APPLE UTILITIES

Apple Checker 3.0

by Ken McCandless
2747 Inspiration Drive
Colorado Springs, CO 80917

Due to numerous requests by Nibble readers that the APPLE CHECKER program ignore spaces and hidden control characters in BASIC programs, I am providing this new version. My main goal was to revise the original APPLE CHECKER to provide this capability, but this was quickly cast aside as the full complexity of the problem became clear. To all of you who entered the original program and are facing the entry of this one, I offer my apologies.

This new version of APPLE CHECKER is **easier to use** and provides a corrected CHECKSUM routine which will catch systematic transposition errors; i.e.; the routine is sensitive to the location of each byte. I wish to thank Ken Wetzel of Ridgecrest, CA for his letter bringing to light a problem of data transposition, and his suggested change to correct it. Thanks again, Ken.

HOW TO RUN APPLE CHECKER 3.0

After you have keyed in APPLE CHECKER 3.0, do a **BSAVE APPLE CHECKER 3.0,A$803, L$257**. To execute the program:

1. For BASIC, activate the basic language
2. BRUN CHECK CODE 3.0
3. LOAD (BLOAD) the program to be checked
4. CALL 25 (must immediately follow step #3).
5. After you are finished with the APPLE CHECKER 3.0 program, you should do an FP or INT before continuing on to other things. (This will reset BASIC back to its normal state.)

NOTE: If you are checking a BINARY program, be sure to load it with the address specification option containing a value of at least **$B00**. This will insure that the BINARY program does not load on top of the APPLE CHECKER program and produce disastrous results. As an example: **BLOAD PROGRAM, A$B00**.

The APPLE CHECKER program must be initialized before you load your BASIC program. If you are doing multiple consecutive runs of this program and are switching between APPLESOFT and INTEGER BASIC, you must re-initialize the program following each BASIC language switch. Initialization is accomplished when the program is BRUN but can be repeated, if APPLE CHECKER 3.0 is in memory, by doing a CALL 2051 or for APPLESOFT by typing an "&" followed by a RETURN. If no BASIC language switch is required for your multiple runs, just LOAD the program to be checked and do a CALL 25.

If you think you will never have a need for doing multiple consecutive runs of the APPLE CHECKER program, you can eliminate the requirement for doing step 5 by changing the byte at $A0E to a HEX D3. (BLOAD APPLE CHECKER 3.0, enter the monitor by a CALL -151, type in A0E:D3 followed by a return, and then resave the program.) This change will not alter the performance of the program but merely causes DOS to do a coldstart of the currently active BASIC language, thereby doing the required cleanup.

HOW IT WORKS

When you BRUN the program all the necessary parameters are set so that neither of the BASICs will load the BASIC program on top of APPLE CHECKER. A jump vector is built at HEX 19 (decimal 25) for easy execution of the program and the AMPERSAND (&) vector is set for easy re-initialization from APPLESOFT.

Once you've LOADed (or BLOADed) the program to be checked and issued a CALL 25, the check code development portion of the program begins. The first thing that the program does is to clear out the work areas, build a table for indirection into DOS and then display the program's identifying header. Next, the program grabs the file type and file name from DOS and displays this information if all is correct. The beginning and ending (length for BINARY files) is picked up from DOS next. Now the work really begins.

For BINARY files, the ending address must be calculated so that the program will know when to stop generating checksums. Once this is accomplished, the program continues into the checksum development loop until it is finished.

For APPLESOFT programs, all of the next line address pointers (the first two bytes of every encoded program line) are zeroed. This is done to eliminate problems with ignoring imbedded spaces and control characters. The program continues by calculating the program's length and ending address, and then branching to the checksum development loop.

For INTEGER BASIC programs, all the line lengths (the first byte of every encoded program line) are zeroed. Again, this is to eliminate problems with ignoring spaces and control characters. The program continues by calculating the program's length and ending address, and then goes to the checksum loop.

This loop gets a byte of the program to be checksummed, and then determines whether it is a BINARY or BASIC file. If it is a BINARY file, it checksums the byte and continues to the end of program check. If it is a BASIC file the program must determine whether it is a space or control character, and if it is either, it will be ignored (with the exception of Control-D). All other characters must be checksummed.

