Apple /// Computer Technical Information

Apple /// Business BASIC 1.3 Source Code Listing

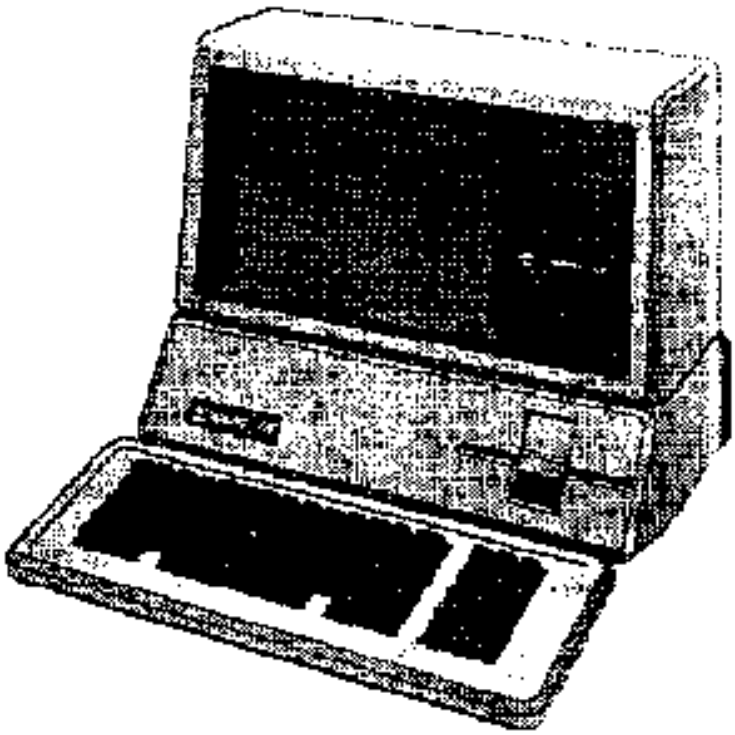

Table of Contents

GENERAL INFORMATION

Apple /// Business BASIC was the BASIC interpreter that Apple computer created for its Apple /// Computer which was released in 1980.

This BASIC appears to be based heavily upon Apple's AppleSoft BASIC which was licenced from Microsoft around 1976. Apple appears to have taken the AppleSoft BASIC source and modified it to support new features for Business BASIC. From internal Apple documents relating to the Apple ///'s development, it appears that Apple created in the late 1970's a BASIC called BASIC III which pre-dated the Apple ///'s development. Apple appears to have converted this BASIC III to become Apple /// Business BASIC.

This BASIC is written in 6502 assembly language. It was assembled on the Apple][(or //e) computers using Apple's EDASM 6502 editor and assembler program. It is interesting (to me at least) that the source code for SOS (the Apple ///'s operating system) was also not ported to the Apple ///. It seems that Apple had no need to put the sources for either Business BASIC or SOS on the /// since the][host worked and changing the source from EDASM format to Pascal /// Assembler format would have taken some time.

This BASIC's chief programmer was Donn Denman who also later wrote Apple's Macintosh BASIC (which was never released).

For information about Business BASIC's features see Apple's "Business BASIC Reference Manual" (two volumes). Also available from Apple is a document describing Business BASIC's variable storage and how to write Business BASIC's invokable modules. I think it would be interesting to obtain the Business BASIC ERS (External Reference Specification) which Apple prepared before each hardware and software project and which served as the blueprint for a project.

From a programming perspective Business BASIC's source code is not well commented. Very few routines have a header comment describing what the routine does and what parameters it uses. The source also does not have a modification history unlike other Apple /// software which Apple produced. Also lacking in the source is a general discussion of how Business BASIC works. This discussion would include the details about the tokenization of keywords how this BASIC generally worked (maybe the BB ERS has this information). This source appears in general to have the same minimal commenting style as AppleSoft BASIC.

Business BASIC's source is not without humor. File CATALOG.TEXT contains the following line:

```
      CMP #$0F      ;ROOT DIRECTORY? (WHAT DOES TOM HAVE TO DO WITH IT?)
```

This line refers to Tom Root who was one of the designers behind the Apple ///'s operating system and its file system. FWIW, I searched in the file for other people's names who worked on the /// software but none were found.

File INVOKE1.TEXT also contains a bit of humor:

```
      ;I WISH I WERE A MOTOROLA 68000,
      ;YES THAT IS WHAT I'D TRUELY LIKE TO BE,
      ;CUZ IF I WERE A M. 68000.
      ;EVERYONE WOULD LOVE TO PROGRAM ME!
```

The author of this document (David Craig) has modified the Business BASIC source files to make them more readable. The original files had no line formatting and were very difficult to read. I have made the following changes:

o Renamed all source files to end with a ".TEXT" suffix.

o Reformatted all the files so they look much more readable. I used a Macintosh program that I created (DTCAsmReFormat) for this purpose which lined up all the assembly language elements in a nice fashion.

o Added a header and a footer to each file listing the file's name.

o Merged all the source files into a single file in the order that the files are assembled. This was done so that I would have just a single file instead of the many original files.

I also used another Macintosh utility (DTCStripTabs) which added the line numbers to each source file and put its own header and footer information.

SOURCE FILE CATALOG

This catalog is arranged the order in which the files were assembled. The line and character counts are for the files after I reformatted them.

File Name	Lines	Chars
BASIC.TEXT	5	93
BASIC.RT.TEXT	5	93
BASIC.D.TEXT	5	93
BASIC.INCL.TEXT	58	2247
B3INVOK.TEXT	170	7866
ZPG.EQUS.TEXT	555	27069
B3RESVB.TEXT	394	14978
INITIAL.TEXT	570	26080
B3MAINC.TEXT	564	27430
EXTRAS.TEXT	312	13576
SOSSTUF.TEXT	351	15697
B3LISTD.TEXT	563	26574
B3GOTOE.TEXT	543	26737
B3INPUF.TEXT	599	28226
B3EVALG.TEXT	399	18518
STRUTILS.TEXT	461	19661
B3MATHK.TEXT	279	12528
B3MATHL.TEXT	517	22356
B3FINPM.TEXT	448	21543
B3EXPON.TEXT	431	18931
B3FREER.TEXT	231	11268
LONGINT.TEXT	647	28719
B3DMPYT.TEXT	228	10986
B3DIMNH.TEXT	517	25863
B3UDEFI.TEXT	378	17495
STRNGSTUF.TEXT	654	29891
INVOKE.TEXT	667	29985
INVOKE1.TEXT	346	15660
B3PRU1.TEXT	723	36115
B3PRU2.TEXT	578	26292
DISKSTUF1.TEXT	366	16105
DISKSTUF2.TEXT	303	13293
DISCMDS.TEXT	598	26914
FILESTUF.TEXT	486	22774
CATALOG.TEXT	495	20529
BASICEND.TEXT	77	3720
Total	14523	665905

BUILD INFORMATION

(from file BUILD1.3.TEXT)

9-Jun-83

Requirements:

*5 attached disks labeled;

1/BASIC1.3.MISC
1/PRODOS.EDASM
1/BASIC1.3.SRC1
1/BASIC1.3.SRC2
1/BASIC1.3.SRC3

*The following hardware;

   *Apple //e (or 48K Apple][Plus with language card in slot 0)

    2 disk drives in slot 6
    1 ProFile drive in slot 5

   *Apple /// with at least 128K of memory

   1 external disk drive

Important:

o Everything that the computer operator must type, will be underlined.
o <CR> means typing the RETURN key.
o <CTRL-Y> means hold down the CONTROL key while pressing the 'Y' key.

ALL VERSIONS

1. Boot /PRODOS.EDASM
2. Type -FILER <CR> to execute ProDos Utility Filer.
3. Transfer all the files from /BASIC1.3.SRC1 to the ProFile.
4. Transfer all the files from /BASIC1.3.SRC2 to the ProFile.
5. Transfer all the files from /BASIC1.3.SRC3 to the ProFile.
6. Exit the Utility Filer and type -EDASM <CR>.
7. If requested, enter the date in the format requested.
8. Set the PREFIX to the volume name of the ProFile.

NORMAL VERSION

1. Type ASM BASIC <CR>.
2. When it is done, type NEW <CR>.
3. Type XLOAD BASIC.0 <CR>.
4. Type MON <CR>.
5. Type 804:0C <CR>.
6. Type <CTRL-Y> <CR>.
7. Insert a SOS disk into drive 2, slot 6, and set PREFIX to the volume
    name of the SOS disk.
8. Type XSAVE SOS.INTERP <CR>. (Any legal, appropriate SOS filename may
    be (used instead of SOS.INTERP.)

9. The SOS disk now has a BASIC Interpreter on it.

RUN-TIME VERSION

1. Type ASM BASIC.RT <CR>.
2. When it is done, type NEW <CR>.
3. Type XLOAD BASIC.RT.0 <CR>.
4. Type MON <CR>.
5. Type 804:0C <CR>.
6. Type <CTRL-Y> <CR>.
7. Insert a SOS disk into drive 2, slot 6, and set PREFIX to the volume name of the SOS disk.
8. Type XSAVE SOS.INTERP <CR>. (Any legal, appropriate SOS filename may be used instead of SOS.INTERP.)
9. The SOS disk now has a Run-Time BASIC on it.

DEBUGGER VERSION

1. Type ASM BASIC.D <CR>.
2. When it is done, set the PREFIX to /PRODOS.EDASM.
3. Type EXIT <CR>.
4. Insert a SOS disk into drive 2, slot 6.
5. Type CREATE SOS.INTERP,S6,D2,T$0C <CR>.
6. Type BLOAD BASIC.D.0,A$800,S5,D1 <CR>.
7. Type BLOAD DEBUGGER.0,A$700E,S6,D1 <CR>.
8. Type BSAVE SOS.INTERP,A$800,L32103,T$0C,S6,D2 <CR>.
9. The SOS disk now has a Debugger BASIC on it.

