86              .BYTE   $86
3755  2F48  8E              .BYTE   $8E
3756
3757  2F49  53              .ASCII  'STXY'
3758  2F4A  54
3759  2F4B  58
3760  2F4C  59
3761
3762  2F4D  E3              .BYTE   OPCCL3
3763  2F4E  96              .BYTE   $96
3764
3765  2F4F  53              .ASCII  'STY'
3766  2F50  54
3767  2F51  59
3768
3769  2F52  A1              .BYTE   OPCCL1
3770  2F53  84              .BYTE   $84
3771  2F54  8C              .BYTE   $8C
3772
3773  2F55  53              .ASCII  'STYX'
3774  2F56  54
3775  2F57  59
3776  2F58  58
3777
3778  2F59  E3              .BYTE   OPCCL3
3779  2F5A  94              .BYTE   $94
3780
3781  2F5B  54              .ASCII  'TAX'
3782  2F5C  41
3783  2F5D  58
3784
3785  2F5E  E5              .BYTE   OPCCL5
3786  2F5F  AA              .BYTE   $AA
3787
```

```
3788  2F60  54              .ASCII  'TAY'
3789  2F61  41
3790  2F62  59
3791
3792  2F63  E5              .BYTE   OPCCL5
3793  2F64  A8              .BYTE   $A8
3794
3795  2F65  54              .ASCII  'TSX'
3796  2F66  53
3797  2F67  58
3798
3799  2F68  E5              .BYTE   OPCCL5
3800  2F69  BA              .BYTE   $BA
3801
3802  2F6A  54              .ASCII  'TXA'
3803  2F6B  58
3804  2F6C  41
3805
3806  2F6D  E5              .BYTE   OPCCL5
3807  2F6E  8A              .BYTE   $8A
3808
3809  2F6F  54              .ASCII  'TXS'
3810  2F70  58
3811  2F71  53
3812
3813  2F72  E5              .BYTE   OPCCL5
3814  2F73  9A              .BYTE   $9A
3815
3816  2F74  54              .ASCII  'TYA'
3817  2F75  59
3818  2F76  41
3819
3820  2F77  E5              .BYTE   OPCCL5
3821  2F78  98              .BYTE   $98
3822
3823  2F79  2E              .ASCII  '.BYTE'
3824  2F7A  42
3825  2F7B  59
3826  2F7C  54
3827  2F7D  45
3828
3829  2F7E  A6              .BYTE   OPCCL6
3830  2F7F  23              .HBYTE  PBYTE
3831  2F80  67              .BYTE   PBYTE
3832
3833  2F81  2E              .ASCII  '.HBYTE'
3834  2F82  48
3835  2F83  42
3836  2F84  59
3837  2F85  54
3838  2F86  45
3839
3840  2F87  A6              .BYTE   OPCCL6
3841  2F88  23              .HBYTE  PHBYTE
```

```
3842  2F89  5F              .BYTE   PHBYTE
3843
3844  2F8A  2E              .ASCII  '.WORD'
3845  2F8B  57
3846  2F8C  4F
3847  2F8D  52
3848  2F8E  44
3849
3850  2F8F  A6              .BYTE   OPCCL6
3851  2F90  23              .HBYTE  PWORD
3852  2F91  73              .BYTE   PWORD
3853
3854  2F92  2E              .ASCII  '.PAGE'
3855  2F93  50
3856  2F94  41
3857  2F95  47
3858  2F96  45
3859
3860  2F97  A6              .BYTE   OPCCL6
3861  2F98  23              .HBYTE  PPAGE
3862  2F99  A8              .BYTE   PPAGE
3863
3864  2F9A  2E              .ASCII  '.ASCII'
3865  2F9B  41
3866  2F9C  53
3867  2F9D  43
3868  2F9E  49
3869  2F9F  49
3870
3871  2FA0  A6              .BYTE   OPCCL6
3872  2FA1  23              .HBYTE  PASCII
3873  2FA2  C1              .BYTE   PASCII
3874
3875  2FA3  2E              .ASCII  '.LOC'
3876  2FA4  4C
3877  2FA5  4F
3878  2FA6  43
3879
3880  2FA7  A6              .BYTE   OPCCL6
3881  2FA8  23              .HBYTE  PLOC
3882  2FA9  83              .BYTE   PLOC
3883
3884  2FAA  2E              .ASCII  '.DEF'
3885  2FAB  44
3886  2FAC  45
3887  2FAD  46
3888
3889  2FAE  A6              .BYTE   OPCCL6
3890  2FAF  24              .HBYTE  PDEF
3891  2FB0  08              .BYTE   PDEF
3892
3893  2FB1  2E              .ASCII  '.END'
3894  2FB2  45
3895  2FB3  4E
```



```
3896  2FB4  44
3897
3898  2FB5  A6              .BYTE   OPCCL6
3899  2FB6  23              .HBYTE  PEND
3900  2FB7  B5              .BYTE   PEND
3901
3902  2FB8  FF      OPTEND: .BYTE   SYMEOT          ;MARKS END OF OPCODE SYMBOL TABLE
```

```
3903                            .PAGE                   ;'START OF USER-DEFINED SYMBOL TABLE'
3904
3905
3906
3907                            ;START OF USER-DEFINED SYMBOL TABLE
3908
3909  2FB9  2E        SYMTAB: .ASCII  '.'             ;THIS IS THE SYMBOL FOR THE LOCATION COUNTER
3910
3911  2FBA  80                .BYTE   USYM            ;PLAY LIKE IT'S USER-DEFINED
3912  2FBB  00        LOCH:   .BYTE   0               ;HIGH ORDER
3913  2FBC  00        LOCL:   .BYTE   0               ;LOW ORDER
3914  2FBD  FF        SYMCLR: .BYTE   SYMEOT          ;WHEN ZAPPING USER-DEFINED SYMBOLS STORE
3915                                                    ;'SYMEOT' FLAG HERE.
3916
3917
```



```
3918                            .PAGE                   ;'PLACE AT END OF PROGRAM'
3919                    ;
3920                    ;
3921                    ;
3922        0000                .END
```

Ref Lines:

LOCL	533✓	613✓	624✓	649✓	681✓	683✓	699✓	846✓	913✓	922✓	3913D
LOCH	534	615	627	651	684	686	696	848	915	919	3912
SYMCLR	512✓	3914D									
SYMTAB	1198	1200	1461	1463	3909D						
OPTEND	3902D										

SET-UP FOR REF'S THRU ZERO-PAGE

ORIGINAL VALUES

```
252D:   <SYMTAB ← B9
2531:   >SYMTAB ← 2F
26B5:   <SYMTAB ← B9
26B9:   >SYMTAB ← 2F
```

CAN PICK UP ALL ✓'s BY SCANNING
$2000 → $23A7
($23A7: $FF ← $60 TO KILL SCAN)

8 BYTES:

```
        LDX#    7
LOOP    CMPX    SYMTAB
        BEQ
        DEX     QUIT
        BPL     LOOP
QUIT
        RTS
```

11 or 15 bytes

+8

Now: 32

Assembler:

Instrs: $2000 → $28C2

Buffer: $28C3 → $28D0

Instrs: $28D1 → $2A16

DATA: $2A17 → $2C61

Instrs: $2C62 → $2C7D

ACCHOU	2880	C1BYTI	2C7C	DIERR	29DB	GETLC	25F1
ADBUFF	28C3	C1BYTO	2C7C	DMPEND	272D	GETLDV	207F
ADDCNT	007C	C1CLSI	2C7C	DMPRET	269C	GETLIN	25CD
ADDLOC	2206	C1CLSO	2C7C	DMPSC	268B	GETODV	2072
ADDS1	2494	C1OPNI	2C7C	DMPSCL	269D	GETSRC	20B6
ADDSY1	24AC	C1OPNO	2C7C	DMPSYB	2685	GETSRT	2500
ADDSYM	2478	CABS	22F6	DMPSYM	2684	GETSV	244B
AERR	0002	CAERR	2318	DNEXT0	26BF	GETSV1	245B
AIASC	2648	CDSPH	2BE3	DNEXT1	26C1	GETSY1	24EC
AIBYTC	267D	CDSPL	2BEA	DOERR	29DF	GETSYM	24EA
AIERR	2667	CHAR	0032	DOP2	224C	GEXPR	2288
AIHBYT	2675	CHART	2810	DSCNT	26FF	GEXPR1	2291
AIHEX	263C	CHKARG	241C	DSHC	26F1	GGTSYM	24E7
AILBYT	266A	CHKBUF	007E	DSHCRT	26F0	GOTEOF	25BC
AIOCT	2642	CL0	0000	DSLCOP	26CB	GSPNOR	2599
AMERR	2636	CL1	0001	DSMDF	26FD	HAKSPA	2C7E
ASCOUT	28B9	CL2	0002	DSONE	2711	HCTZ	281F
ASKDEC	20AD	CL3	0003	DSLDF	26F9	HEXDIG	2948
ASKPAS	2026	CL4	0004	DUMMY	2C64	HEXRET	2954
ASKSYM	20C8	CL5	0005	EACDL	2206	HEXTAB	2BCB
ASKWID	208C	CL6	0006	EDELFF	3669	HPOS	0044
ASM65	2000	CL7	0007	EDINIT	3928	HSTAR	2826
ASYM	2619	CLASS0	22D2	ENDL	21E9	IADD	290E
ASYMC	2639	CLASS1	22E2	ENDLAB	21A7	IDSPE	2C08
ASYUDF	2654	CLASS2	2307	EOFCHR	001A	IDSPH	2C02
ATOH	25F7	CLASS3	230D	EOFM	2AF9	IDSPL	2C0A
ATOMRT	2684	CLASS4	231B	EOLCHR	000D	IERR	0010
BADEVN	20A8	CLASS5	234B	ERRC	29E1	IMUL10	28EC
BBYTO	27F9	CLASS6	2354	ERRCNT	0041	INCONT	29BA
BCRLF	27DB	CLIAH	2C42	ERRDEV	0028	INDONE	24A0
BFLAG	007B	CLIAL	2C4A	ERRFLG	0040	INIAH	2C22
BHEXO	27D5	CLOAH	2C52	ERRMC	285C	INIAL	2C2A
BINDEV	0022	CLOAL	2C5A	ERRME	286C	INIDEV	29B2
BINNM	2A75	CLODEV	29D1	ERRCRM	2887	INNAME	2496
BINOUT	27E5	CLOSEO	2A0D	ERRTAB	28DB	INOAH	2C32
BIRTS	280F	CONE	2340	EXADD	22CB	INOAL	2C3A
BITEMP	0080	CONIDV	0000	EXFNXT	2297	INODEV	29C7
BMLAB	004C	CONINI	2586	EXPR	2294	INPB	2974
BMMDF	004E	CONODV	0000	EXPRET	22D1	INPD1	28D4
BMOPC	004A	COUNT	2C97	FFCHR	000C	INPDEC	28D1
BMSIZ	004B	CPZERO	22EB	FILOUT	2C6F	INPDRT	2902
BMUDF	004D	CRCHR	000D	FLAGS	0038	INPH1	2934
BPRINT	284F	DATBUF	28C5	FLSYMM	2A26	INPHEX	2931
BYTE1	003D	DATCNT	007D	FNCLOP	2742	INPO1	295A
BYTE2	003E	DBONLY	26E9	FNCTAB	273E	INPOCT	2955
BYTE3	003F	DCHKU	26D3	GDEVNC	229C	INPSW	0024
C0BYTI	2C7C	DCHROK	26C5	GERMES	2833	IOERRM	2A17
C0BYTO	2C7C	DEL1	2587	GETBDV	2272	IONEB	2485
C0CLSI	2C7C	DELCHR	007F	GETC	2562	IORET	2991
C0CLSO	2C7C	DEVI	0026	GETCC	256C	ISUB	2923
C0OPNI	2C7C	DEVMAX	0006	GETCH	1E5A	JPENDL	2171
C0OPNO	2C7C	DEVO	0027	GETL1	25D0	JSRIND	0033

JTEMP	0034	MFLSYM	0073	PACKC	2575	SETCON	2A04
KEYINE	2C79	MIOERR	005B	PACKC1	2577	SETSV	2463
LABEL	2187	MLSTN	0061	PAGEM	2AAE	SETSV1	2470
LABERR	2133	MONADR	002A	PAGMAX	0035	SHERR	226D
LADDR	2224	MOVFL	0071	PASCII	23C1	SIERR	24D2
LADDRC	2234	MPAGE	0063	PASEND	2400	SIXLIN	2CBF
LBLOOP	223D	MPASS	006F	PASERR	2405	SIXLOP	2CC5
LBYTO	274E	MREADY	006B	PASNXT	23E0	SLEFT1	2909
LCOPYL	225D	MSRCN	005D	PASS1	22C0	SLEFT2	2906
LCOPY1	2260	MSYMT	0075	PASS2	20ED	SLEFT4	2903
LCRLF	276D	MTITLE	0065	PASSC	20DC	SLERR	24CE
LENDB	224D	MWIDTH	0067	PASSM	2A42	SMERR	24DA
LERR	0080	NBYTES	003C	PBYTE	2367	SOERR	24D6
LFCHR	000A	NCARY	288A	PBYTEC	236C	SPERR	24CA
LHEXO	275B	NERMES	2857	PCHEK	2864	SPNOR	259C
LIERR	21C4	NEWSRC	210C	PCKDEL	257F	SPNOR1	259E
LINADR	0039	NEXTP	20E0	PCKRET	2584	SRCDEV	0020
LINBUF	0100	NHBUA	285D	PDEF	2408	SRCNM	2A5A
LINEL	007F	NOT09	293D	PDMERR	2444	SSERR	24BE
LISTLN	2218	NOTLF	2CA4	PDNCTF	243F	STACK	0100
LISTRT	2280	NSYNC	24FB	PDSERR	2418	START	2000
LMDEF	21C9	NTEMPH	002E	PEATC	2182	STCDBS	00DF
LNOTF	21D2	NTEMPL	002F	PEND	23B5	STCEOS	00BF
LOCH	2FBB	NULLI	2C62	PENDL	2136	STRADR	0030
LOCL	2FBC	NUMBH	002C	PERR	0038	STSDBS	0020
LOUTB	279D	NUMBL	002D	PGOBEG	0020	STSEOS	0040
LOUTC	2769	OBLOOP	27B9	PGZERO	0081	STSP1	007F
LOUTC1	277E	OCTRET	2973	PHBYTE	235F	STSP2	0080
LOUTC2	278E	ODSPH	2C12	PINIT	20FD	SUERR	24C6
LOUTCP	2799	ODSPL	2C1A	PLCC	2383	SWRT	2894
LOUTST	2730	OERR	0004	PNCSY1	2176	SYCUDF	00F7
LPASS2	21AA	OPCCL0	00E0	PNCSYM	2174	SYMADR	0048
LPERR	21BF	OPCCL1	00A1	PPAGE	23A8	SYMCLR	2FBD
LPLOP	2CAE	OPCCL2	00E2	PRET	23C0	SYMCLS	0007
LPTRES	2CD2	OPCCL3	00E3	PRINT	2831	SYMEOT	00FF
LPTSIM	2C98	OPCCL4	00E4	PRICUT	2CBA	SYMFLG	0080
LSP2	2764	OPCCL5	00E5	PRCCLN	211C	SYMLAB	0010
LSP4	2761	OPCCL6	00A6	PRCPC	2139	SYMMDF	0004
LSPACE	2767	OPCTAB	2CE3	PRCPC1	2150	SYMNMX	0009
LSTDEV	0021	OPCVH	0047	PRCPCE	2145	SYMOPC	0020
LSTNM	2A91	OPCVL	0046	PRCLT	2874	SYMSIZ	0040
LTEMP	0045	OPROK	22A5	PRCLT1	2CB9	SYMTAB	2FB9
LWIDTH	0043	OPTEND	2FB8	PWORD	2373	SYMTM	004F
MARESE	28A4	OPTSRC	2524	READYM	2B6E	SYMTMP	0050
MAXDEV	0025	OUTB	2992	RELBAK	2345	SYMTRM	2505
MAXPAG	0029	OUTBIN	27B3	RESETH	2281	SYMTRT	2523
MBINN	005F	OUTCH	1EA0	RSFLG	288F	SYMUDF	0008
MECF	006D	OUTRET	2A03	RTS1	2586	TABCHR	0009
MERR	0040	OUTRT	27F8	SAERR	24C2	TABS	23F1
MERROR	0069	OUTSL	29F9	SERR	0031	TCNT	002E
MESGER	0077	OUTSTR	29EF	SERRT	24DC	TESTN	25A6
MESNER	0079	OVFLM	2B9B	SERRT1	24E2	TESTN1	25A8