If this is not the end of the program to be checksummed, the checksum development loop continues until finished. The program then outputs the length (number of bytes checked, excluding the ignored spaces and control characters) and the checksum. Finally the program cleans up PAGE ZERO and returns to BASIC. Because of all the manipulations to the BASIC program, your loaded program will not be accessible in memory after a run of APPLE CHECKER 3.0. You must reload it to do any checking or to execute it.

If an error is detected during execution of APPLE CHECKER 3.0, the bell will sound and "ERR" will be displayed. To retry you must reload the program to be checked and do a CALL 25.

ERROR CHECKING

Just as in the original version, the error checking is quite simple. The program picks up the file type from the DOS file manager and if the type is a zero you will get an error. Doing a CATALOG will set the file type to a zero. Other DOS functions also set this variable. For this reason you must not execute other DOS functions after the load of the program to be checked before execution of APPLE CHECKER 3.0; disastrous results could take place if you do so.

Another condition that could produce an error occurs when you load a BASIC file, and then switch to the other language before executing APPLE CHECKER 3.0, whether you have initialized APPLE CHECKER or not.

The final condition that could produce an error is to execute APPLE CHECKER 3.0 without loading anything. In this case APPLE CHECKER thinks that itself was the last file loaded and will halt.

MODIFICATIONS FOR A 32K SYSTEM

The program is written for a 48K system with DOS residing in its standard place. If you have a 32K system the following changes must be made to make APPLE CHECKER 3.0 work. Also, if you have DOS loading in some location other than either a standard 48K or a 32K system, you must change the same locations below for your system's values.

1. **BLOAD APPLE CHECKER 3.0**
2. **Enter the monitor (CALL -151)**
3. **855:75**
4. **862:6A**
5. **A47:6A**
6. **A4D:6A**
7. **A53:6A**
8. **A59:6A**
9. **BSAVE APPLE CHECKER 3.0,A$803,L$257**

FINAL NOTE: Remember, REM statements are valid BASIC statements and must be included in your program to obtain the correct checksum. All spaces and control characters, except CONTROL-D, in your BASIC program will be ignored and not included in the checksumming process, so don't worry about the number of spaces in the REMs or any string literals. As an example, the following will produce identical checksums:

```
10 REM INITIALIZATION ROUTINE
20 A="A B C D E"
        AND
10 REMINITIALIZATIONROUTINE
20 A="ABCDE"
```

Since both of the above will produce identical checksums, it should be noted that line 20 in each program is different and will produce different results if used in any comparison, so be alert to spacing in string assignment statements.