SOURCE CODE LISTING

```
+------------------------------------------------------------------------------
|
|  File    : "BASIC.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:32 PM
|  Modified: Wednesday, December 31, 1997             6:26:31 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  BASIC.TEXT
000004 ; ###############################################################################################
000005
000006                 IBUFSIZ   6
000007                 SBUFSIZ   2
000008 DEBUG           EQU       0                       ;DEBUG Flag
000009 RUNTIME         EQU       0                       ;RUNTIME Flag
000010                 CHN       BASIC.INCL
000011
000012 ; ###############################################################################################
000013 ; #   END OF FILE:  BASIC.TEXT
000014 ; #   LINES      :  5
000015 ; #   CHARACTERS :  218
000016 ; ###############################################################################################

+------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 16   CHARACTERS: 762
|
+------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "BASIC.RT.TEXT.PRETTY"
|  Created : Wednesday, December 31, 1997             4:37:09 PM
|  Modified: Wednesday, December 31, 1997             6:26:31 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  BASIC.RT.TEXT
000004 ; ###############################################################################################
000005
000006                 IBUFSIZ   6
000007                 SBUFSIZ   2
000008 DEBUG           EQU       0                       ;DEBUG Flag
000009 RUNTIME         EQU       1                       ;RUNTIME Flag
000010                 CHN       BASIC.INCL
000011
000012 ; ###############################################################################################
000013 ; #   END OF FILE:  BASIC.RT.TEXT
000014 ; #   LINES      :  5
000015 ; #   CHARACTERS :  218
000016 ; ###############################################################################################

+------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 16   CHARACTERS: 768
|
+------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "BASIC.D.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:32 PM
|  Modified: Wednesday, December 31, 1997             6:26:30 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  BASIC.D.TEXT
000004 ; ###############################################################################################
000005
000006                 IBUFSIZ   6
000007                 SBUFSIZ   2
000008 DEBUG           EQU       1                       ;DEBUG Flag
000009 RUNTIME         EQU       0                       ;RUNTIME Flag
000010                 CHN       BASIC.INCL
000011
000012 ; ###############################################################################################
000013 ; #   END OF FILE:  BASIC.D.TEXT
000014 ; #   LINES      :  5
000015 ; #   CHARACTERS :  218
000016 ; ###############################################################################################

+------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 16   CHARACTERS: 766
|
+------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "BASIC.INCL.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:32 PM
|  Modified: Wednesday, December 31, 1997             6:26:31 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  BASIC.INCL.TEXT
000004 ; ###############################################################################################
000005
000006                 DO        RUNTIME
000007                 SBTL      "BASIC Run-Time v1.3"
000008                 ELSE
000009                 SBTL      "BASIC v1.3"
000010                 FIN
000011 * LST SIXUP ;Six Column Symbol Table
000012 ****************************************
000013 **                                    **
000014 **   Business BASIC for the Apple ///   **
000015 **                                    **
000016 **    Copyright Apple Computer, Inc.    **
000017 **       1980, 1981, 1982, 1983         **
000018 **         All Rights Reserved          **
000019 **                                    **
000020 ****************************************
000021                 INCLUDE   B3INVOK
000022                 INCLUDE   ZPG.EQUS
000023                 INCLUDE   B3RESVB
000024                 INCLUDE   INITIAL
000025                 INCLUDE   B3MAINC
000026                 INCLUDE   EXTRAS
000027                 INCLUDE   SOSSTUF
000028                 INCLUDE   B3LISTD
000029                 INCLUDE   B3GOTOE
000030                 INCLUDE   B3INPUF
000031                 INCLUDE   B3EVALG
000032                 INCLUDE   STRUTILS
000033                 INCLUDE   B3MATHK
000034                 INCLUDE   B3MATHL
000035                 INCLUDE   B3FINPM
000036                 INCLUDE   B3EXPON
000037                 INCLUDE   B3FREER
000038                 INCLUDE   LONGINT
000039                 INCLUDE   B3DMPYT
000040                 INCLUDE   B3DIMNH
000041                 INCLUDE   B3UDEFI
000042                 INCLUDE   STRNGSTUF
000043                 INCLUDE   INVOKE
000044                 INCLUDE   INVOKE1
000045                 INCLUDE   B3PRU1
000046                 INCLUDE   B3PRU2
000047                 INCLUDE   DISKSTUF1
000048                 INCLUDE   DISKSTUF2
000049                 INCLUDE   DISCMDS
000050                 INCLUDE   FILESTUF
000051                 INCLUDE   CATALOG
000052                 INCLUDE   BASICEND
000053
000054 ; ###############################################################################################
000055 ; #   END OF FILE:  BASIC.INCL.TEXT
000056 ; #   LINES      :  47
000057 ; #   CHARACTERS :  1633
000058 ; ###############################################################################################

+------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 58   CHARACTERS: 2189
|
+------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3INVOK.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:28 PM
|  Modified: Wednesday, December 31, 1997             4:37:05 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3INVOK.TEXT
000004 ; ###############################################################################################
000005
000006                SBTL      "SYSTEM EQUATES" "
000007 *         MAP OF USER AREA, MAIN MEMORY
000008 *
000009 *            -------------    TXTTAB  ($39)
000010 *           |   PROGRAM   |
000011 *           |-------------|   ARYTAB  ($3D)
000012 *           |    ARRAYS   |
000013 *           |-------------|   VARTAB (& SMVARS) ($3B)
000014 *           | SIMPLE VARS |
000015 *           |             |
000016 *           |      |      |
000017 *           |      V      |
000018 *           |             |   STREND  ($3F) (Floating Pointer to end of strings)
000019 *           ~             ~
000020 *           |      ^      |   FRESPC  ($47) (Floating Pointer to end of variables)
000021 *           |      |      |
000022 *           |             |
000023 *           |   STRINGS   |
000024 *           |-------------|   FRETOP  ($41)
000025 *           |   BUFFERS   |
000026 *           |-------------|   INVTAB  ($43)
000027 *           | INVOKABLES  |
000028 *           |-------------|   PROCTAB ($45)
000029 *           |   BUFFERS   |
000030 *            -------------    MEMSIZ  ($49)
000031 *
000032 *  All areas are RELATIVE to the beginning (or end) of memory, and
000033 *  pointers to within the areas are RELATIVE to the beginning of the area.
000034 *
000035 *       SOS STACK RESIDES AT $FF ON PAGE $17
000036 *     BASIC STACK RESIDES AT $FF ON PAGE $1B
000037 *
000038 *   Format of Variable Entries   (Pointers are all Relative)
000039 *
000040 * Simple Variable Entries:                    Real values take 4 Bytes
000041 *     ------------------------------          Integer values take 2 Bytes
000042 *    | Length | Name | Type | Value |         Long Integer values take 8 Bytes
000043 *     ------------------------------
000044 *         1      n       1       n    Bytes
000045 *
000046 * Array Variable Entries:
000047 *     ------------------------------------------------------------------>
000048 *    | Length | Name | Type |Dim Count| Dim Size | Value 0 | Value 1|
000049 *     ------------------------------------------------------------------>
000050 *         1      n      1        1      2 per Dim     n           n ...
000051 *
000052 * String Variable Entries:
000053 *     -------------------------------------------------
000054 *    | Length | Name | Type | String length | Pointer |
000055 *     -------------------------------------------------
000056 *         1      n      1           1            2   Bytes
000057 *
000058 * String Entries:
000059 *     ------------------------------    The Svar Type Byte indicates if the
000060 *    | String | Svar Type | Pointer |     string is a simple or array variable.
000061 *     ------------------------------     The Pointer points to descriptor's
000062 *         n         1          2     Bytes        byte for type checking.
000063                PAGE
000064                MSB       OFF
000065 *
000066 * HERE ARE SOS CALLS INTERFACE STUFF- EQUATES, ETC.
000067 *
000068 SCRT            EQU       $C0                     ;CREATE
000069 SDST            EQU       $C1                     ;DESTROY
000070 SRNM            EQU       $C2                     ;RENAME
000071 SSFI            EQU       $C3                     ;SET.FILE.INFO
```

```
000072 SGFI            EQU       $C4                     ;GET.FILE.INFO
000073 SVLM            EQU       $C5                     ;VOLUME
000074 SETPREF         EQU       $C6                     ;SET PREFIX
000075 GETPREF         EQU       $C7                     ;GET PREFIX
000076 SOPN            EQU       $C8                     ;OPEN
000077 SNWL            EQU       $C9                     ;NEW.LINE
000078 SRED            EQU       $CA                     ;READ
000079 SWRT            EQU       $CB                     ;WRITE
000080 SCLS            EQU       $CC                     ;CLOSE
000081 SFLS            EQU       $CD                     ;FLUSH
000082 SSTM            EQU       $CE                     ;SET.MARK
000083 SGTM            EQU       $CF                     ;GET.MARK
000084 SSTE            EQU       $D0                     ;SET.EOF
000085 SGTE            EQU       $D1                     ;GET.EOF
000086 SSLVL           EQU       $D2                     ;SET LEVEL
000087 SGLVL           EQU       $D3                     ;GET.LEVEL
000088 SDSTAT          EQU       $82                     ;DEVICE STATUS.
000089 SDCNT           EQU       $83                     ;SOS DEVICE CONTROL
000090 SDGDN           EQU       $84                     ;SOS GET DEVICE NUM
000091 MREQ            EQU       $40                     ;REQUEST.SEG
000092 MCHG            EQU       $42                     ;CHANGE.SEG
000093 MFND            EQU       $41                     ;FIND.SEG
000094 MRLS            EQU       $45                     ;RELEASE.SEG
000095 GETCLOK         EQU       $63                     ;GET.CLOCK
000096 CLDSTRT         EQU       $65                     ;COLD.START
000097 *
000098 * ERROR NUMBERS FROM SOS:
000099 *
000100 SEMEM           EQU       $54                     ;OUT OF FREE MEMORY.
000101 SEBDP           EQU       $40                     ;BAD PATH NAME
000102 SEFNF           EQU       $46                     ;FILE NOT FOUND
000103 SEEOF           EQU       $4C                     ;END OF FILE ERR
000104 SEFNO           EQU       $43                     ;FILE NOT OPEN
000105 SENBK           EQU       $58                     ;NOT A BLOCK DEVICE
000106 SEDFU           EQU       $48                     ;DISK FULL ERROR
000107                 PAGE
000108 *
000109 * TYPE EQUATES:
000110 *
000111 PRGTY           EQU       $09                     ;BASIC PROGRAM TYPE
000112 TXTTYP          EQU       $04                     ;TEXT FILE TYPE
000113 BINTIP          EQU       10                      ;BINARY DATA TYPE
000114 UNKNTY          EQU       $0                      ;UNKNOWN TYPE
000115 PCODTYP         EQU       2                       ;PASCAL CODE
000116 *
000117 * 0=UNKNOWN       1=BAD FILE   2=CODE FILE   3=UCSD TEXT   4=ASCII
000118 * 5=PASCAL DATA   6=BINARY     7=FONT        8=FOTO         9=BASIC PROGRAM
000119 * 10=BASIC DATA  11=WPTEXT    12=SYSTEM     13=RESERVED   14=RESERVED
000120 * 15=DIRECTORY   16=RPS DATA  17=RPS INDEX  18=AFDISCARD  19=AFMODEL
000121 * 20=AF RPT FMT  21=SCREEN LIB
000122 *        224 ($E0) to 255 ($FF) Reserved for PRODOS.
000123                 PAGE
000124 *
000125 * Here is the File Control Block definition (FCB):
000126 *
000127 XRFNM           EQU       0                       ;REFERENCE NUM FOR SOS FOR FILE#N
000128 XUID            EQU       1                       ;BITS 0-3 => TYPE OF FILE
000129 *
000130 *    BIT 4 =>($10 MASK) READ ALLOWED
000131 *    BIT 5 =>($20 MASK) WRITE ALLOWED
000132 *
000133 XBUFPT          EQU       2                       ;POINTER TO BUFFER AREA
000134 XBUFOFS         EQU       4                       ;OFFSET INTO FILE BUFFER
000135 *
000136 * -NOTICE THAT XBUFOFS ALWAYS STARTS AT 0000 BECAUSE
000137 *      SOS DOES ALL THE MESSY WORK.
000138 *
000139 XRNUM           EQU       6                       ;RECORD NUMBER
000140 XRECL           EQU       8                       ;RECORD LENGTH (DEFAULT = 512)
000141 *
000142 *  Position in File (for SOS) = RECNUM * RECLEN
000143 *
000144 XFLGS           EQU       10                      ;HOLDS FLAGS AS FOLLOWS:
000145 *
000146 * BIT 7 => DATA HAS BEEN MODIFIED AND SHOULD BE WRITTEN OUT
000147 * BIT 6 => OPEN OPERATION IS NOT YET COMPLETE (BINARY/TEXT UNDETERMINED)
000148 *
000149 * IF A FILE IS A DIRECTORY TYPE (READING A CATALOG) THEN THE FILE
000150 * WILL APPEAR TO BE A TEXT FILE, AND BITS 0-2 OF XFLGS WILL REPRESENT
000151 * THE STAGE OF CATALOG.
```

```
000152 *
000153 XSEGNM          EQU       11                    ;HOLDS SEGNUM RETURNED BY THE BUFFER MANAGER
000154 XBLKS           EQU       12                    ;FOR A ROOT DIRECTORY FILE HOLDS TOTAL BLOCKS.
000155 FCBLEN          EQU       14                    ;LEN OF EACH ENTRY IN FCB.
000156 *
000157 * HERE ARE THE DATA DESCRIPTORS
000158 *
000159 DDINT           EQU       $12
000160 DDFP            EQU       $14
000161 DDLNT           EQU       $18
000162 DDSTR           EQU       $21
000163 DDMXSTR         EQU       $20
000164 DDEOR           EQU       $00                   ;MUST BE 0 TO MATCH SOS
000165
000166 ; ##############################################################################################
000167 ; #   END OF FILE:  B3INVOK.TEXT
000168 ; #   LINES      :  159
000169 ; #   CHARACTERS :  7145
000170 ; ##############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 170   CHARACTERS: 7696
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "ZPG.EQUS.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:39 PM
|  Modified: Wednesday, December 31, 1997             4:37:16 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  ZPG.EQUS.TEXT
000004 ; ###############################################################################################
000005
000006 STKEND          EQU       $FE
000007 EOFSIZ          EQU       33
000008                 DO        DEBUG
000009 B3PRT1          EQU       $3A00
000010                 ELSE
000011 B3PRT1          EQU       $5000
000012                 FIN
000013 *   $3A00 For DEBUGGER Version
000014 *   $5000 For NORMAL(?) Version
000015 *        & goes up to $B7FF
000016                 DSECT
000017                 SBTL      "Zero Page Stuff" "
000018                 ORG       0
000019 SELFLG          EQU       $EE                     ;Value put in 0,1 by Selector
000020 NUMLEV          EQU       19                      ;NUMBER OF STACK LEVELS RESERVED
000021 TEMPTYP         EQU       1                       ;Flag Byte for STRING temporary.
000022 ARYTYP          EQU       $81                     ;STRING IS MEMBER OF AN ARRAY FLAG.
000023 SIMTYP          EQU       $41
000024 MINPG           EQU       2                       ;MIN High Byte Virtual Pointer.
000025 MAXPG           EQU       $82                     ;MAX High Byte Virtual Pointer (PAGE POINTER).
000026 INFOSIZ         EQU       3                       ;EACH STRING CONTAINS 3 INFO BYTES.
000027 STRSIZ          EQU       3                       ;# OF LOCS PER STRING DESCRIPTOR.
000028 NUMTMP          EQU       4                       ;NUMBER OF STRING TEMPORARIES.
000029                 JMP       INIT                    ;THIS CODE NOT HERE.
000030 ;
000031 ; THIS IS THE 'VOLATILE' STORAGE AREA AND NONE OF IT CAN BE KEPT IN ROM.
000032 ;
000033 ;     --- GENERAL RAM ---:
000034 CHARAC:         DS        1                       ;A DELIMITING CHARACTER.
000035 INTEGR          EQU       CHARAC                  ;A ONE-BYTE INTEGER FROM 'QINT'
000036 ENDCHR:         DS        1                       ;THE OTHER DELIMITING CHARACTER.
000037 DSCRPT          EQU       ENDCHR                  ;DESCRIPTOR FOR FILE OPERATIONS
000038 COUNT:          DS        1                       ;A GENERAL COUNTER.
000039 ;     --- FLAGS ---:
000040 DIMFLG:         DS        1                       ;IN GETTING A PNTER TO A VARIABLE
000041 ;   DIMFLG AND VALTYP MUST BE KEPT IN CONSECUTIVE LOCATIONS
000042 ;
000043 KIMY            EQU       DIMFLG                  ;PLACE TO PRESERVE Y DURING OUT.
000044 VALTYP:         DS        1                       ;THE TYPE INDICATOR.
000045 ;    0=NUMERIC 1=STRING.
000046 INTFLG:         DS        1                       ;TELLS IF INTEGER.
000047 DORES:          DS        1                       ;WHETHER CAN OR CAN'T CRUNCH RES'D WORDS.
000048 ;   TURNED ON WHEN 'DATA' BEING SCANNED BY CRUNCH SO UNQUOTED
000049 ;    STRINGS WON'T BE CRUNCHED.
000050 ;
000051 GARBFL          EQU       DORES                   ;Whether to do garbage Collection.
000052 XSAV            EQU       DORES
000053 SUBFLG:         DS        1                       ;FLAG WHETHER SUB'D VARIABLE ALLOWED.
000054 YSAVE           DS        1
000055 INPFLG:         DS        1                       ;FLAGS WHETHER WE ARE DOING 'INPUT' OR 'READ'
000056 TANSGN:         DS        1                       ;USED IN DETERMINING SIGN OF TANGENT.
000057 ANYNUM          EQU       TANSGN                  ;FLAG IF ANY DIGITS DURING 'FIN'
000058 FILNO:          DS        2                       ;HOLDS THE FILE NUMBER FOR
                                                                 OUTPUT(FOR PRINT#, ETC)
000059 ;    ALSO USED BY OUTPUT#, WHOSE FILE NUMBER IS IN FILNO+1
000060 ;    IF FILNO IS NEGATIVE, OUTPUTS TO THE CURRENT OUTPUT DEVICE
000061 INFLNO:         DS        1                       ;FILE REFERENCE NUMBER FOR 'EXEC'
000062 SVFLNO          DS        1                       ;SAVES THE FILE # OF CURRENT OPERATION
000063 JMPER:          JMP       60000
000064 DELTA           DS        2                       ;For moves
000065 ;     --- STUFF USED IN EVALUATIONS ---:
000066 VARNAM:         DS        2                       ;VARIABLE'S NAME IS STORED HERE.
000067 VARPNT:         DS        2                       ;POINTER TO VARIABLE IN MEMORY.
000068 OPPTR:          DS        2                       ;POINTER TO CURRENT OP'S ENTRY IN 'OPTAB'.
000069 VARTXT          EQU       OPPTR                   ;POINTER INTO LIST OF VARIABLES
000070 DOMASK          EQU       TANSGN                  ;MASK IN USE BY RELATION OPERATIONS.
000071 DEFPNT:         DS        2                       ;POINTER USED IN FUNCTION DEFINITION.
```

```
000072 GRBPNT          EQU       DEFPNT                    ;Another used in Garbage Collection.
000073 SLEFT           DFB       0
000074 FCBNDX          EQU       SLEFT
000075 SWIDTH          DFB       0
000076 SBOTTOM         DFB       0
000077 STOPS           DFB       0
000078 TRMPOS          DS        1                         ;HOLDS TERMINAL POSITION
000079 LINNUM:         DW        0                         ;LOCATION TO STORE LINE NUMBER
000080 ; --- STORAGE FOR TEMPORARY THINGS ---:
000081 TEMPPT:         DS        1                         ;POINTER AT FIRST FREE TEMP DESCRIPTOR.
000082 ;   INITIALIZE TO POINT TO TEMPST.
000083 LASTPT:         DS        2                         ;POINTER TO LAST-USED STRING TEMPORARY.
000084 TEMPST:         DS        STRSIZ*NUMTMP             ;STORAGE FOR NUMTMP TEMP DESCRPT
000085 INDEX1:         DS        2                         ;INDEXES.
000086 INDEX           EQU       INDEX1
000087 INDEX2:         DS        2
000088 ;     --- POINTERS INTO DYNAMIC DATA STRUCTURES ---;
000089 TXTTAB:         DS        2                         ;POINTER TO BEGINNING OF TEXT.
000090 VARTAB:         DS        2                         ;POINTER TO START OF SIMPLE VARIABLE SPACE
000091 ; VARTAB IS UPDATED WHENEVER THE SIZE OF THE PROGRAM CHANGES;
000092 ;    SET TO TXTTAB BY 'SCRATCH' ('NEW').
000093 ;
000094 SMVARS          EQU       VARTAB                    ;POINTS TO THE SIMPLE VARIABLE TABLE.
000095 ARYTAB:         DS        2                         ;POINTER TO BEGINNING OF ARRAY TABLE
000096 ; ARYTAB IS INCREMENTED BY 6 WHENEVER A NEW SIMPLE VARIABLE IS FOUND,
000097 ;    AND SET TO VARTAB BY 'CLEARC'.
000098 ;
000099 STREND:         DS        2                         ;END OF STORAGE IN USE.
000100 ; STREND IS INCREASED WHENEVER A NEW ARRAY OR SIMPLE VARIABLE IS ENCOUNTERED
000101 ;
000102 ;SET TO VARTAB BY 'CLEARC'.
000103 FRETOP:         DS        2                         ;TOP OF STRING FREE SPACE.
000104 INVTAB          DS        2                         ;TABLE OF INVOKABLE ENTRY POINTS.
000105 PROCTAB         DS        2                         ;TABLE OF PERFORMABLE CODE MODULES.
000106 FRESPC:         DS        2                         ;POINTER TO NEW STRING. NOT THE SAME AS FRSPCE!
000107 MEMSIZ:         DS        2                         ;HIGHEST LOCATION IN MEMORY.
000108 ;     --- LINE NUMBERS AND TEXTUAL POINTERS ---:
000109 HIMEM           EQU       MEMSIZ
000110 CURLIN:         DS        2                         ;CURRENT LINE #.
000111 ;    SET TO 0,255 FOR DIRECT STATEMENTS.
000112 OLDLIN:         DS        2                         ;OLD LINE NUMBER (SETUP BY C,'STOP'
000113 ;    OR 'END' IN A PROGRAM).
000114 POKER           EQU       LINNUM
000115 ;    TEMPORARY FOR INPUT AND READ CODE
000116 OLDTXT:         DS        2                         ;OLD TEXT POINTER.
000117 ;    POINTS AT STATEMENT TO BE EXEC'D NEXT.
000118 DATLIN:         DS        2                         ;DATA LINE # -- REMEMBER FOR ERRORS.
000119 DATPTR:         DS        2                         ;POINTER TO DATA.
000120 ; DATPTR IS INITIALIZED TO POINT AT THE ZERO IN FRONT OF TXTTAB BY 'RESTORE',
000121 ;    WHICH IS CALLED BY 'CLEARC',AND IS UPDATED BY EXECUTION OF A 'READ'.
000122 ;
000123 INPPTR:         DS        2                         ;THIS REMEMBERS WHERE INPUT IS COMING FROM.
000124 FDECPT          EQU       VARPNT                    ;POINTER INTO POWERF TENS OF 'FOUT'.
000125 FORPNT:         DS        2                         ;A VARIABLE'S POINTER FOR 'FOR' LOOPS
                                                                        AND 'LET' STATEMENTS
000126 LSTPNT          EQU       FORPNT                    ;PNTR TO LIST STRING.
000127 TKNSAV          EQU       FORPNT+1                  ;USED TO PRESERVE THE TOKEN # IN LIST.
000128 TYPSAV          EQU       OLDLIN+1                  ;USED IN DISK I/O
000129 DSCPNT:         DS        2                         ;POINTER TO A STRING DESCRIPTOR.
000130                 DS        1                         ;FOR TEMPF3.
000131 FOUR6:          DFB       STRSIZ                    ;VARIABLE CONSTANT USED BY GARB COLECT
000132 ;     --- ET CETERA ---:
000133 SIZE            EQU       JMPER+1
000134 OLDOV           EQU       JMPER+2                   ;THE OLD OVERFLOW.
000135 TEMPF3          EQU       DEFPNT                    ;A THIRD FAC TEMPORARY (5 BYTES).
000136 TEMPF1:         DFB       0                         ;FOR TEMPF1S EXTRA BYTE.
000137 HIGHDS:         DS        2                         ;DESINATION OF HIGHEST ELEMENT IN BLT.
000138 PTR1            EQU       HIGHDS
000139 NDXPTR          DS        2                         ;USED IN DISK I/O
000140 HIGHTR:         DS        2                         ;SOURCE OF HIGHEST ELEMENT TO MOVE.
000141 PTR2            EQU       HIGHTR
000142 TEMPF2:         DFB       0                         ;FOREMPF2S EXTRA BYTE.
000143 LENSAV          EQU       TEMPF2                    ;USED IN DISK I/O
000144 LOWDS:          DS        2                         ;LOCATION OF LAST BYTE TRANSFERRED INTO.
000145 LOWTR:          DS        2                         ;LAST THING TO MOVE IN BLT.
000146 PTR3            EQU       LOWDS
000147 SRCHPT          EQU       HIGHTR                    ;JUST A TEMP FOR SCANNING THROUGH VARIABLE NAMES.
000148 ; --- ORDER OF VARS IS: ARRAYS,SIMPLES,...PROG...STRINGS.
000149 ;
000150 ARYPNT          EQU       HIGHDS                    ;A POINTER USED IN ARRAY BUILDING.
```

```
000151 GRBTOP          EQU       LOWTR                ;A POINTER USED IN GARBAGE COLLECTION.
000152 DECCNT          EQU       LOWDS                ;NUMBER OF PLACES BEFORE DECIMAL POINT.
000153 TENEXP          EQU       LOWDS+1              ;HAS A DPT BEEN INPUT?
000154 DPTFLG          EQU       LOWTR                ;BASE TEN EXPONENT.
000155 EXPSGN          EQU       LOWTR+1              ;SIGN OF BASE TEN EXPONENT.
000156 ;     --- THE FLOATING ACCUMULATOR ---:
000157 FAC:            EQU       *
000158 FACEXP:         DFB       0
000159 FACHO:          DFB       0                    ;MOST SIGNIFICANT BYTE OF MANTISSA.
000160 FACMOH:         DFB       0                    ;ONE MORE.
000161 FACT:           EQU       FACMOH               ;OVERLAP MANTISSA & EXPONENTS OF BCD&BIN
000162 FACMO:          DFB       0                    ;MIDDLE ORDER OF MANTISSA.
000163 FACLO:          DFB       0                    ;LEAST SIG BYTE OF MANTISSA.
000164 FACSGN:         DFB       0                    ;SIGN OF FAC (0 OR -1) WHEN UNPACKED.
000165 SGNFLG:         DFB       0                    ;SIGN OF FAC IS PRESERVED BERE BY 'FIN'.
000166 DEGREE          EQU       SGNFLG               ;A COUNT USED BY POLYNOMIALS.
000167 DSCTMP          EQU       FAC                  ;THIS IS WHERE TEMP DESCS ARE BUILT.
000168 INDICE          EQU       FACMO                ;INDICE IS SET UP HERE BY 'QINT'.
000169                 DS        3                    ;FOR THE REST OF THE FAC.
000170 CNTDIGS         DFB       0                    ;FOR ROUNDING AT 10 DIGITS.
000171                 DFB       0                    ;WHY NOT?
000172 ARGEXP:         DFB       0
000173 ARGHO:          DFB       0
000174 ARGMOH:         DFB       0
000175 ARG:            EQU       ARGEXP               ; THE FLOATING POINT ARGUMENT
000176 ARGMO:          DFB       0
000177 ARGLO:          DFB       0
000178 ARGSGN:         DFB       0
000179 ARISGN:         DFB       0                    ;A SIGN REFLECTING THE RESULT.
000180                 DFB       0
000181                 DFB       0
000182 RESHO:          DS        1                    ;RESULT OF MULTIPLIER AND DIVIDER.
000183 RESMOH:         DS        1                    ;ONE MORE BYTE.
000184 RESMO:          DS        1
000185 RESLO:          DS        1
000186 ADDEND          EQU       RESMO                ;TEMPORARY USED BY 'UMULT'.
000187 TEMP            DFB       0                    ;OVERFLOW FOR RES.
000188 RES             EQU       RESHO
000189 FACOV:          DFB       0                    ;OVERFLOW BYTE OF THE FAC.
000190                 DS        3
000191 STRNG1          EQU       ARISGN               ;POINTER TO A STRING OR DESCRIPTOR.
000192 FBUFPT:         DS        2                    ;POINTER INTO FBUFFR USED BY FOUT.
000193 BUFPTR          EQU       FBUFPT               ;POINTER TO BUF USED BY 'CRUNCH'.
000194 STRNG2          EQU       FBUFPT               ;POINTER TO STRING OR DESC.
000195 POLYPT          EQU       FBUFPT               ;POINTER INTO POLYNOMIAL COEFFICIENTS.
000196 CURTOL          EQU       FBUFPT               ;USED BY DIM,PTRGET.
000197 TRFLAG:         DS        1
000198 TEMPFOR         DS        1
000199 ERRTO:          DS        5
000200 ERRLIN:         DS        2
000201 ERRPOS:         DS        3
000202 ERRNUM:         DS        1                    ;PLACE FOR ERROR #
000203 ERRFLG:         DS        1                    ;NEG IF ONERR MODE, V BIT SET FOR ON KBD
000204 REMSTK:         DS        1                    ;SAVE STACK POINTER IN CASE OF ERROR
000205 RNFLG           DS        1                    ;RUN ONLY FLAG
000206 NOUNPT:         DS        1
000207 VRBPT:          DS        1
000208 HEADER          DS        2
000209 PNTSAV          DW        0
000210 LVLCNT          DS        1                    ;FOR NESTED IF..THEN..ELSE
000211 QUOTE           EQU       LVLCNT
000212 TMPPTR          DW        0                    ;IN PTRGET
000213 STRFLG          DFB       0
000214 INVPNT          DS        2
000215 NPOINTS         DFB       0                    ;THE NEXT 3 GUYS ARE TEMPS USED BY
000216 NPARAMS         DFB       0                    ;INVOKE,PERFORM AND MUST SURVIVE
000217 PROCFLG         DFB       0                    ;FRMEVL!!!
000218 IOFLG           EQU       NPOINTS              ;WHETHER DOING INPUT OR OUTPUT TO DISK
000219 KEYSTROK        DFB       0                    ;PERMANENT.  SET IF KEY HIT.
000220 MLTPLR          DW        0                    ;THESE BYTES MUST BE CONSECUTIVE FOR
                                                                  MUL& DIV TO WORK
000221 RSLT            EQU       MLTPLR               ;AND IN THIS ORDER
000222 RMNDR           DW        0
000223 MLTPLR2         DW        0
000224 QUOTNT          EQU       MLTPLR
000225 DVDND           EQU       MLTPLR
000226 DVSR            EQU       MLTPLR2
000227 BITS:           DFB       0                    ;SOMETHING FOR 'SHIFTR' TO USE.
000228                 DS        5
000229 CHRGET:         INC       CHRGET+7             ;BECAUSE SARA GOES BY WHETHER THE ADDR
```

```
000230                  BNE       CHRGOT                    ;IS GOING THROUGH ZERO PAGE,
                                                                    THIS GOES THROUGH THE
000231                  INC       CHRGET+8                  ;BANK STUFF ALSO. VERY FAST CODE USED EVERYWHERE
000232 CHRGOT:          LDA       60000
000233 TXTPTR           EQU       CHRGOT+1
000234                  DS        15                        ;ROOM FOR REST OF ROUTINE. FULL LISTING IN 'INIT'
000235 RNDX             DS        8                         ;RANDOM NUMBER GOES HERE.
000236 NAMPNT           DFB       0                         ;FOR RECURSIVE EXFN'S.
000237 CMDFLG           DFB       0                         ;Flag indicating RUN or CHAIN
000238                  ORG       $E4                       ;THIS MUST BE COMPATABLE WITH PASCAL.
000239 DISPATCH         DS        3
000240 PASSAREG         DFB       0
000241 BANKPNT          DS        16                        ;VAR PARAMETERS GET POINTERS STACKED HERE
000242                  ORG       255                       ;PAGE 1 STUFF COMING UP.
000243 ;    --- PAGE ZERO/ONE BOUNDARY ---.
000244 ;    Stack is located here. I.e., from the end of FBUFFR to STKEND.
000245                  SBTL      "DISPATCH TABLES" "
000246                  REP       53
000247 **                                              **
000248 **               FIRST PART OF BASIC            **
000249 **                                              **
000250                  REP       53
000251 CRUDBUF          EQU       $1E00                     ;THIS PAGE IS FREE!!
000252                  ORG       CRUDBUF
000253                  JMP       SWCHGO
000254                  DFB       0
000255 SAFE             DFB       0,0,0                     ;THESE THREE BYTES NEEDED FOR PERFORM.
000256 RAMLOC           DW        0
000257 RAMLOCB          DFB       0
000258 VRBSTK           DS        32
000259 EOFPTRS          DS        EOFSIZ
000260 EOFLINS          DS        EOFSIZ-1
000261 FCB              DS        FCBLEN*10                 ;HOLD FILE FCBS HERE
000262 CATFCB           DS        FCBLEN                    ;CATALOG PRETENDS TO BE FILE #11
000263 INVBNK           DFB       0
000264 BASICBNK         DFB       0
000265 ISARA            DFB       0
000266 NOUNSTK          DS        128                       ;FORMULA EVALUATION STACK.
000267 SOSLOC           DFB       0                         ;SOS ERROR NUMBER GOES HERE.
000268 OUTREC           DFB       0                         ;OUTPUT RECORD LENGTH FOR LIST.
000269 INDENT           DFB       0                         ;# OF SPACES TO INDENT IN LIST.
000270 EOFSV            DS        1                         ;LAST EOF ENCOUNTERED...
000271                  DEND
000272                  ORG       B3PRT1-$0E                ;Allow room for the header
000273                  ASC       "SOS NTRP" ;SOS file label"
000274                  DW        0000                      ;No optional header
000275                  DW        B3PRT1                    ;Start of real code (Load address)
000276                  DW        BASICEND-GOBASIC          ;Length of BASIC
000277 GOBASIC          JMP       INIT
000278 ;    PUT VARIOUS BUFFERS HERE BEFORE BASIC
000279 SELECTOR         DFB       0                         ;If 0 then boot & run, else SELECTOR
000280 PROGPATH         DFB       0                         ;Program pathname from Selector
000281                  DS        80,0
000282 SOSPATH          DFB       0                         ;SOS pathname from Selector
000283                  DS        80,0
000284 NAMBUF           DS        128
000285                  DFB       80                        ;Length of CATBUF.
000286 CATBUF           DS        80                        ;Actual Catalog output buffer
000287 LENUM            DFB       0
000288 BCDSTR           DS        48
000289 NUMSTR           EQU       BCDSTR
000290 FBUFFR           EQU       BCDSTR+28
000291 LOFBUF           EQU       FBUFFR-1
000292                  DS        3                         ;FOR CRUNCH.
000293 PREBUF           EQU       LOFBUF                    ;END OF PREFIX$ BUFFER (STARTS AT NOUNSTK).
000294 BUF              DS        256
000295 KEYSAVE          DFB       13                        ;KEYBOARD SAVED HERE
000296 STMDSP           DW        END-1
000297                  DW        FOR-1
000298                  DW        NEXT-1
000299                  DW        INPUT-1
000300                  DW        OUTPUT-1
000301                  DW        DIM-1
000302                  DW        READ-1
000303                  DW        DWRITE-1
000304                  DW        DOPEN-1
000305                  DW        DCLOSE-1
000306                  DW        SNERR-1
000307                  DW        MSETTXT-1
000308                  DW        SNERR-1
```

```
000309                  DW      COLD-1
000310                  DW      SNERR-1
000311                  DW      SNERR-1
000312                  DW      SNERR-1
000313                  DW      SNERR-1
000314                  DW      SNERR-1
000315                  DW      WINDOW-1
000316                  DW      INVOKE-1
000317                  DW      PERFORM-1
000318                  DW      SNERR-1
000319                  DW      SNERR-1
000320                  DW      SNERR-1
000321                  DW      HTAB-1
000322                  DW      VTAB-1
000323                  DW      SNERR-1
000324                  DW      SNERR-1
000325                  DW      SNERR-1
000326                  DW      SNERR-1
000327                  DW      SNERR-1
000328                  DW      SNERR-1
000329                  DW      PREFIXSET-1
000330                  DW      SNERR-1
000331                  DW      SNERR-1
000332                  DW      SOUTREC-1
000333                  DW      SINDENT-1
000334                  DW      PROGPFX-1
000335                  DW      SNERR-1
000336                  DW      SNERR-1
000337                  DW      SNERR-1
000338                  DW      SNERR-1
000339                  DW      SNERR-1
000340                  DW      SNERR-1
000341                  DW      RETURN-1
000342                  DW      HOME-1
000343                  DO      DEBUG
000344                  DW      $A1FF                   ;$A1FF FOR "SHIT", SNERR-1 for NORMAL
000345                  ELSE
000346                  DW      SNERR-1
000347                  FIN
000348                  DW      SUBLEFT-1
000349                  DW      OFF-1
000350                  DW      SETTRACE-1
000351                  DW      TRACEOFF-1
000352                  DW      SETNORM-1
000353                  DW      INVERSE-1
000354                  DW      SNERR-1
000355                  DW      RESUME-1
000356                  DW      SNERR-1
000357                  DW      LET-1
000358                  DW      GOTO-1
000359                  DW      IF-1
000360                  DW      RESTOR-1
000361                  DW      SWAP-1
000362                  DW      GOSUB-1
000363                  DW      RETURN-1
000364                  DW      REM-1
000365                  DW      STOP-1
000366                  DW      ONGOTO-1
000367                  DW      SNERR-1
000368                  DW      LOAD-1
000369                  DW      SAVE-1
000370                  DW      DDELETE-1               ;DISK DELETE
000371                  DW      RUN-1
000372                  DW      RENAME-1
000373                  DW      LOCK-1
000374                  DW      UNLOCK-1
000375                  DW      CREATE-1
000376                  DW      EXEC-1
000377                  DW      CHAIN-1
000378                  DW      SNERR-1
000379                  DW      SNERR-1
000380                  DW      SNERR-1                 ;FOR EXPANSION
000381                  DW      CATALOG-1               ;This one is for CATALOG
000382                  DW      SNERR-1
000383                  DW      SNERR-1
000384                  DW      DATAIS-1
000385                  DW      REM-1                   ;REM FOR THE 'IMAGE' STATEMENT
000386                  DW      CATALOG-1               ;This one is for CAT
000387                  DW      DEF-1
000388                  DW      SNERR-1
```

```
000389                  DW      DOPRINT-1
000390                  DW      DELETE-1
000391                  DW      REM-1                   ;FOR THE ELSE STATEMENT
000392                  DW      CONT-1
000393                  DW      LIST-1
000394                  DW      CLEAR-1
000395                  DW      GET-1                   ;FILL W/ GET ADDR.
000396                  DW      SCRATH-1
000397 RESTBL            EQU     *
000398                  DW      NOFREF
000399                  DW      POS                     ;HPOS
000400                  DW      DOVPOS
000401                  DW      DOERRLN
000402                  DW      GIVERR
000403                  DW      GIVKBD
000404                  DW      GIVEOF
000405                  DW      TIMES
000406                  DW      DATES
000407                  DW      PREFIXS
000408                  DW      EXFN
000409                  DW      EXFNS
000410                  DW      GIVOUTREC
000411                  DW      GIVINDENT
000412                  DW      PROGPFXS
000413 FUNDSP:           EQU     *
000414                  DW      SGN
000415                  DW      INT
000416                  DW      ABS
000417                  DW      SNERR                   ;NO USR()!
000418                  DW      TYP
000419                  DW      REC
000420                  DW      SNERR                   ;FOR EXPANSION
000421                  DW      SNERR
000422                  DW      SNERR
000423                  DW      SNERR
000424                  DW      SNERR
000425                  DW      SNERR
000426                  DW      SNERR
000427                  DW      SNERR
000428                  DW      SNERR
000429                  DW      SNERR
000430                  DW      PDLHNDL
000431                  DW      BUTTON
000432                  DW      SQR
000433                  DW      RND
000434                  DW      LOG
000435                  DW      EXP
000436                  DW      COS
000437                  DW      SIN
000438                  DW      TAN
000439                  DW      ATN
000440                  DW      SNERR
000441                  DW      SNERR
000442                  DW      SNERR
000443                  DW      SNERR
000444                  DW      SNERR
000445                  DW      SNERR
000446                  DW      SNERR
000447                  DW      SNERR
000448                  DW      SNERR
000449                  DW      SNERR
000450                  DW      SNERR
000451                  DW      SNERR
000452                  DW      STRS
000453                  DW      HEXS
000454                  DW      CHRS
000455                  DW      LEN
000456                  DW      VAL
000457                  DW      ASC
000458                  DW      DECER
000459                  DW      SNERR
000460                  DW      SNERR
000461                  DW      CONV2FLT
000462                  DW      CONV2LNG
000463                  DW      CONV2STR
000464                  DW      CONV2INT
000465                  DW      LEFTS
000466                  DW      RIGHTS
000467                  DW      MIDS
000468                  DW      INSTR
```

```
000469 PRECTB:          DFB       4,4,4,4                 ;"<,=,>,<="
000470                  DFB       4,4,4                   ;"<>,>=,<=>"
000471                  DFB       8,3,2,6                 ;"^ AND OR MOD"
000472                  DFB       6,6,6,5                 ;"DIV / * -"
000473                  DFB       5,1                     ;"+ :"
000474 OPDSPT:          DW        DOREL
000475 RELNUM           EQU       1
000476                  DW        FPWRT
000477                  DW        ANDOP
000478                  DW        OROP
000479                  DW        TMERR
000480                  DW        TMERR
000481                  DW        FDIVT
000482                  DW        FMULTT
000483                  DW        FSUBT
000484                  DW        FADDT
000485 NUMDSP           EQU       *-OPDSPT
000486                  DW        LDOCOMP
000487                  DW        TMERR
000488                  DW        LAND
000489                  DW        LONGOR
000490                  DW        LREM
000491                  DW        LDIV
000492                  DW        LDIVT
000493                  DW        LMULT
000494                  DW        LSUB
000495                  DW        LADD
000496 *
000497 *     Here are the Bank Equates
000498 *
000499 SYSPAG           EQU       $1601
000500 VARNAMB          EQU       VARNAM+SYSPAG
000501 VARPNTB          EQU       VARPNT+SYSPAG
000502 DEFPNTB          EQU       DEFPNT+SYSPAG
000503 INDEXB           EQU       INDEX+SYSPAG
000504 INDEX1B          EQU       INDEX1+SYSPAG
000505 INDEX2B          EQU       INDEX2+SYSPAG
000506 TXTTABB          EQU       TXTTAB+SYSPAG
000507 VARTABB          EQU       VARTAB+SYSPAG
000508 ARYTABB          EQU       ARYTAB+SYSPAG
000509 STRENDB          EQU       STREND+SYSPAG
000510 FRETOPB          EQU       FRETOP+SYSPAG
000511 FRESPCB          EQU       FRESPC+SYSPAG
000512 MEMSIZB          EQU       MEMSIZ+SYSPAG
000513 HIMEMB           EQU       HIMEM+SYSPAG
000514 OLDTXTB          EQU       OLDTXT+SYSPAG
000515 DATPTRB          EQU       DATPTR+SYSPAG
000516 INPPTRB          EQU       INPPTR+SYSPAG
000517 DSCPNTB          EQU       DSCPNT+SYSPAG
000518 HIGHDSB          EQU       HIGHDS+SYSPAG
000519 NDXPTRB          EQU       NDXPTR+SYSPAG
000520 HIGHTRB          EQU       HIGHTR+SYSPAG
000521 LOWDSB           EQU       LOWDS+SYSPAG
000522 LOWTRB           EQU       LOWTR+SYSPAG
000523 PTR1B            EQU       PTR1+SYSPAG
000524 PTR2B            EQU       PTR2+SYSPAG
000525 PTR3B            EQU       PTR3+SYSPAG
000526 FACB             EQU       FAC+SYSPAG
000527 FACMOB           EQU       FACMO+SYSPAG
000528 ARGMOB           EQU       ARGMO+SYSPAG
000529 HEADERB          EQU       HEADER+SYSPAG
000530 DSCTMPB          EQU       DSCTMP+SYSPAG+1
000531 VARTXTB          EQU       VARTXT+SYSPAG
000532 TXTPTRB          EQU       TXTPTR+SYSPAG
000533 TMPPTRB          EQU       TMPPTR+SYSPAG
000534 SRCHPTB          EQU       SRCHPT+SYSPAG
000535 SMVARSB          EQU       SMVARS+SYSPAG
000536 ARYPNTB          EQU       ARYPNT+SYSPAG
000537 STRNG1B          EQU       STRNG1+SYSPAG
000538 STRNG2B          EQU       STRNG2+SYSPAG
000539 FORPNTB          EQU       FORPNT+SYSPAG
000540 DECCNTB          EQU       DECCNT+SYSPAG
000541 GRBTOPB          EQU       GRBTOP+SYSPAG
000542 DELTAB           EQU       DELTA+SYSPAG
000543 BANKPNTB         EQU       BANKPNT+SYSPAG
000544 PROCTABB         EQU       PROCTAB+SYSPAG
000545 INVTABB          EQU       INVTAB+SYSPAG
000546 POLYPTB          EQU       POLYPT+SYSPAG
000547 INVPNTB          EQU       INVPNT+SYSPAG
000548 ERRTOB           EQU       ERRTO+SYSPAG
```

```
000549 ERRPOSB         EQU       ERRPOS+SYSPAG
000550
000551 ; ##############################################################################################
000552 ; #   END OF FILE:  ZPG.EQUS.TEXT
000553 ; #   LINES      :  544
000554 ; #   CHARACTERS :  25960
000555 ; ##############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 555   CHARACTERS: 26514
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3RESVB.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:31 PM
|  Modified: Wednesday, December 31, 1997             4:37:08 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3RESVB.TEXT
000004 ; ###############################################################################################
000005
000006                  SBTL      "RESERVED WORDS & ERROR MESSAGES"
000007 TMERR:           LDX       #ERRTM
000008                  JMP       ERROR
000009 ; THE LIST OF RESERVED WORDS:
000010 RESLST:          DCI       'END'
000011 ENDTK            EQU       $80
000012                  DCI       'FOR'
000013 FORTK            EQU       ENDTK+1
000014                  DCI       'NEXT'
000015                  DCI       'INPUT'
000016 INPTKN           EQU       FORTK+2
000017                  DCI       'OUTPUT'
000018 OUTTKN           EQU       INPTKN+1
000019                  DCI       'DIM'
000020                  DCI       'READ'
000021                  DCI       'WRITE'
000022                  DCI       'OPEN'
000023 OPENTK           EQU       OUTTKN+4
000024                  DCI       'CLOSE'
000025                  DCI       'A '
000026                  DCI       'TEXT'
000027 TEXTTK           EQU       OPENTK+3
000028                  DCI       'A '
000029                  DCI       'BYE' ;EXPANSION'
000030                  DCI       'A '
000031                  DCI       'A '
000032                  DCI       'A '
000033                  DCI       'A '
000034                  DCI       'A '
000035                  DCI       'WINDOW'
000036                  DCI       'INVOKE'
000037 INVOKTK          EQU       TEXTTK+9
000038                  DCI       'PERFORM'
000039                  DCI       'A '
000040                  DCI       'A '
000041                  DCI       'FRE'
000042 FRETK            EQU       INVOKTK+4
000043                  DCI       'HPOS'
000044 HPOSTK           EQU       FRETK+1
000045                  DCI       'VPOS'
000046 VPOSTK           EQU       HPOSTK+1
000047                  DCI       'ERRLIN'
000048 ERRLINTK         EQU       VPOSTK+1
000049                  DCI       'ERR'
000050 ERRTK            EQU       ERRLINTK+1
000051                  DCI       'KBD'
000052 KBDTK            EQU       ERRTK+1
000053                  DCI       'EOF'
000054 EOFTK            EQU       KBDTK+1
000055                  DCI       'TIME$'
000056                  DCI       'DATE$'
000057                  DCI       'PREFIX$'
000058                  DCI       'EXFN.'
000059                  DCI       'EXFN%.'
000060 EXFNSTK          EQU       EOFTK+5
000061                  DCI       'OUTREC'
000062                  DCI       'INDENT'
000063                  DCI       'PROGPREFIX$' '