```
TESTNS  25A3
TEXTP   0038
TITLEM  2AAF
TLDEV   0023
TPINI   2594
TRTS    2C7C
TVOUT   2C66
TXTADR  0036
TXTINI  258C
UCHKDN  255D
UCHROK  253A
UENDNM  254B
UERR    0020
UNEXT0  2534
UNEXT1  2536
UNOTF   2561
UPDSYM  25AF
USHORT  2554
USKIPN  2544
USLOOP  253C
USYM    0080
USYSR1  2532
USYSRC  252C
WCRLF   2CD8
WIDTHM  2AD5
WROB    27C8
ZADRES  28B0
ZALL    2877
ZERNUM  291C

ERRORS DETECTED: 0

RUN-TIME:  29 SECONDS

5K CORE USED
```

ACCHOU 1729 1731 1733 1735 1766
ADBUFF 1693 1730 1732 1736 1738 1820 1821 - 1836
ADDCNT 367 1692 1694 1695 1714 1811
ADDLOC 680
ADDS1 1077 1085
ADDSY1 1099 1101
ADDSYM 653 1021 1070 1408
AERR 220 1120
AIASC 1359 1394
AIBYTC 1414 1418
AIERR 1421 1429
AIHBYT 1361 1415
AIHEX 1355 1388
AILBYT 1363 1411
AIOCT 1357 1391
AMERR 1375 1380
ASCOUT 1744 1746 1773 1827
ASKDEC 413 462 472 485
ASKPAS 408 415 420 423 442 1084
ASKSYM 504 510
ASKWID 461 464 466
ASM65 383
ASYM 1365 1367
ASYMC 1373 1377 1379 1386
ASYUDF 1371 1400
ATOM 757 771 1352 1412 1416
ATOMRT 1369 1396 1422
BADEVN 474 477 483
BBYTO 921 924 1637 1689
BCRLF 1666 1747
BFLAG 356 1690 1712 1782
BHEXO 1650 1653
BINDEV 234 390 452 1444 1673 2091
BINNM 339 2198
BINOUT 1655 1667 1669 1670 1798
BIRTS 1697 1699 1724
BITEMP 360 1792 1797 1827 1830
BMLAB 383 620 1013
BMMDF 385 1015 1374 1481
BMOPC 381
BMSIZ 382 1035 1052 1489 1508
BMUDF 384 638 1376 1479
BPRINT 1734 1742
BYTE1 281 706 796 809 816 868 891 900 974 1635
BYTE2 282 798 811 818 902
BYTE3 283 820
C0BYTI 2698 2710 2856
C0BYTO 2719 2728 2858
C3CLSI 2777 2786 2868
C0CLSO 2795 2804 2870
C0OPNI 2738 2747 2864
C0OPNO 2756 2765 2866
C1BYTI 2699 2711 2857

C1BYTO 2723 2729 2859
C1CLSI 2778 2787 2869
C1CLSO 2796 2835 2871
C1OPNI 2739 2748 2865
C1OPNO 2757 2766 2867
CABS 807 815 828
CAERR 839 858 863
CDSPH 583 2596
CDSPL 581 2608
CHAR 265 553 566 591 752 758 951 968 970 994 1004 1241 1246 1254
1247 1295 1934 1969 1974
CHART 1691 1700
CHKARG 1031 1034
CHKBUF 359 1743 1745 1753 1754 1768 1769 1771
CL0 152 200
CL1 158 201
CL2 166 203
CL3 170 204
CL4 175 205
CL5 179 206
CL6 184 207
CL7 191
CLASS0 794 2596 2608
CLASS1 805 2597 2609
CLASS2 827 2598 2610
CLASS3 834 2599 2611
CLASS4 843 2600 2612
CLASS5 867 2601 2613
CLASS6 875 2602 2614
CLIAH 2776
CLIAL 2785
CLOAH 2786 2794
CLOAL 2834 2843
CLODEV 2834 2890 2892
CLOSE0 441 2089
CONE 838 859 862
CONIDV 398 410 1341 2079 2692
CONINI 2755 2764 2872
CONODV 395 408 438 669 1078 1432 1445 1453 2046 2081 2693
COUNT 2884 2889 2895 2898 2907 2909 2916
CPZERO 809 860
CRCHR 42 1585 1666 2096 2109 2113 2143 2292 2294 2296 2319 2368 2396 2400
2434 2438 2459 2483 2507 2511 2537 2559
DATBUF 1721 1734 1837 1838
DATCNT 358 1700 1702 1723 1728 1740 1755
DBONLY 1485 1488
DCHKU 1474 1477
DCHROK 1469 1476
DEL1 1257 1265
DELCHR 46 1256
DEVI 242 399 411 541 1340 1962 2041 2080
DEVMAX 240 2704
DEVO 243 396 409 1079 1321 1328 1532 1535 1538 1615 1618 1624 1671 1674

1679 1986 2043 2047 2082
DIERR 1968 2041
DMPEND 1478 1527
DMPRET 1443 1446 1454
DMPSC 1432 1449
DMPSCL 425 1444
DMPSYB 422 1429
DMPSYM 1436 1461
DNEXT0 1466 1496
DNEXT1 1467 1494
DOERR 1994 2043
DOP2 418 424
DSCNT 1487 1502 1504
DSHC 1493 1526
DSHORT 1490 1492
DSLOOP 1468 1473
DSMDF 1482 1503
DSONE 1529 1513
DSUDF 1490 1501
DUMMY 2721 2722 2730 2731 2815
EADDL 662 668 671 674
EDBUFF 2700 2712 2878
EDINIT 2740 2749 2879
ENDL 559 661 956 979
ENDLAB 618 628 654
EOFCHR 40 1335 2814
EOFM 346 2339
EOLCHR 41 594 723 753 759 952 958 963 971 995 1177 1338 1395 1582
ERRC 2319 2342 2044
ERRCNT 286 428 429 535 536 1133 1135
ERRDEV 245 387 2044
ERRFLG 285 549 667 725 1136 1137
ERRNC 430 432
ERRNE 439 441
ERRORM 344 2541
ERRTAB 732 2581
EXADD 782 785
EXPNXT 758 784 786
EXPR 757
EXPRET 760 764 787
FFCHR 44 2259
FILOUT 2829 2833
FLAGS 276 393 500 503 517 523 526 606 645 661 916 929 938 940
1372 1400 1429 1431 1437 1439 1483
FLSYMM 349 2113
FNDLOP 1545 1547
FNDTAB 1543 1601
GDEVNO 450 456 472 492 539
GERMES 351 2400
GETBDV 449 451
GETC 598 713 721 756 770 966 1145 1154 1238 1286 1331 1388 1391 1394
1398 1411 1415 1852 1927 1944
GETCC 1239 1243