Apple Checker 3.0 (Cont.)

```
SOURCE FILE: APPLE CHECKER 3.0
                1 ********************************************************
0000:           2 *                                                      *
0000:           3 *                  A P P L E   C H E C K E R           *
0000:           4 *                       VERSION 3.0                    *
0000:           5 *                                                      *
0000:           6 *  ASSEMBLER:  APPLE 6502 ASSEMBLER/EDITOR             *
0000:           7 *     AUTHOR:  KEN MCCANDLESS                          *
0000:           8 *       DATE:  MAY 18, 1982                            *
0000:           9 *                                                      *
0000:          10 *  THIS PROGRAM GENERATES A CHECKSUM FOR APPLESOFT,     *
0000:          11 *  INTEGER OR BINARY PROGRAMS.  (SPACES AND CONTROL     *
0000:          12 *  CHARACTERS; EXCEPT CONTROL-D, ARE IGNORED.)          *
0000:          13 *                                                      *
0000:          14 *  TO EXECUTE:                                         *
0000:          15 *              1.  BRUN CHECK CODE                     *
0000:          16 *                  (OR BLOAD IT AND THEN CALL 2051)    *
0000:          17 *              2.  (B)LOAD THE PROGRAM FOR CHECKSUMMING *
0000:          18 *              3.  CALL 25   (IF IN THE MONITOR DO  19G) *
00C0:          19 *                                                      *
               20 ********************************************************
0000:          21 *
0000:          22 *                    COPYRIGHT (C) 1982 BY
0000:          23 *                      MICRO-SPARC, INC.
0000:          24 *
----- NEXT OBJECT FILE NAME IS APPLE CHECKER 3.0.OBJO
0803:          25          ORG   $803
0803:          26 *
0000:          27 CKCD     EQU   $00       ;CHECK CODE
0001:          28 BYTE     EQU   $01       ;BEGINNING OF PROGRAM
0001:          29 BEG      EQU   $01       ;  POINTER
0003:          30 END      EQU   $03       ;END OF PROGRAM POINTER
0005:          31 LEN      EQU   $05       ;LENGTH OF PROGRAM
0007:          32 FTYPE    EQU   $07       ;FILE TYPE (0=I,2=A,4=B)
0008:          33 DECVAL   EQU   $08       ;WORK BYTE FOR ASOFT
000A:          34 HLD      EQU   $0A       ;WORK ADDR
0019:          35 ENTRY    EQU   $19       ;ROUTINE ENTRY VECTOR
0024:          36 CHORIZ   EQU   $24
004A:          37 INTLOW   EQU   $4A       ;INTEGER BASIC LOMEM
004C:          38 INTH     EQU   $4C       ;INT PGM END ADDR
0067:          39 ASFTL    EQU   $67       ;A-SOFT PGM START ADDR
00AF:          40 ASFTH    EQU   $AF       ;A-SOFT PGM END ADDR
00CA:          41 INTL     EQU   $CA       ;INT PGM START ADDR
00E0:          42 PBEGL    EQU   $E0
00E6:          43 PBEGH    EQU   $E6
00EC:          44 PENDL    EQU   $EC
00F1:          45 SPEED    EQU   $F1       ;APPLESOFT SPEED
00F2:          46 PENDH    EQU   $F2
03D0:          47 RTNBAS   EQU   $3D0      ;DOS WARMSTART VECTOR
03F6:          48 AMPVEC   EQU   $3F6      ;AMPERSAND VECTOR ADDRESS
0400:          49 LINE0    EQU   $400      ;FIRST BYTE OF TEXT SCREEN AREA
0801:          50 BGNASF   EQU   $801      ;NEW LOAD POINT FOR ASOFT
AA60:          51 BLEN     EQU   $AA60     ;FOR 48K
AA72:          52 BINL     EQU   $AA72     ;FOR 48K
AA75:          53 DOSFN    EQU   $AA75     ;FILE NAME-48K
B5F6:          54 DOSFT    EQU   $B5F6     ;FILE TYPE - 48K
E000:          55 BASICHK  EQU   $E000
0803:          56 *
0803:          57 * MONITOR ROUTINES
0803:          58 *
F941:          59 PRNTAX   EQU   $F941
FC58:          60 HOME     EQU   $FC58
FD8E:          61 CROUT    EQU   $FD8E
FDED:          62 COUT     EQU   $FDED