000064                  DCI       'A '
000065                  DCI       'A '
000066                  DCI       'A '
000067                  DCI       'A '
000068                  DCI       'A '
000069                  DCI       'A ' '
000070                  DCI       'POP'
000071 POPTKN           EQU       EXFNSTK+10
000072                  DCI       'HOME'
```

```
000073                  DO        DEBUG
000074                  DCI       'SHIT' ;'SHIT' For DEBUGGER, 'A' & 3 Spaces for NORMAL '
000075                  ELSE
000076                  DCI       'A   ' '
000077                  FIN
000078                  DCI       'SUB$('
000079                  DCI       'OFF'
000080                  DCI       'TRACE'
000081                  DCI       'NOTRACE'
000082                  DCI       'NORMAL'
000083                  DCI       'INVERSE'
000084                  DCI       'SCALE('
000085 SCALETK          EQU       POPTKN+9
000086                  DCI       'RESUME'
000087                  DCI       'A '
000088                  DCI       'LET'
000089                  DCI       'GOTO'
000090 GOTOTK           EQU       SCALETK+4
000091                  DCI       'IF'
000092 IFTOKN           EQU       GOTOTK+1
000093                  DCI       'RESTORE'
000094                  DCI       'SWAP'
000095                  DCI       'GOSUB'
000096 GOSUTK           EQU       IFTOKN+3
000097                  DCI       'RETURN'
000098                  DCI       'REM'
000099 REMTK            EQU       GOSUTK+2
000100                  DCI       'STOP'
000101                  DCI       'ON'
000102                  DCI       'A '
000103                  DCI       'LOAD'
000104 LDTKN            EQU       REMTK+4                   ;NOTE THAT ALL THE DISK COMMANDS
000105                  DCI       'SAVE' ;THAT DON'T WANT CRUNCHING ARE HERE'
000106                  DCI       'DELETE'
000107                  DCI       'RUN'
000108                  DCI       'RENAME'
000109 RENMTK           EQU       LDTKN+4
000110                  DCI       'LOCK'
000111                  DCI       'UNLOCK'
000112                  DCI       'CREATE'
000113                  DCI       'EXEC'
000114                  DCI       'CHAIN'
000115                  DCI       'A '
000116                  DCI       'A '
000117                  DCI       'A '
000118                  DCI       'CATALOG'
000119 CATATK           EQU       RENMTK+9
000120                  DCI       'A '
000121                  DCI       'A '
000122 DSKCOMS          EQU       CATATK+2
000123                  DCI       'DATA'
000124 DATATK           EQU       DSKCOMS+1
000125                  DCI       'IMAGE'
000126 IMAGETK          EQU       DATATK+1
000127                  DCI       'CAT'
000128 CATTK            EQU       IMAGETK+1
000129                  DCI       'DEF'
000130                  DCI       'A '
000131                  DCI       'PRINT'
000132 PRINTK           EQU       CATTK+3
000133                  DCI       'DEL'
000134                  DCI       'ELSE'
000135 ELSETK           EQU       PRINTK+2
000136                  DCI       'CONT'
000137                  DCI       'LIST'
000138                  DCI       'CLEAR'
000139                  DCI       'GET'
000140                  DCI       'NEW'
000141 SCRATK           EQU       ELSETK+5
000142 ; END OF COMMAND LIST.
000143 RESL2            DCI       'TAB('
000144 TABTK            EQU       $80
000145                  DCI       'TO'
000146 TOTK             EQU       TABTK+1
000147                  DCI       'SPC('
000148 SPCTK            EQU       TOTK+1
000149                  DCI       'USING'
000150 USINGTK          EQU       SPCTK+1
000151                  DCI       'THEN'
000152 THENTK           EQU       USINGTK+1
```

```
000153                DCI       'A '
000154 ATTKN          EQU       THENTK+1
000155                DCI       'MOD'
000156 MODTK          EQU       ATTKN+1
000157                DCI       'STEP'
000158 STEPTK         EQU       MODTK+1
000159                DCI       'AND'
000160 ANDTK          EQU       STEPTK+1
000161                DCI       'OR'
000162 ORTK           EQU       ANDTK+1
000163                DCI       'EXTENSION'
000164 EXTKN          EQU       ORTK+1
000165                DCI       'DIV'
000166 DIVTK          EQU       EXTKN+1
000167                DCI       'A '
000168                DCI       'FN'
000169 FNTK           EQU       DIVTK+2
000170                DCI       'NOT'
000171 NOTTK          EQU       FNTK+1
000172                DCI       'A '
000173                DCI       'A '
000174                DCI       'A '
000175                DCI       'A '
000176                DCI       'A '
000177                DCI       'A '
000178                DCI       'A '
000179                DCI       'A '
000180                DCI       'A '
000181                DCI       'A '
000182                DCI       'A '
000183                DCI       'A '
000184                DCI       'A '
000185                DCI       'AS'
000186 ASTKN          EQU       NOTTK+14
000187                DCI       'SGN('
000188 ONEFUN         EQU       ASTKN+1
000189                DCI       'INT('
000190                DCI       'ABS('
000191                DCI       'A '
000192                DCI       'TYP('
000193                DCI       'REC('
000194                DCI       'A '
000195                DCI       'A '
000196                DCI       'A '
000197                DCI       'A '
000198                DCI       'A '
000199                DCI       'A '
000200                DCI       'A '
000201                DCI       'A '
000202                DCI       'A '
000203                DCI       'A '
000204                DCI       'PDL('
000205                DCI       'BUTTON('
000206                DCI       'SQR('
000207                DCI       'RND('
000208                DCI       'LOG('
000209                DCI       'EXP('
000210                DCI       'COS('
000211                DCI       'SIN('
000212                DCI       'TAN('
000213                DCI       'ATN('
000214                DCI       'A '
000215                DCI       'A '
000216                DCI       'A '
000217                DCI       'A '
000218                DCI       'A '
000219                DCI       'A '
000220                DCI       'A '
000221                DCI       'A '
000222                DCI       'A '
000223                DCI       'A '
000224                DCI       'A '
000225                DCI       'A '
000226                DCI       'STR$('
000227 STRTK          EQU       ONEFUN+38
000228                DCI       'HEX$('
000229 HEXTK          EQU       STRTK+1
000230                DCI       'CHR$('
000231 CHRTK          EQU       HEXTK+1
000232                DCI       'LEN('
```

```
000233 LENTK           EQU     CHRTK+1
000234                 DCI     'VAL('
000235                 DCI     'ASC('
000236                 DCI     'TEN('
000237                 DCI     'A '
000238 DEXPTK          EQU     LENTK+4
000239                 DCI     'A '
000240 DLOGTK          EQU     DEXPTK+1
000241                 DCI     'CONV('
000242 CONVTK          EQU     DLOGTK+1
000243                 DCI     'CONV&('
000244                 DCI     'CONV$('
000245                 DCI     'CONV%('
000246 LASNUM          EQU     CONVTK+3                ;LAST NORMAL FUNCTION
000247                 DCI     'LEFT$('
000248                 DCI     'RIGHT$('
000249                 DCI     'MID$('
000250                 DCI     'INSTR('
000251 INSTRTK         EQU     LASNUM+4
000252                 DFB     0                       ;END OF RESERVED WORD LIST.
000253 RELNOT          EQU     4
000254 OPTAB           DFB     '>','=','<'
000255 RELOPS          EQU     *-OPTAB+RELNOT
000256                 DFB     $5E,ANDTK,ORTK,MODTK
000257                 DFB     DIVTK,'/','*','-'
000258                 DFB     '+',0
000259 ENDOP           EQU     *-OPTAB+RELNOT
000260 NUMOPS          EQU     *-OPTAB
000261 ERRTAB:         DCI     'NEXT WITHOUT FOR'
000262 ERRNF           EQU     1
000263                 DCI     'SYNTAX'
000264 ERRSN           EQU     ERRNF+1
000265                 DCI     'RETURN WITHOUT GOSUB'
000266 ERRRG           EQU     ERRSN+1
000267                 DCI     'OUT OF DATA'
000268 ERROD           EQU     ERRRG+1
000269                 DCI     'ILLEGAL QUANTITY'
000270 ERRFC           EQU     ERROD+1
000271                 DCI     'OVERFLOW'
000272 ERROV           EQU     ERRFC+1
000273                 DCI     'OUT OF MEMORY'
000274 ERROM           EQU     ERROV+1
000275                 DCI     !UNDEF'D                STATEMENT!
000276 ERRUS           EQU     ERROM+1
000277                 DCI     'BAD SUBSCRIPT'
000278 ERRBS           EQU     ERRUS+1
000279                 DCI     'RANGE'
000280 ERRNG           EQU     ERRBS+1
000281                 DCI     'INVOKE'
000282 ERRIN           EQU     ERRNG+1
000283                 DCI     'STACK OVERFLOW'
000284 ERRSK           EQU     ERRIN+1
000285                 DCI     !REDIM'D                ARRAY!
000286 ERRDD           EQU     ERRSK+1
000287                 DCI     'DIVISION BY ZERO'
000288 ERRDV0          EQU     ERRDD+1
000289                 DCI     'ILLEGAL DIRECT'
000290 ERRID           EQU     ERRDV0+1
000291                 DCI     'TYPE MISMATCH'
000292 ERRTM           EQU     ERRID+1
000293                 DCI     'STRING TOO LONG'
000294 ERRLS           EQU     ERRTM+1
000295                 DCI     'FORMULA TOO COMPLEX'
000296 ERRST           EQU     ERRLS+1
000297                 DCI     !CAN'T                  CONTINUE!
000298 ERRCN           EQU     ERRST+1
000299                 DCI     !UNDEF'D                FUNCTION!
000300 ERRUF           EQU     ERRCN+1
000301                 DCI     'VARIABLE'
000302 ERRVA           EQU     ERRUF+1
000303                 DCI     'SOS CALL'
000304 SSSSSS          EQU     ERRVA+1
000305                 DCI     'FILES BUSY'
000306 ERRFB           EQU     SSSSSS+1
000307                 DCI     'NOT SOS'
000308 ERRNS           EQU     ERRFB+1
000309                 DCI     'I/O'
000310 ERRIO           EQU     ERRNS+1
000311                 DCI     'FILE TOO LARGE' '
000312 ERRCR           EQU     ERRIO+1
```

```
000313                DCI       'WRITE PROTECT'
000314 ERRWP          EQU       ERRCR+1
000315                DCI       'DISK SWITCHED'
000316 ERRDS          EQU       ERRWP+1
000317                DCI       'BAD PATH'
000318 ERRBP          EQU       ERRDS+1
000319                DCI       'FILE NOT FOUND'
000320 ERRFN          EQU       ERRBP+1
000321                DCI       'PATH NOT FOUND'
000322 ERRPN          EQU       ERRFN+1
000323                DCI       'VOLUME NOT FOUND'
000324 ERRVN          EQU       ERRPN+1
000325                DCI       'DUPLICATE FILE'
000326 ERRDF          EQU       ERRVN+1
000327                DCI       'DISK FULL'
000328 ERRFU          EQU       ERRDF+1
000329                DCI       'FILE LOCKED'
000330 ERRFL          EQU       ERRFU+1
000331                DCI       'FILE NOT OPEN'
000332 ERRNO          EQU       ERRFL+1
000333                DCI       'DEVICE DISCONNECTED'
000334 ERRDO          EQU       ERRNO+1
000335                DCI       'RESOURCE UNAVAILABLE'
000336 ERRDU          EQU       ERRDO+1
000337                DCI       'DIRECTORY FULL'
000338 ERRFD          EQU       ERRDU+1
000339                DCI       'DUPLICATE VOLUME'
000340 ERRDV          EQU       ERRFD+1
000341 ERR:           ASC       ' ERROR' ;NEEDED FOR ALL ERROR MESSAGES'
000342                DFB       7,0
000343 RTMSG          DFB       13,10,10                ;CR, LF, LF
000344                ASC       'Please Press SPACE BAR'
000345                DFB       0
000346 INTXT:         ASC       ' IN '
000347                DFB       0
000348 BRKTXT         EQU       *
000349                DFB       13,10,10,15             ;CR, LF, LF, Screen On
000350                ASC       'PROGRAM INTERRUPTED'
000351                DFB       7,0
000352 ;   SOS ERROR ==> BASIC ERR #.
000353 ;   NOTE THAT ERR # MUST BE IN ASCENDING ORDER.
000354 ERRTBL         DFB       $10,ERRFN               ;FILE NOT FOUND
000355                DFB       $25,ERRDU               ;RESOURCE UNAVAILABLE
000356                DFB       $27,ERRIO               ;I/O ERROR
000357                DFB       $28,ERRDO               ;DEVICE DISCONNECTED
000358                DFB       $2B,ERRWP               ;WRITE PROTECT
000359                DFB       $2E,ERRDS               ;DISK SWITCHED
000360                DFB       $40,ERRBP               ;BAD PATH
000361                DFB       $43,ERRNO               ;FILE NOT OPEN
000362                DFB       $44,ERRPN               ;PATH NOT FOUND
000363                DFB       $45,ERRVN               ;VOLUME NOT FOUND
000364                DFB       $46,ERRFN               ;FILE NOT FOUND
000365                DFB       $47,ERRDF               ;DUPLICATE FILE
000366                DFB       $48,ERRFU               ;DISK FULL
000367                DFB       $49,ERRFD               ;DIRECTORY FULL
000368                DFB       $4D,ERRCR               ;FILE TOO LARGE
000369                DFB       $4E,ERRFL               ;FILE LOCKED
000370                DFB       $50,ERRFB               ;FILES BUSY
000371                DFB       $51,ERRNS               ;NOT SOS
000372                DFB       $52,ERRNS               ;NOT SOS (APPLE ][ PASCAL)
000373                DFB       $54,ERROM               ;OUT OF MEMORY
000374                DFB       $57,ERRDV               ;DUPLICATE VOLUME
000375                DFB       $58,ERRTM               ;TYPE MISMATCH
000376                DFB       $FF                     ;END OF TABLE.
000377                SKP       9
000378 ERRTABB        EQU       0                       ;ERRTAB BANK #
000379 RESLSTB        EQU       ERRTABB                 ;RESERVED VAR. LIST BANK.
000380 NUMSTRB        EQU       0                       ;NUMSTR BANK #.
000381 CON1MB         EQU       0
000382 FHALFB         EQU       0
000383 SQR0B          EQU       0                       ;SQR0.5
000384 TEN.CB         EQU       0
000385 INTXTB         EQU       0
000386 N.MILB         EQU       0
000387 TEMPF3B        EQU       0
000388 RNDXB          EQU       0
000389
000390 ; ###############################################################################
000391 ; #   END OF FILE:  B3RESVB.TEXT
000392 ; #   LINES      :  383
```

```
000393 ; #   CHARACTERS :  14032
000394 ; ##############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 394   CHARACTERS: 14584
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "INITIAL.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:35 PM
|  Modified: Wednesday, December 31, 1997             4:37:13 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  INITIAL.TEXT
000004 ; ###############################################################################################
000005
000006                  SBTL      "SYSTEM INITIALIZATION CODE."
000007 *
000008 *  Note: after startup and system initialization is complete, the
000009 *        following code area may be reassigned as buffer space since
000010 *        the code will not be used again until a reboot.
000011 *
000012 INITAT           EQU       *
000013 CHSGET           INC       TXTPTR                    ;THIS ROUTINE MOVED TO ZERO PAGE FOR FAST
000014                  BNE       CHSGOT                    ;EXECUTION SPEED
000015                  INC       TXTPTR+1
000016 CHSGOT           LDA       60000
000017                  CMP       #':'                      ;THIS TEST IS USUALLY TRUE(WE HOPE)
000018                  BCS       CHSRT                     ;IF SO, DONE!
000019                  CMP       #'                        '
000020                  BEQ       CHSGET
000021                  SEC                                 ;SET CARRY IF NUMERIC
000022                  SBC       #'0'
000023                  SEC
000024                  SBC       #$100-'0'
000025 CHSRT            RTS
000026                  DFB       0,0,0,0                   ;RANDOM NUMBER SEED.
000027                  DFB       $10,$D6,$3A,$F1
000028 INIT:            EQU       *                         ;<---------------------------------------
                                                                        BASIC COLD START HERE
000029                  LDA       $FFEF                     ;GET CURRENT BANK POINTER
000030                  STA       KBDBNK
000031                  STA       BASICBNK
000032                  STA       DCNTBNK
000033 ;   SOS CALL - REQUEST SEGMENT to disallow $2000-$21FF [0] since that
000034                  BRK                                 ;  area can't be virtually addressed
000035                  DFB       MREQ
000036                  DW        SEGTAB4
000037                  PHA                                 ;Put 0 at top of stack so FNDFOR will stop
000038                  LDA       #SELFLG
000039                  CMP       0                         ;Selector puts $EE in addr 0, 1
000040                  BNE       NOTSEL
000041                  CMP       1
000042                  BNE       NOTSEL
000043                  LDY       #0                        ;SELECTOR puts extended ptr to Program
000044                  LDA       (2),Y                     ;  PATHNAME in locations 2, 3
000045                  TAY                                 ;Length of path as index
000046                  LDA       (2),Y                     ;Move path to PROGPATH buffer
000047                  STA       PROGPATH,Y
000048                  DEY
000049                  BPL       *-6                       ;( Go back to LDA (2),Y )
000050                  STY       SELECTOR                  ;Set flag for SELECTOR
000051 NOTSEL           LDX       #0                        ;This loop initializes the volatile
000052                  TXA                                 ;  Zero Page locations
000053                  INX
000054                  STA       0,X
000055                  STA       SYSPAG-1,X
000056                  INX
000057                  BNE       *-6
000058                  JSR       TRYSEG                    ;GET THE MEMORY FOR THE USER.
000059                  LDX       #INIT-INITAT
000060 MOVCHG:          LDA       INITAT-1,X                ;This loop is to move the CHSGET
000061                  STA       CHRGET-1,X                ;  code between INIT & INITAT to zero
000062                  LDA       #0                        ;  page (where it will be CHRGET).
000063                  STA       SYSPAG+CHRGET-1,X
000064                  DEX                                 ;LOOP TIL DONE
000065                  BNE       MOVCHG
000066                  STX       TRMPOS                    ;TERMINAL POSITION
000067                  STX       TRFLAG                    ;TERMINAL FLAG
000068                  STX       INFLNO                    ;INPUT FILE #
000069                  LDY       MEMSIZ
000070                  LDA       MEMSIZ+1                  ;THIS IS THE SIZE OF MEMORY.
000071                  LDX       MEMSIZB
```

```
000072                  STY       FRETOP
000073                  STA       FRETOP+1                ;TOP OF STRINGS TOO.
000074                  STX       FRETOPB
000075                  LDX       RAMLOC
000076                  LDY       RAMLOC+1                ;LOAD THE ADDRESS
000077                  LDA       RAMLOCB                 ;  (AND BANK)
000078                  STX       TXTTAB                  ;  OF LOW END OF USER RAM SEGMENT
000079                  STY       TXTTAB+1                ;  & SAVE IT IN TXTTAB
000080                  STA       TXTTABB
000081                  LDA       $FFD9                   ;RANDOM COUNTER HERE.
000082                  AND       #$7F
000083                  STA       RNDX+4
000084                  LDA       $FFE9                   ;ANOTHER COUNTER.
000085                  STA       RNDX+5                  ;SEED RANDOM # GENERATOR.
000086                  JSR       P1INIT                  ;FIRST PART OF INIT
000087                  JSR       INITCNS                 ;INIT THE CONSOLE.
000088                  JSR       SCRTCH                  ;SET UP EVERYTHING ELSE.
000089                  JSR       FILSOS                  ;Put SOS prefix in SOSPATH
000090                  LDA       #>SWCHGO
000091                  STA       DISPATCH+1
000092                  LDA       #<SWCHGO
000093                  STA       DISPATCH+2
000094                  LDA       SELECTOR
000095                  BEQ       BOOTRUN
000096 *
000097 *  This entry routine if Program Selector
000098 *
000099                  LDA       PROGPATH                ;Get the length of the program path
000100                  beq       nopgm
000101                  LDA       #>PROGPATH              ;Set up pathname pointer to point
000102                  STA       HELF+1                  ;  to length byte of PROGPATH
000103                  LDA       #<PROGPATH
000104                  STA       HELF+2
000105                  BRK                               ;Here for the SELECTOR
000106                  DFB       SGFI
000107                  DW        HELF
000108                  BEQ       SELLO                   ;Go & RUN the Designated Program
000109 nopgm1           EQU       *
000110                  LDX       #$FF
000111                  STX       CURLIN+1
000112                  JMP       SERROR                  ;If file not found, err msg and return
000113 SELLO            STA       LINNUM                  ;Now RUN Selected Program
000114                  STA       LINNUM+1                ;  starting at line 0
000115                  STA       TXTPTRB
000116                  DEC       CURLIN+1
000117                  LDA       #>PROGPATH+1            ;POINT AT NAME
000118                  STA       TXTPTR
000119                  LDA       #<PROGPATH+1
000120                  STA       TXTPTR+1
000121                  JSR       CHRGOT                  ;Set STATUS on first character
000122                  JMP       RUN
000123                  DO        RUNTIME
000124 nopgm            LDA       #SEFNF
000125                  BNE       nopgm1
000126                  ELSE
000127 nopgm            EQU       *
000128                  FIN
000129 *
000130 *  This entry routine if NO Program Selector or if NO program
000131 *    is specified in the P/S Development environment.
000132 *
000133 BOOTRUN          EQU       *
000134                  JSR       COPYSOS                 ;Put SOS prefix into PROGPATH
000135                  BRK
000136                  DFB       SGFI
000137                  DW        HELF
000138                  BEQ       *+9                     ;Go & RUN the "HELLO" Program
000139                  DO        RUNTIME
000140                  LDX       #$FF
000141                  STX       CURLIN+1
000142                  JMP       SERROR                  ;If RUNTIME, & no HELLO, back to COLD START
000143                  ELSE
000144                  LDX       CURLIN+1
000145                  STX       CURLIN+1
000146                  JMP       MAIN                    ;IGNORE THE FILE NOT FOUND ERROR, GO INTO BASIC
000147                  FIN
000148 *   NOW RUN "HELLO"
000149                  STA       LINNUM                  ;AT LINE 0
000150                  STA       LINNUM+1
000151                  STA       TXTPTRB
```

```
000152                  DEC       CURLIN+1
000153                  LDA       #>HELCN                   ;POINT AT NAME
000154                  STA       TXTPTR
000155                  LDA       #<HELCN
000156                  STA       TXTPTR+1
000157                  JSR       CHRGOT                    ;SET STATUS ON 'H'
000158                  JMP       RUN
000159 *
000160 *   From here on, the code MUST be preserved!
000161 *
000162 *=========================================================================
000163 * HERE IS ROUTINE TO INITIALIZE THE CONSOLE, STUFF
000164 *
000165 INITCNS:         BRK
000166                  DFB       SOPN
000167                  DW        OPNCNST
000168                  BNE       FUK2                      ;ERROR...
000169                  LDA       CONSRFN                   ;REF NUM FOR CONSOLE
000170                  STA       SCHRTB+1
000171                  STA       SLINTB+1
000172                  STA       SINIT+1
000173                  STA       ISNLTB+1
000174                  STA       RDCTC+1                   ;READ FOR CNTROL-C.
000175                  BRK                                 ;GET DEFAULTS ON .CONS
000176                  DFB       SDGDN                     ;SOS CALL - GET DEV NUM
000177                  DW        DGCN
000178                  BNE       FUK2
000179                  LDA       OTDN
000180                  STA       INDN
000181                  STA       INDN2
000182                  STA       INDN3
000183                  STA       INDN4
000184                  STA       INDN5
000185                  STA       INDN6
000186                  BRK                                 ;SOS CALL - D-CONTROL
000187                  DFB       SDCNT                     ;RESET DEVICE
000188                  DW        DINIT
000189                  BNE       FUK2
000190                  BRK
000191                  DFB       SDCNT                     ;SET CNTRL-C SNIFFING ON.
000192                  DW        DCNTR
000193                  BNE       FUK2
000194                  BRK                                 ;SET NEWLINE=TRUE, ON CR
000195                  DFB       SNWL
000196                  DW        ISNLTB
000197                  BNE       FUK2
000198                  BRK                                 ;NOW INITIALIZE OUTPUT
000199                  DFB       SWRT
000200                  DW        SINIT
000201                  BNE       FUK2
000202                  RTS
000203 FUK2            JMP       SERROR
000204 * CHARS TO INIT THE VIDEO--
000205 SICHRS:         DFB       $10,2,$15,$D
000206                  DFB       1                         ;RESET VIEWPORT.
000207                  DFB       $1C                       ;CLEAR SCREEN
000208                  DFB       6                         ;CURSOR OFF
000209                  DFB       7
000210                  DFB       $0D                       ;CARRIAGE RETURN
000211                  DO        RUNTIME
000212                  ASC       '    ' ;4 spaces'
000213                  ELSE
000214                  ASC       '       ' ;7 spaces '
000215                  FIN
000216                  ASC       'Apple Business BASIC ' '
000217                  DO        RUNTIME
000218                  ASC       'Run-Time ' '
000219                  FIN
000220                  ASC       'v1.3' '
000221                  DO        DEBUG
000222                  ASC       'D' ;Put in the 'D' if Debugger Version '
000223                  FIN
000224                  ASC       ' - Copyright Apple Computer, 1980-83'
000225                  DO        RUNTIME
000226                  ELSE
000227                  ASC       '       ' ;7 spaces '
000228                  FIN
000229                  DFB       0,0,0
000230                  DFB       $0D,$0A,$0A               ;CR,LF.
000231 SICLEN          EQU       *-SICHRS
```

```
000232                  DFB       $DA,$19,$83
000233 P1INIT:          LDX       #255                ;MAKE IT LOOK DIRECT IN CASE OF
000234                  STX       CURLIN+1            ;ERROR MESSAGE.
000235                  STX       FILNO
000236                  STX       FILNO+1
000237                  LDA       #$80
000238                  STA       OLDTXTB
000239                  LDA       #$4C                ;A JMP opcode to set up the jumps to
000240                  STA       DISPATCH            ; DISPATCH and JMPER.  (In both cases,
000241                  STA       JMPER               ;   bytes +1 & +2 are filled w/address)
000242                  LDX       #0
000243                  STX       LASTPT+1
000244                  STX       ERRFLG              ;NO ERROR OR ON KBD
000245                  LDA       #STRSIZ
000246                  STA       FOUR6
000247                  LDX       #TEMPST
000248                  STX       TEMPPT              ;SET UP STRING TEMPORARIES.
000249                  LDA       #80
000250                  STA       OUTREC              ;DEFAULT RIGHT HAND MARGIN FOR LIST.
000251                  LDA       #2
000252                  STA       INDENT              ;DEFAULT # OF SPACES TO INDENT FOR-NEXT LISTINGS.
000253                  LDA       #0                  ;Expand to all possible memory
000254                  JSR       EXPAND
000255                  LDY       #0
000256                  STY       RNFLG               ;RUN FLAG (0 until a pgm runs)
000257                  TYA
000258                  STY       ERRFLG              ; MISC INITIALIZATION
000259                  DEY                           ;STORE $FF
000260                  STY       FILNO               ;  IN OUTPUT FILENUM
000261                  STY       FILNO+1             ;  & OUTPUT#
000262                  LDY       #FCBLEN*10          ;CLEAR OUT FILE FCBS
000263                  STA       FCB-1,Y
000264                  DEY
000265                  BNE       *-4
000266                  STA       (TXTTAB),Y          ;SET UP TEXT TABLE.
000267                  INY
000268                  STA       (TXTTAB),Y
000269                  INY                           ;TXTTAB ALWAYS STARTS ON PAGE BOUNDARY.
000270                  STA       (TXTTAB),Y
000271                  LDA       TXTTAB
000272                  CLC
000273                  ADC       #3
000274                  STA       ARYTAB
000275                  LDA       TXTTAB+1
000276                  ADC       #0
000277                  LDY       TXTTABB
000278                  JSR       FIXADC
000279                  STA       ARYTAB+1
000280                  TYA
000281                  ADC       #0
000282                  STA       ARYTABB
000283                  RTS
000284                  SBTL      "GENERAL STORAGE MANAGEMENT ROUTINES."
000285 ; Find a FOR entry on the Stack via VARPNT.
000286 FORSIZ           EQU       $14
000287 FNDFOR:          TSX                           ;Load X register with Stack Pointer
000288                  INX
000289                  INX
000290                  INX
000291                  INX                           ;IGNORE ADR(NEWSTT) AND RTS ADDR.
000292 FFLOOP:          LDA       257,X               ;GET STACK ENTRY.
000293                  CMP       #FORTK              ;Is it a FOR token?
000294                  BNE       FFRTS               ;No, no FOR loops with this Pntr.
000295                  LDA       FORPNT+1            ;GET HIGH.
000296                  ORA       FORPNT              ;IS IT ZERO?
000297                  BNE       CMPFOR
000298                  LDA       259,X               ;PNTR IS ZERO, SO ASSUME THIS ONE.
000299                  STA       FORPNT
000300                  LDA       260,X
000301                  STA       FORPNT+1
000302                  LDA       258,X               ;FAKE ARRAY AND INT FLAGS TOO.
000303                  STA       TEMPFOR
000304                  STA       INTFLG
000305                  ROR       A
000306                  ROR       ISARA
000307 CMPFOR:          LDA       FORPNT+1
000308                  CMP       260,X
000309                  BNE       ADDFRS              ;NOT THIS ONE.
000310                  LDA       FORPNT              ;GET D_WN.
000311                  CMP       259,X
```

```
000312                BNE       ADDFRS
000313                LDA       TEMPFOR
000314                CMP       258,X
000315                BEQ       FFRTS                   ;WE GOT IT! WE GOT IT!
000316 ADDFRS:        TXA
000317                CLC                               ;ADD 16 TO X.
000318                ADC       #FORSIZ
000319                TAX                               ;RESULT BACK INTO X.
000320                BNE       FFLOOP
000321 FFRTS:         RTS                               ;RETURN TO CALLER.
000322 ;
000323 * Here is the Block Transfer Up routine. (HIGHDS)<(LOWTR).(HIGHTR)M
000324 * ON ENTRY:
000325 *  HIGHDS is the Destination of the highest byte transferred.
000326 *  LOWTR is the lowest byte to be transferred.
000327 *  HIGHTR is the highest byte to be transferred.
000328 * ON EXIT:
000329 *  LOWTR is unchanged, HIGHTR is somewhere within a page of LOWTR,
000330 *  HIGHDS is lowest address transferred into.
000331 BLTU:          JSR       REASON
000332                STA       STREND                  ;THIS IS WHAT EVERYBODY CALLS
000333                STY       STREND+1
000334                STX       STRENDB
000335 BLTUC          LDY       HIGHTR                  ;SUBTRACT LOWTR FROM HIGHTR
000336                CPY       LOWTR                   ;TO FIND OUT HOW MUCH LEFT TO MOVE.
000337                LDA       HIGHTR+1                ;AND MOVE PAGES IF MORE THAN A PAGE.
000338                SBC       LOWTR+1
000339                LDY       HIGHTRB
000340                LDX       LOWTRB
000341                JSR       FIXAYX
000342                CPY       #0                      ;MORE THAN A BANK!  MOVE PAGES.
000343                BNE       MV256
000344                CMP       #1                      ;MORE THAN A PAGE?
000345                BCC       MVEND                   ;IF NOT, FINISH UP.
000346 MV256:         LDX       #HIGHDS                 ;NOW MOVE ONE PAGE OF DATA "UP" IN MEM
000347                JSR       SUB256
000348                LDX       #HIGHTR
000349                JSR       SUB256                  ;ADJUST DOWN A PAGE...
000350                LDY       #$FF                    ;MOVE IT NOW..
000351 BLK2:          LDA       (HIGHTR),Y
000352                STA       (HIGHDS),Y
000353                DEY
000354                BNE       BLK2
000355                LDA       (HIGHTR),Y              ;MOVE ONE MORE BYTE
000356                STA       (HIGHDS),Y
000357                JMP       BLTUC                   ;AND LOOP...
000358 MVEND:         LDA       HIGHTR                  ;CARRY IS SET
000359                SEC
000360                SBC       LOWTR
000361                BEQ       MVDONE                  ;ALL DONE, HOW  CONVENIENT
000362                PHA                               ;SAVE DIFFERENCE (HIGHTR-LOWTR) (IS #BYTES TO MOVE  MOD 256)
000363                LDY       HIGHDSB                 ;ADJUST HIGHDS TO FINAL MOVE LOCATION
000364                STA       HIGHDSB                 ;CHEAP-SHIT TEMPORARY
000365                LDA       HIGHDS
000366                SEC
000367                SBC       HIGHDSB                 ;HIGHDS-DIFFERENCE(HIGHTR-LOWTR)
000368                STA       HIGHDS
000369                LDA       HIGHDS+1
000370                SBC       #0
000371                CPY       #$80                    ;ARE WE MOVING IN THE STACK?
000372                BCC       *+5
000373                JSR       FIXSB2
000374                STA       HIGHDS+1
000375                STY       HIGHDSB
000376                PLA
000377                TAY                               ;GET INDEX OF # BYTES TO MOVE
000378                DEY
000379                LDA       (LOWTR),Y
000380                STA       (HIGHDS),Y
000381                DEY
000382                CPY       #$FF
000383                BNE       *-7
000384 MVDONE:        RTS
000385 SUB256:        LDA       1,X                     ;LOWER PTR BY ONE PAGE
000386                LDY       SYSPAG,X                ;GET BANK
000387                SEC
000388                SBC       #1
000389                JSR       FIXSBC
000390                STA       1,X
000391                TYA                               ;THE !#$%&'() 6502 DOESN'T HAVE STY ABS,X
```

```
000392                  STA       SYSPAG,X
000393                  RTS
000394 * MOVE MEMORY DOWN ROUTINE: MOVE (LOWTR)<(INDEX1).(STREND)
000395 * PRESERVES LOWTR. ADJUSTS ALL VARIABLE TABLE POINTERS, DOS BUFFERS,
000396 * ETC.
000397 MVDWN:           LDA       LOWTR                     ;SAVE LOWTR ON THE STACK
000398                  PHA
000399                  LDA       LOWTR+1
000400                  PHA
000401                  LDA       LOWTRB
000402                  PHA
000403                  LDY       #0
000404 MVDWN0          LDA       INDEX1                    ;IF NO MEMORY TO MOVE,
000405                  CMP       STREND                    ;DON'T MOVE ANY!
000406                  BNE       MVDWN1
000407                  LDA       INDEX1+1
000408                  CMP       STREND+1
000409                  BNE       MVDWN1
000410                  LDA       INDEX1B
000411                  CMP       STRENDB
000412                  BEQ       MVDWN3
000413 MVDWN1          LDA       (INDEX),Y                 ;MOVE A BYTE
000414                  STA       (LOWTR),Y
000415                  INC       INDEX1                    ;NEXT BYTE TO MOVE
000416                  BNE       MVDWN4
000417                  LDX       INDEX1+1
000418                  INX                                 ;INC INDEX1+1
000419                  CPX       #MAXPG
000420                  BCC       *+7
000421                  LDX       #MINPG
000422                  INC       INDEX1B
000423                  STX       INDEX1+1
000424 MVDWN4          INC       LOWTR
000425                  BNE       MVDWN0
000426                  LDX       LOWTR+1                   ;INC LOWTR+1
000427                  INX
000428                  CPX       #MAXPG
000429                  BCC       *+7
000430                  LDX       #MINPG
000431                  INC       LOWTRB
000432                  STX       LOWTR+1
000433                  BNE       MVDWN0                    ;ALWAYS
000434 MVDWN3:         PLA                                 ;RESTORE LOWTR
000435                  STA       LOWTRB
000436                  PLA
000437                  STA       LOWTR+1
000438                  PLA
000439                  STA       LOWTR
000440 * HEY MAN, LIKE WOW, THE STUFF IS MOVED. (WHAT STUFF OCCIFER?)
000441                  LDX       #5                        ;FINISH THE MOVE OPERATION.
000442 MVDWN2          CLC                                 ;ADJUST THE POINTERS
000443                  LDA       VARTAB-1,X
000444                  ADC       DELTA
000445                  STA       VARTAB-1,X
000446                  LDA       VARTAB,X
000447                  ADC       DELTA+1
000448                  LDY       VARTABB-1,X
000449                  JSR       FIXADC
000450                  STA       VARTAB,X
000451                  TYA
000452                  ADC       DELTAB
000453                  STA       VARTABB-1,X
000454                  DEX
000455                  DEX
000456                  BPL       MVDWN2                    ;DO THE ZERO PAGE POINTERS FOR BASIC3
000457                  RTS
000458 * MOVE UP ROUTINE. (STREND+DELTA)<(LOWTR).(STREND)
000459 * PRESERVES LOWTR
000460 MVUP:           LDA       STREND+1                  ;THIS REALLY JUST DOES A 'JSR BLTU'
000461                  STA       HIGHTR+1                  ;BUT IT'S MORE CONVENIENT THIS WAY
000462                  LDA       STRENDB
000463                  STA       HIGHTRB
000464                  LDA       STREND
000465                  STA       HIGHTR                    ;HIGHEST LOC TO MOVE=HIGHTR
000466                  CLC
000467                  ADC       DELTA                     ;CALCULATE THE DESTINATION OF THE MOVE
000468                  STA       HIGHDS
000469                  LDA       STREND+1
000470                  ADC       DELTA+1
000471                  LDY       STRENDB
```

```
000472                  JSR       FIXADC
000473                  STA       HIGHDS+1
000474                  TYA
000475                  ADC       DELTAB
000476                  STA       HIGHDSB
000477                  TAX
000478                  LDY       HIGHDS+1                ;MUST SET UP A,Y REGS
000479                  LDA       HIGHDS                  ;FOR 'BLTU'
000480                  JSR       BLTU
000481                  LDX       #3                      ;FINISH THE MOVE. STREND WAS ADJUSTED
000482                  JMP       MVDWN2                  ;BY BLTU, SO LEAVE IT ALONE
000483 ; This routine is used to ascertain that a given number of locations
000484 ;   remain available for the Stack.  The Call is:
000485 ;         LDA  #Number of 2-byte entries needed.
000486 ;         JSR  GETSTK
000487 ; This routine must be called by any routine which puts an arbitrary
000488 ;   amount of stuff on the stack, i.e. a recursive routine like FRMEVL.
000489 ; It is also called by routines such as GOSUB and FOR which make
000490 ;   permanent entries on the stack.
000491 ; Routines which merely use and free up the guaranteed NUMLEV locations
000492 ;   need not call this.
000493 ; ON EXIT:
000494 ;    A and X have been modified.
000495 GETSTK:          ASL       A                       ;MULT A BY 2. NB, CLEARS C BIT.
000496                  ADC       #NUMLEV*2+3+13          ;MAKE SURE 2*NUMLEV+13 LOCS
000497 ;  (13 BECAUSE OF FBUFFR)
000498                  BCS       OSERR                   ;WILL REMAIN IN STACK.
000499                  STA       INDEX
000500                  TSX                               ;GET STACKED.
000501                  CPX       INDEX                   ;COMPARE.
000502                  BCC       OSERR                   ;IF STACK.LE.INDEX1, OM.
000503                  RTS
000504 ; Subroutine: REASON
000505 ; Purpose: Makes certain that a given address lies below FRETOP.
000506 ; On Entry: Y, A hold the address in question
000507 ;           X holds the bank of the address in question
000508 ; On Exit:  Y, A, X unchanged if address is valid
000509 ;           OUT OF MEMORY error if address is not valid
000510 REASON           CPX       FRETOPB                 ;Compare Banks
000511                  BCC       REARTS
000512                  BNE       TRYMOR
000513                  CPY       FRETOP+1
000514                  BCC       REARTS
000515                  BNE       TRYMOR                  ;GO GARB COLLECT.
000516                  CMP       FRETOP
000517                  BCC       REARTS
000518 TRYMOR:          PHA
000519                  TXA
000520                  PHA
000521                  LDX       #10+1                   ;IF TEMPF2 HAS ZERO IN BETWEEN.
000522                  TYA
000523 REASAV:          PHA
000524                  LDA       HIGHDS-1,X              ;SAVE HIGHDS ON STACK.
000525                  DEX
000526                  BPL       REASAV                  ;PUT 10 OF THEM ON STK.
000527                  LDA       LOWDSB
000528                  PHA
000529                  LDA       LOWTRB
000530                  PHA
000531                  LDA       HIGHDSB
000532                  PHA
000533                  LDA       HIGHTRB
000534                  PHA
000535                  JSR       GARBA2                  ;GO GARB COLLECT.
000536                  PLA
000537                  STA       HIGHTRB
000538                  PLA
000539                  STA       HIGHDSB
000540                  PLA
000541                  STA       LOWTRB
000542                  PLA
000543                  STA       LOWDSB
000544                  LDX       #$F5                    ;THIS WORKS CUZ IT'S PAGE ZERO.
000545 REASTO:          PLA
000546                  STA       HIGHDS+10+1,X           ;RESTORE AFTER GARB COLLECT.
000547                  INX
000548                  BMI       REASTO
000549                  PLA
000550                  TAY
000551                  PLA                               ;RESTORE A AND Y.
```

```
000552                TAX
000553                PLA
000554                CPX       FRETOPB
000555                BCC       REARTS
000556                BNE       OMERR
000557                CPY       FRETOP+1                ;COMPARE HIGHS
000558                BCC       REARTS
000559                BNE       OMERR                   ;HIGHER IS BAD.
000560                CMP       FRETOP                  ;AND THE LOWS.
000561                BCS       OMERR
000562 REARTS:         RTS
000563 OSERR           LDX       #ERRSK                  ;Stack Overflow
000564                BNE       ERROR
000565
000566 ; ###############################################################################################
000567 ; #   END OF FILE:  INITIAL.TEXT
000568 ; #   LINES      :  559
000569 ; #   CHARACTERS :  24958
000570 ; ###############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 570   CHARACTERS: 25510
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3MAINC.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:29 PM
|  Modified: Wednesday, December 31, 1997             4:37:06 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3MAINC.TEXT
000004 ; ###############################################################################################
000005
000006                  SBTL      "ERROR HANDLER, READY, TERMINAL INPUT, COMPACTIFY"
000007 OMERR:           LDX       #ERROM
000008 ERROR:           EQU       *
000009                  TXA                                 ;Put ERROR code into Acc
000010                  PHA                                 ;  (This is a Push X on stack)
000011                  JSR       SETSOS                    ;Reset the SOS prefix
000012                  LDA       RNFLG
000013                  BNE       ERRORR
000014                  LDA       #>PROGPATH+1              ;Get pointer to PROG NAME
000015                  LDY       #<PROGPATH+1
000016                  LDX       #ERRTABB
000017                  JSR       STROUTR                   ;OUTPUT IT.
000018 ERRORR           EQU       *
000019                  PLA                                 ; (This is a Pull X from Stack)
000020                  TAX                                 ;Restore ERROR Code into X
000021                  LDA       #0
000022                  STA       CMDFLG                    ;Make sure CMDFLG is 0
000023                  LDA       FILNO+1                   ;MAKE SURE WE OUTPUT TO THE PROPER PLACE
000024                  STA       FILNO                     ;IN CASE ERROR WAS DURING A PRINT# OP.
000025                  STX       ERRNUM
000026 FREALL           LDA       LASTPT                    ;FREE UP A LOOSE STRING MAYBE?
000027                  LDY       LASTPT+1
000028                  LDX       #0
000029                  JSR       FRETNOW                   ;FREE UP EACH TEMPORARY, AND IT'S STRING.
000030                  LDA       #>TEMPST
000031                  CMP       TEMPPT
000032                  BCC       FREALL
000033                  LDX       ERRNUM
000034                  LDY       CURLIN+1                  ;NO "ON ERR" TRAPPING IN IMM MODE
000035                  INY
000036                  BEQ       DNTTRAP
000037                  DEY
000038                  STY       ERRLIN+1
000039                  STY       OLDLIN+1
000040                  LDY       CURLIN
000041                  STY       ERRLIN
000042                  STY       OLDLIN
000043                  BIT       ERRFLG
000044                  BPL       DNTTRAP
000045                  JMP       HNDLERR
000046 DNTTRAP          LDA       #255
000047                  STA       FILNO
000048                  STA       FILNO+1
000049                  LDA       #0
000050                  JSR       EXPAND                    ;GIVE USER BACK ALL HIS MEMORY.
000051                  LDA       #15                       ;SCREEN ON.
000052                  JSR       OUTDO
000053                  JSR       CRDO                      ;OUTPUT CRLF.
000054                  JSR       OUTQST                    ;PRINT A QUESTION MARK
000055                  LDX       REMSTK                    ;KEEP STACK CLEAN
000056                  TXS
000057                  LDX       ERRNUM
000058                  LDA       #ERRTABB
000059                  STA       INDEXB
000060                  LDA       #<ERRTAB
000061                  STA       INDEX+1
000062                  LDA       #>ERRTAB
000063                  STA       INDEX
000064                  LDY       #0
000065 FINDHERR         DEX
000066                  BEQ       GOTHER
000067 CHECKHER         LDA       (INDEX),Y