```
GETCH    2822    2841
GETL1    1330    1334    1339
GETLC    1342    1345
GETLDV    455     457
GETLIN    486     506     514    1329
GETODV    448     522
GETSRC    491     493     537
GETSRT   1149    1156
GETSV     573     622    1031    1386
GETSV1   1036    1040
GETSY1   1147    1155
GETSYM    550     561    1146    1367
GEXPR     752     794     805     827     834     843     886     889     898     911    1007
GEXPR1    754     756
GGTSYM    999    1010    1145
GOTEOF   1321    1336
GSPNOR    962    1286    1289    1294    1346
HAKSPA   2850    2851
HCTR      942    1737
HEXDIG   1917    1924
HEXRET   1919    1921    1929
HEXTAB   1567    1654    2563
HPOS      269     743    1543    1590    1594    1603    1608    1622
HSTAR     918    1710
IADD      765    1851    1864    1881
IDSPE    2723    2724
IDSPH    1965    2697    2704
IDSPL    1963    2709
IERR      217    1128
IMUL10   1847    1857
INCONT   2012    2029    2037
INDONE   1097    1092
INIAH    2011    2737
INIAL    2009    2746
INIDEV    420     543    2009
INNAME   1096    1091
INOAH    2028    2755
INOAL    2026    2764
INODEV    397     454     460    2026
INPB     1240    1958
INPD1    1844    1853
INPDEC    437    1366    1843
INPDRT   1845    1867
INPH1    1914    1928
INPHEX   1389    1913
INPO1    1935    1945
INPOCT   1392    1933
INPSW     237     392    1238    1330    1332
IOERRM    337    2396
IONEB    1104    1106
IORET    1975    2018
ISUB      783     850     855    1899
JPENDL    586     630     633     637
```

JSRIND 267 585 1967 1993 2016
JTEMP 263 582 584 876 878 879 1964 1966 1988 1990 2010 2012 2027 2035
KEYINE 2697 2709 2841
LABEL 555 557 604
LABERR 558
LADDR 693 696
LADDRC 695 702
LBLOOP 706 712
LBYTO 698 701 708 1514 1518 1554
LCOPYL 721 724
LCPPY1 715 720 722
LCRLF 1465 1525 1527 1584
LENDB 704 713
LERR 214 1126
LFCHR 43 1333 1343 1587 1668 2097 2110 2114 2144 2293 2295 2297 2320 2369
2397 2431 2435 2439 2460 2484 2508 2512 2538 2560
LHEXO 1559 1566
LIERR 621 632 646
LINADR 274 1276 1278
LINBUF 58 274
LINEL 51 1260
LISTLN 666 673 691
LISTRT 726 738
LMDEF 629 634
LNOTF 625 645
LOCH 534 615 627 651 684 686 696 848 915 919 3912
LOCL 533 613 624 649 681 683 699 846 913 922 3913
LOUTB 1586 1588 1599 1614
LOUTC 718 722 733 1534 1568 1582
LOUTC1 1583 1593
LOUTC2 1602 1605
LOUTCP 1596 1608
LOUTST 442 737 934 1522 1532
LPASS2 607 620
LPERR 625 629
LPLOP 2897 2899
LPTRES 2760 2769 2911 2915
LPTSIM 2723 2732 2887
LSP2 1572 1573
LSP4 694 1572
LSPACE 702 709 1505 1573 1574 1602
LSTDEV 233 391 437 458 664 670 931 1452 2089
LSTNM 343 2228
LTEMP 293 1549 1604
LWIDTH 288 467 1435 1447 1451 1595
MARESE 521 1738 1810
MAXDEV 240 476
MAXPAG 248 1376
MBINN 339 449
MEOF 346 1324
MERR 215 1132
MERROR 344 736
MESGER 351 431

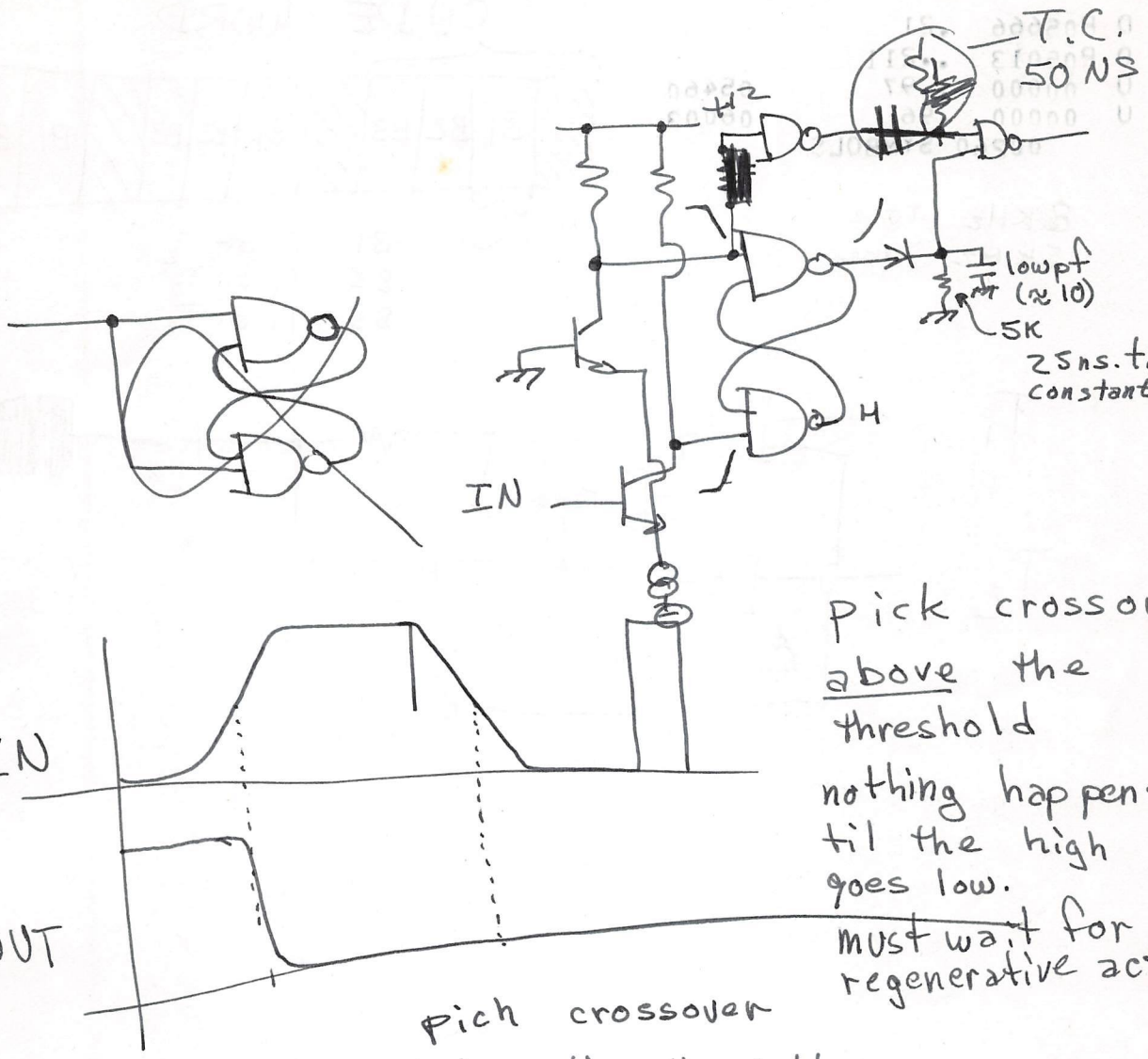
MESNER 362 427
MFLSYM 349 504
MICERR 337 2048
MLSTN 340 455
MONADR 251 2050
MOVFL 348 1080
MPAGE 341 933
MPASS 347 412
MREADY 345 558
MSRCN 338 491
MSYMT 350 1521
MTITLE 342 406
MWIDTH 343 461
NBYTES 280 548 572 682 692 703 711 800 813 822 836 845 870 893
974 976 984 1632 1639
NCARY 1770 1772
NERMES 362 2438
NEWSRC 537 1326
NEXTP 514 518
NHBUA 1737 1739
NOT09 1915 1918
NOTLF 2888 2890 2892
NSYMC 1151 1154
NTEMPH 259 775 849 852 1850 1861 1886 1904
NTEMPL 260 773 847 854 1848 1859 1883 1901
NULLI 2701 2702 2713 2714 2814
NUMBH 255 414 463 473 576 616 626 652 768 774 777 806 819 837
856 877 887 901 914 1039 1055 1097 1403 1417 1860 1876 1885 1887
1894 1903 1905
NUMBL 256 416 465 475 574 614 623 650 697 700 707 766 772 779
797 810 817 875 890 899 912 920 923 1042 1058 1397 1405 1413
1421 1513 1517 1554 1560 1636 1645 1651 1689 1828 1858 1875 1882 1884
1893 1900 1902 1925 1926 1942 1943
OBLOOP 1635 1640
OCTRET 1936 1938 1946
ODSPH 1989 2718
ODSPL 1987 2727
OERR 219 1130
OPCCL0 207 2944 2995 3170 3221 3237 3284 3388 3439 3464 3521 3639
OPCCL1 201 2951 2960 3002 3011 3053 3062 3091 3177 3186 3228 3244 3252 3261
3271 3300 3334 3343 3395 3404 3446 3455 3471 3480 3496 3505 3528 3537
3607 3616 3646 3655 3710 3719 3753 3769
OPCCL2 233 2969 3020 3195 3309 3365 3373 3380 3413 3546 3664 3728
OPCCL3 224 2978 2987 3029 3038 3204 3213 3318 3327 3422 3431 3555 3564 3673
3682 3737 3746 3762 3778
OPCCL4 245 3070 3077 3084 3099 3106 3113 3127 3134
OPCCL5 216 3146 3123 3141 3148 3155 3162 3269 3276 3351 3358 3489 3513 3571
3578 3585 3592 3600 3624 3631 3689 3696 3703 3785 3792 3799 3806 3813
3823
OPCCL6 207 3829 3840 3850 3860 3871 3880 3889 3898
OPCTAB 1194 1196 2936
OPCVH 293 577 808
OPCVL 292 575 795 815 859 867