FDDA:          63 PRBYTE   EQU   $FDDA
FF2D:          64 PRERR    EQU   $FF2D     ;PRINT "ERR"
0803:          65 *
0803:          66 * SETUP FOR ROUTINE
0803:          67 *
0803:D8        68 INIT     CLD
0804:A9 01     69          LDA   #1
0806:85 4A     70          STA   INTLOW    ;SET INTEGER BASIC LOMEM
0808:A9 0B     71          LDA   #<BGNASF
080A:85 68     72          STA   ASFTL+1   ;SET ASFT BEG ADDR TO $801
080C:85 4B     73          STA   INTLOW+1  ;SET INT LOMEM
080E:A9 00     74          LDA   #0
0810:8D 00 0B  75          STA   BGNASF-1  ;MAKE ASFT HAPPY
0813:A9 4C     76          LDA   #$4C      ;SET UP JUMP ADDR FOR ROUTINE
0815:85 19     77          STA   ENTRY
0817:A9 2A     78          LDA   #>START
0819:85 1A     79          STA   ENTRY+1
081B:A9 08     80          LDA   #<START
081D:85 1B     81          STA   ENTRY+2
081F:A9 03     82          LDA   #>INIT    ;SET UP & FOR RE-INIT
0821:8D F6 03  83          STA   AMPVEC
0824:A9 08     84          LDA   #<INIT
0826:8D F7 03  85          STA   AMPVEC+1
0829:60        86          RTS
082A:          87 *
082A:          88 * ROUTINE STARTING POINT
082A:          89 *
082A:D8        90 START    CLD
082B:A2 00     91          LDX   #0
082D:A0 08     92          LDY   #$08      ;CLEAR WORK AREAS
082F:96 00     93 LPCLR    STX   CKCD,Y
0831:88        94          DEY
0832:10 FB     95          BPL   LPCLR
0834:BD 42 0A  96 LPP      LDA   PDATA,X   ;BUILD INDIRECT TABLE
0837:95 E0     97          STA   PBEGL,X
0839:E8        98          INX
083A:E0 1A     99          CPX   #26
083C:D0 F6    100          BNE   LPP
083E:         101 *
083E:         102 * OUTPUT HEADER
083E:         103 *
083E:20 58 FC 104          JSR   HOME
0841:20 8E FD 105          JSR   CROUT
0844:A9 0D    106          LDA   #13
0846:85 24    107          STA   CHORIZ
0848:A2 15    108          LDX   #21
084A:BD 2C 0A 109 LP1      LDA   INP,X
084D:20 ED FD 110          JSR   COUT
0850:CA       111          DEX
0851:10 F7    112          BPL   LP1
0853:         113 *
0853:         114 * DO FILE TYPE AND SET UP PARAMETERS
0853:         115 *
0853:AD F6 B5 116          LDA   DOSFT     ;GET FILE TYPE
0856:F0 65    117          BEQ   GETAGN    ;NOT A VALID TYPE
0858:0A       118          ASL   A         ;DISPOSE OF LOCK BIT
0859:4A       119          LSR   A
085A:4A       120          LSR   A         ;CHANGE INTEGER FROM 1 TO 0
085B:0A       121          ASL   A
085C:85 07    122          STA   FTYPE     ;SAVE IT FOR LATER
085E:         123 * OUTPUT FILE NAME
085E:A2 00    124          LDX   #0        ;OUTPUT FILE NAME
0860:BD 75 AA 125 FNOUT    LDA   DOSFN,X
0863:20 ED FD 126          JSR   COUT
0866:E8       127          INX
0867:E0 1E    128          CPX   #30
0869:D0 F5    129          BNE   FNOUT
086B:         130 * OUTPUT FILE TYPE
086B:A2 06    131          LDX   #6
086D:BD 25 0A 132 OFTYP    LDA   CTYP.X
0870:20 ED FD 133          JSR   COUT
0873:CA       134          DEX
0874:10 F7    135          BPL   OFTYP
0876:A5 07    136          LDA   FTYPE     ;BUILD AND OUTPUT FILE TYPE
0878:4A       137          LSR   A
0879:18       138          CLC
087A:69 C0    139          ADC   #$C0
087C:C9 C0    140          CMP   #$C0
087E:D0 02    141          BNE   OTTP
0880:A9 C9    142          LDA   #$C9
0882:20 ED FD 143 OTTP     JSR   COUT
0885:         144 * BUILD INDIRECT ADDRESSES
0885:A6 07    145          LDX   FTYPE
0887:A1 E0    146          LDA   (PBEGL,X) ;GET APPROPRIATE BEG LO ADDR
0889:85 01    147          STA   BEG
088B:A1 E6    148          LDA   (PBEGH,X) ;GET BEG HI ADDR
088D:85 02    149          STA   BEG+1
088F:A1 EC    150          LDA   (PENDL,X) ;GET APPROPRIATE END LO ADDR
0891:85 03    151          STA   END