000068                  INY
000069                  BNE       *+4
000070                  INC       INDEX+1                   ;NEVER CROSSES BANK BOUNDERY.
000071                  CMP       #$80
000072                  BCC       CHECKHER
```

```
000073                  BCS     FINDHERR
000074 GOTHER           LDA     (INDEX),Y
000075                  INY
000076                  BNE     *+4
000077                  INC     INDEX+1
000078                  PHA
000079                  JSR     OUTDO
000080                  PLA
000081                  CMP     #$80
000082                  BCC     GOTHER
000083                  LDA     #>ERR                   ;Get pointer to ERROR.
000084                  LDY     #<ERR
000085                  LDX     #TEMPST
000086                  STX     TEMPPT
000087                  LDX     #ERRTABB
000088 ERRFIN:          JSR     STROUTR                 ;OUTPUT IT.
000089                  LDY     CURLIN+1
000090                  INY                             ;WAS NUMBER 64000?
000091                  BEQ     *+5                     ;YES, DON'T TYPE LINE NUMBER.
000092                  JSR     INPRT
000093                  JSR     CRDO                    ;KICK ONE IN FOR FUN
000094                  LDX     INFLNO                  ;EXEC FILE GOING?
000095                  BEQ     READY                   ;NO
000096                  LDA     #SEEOF
000097                  JSR     EXCCLS                  ;CLOSE THE EXEC FILE.
000098 READY            EQU     *
000099                  JSR     SETSOS
000100                  DO      RUNTIME
000101                  LDA     ERRNUM                  ;Was it an Error or a finished pgm?
000102                  BEQ     READYGO                 ;  0= finished pgm
000103                  LDA     #>RTMSG                 ;Get pointer to RUN TIME continue msg
000104                  LDY     #<RTMSG
000105                  LDX     #ERRTABB
000106                  JSR     STROUTR                 ;OUTPUT IT.
000107 READY1           JSR     DOAGET                  ;Get a SPACE from user to acknowledge
000108                  LDA     KEYSAVE
000109                  CMP     #$20                    ;Was a SPACE entered?
000110                  BNE     READY1                  ;No, try again
000111 READYGO          JMP     COLD1                   ;Yes, jump to COLD START
000112                  ELSE
000113                  BRK
000114                  DFB     SFLS
000115                  DW      RFLUSH
000116                  FIN
000117 MAIN:            LDA     #$FF                    ;OUTPUT TO OUTPUT DEVICE
000118                  STA     CURLIN+1
000119                  LDA     #1
000120                  STA     RNFLG                   ;Set the already ran flag
000121                  JSR     SETSOS                  ;Just in case...
000122                  JSR     SPROGPFX
000123                  JSR     VPOS
000124                  LDA     CURX
000125                  STA     TRMPOS                  ;IF NOT AT THE BEGINNING OF A LINE,
000126                  BEQ     *+5                     ;PRINT A CARRIAGE RETURN
000127                  JSR     CRDO
000128                  LDA     KBDKEY
000129                  PHA
000130                  LDA     #0
000131                  STA     KBDKEY
000132                  BRK
000133                  DFB     SDCNT                   ;Reset .CONSOLE input
000134                  DW      DKBD
000135                  PLA
000136                  STA     KBDKEY
000137                  LDX     #$29                    ;PAREN FOR PROMPT
000138                  STX     KEYSTROK                ;NO CNTRL-C OR KBD HIT.
000139                  JSR     INLINB
000140                  STX     TXTPTR
000141                  STY     TXTPTR+1
000142                  STA     TXTPTRB
000143                  JSR     CHRGET
000144                  TAX                             ;SET ZERO FLAG BASED ON A TERMINATOR
000145                  BEQ     MAIN                    ;IF BLANK LINE, GET ANOTHER.
000146                  LDX     #255                    ;SET DIRECT LINE NUMBER.
000147                  STX     CURLIN+1
000148                  BCC     MAIN1                   ;IS A LINE NUMBER. NOT DIRECT.
000149                  STX     LOWTR+1
000150                  JSR     CRUNCH                  ;COMPACTIFY.
000151                  JMP     GONE                    ;EXECUTE IT.
000152 MAIN1:           JSR     LINGET                  ;READ LINE NUMBER INTO 'LINNUM'
```

```
000153                  LDA       #0                      ;TELL CRUNCHER LINE IS NOT DIRECT
000154                  STA       LOWTR+1
000155                  STA       OLDTXTB                 ;DISALLOW CONTINUING
000156                  JSR       CRUNCH
000157                  STY       COUNT                   ;RETAIN CHARACTER COUNT.
000158                  STY       BUF-3                   ;SAVE LENGTH OF LINE IN BUF
000159                  JSR       FNDLIN
000160                  BCC       NODEL                   ;NO MATCH, SO DON'T DELETE.
000161 * DELETE THE OLD LINE FROM THE PROGRAM AREA.
000162                  LDY       #00
000163                  LDA       #$7F                    ;TWO'S COMPLIMENT SINCE WE ARE MOVING
000164                  STA       DELTA+1                 ; DOWN
000165                  LDA       #$FF
000166                  STA       DELTAB
000167                  EOR       (LOWTR),Y
000168                  STA       DELTA                   ;CALCULATE DELTA FOR D.O.S.
000169                  INC       DELTA
000170                  LDA       (LOWTR),Y               ;GET LINE LENGTH
000171                  CLC
000172                  ADC       LOWTR                   ;CALCULATE POSITION OF NEXT LINE
000173                  STA       INDEX1
000174                  LDA       LOWTR+1                 ;BEGIN MOVE HERE.
000175                  ADC       #0
000176                  LDY       LOWTRB
000177                  JSR       FIXADC                  ;ADJUSTS PAGE, BANK COUNT SO PAGE BETWEEN 2,82.
000178                  STA       INDEX1+1
000179                  TYA
000180                  ADC       #0
000181                  STA       INDEX1B
000182                  JSR       MVDWN                   ;MOVES, DOES DELTA ON PTRS
000183 NODEL            LDA       BUF                     ;ANYTHING ON THIS LINE?
000184                  BEQ       FINI                    ;BRANCH IF NOT
000185                  LDA       #0                      ;SET DELTA
000186                  STA       DELTA+1
000187                  STA       DELTAB
000188                  LDA       COUNT
000189                  STA       DELTA
000190                  JSR       MVUP
000191                  LDA       LINNUM
000192                  LDY       LINNUM+1                ;POSITION THE BINARY LINE NUMBER
000193                  STA       BUF-2
000194                  STY       BUF-2+1                 ;IN FRONT OF BUF
000195                  LDY       COUNT
000196 STOLOP:          LDA       BUF-4,Y
000197                  DEY
000198                  STA       (LOWTR),Y
000199                  BNE       STOLOP
000200 FINI:            JSR       STXTPT                  ;CLEAN UP THIS CRAP
000201                  JSR       FLOAD
000202 ;AND SET TXTPTR TO TXTTAB-1.
000203                  JMP       MAIN                    ;YES, CHEAD HAS FINISHED.
000204 INLIN:           LDX       #0                      ;NO PROMPT CHARACTER
000205 INLINB           JSR       INPUTLIN
000206 GDBUFS:          LDA       #0                      ;PUT A ZERO AT THE END
000207                  STA       BUF,X
000208                  LDA       #0
000209                  STA       YSAVE                   ;BANK # SAVED HERE.
000210                  LDX       #>BUF-1
000211                  LDY       #<BUF-1                 ;POINT AT THE BEGINNING
000212                  RTS
000213 ;OF THE TEXT POINTER TO GET TO BUF
000214 GCRNCHED:        JMP       CRNCHED
000215 CRUNCH:          LDA       TXTPTR                  ;NEED A PLACE TO START
000216                  SEC
000217                  SBC       #>BUF
000218                  TAX
000219                  LDA       #3                      ;INITIALLY NOWHERE IN THE LINE
000220                  STA       BUFPTR                  ;AND NOT IN A DATA STATEMENT
000221                  STA       DORES                   ; BIT 7 OF DORES INDICATES WHETHER
                                                                  IN A DATA STATEMENT
000222                  STA       INTFLG
000223                  STA       BUF-1,X                 ;BECAUSE NO SPACE NEEDED AFTER LINE NUMBER
000224 CLOOP            LDY       #0                      ;CHECK NOW IF WE ARE AT A RESERVED WORD
000225                  STY       COUNT+1                 ;NOT AN ESCAPE TOKEN
000226                  LDA       #$80                    ;START TOKEN NUMBER COUNT
000227                  STA       COUNT
000228                  LDA       #>RESLST
000229                  STA       FAC                     ;(FAC) POINTS AT RESERVED WORD LIST
000230                  LDA       #<RESLST
000231                  STA       FAC+1
```

```
000232                LDA       #RESLSTB
000233                STA       FACB
000234                LDA       BUF,X                   ;GET THE NEXT CHAR OF INPUT
000235                BEQ       GCRNCHED                ;IF AN EOL, GO HOME!
000236                AND       #$7F
000237                CMP       #$3A                    ;COLON?
000238                BEQ       ONCRNC                  ;YES, TURN CRUNCHING BACK ON
000239                CMP       #$2C                    ;COMMA?
000240                BNE       CL1
000241                BIT       INTFLG
000242                BPL       *+6
000243                ROR       DORES
000244                BMI       CL1
000245                BIT       DORES                   ;IF A COMMA, AND A DISK COMMAND, RE-ALLOW CRUNCHING
000246                BVS       CL1                     ;NOT A DISK COM
000247 ONCRNC:        STA       DORES                   ;THIS WILL RE-ALLOW CRUNCHING
000248 CL1:           BIT       DORES                   ;CRUNCH THIS CHAR?
000249                BMI       STUFIT                  ;NO, STICK IT
000250                CMP       #$21                    ;IGNORE SPACES
000251                BCS       *+5
000252                JMP       NXCHR
000253                CMP       #'!'                    ;REM TOKEN?
000254                BCC       STUFIT
000255                BNE       SHORT1
000256                LDA       #REMTK
000257                BCS       STUFIT
000258 SHORT1         CMP       #'?'                    ;PRINT TOKEN?
000259                BCC       STUFIT
000260                BNE       NPRNT
000261                LDA       #PRINTK
000262                BCS       STUFIT
000263 GNMTCH         JMP       NOMTCH
000264 NPRNT          STX       TEMP                    ;SAVE FOR WHEN LOOPING BACK IN
000265 NPR2           LDX       TEMP                    ;LOOP BACK TO HERE
000266 NPR3           LDA       BUF,X
000267                BEQ       NSPC
000268                AND       #$7F
000269                CMP       #$21                    ;SPACES?
000270                BCS       NSPC                    ;NO
000271                INX                               ;YES, SKIP IT
000272                BNE       NPR3
000273 NSPC           CMP       #'A'+$20
000274                BCC       NSP2
000275                CMP       #'Z'+$21
000276                BCS       NSP2
000277                AND       #$DF                    ;KILL $20 BIT SO UPPER=LOWER CASE.
000278 NSP2           EOR       (FAC),Y
000279                INY
000280                INX
000281                ASL       A                       ;LEAST SIGN. 7 BITS MUST MATCH
000282                BNE       GNMTCH                  ;NAW, DAMN HIM!
000283                BCC       NPR3                    ;SURE DOES SO FAR (NOT END OF RESERVED WORD)
000284 * WE FOUND A RESERVED WORD!(I THINK- SEE IF IMBEDDED IN NON-ALPHA,
000285 * NON-DIGIT DELIMS)
000286                LDA       BUF,X                   ;CHECK CHAR AFTER THE WORD
000287                AND       #$7F
000288                JSR       CKSEP                   ;IS IT A SEPERATOR?
000289                BCC       YMTCH                   ;YUP, IT MATCHED
000290                DEY                               ;DID THE WORD END ON A SEPARATOR?
                                                                I.E., LOMEM: OR DEC(?
000291                LDA       (FAC),Y                 ;CHECK LAST CHAR IN WORD
000292                AND       #$7F                    ;WIPE HIGH BIT
000293                JSR       CKSEP
000294                BCC       YMTCH                   ;YES IT MATCHED
000295                LDA       COUNT                   ;WAS IT A FN TOKEN?
000296                CMP       #EXFNSTK-1
000297                BEQ       YMTCH
000298                CMP       #EXFNSTK
000299                BEQ       YMTCH
000300                CMP       #FNTK
000301                BNE       GNMTCH                  ;NO.
000302                BIT       COUNT+1                 ;MUST BE AN ESCAPE TOKEN
000303                BPL       GNMTCH
000304 YMTCH          TXA                               ;SAVE X-REG.
000305                PHA                               ;ON THE STACK
000306                LDX       TEMP                    ;WAS THE CHARACTER BEFORE IT A SEPERATOR?
000307                LDA       BUF-1,X
000308                JSR       CKSEP
000309                PLA                               ;RESTORE X-REG
000310                TAX
```

```
000311                BCS       GNMTCH
000312                DEX                           ;DON'T GO BEYOND RESERVED WORD IN PROGRAM
000313                LDA       COUNT+1             ;IS IT AN ESCAPE TOKEN?
000314                BMI       STUFIT              ;YES, STUFF THE $FF
000315                LDA       COUNT               ;NO, JUST STUFF A NORMAL TOKEN
000316 STUFIT         LDY       BUFPTR              ;GET INDEX WHERE TO PUT THIS BYTE
000317                STA       BUF-3,Y
000318                INY                           ;ADVANCE POINTER
000319                CMP       #$FF                ;ESCAPE TOKEN?
000320                BNE       NESC                ;  THERE IS NO ESCAPE!!!!!!
000321                LDA       COUNT               ;IF SO, STUFF BOTH BYTES
000322                STA       BUF-3,Y
000323                INY
000324                LDA       #0                  ;DON'T WANT TO MATCH REMTK OR STUFF
000325 NESC           STY       BUFPTR              ;SAVE BACK THE POINTER
000326 NXCHR          INX                           ;GET NEXT CHAR IN THE LINE
000327                CMP       #REMTK              ;DID WE STUFF A REMTK?
000328                BEQ       REMIT               ;IF SO, REM-ARKABLE
000329                CMP       #DATATK
000330                BEQ       ITDIR15
000331                CMP       #$22
000332                BEQ       DOQUOT
000333                CMP       #IMAGETK+2
000334                BCS       GCLOOP
000335                LDY       LOWTR+1             ;ARE WE IN IMMEDIATE MODE?
000336                INY
000337                BEQ       ITDIR               ;YES, DON'T CRUNCH
000338                CMP       #DSKCOMS+1          ;IF IN DEFERRED MODE...
000339                BCC       GCLOOP              ;CRUNCH NORMAL
000340 ITDIR          CMP       #OPENTK             ;OPEN DOES IT BACKWARDS.
000341                BNE       ITDIR1
000342                ROR       INTFLG              ;SET HIGH BIT
000343                BMI       GCLOOP              ;ALWAYS
000344 ITDIR1         CMP       #INVOKTK
000345                BEQ       ITDIR15
000346                CMP       #LDTKN
000347                BCC       GCLOOP
000348                CMP       #RENMTK             ;IS IT RENAME?
000349                BNE       ITDIR2              ;NO
000350 ITDIR15        SEC
000351                DFB       44
000352 ITDIR2         CMP       #DSKCOMS+1          ;DORES: BIT 7 ON- DONT CRUNCH UNTIL A :
000353                ROR       DORES               ;SET BIT 7 OF DORES
000354                SEC
000355                ROR       DORES               ;DORES: BIT 6 OFF-- DON'T CRUNCH UNTIL A COMMA, ELSE UNTIL ':'
000356 GCLOOP         JMP       CLOOP
000357 REMIT          LDA       #0                  ;A 'REM' ENDS AT THE END OF THE LINE ONLY
000358 DOQUOT         DEX                           ;START WITH CORRECT CHAR FROM INPUT LINE
000359                STA       ENDCHR              ;THIS IS WHAT THE UN-CRUNCHED AREA MAY END ON
000360                LDY       BUFPTR              ;GET WHERE TO STUFF CHARS
000361                DEY
000362 DOQ2           INY
000363                INX
000364                LDA       BUF,X
000365                STA       BUF-3,Y             ;MOVE CHAR
000366                STY       BUFPTR
000367                BEQ       CRNCHED             ;END OF THE LINE
000368                CMP       ENDCHR              ;END CHAR REACHED?
000369                BNE       DOQ2                ;NO, LOOP
000370                INY
000371                STY       BUFPTR              ;SAVE BACK
000372                INX
000373                JMP       GCLOOP
000374 * THIS WORD DIDN'T MATCH. TRY THE NEXT ONE
000375 NOMTCH         DEY
000376                BEQ       FNDNXT
000377                DEY
000378 FNDNXT         INY                           ;FIND THE NEXT RESERVED WORD
000379                LDA       (FAC),Y
000380                BPL       FNDNXT              ;WORD ENDS ON A NEGATIVE CHARACTER
000381                SEC
000382                TYA                           ;ADD LENGTH OF THIS WORD TO (FAC)
000383                ADC       FAC
000384                STA       FAC
000385                BCC       *+4
000386                INC       FAC+1               ;NEVER CROSSES BANK BOUNDARY.
000387                LDY       #0
000388                LDX       COUNT
000389                INX                           ;ADVANCE TOKEN COUNT
000390                CPX       #SCRATK+1           ;PAST THE STATEMENTS?
```

```
000391                  BNE       FND2
000392                  LDX       #$FF                    ;SET ESCAPE MODE
000393                  STX       COUNT+1
000394                  LDX       #$80
000395 FND2             STX       COUNT
000396                  LDA       (FAC),Y                 ;ARE WE AT THE END OF THE LIST?
000397                  BNE       GNPR2                   ;NO, KEEP GOING
000398                  LDX       TEMP                    ;MAKE SURE TO GET THE RIGHT CHAR
000399                  LDA       BUF,X                   ;YES, STUFF THIS CHARACTER
000400                  AND       #$7F
000401                  JMP       STUFIT
000402 * LINE IS FINISHED CRUNCHING
000403 CRNCHED          LDY       BUFPTR
000404                  STA       BUF-3,Y
000405                  STA       BUF-2,Y                 ;I DON'T KNOW WHY, BUT IT NEEDS THIS
000406                  LDA       #0
000407                  STA       TXTPTRB
000408                  LDA       #<BUF-1
000409                  STA       TXTPTR+1
000410                  LDA       #>BUF-1
000411                  STA       TXTPTR
000412                  INY
000413                  RTS
000414 GNPR2            JMP       NPR2
000415 CKSEP:           JSR       ISLETC                  ;IS IT A LETTER IN BETWEEN
000416                  BCS       CKRTS
000417                  CMP       #'9'+1                  ;IF NOT A DIGIT OR A SPECIAL, ITS A SEPERATOR.
000418                  BCS       CKRT1
000419                  CMP       #'0'                    ;A DIGIT?
000420                  BCS       CKRTS
000421                  CMP       #'.'
000422                  BEQ       CKRTS
000423 CKRT1            CLC
000424 CKRTS            RTS
000425 ; FNDLIN searches the program text for the line whose number is passed
000426 ;   in LINNUM.  There are only two possible returns:
000427 ; 1) Carry Set.
000428 ;    LOWTR points to the link byte in the line that was searched for.
000429 ; 2) Carry Clear.
000430 ;    Line not found.  LOWTR points to the first line in the program with
000431 ;    a line number greater than the one sought after.
000432 FNDLIN:          LDA       TXTTAB
000433                  LDX       TXTTAB+1                ;LOAD X,A WITH TXTTAB
000434                  LDY       TXTTABB                 ;Y WITH BANK #.
000435 FNDLNC0          STY       LOWTRB
000436 FNDLNC1          STX       LOWTR+1
000437 FNDLNC           STA       LOWTR
000438                  LDY       #0                      ;POINT TO LINK.
000439                  LDA       (LOWTR),Y               ;SEE IF LINK IS 0
000440                  BEQ       FLINRT
000441                  LDY       #2
000442                  TAX
000443                  LDA       LINNUM+1                ;COMP HIGH ORDERS OF LINE NUMBERS.
000444                  CMP       (LOWTR),Y
000445                  BCC       FLNRTS                  ;NO SUCH LINE NUMBER.
000446                  BNE       AFFRTS                  ;CHECK NEXT LINE.
000447                  LDA       LINNUM
000448                  DEY
000449                  CMP       (LOWTR),Y               ;COMPARE LOW ORDERS.
000450                  BCC       FLNRTS                  ;NO SUCH NUMBER.
000451                  BEQ       FLNRTS                  ;RETURN WITH CARRY SET.
000452 AFFRTS:          TXA
000453                  CLC
000454                  ADC       LOWTR                   ;COMPUTE NEXT RELATIVE LINE POSITION
000455                  BCC       FNDLNC                  ;BRANCH IF DONE
000456                  LDX       LOWTR+1                 ;INC LOWTR+1
000457                  INX
000458                  CPX       #MAXPG
000459                  BCC       FNDLNC1
000460                  PHA
000461                  TXA
000462                  SBC       #MAXPG-MINPG
000463                  TAX
000464                  PLA
000465                  INC       LOWTRB
000466                  BNE       FNDLNC1                 ;ALWAYS
000467                  BCS       FNDLNC                  ;ALWAYS BRANCHES.
000468 FLINRT:          CLC                               ;C MAY BE HIGH.
000469 FLNRTS:          RTS                               ;RETURN TO CALLER.
000470 ;
```

```
000471 ; The NEW command clears the program text as well as variable space.
000472 SCRATH:         BNE       FLNRTS                    ;MAKE SURE THERE IS A TERMINATOR.
000473 SCRTCH:         JSR       CLSALL                    ;CLOSE FILES BEFORE CLEARING FCB (P1INIT).
000474                 JSR       P1INIT                    ;AND CLEAN UP EVERYTHING
000475 RUNC:           JSR       STXTPT
000476                 LDA       #0                        ;SET ZERO FLAG
000477                 STA       CURLIN+1                  ;SO DOESN'T THINK IN IMMEDIATE MODE
000478 ; THIS CODE IS FOR THE CLEAR COMMAND.
000479 CLEAR:          BNE       STKRTS                    ;SYNTAX ERROR IF NO TERMINATOR.
000480 ; CLEARC IS SUBROUTINE WHICH INITIALIZES THE VARIABLE AND
000481 ; ARRAY SPACE BY RESETING ARYTAB (END OF SIMPLE VARIABLE)
000482 ; AND STREND (END OF ARRAY STORAGE). IT FALLS INTO
000483 ; 'STKINI' WHICH RESETS THE STACK.
000484 CLEARC:         JSR       CLSALL
000485 CLEARL          LDA       MEMSIZ
000486                 LDX       MEMSIZB
000487                 LDY       MEMSIZ+1                  ;FREE UP STRING SPACE.
000488                 STA       FRETOP
000489                 STY       FRETOP+1
000490                 STX       FRETOPB
000491                 LDA       ARYTAB
000492                 LDY       ARYTAB+1                  ;LIBERATE THE
000493                 LDX       ARYTABB
000494                 STA       VARTAB
000495                 STY       VARTAB+1                  ;VARIABLES AND
000496                 STX       VARTABB
000497                 STA       STREND
000498                 STY       STREND+1                  ;ARRAYS.
000499                 STX       STRENDB
000500                 LDA       #0
000501                 STA       KEYSAVE                   ;ZERO OUT KBD VARIABLE.
000502                 STA       ERRNUM                    ;ZERO OUT ERR VARIABLE.
000503                 STA       ERRLIN                    ;ZERO OUT ERRLIN VARIABLE.
000504                 STA       ERRLIN+1
000505                 STA       EOFSV
000506 FLOAD:          LDA       #0                        ;GET ALL AVAIL MEM BACK
000507                 JSR       EXPAND
000508                 JSR       RESTOR                    ;RESTORE DATA.
000509 ;
000510 ; Procedure: STKINI
000511 ; Function: Resets the stack pointer
000512 ; On Exit: GOSUB and FOR entries eliminated
000513 ;           String temporaries are freed up
000514 ;           SUBFLG is reset
000515 ;           CONTinuing is Prohibited
000516 ;           A dummy entry is left at the bottom of the stack so there
000517 ;             be a non-FOR entry at the bottom
000518 STKINI          PLA                                 ;SETUP RETURN ADDRESS.
000519                 TAY
000520                 PLA
000521                 LDX       #STKEND                   ;HAVE STACK POINT TO RETURN ADDRESS.
000522                 STX       REMSTK
000523                 TXS
000524                 PHA
000525                 TYA
000526                 PHA
000527                 LDA       #2
000528                 STA       VRBPT                     ;INITIALIZE VERB POINTER TO POINT PAST EOL ENTRY.
000529                 LDA       #0
000530                 STA       VRBSTK+1                  ;PUT EOL PRECIDENCE ON THE STACK;
000531                 STA       NOUNPT                    ;FORMULA EVALUATOR STACK NOW RESET.
000532                 STA       SUBFLG                    ;ALLOW SUBSCRIPTS.
000533 STKRTS:         RTS
000534 STXTPT:         SEC
000535                 LDA       TXTTAB
000536                 SBC       #1
000537                 STA       TXTPTR
000538                 LDA       TXTTAB+1
000539                 SBC       #0
000540                 LDY       TXTTABB
000541                 JSR       FIXSBC
000542                 STA       TXTPTR+1                  ;SETUP TEXT POINTER.
000543                 TYA
000544                 SBC       #0
000545                 STA       TXTPTRB
000546                 LDY       #0
000547                 TYA
000548                 STA       (TXTPTR),Y                ;STUFF A ZERO AT BEGINNING OF PROGRAM.
000549 CLEARONS        JSR       OFFKBD
000550                 LDY       #EOFSIZ
```

```
000551                  LDA       #0
000552                  STA       KEYSTROK
000553 CLEOFS:          STA       EOFPTRS-1,Y
000554                  DEY
000555                  BNE       CLEOFS
000556                  STY       ERRFLG
000557                  STY       ERRPOSB                   ;NO ERRORS SO FAR.
000558                  RTS
000559
000560 ; ###############################################################################################
000561 ; #   END OF FILE:  B3MAINC.TEXT
000562 ; #   LINES      :  553
000563 ; #   CHARACTERS :  26314
000564 ; ###############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 564   CHARACTERS: 26866
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "EXTRAS.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:34 PM
|  Modified: Wednesday, December 31, 1997             4:37:11 PM
|
+------------------------------------------------------------------------------

000001 ; ###########################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  EXTRAS.TEXT
000004 ; ###########################################################################################
000005
000006                  SBTL      "APPLESOFT EXTENSIONS"
000007 SOUTREC          JSR       CHKBYT                    ;SET OUTREC ROUTINE.  GET = AND EXPR IN X
000008                  STX       OUTREC                    ;AND SET VARIABLE.
000009                  RTS
000010 SINDENT          JSR       CHKBYT                    ;SET INDENT ROUTINE
000011                  STX       INDENT                    ;GET = AND EXPRESSION IN X.
000012                  RTS
000013 CHKBYT           JSR       CHKEQL                    ;CHECK FOR AN "=".
000014                  JMP       GETBYT                    ;EVALUATE EXPRESSION INTO X REGISTER.
000015 HTAB             JSR       HTABB                     ;DO THE TABBING.
000016                  STA       TRMPOS
000017                  RTS
000018 TOOBIG           JMP       FCERR
000019 HTABB            LDA       #24                       ;HTAB CHAR.
000020                  DFB       44
000021 VTAB             LDA       #25                       ;VTAB COMMAND.
000022                  PHA                                 ;SAVE IT.
000023                  JSR       CHKEQL                    ;EAT THE EQUALS.
000024                  JSR       GETBYT
000025                  PLA
000026 VWINDER          JSR       PRNACHAR                  ;SEND THE COMMAND.
000027                  TXA                                 ;GET THE ARGUMENT.
000028                  JMP       DOITOUT                   ;SEND THE ARGUMENT.
000029 SETNORM          LDA       #17                       ;NORMAL VIDEO.
000030                  DFB       44                        ;SKIP 2 BYTES.
000031 INVERSE          LDA       #18                       ;INVERSE VIDEO.
000032                  JMP       PRNACHAR                  ;DO IT!!
000033 SETTRACE:        SEC
000034                  BCC       SETTRACE
000035                  ORG       *-1
000036 TRACEOFF:        CLC                                 ;ADJUST TRFLAG FOR TRACE.
000037                  BNE       RTSBCK                    ;IF A TERMINATOR IMMEDIATELY FOLLOWING, ERROR!
000038                  ROR       TRFLAG
000039 RTSBCK           RTS                                 ;BACK TO CALLER.
000040 ONERR            JSR       ERRDIR                    ;DON'T DO DIRECT
000041                  LDA       TXTPTR
000042                  STA       ERRTO
000043                  LDA       TXTPTR+1
000044                  STA       ERRTO+1
000045                  LDA       TXTPTRB
000046                  STA       ERRTOB
000047                  LDA       ERRFLG                    ;SET MINUS BIT
000048                  ORA       #$80
000049                  STA       ERRFLG
000050                  LDA       CURLIN                    ;ALL INFO FOR 'GOTO' COMMAND
000051                  STA       ERRTO+3
000052                  LDA       CURLIN+1
000053                  STA       ERRTO+4
000054                  JSR       REMN                      ;SKIP REST OF LINE.
000055                  JMP       ADDON                     ;FINISH.
000056 HNDLERR:         EQU       *
000057                  LDX       REMSTK                    ;PRESERVE STACK POINTER
000058                  TXS
000059                  LDA       OLDTXT
000060                  STA       ERRPOS
000061                  LDA       OLDTXT+1
000062                  STA       ERRPOS+1
000063                  LDA       OLDTXTB
000064                  STA       ERRPOSB
000065 *--ALL USER INFO NOW THERE.
000066                  LDA       ERRTO
000067                  STA       TXTPTR
000068                  LDA       ERRTO+1
000069                  STA       TXTPTR+1
000070                  LDA       ERRTOB
000071                  STA       TXTPTRB
000072                  LDA       ERRTO+3
```

```
000073                  STA       CURLIN
000074                  LDA       ERRTO+4
000075                  STA       CURLIN+1
000076                  JMP       DIRCON                  ;BACK TO NEWSTT.
000077 RESUME:           JSR       ERRDIR                  ;MUST BE DEFFERED.
000078                  LDA       ERRPOSB                 ;ANY ERRORS YET?
000079                  BEQ       RSM2
000080                  STA       TXTPTRB
000081                  LDA       ERRLIN
000082                  STA       CURLIN
000083                  LDA       ERRLIN+1
000084                  STA       CURLIN+1
000085                  LDA       ERRPOS
000086                  STA       TXTPTR
000087                  LDA       ERRPOS+1
000088                  STA       TXTPTR+1
000089 RSM2:             RTS                               ;BACK TO TRY AGAIN.....
000090 *
000091 * THIS ROUTINE TESTS FOR AN ESCAPE TOKEN.  RETURNS THE
000092 * A-REG=CHAR POINTED AT BY (TXTPTR). RETURNS Z FLAG SET IF
000093 * EQUAL.
000094 TRYESC:           STY       YSAVE
000095                  LDY       #1                      ;TEST SECOND BYTE IN SEQUENCE
000096                  CMP       (TXTPTR),Y
000097                  BNE       TRY2                    ;NO MATCH, FORGET IT
000098                  DEY
000099                  LDA       (TXTPTR),Y              ;IS IT AN ESCAPE TOKEN ?
000100                  CMP       #$FF
000101                  BNE       TRY2                    ;SORRY CHARLIE
000102                  JSR       CHRGET                  ;EAT THE ESCAPE TOKEN
000103                  LDA       #$00                    ;AND SET THE Z FLAG
000104 TRY2              PHP
000105                  JSR       CHRGOT                  ;GET THE CHAR AT THIS POSITION
000106                  LDY       YSAVE
000107                  PLP
000108                  RTS
000109 MSTESC            PHA
000110                  LDA       #$FF
000111                  JSR       SYNCHR
000112                  PLA
000113                  JMP       SYNCHR
000114 ONKBD:            JSR       ERRDIR
000115                  LDA       #1
000116                  STA       KBDKEY                  ;PRIORITY 1.
000117                  BRK                               ;SOS
000118                  DFB       SDCNT                   ;DEVICE-CONTROL.
000119                  DW        DKBD                    ;ON ANY-KEY EVENT.
000120                  LDX       #11                     ;HANDLE AS FILE NUMBER 11
000121                  BNE       ONEOF2
000122 ONEOF:            JSR       CHRGET
000123                  JSR       ERRDIR                  ;NOT IN DIRECT MODE YOU DON'T!!!
000124                  JSR       GTFLNO                  ;GET FILE NUMBER
000125                  INX                               ;+1 FOR KICKS
000126                  JSR       DECTPT
000127 ONEOF2            TXA
000128                  STA       YSAVE
000129                  ASL       A                       ;*3 TO FORM INDEX INTO EOFPTRS
000130                  ADC       YSAVE
000131                  TAX
000132                  JSR       RELTXT                  ;MAKE TXTPTR RELATIVE.
000133                  LDA       TXTPTR
000134                  STA       EOFPTRS-2,X
000135                  LDA       TXTPTR+1
000136                  STA       EOFPTRS-1,X
000137                  LDA       TXTPTRB
000138                  STA       EOFPTRS-3,X
000139                  JSR       RELTXT                  ;MAKE TXTPTR ABSOLUTE AGAIN.
000140                  LDA       CURLIN                  ;SAVE LINE NUMBER
000141                  STA       EOFLINS-3,X
000142                  LDA       CURLIN+1
000143                  STA       EOFLINS-2,X
000144                  JSR       REMN                    ;SKIP REST OF LINE.
000145                  JMP       ADDON
000146 RELTXT            EQU       *                       ;MAKE TXTPTR RELATIVE (OR ABSOLUTE).
000147                  SEC
000148                  LDA       TXTTAB
000149                  SBC       TXTPTR
000150                  STA       TXTPTR
000151                  LDA       TXTTAB+1
000152                  SBC       TXTPTR+1
```

```
000153                LDY       TXTTABB
000154                JSR       FIXSBC
000155                STA       TXTPTR+1
000156                TYA
000157                SBC       TXTPTRB
000158                STA       TXTPTRB
000159                RTS
000160 OFF:            EQU       *
000161                PHA
000162                JSR       CHRGET                    ;EAT NEXT TOKEN
000163                PLA
000164                CMP       #KBDTK
000165                BEQ       OFFKBD
000166                CMP       #EOFTK
000167                BEQ       OFFEOF
000168                CMP       #ERRTK
000169                BNE       RTS999
000170                LDA       ERRFLG                    ;CLEAR ON ERR BIT
000171                AND       #$7F
000172                STA       ERRFLG
000173 RTS999         RTS
000174 OFFKBD:        LDA       #0
000175                STA       KBDKEY
000176                BRK
000177                DFB       SDCNT                     ;DEVICE-CONTROL.
000178                DW        DKBD                      ;ON ANY-KEY EVENT.
000179                LDX       #11                       ;FILE NUMBER 11
000180                BNE       OFFEOF2
000181 OFFEOF:        JSR       GTFLNO
000182                INX
000183 OFFEOF2:       TXA
000184                STA       YSAVE                     ;*3.
000185                ASL       A
000186                ADC       YSAVE
000187                TAX
000188                LDA       #0
000189                STA       EOFPTRS-2,X
000190                STA       EOFPTRS-1,X
000191                STA       EOFPTRS-3,X
000192                RTS
000193 CHKEOF:        INC       SVFLNO                    ;ONE MORE FOR KICKS
000194                LDA       SVFLNO
000195                STA       EOFSV
000196                ASL       A
000197                ADC       SVFLNO
000198                LDX       REMSTK                    ;QUICK FIX THE STACK BEFORE ANYONE NOTICES
000199                TXS
000200                TAX
000201 CKEOF2         LDA       FILNO+1                   ;GO TO NORMAL OUTPUT
000202                STA       FILNO
000203                LDA       EOFPTRS-3,X
000204                BEQ       GIVOD                     ;NO EOF, BLOW HIM OUT.
000205                STA       TXTPTRB
000206                LDA       EOFPTRS-2,X
000207                STA       TXTPTR
000208                LDA       EOFPTRS-1,X
000209                STA       TXTPTR+1
000210                LDA       EOFLINS-3,X               ;GET LINE OF ON KBD ROUTINE
000211                STA       CURLIN
000212                LDA       EOFLINS-2,X
000213                STA       CURLIN+1
000214                JSR       RELTXT
000215                JMP       GONE
000216 GIVOD:         LDX       #ERROD                    ;OUT-OF-DATA ERROR
000217                JMP       ERROR
000218 KEYHIT         LDA       KEYSTROK                  ;TURN OFF THE ONKBD FLAG
000219                AND       #$7F
000220                STA       KEYSTROK
000221                JSR       PSHTXT4                   ;PUT A 'GOSUB' ENTRY ON THE STACK
000222                LDX       #EOFSIZ
000223                BNE       CKEOF2                    ;ALWAYS TAKEN
000224 *
000225 * HERE IS THE DELETE CODE...
000226 *
000227 BLOWDEL:       JMP       SNERR
000228 DELETE         BCS       BLOWDEL                   ;MUST HAVE A DIGIT FIRST
000229                JSR       LINGET                    ;MUST HAVE LINE,LINE
000230                JSR       FNDLIN                    ;GET ITS POSITION
000231                LDX       CURLIN+1
000232                INX
```

```
000233                  BEQ       RNGOK
000234                  LDA       CURLIN
000235                  CMP       LINNUM
000236                  LDA       CURLIN+1
000237                  SBC       LINNUM+1
000238                  BCC       RNGOK
000239 RNGERR            LDX       #ERRNG
000240                  JMP       ERROR
000241 RNGOK             LDA       LOWTR                   ;SAVE FOR A MOMENT
000242                  PHA
000243                  LDA       LOWTR+1
000244                  PHA
000245                  LDA       LOWTRB
000246                  PHA
000247                  JSR       CHRGOT                  ;MUST HAVE A COMMA, 'TO', OR A DASH
000248                  BEQ       JSONEL                  ;JES ONE LINE, MA...
000249                  CMP       #','
000250                  BEQ       DELOK
000251                  CMP       #'-'
000252                  BEQ       DELOK
000253                  LDA       #TOTK
000254                  JSR       TRYESC
000255                  BNE       BLOWDEL                 ;IF NONE, FUCK HIM!
000256 DELOK:            JSR       CHRGET                  ;EAT SEPARATOR
000257                  BCS       BLOWDEL                 ;COMMA MUST BE FOLLOWED BY DIGIT.
000258                  JSR       LINGET
000259 JSONEL            INC       LINNUM                  ;DELETE TO THE BEGINNING OF THE NEXT LINE NUMBER
000260                  BNE       *+4
000261                  INC       LINNUM+1
000262                  JSR       FNDLIN
000263                  LDA       LOWTR
000264                  STA       INDEX1                  ;SET UP FOR MOVE DOWN
000265                  LDA       LOWTR+1
000266                  STA       INDEX1+1
000267                  LDA       LOWTRB
000268                  STA       INDEX1B
000269                  PLA
000270                  STA       LOWTRB
000271                  PLA
000272                  STA       LOWTR+1
000273                  PLA
000274                  SEC                               ;CALCULATE THE DELTA FOR THE MOVE
000275                  STA       LOWTR
000276                  SBC       INDEX1
000277                  STA       DELTA
000278                  LDA       LOWTR+1
000279                  SBC       INDEX1+1
000280                  LDY       LOWTRB
000281                  JSR       FIXSBC
000282                  STA       DELTA+1
000283                  TAX
000284                  BPL       *+3
000285                  RTS
000286                  TYA
000287                  SBC       INDEX1B
000288                  STA       DELTAB
000289 ; OR NOTHING THERE TO DELETE)
000290                  JSR       MVDWN                   ;MOVE IT!
000291                  JMP       FLOAD                   ;CLEAN UP THINGS A BIT...
000292 MSETTXT:          BNE       TXTRTS                  ;TEXT Command
000293                  LDX       #0
000294 TXTOUT:           LDA       TXTCHRS,X
000295                  BEQ       TXTRTS
000296                  JSR       PRNACHAR
000297                  INX
000298                  BNE       TXTOUT
000299 TXTRTS            RTS
000300 HOME              BNE       TXTRTS
000301                  LDA       #$1C                    ;CLEAR SCREEN-
000302                  JMP       PRNACHAR
000303 TXTCHRS           DFB       $10,2,6                 ;Set text mode, Mode 2, Cursor Off
000304                  DFB       $15,$D                  ;Set text option, option value $D
000305                  DFB       $11,1                   ;Normal video, Save environment
                                                                & release viewport
000306                  DFB       $F,$0D,0                ;Screen on, CR, (end of list)
000307
000308 ; ###############################################################################################
000309 ; #   END OF FILE:  EXTRAS.TEXT
000310 ; #   LINES      :  301
000311 ; #   CHARACTERS :  12714
```

```
000312 ; ##############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 312   CHARACTERS: 13264
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "SOSSTUF.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:37 PM
|  Modified: Wednesday, December 31, 1997             4:37:15 PM
|
+------------------------------------------------------------------------------

000001 ; ###########################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  SOSSTUF.TEXT
000004 ; ###########################################################################################
000005
000006                  SBTL      "SOS INTERFACE STUFF"
000007 *
000008 *  NOTE--> This region must NOT be Write Protected!!
000009 *
000010 OPNCNST          DFB       4                         ;OPEN CONSOLE FILE
000011                  DW        CONSFN
000012 CONSRFN          DFB       0                         ;Ref Num returned here
000013                  DW        0                         ;(for Bob's Bug)
000014                  DFB       0                         ;# requests on the OPEN
000015 CONSFN           DFB       8                         ;Len of .CONSOLE
000016                  ASC       ".CONSOLE"
000017 SINIT            DFB       3                         ;Console Init table
000018                  DFB       0                         ;Ref Num goes here
000019 SPRNTPL          DW        SICHRS
000020 OUTSTRL          DW        SICLEN
000021 ISNLTB           DFB       3                         ;IS.NEW.LINE table
000022                  DFB       0                         ;Ref Num goes here
000023                  DFB       $FF,$0D                   ;Enable CR
000024 RDCTC            DFB       4
000025 GETREF           DFB       0
000026                  DW        KEYSAVE                   ;Location for the One Byte Read
000027 RFLUSH           DW        1                         ;The 2 bytes double as a parameter list for Flush
000028 RNDGOT           DW        0
000029 SCHRTB           DFB       0                         ;PCOUNT
000030                  DFB       0                         ;Console Ref Num
000031                  DW        OUTCHAR                   ;Buffer...
000032                  DW        1                         ;Single Char at a time
000033                  DW        0                         ;# bytes read (ignored here)
000034 OUTCHAR          DFB       0                         ;Char to READ/PRINT
000035 SLINTB           DFB       4                         ;READ a line
000036                  DFB       0                         ;Ref Num
000037                  DW        BUF
000038                  DW        INALLWD
000039 SNOCHRS          DW        0                         ;# actually read
000040 DINIT            DFB       3
000041 INDN             DFB       0,0
000042                  DW        *
000043 DGCN             DFB       2
000044                  DW        CONSFN
000045 OTDN             DFB       0
000046 DKBD             DFB       3
000047 INDN3            DFB       0
000048                  DFB       8
000049                  DW        KBDKEY
000050 KBDKEY           DFB       1
000051                  DFB       1
000052                  DW        KBDEVNT
000053 KBDBNK           DFB       0
000054 DCNTR            DFB       3
000055 INDN2            DFB       0
000056                  DFB       6                         ;Attention-key event
000057                  DW        DCCPRI
000058 DCCPRI           DFB       1
000059                  DFB       1
000060                  DW        CTCEVNT
000061 DCNTBNK          DFB       0
000062                  DFB       3                         ;Control-C
000063 DFLUSH           DFB       3
000064 INDN6            DFB       0
000065                  DFB       5
000066                  DW        $3000                     ;Dummy Parameter
000067 DECHO            DFB       3
000068 INDN4            DFB       0
000069                  DFB       11                        ;Screen echoing
000070                  DW        ECHOFLG
000071 ECHOFLG          DFB       0
000072 REDCUR           DFB       3                         ;Read position of the cursor
```

```
000073 INDN5           DFB       0                       ;Console Ref Num
000074                 DFB       16
000075                 DW        CURX
000076 CURX            DFB       0                       ;Cursor X position
000077 CURY            DFB       0                       ;Cursor Y position
000078 CTCOFF          LDA       #0                      ;Priority 0 turns off CTL-C sniffing
000079                 STA       DCCPRI
000080 CTCON           BRK                               ;Reinstate the interrupt
000081                 DFB       SDCNT
000082                 DW        DCNTR
000083                 LDA       #1
000084                 STA       DCCPRI
000085                 RTS
000086 KBDEVNT         EQU       *
000087                 LDA       #$80
000088                 DFB       44                      ;Skip the next LDA
000089 CTCEVNT         LDA       #$40
000090                 ORA       KEYSTROK                ;Set the Flags
000091                 STA       KEYSTROK
000092                 LDA       CONSRFN
000093                 STA       GETREF
000094 DOAGET          JSR       ECHOFF