```
OPROK     762    765
OPTEND   3902
OPTSRC    564   1194
OUTR     1344   1621   1677   1981   2069
OUTBIN    663   1632
OUTCH    2823   2830   2833   2835
OUTRET   2068   2272
OUTRT    1633   1641   1681
OUTSL    2067   2271
OUTSTR    417    434    485    505   1081   1325   1536   2049   2062
OVFLM     348   2511
PACKC     959   1264
PACKC1   1255   1337
PAGEM     341   2259
PAGMAX     67    248
PASCII    951   3872   3873
PASEND    964    969    972    983
PASERR    953    986
PASNXT    966    982
PASS1     421    500
PASS2     424    521
PASSC     508    513    527
PASSM     347   2143
PBYTE     889   3830   3831
PBYTEC    888    891
PCHEK    1741   1743
PCKDEL   1259   1267
PCKRET   1262   1266
PDEF      994   3890   3891
PDMERR   1014   1016   1022
PDNOTF   1012   1020
PDSERR    996   1002   1006
PEATC     598    618
PEND      938   3899   3900
PENDL     569    568    570    586    593    595
PERR      218   1124
PGOBEG     82    227
PGZERO    365
PHBYTE    886   3841   3842
PINIT     513    531
PLOC      911   3881   3882
PNOSY1    502    599
PNOSYM    552    591
PPAGE     929   3861   3862
PRET      917    930    941    943
PRINT    1705   1707   1710   1721
PRIOUT   2893   2896   2902   2920   2922
PROCLN    515    547
PROPC     561    597
PROPC1    565    571
PROPCE    563    566
PROUT    1724   1749
PROUT1   2901
```

PWORD 898 3851 3852
READYM 345 2463
RELBAK 857 861
RESETH 691 742
RSFLG 1698 1781 1810
RTS1 1261 1263 2737 2741 2742 2746 2750 2751 2758 2759 2767 2768 2776 2779
2780 2781 2785 2788 2789 2790 2794 2797 2798 2799 2803 2806 2807 2808
2872
SAERR 839 1120
SERR 221 1118
SERRT 1119 1121 1123 1125 1127 1129 1131 1133
SERRT1 1134 1136
SETCON 531 1323 2079
SETSV 617 1319 1049 1102
SETSV1 1353 1357
SHERR 728 730 735
SIERR 632 1128 1429
SIXLIN 2891 2900 2925
SIXLOP 2908 2910
SLEFT1 1862 1863 1865 1874 1875 1941
SLEFT2 1873 1874 1940
SLEFT4 1873 1924
SLERR 558 1126
SMERR 636 1124 1132 1380
SOERR 569 1130
SPERR 629 1124
SPNOR 1147 1287 1353
SPNOR1 1288
SRCDEV 232 389 494 540
SRCNM 338 2169
SSERR 986 1033 1118
STACK 65 1620 1676 1992
START 36 378
STCOBS 63 1438
STCEOS 60 502 524
STRADR 262 2063 2065 2067
STSOBS 62 1430 1484
STSFOS 59 516 939
STSP1 55 501
STSP2 56 525
SUERR 1122 1378
SWRT 1727 1792
SYCUDF 130 610 1017
SYMADR 295 578 612 635 1018 1023 1034 1038 1041 1051 1056 1059 1073 1074
1088 1101 1109 1195 1199 1201 1209 1211 1225 1306 1307 1308 1310 1462
1464 1473 1488 1507 1511 1516
SYMCLR 512 3914
SYMCLS 137 579
SYMEOT 139 511 1128 1215 1477 3902 3914
SYMFLG 120 200 201 203 204 205 206 207 209
SYMLAB 127 303 611 648
SYMMDF 132 305 634 1022
SYMNMX 324 1150

SYMOPC 125 200 201 203 204 205 206 207 301
SYMSIZ 123 200 203 204 205 206 302 1100 1217
SYMTAB 1198 1230 1461 1463 3909
SYMTM 329 350
SYMTMP 312 324 1086 1152 1157 1207 1472 1475
SYMTRM 1148 1165
SYMTRT 1166 1168 1170 1172 1174 1176 1178 1180
SYMUDF 129 334 1407
TABCHR 45 556 596 717 1169 1598
TABS 1546 1548 2620
TCNT 258
TESTN 1295 1844 1914
TESTN1 1296 1364
TESTNS 1294
TEXTP 272 954 961 977 981 997 1009 1243 1244 1255 1262 1281
TITLEM 342 2260
TLOEV 235 665 672 932 1433 1455 1534 1617
TPINI 1283
TRTS 2842 2856 2857 2858 2859 2864 2865 2866 2867 2868 2869 2870 2871
TVOUT 2718 2727 2828 2902
TXTADR 271 1245 1258 1277 1279
TXTINI 394 744 957 1276 1329 1345
UCHKDN 1218 1225
UCHROK 1215 1210
UENDNM 1212 1215
UERR 216 1122
UNEXT0 1202 1224
UNEXT1 1203 1222
UNOTF 1216 1229
UPDSYM 1223 1304 1495
USHORT 1218 1220
USKIPN 1211 1214 1226
USLOOP 1204 1207
USYM 229 647 1020 1406 3911
USYSR1 1197 1201
USYSRC 624 1211 1198 1370
WCRLF 2897 2908 2919
WIDTHM 343 2300
WROB 1645 1829
ZADRES 1813 1819
ZALL 1748 1752 1812
ZERNUM 755 1031 1352 1419 1843 1892 1913 1933