0893:A1 F2    152          LDA   (PENDH,X) ;GET END HI ADDR
0895:85 04    153          STA   END+1
0897:E6 08    154          INC   DECVAL    ;TO REDUCE END ADDR BY 1
0899:A5 07    155          LDA   FTYPE
089B:C9 04    156          CMP   #4        ;BIN FILE?
089D:D0 27    157          BNE   BASIC     ;NO
089F:         158 *
089F:         159 * SET UP FOR BINARY FILE
089F:         160 *
089F:A5 02    161          LDA   BEG+1     ;IF LOAD ADDR HI IS SAME
08A1:C9 08    162          CMP   #<INIT    ;THEN NOT A BINARY FILE!
08A3:F0 18    163          BEQ   GETAGN
08A5:A5 03    164          LDA   END
08A7:85 05    165          STA   LEN
08A9:A5 04    166          LDA   END+1
08AB:85 06    167          STA   LEN+1
08AD:18       168          CLC             ;COMPUTE END ADDR
08AE:A5 01    169          LDA   BEG       ;END=BEG+LEN-1
08B0:65 05    170          ADC   LEN
08B2:85 03    171          STA   END
08B4:A5 02    172          LDA   BEG+1
08B6:65 06    173          ADC   LEN+1
08B8:85 04    174          STA   END+1
08BA:4C 3A 09 175          JMP   CONT
08BD:         176 *
08BD:         177 * RESTART
08BD:         178 *
08BD:20 8E FD 179 GETAGN   JSR   CROUT
08C0:20 2D FF 180          JSR   PRERR     ;OUTPUT "ERR"
08C3:4C DC 09 181          JMP   CLNUP     ;CLEAN UP THE ZERO PAGE MESS & QUIT
08C6:         182 *
08C6:         183 * SET UP FOR BASIC FILE
08C6:         184 *
08C6:A5 07    185 BASIC    LDA   FTYPE
08C8:F0 32    186          BEQ   IBASIC    ;0=INTEGER  2=ASOFT
08CA:AD 00 E0 187          LDA   BASICHK   ;IS APPLESOFT ACTIVE?
08CD:C9 4C    188          CMP   #$4C
08CF:D0 EC    189          BNE   GETAGN
08D1:A0 00    190 CLRADD   LDY   #0
08D3:B1 01    191          LDA   (BEG),Y   ;GET NEXT LINE ADDR LO
08D5:85 0A    192          STA   HLD       ;SAVE IT
08D7:C8       193          INY
08D8:B1 01    194          LDA   (BEG),Y   ;GET NEXT LINE ADDR HI
08DA:85 0B    195          STA   HLD+1     ;SAVE IT ALSO
08DC:A9 00    196          LDA   #0        ;TO CLEAR ADDR
08DE:A8       197          TAY             ;RESET Y
08DF:91 01    198          STA   (BEG),Y   ;0 NEXT ADDR PTR LO
08E1:C8       199          INY
08E2:91 01    200          STA   (BEG),Y   ;0 NEXT ADDR PTR HI
08E4:A5 0A    201          LDA   HLD       ;SET UP FOR NEXT ONE
08E6:85 01    202          STA   BEG
08E8:A5 0B    203          LDA   HLD+1
08EA:85 02    204          STA   BEG+1
08EC:D0 E3    205          BNE   CLRADD
08EE:A6 07    206          LDX   FTYPE
08F0:A1 E0    207          LDA   (PBEGL,X) ;REBUILD BEG ADDR
08F2:85 01    208          STA   BEG
08F4:A1 E6    209          LDA   (PBEGH,X)
08F6:85 02    210          STA   BEG+1
08F8:E6 08    211          INC   DECVAL    ;REDUCE END ADDR BY 2
08FA:D0 3E    212          BNE   CONT
08FC:         213 *
08FC:AD 00 E0 214 IBASIC   LDA   BASICHK   ;IS INTEGER ACTIVE?
08FF:C9 20    215          CMP   #$20
0901:D0 BA    216          BNE   GETAGN
0903:         217 * ZERO OUT INTEGER LINE LENGTHS
0903:A0 00    218          LDY   #0
0905:A5 01    219 INTLP    LDA   BEG
0907:85 0A    220          STA   HLD
0909:A5 02    221          LDA   BEG+1
090B:85 0B    222          STA   HLD+1
090D:C5 04    223          CMP   END+1     ;CHECK IF FINISHED
090F:90 17    224          BLT   INTCLR    ;NOT DONE-CLEAR IT
0911:F0 02    225          BEQ   INTLW     ;HI ADDRS THE SAME-CHECK LO ADDRS
0913:B0 06    226          BGE   INTDN     ;(JUST FOR MURPHEY)
0915:A5 0A    227 INTLW    LDA   HLD       ;CHECK LO ADDR
0917:C5 03    228          CMP   END
0919:90 0D    229          BLT   INTCLR    ;NOT DONE - CLEAR IT
091B:A6 07    230 INTDN    LDX   FTYPE     ;RESET BEG POINTER
091D:A1 E0    231          LDA   (PBEGL,X)
091F:85 01    232          STA   BEG
0921:A1 E6    233          LDA   (PBEGH,X)
```