000095                 JSR       CTCOFF
000096                 BRK
000097                 DFB       SRED
000098                 DW        RDCTC
000099                 PHA                               ;Save error code
000100                 JSR       ECHOON
000101                 PLA
000102                 BEQ       SOSRTS                  ;No SOS error
000103                 LDX       GETREF                  ;Were we EXECing or doing a GET #?
000104                 CPX       INFLNO
000105                 BEQ       *+5
000106                 JMP       DSKEOF                  ;A GET #, Check if ON EOF# set
000107                 JSR       EXCCLS                  ;Close the EXEC file
000108                 LDX       SLINTB+1                ;Restore the Ref Num
000109                 STX       GETREF
000110                 BNE       DOAGET                  ;Always
000111 ECHOON          LDA       #$80
000112                 DFB       44                      ;Skip the next LDA
000113 ECHOFF          LDA       #0                      ;Disable console echoing
000114                 STA       ECHOFLG
000115                 BRK
000116                 DFB       SDCNT
000117                 DW        DECHO
000118                 JMP       CTCON
000119 *
000120 *  Here is the routine called by all the Disk Commands
000121 *
000122 GOSOS           BRK                               ;Call SOS
000123 SCN             DFB       0                       ;SOS call number
000124                 DW        SOSTBL
000125 SOSRTS          RTS
000126 *
000127 *  Here are all the tables used by SOS
000128 *
000129 SOSTBL          DFB       0                       ;Parameter count
000130 PTHPTR          DW        0                       ;Path pointer
000131 RWRFNM          EQU       PTHPTR                  ;Refnum
000132 NWLNB           EQU       PTHPTR+1                ;IS_NEW_LINE Boolean
000133 SBFPTR          EQU       PTHPTR+1                ;Pointer to Buffer for READ/WRITE
000134 BASE            EQU       PTHPTR+1                ;Base indicator for SET_MARK
000135 OUTMRK          EQU       PTHPTR+1                ;File Position & length return value
000136 ISRCHMD         EQU       PTHPTR                  ;Search mode for FIND_SEG
000137 ISEGID          EQU       PTHPTR+1
000138 SEGNUM          EQU       PTHPTR                  ;Used by RELEASE_SEG
000139 JMODE           EQU       PTHPTR
000140 CRTLST          DW        0                       ;Pointer to CREATE list
000141 NWPTHNM         EQU       CRTLST                  ;New path name for RENAME
000142 FLSTPTR         EQU       CRTLST                  ;File list pointer
000143 REFOUT          EQU       CRTLST                  ;Returned Ref Num from OPEN
000144 OPNLST          EQU       CRTLST+1                ;OPEN parameter list pointer
000145 NLCHR           EQU       CRTLST                  ;NEW_LINE character
000146 INBYTES         EQU       CRTLST+1                ;Bytes to READ/WRITE
000147 DSPLMNT         EQU       CRTLST                  ;Displacement for POSITION
000148 IOPGCN          EQU       CRTLST
000149 INLNGTH         DFB       0                       ;Length for CREATE
000150 OPNLNGTH        DFB       0                       ;Length for OPEN
000151 OUTBYTES        EQU       OPNLNGTH                ;# of bytes really read
000152 BSBNKP          EQU       INLNGTH
```

```
000153                  DW        0                       ;Out Limit Bank/Pages goes here
000154 OSEGNM           DFB       0
000155 *
000156 *  CREATE List
000157 *
000158 CRTTBL           EQU       *
000159 INFLID           DFB       0                       ;In file ID
000160 INAUXID          DW        0                       ;Aux ID (BASIC puts Record Length here)
000161 INSTRTYP         DFB       1                       ;Storage Type
000162 INEOF            DS        4                       ;(BASIC doesn't use this as yet)
000163 *
000164 *  File Info Table
000165 *
000166 FATRB            DFB       0                       ;File Attributes (Access Code)
000167 FID              DFB       0                       ;File ID
000168 FAUX             DW        0                       ;Auxiliary ID
000169 FSTYP            DFB       0                       ;Storage type
000170 FEOF             DW        0                       ;End of File indicator
000171                  DW        0                       ;  (4 bytes)
000172 FBLKS            DW        0                       ;Block count
000173                  DW        0                       ;Date Parameter
000174 *
000175 *  OPEN Table
000176 *
000177 INREQ            DFB       0                       ;Requested access (READ/WRITE)
000178 HELF             DFB       3                       ;Do a GET_FILE_INFO on "HELLO"
000179                  DW        HELCN-1
000180                  DW        FATRB
000181                  DFB       3
000182                  DFB       5                       ;Length of "HELLO"
000183 HELCN            ASC       "HELLO"
000184                  DFB       0                       ;Terminator
000185 *
000186 *  Here is the main SOS interface routine
000187 GETFI            JSR       GETFISET                ;Fall into SETGO
000188 SETGO            JSR       SETUP
000189                  JSR       GOSOS                   ;Do the actual SOS call
000190                  BEQ       NOWRTS
000191                  JMP       SERROR
000192 SETUP            LDA       SCNUMT,Y
000193                  STA       SCN                     ;SOS Call number
000194                  LDA       #>SOSTBL
000195                  STA       SCN+1
000196                  LDA       #<SOSTBL
000197                  STA       SCN+2
000198                  LDA       PCNTT,Y
000199                  STA       SOSTBL
000200 NOWRTS           RTS
000201 *
000202 *  Table of System (SOS) Call Values, etc.
000203 *
000204 SCNUMT           DFB       SCRT                    ;CREATE
000205 CRT              EQU       0
000206                  DFB       SDST                    ;DESTROY
000207 DST              EQU       CRT+1
000208                  DFB       SRNM                    ;RENAME
000209 RNM              EQU       DST+1
000210                  DFB       SSFI                    ;SET_FILE_INFO
000211 SFI              EQU       RNM+1
000212                  DFB       SGFI                    ;GET_FILE_INFO
000213 GFI              EQU       SFI+1
000214                  DFB       SOPN                    ;OPEN
000215 OPN              EQU       GFI+1
000216                  DFB       SNWL                    ;NEW_LINE
000217 NWL              EQU       OPN+1
000218                  DFB       SRED                    ;READ
000219 RED              EQU       NWL+1
000220                  DFB       SWRT                    ;WRITE
000221 WRT              EQU       RED+1
000222                  DFB       SCLS                    ;CLOSE
000223 CLS              EQU       WRT+1
000224                  DFB       SSTM                    ;SET_MARK
000225 STM              EQU       CLS+1
000226                  DFB       SGTM                    ;GET_MARK
000227 GTM              EQU       STM+1
000228                  DFB       MRLS                    ;RELEASE_SEG
000229 RLS              EQU       GTM+1
000230                  DFB       MFND                    ;FIND_SEG
000231 FND              EQU       RLS+1
000232                  DFB       SSTE                    ;SET_EOF
```

```
000233 STE             EQU       FND+1
000234                 DFB       $64                     ;Read Joysticks
000235 PDL             EQU       STE+1
000236 *
000237 *  Parameter count table
000238 *
000239 PCNTT           DFB       3,1,2                   ;NOTE: This table must be in the
000240                 DFB       3,3,4                   ;      as SCNUMT (above).
000241                 DFB       3,4,3
000242                 DFB       1,3,2
000243                 DFB       1,6,3,2
000244                 SBTL      "MULTIPLE PREFIX STUFF"
000245 *
000246 * The PROGRAM_PREFIX support stuff follows here.
000247 *
000248 SPROGPFX        EQU       *                       ;STRIP THE FILENAME OFF THE PREFIX
000249                 LDY       PROGPATH                ;Get length including filename
000250                 BEQ       PRGRTS
000251                 INY                               ;Start at the last char
000252 PRGLP           DEY
000253                 LDA       PROGPATH,Y
000254                 CMP       #'/'
000255                 BNE       PRGLP
000256                 STY       PROGPATH                ;The PROGPATH Prefix length
000257 PRGRTS          RTS
000258 *
000259 * Set the PREFIX to either the SOS prefix or the PROGPATH prefix
000260 *
000261 SETSOS          EQU       *
000262                 LDA       #>SOSPATH               ;Low byte of SOS pathname buffer
000263                 STA       SHFTPFX
000264                 LDA       #<SOSPATH
000265                 STA       SHFTPFX+1
000266                 JMP       SETPFX
000267 SETPROG         EQU       *
000268                 LDA       CMDFLG                  ;Check if we came from RUN or CHAIN
000269                 BEQ       *+5                     ;Neither so don't redo PROGPATH
000270                 JSR       FILPROG                 ;Redo PROGPATH if needed
000271                 LDA       #>PROGPATH              ;Low byte of PROG pathname buffer
000272                 STA       SHFTPFX
000273                 LDA       #<PROGPATH
000274                 STA       SHFTPFX+1
000275 SETPFX          EQU       *
000276                 BRK
000277                 DFB       SETPREF                 ;SOS SET_PREFIX
000278                 DW        SHFTPFX-1
000279                 BEQ       PRGRTS
000280 MPERR           JMP       SERROR                  ;Multiple path SOS error jump
000281 *
000282 * Put the SOS prefix into the SOSPATH buffer
000283 *
000284 FILSOS          BRK
000285                 DFB       GETPREF
000286                 DW        PREFTAB
000287                 BNE       MPERR
000288                 LDY       CATBUF+1                ;Length of prefix from SOS
000289                 LDA       CATBUF+1,Y              ;(BPL *-7 back to here)
000290                 STA       SOSPATH,Y
000291                 DEY
000292                 BPL       *-7
000293                 RTS
000294 *
000295 * Put the PROG prefix into the PROGPATH buffer and Strip it
000296 *
000297 FILPROG         EQU       *
000298                 LDA       NAMBUF+1                ;Check for FULL qualification
000299                 CMP       #'.'                    ;Is first char a "." (device) or
000300                 BEQ       FILPROG2                ; a "/" (Volume) ?
000301                 CMP       #'/'
000302                 BNE       FILPROG1                ;Neither, so don't change PROGPATH
000303 FILPROG2        EQU       *
000304                 LDY       NAMBUF
000305                 LDA       NAMBUF,Y
000306                 STA       PROGPATH,Y
000307                 DEY
000308                 BPL       *-7
000309                 JSR       SPROGPFX
000310 FILPROG1        LDA       #0
000311                 STA       CMDFLG
000312                 RTS
```

```
000313 *
000314 *  Copy the SOS prefix from SOSPATH to PROGPATH for the Boot & Run
000315 *    environment (or if no Program Pathname is supplied)
000316 *
000317 COPYSOS         EQU       *
000318                LDY       SOSPATH
000319                LDA       SOSPATH,Y               ;(BPL *-7 back to here)
000320                STA       PROGPATH,Y
000321                DEY
000322                BPL       *-7
000323                RTS
000324 *
000325 *  The following subroutine guarantees that the PROGPATH is Volume
000326 *    oriented rather than device oriented.
000327 CNVTPFX1        EQU       *
000328                LDA       #>PROGPATH              ;Low byte of PROG pathname buffer
000329                STA       SHFTPFX
000330                LDA       #<PROGPATH
000331                STA       SHFTPFX+1
000332                BRK
000333                DFB       SETPREF                 ;SOS SET_PREFIX
000334                DW        SHFTPFX-1
000335                LDA       #2
000336                STA       SHFTPFX-1
000337                BRK
000338                DFB       GETPREF                 ;SOS GET_PREFIX
000339                DW        SHFTPFX-1
000340                DEC       SHFTPFX-1               ;Reset SHFTPFX-1 to 1
000341                RTS
000342 *  Parameter blocks for above calls
000343                DFB       1                       ;Set Prefix=1; Get Prefix=2
000344 SHFTPFX         DW        0
000345                DFB       80
000346
000347 ; ##############################################################################################
000348 ; #   END OF FILE:  SOSSTUF.TEXT
000349 ; #   LINES      :  340
000350 ; #   CHARACTERS :  14794
000351 ; ##############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 351   CHARACTERS: 15346
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3LISTD.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:28 PM
|  Modified: Wednesday, December 31, 1997             4:37:05 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3LISTD.TEXT
000004 ; ###############################################################################################
000005
000006                  SBTL      "THE 'LIST' COMMAND."
000007 LIST:            EQU       *
000008                  PHP                                 ;SAVE CHAR STATUS
000009                  LDX       #0
000010                  STX       DELTA
000011                  JSR       LINGET                    ;GET LINE NUMBER INTO NUMLIN.
000012                  JSR       FNDLIN                    ;FIND LINE .GE. NUMLIN.
000013                  PLP
000014                  BNE       GOLS1                     ;IF NOT A TERM.
000015 LXZYQ            LDA       #255
000016                  STA       LINNUM+1
000017 GOLS1            JSR       CHRGOT                    ;GET LAST CHARACTER.
000018                  BEQ       LSTEND                    ;IF END OF LINE, # IS THE END.
000019                  CMP       #'-'                      ;DASH?
000020                  BEQ       LSTOK
000021                  CMP       #$2C
000022                  BEQ       LSTOK
000023                  LDA       #TOTK
000024                  JSR       TRYESC
000025                  BEQ       LSTOK                     ;A "TO" "-" OR "," O.K. OTHERWISE....
000026                  JMP       SNERR                     ;SYNTAX ERROR!
000027 LSTOK            JSR       CHRGET                    ;GET NEXT CHAR.
000028                  BEQ       LXZYQ                     ;SO LIST 0 WILL WORK.
000029                  JSR       LINGET                    ;GET END #.
000030                  JSR       CHRGOT                    ;WHAT CHAR IS HERE?
000031                  BNE       LSTRTS                    ;IF NOT TERMINATOR, ERROR.
000032 LSTEND           LDA       LOWTR
000033                  STA       VARNAM
000034                  LDA       LOWTR+1
000035                  STA       VARNAM+1
000036                  LDA       LOWTRB
000037                  STA       VARNAMB
000038 LIST4            LDY       #0
000039                  LDA       (VARNAM),Y                ;IS LINK ZERO?
000040                  BEQ       GREEDY                    ;YES, GO TO READY.
000041                  BIT       KEYSTROK                  ;CNTRL-C HIT?
000042                  BVC       *+5                       ;NO, SKIP
000043                  JMP       ISCNTC
000044                  JSR       CRDO                      ;PRINT CRLF TO START WITH.
000045                  INY
000046                  LDA       (VARNAM),Y
000047                  TAX
000048                  INY
000049                  LDA       (VARNAM),Y                ;GET LINE NUMBER.
000050                  CMP       LINNUM+1                  ;SEE IF BEYOND LAST.
000051                  BNE       TSTDUN                    ;GO DETERMINE RELATION.
000052                  CPX       LINNUM                    ;WAS EQUAL SO TEST LOW ORDER.
000053                  BEQ       TYPLIN                    ;EQUAL, SO LIST IT.
000054 TSTDUN:          BCS       LSTRTS                    ;IF LINE IS GR THAN LAST, THEN DUNE.
000055 TYPLIN:          STY       LSTPNT
000056                  PHA                                 ;PRESERVE A SO WE CAN PRINT A SPACE.
000057                  JSR       ROUTSPC                   ;BEFORE THE LINE NUMBER
000058                  PLA                                 ;RESTORE A  (PART OF LINE #)
000059                  JSR       LINPRT                    ;PRINT AS INT WITHOUT LEADING SPACE.
000060                  LDX       INDENT                    ;NUMBER OF SPACES PER TAB.
000061                  BEQ       PLOOP3
000062 PLOOP1           LDY       DELTA                     ;NUMBER OF TABS.
000063 PLOOP0           JSR       OUTSPC                    ;PRINT CHAR.
000064                  DEY
000065                  BPL       PLOOP0                    ;OUTPUT THE RIGHT # OF SPACES.
000066                  DEX
000067                  BNE       PLOOP1
000068 PLOOP3           LDY       TRMPOS                    ;SAVE CURRENT CURSOR INDENT POSITION IN CASE
000069                  STY       DELTA+1                   ;THIS LINE WRAPS AROUND.
000070                  LDY       LSTPNT                    ;GET POINTER TO LINE BACK
000071 PLOOP5           CMP       #'"'                      ;CHECK FOR QUOTED STRINGS.
000072                  BEQ       PLOOP7
```

```
000073 PLOOP:          JSR       ROUTDO                    ;PRINT CHAR.
000074 ; HERE IS WHERE WE SHOULD CHECK IF THE LISTING SHOULD ADVANCE
000075 ; TO THE NEXT LINE.
000076 PLOOP2:         EQU       *                         ;HERE IS LOOP POINT FROM JPL2
000077                 INY
000078                 LDA       (VARNAM),Y                ;GET NEXT CHAR. IS IT ZERO?
000079                 BNE       QPLOP                     ;YES. END OF LINE.
000080 PLOOP8          SEC                                 ;COMPUTE RELATIVE LINE POSITION
000081                 TYA
000082                 ADC       VARNAM
000083                 STA       VARNAM
000084                 BCC       LIST4
000085                 LDX       VARNAM+1                  ;INC VARNAM+1
000086                 INX
000087                 CPX       #MAXPG
000088                 BCC       *+7
000089                 LDX       #MINPG
000090                 INC       VARNAMB
000091                 STX       VARNAM+1
000092                 JMP       LIST4                     ;BRANCH IF SOMETHING TO LIST.
000093 GETNXTW:        INY                                 ;NEXT CHAR IN
000094                 BNE       *+4                       ;RESLST
000095                 INC       FAC+1                     ;NEVER CROSSES BANK BOUNDARY.
000096                 LDA       (FAC),Y
000097 LSTRTS          RTS
000098 GREEDY          JMP       CRDO
000099 PLOOP7          JSR       ROUTDO
000100                 INY
000101                 LDA       (VARNAM),Y
000102                 BEQ       PLOOP8
000103                 CMP       #'""'
000104                 BNE       PLOOP7
000105                 BEQ       PLOOP
000106 ;IS IT A TOKEN?
000107 QPLOP:          BPL       PLOOP5                    ;NO, HEAD FOR PRINTER.
000108                 CMP       #FORTK
000109                 BNE       *+4
000110                 INC       DELTA
000111                 CMP       #FORTK+1                  ;THIS IS NEXTTK.
000112                 BNE       NOTSPEC4
000113                 STY       YSAVE
000114 NOTSPEC0        DEC       DELTA
000115                 BPL       NOTSPEC2
000116                 INC       DELTA                     ;BACK TO 0.
000117 NOTSPEC2        INY
000118                 LDA       (VARNAM),Y
000119                 BEQ       NOTSPEC3
000120                 CMP       #':'
000121                 BEQ       NOTSPEC3
000122                 CMP       #','
000123                 BEQ       NOTSPEC0
000124                 BNE       NOTSPEC2
000125 NOTSPEC3        LDY       YSAVE
000126                 LDA       (VARNAM),Y
000127 NOTSPEC4        LDX       #>RESLST
000128                 STX       FAC
000129                 LDX       #<RESLST-256
000130                 STX       FAC+1
000131                 LDX       #RESLSTB
000132                 STX       FACB
000133                 TAX                                 ;SET INPFLG NEGATIVE IF A NORMAL TOKEN,
000134                 INX                                 ;=00 FOR AN ESCAPE TOKEN
000135                 STX       INPFLG
000136                 CMP       #$FF
000137                 BNE       NRMTKN                    ;GET RESERVED WORD FROM RESLST IF A STATEMENT,
000138                 INY
000139                 LDA       (VARNAM),Y                ;FROM RESL2 IF AN ESCAPE TOKEN
000140                 LDX       #>RESL2
000141                 STX       FAC
000142                 LDX       #<RESL2-256
000143                 STX       FAC+1
000144                 LDX       #RESLSTB
000145                 STX       FACB
000146 NRMTKN          SEC
000147                 SBC       #128                      ;GET RID OF SIGN BIT AND ADD 1.
000148                 TAX                                 ;MAKE IT A COUNTER.
000149                 STX       TKNSAV                    ;SAVE TOKEN # FOR LISTING FANCY
000150                 STY       LSTPNT                    ;SAVE POINTER TO LINE.
000151                 LDY       #255                      ;LOOK AT RES'D WORD LIST.
000152 RESRCH:         DEX                                 ;IS THIS THE RES'D WORD?
```

```
000153                  BMI       PRIT25                    ;YES, GO TOSS IT UP
000154 RESCR1:          JSR       GETNXTW
000155                  BPL       RESCR1                    ;NO, CONTINUE PASSING.
000156                  BMI       RESRCH
000157 PRIT25:          EQU       *
000158                  LDA       LSTPNT                    ;SOMETIMES WE DON'T EVEN WANT THEM FOR
000159                  CMP       #3                        ; SPECIAL FUNCTIONS (LIKE AT THE
                                                                        BEGINNING OF A LINE)
000160                  BEQ       PRIT3
000161                  TYA                                 ;SAVE INDEX TO TOKEN'S TEXT
000162                  PHA
000163                  LDY       LSTPNT                    ;DON'T WANT EXTRA SPACES WHEN
                                                                        THE LAST THING WAS A TOKEN
000164                  DEY
000165                  BIT       INPFLG                    ;AN ESCAPE TOKEN?
000166                  BMI       PRIT26
000167                  DEY
000168                  LDA       TKNSAV                    ;CHEK FOR VALID ESC TOKEN
000169                  CMP       #ONEFUN-$80               ;A SPECIAL WORD?
000170                  BCS       PRIT27
000171 PRIT26           LDA       (VARNAM),Y                ;SO CHECK FOR THAT CASE...
000172                  CMP       #':'
000173                  BEQ       PRIT27
000174                  ASL       A                         ;HI BIT TELLS ALL. SHIFT IT TO CARRY
000175 PRIT27           PLA                                 ;FIRST MUST RESTORE Y
000176                  TAY
000177                  BCS       PRIT3                     ;NO SPACE IF LAST WAS TOKEN
000178                  JSR       ROUTSPC
000179 PRIT3:           JSR       GETNXTW                   ;PRINT THE RESERVED WORD
000180                  BMI       PRFINIS
000181                  JSR       ROUTDO                    ;PRINT THE CHARACTER
000182                  JMP       PRIT3                     ;ALL OF THE WORD
000183 PRFINIS:         AND       #$7F
000184                  JSR       ROUTDO                    ;PRINT LAST CHAR OF THE WORD
000185                  LDY       LSTPNT                    ;IS THE WORD FOLLOWED BY A SEPERATOR?
000186                  INY
000187                  CMP       #'A'                      ;DON'T OUTPUT EXTRA SPACES FOR TOKENS WITH
000188                  BCC       JPL2                      ; NON ALPHA ENDINGS SUCH AS PR#, CHR$(, ETC...
000189                  LDA       (VARNAM),Y                ;IF A TOKEN ALSO, PRINT A SPACE
000190                  BMI       PRF2
000191                  JSR       CKSEP
000192                  BCC       JPL2                      ;BRANCH IF A SEPERATOR
000193 PRF2:            JSR       ROUTSPC
000194 JPL2:            DEY
000195                  JMP       PLOOP2
000196                  SBTL      "RELATIVE FORPNT CREATE."
000197 PNTREL           STX       KIMY
000198                  LDX       #FORPNT
000199 PNTRL1           LDA       SMVARS                    ;MAKE THE VARIABLE POINTER RELATIVE
000200                  SEC                                 ;TO SMVARS. THIS IS IN CASE
000201                  SBC       0,X                       ;THE SIMPLE VARIABLE TABLE CHANGES DURING
000202                  STA       0,X                       ;LOOP STATEMENTS.
000203                  LDA       SMVARS+1
000204                  SBC       1,X
000205                  LDY       SMVARSB
000206                  JSR       FIXSBC
000207                  STA       1,X
000208                  TYA
000209                  SBC       SYSPAG,X
000210                  STA       SYSPAG,X
000211                  LDX       KIMY
000212                  RTS
000213                  SBTL      "THE 'FOR' STATEMENT."
000214 FOR:             JSR       LET
000215                  LDA       VALTYP
000216                  BEQ       *+5
000217                  JMP       TMERR                     ;ONLY INTEGER AND FLOATING.
000218                  LDA       ISARA
000219                  ROL       A
000220                  LDA       INTFLG                    ;WAS IT AN INT?
000221                  ADC       #0                        ;IN TO LOW BIT.
000222                  STA       TEMPFOR                   ;HIGH BIT IF INTEGER. LOW BIT IF ARRAY.
000223 ;READ THE VARIABLE AND ASSIGN IT
000224 ;THE CORRECT INITIAL VALUE AND STORE
000225 ;A POINTER TO THE VARIABLE IN VARPNT.
000226                  JSR       RELPTR                    ;MAKE FORPNT TO START OF VARS TABLE.
000227                  LDA       FORPNT+1
000228                  LDY       FORPNTB
000229                  JSR       FIXAY
000230                  STA       FORPNT+1
```

```
000231                  CPY       #$FE
000232                  BCS       *+5
000233                  JMP       OVERR
000234                  JSR       FNDFOR                  ;PNTR IS IN VARPNT, AND FORPNT.
000235                  BNE       NOTOL
000236                  LDA       #1
000237                  STA       HIGHDSB
000238                  STA       HIGHTRB
000239                  STA       LOWTRB
000240                  STA       HIGHDS+1
000241                  STA       HIGHTR+1
000242                  STA       LOWTR+1
000243                  INX
000244                  STX       HIGHTR
000245                  TXA
000246                  CLC
000247                  ADC       #FORSIZ
000248                  STA       HIGHDS
000249                  TSX
000250                  STX       LOWTR
000251                  JSR       BLTUC
000252                  TSX
000253                  TXA
000254                  CLC
000255                  ADC       #FORSIZ
000256                  TAX
000257                  TXS
000258 NOTOL:           PLA                               ;GET RID OF NEWSTT RETURN ADDRESS
000259                  PLA                               ;IN CASE THIS IS A TOTALLY NEW ENTRY.
000260                  LDA       #10
000261                  JSR       GETSTK                  ;MAKE SURE 20 BYTES ARE AVAILABLE.
000262                  JSR       SVTXT                   ;SAVE THE TEXT.
000263                  JSR       DATA                    ;MOVE TXTPTR TO END OF FOR STATEMENT.
000264                  JSR       PSHTXT3                 ;PUT FOR ENTRY ON THE STACK.
000265                  PLA                               ;GET RID OF THE 'GOSUB' TOKEN
000266                  JSR       RSTTXT                  ;RESTORE TXTPTR TO PREVIOUS VALUE
000267                  LDA       #TOTK
000268                  JSR       MSTESC                  ;'TO' IS NECESSARY.
000269                  JSR       CHKNUM                  ;VALUE MUST BE A NUMBER.
000270                  JSR       FRMNUM                  ;GET UPPER VALUE INTO FAC.
000271                  LDA       FACSGN                  ;PACK FAC.
000272                  ORA       #127
000273                  AND       FACHO
000274                  STA       FACHO                   ;SET PACKED SIGN BIT.
000275                  LDA       #>LDFONE
000276                  LDY       #<LDFONE
000277                  STA       INDEX1
000278                  STY       INDEX1+1
000279                  JMP       FORPSH                  ;PUT FAC ONTO STACK, PACKED.
000280 LDFONE:          LDA       #>FONE
000281                  LDY       #<FONE                  ;PUT 1.0 INTO FAC.
000282                  LDX       #0
000283                  JSR       MOVFM
000284                  LDA       #STEPTK
000285                  JSR       TRYESC                  ;A STEP IS GIVEN?
000286                  BNE       ONEON                   ;NO. AUME 1.0.
000287                  JSR       CHRGET                  ;YES. ADVANCE POINTER.
000288                  JSR       FRMNUM                  ;READ THE STEP.
000289 ONEON            JSR       SIGN                    ;GET SIGN IN ACCA.
000290                  JSR       PUSHF                   ;PUSH FAC ONTO STACK (THRU A).
000291                  LDA       FORPNT+1
000292                  PHA
000293                  LDA       FORPNT
000294                  PHA                               ;PUT PNTR TO VARIABLE ON STACK.
000295                  LDA       TEMPFOR
000296                  PHA
000297                  LDA       #FORTK                  ;PUT A FORTK ONTO STACK.
000298                  PHA
000299 ;FALL INTO NEWSTT
000300                  SBTL      "NEW STATEMENT FETCHER."
000301 ; Back here for new statement.  Char pointed to by TXTPTR is a : or
000302 ;   the End-of-line terminator.  The address of its location is left
000303 ;   on the stack when a statement is executed, so that it can merely
000304 ;   do a RTS when it is done.
000305 NEWSTT:         EQU       *
000306                  LDA       #0
000307                  STA       CMDFLG                  ;Reset CMDFLG to 0
000308                  JSR       SETSOS                  ;Set prefix to SOS PREFIX...
000309                  TSX                               ;IN CASE OF ERROR.
000310                  STX       REMSTK
```

```
000311                  LDA       FILNO+1                   ;MAKE SURE OUTPUT IS CORRECT
000312                  STA       FILNO
000313                  JSR       CHRGOT                    ;MUST HAVE A TERMINATOR
000314                  BNE       SNERR1
000315                  LDX       CURLIN+1
000316                  INX
000317                  BEQ       DIRCON
000318 NWSTT             LDA       TXTPTR
000319                  LDY       TXTPTR+1
000320                  STA       OLDTXT
000321                  STY       OLDTXT+1                  ;SAVE IN CASE OF RESTART BY INPUT.
000322                  LDA       TXTPTRB
000323                  STA       OLDTXTB
000324 DIRCON            BIT       KEYSTROK                  ;KEY OR CNTRL-C?
000325                  BVS       ISCTRLC                   ;CNTROL-C HIT.
000326                  BMI       ISAKEY                    ;YES, A KEY IS HIT.
000327                  LDY       #0
000328                  LDA       (TXTPTR),Y
000329                  BNE       GONE                      ;IF NOT EOL, DO STATEMENT
000330                  INY                                 ;LOOK AT LINK.
000331                  LDA       (TXTPTR),Y                ;IS LINK 0?
000332                  CLC
000333                  BEQ       INTERM                    ;YES - RAN OFF THE END.
000334                  INY                                 ;PUT LINE NUMB IN CURLIN.
000335                  LDA       (TXTPTR),Y
000336                  STA       CURLIN
000337                  INY
000338                  LDA       (TXTPTR),Y
000339                  STA       CURLIN+1
000340                  TYA
000341                  ADC       TXTPTR
000342                  STA       TXTPTR
000343                  BCC       CHKPGE
000344                  INC       TXTPTR+1
000345 CHKPGE            LDA       TXTPTR+1
000346                  CMP       #MAXPG
000347                  BCC       GONE
000348                  INC       TXTPTRB
000349                  SBC       #MAXPG-MINPG
000350                  STA       TXTPTR+1
000351 GONE              TSX
000352                  STX       REMSTK
000353                  BIT       TRFLAG                    ;IN TRACE MODE?
000354                  BPL       GOFORIT                   ;IF NOT, DO LINE
000355                  LDX       CURLIN+1                  ;IN DIRECT MODE?
000356                  INX
000357                  BEQ       GOFORIT                   ;IF SO, DON'T TRACE
000358                  LDA       #'#'
000359                  JSR       OUTDO                     ;FOR TRACE FORMAT
000360                  LDX       CURLIN
000361                  LDA       CURLIN+1
000362                  JSR       LINPRT
000363                  JSR       OUTSPC                    ;TRAILING BLANK.
000364 GOFORIT:          JSR       CHRGET
000365                  JSR       GONE3
000366 NEWRET            EQU       *-1
000367                  JMP       NEWSTT
000368 SNERR1            JMP       SNERR
000369 ISAKEY            JMP       KEYHIT
000370 ISCTRLC           JSR       ISCNTC
000371 GONE3:            BEQ       ISCRTS                    ;IF TERMINATOR, TRY AGAIN.
000372 ; No need to set up Carry since it will be Set if non-numeric,
000373 ;   and numerics will cause a SYNTAX ERROR like they should.
000374                  SBC       #ENDTK                    ;' ON ... GOTO AND GOSUB' COME HERE.
000375                  LDX       #TEMPST                   ;RESET TEMPS.
000376                  STX       TEMPPT
000377                  BCC       GLET
000378                  CMP       #SCRATK-ENDTK+1
000379                  BCS       SNERR1                    ;A Reserved word but not legally used
000380                  ASL       A                         ;MULTIPLY BY TWO.
000381                  TAY                                 ;MAKE AN INDEX.
000382                  LDA       STMDSP+1,Y
000383                  PHA
000384                  LDA       STMDSP,Y
000385                  PHA                                 ;PUT DISP ADDR ONTO STACK.
000386                  JMP       CHRGET
000387 INTERM:           BEQ       ENDCON                    ;GO ALL THE WAY
000388 GLET:             JMP       LET                       ;MUST BE A LET
000389                  SBTL      "RESTORE,STOP,END,CONTINUE,NULL,CLEAR."
000390 RESTOR:           SEC
```

```
000391                  LDX       TXTTABB
000392                  LDA       TXTTAB
000393                  SBC       #1
000394                  LDY       TXTTAB+1
000395                  BCS       RESFIN
000396                  DEY
000397                  JSR       FIXYX
000398 RESFIN:          STA       DATPTR
000399                  STX       DATPTRB
000400                  STY       DATPTR+1                ;READ FINISHES COME TO 'RESFIN'.
000401 ISCRTS:          RTS
000402 ;WAS IT A CONTROL-C??
000403 ISCNTC:          LDA       KEYSTROK
000404 ISCNTC2          AND       #$BF                    ;TURN OF $40 BIT.
000405                  STA       KEYSTROK
000406                  LDX       #$FF                    ;FOR BREAK ERROR NUMBER.
000407                  CMP       #$80
000408                  BCS       ISRESET                 ;SET FROM NEWSTT
000409                  LDY       CURLIN+1
000410                  INY
000411                  BEQ       ISRESET                 ;IF IN IMM MODE DON'T UPDATE ERRLIN.
000412                  DEY
000413                  STY       ERRLIN+1
000414                  STX       ERRNUM
000415                  LDY       CURLIN
000416                  STY       ERRLIN
000417                  BIT       ERRFLG                  ;IN ONERR MODE?
000418                  BPL       *+5                     ;IF SO, JUMP TO 'HNDLERR'
000419                  JMP       HNDLERR
000420 ISRESET          STX       FILNO                   ;DON'T OUTPUT TO A FILE.
000421                  SEC
000422                  BCS       STOP2                   ;C IS CLEAR FOR END, SET OTHERWISE.
000423 END:             CLC
000424 STOP:            BNE       CONTRT                  ;RETURN IF NOT CONT-C OR
000425 ;IF NO TERMINATOR FOR STOP OR END.
000426 ;C=0 SO WILL NOT PRINT 'BREAK'.
000427 STOP2            LDA       #255
000428                  STA       FILNO
000429                  STA       FILNO+1
000430                  LDA       TXTPTR
000431                  LDY       TXTPTR+1
000432                  LDX       CURLIN+1
000433                  INX
000434                  BEQ       DIRIS
000435                  STA       OLDTXT
000436                  STY       OLDTXT+1
000437                  LDA       TXTPTRB
000438                  STA       OLDTXTB
000439                  LDA       CURLIN
000440                  LDY       CURLIN+1
000441                  STA       OLDLIN
000442                  STY       OLDLIN+1
000443 DIRIS:           PLA                               ;POP OFF NEWSTT ADDR.
000444                  PLA
000445 ENDCON           BCC       GORDY                   ;CARRY CLEAR SO DON'T PRINT 'BREAK'.
000446                  BRK                               ;FLUSH THE INPUT BUFFER.
000447                  DFB       SDCNT
000448                  DW        DFLUSH
000449                  JSR       CRDO
000450                  LDA       #>BRKTXT
000451                  LDX       #0
000452                  LDY       #<BRKTXT
000453                  JMP       ERRFIN
000454 GORDY:           JMP       READY
000455 CONT:            BNE       CONTRT                  ;MAKE SURE THERE IS A TERMINATOR.
000456                  LDX       CURLIN+1
000457                  INX
000458                  BNE       CONTRT
000459                  STX       KEYSTROK
000460                  BRK
000461                  DFB       SDCNT
000462                  DW        DKBD
000463                  LDX       #ERRCN                  ;CONTINUE ERROR.
000464                  LDY       OLDTXTB                 ;A STORED TXTPTR OF ZERO IS SETUP
000465 ;BY STKINI AND INDICATES THERE IS
000466 ;NOTHING TO CTINUE.
000467                  BNE       *+5
000468                  JMP       ERROR                   ;'STOP', 'END', TYPING CRLF TO
000469 ;'INPUT' AND  C SETUP OLDTXT.
000470                  LDA       OLDTXT
```

```
000471                  STA       TXTPTR
000472                  STY       TXTPTRB
000473                  LDA       OLDTXT+1
000474                  STA       TXTPTR+1
000475                  LDA       OLDLIN
000476                  LDY       OLDLIN+1
000477                  STA       CURLIN
000478                  STY       CURLIN+1
000479                  PLA
000480                  PLA
000481                  JMP       NWSTT
000482 CONTRT:           RTS                           ;RETURN TO CALLER.
000483 COLD              JSR       CHRGOT              ;Check for a terminator
000484                  BEQ       COLD1               ;Terminator there, ok
000485                  JMP       SNERR               ;  else it's a SYNTAX ERROR
000486 COLD1             JSR       CLSALL              ;CLOSE ALL OPEN BASIC FILES
000487                  BRK
000488                  DFB       SCLS                ;SOS CLOSE_ALL
000489                  DW        *-2                 ;REFERENCE 0 AS REF.NUM
000490                  BRK
000491                  DFB       CLDSTRT
000492                  DW        *-2                 ;THIS SOS CALL DOES NOT RETURN TO ANYWHERE!
000493                  SBTL      "RUN,GOTO,GOSUB,RETURN."
000494 RUN:              BNE       *+5                 ;If nothing follows RUN, then RUN the
000495                  JMP       RUNC                ;  program in memory from the beginning
000496                  BCC       RUNL                ;RUN from a specific line number
000497                  CMP       #$2C                ;Did the jerk type a comma?
000498                  BEQ       RUNL2               ;Yes, RUN from the line #
000499                  JSR       LDRUN               ;Prepare for LOAD & RUN
000500                  LDA       #1
000501                  STA       RNFLG
000502                  JSR       DOLD2               ;LOAD the program...
000503                  JSR       CLEARC              ;CLEAN UP CRAP
000504                  JSR       CNVTPFX1            ;Convert pathname from Device to Volume
000505                  LSR       TRFLAG              ;TRACE OFF.
000506                  JMP       FRUN                ;FINISH UP.
000507 RUNL2             JSR       CHKCOM
000508 RUNL:             JSR       CLEARC
000509                  JSR       RUNC2               ;RUN A LINE IN THIS PROGRAM
000510                  JMP       NEWSTT
000511 ;
000512 ; A GOSUB entry on the stack has the following format: (in PULL order)
000513 ;   GOSUTK - 1 Byte
000514 ;   Current line number - 2 Bytes (Lo, Hi)
000515 ;   Pointer into text of the GOSUB statement - 3 Bytes (Bank, Hi, Lo)
000516 ; Total: 6 Bytes.
000517 PSHTXT:           LDA       #4                  ;THIS ROUTINE IS FOR 'FOR', 'GOSUB', 'ON KBD'
000518                  JSR       GETSTK              ;IS THERE ROOM ON THE STACK
000519 PSHTXT2:          PLA                           ;SAVE RETURN ADDRESS
000520                  STA       PNTSAV
000521                  PLA
000522                  TAX                           ;IN THE Y,X REGS
000523                  PLA
000524                  STA       INDEX
000525                  PLA
000526                  STA       INDEX+1             ;SAVE 2 RETURNS.
000527                  SEC                           ;PUSH RELATIVE TXTPTR
000528                  LDA       TXTPTR
000529                  SBC       TXTTAB
000530                  PHA
000531                  LDA       TXTPTR+1
000532                  SBC       TXTTAB+1
000533                  LDY       TXTPTRB
000534                  JSR       FIXSBC
000535                  PHA
000536                  TYA
000537                  SBC       TXTTABB
000538                  PHA
000539                  LDA       CURLIN+1
000540                  PHA
000541                  LDA       CURLIN
000542                  PHA
000543                  LDA       #GOSUTK
000544                  PHA
000545                  LDA       INDEX+1
000546                  PHA
000547                  LDA       INDEX
000548                  PHA
000549                  TXA                           ;RESTORE RETURN ADDRESS
000550                  PHA
```

```
000551                  LDA       PNTSAV
000552                  PHA
000553 PSHTRT           RTS                                 ;GO HOME
000554 PSHTXT3          JSR       PSHTXT2                   ;MUST BE JSR FOR EXTRA RETURN ADDRESS ON STACK.
000555                  RTS
000556 PSHTXT4          JSR       PSHTXT
000557                  RTS
000558
000559 ; ###############################################################################################
000560 ; #   END OF FILE:  B3LISTD.TEXT
000561 ; #   LINES      :  552
000562 ; #   CHARACTERS :  25459
000563 ; ###############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 563   CHARACTERS: 26011
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3GOTOE.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:27 PM
|  Modified: Wednesday, December 31, 1997             4:37:04 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3GOTOE.TEXT
000004 ; ###############################################################################################
000005
000006 GOSUB:          JSR       SVTXT                     ;SAVE THE CURRENT TXTPTR
000007                 JSR       CHRGOT
000008                 JSR       LINGET                    ;EAT THE LINE #.
000009                 JSR       PSHTXT                    ;PUT A GOSUB ENTRY ON THE STACK.
000010                 JSR       RSTTXT                    ;RESORE TXTPTR.
000011 RUNC2:          JSR       CHRGOT                    ;GET CHARACTER & SET CODES FOR LINGET.
000012                 JSR       GOTO                      ;USE RTS SCHEME TO 'NEWSTT'.
000013                 RTS
000014 GOTO:           JSR       LINGET                    ;PICK UP THE LINE NUMBER IN 'LINNUM'.
000015                 JSR       CHRGOT
000016                 BNE       PSHTRT
000017 GOTOB           JSR       REMN                      ;SKIP TO END OF LINE.
000018                 LDA       CURLIN
000019                 CMP       LINNUM                    ;DON'T SEARCH ENTIRE PROGRAM IF
000020                 LDA       CURLIN+1                  ;LOOKING FOR A LINE WITH A LARGER
000021                 SBC       LINNUM+1                  ;LINE NUMBER
000022                 BCS       LUK4IT                    ;TOO BAD, SEARCH ENTIRE PROGRAM
000023                 TYA
000024                 LDY       TXTPTRB
000025                 SEC
000026                 ADC       TXTPTR
000027                 LDX       TXTPTR+1
000028                 BCC       LUKALL
000029                 INX
000030                 CPX       #MAXPG
000031                 BCC       LUKALL
000032                 INY
000033                 LDX       #MINPG
000034                 BNE       LUKALL
000035                 BCS       LUKALL                    ;ALWAYS GOES.
000036 LUK4IT:         LDA       TXTTAB
000037                 LDX       TXTTAB+1
000038                 LDY       TXTTABB
000039 LUKALL:         JSR       FNDLNC0                   ;X,A ARE ALL SET UP.
000040                 BCC       USERR                     ;GOTO LINE IS NONEXISTANT.
000041                 LDA       LOWTR
000042                 SBC       #1
000043                 STA       TXTPTR
000044                 LDA       LOWTR+1
000045                 SBC       #0
000046                 LDY       LOWTRB
000047                 JSR       FIXSBC
000048                 STA       TXTPTR+1
000049                 TYA
000050                 SBC       #0
000051                 STA       TXTPTRB
000052 GORTS:          RTS                                 ;PROCESS THE STATEMENT.
000053 DATAIS          JSR       ERRDIR                    ;DATA STATEMENT.  MUST BE IN DEFFERED.
000054                 BNE       REM
000055 ; RETURN restores line # and TXTPTR from stack, and eliminates
000056 ;   all FOR entries in front of GOSUB
000057 RETURN:         BNE       GORTS                     ;NO TERM. BLOW UP.
000058                 LDA       #255
000059                 STA       TEMPFOR                   ;MAKE SURE NO MATCH WILL BE FOUND
000060                 JSR       FNDFOR                    ;GO PAST ALL THE 'FOR' ENTRIES.
000061                 CMP       #GOSUTK                   ;RETURN WITHOUT GOSUB?
000062                 BEQ       RETU1
000063                 LDX       #ERRRG
000064                 DFB       44
000065 USERR:          LDX       #ERRUS                    ;NO MATCH SO 'US' ERROR.
000066                 JMP       ERROR                     ;YES.
000067 RETU1:          TXS                                 ;REMOVE GOSUTK.
000068                 PLA
000069                 PLA
000070                 CPY       #>POPTKN*2                ;POP STATEMENT
000071                 BEQ       DOPOP
000072                 STA       CURLIN
```

```
000073                PLA
000074                STA       CURLIN+1                ;Get line Number 'GOSUB' was on.
000075                PLA
000076                STA       TXTPTRB                 ;GET BANK TO RETURN TO
000077                CLC                               ;SINCE IT IS RELATIVE
000078                PLA
000079                TAY                               ;TXTPTR WAS PUSHED ON BASSACKWARDS
000080                PLA
000081                ADC       TXTTAB
000082                STA       TXTPTR
000083                TYA
000084                ADC       TXTTAB+1
000085                LDY       TXTTABB
000086                JSR       FIXADC