```
      FUNCTION ZAPGAUSS (Y)
      DIMENSION TN(42)
      DATA (TN=-3,2905,-1,96,-1,6449,-1,4395,-1,2816,-1,1503,-1,0364,
     1-,9346,-,8416,-,7554,-,6745,-,5978,-,5244,-,4538,-,3853,-,3186,
     2-,2533,-,1891,-,1257,-,0627,0,,,0627,,1257,,1891,,2533,,3186,
     3,3853,,4538,,5244,,5978,,6745,,7554,,8416,,9346,1,0364,1,1503,
     41,2816,1.4395,1,6449,1,96,3,2905,3,2905)
      DATA (T=40,),(D=0,5)
      R=RANF(-1)
   10 M=T*R  $  N=M+1
      D=T*R-M
      ZAPGAUSS=TN(N)+D*(TN(N+1)-TN(N))
      RETURN
      END
```

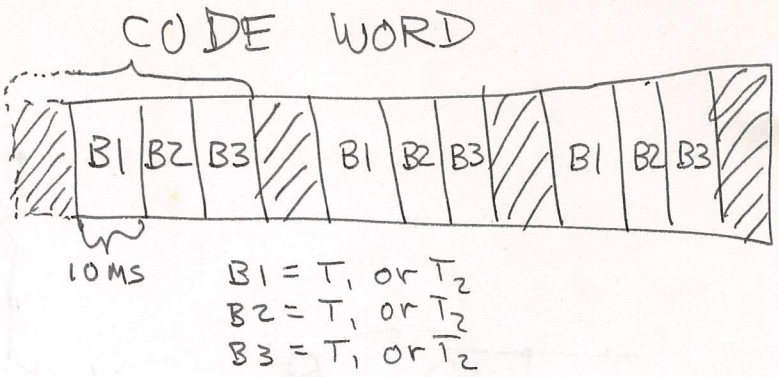


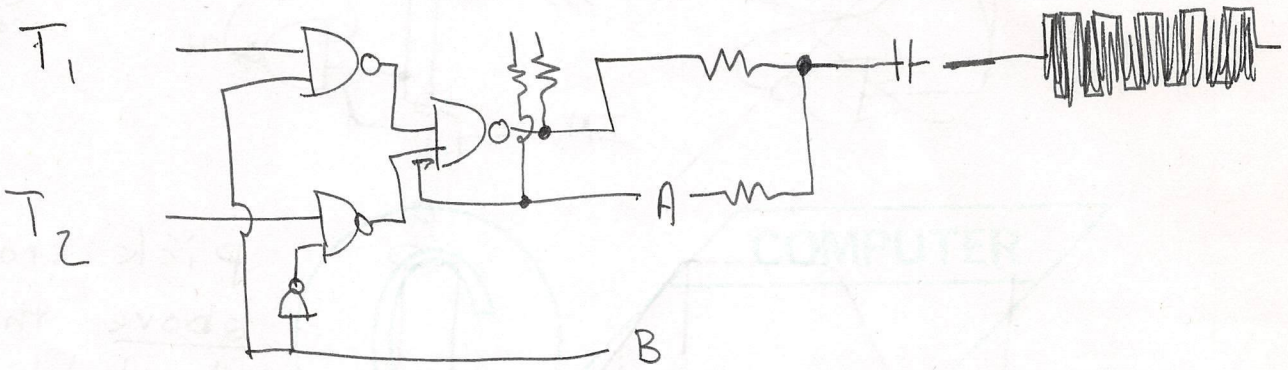
pick crossover above the threshold
nothing happens til the high goes low.
must wait for regenerative action

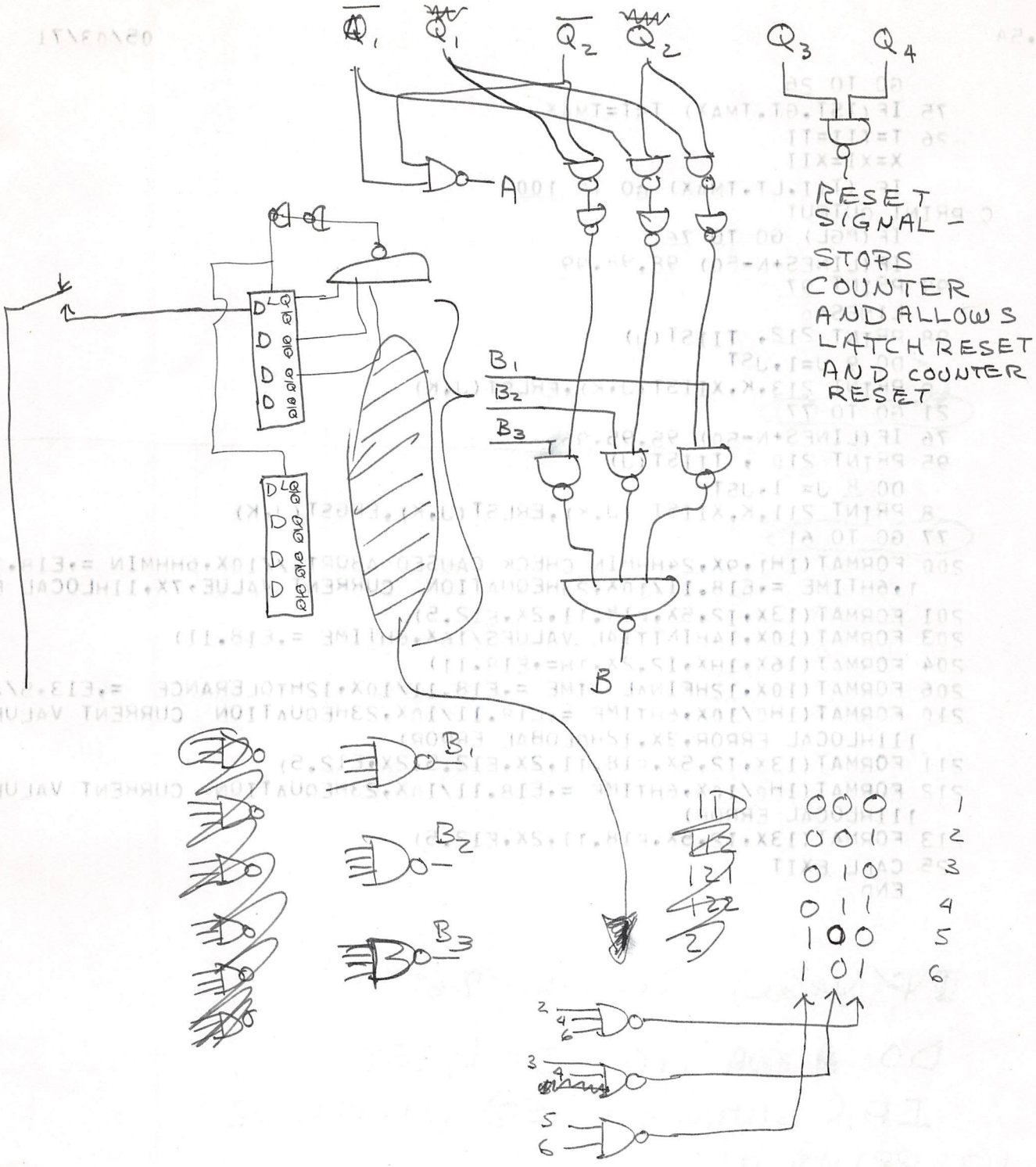
pick crossover below the threshold.
Then slow waveforms can cause troubles.