continued on page 188

continued from page 45

APPLE CAL
The Time Machine (Cont.)

```
20100  FOR I = 1 TO 2 STEP 0
20110  VTAB 8: FOR J = 9 TO 28
20120  HTAB J: PRINT " ";
20130  FOR D = 1 TO L: NEXT D
20140  HTAB J: PRINT "*";
20150  NEXT J
20160  IF  PEEK ( - 16384) > 127 THEN 20350
20170  FOR J = 8 TO 11
20180  VTAB J: HTAB 29: PRINT " ";
20190  FOR D = 1 TO L: NEXT D
20195  HTAB 29: PRINT "*";
20200  NEXT J
20210  IF  PEEK ( - 16384) > 127 THEN 20350
20220  VTAB 12: FOR J = 29 TO 10 STEP  - 1
20230  HTAB J: PRINT " ";
20240  FOR D = 1 TO L: NEXT D
20250  HTAB J: PRINT "*";
20260  NEXT J
20270  IF  PEEK ( - 16384) > 127 THEN 20350
20280  FOR J = 12 TO 9 STEP  - 1
20290  HTAB 9: VTAB J: PRINT " ";
20300  FOR D = 1 TO L: NEXT D
20310  HTAB 9: PRINT "*";
20320  NEXT J
20330  IF  PEEK ( - 16384) > 127 THEN 20350
20340  NEXT I
20350  POKE  - 16368,0
20360  PRINT  CHR$ (7)
20370  RETURN
20400  REM  ** FIX STACK ONERR **
20405  REM  FROM PG 136, APPLE REF MANUAL
20410 A$ = "104168104166223154072152072096"
20420  FOR I = 1 TO 10
20430  POKE 767 + I, VAL ( MID$ (A$,I * 3 - 2,3))
20440  NEXT
20450  RETURN
```