000087                STA       TXTPTR+1
000088                TYA
000089                ADC       TXTPTRB
000090                STA       TXTPTRB
000091                JMP       NWSTT
000092 DATA:          JSR       DATAN                   ;SKIP TO END OF STATEMENT,
000093 ADDON:         TYA
000094                CLC
000095                ADC       TXTPTR
000096                STA       TXTPTR
000097                BCC       REMRTS
000098                LDA       TXTPTR+1
000099                ADC       #0
000100                LDY       TXTPTRB
000101                JSR       FIXADC
000102                STY       TXTPTRB
000103                STA       TXTPTR+1
000104 REMRTS:        RTS                               ;'NEWSTT' RTS ADDR IS STILL THERE.
000105 SNERR2:        JMP       SNERR
000106 REM            JSR       REMN                    ;SKIP REST OF STATEMENT.
000107                BEQ       ADDON                   ;ALWAYS BRANCHES.
000108 DATAN:         LDX       #':'                    ;'DATA' TERMINATES ON ':' AND NULL.
000109                DFB       44
000110 REMN:          LDX       #0
000111                STX       CHARAC                  ;PRESERVE IT.
000112                LDY       #0                      ;THIS MAKES CHARAC=0 AFTER SWAP.
000113                STY       ENDCHR
000114 EXCHQT:        LDA       ENDCHR
000115                LDX       CHARAC
000116                STA       CHARAC
000117                STX       ENDCHR
000118 REMER:         LDA       (TXTPTR),Y
000119                BEQ       REMRTS                  ;NULL ALWAYS TERMINATES.
000120                CMP       ENDCHR                  ;IS IT THE OTHER TERMINATOR?
000121                BEQ       REMRTS                  ;YES, IT'S FINISHED.
000122                INY                               ;PROGRESS TO NEXT CHARACTER.
000123                CMP       #34                     ;IS IT A QUOTE?
000124                BNE       REMER                   ;NO, JUST CONTINUE.
000125                BEQ       EXCHQT                  ;YES, TIME TO TRADE.
000126 DOPOP:         PLA                               ;GET OTHER STUFF OFF STACK
000127                PLA
000128                PLA
000129                PLA                               ;NEWSTT ADDR STILL THERE
000130                JMP       NEWSTT                  ;SO GO BACK.....
000131                PAGE
000132                SBTL      "'ON ... GO TO ...' CODE."
000133 ONGOTO:        CMP       #ERRTK                  ;IS IT AN 'ON ERR','ON KBD'
000134                BEQ       GOERR
000135                CMP       #KBDTK
000136                BEQ       GOKBD
000137                CMP       #EOFTK
000138                BNE       ONGOTO2
000139                JMP       ONEOF
000140 ONGOTO2        JSR       GETBYT                  ;GET VALUE IN FACLO.
000141                PHA                               ;SAVE FOR LATER.
000142                CMP       #GOSUTK                 ;AN 'ON ... GOSUB' PERHAPS?
000143                BEQ       ONGLOP                  ;YES.
000144 SNERR3:        CMP       #GOTOTK                 ;MUST BE 'GOTOTK'.
000145                BNE       SNERR2
000146 ONGLOP:        DEC       FACLO
000147                BNE       ONGLP1                  ;SKIP ANOTHER LINE NUMBER.
000148                PLA                               ;GET TOKEN
000149                CMP       #GOSUTK                 ;GOSUB?
000150                BNE       DOONGT                  ;NO, DO ON GOSUB
000151                JSR       SVTXT                   ;SAVE CURRENT TXTPTR.
000152                JSR       DATA                    ;SKIP THE REST OF STATEMENT.
```

```
000153                JSR       PSHTXT                    ;DO A GOSUB
000154                JSR       RSTTXT                    ;RESTORE THE TEXTPTR.
000155                JSR       DOONGT
000156                RTS
000157 DOONGT          JSR       CHRGET                    ;EAT THE COMMA.
000158                JSR       LINGET                    ;FIND THE LINE
000159                JMP       GOTOB                     ;AND POSITION TO IT
000160 ONGLP1:         JSR       CHRGET                    ;ADVANCE AND SET CODES.
000161                JSR       LINGET
000162                CMP       #44                       ;IS IT A COMMA?
000163                BEQ       ONGLOP
000164                PLA                                 ;REMOVE STACK ENTRY (TOKEN).
000165 ONGRTS:         RTS                                 ;EITHER END-OF-LINE OR SYNTAX ERROR.
000166 GOERR           JMP       ONERR
000167 GOKBD           JMP       ONKBD
000168                PAGE
000169                SBTL      "LINGET -- READ LINE # INTO LINNUM"
000170 ; 'LINGET' reads a line number from the Current Text position
000171 ;   Line numbers range from 0 to 64000-1.
000172 ;   The answer is returned in 'LINNUM'.
000173 ; 'TXTPTR' is updated to point to the termination character,
000174 ;   A = the termination character with condition codes set up
000175 ;   to reflect its value.
000176 LINGET:         LDX       #0
000177                STX       LINNUM                    ;INITIALIZE LINE NUMBER TO ZERO.
000178                STX       LINNUM+1
000179 MORLIN:         BCS       ONGRTS                    ;IT IS NOT A DIGIT.
000180                SBC       #'0'-1                    ;-1 SINCE C=0.
000181                STA       CHARAC                    ;SAVE CHARACTER.
000182                LDA       LINNUM+1
000183                STA       INDEX
000184                CMP       #25                       ;LINE NUMBER WILL BE .LT. 64000?
000185                BCS       SNERR3
000186                LDA       LINNUM
000187                ASL       A                         ;MULTIPLY BY 10.
000188                ROL       INDEX
000189                ASL       A
000190                ROL       INDEX
000191                ADC       LINNUM
000192                STA       LINNUM
000193                LDA       INDEX
000194                ADC       LINNUM+1
000195                STA       LINNUM+1
000196                ASL       LINNUM
000197                ROL       LINNUM+1
000198                LDA       LINNUM
000199                ADC       CHARAC                    ;ADD IN DIGIT.
000200                STA       LINNUM
000201                BCC       NXTLGC
000202                INC       LINNUM+1
000203 NXTLGC:         JSR       CHRGET
000204                JMP       MORLIN
000205                PAGE
000206                SBTL      "'LET' CODE."
000207 LET:            JSR       MYPTRGET                  ;GET PTR TO VAR INTO VARPNT, FORPNT
000208                JSR       CHKEQL                    ;'=' IS NECESSARY
000209                LDA       INTFLG                    ;SAVE FOR LATER.
000210                PHA
000211                LDA       ISARA                     ;GET WHEATHER ARRAY OR NOT.
000212                PHA
000213                JSR       FRMEVL                    ;Get value of formula into 'FAC'.
000214                JMP       LETP3
000215 LETP2:          LDA       INTFLG                    ;DOS INTERFACE ENTERS HERE
000216                PHA
000217                LDA       ISARA                     ;THIS LINE MAY NOT BE NEEDED
000218                PHA
000219 LETP3           PLA                                 ;GET BACK ISARA.
000220                STA       ISARA
000221                BIT       VALTYP                    ;MAKE SURE 'VALTYP' SPECIFIES NUMERIC.
000222                BMI       COPSTR                    ;IF NUMERIC, COPY IT.
000223                BVS       BMOVVF
000224                PLA                                 ;GET NUMBER TYPE.
000225                STA       INTFLG                    ;FOR "FOR".
000226 QINTGR:         BPL       COPFLT                    ;STORE A FLTING NUMR.
000227                JSR       ROUND                     ;ROUND INTEGER.
000228                JSR       AYINT                     ;MAKE 2-BYTE NUMBER.
000229                LDY       #0
000230                LDA       FACMO                     ;GET HIGH.
000231                STA       (FORPNT),Y                ;STORE IT.
000232                INY
```

```
000233                  LDA       FACLO                   ;GET LOW.
000234                  STA       (FORPNT),Y
000235                  RTS
000236 COPFLT:          JMP       MOVVF                   ;PUT NUMBER AT FORPNT.
000237 BMOVVF           PLA                               ;POP OFF VALTYP, WE DON'T NEED IT.
000238 BMOVF1           LDX       FORPNTB
000239                  LDA       FORPNT                  ;LOW BYTE OF PLACE TO MOVE TO.
000240                  LDY       FORPNT+1
000241                  JMP       STFACT
000242 COPSTR:          EQU       *
000243                  PLA                               ;IF STRING, NO INTFLG.
000244 INPCOM:          EQU       *
000245 ; ADD IN DIGIT TO FAC.
000246                  LDY       FACMOB                  ;WAS THIS A TEMP, OR VARIABLE?
000247                  BNE       COPY                    ;IT IS A VARIABLE, MAKE A COPY.
000248                  LDA       FACMO                   ; else here for DNTCPY:
000249                  LDY       FACMO+1
000250                  LDX       FACMOB
000251                  JMP       COPY.C
000252 COPY:            LDY       #0
000253                  LDA       (FACMO),Y
000254                  JSR       STRINI                  ;GET ROOM TO COPY STRING INTO.
000255 COPY.M           LDA       DSCPNT
000256                  LDY       DSCPNT+1                ;GET POINTER TO OLD DESCRIPTOR, SO
000257                  LDX       DSCPNTB
000258                  STA       STRNG1
000259                  STY       STRNG1+1                ;MOVINS CAN FINSTRING.
000260                  STX       STRNG1B
000261                  JSR       MOVINS                  ;COPY IT.
000262                  JSR       PUTNEW
000263                  LDA       FACMO
000264                  LDY       FACMO+1
000265                  LDX       FACMOB
000266 COPY.C:          STA       DSCPNT
000267                  STX       DSCPNTB
000268                  STY       DSCPNT+1                ;REMEMBER POINTER TO DESCRIPTOR.
000269                  JSR       FRETMS                  ;FREE UP THE TEMPORARY WITHOUT
000270 ;   FREEING UP ANY STRING SPACE.
000271                  LDA       FORPNT
000272                  LDY       FORPNT+1
000273                  LDX       FORPNTB
000274                  JSR       NOTNW2                  ;PUT THE POINTER TO THE STRING IN INDEX.
000275                  JSR       FRESPA                  ;FREE THE BUGGER IF POSSIBLE
000276                  LDY       #$2                     ;RESET Y.
000277 COPY.S           LDA       (DSCPNT),Y
000278                  STA       (FORPNT),Y
000279                  DEY
000280                  BPL       COPY.S
000281                  TAX                               ;SET Z FLAG.
000282                  BEQ       LETRTS                  ;DON'T BUILD BACKPOINTER FOR NULL STRINGS
000283 FIXBAK           JSR       RELPTR
000284                  LDA       FORPNT+1
000285                  LDY       FORPNTB
000286                  JSR       FIXAY                   ;PACK A WITH LOW BIT FROM Y.
000287                  CPY       #$FF                    ;CHECK IF <64K!
000288                  BCC       VAERR                   ;VARIABLE ERROR!
000289                  LDY       #2                      ;FOR (),Y.
000290                  STA       (HIGHDS),Y
000291                  DEY
000292                  LDA       FORPNT
000293                  STA       (HIGHDS),Y
000294                  TXA
000295                  BMI       *+5
000296                  LDA       #SIMTYP                 ;$41
000297                  DFB       44                      ;SKIP 2
000298                  LDA       #ARYTYP                 ;$81
000299                  DEY
000300                  STA       (HIGHDS),Y              ;STORE TYPE BYTE.
000301 LETRTS           RTS
000302 RELPTR           LDX       ISARA                   ;IS THIS AN ARRAY?
000303                  BPL       ISSIMP                  ;NO, SIMPLE SIMON
000304 RELPTR2          LDA       ARYTAB
000305                  SEC
000306                  SBC       FORPNT                  ;FORPNT=RELATIVE POINTER TO VARIABLE.
000307                  STA       FORPNT
000308                  LDA       ARYTAB+1
000309                  SBC       FORPNT+1
000310                  LDY       ARYTABB
000311                  JSR       FIXSBC
000312                  STA       FORPNT+1
```

```
000313                  TYA
000314                  SBC       FORPNTB
000315                  STA       FORPNTB
000316                  BCC       GOTRELA                 ;ALWAYS
000317                  JMP       OMERR                   ;MORE THAN 64K OF ARRAYS GIVES "OUT OF MEMORY".
000318 ISSIMP           JSR       PNTREL
000319 GOTRELA          RTS
000320 VAERR            LDX       #ERRVA
000321                  JMP       ERROR
000322                  SBTL      "PRINT CODE."
000323 STRDON:          JSR       STRPRT
000324 NEWCHR:          JSR       CHRGOT                  ;REGET LAST CHARACTER.
000325 PRINT:           BEQ       CRDO                    ;TERMINATOR SO TYPE CRLF.
000326 PRINTC:          BEQ       PRTRTS                  ;HERE AFTER SEEING TAB(X) OR , OR ;
000327 ;  IN WHICH CASE A TERMINATOR DOES NOT
000328 ;  MEAN TYPE A CRLF BUT JUST RTS.
000329                  LDA       #TABTK                  ;TAB FUNCTION?
000330                  JSR       TRYESC
000331                  BEQ       TABER                   ;YES.
000332                  LDA       #SPCTK                  ;SPACE FUNCTION?
000333                  JSR       TRYESC
000334                  CLC                               ;REMEMBER IF IT IS.
000335                  BEQ       TABER
000336                  CMP       #44                     ;A COMMA?
000337                  BEQ       COMPRT                  ;YES.
000338                  CMP       #59                     ;A SEMICOLON?
000339                  BNE       *+5                     ;NO.
000340                  JMP       NOTABR                  ;YES.
000341                  LDA       #$20                    ;WE CAN GET ANYTHING FROM FRMEVL
000342                  STA       VALTYP
000343                  JSR       FRMEVL                  ;EVALUATE THE FORMULA.
000344                  BIT       VALTYP                  ;A STRING?
000345                  BMI       STRDON                  ;YES.
000346                  BVC       ISBIN
000347                  JSR       LOUT
000348                  LDA       #>NUMSTR
000349                  STA       INDEX
000350                  LDA       #<NUMSTR
000351                  STA       INDEX+1
000352                  LDA       #NUMSTRB
000353                  STA       INDEXB
000354                  LDX       LENUM
000355                  JSR       STRPR3
000356                  BEQ       NEWCHR                  ;ALWAYS TAKEN.
000357 ISBIN            JSR       FOUT
000358                  JSR       STRLIT
000359                  JMP       STRDON
000360 CRDO:            EQU       *
000361                  LDA       #13                     ;MAKE TRMPOS LESS THAN LINE LENGTH.
000362                  JSR       OUTDO
000363                  STX       TEMP
000364                  STY       KIMY
000365                  LDX       FILNO                   ;IF OUTPUT TO A DEVICE, GIVE LF
000366                  BMI       DOLF                    ;LINE FEED TO CONSOLE FOR SURE
000367                  JSR       GTFLNO1
000368                  LDA       FCB,Y                   ;IS IT A DEVICE (NOT A DISK FILE?)
000369                  BPL       DOLFRT                  ;BRANCH IF A DISK FILE
000370 DOLF             LDA       #10                     ;OUTPUT THE LF
000371                  JSR       OUTDO
000372 DOLFRT           LDX       #0
000373                  STX       TRMPOS
000374                  LDX       TEMP
000375                  LDY       KIMY
000376 PRTRTS:          RTS
000377 PRTRTS1          BIT       VALTYP                  ;IS THIS REALY A STRING TYPE?
000378                  BPL       PRTRTS                  ;ONLY STRINGS FALL THROUGH SO THEY
000379                  JMP       FRECNOW                 ;FREE THE USED STRING.
000380 COMPRT           LDA       TRMPOS
000381                  ADC       #$0F                    ;16 POSITIONS PER COLUMN
000382                  AND       #$F0                    ;ROUND DOWN.
000383 *  CMP WNDWDTH ;ARE WE OUT OF THE WINDOW?
000384                  SEC
000385                  SBC       TRMPOS
000386                  TAX
000387                  JMP       XSPAC1                  ;PUT OUT THAT MANY SPACES.
000388 TABER:           PHP                               ;REMEMBER IF SPC OR TAB FUNCTION.
000389                  JSR       GTBYTC                  ;GET VALUE INTO ACCX.
000390                  STX       TEMP
000391                  JSR       CHKCLS                  ;MAKE SURE A CLOSING PAREN
000392                  JSR       DECTPT                  ;DON'T IGNORE NEXT THING IN LIST
```

```
000393                  PLP
000394                  LDX       TEMP
000395                  BCC       XSPAC                   ;PRINT X SPACES.
000396                  BNE       *+5                     ;TAB(0) IS ILLEGAL
000397                  JMP       TOOBIG
000398                  DEX                               ;COLUMN 1 IS FIRST
000399                  TXA
000400                  SBC       TRMPOS
000401                  BCC       NOTABR                  ;NEGATIVE, DON'T PRINT ANY.
000402                  TAX
000403 XSPAC:            INX
000404 XSPAC2:           EQU       *
000405                  DEX                               ;DECREMENT THE COUNT.
000406                  BNE       XSPAC1
000407 NOTABR:           JSR       CHRGET                  ;GET NEXT CHARACTER
000408                  JMP       PRINTC                  ;DON'T CALL CRDO.
000409 XSPAC1:           EQU       *
000410                  JSR       OUTSPC
000411                  BNE       XSPAC2
000412 ; PRINT STRING POINTED TO BY Y,A WHICH ENDS WITH A ZERO.
000413 ; IF STRING IS BELOW DSCTMP IT WILL BE COPIED INTO STRING
000414 STROUT:           JSR       STRLIT                  ;GET A STRING LITERAL.
000415 ; PRINT THE STRING WHOSE DESCRIPTOR IS POINTED TO BY FACMO
000416 STRPRT:           JSR       NOTFAC                  ;GET POINTER TO STRING INTO INDEX.
000417                  TAX                               ;SO STRPR3 WILL WORK
000418 STRPR3:           STX       OUTSTRL
000419                  TXA
000420                  BEQ       STRPR4                  ;IF A NULL STRING, SCREW IT.
000421                  LDY       SINIT+1
000422                  BIT       FILNO                   ;ARE WE OUTPUTTING TO A FILE?
000423                  BMI       GOOUT                   ;IF SO, DO IT SPECIAL
000424                  PHA
000425                  LDX       FILNO                   ;OUTPUT TO FILE
000426                  JSR       GTFLNO1
000427                  JSR       PRETXT
000428                  JSR       TSTOUT
000429                  LDA       FCB,Y                   ;GET REF.NUM
000430                  LDY       SINIT+1
000431                  STA       SINIT+1
000432                  PLA
000433 GOOUT             CLC
000434                  ADC       TRMPOS
000435                  STA       TRMPOS                  ;DON'T CARE IF CARRY.
000436                  LDA       #>INDEX
000437                  STA       SPRNTPL
000438                  LDA       #<INDEX
000439                  STA       SPRNTPL+1
000440                  BRK                               ;TELL SOS TO PRINT THE STRING
000441                  DFB       SWRT                    ;WRITE TO THE CONSOLE
000442                  DW        SINIT
000443                  STY       SINIT+1
000444                  BNE       OTDOER
000445 STRPR4            JMP       PRTRTS1
000446 ; 'OUTDO' outputs the character in ACC, using 'CNTWFL'
000447 ;   (Suppress or not), TRMPOS (Print Head Position),
000448 ;   Timing, etc.. No Registers are changed.
000449 OUTSPC:           EQU       *
000450                  LDA       #$20
000451                  DFB       44
000452 OUTQST:           LDA       #'?'
000453 OUTDO             INC       TRMPOS                  ;INC CURSOR POSITION.
000454                  BIT       FILNO                   ;OUT TO A FILE?
000455                  BMI       OUTLOC                  ;NO, OUT TO CONSOLE
000456                  STA       OUTCHAR                 ;OUT A CHAR TO A FILE
000457                  TXA                               ;SAVE REGS
000458                  PHA
000459                  TYA
000460                  PHA
000461                  LDA       #3
000462                  STA       SCHRTB
000463                  LDX       FILNO
000464                  JSR       GTFLNO1
000465                  JSR       PRETXT
000466                  JSR       TSTOUT
000467                  LDA       FCB,Y
000468                  LDY       SCHRTB+1
000469                  STA       SCHRTB+1
000470                  BRK
000471                  DFB       SWRT
000472                  DW        SCHRTB
```

```
000473                  STY       SCHRTB+1
000474                  BNE       OTDOER
000475                  PLA                             ;RESTORE REGS
000476                  TAY
000477                  PLA
000478                  TAX
000479                  LDA       OUTCHAR
000480                  RTS
000481 OTDOER            JMP       SERROR
000482 OUTLOC            JMP       PRNACHAR
000483 DOPRINT:          CMP       #'#'                  ;IS THIS A FILE ACCESS?
000484                  BNE       GPRNT2                ;NO
000485                  LSR       IOFLG
000486                  JSR       FILNUM                ;GET FILE NUMBER,STUFF
000487                  LDA       SVFLNO
000488                  STA       FILNO                 ;THIS IS THE OUTPUT FILE NUMBER
000489                  JSR       CHRGOT                ;IS THERE A SEMI-COLON?
000490                  BEQ       GPR3                  ;NO, A TERMINATOR, TERMINATE HIM
000491                  CMP       #$3B                  ;IS THERE A SEMI-COLON?
000492                  BNE       GPRNT
000493                  JSR       SYNCHR
000494 GPRNT:            LDA       #USINGTK              ;IS THIS PRINT USING?
000495                  JSR       TRYESC
000496                  BNE       GPR3
000497                  JSR       PRUSING               ;DO A PRINT USING OP
000498                  RTS
000499 GPR3              JSR       CHRGOT
000500                  JMP       PRINT                 ;DO THE PRINT...
000501 GPRNT2            BIT       FILNO                 ;IS THIS A PRINT WITH NO OUTPUT# ?
000502                  BPL       GPR4                  ;IF YES THEN READ THE CURSOR POSITION
                                                                 OFF OF SCREEN.
000503                  JSR       VPOS                  ;READ THE CURSOR POSITION.
000504                  LDA       CURX
000505                  STA       TRMPOS                ;NOW WE KNOW WHERE THE CURSOR REALLY IS.
000506 GPR4              JMP       GPRNT
000507 STROUTR           EQU       *                     ;FAST VERSION OF STROUT.
000508                  STA       INDEX                 ;DOESN'T USE STRING CODE.
000509                  STY       INDEX+1
000510                  STX       INDEXB
000511                  LDY       #$FF
000512 STRLUP            INY
000513                  LDA       (INDEX),Y             ;SCAN FOR A NULL.
000514                  BNE       STRLUP
000515                  TYA                             ;CHARACTER COUNT
000516                  TAX
000517                  LDA       #0
000518                  STA       VALTYP
000519                  JMP       STRPR3                ;PRINT THE STRING.
000520 ROUTSPC           LDA       #'                    ' ;ROUTSPC IS OUTSPC WITH WRAP ON /OUTREC/.
000521 ROUTDO            PHA                             ;ROUTDO is OUTDO with Wrap on /OUTREC/.
000522                  LDA       OUTREC                ;ARE WE LISTING BEYOND THE RIGHT HAND MARGIN?
000523                  BEQ       ROUTDONE              ;OUTREC=0 TURNS OFF WRAP MODE.
000524                  CLC
000525                  SBC       TRMPOS                ;CURRENT CUSOR BEYOND RIGHT MARGIN?
000526                  BEQ       *+4
000527                  BCS       ROUTDONE              ;NO, THEN JUST OUTPUT THE CHARACTER.
000528                  JSR       CRDO                  ;INSERT A CARRIAGE RETURN,
000529 TSTTAB            JSR       OUTSPC                ;AND MANY BLANKS INTO THE OUTPUT.
000530                  LDA       DELTA+1
000531                  CMP       TRMPOS                ;CURSOR BACK TO LEFT HAND MARGIN YET?
000532                  BCS       TSTTAB                ;YES, CONTINUE WITH NORMAL LISTING.
000533                  CMP       OUTREC                ;ANY SPACE LEFT TO LIST?
000534                  BCC       *+5                   ;YES, NO ERROR.
000535                  JMP       RNGERR                ;NO, GIVE RANGE ERROR.
000536 ROUTDONE          PLA                             ;RESTORE NEXT CHAR.
000537                  JMP       OUTDO                 ;AND LIST IT.
000538
000539 ; ##########################################################################################
000540 ; #   END OF FILE:  B3GOTOE.TEXT
000541 ; #   LINES      :  532
000542 ; #   CHARACTERS :  25642
000543 ; ##########################################################################################
```

```
+-----------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 543   CHARACTERS: 26194
|
+-----------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3INPUF.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:27 PM
|  Modified: Wednesday, December 31, 1997             4:37:04 PM
|
+------------------------------------------------------------------------------

000001 ; ###########################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3INPUF.TEXT
000004 ; ###########################################################################################
000005
000006                  PAGE
000007                  SBTL      "INPUT AND READ CODE"
000008 TRMNOK:          LDA       INPFLG
000009                  BEQ       TRMNO1                    ;TRY AGAIN ON INPUT
000010                  BPL       SNERR4
000011                  LDA       DATLIN
000012                  LDY       DATLIN+1                  ;GET DATA LINE NUMBER.
000013                  STA       CURLIN
000014                  STY       CURLIN+1                  ;MAKE IT CURRENT LINE.
000015 SNERR4:          JMP       SNERR
000016 TRMNO2:          PLA
000017 TRMNO1           BIT       FILNO                     ;INPUT #?
000018                  BMI       *+5
000019                  JMP       TMERR
000020                  BIT       ERRFLG                    ;ON ERR IN EFFECT?
000021                  BPL       DOAGIN                    ;NO.
000022                  LDY       CURLIN+1
000023                  INY
000024                  BEQ       DOAGIN
000025                  LDX       #254                      ;ERROR CODE IS 254 FOR BAD INPUT.
000026                  JMP       ERROR
000027 DOAGIN:          EQU       *
000028                  LDA       #>TRYAGN
000029                  LDX       #0
000030                  LDY       #<TRYAGN
000031                  JSR       STROUT                    ;PRINT RETRY MSG
000032                  JSR       DOAG2                     ;RESTORE POINTERS
000033                  PLA                                 ;DON'T GO BACK TO NEWSTT
000034                  PLA                                 ;  but rather NWSTT which doesn't check
000035                  JMP       NWSTT                     ;CHECK FOR A SEPERATOR.
000036 DOAG2            LDA       OLDTXT
000037                  LDY       OLDTXT+1                  ;POINT AT START
000038                  LDX       OLDTXTB
000039                  STX       TXTPTRB
000040                  STA       TXTPTR
000041                  STY       TXTPTR+1                  ;OF THIS CURRENT LINE.
000042 GTRTS            RTS
000043 ;
000044 ; Procedure: GET
000045 ; Function: Fetch a single byte from the current INPUT device.  No echo..echo..
000046 GET:             JSR       ERRDIR                    ;Illegal in Direct Mode
000047                  LDX       INFLNO                    ;Get INPUT file #
000048                  BNE       *+5
000049                  LDX       SLINTB+1
000050                  STX       GETREF
000051                  CMP       #'#'                      ;File? or from Keyboard?
000052                  BNE       GOTREF
000053                  JSR       GTFLNO                    ;File! Get file number
000054                  STA       GETREF
000055                  JSR       CHKSMC
000056 GOTREF           LDX       #>BUF+1
000057                  LDY       #<BUF+1                   ;POINT TO 0.
000058                  LDA       #$80
000059                  STA       QUOTE
000060                  LDA       #0                        ;TO STUFF AND TO POINT.
000061                  STA       YSAVE
000062                  STA       BUF+1
000063                  STA       BUF
000064                  LDA       #64                       ;TURN ON V-BIT.
000065                  JMP       INPCO1                    ;DO THE GET.
000066 INPUT            JSR       ERRDIR                    ;NOT DIRECT NOW!
000067                  JSR       CHRGOT
000068                  CMP       #34                       ;A QUOTE?
000069                  BNE       NOTQT0                    ;NO MESSA.
000070                  JSR       STRTXT                    ;LITERALIZE THE STRING IN TEXT
000071                  JSR       CHRGOT                    ;WHAT CHARACTER AFTER THE MESSAGE?
000072                  CMP       #$2C                      ;COMMAS ARE OK NOW.
```

```
000073                BEQ       INPAOK
000074                LDA       #59
000075 INPAOK         JSR       SYNCHR                ;MUST END WITH SEMICOLON.(OR COMMA)
000076                JSR       STRPRT                ;PRINT IT OUT.
000077                JMP       NOTQTII
000078 NOTQT0         CMP       #'#'                  ;TIME FOR A DISK INPUT?
000079                BNE       NOTQTI                ;NO, PROCESS NORMAL PRINT.
000080                ROR       IOFLG
000081                JSR       FILNUM                ;GET THE FILE NUMBER
000082                JSR       CHRGOT
000083                CMP       #$3B
000084                BNE       GTRTS                 ;IF NOT A SEMI COLON, GO HOME
000085                LDA       SVFLNO
000086                STA       FILNO
000087                JSR       CHRGET
000088                JSR       DSKLIN                ;GET A LINE OF INPUT FROM THE DISK
000089                LDA       #44                   ;STUFF A COMMA BEFORE THE LINE
000090                STA       BUF-1
000091                JMP       INPCON                ;AND CONTINUE...
000092 NOTQTI:        JSR       OUTQST                ;PRINT A ? FOR INPUT
000093 NOTQTII        LDX       #0                    ;SPECIFY NORMAL TERMINATORS
000094                STX       QUOTE                 ;FOR LATER.
000095                DEX
000096                STX       FILNO                 ;NOT DISK INPUT.
000097                LDA       #44                   ;GET COMMA.
000098                STA       BUF-1
000099                JSR       INLIN                 ;INPUT A LINE OF TEXT.
000100                LDA       BUF                   ;ANYTHING INPUT?
000101                CMP       #$03                  ;CONTROL-C AT FRONT OF LINE?
000102                BNE       INPCON                ;YES, CONTINUE
000103                JSR       DOAG2                 ;PRESERVE POINTERS SO WE CAN CONTINUE
000104                LDA       #3                    ;TELL WE HAD A CONTROL-C
000105                JMP       ISCNTC2
000106 QINLIN:        EQU       *
000107                JSR       OUTQST
000108                JMP       INLIN
000109 READ:          LDX       DATPTRB
000110                STX       YSAVE
000111                LDX       DATPTR
000112                LDY       DATPTR+1              ;GET LAST DATA LOCATION.
000113                CMP       #'#'                  ;TIME TO READ DATA FROM THE DISK?
000114                BNE       READON                ;NO.
000115                JMP       DREAD
000116 READON         LDA       #0
000117                STA       QUOTE
000118                LDA       #$98                  ;FOR STUFF..
000119                DFB       44                    ;SKIP OVER LDA #0 OPERATION.
000120 INPCON:        LDA       #0
000121 INPCO1:        STA       INPFLG                ;STORE THE FLAG.
000122                STX       INPPTR
000123                LDX       YSAVE
000124                STX       INPPTRB
000125                STY       INPPTR+1
000126 INLOOP:        JSR       MYPTRGET              ;READ VARIABLE LIST.
000127                BIT       INPFLG                ;IS THIS AN INPUT STATEMENT?
000128                BMI       INLP2                 ;NO, A READ STATEMENT.
000129                JSR       CHRGOT                ;IS THIS THE LAST VAR IN THE LIST?
000130                BNE       INLP2                 ;NO
000131                LDA       #$80                  ;SPECIFY THAT THERE ARE NO TERMINATORS
000132                STA       QUOTE
000133 ;RETURNS PNTR TOP VAR IN VARPNT.
000134 INLP2          JSR       SVTXT                 ;SAVE THE TEXT POINTER IN VARTXT.
000135                LDX       INPPTR
000136                LDY       INPPTR+1
000137                LDA       INPPTRB
000138                STA       TXTPTRB
000139                STX       TXTPTR
000140                STY       TXTPTR+1
000141                LDY       #0                    ;SEE IF CHAR IS EOL
000142                LDA       (TXTPTR),Y
000143                BNE       DATBK1
000144                BIT       INPFLG
000145                BVC       QDATA
000146                JSR       DOAGET                ;JUST A SINGLE CHAR
000147                LDX       #>KEYSAVE
000148                LDY       #<KEYSAVE
000149                STX       STRNG1
000150                STY       STRNG1+1
000151                LDA       #0
000152                STA       STRNG1B
```

```
000153                LDA       VALTYP
000154                PHA
000155                LDY       RNDGOT
000156                JSR       STRCP
000157                PLA
000158                STA       VALTYP
000159                JSR       CHANGTP                 ;CHANGE TYPE TO WHAT IT SHOULD BE.
000160                JSR       LETP2                   ;DO THE ASSIGNMENT.
000161                JMP       STRDN2                  ;LOOP.
000162 QDATA:          BMI       DTLP1                   ;SEARCH FOR ANOTHER DATA STATEMENT.
000163                BIT       FILNO                   ;IS THIS A FILE INPUT?
000164                BMI       QD11
000165                JSR       DSKLIN                  ;INPUT A LINE FROM THE DISK
000166                JMP       DATBK
000167 QD11            JSR       OUTQST
000168                JSR       QINLIN                  ;GET ANOTHER LINE.
000169 DATBK:          STX       TXTPTR
000170                LDA       INPPTRB
000171                STA       TXTPTRB
000172                STY       TXTPTR+1                ;SET FOR 'CHRGET'.
000173 DATBK1:         LDY       #1
000174                JSR       ADDON
000175                BIT       VALTYP                  ;GET VALUE TYPE.
000176                BPL       NUMINS                  ;INPUT A NUMBER IF NUMERIC.
000177                BIT       INPFLG                  ;GET?
000178                BVC       SETQUT                  ;NO, GO SET QUOTE.
000179                INX
000180                STX       TXTPTR
000181 NOTERMS         LDA       #0                      ;ZERO TERMINATORS.
000182                STA       CHARAC
000183                BEQ       RESETC
000184 DTLP1           JMP       DATLOP
000185 SETQUT          LDY       #0
000186                LDA       (TXTPTR),Y
000187                STA       CHARAC                  ;ASSUME QUOTED STRING
000188                BIT       QUOTE                   ;DO WE TAKE ANYTHING?
000189                BMI       NOTERMS                 ;IF SO, NO TERMINATORS!
000190                CMP       #34                     ;TERMINATORS OK?
000191                BEQ       NOWGET                  ;YES.
000192                LDA       #44                     ;COMMA.
000193                STA       CHARAC                  ;ONLY STOP ON COMMAS
000194 RESETC:         CLC
000195 NOWGET:         STA       ENDCHR
000196                LDA       TXTPTR
000197                LDX       TXTPTRB
000198                LDY       TXTPTR+1
000199                ADC       #0                      ;C IS SET PROPERLY ABOVE.
000200                BCC       NOWGE1
000201                INY
000202                CPY       #MAXPG
000203                BCC       *+5
000204                INX
000205                LDY       #MINPG
000206 NOWGE1:         JSR       STRLT2                  ;MAKE A STRING DESCRIPTOR FOR VALUE
000207                JSR       ST2TXT                  ;SET TEXT POINTER.
000208                JSR       INPCOM                  ;DO ASSIGNMENT.
000209                JMP       STRDN2
000210 NUMINS:         PHA
000211                LDA       BUF                     ;BLANK INPUT?
000212                BEQ       MAYBAD
000213 NUMINS2:        PLA
000214                BVS       NUMBCD
000215                JSR       FIN                     ;GET VALUE.
000216                BIT       INPFLG                  ;WATCH FOR GET'S!
000217                BVS       NUMINS3
000218                BMI       NUMINS3
000219                PHA
000220                LDA       ANYNUM
000221                BMI       MAYBAD
000222                PLA
000223 NUMINS3         LDA       INTFLG                  ;SET CODES ON FLAG.
000224                JSR       QINTGR                  ;GO DECIDE ON FLOAT.
000225 STRDN2:         JSR       CHRGOT                  ;READ LAST CHARACTER.
000226                BEQ       TRMOK                   ;':' OR EOL IS OK.
000227                CMP       #44                     ;A COMMA?
000228                BEQ       *+5
000229                JMP       TRMNOK
000230 TRMOK:          LDA       TXTPTR
000231                LDX       TXTPTRB
000232                LDY       TXTPTR+1
```

```
000233                  STA       INPPTR
000234                  STX       INPPTRB
000235                  STY       INPPTR+1                ;SAVE FOR MORE READS.
000236                  JSR       RSTTXT                  ;RESTORE THE TEXT POINTER FROM VARTEXT.
000237                  JSR       CHRGOT                  ;LOOK AT LAST VARIABLE LIST CHARACTER.
000238                  BEQ       VAREND                  ;THAT'S THE END OF THE LIST.
000239                  JSR       CHKCOM                  ;NOT END. CHECK FOR COMMA.
000240                  JMP       INLOOP
000241 NUMBCD:          JSR       LINP
000242                  JSR       BMOVF1
000243                  JMP       STRDN2
000244 MAYBAD:          LSR       ANYNUM
000245                  LDA       INPFLG
000246                  BNE       NUMINS2
000247                  JMP       TRMNO2
000248 ; SUBROUTINE TO FIND DATA
000249 DATLOP:          JSR       DATAN                   ;SKIP SOME TEXT.
000250                  INY                               ;ADVANCE ONE AT LEAST
000251                  TAX                               ;END OF LINE?
000252                  BNE       NOWLIN                  ;SHO AIN'T.
000253                  LDX       #ERROD                  ;YES = 'NO DATA' ERROR.
000254                  LDA       (TXTPTR),Y
000255                  BNE       *+5
000256                  JMP       ERROR
000257                  INY
000258                  LDA       (TXTPTR),Y              ;GET HIGH BYTE OF LINE NUMBER.
000259                  STA       DATLIN
000260                  INY
000261                  LDA       (TXTPTR),Y              ;GET LOW BYTE.
000262                  INY
000263                  STA       DATLIN+1
000264 NOWLIN:          LDA       (TXTPTR),Y              ;HOW IS IT?
000265                  TAX
000266                  JSR       ADDON                   ;ADD Y TO TXTPTR.
000267                  CPX       #DATATK                 ;IS IT A 'DATA' STATEMENT.
000268                  BNE       DATLOP                  ;NOT QUITE RIGHT. KEEP LOOKING.
000269                  JSR       CHRGET
000270                  JSR       DECTPT
000271                  JMP       DATBK1
000272 VAREND:          LDA       #0                      ;RESET TERMINATOR FLAG
000273                  STA       QUOTE
000274                  LDA       INPPTR
000275                  LDY       INPPTR+1                ;PUT AWAY A NEW DATA PNTR MAYBE.
000276                  LDX       INPPTRB
000277                  BIT       INPFLG
000278                  BPL       VARY0
000279                  JMP       RESFIN
000280 VARY0:           LDY       #0
000281                  LDA       (INPPTR),Y              ;LAST DATA CHR COULD HAVE BEEN
000282 ;COMMA OR COLON BUT SHOULD BE NULL.
000283                  BEQ       INPRTS                  ;IT IS NUL
000284                  JSR       INPRTS                  ;CLOSE UP OUTPUT FILE.
000285                  BIT       ERRFLG                  ;ERROR TRAPPING ON?
000286                  BPL       INPVAR0                 ;NOPE.
000287                  LDX       #253                    ;EXTRA IGNORED ERROR
000288                  LDY       CURLIN+1
000289                  INY
000290                  BEQ       INPVAR0
000291                  JMP       ERROR                   ;REPORT THE ERROR
000292 INPVAR0          LDA       #>EXIGNT
000293                  LDX       #0
000294                  LDY       #<EXIGNT
000295                  JMP       ERRFIN                  ;PRINT ERROR, IN LINNUM.
000296 INPRTS:          LDA       FILNO+1                 ;RETURN I/O.
000297                  STA       FILNO
000298                  RTS
000299 SVTXT            LDA       TXTPTR                  ;SAVE THE TXTPTR IN VARTXT.
000300                  LDY       TXTPTR+1
000301                  LDX       TXTPTRB
000302                  STX       VARTXTB
000303                  STA       VARTXT
000304                  STY       VARTXT+1
000305                  RTS
000306 RSTTXT           LDA       VARTXT                  ;RESTORE THE TXTPTR FROM VARTXT.
000307                  STA       TXTPTR
000308                  LDA       VARTXT+1
000309                  STA       TXTPTR+1
000310                  LDA       VARTXTB
000311                  STA       TXTPTRB
000312                  RTS
```

```
000313 EXIGNT:         ASC       '?EXTRA IGNORED'
000314                 DFB       0
000315 TRYAGN:         ASC       '?REENTER'
000316                 DFB       13,10,0
000317 CHANGTP         BIT       VALTYP
000318                 BMI       ISSTRIN
000319                 BVS       ISLONG
000320                 JMP       VAL
000321 ISSTRIN         RTS                                 ;ALREADY THE RIGHT TYPE.
000322 ISLONG          JMP       STR2LNG
000323                 SBTL      "NEXT CODE"
000324 ; A FOR entry on the stack has the following format: (in PULL order)
000325 ;   FORTK - 1 Byte
000326 ;   Temp FOR counter - 1 Byte (Currently unused)
000327 ;   Pointer to the loop variable - 2 Bytes
000328 ;   Step value - 5 Bytes (Sign of step is irrelevant)
000329 ;   Sign of Step - 1 Byte (0-$7F Positive step, $80-$FF Negative step)
000330 ;   Limit value (Packed) - 5 Bytes
000331 ;   Line # of the FOR statement - 2 Bytes
000332 ;   Text pointer to end of the FOR Statement - 3 Bytes
000333 ; TOTAL: 20 Bytes
000334 NEXT:          BNE       GETFOR
000335                 LDY       #0                        ;WITHOUT ARG CALL 'FNDFOR' WITH
000336                 STY       FORPNT+1
000337                 STY       FORPNT
000338                 BEQ       STXFOR                    ;FORPNT=0.
000339 GETFOR:         JSR       MYPTRGET                  ;GET A POINTER TO LOOP VARIABLE
000340                 JSR       RELPTR
000341                 LDA       ISARA
000342                 ROL       A
000343                 LDA       INTFLG
000344                 ADC       #0                        ;HIGH BIT OF ISARA INTO LOW BIT.
000345                 STA       TEMPFOR
000346                 LDA       FORPNT+1
000347                 LDY       FORPNTB
000348                 JSR       FIXAY
000349                 STA       FORPNT+1
000350 STXFOR          JSR       FNDFOR                    ;FIND THE MATCHING ENTRY IF ANY.
000351                 BEQ       HAVFOR
000352                 LDX       #ERRNF                    ;'NEXT WITHOUT FOR'.
000353                 JMP       ERROR
000354 HAVFOR:         LDA       #$FE
000355                 STA       FORPNTB
000356                 TXS                                 ;SETUP STACK. CHOP FIRST.
000357                 TXA
000358                 CLC
000359                 ADC       #5                        ;POINT TO INCREMENT.
000360                 TAY                                 ;SET LO ADDR OF THING TO MOVE.
000361                 ADC       #6
000362                 STA       INDEX2
000363                 TYA
000364                 LDY       #1                        ;SET HI ADDR OF THING TO MOVE.
000365                 LDX       #0
000366                 JSR       MOVFM                     ;GET QUAN INTO FAC.
000367                 TSX
000368                 LDA       257+7+2,X                 ;SET SIGN CORRECTLY.
000369                 STA       FACSGN
000370                 JSR       RELPTR
000371                 BIT       INTFLG
000372                 BPL       HAVFLT
000373                 JSR       MOVAF                     ;SAVE FAC
000374                 LDA       FORPNT
000375                 STA       FACMO
000376                 LDA       FORPNT+1
000377                 STA       FACMO+1
000378                 LDA       FORPNTB
000379                 STA       FACMOB
000380                 JSR       GOOO2                     ;GET INTEGER POINTED TO BY FACMO INTO FAC.
000381                 JMP       HAVSUM
000382 HAVFLT          LDY       FORPNT+1
000383                 LDX       FORPNTB
000384                 LDA       FORPNT
000385                 JSR       CONUPK                    ;ADD INC TO LO VARIABLE.
000386 HAVSUM          LDA       ARGSGN                    ;MUST SET UP ARISGN BEFORE FADD.
000387                 EOR       FACSGN
000388                 STA       ARISGN
000389                 LDA       FACEXP                    ;SET UP Z FLAG TOO.
000390                 JSR       FADDT
000391                 LDY       #1
000392                 JSR       FCOMPN                    ;COMPARE FAC WITH UPPER VALUE.
```

```
000393                  PHA                             ;SAVE RESULT OF COMPARE.
000394                  BIT       INTFLG
000395                  JSR       QINTGR                ;STORE THE RESULT SAME AS LET DOES.
000396                  PLA                             ;RESTORE RESULT OF COMPARE.
000397                  TSX
000398                  SEC
000399                  SBC       257+7+2,X             ;SUBTRACT SIGN OF INC FROM SIGN OF
000400 ;OF (CURRENT VALUE-FINAL VALUE).
000401                  BEQ       LOOPDN                ;IF SIGN (FINAL-CURRENT)-SIGN STEP=0
000402 ;THEN LOOP IS DONE.
000403                  LDA       257+12+3,X
000404                  STA       CURLIN                ;STORE LINE NUMBER OF 'FOR' STATEMENT.
000405                  LDA       257+13+3,X
000406                  STA       CURLIN+1
000407                  LDA       257+16+3,X
000408                  CLC
000409                  ADC       TXTTAB                ;SINCE A RELATIVE POINTER WAS PUSHED
000410                  STA       TXTPTR                ;STORE TEXT PNTR INTO 'FOR' STATEMENT.
000411                  LDA       257+15+3,X
000412                  ADC       TXTTAB+1
000413                  LDY       TXTTABB
000414                  JSR       FIXADC
000415                  STA       TXTPTR+1
000416                  TYA
000417                  ADC       257+14+3,X            ;WONDERFUL CODE!