```
D P05666   .21
D P05013   ..211
U  00000   ..97          05460
U  00000   .96           06003
     00260 SYMBOLS
```

8 KHz Tone
5 KHz Tone






$\overline{Q_1}$
Q_1
$\overline{Q_2}$
Q_2
Q_3
Q_4
A
RESET SIGNAL - STOPS COUNTER AND ALLOWS LATCH RESET AND COUNTER RESET
B_1
B_2
B_3
B
000 1
001 2
010 3
011 4
100 5
101 6

```
      GO TO 26
   75 IF(TST.GT.TMAX) TST=TMAX
   26 T=TII=TI
      X=XI=XII
      IF (TII.LT.TMAX) GO TO 100
C PRINT OUTPUT
      IF(MGL) GO TO 76
      IF(LINES+N=50) 98,98,99
   99 PRINT 97
      LINES=0
   98 PRINT 212, TIIST(J)
      DO 9 J=1,JST
    9 PRINT 213,K,XIIST(J,K),ERLST(J,K)
   21 GO TO 77
   76 IF(LINES+N=50) 95,95,96
   95 PRINT 210 , TIIST(J)
      DO 8 J= 1,JST
    8 PRINT 211,K,XIIST (J,K),ERLST(J,K),ERGST(J,K)
   77 GO TO 61
  200 FORMAT(1H1,9X,24HHHMIN CHECK CAUSED ABORT.//10X,6HHMIN =,E18.11/10X
     1,6HTIME =,E18.11/10X,23HEQUATION  CURRENT VALUE,7X,11HLOCAL ERROR)
  201 FORMAT(13X,I2,5X,E18.11,2X,E12.5)
  203 FORMAT(10X,14HINITIAL VALUES/16X,6HTIME =,E18.11)
  204 FORMAT(16X,1HX,I2,2X,1H=,E18.11)
  206 FORMAT(10X,12HFINAL TIME =,E18.11/10X,12HTOLERANCE  =,E13.5//)
  210 FORMAT(1H0/10X,6HTIME =,E18.11/10X,23HEQUATION  CURRENT VALUE,7X,
     111HLOCAL ERROR,3X,12HGLOBAL ERROR)
  211 FORMAT(13X,I2,5X,E18.11,2X,E12.5,2X,E12.5)
  212 FORMAT(1H0/10X,6HTIME =,E18.11/10X,23HEQUATION  CURRENT VALUE,7X,
     111HLOCAL ERROR)
  213 FORMAT(13X,I2,5X,E18.11,2X,E12.5)
   25 CALL EXIT
      END
```

IF (MGL) GO TO 76

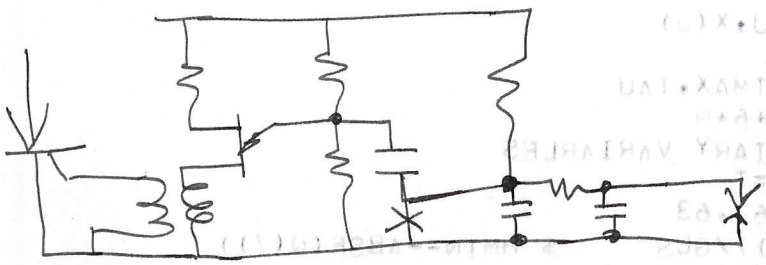
DO 110 I=1,JST

IF (LINES + N -50) 111, 111, 112

112 PRINT 97

LINES =0

DO 113 K=1,N