continued from page 87

Apple Checker 3.0 (Cont.)

```
0923:85 02      234          STA  BEG+1
0925:4C 3A 09   235          JMP  CONT
0928:B1 0A      236 INTCLR   LDA  (HLD),Y   ;GET LENGTH
092A:18         237          CLC
092B:65 01      238          ADC  BEG       ;CALCULATE NEXT LENGHT ADDR
092D:85 01      239          STA  BEG
092F:A5 02      240          LDA  BEG+1
0931:69 00      241          ADC  #0
0933:85 02      242          STA  BEG+1
0935:98         243          TYA             ;ZERO OUT LENGHT BYTE
0936:91 0A      244          STA  (HLD),Y
0938:F0 CB      245          BEQ  INTLP     ;DO IT ALL AGAIN
093A:           246 *
093A:38         247 CONT     SEC             ;COMPUTE END=END-DECVAL
093B:A5 03      248          LDA  END
093D:E5 08      249          SBC  DECVAL
093F:85 03      250          STA  END
0941:A5 04      251          LDA  END+1
0943:E9 00      252          SBC  #0
0945:85 04      253          STA  END+1
0947:A5 07      254          LDA  FTYPE     ;IF BINARY-GO DO IT
0949:C9 04      255          CMP  #4
094B:F0 0D      256          BEQ  DOCKCD
094D:38         257          SEC            ;COMPUTE LEN FOR BASIC FILES
094E:A5 03      258          LDA  END       ;LEN=END-BEG
0950:E5 01      259          SBC  BEG
0952:85 05      260          STA  LEN
0954:A5 04      261          LDA  END+1
0956:E5 02      262          SBC  BEG+1
0958:85 06      263          STA  LEN+1
095A:           264 *
095A:           265 * DO CKCD ROUTINE
095A:           266 *
095A:A0 00      267 DOCKCD   LDY  #0        ;0 TO Y REG
095C:B1 01      268          LDA  (BYTE),Y  ;GET BYTE OF PROGRAM
095E:8D 00 04   269          STA  LINE0     ;SHOW ACTIVITY
0961:A6 07      270          LDX  FTYPE     ;DETERMINE IF MUST IGNORE
0963:F0 0E      271          BEQ  CHKINT    ;IT'S INTEGER-IGNORE IF NECESSARY
0965:E0 04      272          CPX  #4        ;IS IT BINARY?
0967:F0 28      273          BEQ  SUMIT     ;YES-SO GO CODE IT
0969:           274 * IT'S APPLESOFT
0969:C9 21      275          CMP  #$21      ;CHAR>SPACE?
096B:B0 24      276          BGE  SUMIT     ;YES-SUMIT
096D:C9 04      277          CMP  #4        ;CTL-D?
096F:F0 20      278          BEQ  SUMIT     ;YES-SUMIT
0971:D0 0C      279          BNE  IGNCHR    ;NO-MUST BE CTL-CHAR OR SPACE-IGNORE I
T
0973:C9 80      280 CHKINT   CMP  #$80      ;CHAR<CTL-@?
0975:90 1A      281          BLT  SUMIT     ;YES-SUMIT
0977:C9 A1      282          CMP  #$A1      ;CHAR>SPACE?
0979:B0 16      283          BGE  SUMIT     ;YES-SUMIT
097B:C9 84      284          CMP  #$84      ;CTL-D?
097D:F0 12      285          BEQ  SUMIT     ;YES-SUMIT
097F:           286 * IGNORE CHARACTER ROUTINE
097F:38         287 IGNCHR   SEC             ;DEC LEN FOR EACH CHAR
                                                IGNORED
0980:A5 05      288          LDA  LEN
0982:E9 01      289          SBC  #1
0984:85 05      290          STA  LEN
0986:A5 06      291          LDA  LEN+1
0988:F0 04      292          BEQ  SPOUT
098A:E9 00      293          SBC  #0
098C:85 06      294          STA  LEN+1
098E:4C 9B 09   295 SPOUT    JMP  NXT       ;CONTINUE
0991:           296 * BUILD CODE
0991:18         297 SUMIT    CLC
0992:45 00      298          EOR  CKCD      ;DEVELOP CHECK CODE
0994:2A         299          ROL  A
0995:65 00      300          ADC  CKCD
0997:69 00      301          ADC  #0
0999:85 00      302          STA  CKCD
099B:           303 *
099B:           304 * ADJUST POINTER FOR NEXT PROGRAM BYTE
099B:           305 *
099B:18         306 NXT      CLC             ;CLEAR CARRY IND
099C:A5 01      307          LDA  BYTE      ;GET LO ADDR
099E:69 01      308          ADC  #1        ;INCREMENT IT BY 1
09A0:85 01      309          STA  BYTE      ;STORE IT BACK
09A2:98         310          TYA             ;0 TO A REG
09A3:65 02      311          ADC  BYTE+1    ;INCREMENT HI ADDR WITH
                                                CARRY (IF ANY)