000418                  STA       TXTPTRB
000419 NEWSGO:          JMP       NEWSTT                ;PROCESS NEXT STATEMENT.
000420 LOOPDN:          TXA
000421                  ADC       #16+3                 ;ADDS 16 WITH CARRY.
000422                  TAX
000423                  TXS                             ;NEW STACK PNTR.
000424                  JSR       CHRGOT
000425                  CMP       #44                   ;COMMA AT END?
000426                  BNE       NEWSGO
000427                  JSR       CHRGET
000428                  JSR       GETFOR                ;DO NEXT BUT DON'T ALLOW BLANK VARIABLE
000429 ;PNTR. VARPNT IS THE STK PNTR WHICH
000430 ;NEVER MATCHES ANY POINTER.
000431 ;JSR TO PUT ON DUMMY NEWSTT ADDR.
000432                  PAGE
000433                  SBTL      "FORMULA EVALUATION CODE."
000434 ; THESE ROUTINES CHECK FOR CERTAIN 'VALTYP'.
000435 ; C IS NOT PRESERVED.
000436 FRMNUM:          LDA       #0
000437                  STA       VALTYP
000438                  JSR       DOPAR
000439 CHKNUM:          CLC
000440                  DFB       36
000441 CHKSTR:          SEC                             ;SET CARRY.
000442                  BIT       VALTYP                ;WILL NOT F UP 'VALTYP'.
000443                  BMI       DOCSTR
000444                  BVS       CHKERR                ;CAN'T BE DOUBLE PRECESION.
000445                  BCS       CHKERR
000446 CHKOK:           RTS
000447                  BVC       CHKERR
000448 DOCSTR:          BCS       CHKOK
000449 CHKERR:          LDX       #ERRTM
000450                  JMP       ERROR
000451 ;
000452 ; Procedure: FRMEVL (Formula Evaluator)
000453 ; Function:  Formula evaluation
000454 ; On Entry:  TXTPTR points to first character of the formula
000455 ; On Exit:   Acc unknown
000456 ;            TXTPTR points to the terminator
000457 ;            Result of evaluation is left in FAC
000458 FRMEVL:          LDA       #0
000459                  STA       NAMPNT
000460                  STA       NOUNPT
000461                  STA       VRBSTK+1
000462                  STA       STRFLG
000463                  STA       INTFLG
000464                  LDA       #2
000465                  STA       VRBPT
000466 FRMEVL1A:        LDA       VALTYP
000467                  PHA
000468                  JSR       DECTPT
000469 FRMEVL1          JSR       EVAL
000470 EVALRET          EQU       *-1                   ;(B3INVKE uses this to check a Call Location)
000471                  JSR       CHRGOT
000472                  LDY       #0
```

```
000473                  LDX       #NUMOPS
000474                  CMP       #$80
000475                  BCC       FNDOPR
000476                  CMP       #$FF
000477                  BNE       NOOPR
000478                  INY
000479                  LDA       (TXTPTR),Y
000480 FNDOPR:          CMP       OPTAB-1,X
000481                  BEQ       FOUNDOP
000482                  DEX
000483                  BNE       FNDOPR
000484 NOOPR            LDX       #NUMOPS                 ;NO OPERATOR-END OF EXPR
000485                  DEY
000486 ;OPERATOR NUMBER IN X... GET PRECEDENCE.
000487 FOUNDOP:         PHA
000488                  CLC
000489                  TXA
000490                  ADC       #RELNOT                 ;MAP UP PAST COMBINED RELATIONALS.
000491                  TAX
000492                  PLA
000493                  DEY                               ;WAS IT AN ESCAPE TOKEN?
000494                  BMI       FOP2                    ;NO.
000495                  JSR       CHRGET
000496 FOP2:            CPX       #RELOPS+1
000497                  BCS       NOTREL
000498                  LDX       #0
000499                  STX       DOMASK
000500 GTRLOP:          CMP       #'>'+1
000501                  BCS       NOGTRL                  ;MAP RELATIONALS INTO BIT MAP:
000502                  CMP       #'<'                    ;    >  001.
000503                  BCC       NOGTRL                  ;    =  010.
000504                  EOR       #$3F                    ;    <  100.
000505                  CMP       #3
000506                  ADC       #0                      ;ADD IN CARRY.
000507                  EOR       DOMASK                  ;DOMASK MUST GET BIGGER
000508                  CMP       DOMASK                  ;OR ELSE HE HAD TWO OPS SAME.
000509                  BCC       SNERR5                  ;GIVE SYNERR IN THAT CASE.
000510                  STA       DOMASK
000511                  JSR       CHRGET
000512                  JMP       GTRLOP
000513 NOGTRL:          JSR       DECTPT                  ;BACK UP TO PNT AT LAST
000514                  LDA       DOMASK
000515 FRMEVLZ:         TAX
000516 NOTREL:          LDY       VRBPT
000517                  LDA       VRBSTK-1,Y              ;GET TYPE,PRECIDENCE.
000518                  AND       #$3F                    ;A=PRECIDENCE.
000519                  CMP       PRECTB-1,X              ;LAST OPERATOR HIGHER PRECIDENCE?
000520                  BCS       DOLAST                  ;YES, GO DO IT!
000521 ;THE CURRENT OPERATOR IS OF HIGHER PRECEDENCE
000522 ; THAN THE PREVIOUS ONE (A+B*C WITH * AS CURRENT OP)
000523 ; PUSH THE FAC ONTO THE NOUN STACK, AND THE OPERATOR
000524 ; ONTO THE VERB STACK.
000525                  LDA       VALTYP                  ;GET TYPE OF OPERAND.
000526                  AND       #$C0
000527                  ORA       PRECTB-1,X              ;TYPE, PRECIDENCE IN SAME BYTE.
000528                  STA       VRBSTK+1,Y              ;PUT ON PRECEDENCE
000529                  TXA
000530                  STA       VRBSTK,Y                ;AND OPERATOR RIIGHT BEFORE IT
000531                  INY
000532                  INY
000533                  CMP       #ENDOP                  ;End of expression if A=0.
000534                  BNE       NT1                     ;NO, DON'T CHECK PREVIOUS OPERATOR.
000535                  LDA       VRBSTK-3,Y
000536                  BEQ       EXPRDN2                 ;WE'RE DONE NOW.
000537 NT1              STY       VRBPT                   ;UPDATE VERB POINTER.
000538                  CPY       #$20                    ;TOO COMPLEX AN EXPRESSION?
000539                  BCC       STKOK
000540                  JMP       CMPLXERR
000541 STKOK            LDA       NOUNPT
000542                  ADC       #9                      ;CARRY IS CLEAR. 9 BYTES.
000543                  STA       NOUNPT                  ;NOUNPT UPDATED FOR ONE 12 BYTE ENTRY.
000544                  TAY
000545                  LDX       #7
000546 ; Push FAC onto NOUN Stack now.
000547 PSHFAC:          LDA       FACEXP,X
000548                  STA       NOUNSTK-1,Y
000549                  DEY
000550                  DEX
000551                  BPL       PSHFAC                  ;ALL 8 BYTES!
000552                  LDA       FACMOB                  ;FOR CAT.
```

```
000553                  STA       NOUNSTK-1,Y
000554                  JMP       FRMEVL1
000555 SNERR5           JMP       SNERR
000556 EXPRDN2          PLA                               ;REMEMBER WHAT THE FORMULA SHOULD HAVE BEEN?
000557                  CMP       #$20
000558                  BEQ       EXPRDN3
000559                  EOR       VALTYP                  ;WAS THAT WHAT WE GOT?
000560                  BNE       CANTFIX                 ;NO, WELL, WE'RE DONE THEN.
000561 EXPRDN3          RTS
000562 ;THE CURRENT OPERATOR IS OF LOWER PRECEDENCE
000563 ; THAN THE PREVIOUS, SO EXECUTE THE PREVIOUS OPERATOR.
000564 ; FIRST PULL ARG OFF STACK...
000565 ;
000566 DOLAST:          TXA                               ;PUSH THIS OPERATOR SO CAN LOOP BACK.
000567                  PHA                               ;ONLY SAVE OPERATOR - WILL REGET PRECEDENCE.
000568                  LDA       NOUNPT
000569                  TAY
000570                  SBC       #9                      ;PULL 9 BYTES OFF STACK.(CARRY IS SET)
000571                  STA       NOUNPT
000572                  LDX       #8
000573 PLLARG:          LDA       NOUNSTK-1,Y
000574                  STA       ARG-1,X
000575                  DEY
000576                  DEX
000577                  BNE       PLLARG
000578                  LDA       NOUNSTK-1,Y
000579                  STA       ARGMOB                  ;FOR CAT.
000580                  LDY       VRBPT
000581                  DEY
000582                  DEY                               ;BACK UP POINTER....
000583                  STY       VRBPT
000584                  LDA       VRBSTK+1,Y              ;GET PRECEDENCE AND TYPE.
000585                  AND       #$C0                    ;GOT TYPE.
000586                  STA       TEMP                    ;SAVE IN TEMP AREA.
000587                  LDA       VALTYP                  ;GET PREVIOUS TYPE.
000588                  AND       #$C0
000589                  EOR       TEMP                    ;ARE THEY THE SAME?
000590                  BEQ       GETOP                   ;IF SO, WE'RE COOL.
000591 CANTFIX          JMP       CHKERR                  ;TYPE MISMATCH ERROR.
000592 GETOP            LDA       VRBSTK,Y                ;GET OPERATOR...
000593                  STA       DOMASK                  ;GET OPERATOR MASK.
000594
000595 ; ##############################################################################################
000596 ; #   END OF FILE:  B3INPUF.TEXT
000597 ; #   LINES      :  588
000598 ; #   CHARACTERS :  27075
000599 ; ##############################################################################################
```

```
+-------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 599   CHARACTERS: 27627
|
+-------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3EVALG.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:25 PM
|  Modified: Wednesday, December 31, 1997             4:37:02 PM
|
+------------------------------------------------------------------------------

000001 ; ###########################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3EVALG.TEXT
000004 ; ###########################################################################################
000005
000006                  SEC
000007                  SBC       #6
000008                  CMP       #1
000009                  BPL       *+4
000010                  LDA       #1
000011                  ASL       A
000012                  TAX
000013                  BIT       VALTYP                    ;BUT IF NOT REAL LOOK OUT!
000014                  BMI       ASTRNG                    ;IT'S A STRING.
000015                  CLC
000016                  BVC       NOTDBLP                   ;IT'S NOT DBL PRECISION EITHER
000017                  ADC       #NUMDSP                   ;MAP UP INTO DBL
000018                  TAX
000019                  BCC       RDY2GO
000020 NOTDBLP:         EQU       *
000021                  LDA       ARGSGN
000022                  EOR       FACSGN
000023                  STA       ARISGN
000024 RDY2GO:          LDA       OPDSPT-2,X
000025                  STA       JMPER+1
000026                  LDA       OPDSPT-1,X
000027                  STA       JMPER+2
000028                  LDA       FACEXP
000029                  JSR       JMPER                     ;DO OPERATION...
000030 FRMEV3:          PLA                                 ;GET CURRENT OP BACK AND LOOP.
000031                  JMP       FRMEVLZ
000032 ASTRNG:          CMP       #NUMDSP                   ;AN ADD ON STRINGS
000033                  BEQ       PLUSOK                    ;IS OK.
000034                  CMP       #2*RELNUM                 ;IS THIS A RELATIONAL?
000035                  BEQ       NOTDBLP                   ;YES, GO DO IT. NOTE: CARRY SET INDICATES STRING.
000036 EVALER:          JMP       SNERR
000037 PLUSOK           JSR       CONCAT                    ;GO CONCATENATE THE STRINGS
000038                  JMP       FRMEV3                    ;GET BACK OPERATOR AND LOOP.
000039 ;  HERE IS THE EVAL ROUTINE - EVALUATE A SINGLE
000040 ;   VALUE.  IF A FORMULA IN PARENTHESES, CALL
000041 ;   FRMEVL RIGHT BACK!
000042 EVAL             JSR       CHRGET                    ;GET FIRST CHAR OF VALUE.
000043                  BCS       EVAL1                     ;IF A DIGIT, MUST BE A CONSTANT...
000044 ACONSTNT:        LDA       VALTYP                    ;CHECK DATA TYPE.
000045                  CMP       #$21
000046                  BCC       RLBNY                     ;VALTYP 0 MEANS GET FLOATING POINT.
000047                  JMP       LINP                      ;LONG INTEGER INPUT INTO FAC.
000048 RLBNY            LDA       #0
000049                  STA       VALTYP
000050                  JSR       FIN                       ;BINAY FLOATING POINT INPUT.
000051                  BIT       ANYNUM
000052                  BMI       EVALER
000053                  RTS
000054 EVAL1            CMP       #$FF                      ;HANDLE ESCAPE TOKENS
000055                  BEQ       PARC22                    ;(FUNCTIONS OR RESERVED WORDS)
000056                  CMP       #$80                      ;A RES. WORD?
000057                  BCS       PARC23
000058                  JSR       ISLETC                    ;A LETTER? IF SO, A VARIABLE.
000059                  BCC       NOVAR                     ;NO.
000060                  JMP       ISVAR
000061 NOVAR            CMP       #'.'                      ;LEADING CHARACTER OF CONSTANT?
000062                  BEQ       ACONSTNT
000063                  CMP       #'+'
000064                  BEQ       EVAL
000065                  CMP       #34                       ;A QUOTE? A STRING?
000066                  BNE       EVAL3
000067 STRTXT:          LDA       TXTPTR
000068                  LDX       TXTPTRB
000069                  LDY       TXTPTR+1
000070                  ADC       #0                        ;TO INC, ADD C=1.
000071                  BCC       STRTX2
000072                  INY
```

```
000073                CPY       #MAXPG
000074                BCC       *+5
000075                INX
000076                LDY       #MINPG
000077 STRTX2:        JSR       STRLIT                ;YES. PROCESS IT.
000078                JMP       ST2TXT
000079 NOTDO          JSR       DECTPT                ;DECRIMENT TXTPTR.
000080                JSR       EVAL                  ;CALL MYSELF!
000081                BIT       VALTYP                ;REAL OR DOUBLE PRECISION?
000082                BVC       *+5
000083                JMP       BCDTSTR
000084                LDA       FACEXP
000085                BEQ       RLONE
000086                LDY       #0
000087                DFB       44
000088 RLONE:         LDY       #1
000089                JMP       SNGFLT
000090 EVAL3:         CMP       #'-'
000091                BEQ       DOMIN
000092                JMP       PARCHK
000093 PARC22         JSR       CHRGET                ;EAT THE ESCAPE TOKEN
000094                TAX                             ;KEEP THIS BYTE.
000095                JSR       CHRGET                ;POINT PAST IT.
000096                TXA                             ;GOT THE BYTE BACK.
000097                JMP       ISFUN
000098 PARC23         CMP       #FRETK
000099                BCC       SNERR6
000100                CMP       #POPTKN
000101                BCS       SNERR6
000102                ASL       A
000103                TAX
000104                JSR       CHRGET
000105                LDA       RESTBL-FRETK-FRETK+256,X
000106                STA       JMPER+1
000107                LDA       RESTBL-FRETK-FRETK+257,X
000108                STA       JMPER+2
000109                JMP       JMPER
000110 VPOS           BRK
000111                DFB       SDSTAT                ;CALL SOS FOR CONSOLE STATUS
000112                DW        REDCUR                ;SPECIFYING A READ OF THE CURSOR POSITION.
000113                LDY       CURY
000114                RTS
000115 DOERRLN        LDA       ERRLIN+1
000116                LDY       ERRLIN
000117                STA       FACHO
000118                STY       FACHO+1
000119                LDX       #$90
000120                SEC
000121                JMP       FLOATC
000122 GIVOUTREC      LDY       OUTREC
000123                JMP       SNGFLT
000124 GIVINDENT      LDY       INDENT
000125                JMP       SNGFLT
000126 GIVKBD         LDY       KEYSAVE               ;SEE WHAT HE TYPED TO GET THE INTERRUPT.
000127                JMP       SNGFLT
000128 GIVEOF         LDY       EOFSV
000129                DFB       44
000130 GIVERR         LDY       ERRNUM
000131                JMP       SNGFLT
000132 BCDTSTR        LDA       #>FAC                 ;RETURN WITH A,FLAGS = OR OF ALL FAC BYTES.
000133                LDY       #<FAC
000134                JSR       LORALL
000135                BEQ       ISFALS
000136                JMP       LONGST0
000137 ISFALS         JMP       LONGST1
000138 DOMIN:         JSR       EVAL                  ;EVALUATE THE CONSTANT.
000139                BIT       VALTYP                ;TYPE = FLOATING?
000140                BVC       CHGSGN                ;YES, OK.
000141                JMP       LTWSCOMP
000142 CHGSGN         JMP       NEGOP
000143 SNERR6         JMP       SNERR
000144 ;WE ENCOUNTERED AN EXPRESSION IN PARENTHESES.
000145 ;GO EVALUATE IT.  PUT A 0 ON VERBSTACK TO PREVENT
000146 ;PROBLEMS.
000147 ;AN UPPER-BOUND.
000148 DOPAR          LDA       #0
000149                LDY       VRBPT
000150                INY
000151                STA       VRBSTK,Y              ;LAST PRCEDENCE 0
000152                INY
```

```
000153                STY       VRBPT                   ;STCK PNTR INCRMNTD.
000154                CPY       #$20
000155                BCS       CMPLXERR
000156                JSR       FRMEVL1A                ;BUT DEC TXTPTR FIRST
000157                DEC       VRBPT
000158                DEC       VRBPT                   ;POINT BELOW 'TOP OF STACK' CHARATCER
000159                RTS
000160 PARCHK:         JSR       CHKOPN                  ;ONLY POSSIBILITY LEFT IS
000161 PARCHK2         JSR       DOPAR
000162 ;RECURSIVELY EVALUATE THE FORMULA.
000163 CHKCLS:         LDA       #41                     ;CHECK FOR A RIGHT PAREN
000164                DFB       44
000165 CHKSMC:         LDA       #$3B                    ;A SEMICOLON.
000166                DFB       44
000167 CHKEQL:         LDA       #'='
000168                DFB       44
000169 CHKPND:         LDA       #'#'
000170                DFB       44
000171 CHKOPN:         LDA       #40
000172                DFB       44
000173 CHKCOM:         LDA       #44
000174 SYNCHR:         LDY       #0
000175                CMP       (TXTPTR),Y              ;CHARACTERS EQUAL?
000176                BNE       SNERR
000177                JMP       CHRGET
000178 CMPLXERR        LDX       #ERRST
000179                JMP       ERROR
000180 SNERR:          LDX       #ERRSN                  ;'SYNTAX ERROR'
000181                JMP       ERROR
000182 ISVAR:          LDX       #$FF                    ;ENTRY TO PTRGET THAT DOESN'T
                                                                 CREATE UNKNOWN ARRAYS.
000183                JSR       PTREVL
000184                LDX       VARPNTB
000185 ISVRET2         STA       FACMO
000186                STY       FACMO+1
000187                STX       FACMOB
000188                BIT       VALTYP
000189                BVC       GOOO                    ;STRING IS SET UP.
000190                BPL       DBLVAR                  ;BCD VAR
000191                LDX       #0
000192                STX       FACOV
000193                RTS
000194 DBLVAR:         JMP       DMOVFM                  ;DOUBLE MOVE MEMORY INTO FAC.
000195 DECTPT:         TXA                               ;SAVE THE X REGISTER
000196                PHA
000197                LDA       TXTPTR
000198                BNE       DECTP1                  ;NO CARRY UNLESS ZERO.
000199                DEC       TXTPTR+1                ;HIGH BYTE DECRIMENTED.
000200                LDX       TXTPTRB
000201                LDY       TXTPTR+1
000202                JSR       FIXYX
000203                STX       TXTPTRB
000204                STY       TXTPTR+1
000205 DECTP1          DEC       TXTPTR                  ;LOW BYTE DECRIMENTED.
000206                PLA                               ;RESTORE THE X REGISTER
000207                TAX
000208                RTS
000209 GOOO:           EQU       *
000210                BIT       INTFLG
000211                BPL       GOOOOO
000212 GOOO2:          LDY       #0
000213                LDA       (FACMO),Y               ;FETCH HIGH.
000214                TAX
000215                INY
000216                LDA       (FACMO),Y
000217                TAY                               ;PUT LOW IN Y.
000218                TXA                               ;GET HIGH IN A.
000219                JMP       GIVAYF                  ;FLOAT AND RETURN.
000220 GOOOOO:         EQU       *
000221                JMP       MOVFM                   ;MOVE ACTUAL VALUE IN.
000222 ISFUN:          CMP       #ONEFUN
000223                BCS       GOODFUN
000224                CMP       #FNTK
000225                BNE       *+5
000226                JMP       FNDOER                  ;USER DEFINED FUNCTION.
000227                CMP       #NOTTK
000228                BNE       SNERR
000229                JMP       NOTDO
000230 GOODFUN         ASL       A
000231                PHA
```

```
000232                TAX
000233                CPX       #2*LASNUM-256+1          ;IS IT PAST 'LASNUM'?
000234                BCC       OKNORM                   ;NO, MUST BE NORMAL FUNCTION.
000235 ; MOST FUNCTIONS TAKE A SINGLE ARGUMENT.
000236 ; THE RETURN ADDRESS OF THESE FUNCTIONS IS 'CHKNUM'
000237 ; WHICH ASCERTAINS THAT VALTYP=0  (NUMERIC).
000238 ; NORMAL FUNCTIONS THAT RETURN STRING RESULTS
000239 ; (E.G., CHR$) MUST POP OFF THAT RETURN ADDR AND
000240 ; RETURN DIRECTLY TO 'FRMEVL'.
000241 ; SO-CALLED 'FUNNY' FUNCTIONS CAN TAKE MORE THAN ONE ARG
000242 ; THE FIRST OF WHICH MUST BE STRING AND THE SECOND OF ICH
000243 ; MUST BE A NUMBER BETWEEN 0 AND 255.
000244 ; CLOSED PARENTHESIS MUST BE CHECKED AND RETURN IS DIRECT
000245 ; TO 'FRMEVL' WITH THE TEXT PNTR POINTING BEYOND THE ')'.
000246 ; THE POINTER TO THE DESCRIPTOR OF THE STRING ARGUMENT
000247 ; IS STORED ON THE STACK UNDERNEATH THE VALUE OF THE
000248 ; INTEGER ARGUMENT.
000249                LDA       #$FF
000250                STA       VALTYP
000251                JSR       DOPAR                    ;EAT OPEN PAREN AND FIRST ARG.
000252                JSR       CHKCOM                   ;TWO ARGS SO COMMA MUST DELIMIT.
000253                PLA                                ;GET NCTION NUMBER.
000254                TAX
000255                LDA       FACMOB
000256                PHA
000257                LDA       FACMO+1
000258                PHA
000259                LDA       FACMO
000260                PHA                                ;SAVE PNTR AT STRNG DESCPTR.
000261                TXA
000262                PHA                                ;RESAVE FUNCTION NUMBER.
000263 ;THIS MUST BE ON STACK SINCE RECURSIVE.
000264                CMP       #2*INSTRTK-256           ;INSTR FUNCTION?
000265                BNE       NORMFN2
000266                LDA       #$FF                     ;GET AN OTHER STRING
000267                STA       VALTYP
000268                JSR       DOPAR
000269                PLA                                ;FUNCTION NUMBER.
000270                TAX
000271                LDA       FACMOB
000272                PHA
000273                LDA       FACMO+1
000274                PHA
000275                LDA       FACMO
000276                PHA
000277                TXA
000278                PHA
000279                JSR       CHRGOT                   ;COMMA
000280                CMP       #','
000281                BEQ       NORMFUN
000282                LDX       #1
000283                BNE       *+8                      ;ALWAYS
000284 NORMFUN         JSR       CHRGET
000285 NORMFN2         JSR       GETBYT
000286                PLA                                ;GET FUNCTION NUMBER.
000287                TAY
000288                TXA
000289                PHA
000290                JMP       FINGO                    ;DISPATCH TO FUNCTION.
000291 OKNORM:         CPX       #2*LENTK-256             ;IF BIN ARGUMENT.
000292                BCC       ISABIN
000293                CPX       #2*CONVTK-256            ;IF STRING ARGUMENT.
000294                BCC       ISASTRNG
000295                LDA       #$20
000296                DFB       44
000297 ISASTRNG        LDA       #$FF
000298                DFB       44
000299 ISABIN          LDA       #0
000300                STA       VALTYP
000301                JSR       PARCHK2                  ;READ A FORMULA WITH A RIGHT PAREN
000302                PLA                                ;GET DISPATCH FUNCTION.
000303                TAY
000304 FINGO:          LDA       FUNDSP-ONEFUN-ONEFUN+256,Y ;MODIFY DISPATCH ADR
000305                STA       JMPER+1
000306                LDA       FUNDSP-ONEFUN-ONEFUN+257,Y
000307                STA       JMPER+2
000308                JMP       JMPER                    ;GO DO IT!
000309 OROP:           LDA       ARGEXP
000310                ORA       FACEXP
000311                BNE       GIVE1
```

```
000312 ANDOP:          LDA       ARGEXP
000313                 BEQ       GIVE0
000314                 LDA       FACEXP
000315                 BNE       GIVE1
000316 GIVE0:          LDY       #0
000317                 DFB       44
000318 GIVE1:          LDY       #1
000319                 JMP       SNGFLT
000320 ; TIME TO PERFORM A RELATIONAL OPERATOR.
000321 ; DOMASK CONTAINS THE BITS AS TO WHICH RELATIONAL
000322 ; OPERATOR IT WAS. CARRY BIT ON=STRING COMPARE.
000323 DOREL:          BIT       VALTYP                ;CHECK TYPE.
000324                 BMI       STRCMP                ;IT IS A STRING.
000325                 JSR       FCOMPARG
000326                 TAX
000327                 JMP       QCOMP
000328 STRCMP          JSR       FREFAC                ;FREE THE FACLO STRING.
000329                 PHP                             ;SAVE THE FREEUP STATUS.
000330                 STA       DSCTMP                ;SAVE FOR LATER.
000331                 STX       DSCTMP+1
000332                 STY       DSCTMP+1+1
000333                 LDA       INDEXB
000334                 STA       DSCTMPB
000335                 LDA       ARGMO
000336                 LDY       ARGMO+1               ;GET POINTER TO OTHER STRING.
000337                 LDX       ARGMOB
000338                 JSR       FRETMP                ;FREES FIRST DESC POINTER.
000339                 PHP                             ;SAVE THE FREEUP STATUS.
000340                 STX       ARGMO
000341                 STY       ARGMO+1
000342                 LDX       INDEXB
000343                 STX       ARGMOB
000344                 TAX                             ;COPY COUNT INTO X.
000345                 PHA                             ;SAVE THE COUNT FOR LATER.
000346                 SEC
000347                 SBC       DSCTMP                ;WHICH IS GREATER. IF 0, ALL SET UP.
000348                 BEQ       STASGN                ;JUST PUT SIGN OF DIFFEREE AWAY.
000349                 LDA       #1
000350                 BCC       STASGN                ;SIGN IS POSITIVE.
000351                 LDX       DSCTMP                ;LENGTH OF FAC IS SHORTER.
000352                 LDA       #$100-1               ;GET A MINUS 1 FOR NEGATIVES.
000353 STASGN:         STA       FACSGN                ;KEEP FOR LATER.
000354                 LDY       #255                  ;SET POINTER TO FIRST STRING. (ARG.)
000355                 INX                             ;TO LOOP PROPERLY.
000356 NXTCMP:         INY
000357                 DEX                             ;ANY CHARACTERS LEFT TO COMPARE?
000358                 BNE       GETCMP                ;NOT DONE YET.
000359                 LDX       FACSGN                ;USE SIGN OF LENGTH DIFFERENCE
000360 ;SINCE ALL CHARACTERS ARE THE SAME.
000361 QCOMP:          BMI       DOCMP                 ;C IS ALWAYS SET THEN.
000362                 CLC
000363                 BCC       DOCMP                 ;ALWAYS BRCH.
000364 GETCMP:         LDA       (ARGMO),Y             ;GET NEXT CHAR TO COMPARE.
000365                 CMP       (DSCTMP+1),Y          ;SAME?
000366                 BEQ       NXTCMP                ;YEP. TRY FURTHER.
000367                 LDX       #$100-1               ;SET A POSITIVE DIFFERENCE.
000368                 BCS       DOCMP                 ;PUT STACK BACK TOGETHER.
000369                 LDX       #1                    ;SET A NEGATIVE DIFFERENCE.
000370 DOCMP:          INX                             ;-1 TO 1, 0 TO 2, 1 TO 4.
000371                 TXA
000372                 ROL       A
000373                 AND       DOMASK
000374                 BEQ       GOFLOT
000375                 LDA       #1                    ;ALL OTHER RESULTS EVALUATE TO 1.
000376 GOFLOT          BIT       VALTYP                ;IS THIS A STRING COMPARE?
000377                 BPL       GFLOT1                ;NO, SO GO FLOAT THE RESULT.
000378                 STA       FOUR6                 ;TEMP SAVE FOR THE VALUE.
000379                 LDA       #0                    ;RESULT WILL BE NUMERIC.
000380                 STA       VALTYP
000381                 PLA                             ;GET BACK THE ARG STRING LENGTH.
000382                 LDX       ARGMO
000383                 LDY       ARGMO+1               ;REGS CONTAIN THE ARG STRING DESCRIPTOR.
000384                 STX       INDEX
000385                 STY       INDEX+1
000386                 LDX       ARGMOB
000387                 STX       INDEXB
000388                 PLP                             ;THE FREEUP STATUS.
000389                 JSR       FRENOW                ;ACTUALLY FREE THE MEMORY.
000390                 PLP                             ;FREE STATUS FROM THE FAC STRING.
000391                 JSR       FREFC1                ;FREE THE MEMORY UP.
```

```
000392                  LDA       FOUR6                    ;GET BACK THE RESULT OF THE COMPARE.
000393 GFLOT1           JMP       FLOAT                    ;FLOAT THE ONE BYTE RESULT INTO FAC.
000394
000395 ; ###############################################################################################
000396 ; #   END OF FILE:  B3EVALG.TEXT
000397 ; #   LINES      :  388
000398 ; #   CHARACTERS :  17567
000399 ; ###############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 399   CHARACTERS: 18119
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "STRUTILS.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:38 PM
|  Modified: Wednesday, December 31, 1997             4:37:16 PM
|
+------------------------------------------------------------------------------

000001 ; ###########################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  STRUTILS.TEXT
000004 ; ###########################################################################################
000005
000006 ;
000007 ; Procedure: GETSPA
000008 ; Purpose: To get space for character string.  (May force garbage
000009 ;          collection to take place.)
000010 ; On Entry:
000011 ;   A = # of characters (bytes)
000012 ; On Exit:
000013 ;   A unchanged
000014 ;   X, Y = pointer to space
000015 ;   FRESPC set up with pointer also
000016 ; On Error:
000017 ;    OUT OF STRING SPACE type error.
000018 GETSPA:         LSR       GARBFL                    ;SIGNAL NO GARB COLLECTION YET.
000019                 AND       #$FF                      ;SET Z FLAG
000020                 BEQ       STRRT2                    ;ALWAYS SPACE FOR A NULL
000021                 PHA
000022 TRYAG0          LDA       FRETOP
000023                 LDY       FRETOP+1
000024                 LDX       FRETOPB
000025                 SEC
000026                 SBC       #INFOSIZ
000027                 BCS       TRYAG1
000028                 JSR       FIXXY
000029 TRYAG1          STY       HIGHDS+1
000030                 STX       HIGHDSB                   ;POINTER TO INFO BYTES.
000031                 STA       HIGHDS
000032                 PLA
000033                 PHA
000034                 EOR       #255                      ;SUBTRACT A FROM HIGHDS.
000035                 SEC                                 ;ADD 1 TO COMPLETE NEGATION
000036                 ADC       HIGHDS
000037                 BCS       TRYAG4
000038                 JSR       FIXXY
000039 TRYAG4          CPX       STRENDB
000040                 BCC       GARBAG
000041                 BNE       TRYAG5
000042                 CPY       STREND+1                  ;COMPARE HIGH ORDERS.
000043                 BCC       GARBAG
000044                 BNE       TRYAG5
000045                 CMP       STREND                    ;COMPARE LOW ORDERS.
000046                 BCC       GARBAG
000047 TRYAG5          STA       FRESPC
000048                 STY       FRESPC+1
000049                 STX       FRESPCB
000050                 PHA
000051                 CLC
000052                 SBC       HIMEM                     ;STRING SPACE TOO BIG?
000053                 TYA
000054                 SBC       HIMEM+1
000055                 TXA
000056                 SBC       HIMEMB
000057                 CLC
000058                 ADC       #2                        ;2 BANKS=64K.
000059                 PLA
000060                 BCC       GARBAG                    ;GARBAGE COLLECT IF MORE THAN 64K OF STRINGS.
000061                 STX       FRETOPB
000062                 STA       FRETOP
000063                 STY       FRETOP+1                  ;SAVE NEW FRETOP.
000064                 LDX       FRESPC
000065                 LDY       FRESPC+1
000066                 PLA
000067 STRRT2          RTS                                 ;ALL DONE.
000068 ;
000069 ;  Here is the Garbage Collection Schtick.
000070 GARBAG:         LDX       #ERROM                    ;OUT OF MEMORY error
000071                 BIT       GARBFL                    ;ALREADY GARBAGE COLLECTED?
000072                 BPL       GARBA1
```

```
000073                LDA       #4
000074                JSR       EXPAND
000075 GARBA1         JSR       GARBA2                ;CRUNCH.
000076                LDA       #$80
000077                STA       GARBFL                ;CRUCHING HAS BEEN DONE.
000078                BNE       TRYAG0                ;ALWAYS
000079 GARBA2:        EQU       *                     ;START FROM TOP DOWN.
000080                LDA       #0
000081                STA       TEMP
000082                STA       CNTDIGS
000083                STA       INPFLG                ;EOR MASK.
000084                STA       GRBPNT                ;# OF BYTES TO OFFSET STARTS AT 0.
000085                STA       GRBPNT+1
000086                LDA       HIMEM                 ;OR VICA VERCA.
000087                LDX       HIMEMB
000088                LDY       HIMEM+1
000089                STA       HIGHDS                ;HIGH DESTINATION.
000090                STX       HIGHDSB
000091                STY       HIGHDS+1
000092                JMP       GSUBIN3               ;ALWAYS!
000093 GMOVPTR        LDY       #0
000094                LDA       (DECCNT),Y            ;GET ARRAY OR SIMPLE VARIABLE FLAG
000095                STA       TEMP
000096                INY
000097                LDA       (DECCNT),Y
000098                STA       HEADER                ;LOW BYTE RELATIVE ADDRESS.
000099                INY
000100                LDA       (DECCNT),Y
000101                STA       HEADER+1
000102                STA       ANYNUM                ;SAVE LENGTH IF FREE.
000103                LDA       TEMP                  ;0 IFF FREE.
000104                BEQ       SUBYSAV               ;IF THIS HUNK FREE.
000105                EOR       INPFLG                ;EOR WITH MASK.
000106                AND       #01
000107                STA       CNTDIGS               ;SET IF FIRST NON-FREE.
000108                LDA       #1
000109                STA       INPFLG                ;NON-FREE HUNK SETS MASK.
000110                LDA       TEMP
000111                CMP       #TEMPTYP              ;IS THIS A TEMP?
000112                BNE       *+5
000113                JMP       GFINDTEMP             ;MUST BE A TEMP. DESCR.
000114                CMP       #ARYTYP               ;ARRAY?
000115                BCC       GRELSIM               ;NO, MUST BE SIMPLE.
000116                LDA       ARYTAB+1
000117                PHA
000118                LDA       ARYTABB
000119                PHA
000120                LDA       ARYTAB
000121                BCS       GARBREL               ;ALWAYS.
000122 GRELSIM        LDA       VARTAB+1
000123                PHA
000124                LDA       VARTABB
000125                PHA
000126                LDA       VARTAB
000127 GARBREL        SEC
000128                SBC       HEADER                ;HEADER=BASE ADDR. - REL ADDR.
000129                STA       HEADER
000130                PLA
000131                TAY
000132                PLA
000133                LDX       HEADER+1
000134                JSR       FIXYAX
000135                STA       HEADER+1              ;NOW POINTS TO DESCRIPTOR.
000136                TYA
000137                SBC       #$FE                  ;IN 64K SPACE.
000138                STA       HEADERB
000139                LDY       #0
000140                LDA       (HEADER),Y            ;BYTE 1 OF DESCRIPTOR
000141                STA       ANYNUM                ;STRING LENGTH.
000142 GRBCMP2        LDY       #1
000143                SEC
000144                LDA       (HEADER),Y            ;ADJUST DESCRIPTOR'S POINTER
000145                SBC       GRBPNT
000146                STA       (HEADER),Y            ;BY OFFSET (GRBPNT).
000147                INY
000148                LDA       (HEADER),Y
000149                SBC       GRBPNT+1
000150                STA       (HEADER),Y
000151                BCS       SUBYS2                ;ALWAYS
000152 SUBYSAV        LDA       GRBPNT
```

```
000153                  LDY       GRBPNT+1
000154                  CLC
000155                  ADC       ANYNUM                  ;LENGTH.
000156                  BCC       *+3
000157                  INY
000158                  CLC
000159                  ADC       #INFOSIZ
000160                  BCC       *+3
000161                  INY
000162                  STA       GRBPNT
000163                  STY       GRBPNT+1
000164                  LDA       INPFLG
000165                  ORA       TEMP
000166                  STA       CNTDIGS
000167                  LDA       #0
000168                  STA       INPFLG
000169 SUBYS2           LDA       DECCNT
000170                  LDY       DECCNT+1
000171                  LDX       DECCNTB
000172                  SEC
000173                  SBC       ANYNUM                  ;SUBTRACT LENGTH OF STRING FROM POINTER.
000174                  BCS       GSUBINF
000175                  JSR       FIXXY
000176                  STX       DECCNTB
000177                  STY       DECCNT+1
000178 GSUBINF          PHA
000179                  LDA       CNTDIGS                 ;AT A TRANSITION?
000180                  BEQ       GSUBIN2                 ;NO, SKIP.
000181                  LDA       TEMP                    ;THIS HUNK FREE?
000182                  BEQ       GSUBIN1
000183                  LDA       FRESPC                  ;GET BACK ADDR OF BEGINING OF THIS HUNK.
000184                  STA       HIGHTR                  ;THIS IS THE BEGINNING OF THE HUNK TO MOVE.
000185                  LDA       FRESPCB
000186                  STA       HIGHTRB
000187                  LDA       FRESPC+1
000188                  STA       HIGHTR+1
000189                  JMP       GSUBIN2                 ;ALWAYS.
000190 GSUBIN1          LDA       FRESPC                  ;THIS IS EXECUTED ON THE END OF HUNK TO MOVE.
000191                  STA       LOWTR
000192                  LDA       FRESPCB
000193                  STA       LOWTRB
000194                  LDA       FRESPC+1
000195                  STA       LOWTR+1
000196                  TXA
000197                  PHA
000198                  TYA
000199                  PHA
000200                  JSR       BLTUC
000201                  PLA
000202                  TAY
000203                  PLA
000204                  TAX
000205 GSUBIN2          PLA                               ;IN THE MIDDLE OF FREE OR USED STUFF.
000206 GSUBIN3          STA       FRESPC                  ;GET READY TO SUBTACT OFF THE INFO BYTES
000207                  STY       FRESPC+1
000208                  STX       FRESPCB
000209                  CPX       FRETOPB
000210                  BNE       GMOVCHK                 ;If FRESPC has reached FRETOP then
000211                  CPY       FRETOP+1                ; all is collected
000212                  BNE       GMOVCHK
000213                  CMP       FRETOP
000214                  BNE       GMOVCHK
000215                  LDA       TEMP                    ;ONLY FREE STUFF LEFT?
000216                  BEQ       GRBDON                  ;YES, THEN ALL DONE.
000217                  LDA       FRETOP
000218                  STA       LOWTR                   ;NO, SO TRANSFER THE LAST HUNK.
000219                  STX       LOWTRB
000220                  STY       LOWTR+1
000221                  JMP       GRBLTUC
000222 GMOVCHK          SEC
000223                  SBC       #INFOSIZ                ;OF THE NEXT PIECE OF STRING SPACE.
000224                  BCS       GSUBIN4
000225                  JSR       FIXXY
000226 GSUBIN4          STX       DECCNTB
000227                  STY       DECCNT+1
000228                  STA       DECCNT
000229                  JMP       GMOVPTR
000230 *WE NOW ARE POINTING AT THE NEXT FREE HUNK, AND WE HAVE
000231 * HAVE JUST FIXED THE POINTERS TO THE LAST USED HUNK
000232 * SO WE ARE READY TO DO A BLOCK MOVE.
```

```
000233 GRBLTUC         JSR       BLTUC                    ;MOVE IT!
000234 GRBDON          EQU       *
000235                 LDA       HIGHDS                   ;LAST USED STORAGE.
000236                 STA       FRETOP
000237                 LDA       HIGHDS+1
000238                 STA       FRETOP+1
000239                 LDA       HIGHDSB
000240                 STA       FRETOPB
000241                 RTS
000242 GFINDTEMP       LDA       #0
000243                 STA       HEADERB
000244                 LDA       #>TEMPST                 ;LOW BYTE OF POINTER TO
000245                 LDX       #<TEMPST                 ;TEMPORARY DESCRIPTORS.
000246                 STA       HEADER
000247                 STX       HEADER+1                 ;CURRENT DESCRIPTOR.
000248 GARBTEMP        CMP       TEMPPT                   ;LAST DESCRPTOR?
000249                 BEQ       NOTEMP                   ;YES, THIS STRING DIDN'T GET CLAIMED.
000250                 LDY       #0
000251                 LDA       (HEADER),Y               ;GET LENGTH.
000252                 BEQ       DVARTS                   ;NULL STRING, DON'T CARE.
000253                 STA       ANYNUM
000254                 INY
000255                 INY
000256                 LDA       (HEADER),Y               ;HIGH BYTE OF POINTER
000257                 TAX
000258                 DEY
000259                 LDA       HIMEM
000260                 SEC
000261                 SBC       (HEADER),Y               ;LOW BYTE OF POINTER.
000262                 STA       LOWTR
000263                 LDA       HIMEM+1
000264                 LDY       HIMEMB
000265                 JSR       FIXYAX                   ;SUBTRACT X FROM Y.A
000266                 STA       LOWTR+1
000267                 STY       LOWTRB
000268                 LDA       LOWTR
000269                 CLC
000270                 ADC       ANYNUM
000271                 STA       LOWTR
000272                 LDY       LOWTR+1
000273                 LDX       LOWTRB
000274                 BCC       *+6
000275                 INY
000276                 JSR       FIXYX
000277                 CPX       DECCNTB
000278                 BNE       DVARTS
000279                 CPY       DECCNT+1                 ;IS THIS MY DESCRIPTOR?
000280                 BNE       DVARTS                   ;NO, TRY NEXT ONE.
000281                 CMP       DECCNT
000282                 BNE       DVARTS
000283 ;I FOUND IT!   IT'S RIGHT IN BETWEEN THE LABIA MINORA.
000284                 JMP       GRBCMP2                  ;UPDATE IT AND DO NEXT STRING.
000285 DVARTS          LDA       #STRSIZ                  ;SIZE OF A DESCRIPTOR.
000286                 CLC
000287                 ADC       HEADER                   ;POINT TO NEXT DESCR.
000288                 STA       HEADER
000289                 JMP       GARBTEMP
000290 NOTEMP          LDX       #ERRVA                   ;CURRENT STRING UNLABLED, AND NOT TEMP.
000291                 JMP       ERROR
000292 ;
000293 ; The following routine Concatenates two strings.
000294 ;   ARG holds data on the first one at this point.
000295 ;   FAC holds data on the second one.
000296 ;   Format is: Len, Address, TEMP pointer
000297 CONCAT          LDY       #0
000298                 LDA       (FACMO),Y                ;Get length of first
000299                 CLC
000300                 ADC       (ARGMO),Y                ;Add length of second
000301                 BCC       SIZEOK                   ;If total<256, it's cool.
000302                 JSR       CAT11
000303                 LDX       #ERRLS                   ;STRING TOO LONG ERR.
000304                 JMP       ERROR
000305 SIZEOK          JSR       STRINI                   ;GET SPACE, SET DSCTMP UP.
000306                 LDX       ARGMO
000307                 LDY       ARGMO+1
000308                 STX       STRNG1
000309                 STY       STRNG1+1                 ;SET UP STRNG1 FOR THE MOVE..
000310                 LDX       ARGMOB
000311                 STX       STRNG1B
000312                 JSR       MOVINS                   ;MOVE IN 1ST ARG.
```

```
000313                  CLC
000314                  ADC       FRESPC
000315                  STA       FRESPC
000316                  BCC       CAT10
000317                  INC       FRESPC+1
000318                  LDA       FRESPC+1                ;INC FRESPC+1
000319                  CMP       #MAXPG
000320                  BCC       CAT10
000321                  SBC       #MAXPG-MINPG
000322                  INC       FRESPCB
000323                  STA       FRESPC+1
000324 CAT10             LDX       FACMO
000325                  LDY       FACMO+1
000326                  STX       STRNG1
000327                  STY       STRNG1+1                ;SET UP STRNG1 FOR THE 2ND MOVE.
000328                  LDX       FACMOB
000329                  STX       STRNG1B
000330                  JSR       MOVINS                  ;MOVE IT IN.
000331                  JSR       CAT11
000332                  JMP       PUTNEW
000333 CAT11             JSR       FRECNOW                 ;FREE IT UP IF POSSIBLE.
000334                  LDA       ARGMO
000335                  LDY       ARGMO+1
000336                  LDX       ARGMOB
000337                  JMP       FRETNOW                 ;IF POSSIBLE, FREE IT UP.
000338 MAKREL            PHA
000339                  LDA       HIMEM
000340                  SEC
000341                  SBC       INDEX
000342                  STA       INDEX                   ;TAKES REL. ADDR AND MAKES ABSOLUTE,
000343                  LDA       HIMEM+1                 ;OR VICA VERCA.
000344                  STY       YSAVE
000345                  LDY       HIMEMB
000346                  SBC       INDEX+1
000347                  JSR       FIXSBC
000348                  STA       INDEX+1
000349                  TYA
000350                  SBC       INDEXB
000351                  STA       INDEXB
000352                  LDY       YSAVE
000353                  PLA
000354                  RTS
000355 MOVINS            LDY       #0                      ;GET ADDR OF STRNG.
000356                  LDA       (STRNG1),Y
000357                  PHA
000358                  INY
000359                  LDA       (STRNG1),Y
000360                  TAX
000361                  INY
000362                  LDA       (STRNG1),Y
000363                  LDY       #0