```
      PROGRAM IVSYST
      DIMENSION Q(20),XIIST(100,8),TIIST(100),ERLST(100,8),ERGST(100,8)
      DIMENSION XI(8),XII(8),XIII(8),XIV(8),X(8),ERL(8),ERG(8)
      EQUIVALENCE (Q(11),X(1))
C     INPUT PROBLEM BOUNDARY VALUES
      CALL INITIAL (X,T,N,H)
C     INPUT INTEGRATION CONTROL PARAMETERS
   61 CALL DATA(Q,IND)
      IF (IND) RETURN
      MGL=Q(6)
      N=Q(9)
      JST=0.
      GUS=1.2
      TAU=Q(8)
      LINES=0
      PRINT 203,T
      DO 205 J=1,N
      PRINT 204, J,X(J)
  205 CONTINUE
      PRINT 206, TMAX,TAU
      LINES=LINES+6+N
C     INITIALIZE AUXILIARY VARIABLES
      TI=TIII=TIV=T
      IF(Q(5))62,63,63
   62 H=-ABSF(Q(4))/GUS      $ HMIN=-ABSF(Q(7))
      IF(Q(1)-Q(2))64,65,66
   63 H=ABSF(Q(4))/GUS       $ HMIN=ABSF(Q(7))
      IF(Q(1)-Q(2))66,65,64
   64 T=Q(2)    $ TMAX=Q(1)     $ GO TO 67
   65 PRINT 214     $ RETURN
  211 FORMAT (42H1INTERRUPT. CHECK INTEGRATION LIMIT INPUTS)
   66 T=Q(1)     $ TMAX=Q(2)
   67 DELT=(TMAX-T)/Q(3)
      IST=0
      TST=T+DELT
      DO 1 K=1,N
    1 XI(K)=XII(K)=XIII(K)=XIV(K)=X(K)
C     VARIABLE STEP PARALLEL INTEGRATION
  100 H=GUS*H
   29 IF(Q(5))68,69,69
   68 IF(HMIN-H)30,70,70
   69 IF(H-HMIN)30,71,71
   70 IF(TST-T-H)10,11,11
   71 IF(T+H-TST)10,11,11
   30 PRINT 200, HMIN,TII
      DO 31 K=1,N
   31 PRINT 201, K,XII(K),ERL(K)
      PRINT 97
      GO TO 25
   10 CALL RKS(XI,TI,N,H)
  101 H2=H/2.
      CALL RKS(XII,TII,N,H2)
      DO 2 K=1,N
    2 XIII(K)=XII(K)
      TIII=TII
      CALL RKS(XII,TII,N,H2)
```

.62

,,

214

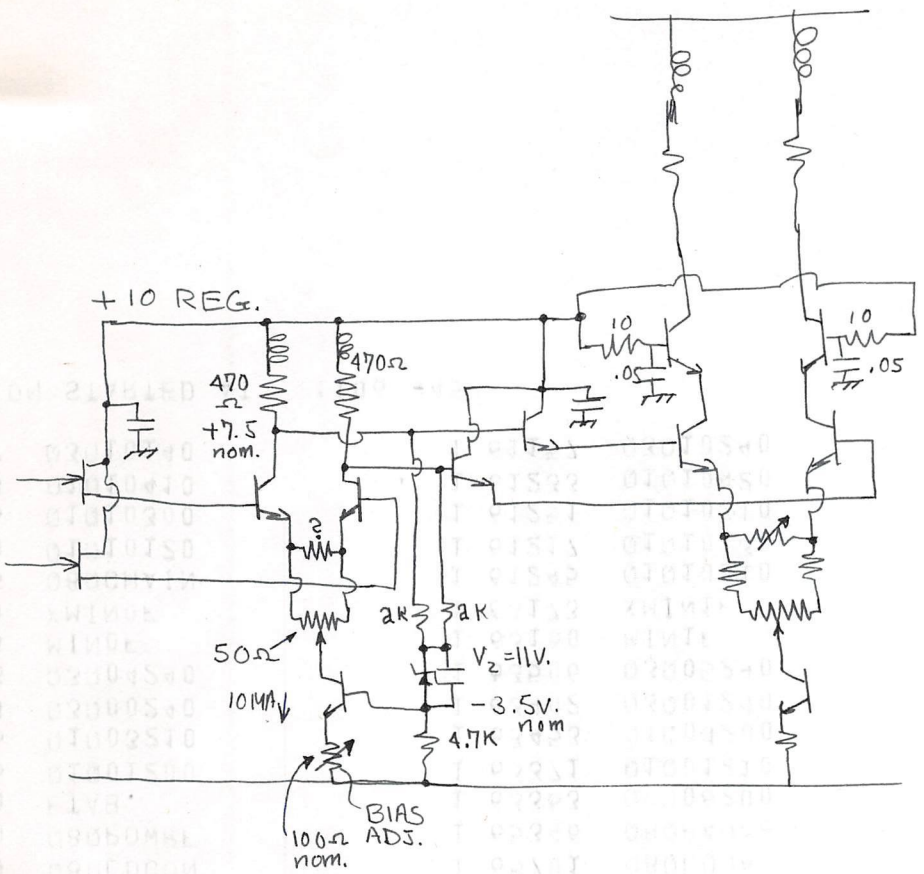
```
OB MESSAGES
                JOB, 153 , 180,BRUHNS
        BK.PU   PUNCHING     149
        BK.PR   FINISHED PRNTING     149
                NOEX
                SKIP    ASSEMERS
EQUENCE NUMBER 4  153 TERMINATED AT    154215 ELAPSED TIME= 00 HRS. 00 MIN. 23 SEC
```

```
      SUBROUTINE DERIV ( XDOT,X,T,NE)
      DIMENSION X(8),XDOT(8)
      XDOT(1)=X(1)*COS(T)
       XDOT (2) = COS(T)
      RETURN
      END
```

COMPUTER CENTER

```
      SUBROUTINE DERIV ( XDOT,X,T,NE)
      DIMENSION X(8),XDOT(8)
      XDOT(1)=1.,*4.,*X(2)
      XDOT(2)=X(1)
      XDOT(3)=1.,*4.,*X(4)
      XDOT(4)=X(3)
      XDOT(5)=1.,-4.,*X(6)
      XDOT(6)=X(5)
      XDOT(7)=2.,-4.,*X(8)
      XDOT(8)=X(7)
      RETURN
      END
```

SUBROUTINE DERIV (XDOT, X, T, NE, NSET)
DIMENSION X(8), XDOT(8)
NE NE
GO TO (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) NSET
1 CONTINUE
EQNS
RETURN
2 CONTINUE
EQNS
RETURN
3 CONTINUE
RETURN
END


+10 REG.
470 Ω +7.5 nom.
470Ω
10
.05
10
.05
2K
2K
V_2 = 11V.
-3.5V. nom
4.7K
50Ω
10MA
100Ω nom.
BIAS ADJ.
- 5 reg.

10 mv/cm

```
PROGRAM NAMES
1 71535  IVSYST     06242      1 70642  RKS         00673      1 70505  DERIV     0013
1 70237  ALLOC,     00234      1 70205  Q7QLODLC    00032      1 65737  IOH,      0224
1 65663  Q8QLOADA   00022      1 65616  TSH,        00045      1 65563  Q8QIFIOC  0003
1 65340  Q8QPAUSE   00040      1 65133  Q8QERROR    00205      1 63553  IOP,      0136
1 63325  STH,       00036      1 63253  EXPF        00052      1 63132  MAX1F     0012
1 61766  DATA       00732      1 61343  IOS,        00423      1 61157  Q1QSTORE  0016

PROGRAM EXTENS,
NONE

LABELED COMMON
NONE

NUMBERED COMMON
NONE

ENTRY POINTS
0 77777  SENTRY                1 76771  IVSYST                1 61105  Q8QENTRY
1 61662  THEND,                1 61101  Q8QDICT,              1 71307  INITIAL
1 71153  RKS                   1 61102  EXIT                  1 62752  SINF
1 63262  EXPF                  1 62727  COSF                  1 63331  STH,
1 63457  Q1Q04210              1 61222  Q1Q10210              1 63416  Q1Q02210
1 63556  IOP,                  1 61416  Q8QHIST,              1 65134  Q8QERROR
1 65403  Q2Q07111              1 65566  Q8QIFIOC              1 65622  TSH,
1 65714  XFIXF                 1 65763  IOH,                  1 61363  IOS,
1 65737  BCDBUF,               1 70205  Q7QLODLC              1 70375  ALLOC,
1 70351  BUSY,                 1 70360  IRETURN,              1 70473  Q2QLOADA
1 70060  ,TSERR,               1 61657  QNDOUBL,              1 70441  ALLOCIN,
1 70037  ,REPCNT,              1 65710  Q8QXFIXF              1 65710  Q8QXINTF
1 65704  Q8QLDCON              1 65701  Q8QLODA               1 65572  Q8QIFEOF
1 65410  Q8QPOWRF              1 65346  Q8QPAUSE              1 65133  Q8NOTRAC
1 64600  ETAB,                 1 63363  Q0Q06200              1 63366  Q1Q00200
1 63366  Q1Q01200              1 63371  Q1Q01210              1 63410  Q1Q02200
1 63436  Q1Q03210              1 63453  Q1Q04200              1 63472  Q1Q05200
1 63374  Q3Q00240              1 63402  Q3Q01240              1 63422  Q3Q02240
1 63463  Q3Q04240              1 63506  Q3Q05240              1 63256  Q8QEXPF
1 63164  MIN0F                 1 63160  MIN1F                 1 63135  XMAX0F
1 63160  XMIN0F                1 63173  XMIN1F                1 62723  Q8QCOSF
1 61535  Q8QCHAIN              1 61245  Q1Q10010              1 61245  Q1Q10020
1 61240  Q1Q10120              1 61217  Q1Q10130              1 61274  Q1Q10200
1 61266  Q1Q10300              1 61251  Q1Q10310              1 61240  Q1Q10320
1 61254  Q1Q10410              1 61233  Q1Q10420              1 61233  Q1Q10430
1 61157  Q3Q10140              1 61157  Q3Q10240              1 61157  Q3Q10340

EXECUTION STARTED AT   1106 =45
```