09A5:85 02      312          STA  BYTE+1    ;STORE IT IN HI ADDR
09A7:           313 *
09A7:           314 * CHECK FOR END OF PROGRAM
09A7:           315 *
09A7:18         316          CLC             ;CLEAR CARRY IND
09A8:A5 02      317          LDA  BYTE+1    ;COMPARE HI ADDR
09AA:C5 04      318          CMP  END+1
09AC:90 AC      319          BCC  DOCKCD    ;BYTE<END
09AE:D0 07      320          BNE  ALLDONE   ;BYTE>END
09B0:18         321          CLC
09B1:A5 03      322          LDA  END       ;COMPARE LO ADDR
09B3:C5 01      323          CMP  BYTE
09B5:B0 A3      324          BCS  DOCKCD    ;END<BYTE
09B7:           325 *
09B7:           326 * EXECUTION IS DONE
09B7:           327 *
09B7:A2 09      328 ALLDONE  LDX  #9        ;OUTPUT CONSTANT "LENGTH: "
09B9:BD 10 0A   329 LP2      LDA  LENGT,X
09BC:20 ED FD   330          JSR  COUT
09BF:CA         331          DEX
09C0:10 F7      332          BPL  LP2
09C2:A5 06      333          LDA  LEN+1     ;GET HI BYTE OF LEN
09C4:A6 05      334          LDX  LEN       ;GET LO BYTE OF LEN
09C6:20 41 F9   335          JSR  PRNTAX    ;OUTPUT LENGTH
09C9:A2 0A      336          LDX  #$0A      ;OUTPUT "CHECKSUM: "
09CB:BD 1A 0A   337 LP3      LDA  TOT,X
09CE:20 ED FD   338          JSR  COUT
09D1:CA         339          DEX
09D2:10 F7      340          BPL  LP3
09D4:A5 00      341          LDA  CKCD      ;GET CKCD TOTAL
09D6:20 DA FD   342          JSR  PRBYTE    ;OUTPUT CHECKSUM
09D9:20 8E FD   343          JSR  CROUT
09DC:A9 A0      344 CLNUP    LDA  #$A0      ;BLANK OUT ACTIVITY IND
09DE:8D 00 04   345          STA  LINE0
09E1:A2 04      346          LDX  #4
09E3:A9 08      347          LDA  #8
09E5:81 E6      348          STA  (PBEGH,X) ;RESET BIN LOAD ADDR
09E7:A0 19      349 OUT      LDY  #25       ;CLEAN UP PAGE ZERO FOR BASIC
09E9:A9 00      350          LDA  #0
09EB:99 E0 00   351 ENDLP    STA  PBEGL,Y
09EE:88         352          DEY
09EF:D0 FA      353          BNE  ENDLP
09F1:8D 01 0B   354          STA  BGNASF
09F4:8D 02 0B   355          STA  BGNASF+1
09F7:8D 03 0B   356          STA  BGNASF+2  ;ZERO ASOFT NXT LINE ADDR
09FA:8D F6 B5   357          STA  DOSFT     ;0 TO FILE TYPE IN DOS
09FD:A9 01      358          LDA  #1
09FF:85 F1      359          STA  SPEED     ;SET APPLESOFT SPEED TO MAX
0A01:A9 04      360          LDA  #4        ;TERMINATE ASOFT PGM
0A03:85 AF      361          STA  ASFTH
0A05:A5 4C      362          LDA  INTH      ;TERMINAT INTEGER PGM
0A07:85 CA      363          STA  INTL
0A09:A5 4D      364          LDA  INTH+1
0A0B:85 CB      365          STA  INTL+1
0A0D:4C D0 03   366          JMP  RTNBAS    ;RETURN TO BASIC THROUGH DOS
0A10:           367 *
0A10:           368 *
0A10:           369 *
0A10:A0 BA CB   370 LENGT    ASC  "         :HTGNEL"
0A13:D4 C7 CE
0A16:C5 CC
0A18:8D 8D      371          DFB  $8D,$8D
0A1A:A0 BA CD   372 TOT      ASC  "         :MUSKCEHC"
0A1D:D5 D3 CB
0A20:C3 C5 CB
0A23:C3
0A24:8D         373          DFB  $8D
0A25:A0 BA C5   374 CTYP     ASC  "         :EPYT'
0A28:D0 D9 D4
0A2B:8D         375          DFB  $8D
0A2C:A0 BA CE   376 INP      ASC  "         :NO"
0A2F:CF
0A30:8D 8D 8D   377          DFB  $8D,$8D,$8D,$8D
0A33:8D
0A34:B0 AE B3   378          ASC  "0.3      EDOC KCEHC"
0A37:A0 C5 C4
0A3A:CF C3 A0
0A3D:CB C3 C5
0A40:C8 C3
0A42:           379 *
0A42:           380 * INDIRECT ADDRESSES
0A42:           381 *
0A42:CA 00      382 PDATA    DW   INTL
0A44:67 00      383          DW   ASFTL
0A46:72 AA      384          DW   BINL
0A48:CB 00      385          DW   INTL+1
0A4A:68 00      386          DW   ASFTL+1
0A4C:73 AA      387          DW   BINL+1
0A4E:4C 00      388          DW   INTH
0A50:AF 00      389          DW   ASFTH
0A52:60 AA      390          DW   BLEN
0A54:4D 00      391          DW   INTH+1
0A56:B0 00      392          DW   ASFTH+1
0A58:61 AA      393          DW   BLEN+1
0A5A:           394 *

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```