000364                  JSR       FIXYA
000365                  STY       INDEXB
000366                  STX       INDEX
000367                  STA       INDEX+1
000368                  JSR       MAKREL
000369                  PLA
000370 MOVDO:            TAY
000371                  BEQ       MVSTRT
000372                  PHA
000373 MOVLP:            DEY
000374                  LDA       (INDEX),Y
000375                  STA       (FRESPC),Y
000376                  TYA
000377                  BNE       MOVLP
000378                  PLA
000379 MVSTRT:           RTS
000380 FREFAC            LDA       FACMO                   ;Free up string pointed to by FAC
000381                  LDX       FACMOB
000382                  LDY       FACMO+1
000383 ;
000384 ; Procedure: FRETMP
000385 ; Pass a string descriptor pointer in A, Y.  A check is made to see
000386 ;   if the string descriptor points to a Temporary descriptor allocated
000387 ;   by PUTNEW.  If so, the temporary is freed up by updating TEMPPT.
000388 ; If a Temp is freed up, a further check sees if the string that Temp
000389 ;   pointed to is the lowest part of the string.  If so, FRETOP is
000390 ;   updated to reflect the fact that the temp is no longer in use.
000391 ; On Exit:
000392 ;   X, Y hold the address of the actual string
```

```
000393 ;   A holds its length
000394 FRETMP          STA       INDEX
000395                 STX       INDEXB
000396                 STY       INDEX+1                  ;GET LENGTH FOR LATER.
000397                 JSR       FRETMS                   ;FREE UP THE TEMPORARY DESC.
000398 ;
000399 ; Procedure: NOTNOW
000400 ; On Entry:
000401 ;   INDEX holds a pointer to a string descriptor
000402 ; On Exit:
000403 ;   A holds length of the string
000404 ;   INDEX holds pointer to the actual string
000405 NOTNOW          PHP                                ;SAVE CODES.
000406                 LDY       #0                       ;PREP TO GET STUFF.
000407                 LDA       (INDEX),Y                ;GET COUNT AND
000408                 PHA                                ;SAVE IT.
000409                 INY
000410                 LDA       (INDEX),Y
000411                 TAX                                ;SAVE LOW ORDER.
000412                 INY
000413                 LDA       (INDEX),Y
000414                 LDY       #0
000415                 JSR       FIXYA
000416                 STY       INDEXB
000417                 STA       INDEX+1                  ;SAVE HIGH ORDER.
000418                 STX       INDEX
000419                 JSR       MAKREL
000420                 LDX       INDEX
000421                 LDY       INDEX+1
000422                 PLA
000423                 PLP
000424                 RTS
000425 FREFC1          PHP                                ;SAVE THE FREE STATUS
000426                 JSR       FREFAC                   ;GET THE DESCRIPTOR TO THE STRING IN QUESTION.
000427                 PLP                                ;Get the FREE STATUS back
000428 FRENOW          BNE       FRERTS                   ;EXIT IF NOT TIME TO FREE A TEMP.
000429 FRESPA          EQU       *                        ;Free up String Space
000430                 CMP       #0
000431                 BEQ       FRERTS                   ;DON'T FRE NULL STRINGS
000432                 JMP       TSTFRE
000433 FRETMS          PHA                                ;Save location of Temporary
000434                 LDA       #>TEMPST                 ;Get TEMP starting point
000435                 CMP       TEMPPT                   ;Current pointer past start of TEMPs?
000436                 PLA
000437                 BCC       *+5                      ;YES
000438                 LDA       #$FF                     ;NO, CLEAR Z FLAG.
000439                 RTS
000440                 CPY       LASTPT+1                 ;LAST ENTRY TO TEMP?
000441                 BNE       FRERTS
000442                 CMP       LASTPT
000443                 BNE       FRERTS
000444                 STA       TEMPPT
000445                 SBC       #STRSIZ                  ;POINT TO LAST ONE.
000446                 STA       LASTPT                   ;UPDATE TEMP PNTR.
000447                 LDY       #0                       ;ALSO CLEARS ZFLG SO WE DO REST OF FRETMP.
000448 FRERTS:         RTS       ALL                      DONE.
000449 NOTFAC          LDA       FACMO
000450                 LDY       FACMO+1
000451                 LDX       FACMOB
000452 NOTNW2          STA       INDEX                    ;ALT ENTRY FOR NOTNOW.
000453                 STY       INDEX+1
000454                 STX       INDEXB
000455                 JMP       NOTNOW
000456
000457 ; ###############################################################################################
000458 ; #   END OF FILE:  STRUTILS.TEXT
000459 ; #   LINES      :  450
000460 ; #   CHARACTERS :  18646
000461 ; ###############################################################################################
```

```
+-------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 461   CHARACTERS: 19200
|
+-------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3MATHK.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:29 PM
|  Modified: Wednesday, December 31, 1997             4:37:06 PM
|
+------------------------------------------------------------------------------

000001 ; ##############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3MATHK.TEXT
000004 ; ##############################################################################################
000005
000006                 SBTL      "FLOATING POINT MATH PACKAGE CONFIGURATION."
000007 ; FLOATING POINT NUMBER REPRESENTATION
000008 ;
000009 ; The floating point format is as follows:
000010 ;
000011 ; The exponent is stored in excess of 128, ie.  with a bias of 128, so, the
000012 ;   exponent is a signed 8-bit number with 128 added.  An exponent of ZERO means
000013 ;   that the number is zero.  The other bytes may not be assumed to be zero.
000014 ;
000015 ; The mantissa is 23 bits long.  The binary point is to the left of the MSB
000016 ;   of the mantissa.  The mantissa is positive with a 1 as a 24th bit assumed
000017 ;   to be between the the binary point and the MSB.
000018 ;
000019 ; The number in memory looks like this: 84 A0 00 00   (The number is -10.)
000020 ;
000021 ;      Exponent     Sign              Mantissa
000022 ;  ---------------------------------------------------------------------
000023 ; | 1 0 0 0 0 1 0 0 | 1 | 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
000024 ;  ---------------------------------------------------------------------
000025 ;       8 Bits      1 Bit             23 bits
000026 ;
000027 ; To evaluate a number like the one above, first evaluate the exponent.
000028 ;
000029 ;      1 0 0 0 0 1 0 0 = 132
000030 ;
000031 ; subtract the bias (128) to get the true exponent, in this case it is 4.
000032 ; Next evaluate the mantissa:
000033 ;
000034 ; . 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
000035 ;   ^
000036 ;   | This is the IMPLIED bit between the MSB and the binary point.
000037 ;
000038 ;                                                      -1    -3
000039 ; . 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 = 2   + 2
000040 ;                                                    =.5   + .125
000041 ;                                                    =.625
000042 ;
000043 ; Next, multiply the mantissa by 2 ^ exponent.
000044 ;
000045 ; .625 * 2 ^ 4 = .625 * 16 = 10
000046 ;
000047 ; Since the sign bit is a 1, the sign of the number is negative and the final
000048 ;   result is -10.
000049                 PAGE
000050 ; TO KEEP THE SAME NUMBER IN THE FAC WHILE SHIFTING,
000051 ;  TO SHIFT RIGHT, EXP:=EXP+1
000052 ;  TO SHIFT LEFT,  EXP:=EXP-1
000053 ; Arithmetic routine calling convention for 1 argument functions:
000054 ;    The argument is in the FAC (Floating point Accumulator).
000055 ;    The result is left in the FAC.
000056 ; Arithmetic routine calling convention for 2 argument functions:
000057 ;    The 1st argument is in ARG (ARGEXP, HO, MO, LO AND ARGSGN)
000058 ;    The 2nd argument is in the FAC.
000059 ;    The result is left in the FAC.
000060 ; THE 'T' ENTRY POINTS TO THE 2-ARGUMENT OPERATIONS HAVE B
000061 ; SETUP IN RESPECTIVE REGISTERS. BEFORE CALLING ARG MAY
000062 ; POPPED OFF THE STACK AND INTO ARG, FOR EXAMPLE.
000063 ; THE OTHER ENTRY POINTSSUMES Y,A POINTS TO THE ARGUMENT
000064 ; SOMEWHERE IN MEMORY. IT IS UNPACKED INTO ARG BY 'CONUPK'
000065 ; ON THE STACK, THE SGN IS PUSHED ON FIRST, THE LO,MO,HO &
000066 ; NOTE ALL THINGS ARE KEPT UNPACKED IN ARG, FAC & ON THE S
000067 ; IT IS ONLY WHEN SOMETHING IS STORED THAT IT IS PACKED
000068 ; BYTES. UNPACKED FORMAT HAS A SGN BYTE REFLECTING THE S
000069 ; NUMBER (POSITIVE=0, NEGATIVE=-1) A HO,MO & LO WITH THE H
000070 ; OF THE HO TURNED ON. THE EXP IS THE SAMES STORED FORMAT.
000071 ; THIS IS DONE FOR SPEED OF OPERATION.
000072                 PAGE
```

```
000073                  SBTL    "FLOATING POINT ADDITION AND SUBTRACTION."
000074 FADDH:           LDA     #>FHALF
000075                  LDY     #<FHALF                 ;ENTRY TO ADD 1/2.
000076                  JMP     FADD                    ;UNPACK AND GO ADD UIT.
000077 FSUB             LDX     #0
000078                  JSR     CONUPK                  ;UNPACK ARGUMENT INTO ARG.
000079 FSUBT:           LDA     FACSGN
000080                  EOR     #$FF                    ;COMPLEMENT IT.
000081                  STA     FACSGN
000082                  EOR     ARGSGN                  ;COMPLEMENT ARISGN.
000083                  STA     ARISGN
000084                  LDA     FACEXP                  ;Y=ARGEXP..
000085                  JMP     FADDT
000086 FADD5:           JSR     SHIFTR                  ;DO A LONG SHIF
000087                  BCC     FADD4                   ;CONTINUE WITH ADDITION.
000088 FADD             LDX     #0
000089                  JSR     CONUPK
000090 FADDT:           EQU     *
000091                  BNE     *+5
000092                  JMP     MOVFA                   ;IF FAC=0, RESULT IS IN ARG.
000093                  LDX     FACOV
000094                  STX     OLDOV
000095                  LDX     #ARGEXP                 ;DEFAULT IS SHIFT ARGUMENT.
000096                  LDA     ARGEXP                  ;IF ARG=0, FAC IS RESULT.
000097 FADDC:           TAY                             ;ALSO COPY ACCA INTO ACCY.
000098                  BNE     *+3                     ;RETURN IF ZERO
000099                  RTS
000100                  SEC
000101                  SBC     FACEXP
000102                  BEQ     FADD4                   ;NO SHIFTING.
000103                  BCC     FADDA                   ;BR IF ARGEXP.LT.FACEXP.
000104                  STY     FACEXP                  ;RESULTING EXPONENT.
000105                  LDY     ARGSGN                  ;SINCE ARG IS BIGGER, IT'S
000106                  STY     FACSGN                  ;SIGN IS SIGN OF RESU.
000107                  EOR     #$FF                    ;SHIFT A NEGATIVE NUMBER OF PLACES.
000108                  ADC     #0                      ;COMPLETE NEGATION. W/ C=1.
000109                  LDY     #0                      ;ZERO OLDOV.
000110                  STY     OLDOV
000111                  LDX     #FAC                    ;SHIFT THE FAC INSTEAD.
000112                  BNE     FADD1
000113 FADDA:           LDY     #0
000114                  STY     FACOV
000115 FADD1:           CMP     #$100-7                 ;FOR SPEED AND NECESSITY.  GETS
000116 ;MOST LIKELY CASE TO SHIFTR FASTEST
000117 ;AND ALLOWS SHIFTING OF NEG NUMS
000118 ;BY 'QINT'.
000119                  BMI     FADD5                   ;SHIFT BIG.
000120                  TAY
000121                  LDA     FACOV                   ;SET FACOV.
000122                  LSR     1,X                     ;GETS 0 IN MOST SIG BIT.
000123                  JSR     ROLSHF                  ;DO THE ROLLING.
000124 FADD4:           BIT     ARISGN                  ;GET SULTING SIGN.
000125                  BPL     FADD2                   ;IF POSITIVE, ADD.
000126 ;CARRY IS CLEAR.
000127                  LDY     #FACEXP
000128                  CPX     #ARGEXP                 ;FAC IS BIGGER.
000129                  BEQ     SUBIT
000130                  LDY     #ARGEXP                 ;ARG IS BIGGER.
000131 SUBIT:           SEC
000132                  EOR     #$FF
000133                  ADC     OLDOV
000134                  STA     FACOV
000135                  LDA     3+1,Y
000136                  SBC     3+1,X
000137                  STA     FACLO
000138                  LDA     2+1,Y
000139                  SBC     2+1,X
000140                  STA     FACMO
000141                  LDA     2,Y
000142                  SBC     2,X
000143                  STA     FACMOH
000144                  LDA     1,Y
000145                  SBC     1,X
000146                  STA     FACHO
000147 FADFLT:          BCS     NORMAL                  ;HERE IF SIGNS DIFFER. IF CARRY,
000148 ;FAC IS SET OK.
000149                  JSR     NEGFAC                  ;NEGATE FAC.
000150 NORMAL:          LDY     #0
000151                  TYA
000152                  CLC
```

```
000153 NORM3:          LDX       FACHO
000154                 BNE       NORM1
000155                 LDX       FACHO+1                ;SHIFT 8 BITS AT A TIME FOR SPEED.
000156                 STX       FACHO
000157                 LDX       FACMOH+1
000158                 STX       FACMOH
000159                 LDX       FACMO+1
000160                 STX       FACMO
000161                 LDX       FACOV
000162                 STX       FACLO
000163                 STY       FACOV
000164                 ADC       #08
000165                 CMP       #$20
000166                 BNE       NORM3
000167 ZEROFC:         LDA       #0                     ;NOT NEED BY NORMAL BUT BY OTHERS.
000168 ZEROF1:         STA       FACEXP                 ;NUMBER MUST BE ZERO.
000169 ZEROML:         STA       FACSGN                 ;MAKE SIGN POSITIVE.
000170                 RTS                              ;ALL DONE.
000171 FADD2:          ADC       OLDOV
000172                 STA       FACOV
000173                 LDA       FACLO
000174                 ADC       ARGLO
000175                 STA       FACLO
000176                 LDA       FACMO
000177                 ADC       ARGMO
000178                 STA       FACMO
000179                 LDA       FACMOH
000180                 ADC       ARGMOH
000181                 STA       FACMOH
000182                 LDA       FACHO
000183                 ADC       ARGHO
000184                 STA       FACHO
000185                 JMP       SQUEEZ                 ;GO ROUND IF SIGNS SAME.
000186 NORM2:          ADC       #1                     ;DECREMENT SHIFT COUNT.
000187                 ASL       FACOV                  ;SHIFT ALL LEFT ONE BIT.
000188                 ROL       FACLO
000189                 ROL       FACMO
000190                 ROL       FACMOH
000191                 ROL       FACHO
000192 NORM1:          BPL       NORM2                  ;IF MSB=0 SHIFT AGAIN.
000193                 SEC
000194                 SBC       FACEXP
000195                 BCS       ZEROFC
000196                 EOR       #$FF
000197                 ADC       #1                     ;COMPLEMENT.
000198                 STA       FACEXP
000199 SQUEEZ:         BCC       RNDRTS                 ;BITS TO SHIFT?
000200 RNDSHF:         INC       FACEXP
000201                 BEQ       OVERR
000202                 ROR       FACHO
000203                 ROR       FACMOH
000204                 ROR       FACMO
000205                 ROR       FACLO
000206                 ROR       FACOV
000207 RNDRTS:         RTS                              ;ALL DONE ADDING.
000208 NEGFAC:         LDA       FACSGN
000209                 EOR       #255
000210                 STA       FACSGN                 ;COMPLEMENT FAC  ENTIRELY.
000211 NEGFCH:         STX       DPTFLG
000212                 LDX       #3                     ;COMPLEMENT JUST THE NUMBER.
000213 NEGFCH1         LDA       FACHO,X
000214                 EOR       #$FF
000215                 STA       FACHO,X
000216                 DEX
000217                 BPL       NEGFCH1
000218                 LDX       DPTFLG
000219                 LDA       FACOV
000220                 EOR       #255
000221                 STA       FACOV
000222                 INC       FACOV
000223                 BNE       INCFRT
000224 INCFAC:         INC       FACLO
000225                 BNE       INCFRT
000226                 INC       FACMO
000227                 BNE       INCFRT                 ;IF NO CARRY, RETURN.
000228                 INC       FACMOH
000229                 BNE       INCFRT
000230                 INC       FACHO                  ;CARRY INCREMENT.
000231 INCFRT:         RTS
000232 OVERR:          LDX       #ERROV
```

```
000233                  JMP       ERROR                     ;TELL USER.
000234 ; 'SHIFTR' SHIFTS X+1:X+3 -ACCA  BITS RIGHT.
000235 ; SHIFTS BYTES TO START WITH IF POSSIBLE.
000236 MULSHF:          LDX       #RESHO-1                  ;ENTRY POINT FOR MULTIPLIER.
000237                  LDY       #0
000238                  STY       BITS
000239 SHFTR2:          LDY       3+1,X                     ;SHIFT BYTES FIRST.
000240                  STY       FACOV
000241                  LDY       3,X
000242                  STY       4,X
000243                  LDY       2,X                       ;GET MO.
000244                  STY       3,X                       ;STORE LO.
000245                  LDY       1,X                       ;GET HO.
000246                  STY       2,X                       ;STORE MO.
000247                  LDY       BITS
000248                  STY       1,X                       ;STORE HO.
000249 SHFTR0           ADC       #8
000250                  BMI       SHFTR2
000251                  BEQ       SHFTR2
000252                  SBC       #8                        ;C CAN BE EITHER 1,0 AND IT WORKS.
000253                  TAY
000254                  LDA       FACOV
000255                  BCS       SHFTRT                    ;EQUIV TO BEQ HERE.
000256 SHFTR3:          ASL       1,X
000257                  BCC       SHFTR4
000258                  INC       1,X
000259 SHFTR4:          ROR       1,X
000260                  ROR       1,X                       ;YES, TWO OF THEM.
000261 ROLSHF:          EQU       *
000262                  ROR       2,X
000263                  ROR       3,X
000264                  ROR       4,X                       ;ONE MO TIME.
000265                  ROR       A                         ;ROTATE ARGUMENT 1 BIT RIGHT.
000266                  INY
000267                  BNE       SHFTR3                    ;$$$ ( MOST EXPENSIVE ! )
000268 SHFTRT:          CLC                                 ;CLEAR OUTPUT OF FACOV.
000269                  RTS
000270 SHIFTR           LDY       #0                        ;THIS CURES MANY AILMENTS.
000271                  STY       BITS
000272 SHFTR1           EQU       *                         ;THIS ENTRY USED BY QINT.
000273                  JMP       SHFTR0
000274
000275 ; ##############################################################################################
000276 ; #   END OF FILE:  B3MATHK.TEXT
000277 ; #   LINES      :  268
000278 ; #   CHARACTERS :  11697
000279 ; ##############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 279   CHARACTERS: 12249
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3MATHL.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:30 PM
|  Modified: Wednesday, December 31, 1997             4:37:06 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3MATHL.TEXT
000004 ; ###############################################################################################
000005
000006                  PAGE
000007                  SBTL      "NATURAL LOG FUNCTION."
000008 ; CALCULATION IS BY:
000009 ; LN(F*2N)=(N+LOG2(F))*LN(2)
000010 ; AN APPXIMATION POLYNOMIAL IS USED TO CALCULATE LOG2(F).
000011 ;  CONSTANTS USED BY LOG:
000012 FONE:            DFB       $81                     ; 1.0
000013                  DFB       000
000014                  DFB       000
000015                  DFB       000
000016                  DFB       0
000017 LOGCN2:          DFB       3                       ;DEGREE-1
000018                  DFB       $7F                     ;.43425594188
000019                  DFB       $5E
000020                  DFB       $56
000021                  DFB       $CB
000022                  DFB       $79
000023                  DFB       $80                     ; .57658454134
000024                  DFB       $13
000025                  DFB       $9B
000026                  DFB       $0B
000027                  DFB       $64
000028                  DFB       $80                     ; .96180075921
000029                  DFB       $76
000030                  DFB       $38
000031                  DFB       $93
000032                  DFB       $16
000033                  DFB       $82                     ; 2.8853900728
000034                  DFB       $38
000035                  DFB       $AA
000036                  DFB       $3B
000037                  DFB       $20
000038 SQR0.5:          DFB       $80                     ; S(0.5)
000039                  DFB       $35
000040                  DFB       $04
000041                  DFB       $F3
000042                  DFB       $34
000043 SQR2.0:          DFB       $81                     ; SQR(2.0)
000044                  DFB       $35
000045                  DFB       $04
000046                  DFB       $F3
000047                  DFB       $34
000048 NEGHLF:          DFB       $80                     ; -1/2
000049                  DFB       $80
000050                  DFB       000
000051                  DFB       000
000052                  DFB       0
000053 LOG2:            DFB       $80                     ; LN(2)
000054                  DFB       $31
000055                  DFB       $72
000056                  DFB       $17
000057                  DFB       $F8
000058 LOG:             JSR       SIGN                    ;IS IT POSITIVE?
000059                  BEQ       LOGERR
000060                  BPL       LOG1
000061 LOGERR:          JMP       FCERR                   ;CAN'T TOLERATE NEG OR ZERO.
000062 LOG1:            LDA       FACEXP                  ;GET EXPONENT INTO ACCA.
000063                  SBC       #$7F                    ;REMOVE BIAS. (CARRY IS OFF)
000064                  PHA                               ;SE AWHILE.
000065                  LDA       #$80
000066                  STA       FACEXP                  ;RESULT IS FAC IN RANGE 0.5,1.
000067                  LDA       #>SQR0.5
000068                  LDY       #<SQR0.5                ;GET POINTER TO SQR(0.5).
000069 ; CALCULATE (F-SQR(.5))/(F+SQR(.5))
000070                  JSR       FADD                    ;ADD TO FAC.
000071                  LDA       #>SQR2.0
000072                  LDY       #<SQR2.0                ;GET SQR(2.).
```

```
000073                  JSR       FDIV
000074                  LDA       #>FONE
000075                  LDY       #<FONE
000076                  JSR       FSUB
000077                  LDA       #>LOGCN2
000078                  LDY       #<LOGCN2
000079                  JSR       POLYX                   ;EVALUATE APPROXIMATION POLYNOMIAL.
000080                  LDA       #>NEGHLF
000081                  LDY       #<NEGHLF                ;ADD IN LAST CONSTANT.
000082                  JSR       FADD
000083                  PLA                               ;GET EXPONENT BACK.
000084                  JSR       FINLOG                  ;ADD IT IN.
000085                  LDA       #>LOG2
000086                  LDY       #<LOG2                  ;MULTIPLY RESULT BY LOG(2.0).
000087                  LDX       #SQR0B                  ; JMP FMULT   ;MULTIPLY TOGETHER.
000088                  PAGE
000089                  SBTL      "FLOATING MULTIPLICATION AND DIVISION."
000090 FMULT           LDX       #0                      ;MULTIPLICATION  FAC:=ARG*FAC.
000091                  JSR       CONUPK                  ;UNPACK THE CONSTANT INTO ARG FOR USE.
000092 FMULTT:         BNE       *+5
000093                  JMP       MULTRT                  ;IF FAC=0, RETURN. FAC IS SET.
000094                  JSR       MULDIV                  ;FIX UP THE EXPONENTS.
000095                  LDA       #0                      ;TO CLEAR RESULT.
000096                  STA       RESHO
000097                  STA       RESMOH
000098                  STA       RESMO
000099                  STA       RESLO
000100                  LDA       FACOV
000101                  JSR       MLTPLY
000102                  LDA       FACLO                   ;MLTPLARG BY FACLO.
000103                  JSR       MLTPLY
000104                  LDA       FACMO                   ;MLTPLY ARG BY FACMO.
000105                  JSR       MLTPLY
000106                  LDA       FACMOH
000107                  JSR       MLTPLY
000108                  LDA       FACHO                   ;MLTPLY ARG BY FACHO.
000109                  JSR       MLTPL1
000110                  JMP       MOVFR                   ;MOVE RESULT INTO FAC,
000111 ;NORMALIZE RESULT, AND RETURN.
000112 MLTPLY:         BNE       *+5
000113                  JMP       MULSHF                  ;SHIFT RESULT RIGHT 1 BYTE.
000114 MLTPL1:         LSR       A
000115                  ORA       #$80
000116 MLTPL2:         TAY
000117                  BCC       MLTPL3                  ;IT MULT BIT=0, JUST SHIFT.
000118                  CLC
000119                  LDA       RESLO
000120                  ADC       ARGLO
000121                  STA       RESLO
000122                  LDA       RESMO
000123                  ADC       ARGMO
000124                  STA       RESMO
000125                  LDA       RESMOH
000126                  ADC       ARGMOH
000127                  STA       RESMOH
000128                  LDA       RESHO
000129                  ADC       ARGHO
000130                  STA       RESHO
000131 MLTPL3:         ROR       RESHO
000132                  ROR       RESMOH
000133                  ROR       RESMO
000134                  ROR       RESLO
000135                  ROR       FACOV                   ;SAVE FOR ROUNDING.
000136                  TYA
000137                  LSR       A                       ;CLEAR MSB SO WE GET A CLOSER TO 0.
000138                  BNE       MLTPL2                  ;SLOW AS A TURTLE !
000139 MULTRT:         RTS
000140 ;
000141 ;  Routine to Unpack MEMORY into ARG.
000142 ;
000143 CONUPK:         STA       INDEX1
000144                  STX       INDEX1B
000145                  STY       INDEX1+1
000146                  LDY       #3+1
000147                  LDA       (INDEX1),Y
000148                  CPX       #0
000149                  BEQ       *+4
000150                  LDA       #0
000151                  STA       ARGLO
000152                  DEY
```

```
000153                  LDA       (INDEX1),Y
000154                  STA       ARGMO
000155                  DEY
000156                  LDA       (INDEX1),Y
000157                  STA       ARGMOH
000158                  DEY
000159                  LDA       (INDEX1),Y
000160                  STA       ARGSGN
000161                  EOR       FACSGN
000162                  STA       ARISGN
000163                  LDA       ARGSGN
000164                  ORA       #$80
000165                  STA       ARGHO
000166                  DEY
000167                  LDA       (INDEX1),Y
000168                  STA       ARGEXP
000169                  LDA       FACEXP                    ;SET CODES OF FACEXP.
000170                  RTS
000171 ; Check special cases and ADD Exponents for FMULT, FDIV.
000172 MULDIV:          LDA       ARGEXP                    ;EXP OF ARG=0?
000173 MLDEXP:          BEQ       ZEREMV                    ;SO WE GET ZERO EXPONENT.
000174                  CLC
000175                  ADC       FACEXP                    ;RESULT IS IN ACCA.
000176                  BCC       TRYOFF                    ;FIND C XOR N.
000177                  BMI       GOOVER                    ;OVERFLOW IF BITS MATCH.
000178                  CLC
000179                  DFB       44
000180 TRYOFF:          BPL       ZEREMV                    ;UNDERFLOW.
000181                  ADC       #$80                      ;ADD BIAS.
000182                  STA       FACEXP
000183                  BNE       *+5
000184                  JMP       ZEROML                    ;ZE THE REST OF IT.
000185                  LDA       ARISGN
000186                  STA       FACSGN                    ;ARISGN IS RESULT'S SIGN.
000187                  RTS                                 ;DONE.
000188 MLDVEX:          LDA       FACSGN                    ;GET SIGN.
000189                  EOR       #$FF                      ;COMPLEMENT IT.
000190                  BMI       GOOVER
000191 ZEREMV:          PLA                                 ;GET ADDR OFF STACK.
000192                  PLA
000193                  JMP       ZEROFC                    ;UNDERFLOW.
000194 GOOVER:          JMP       OVERR                     ;OVERFLOW.
000195 ;
000196 ;   Multiply FAC by 10.
000197 MUL10:           JSR       MOVAF                     ;COPY FAC INTO ARG.
000198                  TAX
000199                  BEQ       MUL10R                    ;IF FAC=0, GOT ANSWER.
000200                  CLC
000201                  ADC       #2                        ;AUGMENT EXP BY 2.
000202                  BCS       GOOVER                    ;OVERFLOW.
000203                  LDX       #0
000204                  STX       ARISGN                    ;SIGNS ARE SAME.
000205                  JSR       FADDC                     ;ADD TOGETHER.
000206                  INC       FACEXP                    ;MULTIPLY BY TWO.
000207                  BEQ       GOOVER                    ;OVERFLOW.
000208 MUL10R:          RTS
000209 ;
000210 ;   Divide FAC by 10.
000211 TEN.C:           DFB       $84
000212                  DFB       $20
000213                  DFB       000
000214                  DFB       000
000215                  DFB       0
000216 DIV10:           JSR       MOVAF                     ;MOVE FAC TO ARG.
000217                  LDA       #>TEN.C
000218                  LDY       #<TEN.C                   ;POINT TO CONSTANT OF 10.0
000219                  LDX       #0                        ;SIGNS ARE BOTH POSITIVE.
000220 FDIVF:           STX       ARISGN
000221                  LDX       #TEN.CB
000222                  JSR       MOVFM                     ;PUT IT INTO FAC.
000223                  JMP       FDIVT                     ;SKIP OVER NEXT TWO BYTES.
000224 FDIV             LDX       #0
000225                  JSR       CONUPK                    ;UNPACK CONSTANT.
000226 FDIVT:           BNE       NO0ERR                    ;CAN'T DIVIDE BY ZERO !
000227                  JMP       DV0ERR
000228 ;(NOT ENOUGH ROOM TO STORE RESULT.)
000229 NO0ERR           JSR       ROUND                     ;TAKE FACOV INTO ACCT IN FAC.
000230                  LDA       #0                        ;NEGATE FACEXP.
000231                  SEC
000232                  SBC       FACEXP
```

```
000233                  STA       FACEXP
000234                  JSR       MULDIV                    ;FIX UP EXPONENTS.
000235                  INC       FACEXP                    ;SCALE IT RIGHT.
000236                  BEQ       GOOVER                    ;OVERFLOW.
000237                  LDX       #$100-3-1                 ;SETUP PROCEDURE.
000238                  LDA       #1
000239 DIVIDE:          EQU       *                         ;THIS IS THE BEST CODE IN THE WHOLE PILE
000240                  LDY       ARGHO                     ;SEE WHAT RELATION HOLDS.
000241                  CPY       FACHO
000242                  BNE       SAVQUO                    ;C=0,1. N(C=0)=0.
000243                  LDY       ARGMOH
000244                  CPY       FACMOH
000245                  BNE       SAVQUO
000246                  LDY       ARGMO
000247                  CPY       FACMO
000248                  BNE       SAVQUO
000249                  LDY       ARGLO
000250                  CPY       FACLO
000251 SAVQUO:          PHP
000252                  ROL       A                         ;SAVE RESULT.
000253                  BCC       QSHFT                     ;IF NOT DONE, CONTINUE.
000254                  INX
000255                  STA       RESLO,X
000256                  BEQ       LD100
000257                  BPL       DIVNRM                    ;NOTE THIS REQ 1 MO RAM THEN NECESS.
000258                  LDA       #1
000259 QSHFT:           PLP                                 ;RETURN CONDITION CODES.
000260                  BCS       DIVSUB                    ;FAC .LE. ARG.
000261 SHFARG:          ASL       ARGLO                     ;SHIFT ARG ONE PLACE LEFT.
000262                  ROL       ARGMO
000263                  ROL       ARGMOH
000264                  ROL       ARGHO
000265                  BCS       SAVQUO                    ;SAVE A RESULT OF ONE FOR THIS POSITION
000266 ;AND DIVIDE.
000267                  BMI       DIVIDE                    ;IF MSB ON, GO DECIDE WHETHER TO SUB.
000268                  BPL       SAVQUO
000269 DIVSUB:          TAY                                 ;NOTICE C MUST BE ON HERE.
000270                  LDA       ARGLO
000271                  SBC       FACLO
000272                  STA       ARGLO
000273                  LDA       ARGMO
000274                  SBC       FACMO
000275                  STA       ARGMO
000276                  LDA       ARGMOH
000277                  SBC       FACMOH
000278                  STA       ARGMOH
000279                  LDA       ARGHO
000280                  SBC       FACHO
000281                  STA       ARGHO
000282                  TYA
000283                  JMP       SHFARG
000284 LD100:           LDA       #$40                      ;ONLY WANT TWO MORE BITS.
000285                  BNE       QSHFT                     ;ALWAYS BRANCHES.
000286 DIVNRM:          EQU       *
000287                  ROR       A
000288                  ROR       A
000289                  ROR       A
000290                  AND       #$C0
000291 ;GET LAST TWO BITS INTO MSB AND B6.
000292                  STA       FACOV
000293                  PLP                                 ;TO GET GARBAGE OFF STACK.
000294                  JMP       MOVFR                     ;MOVE RESULT INTO FAC, THEN
000295 ;NORMALIZE RESULTND RETURN.
000296 DV0ERR:          LDX       #ERRDV0
000297                  JMP       ERROR
000298                  PAGE
000299                  SBTL      "FLOATING POINT MOVEMENT ROUTINES."
000300 ;MOVE RESULT TO FAC.
000301 MOVFR:           LDA       RESHO
000302                  STA       FACHO
000303                  LDA       RESMOH
000304                  STA       FACMOH
000305                  LDA       RESMO
000306                  STA       FACMO
000307                  LDA       RESLO                     ;MOVE LO AND SGN.
000308                  STA       FACLO
000309                  JMP       NORMAL                    ;ALL DONE.
000310 ;MOVE MEMORY INTO FAC (UNPACKED).
000311 MOVFM:           STA       INDEX1
000312                  STX       INDEXB
```

```
000313                  STY       INDEX1+1
000314                  LDY       #4
000315                  LDA       #0                      ;IF MEMORY IS A VARIABLE THEN ONLY 4 BYTES.
000316                  CPX       #0                      ;VARIABLE?
000317                  BNE       *+4                     ;YES, STORE A ZERO IN FACLO.
000318                  LDA       (INDEX),Y               ;NO, STORE LOW BYTE IN FACLO.
000319                  STA       FACLO
000320                  DEY
000321                  LDA       (INDEX1),Y
000322                  STA       FACMO
000323                  DEY
000324                  LDA       (INDEX1),Y
000325                  STA       FACMOH
000326                  DEY
000327                  LDA       (INDEX1),Y
000328                  STA       FACSGN
000329                  ORA       #$80
000330                  STA       FACHO
000331                  DEY
000332                  LDA       (INDEX1),Y
000333                  STA       FACEXP                  ;LEAVE SWITCHES SET ON EXP.
000334                  STY       FACOV
000335                  RTS
000336 ;MOVE NUMBER FROM FAC TO MEMORY.
000337 MOV2F:           LDX       #TEMPF2
000338                  DFB       44
000339 MOV1F:           LDX       #TEMPF1
000340                  LDY       #0
000341                  TYA
000342                  BEQ       MOVMF                   ;ALWAYS BRANCHES.
000343 MOVVF:           LDX       FORPNT
000344                  LDA       FORPNTB
000345                  LDY       FORPNT+1
000346 MOVMF:           STX       INDEX1
000347                  STY       INDEX1+1
000348                  STA       INDEX1B
000349 FOURBYT          LDY       #4
000350                  TAX
000351                  BNE       FURBYT                  ;SAVING A VARIBLE--ROUND TO 4 BYTES AND STORE.
000352                  JSR       ROUND                   ;ROUND TO 5.
000353                  LDA       FACLO                   ;GET 5TH BYTE.
000354                  STA       (INDEX),Y
000355                  BNE       *+5                     ;IF ZERO THEN ROUNDER O.K.
000356 FURBYT           JSR       ROUNDER
000357                  DEY
000358                  LDA       FACMO
000359                  STA       (INDEX),Y
000360                  DEY
000361                  LDA       FACMOH
000362                  STA       (INDEX),Y
000363                  DEY
000364                  LDA       FACSGN                  ;INCLUDE SIGN IN FACHO
000365                  ORA       #$7F
000366                  AND       FACHO
000367                  STA       (INDEX),Y
000368                  DEY
000369                  LDA       FACEXP
000370                  STA       (INDEX),Y
000371                  STY       FACOV                   ;ZERO IT SINCE ROUNDED.
000372                  RTS                               ;Y=0.
000373 ROUNDER          JSR       ROUND
000374                  LDA       FACLO
000375                  BPL       RONDRTS
000376                  ASL       A                       ;KILL HIGH BIT
000377                  STA       TEMP                    ;ROUND UP IF NOT ALL ZEROS.
000378                  LDA       FACMO
000379                  AND       #1
000380                  ORA       TEMP                    ;ROUND UP IF NOT 0.
000381                  BEQ       RONDRTS
000382                  LDA       #$FF
000383                  STA       FACLO
000384                  JSR       INCRND                  ;INCRIMENT THE FAC.
000385 RONDRTS          LDA       #0
000386                  STA       FACLO
000387                  RTS
000388 ;MOVE ARG INTO FAC.
000389 MOVFA:           LDA       ARGSGN
000390 MOVFA1:          STA       FACSGN
000391                  LDX       #4+1
000392 MOVFAL:          LDA       ARGEXP-1,X
```

```
000393                  STA       FACEXP-1,X
000394                  DEX
000395                  BNE       MOVFAL
000396                  STX       FACOV
000397                  RTS
000398 ;MOVE FAC INTO ARG.
000399 MOVAF:           JSR       ROUND
000400 MOVEF:           LDX       #5+1
000401 MOVAFL:          LDA       FACEXP-1,X
000402                  STA       ARGEXP-1,X
000403                  DEX
000404                  BNE       MOVAFL
000405                  STX       FACOV                     ;ZERO IT SINCE ROUNDED.
000406                  RTS
000407 ROUND:           LDA       FACEXP                    ;ZERO?
000408                  BEQ       RONRTS                    ;YES. DONE ROUNDING.
000409                  ASL       FACOV                     ;ROUND?
000410                  BCC       RONRTS                    ;NO. MSB OFF.
000411 INCRND           JSR       INCFAC                    ;YES, ADD ONE TO LSB(FAC).
000412                  BNE       RONRTS                    ;NO CARRY MEANS DONE.
000413                  JSR       RNDSHF                    ;SQUEEZ MSB IN AND RTS.
000414 RONRTS           RTS
000415 ;NOTE: C=1 since INCFAC doesn't touch C.
000416                  PAGE
000417                  SBTL      "SIGN, SGN, FLOAT, NEG, ABS."
000418 ;PUT SIGN OF FAC IN ACCA.
000419 SIGN:            LDA       FACEXP
000420                  BEQ       SIGNRT                    ;IF NUMBER IS ZERO, SO IS RESULT.
000421 FCSIGN:          LDA       FACSGN
000422 FCOMPS:          ROL       A
000423                  LDA       #$100-1                   ;ASSUME NEGATIVE.
000424                  BCS       SIGNRT
000425                  LDA       #1                        ;GET .
000426 SIGNRT:          RTS
000427 ;SGN FUNCTION.
000428 SGN:             JSR       SIGN
000429 ;FLOAT THE SIGNED INTEGER IN ACCA.
000430 FLOAT:           STA       FACHO                     ;PUT ACCA IN HIGH ORDER.
000431                  LDA       #0
000432                  STA       FACHO+1
000433                  LDX       #$88                      ;GET THE EXPONENT.
000434 ;FLOAT THE SIGNED NUMBER IN FAC.
000435 FLOATS:          LDA       FACHO
000436                  EOR       #$FF
000437                  ROL       A                         ;GET COMP OF SIGN IN CARRY.
000438 FLOATC:          LDA       #0                        ;ZERO ACCA BUT NOT CARRY.
000439                  STA       FACLO
000440                  STA       FACMO
000441                  STX       FACEXP
000442                  STA       FACOV
000443                  STA       FACSGN
000444                  JMP       FADFLT
000445 ;ABSOLUTE VALUE OF FAC.
000446 ABS:             LSR       FACSGN
000447                  RTS
000448                  PAGE
000449                  SBTL      "COMPARE TWO NUMBERS."
000450 ;A=1 IF ARG .LT. FAC.
000451 ;A=0 IF ARG=FAC.
000452 ;A=-1 IF ARG .GT. FAC.
000453 FCOMPARG         LDA       #25
000454                  STA       KIMY                      ;FIRST 24 BITS MUST MATCH TO BE EQUAL.
000455 FCOMPA1          LDA       FACSGN
000456                  EOR       ARGSGN                    ;ARE THE SIGNS DIFFERENT?
000457                  BMI       FCSIGN                    ;YES, SO RESULT IS SIGN OF FAC AGAIN.
000458                  LDX       #5
000459 PHFAC34          LDA       FAC,X
000460                  PHA                                 ;SAVE FAC.
000461                  DEX
000462                  BPL       PHFAC34
000463                  JSR       FSUBT                     ;FIND THE DIFFERENCE.
000464                  LDY       FACSGN                    ;SIGN OF DIFFERENCE.
000465                  LDA       FACEXP
000466                  STA       TEMP
000467                  LDX       #$FA                      ;-6.
000468 PHFAC35          PLA
000469                  STA       FAC+6,X                   ;WORKS CUZ IT'S ZERO PAGE.
000470                  INX
000471                  BNE       PHFAC35                   ;RESTORE ORIGINAL FAC.
000472                  LDA       TEMP
```

```
000473                  BEQ       ISEQU37
000474                  LDA       FACEXP                    ;OLD EXPONENT.
000475                  SEC
000476                  SBC       TEMP                      ;NEW EXPONENT
000477                  BCC       NTEQU36                   ;IF NEW EXPONENT MUCH LESS THAN
000478                  CMP       KIMY                      ;OLD THEN THE RESULT WAS (NEARLY) ZERO.
000479                  BCS       ISEQU37
000480 NTEQU36          TYA                                 ;SIGN OF DIFFERENCE.
000481                  EOR       #$FF                      ;GET THE RESULT DEPENDING ON SIGN.
000482                  JMP       FCOMPS
000483 ISEQU37          LDA       #0
000484                  RTS
000485 FCOMPN           LDA       INDEX2
000486                  LDX       #0                        ;THIS ENTRY FOR "NEXT";
000487 FCOMP            JSR       CONUPK
000488                  LDA       #33                       ;FIRST 32 BITS MUST MATCH TO BE EQUAL.
000489                  STA       KIMY
000490                  JMP       FCOMPA1
000491                  PAGE
000492                  SBTL      "GREATEST INTEGER FUNCTION."
000493 ;QUICK GREATEST INTEGER FUNCTION.
000494 ;LEAVES INT(FAC) IN FACHO&MO&LO SIGNED.
000495 ;ASSUMES FAC .LT. 223 = 8388608
000496 QINT:            LDA       #0
000497                  STA       BITS                      ;IN CASE ITS POSATIVE.
000498                  LDA       FACEXP
000499                  BEQ       CLRFAC                    ;IF ZERO, GOT IT.
000500                  SEC
000501                  SBC       #$A0                      ;GET NUMBER OF PLACES TO SHIFT.
000502                  BIT       FACSGN
000503                  BPL       QISHFT
000504                  TAX
000505                  LDA       #$FF
000506                  STA       BITS                      ;PUT 255 IN WHEN SHFTR SHIFTS BYTES.
000507                  JSR       NEGFCH                    ;TRULY NEGATE QUANTITY IN FAC.
000508                  TXA
000509 QISHFT:          LDX       #FAC
000510                  CMP       #$100-7
000511                  BPL       QINT1                     ;IF NUMBER OF PLACES .GE. 7
000512
000513 ; ##############################################################################################
000514 ; #   END OF FILE:  B3MATHL.TEXT
000515 ; #   LINES      :  506
000516 ; #   CHARACTERS :  21287
000517 ; ##############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 517   CHARACTERS: 21839
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3FINPM.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:26 PM
|  Modified: Wednesday, December 31, 1997             4:37:03 PM
|
+------------------------------------------------------------------------------

000001 ; ###########################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3FINPM.TEXT
000004 ; ###########################################################################################
000005
000006 ;  Shift 1 place at a time.
000007                JMP       SHFTR1                    ;START SHIFTING BYTES, THEN BITS.
000008 QINT1:         TAY                                 ;PUT COUNT IN COUNTER.
000009                LDA       FACSGN
000010                AND       #$80                      ;GET SIGN BIT.
000011                LSR       FACHO                     ;SAVE FIR SHIFTED BYTE.
000012                ORA       FACHO
000013                STA       FACHO
000014                JMP       ROLSHF                    ;SHIFT THE REST.
000015 ; GREATEST INTEGER FUNCTION.
000016 INT:           LDA       FACEXP
000017                CMP       #$98+8
000018                BCS       INTRTS                    ;FORGET IT.
000019                JSR       QINT
000020                STY       FACOV                     ;CLR OVERFLOW BYTE.
000021                LDA       FACSGN
000022                STY       FACSGN                    ;MAKE FAC LOOK POSITIVE.
000023                EOR       #$80                      ;GET COMPLEMENT OF SIGN IN CARRY.
000024                ROL       A
000025                LDA       #$98+8
000026                STA       FACEXP
000027                LDA       FACLO
000028                STA       INTEGR
000029                JMP       FADFLT
000030 CLRFAC:        STA       FACHO                     ;MAKE IT REALLY ZERO.
000031                STA       FACMOH
000032                STA       FACMO
000033                STA       FACLO
000034                TAY
000035 INTRTS:        RTS
000036                SBTL      "FLOATING POINT INPUT ROUTINE."
000037 ; Procedure: FIN
000038 ; On Entry: TXTPTR points to the 1st character in a text buffer.
000039 ; Function: packs the digits into FAC as an Integer & keeps track of
000040 ;           where the decimal point is.
000041 ;      DPTFLG tells whether a decimal point has been seen.
000042 ;      DECCNT is the number of digits after the decimal point.
000043 ; On Exit: DECCNT and the exponent are used to determine how many
000044 ;          times to multiply or divide by 10 to get the correct number
000045 FIN:           JSR       CHRGOT
000046                LDY       #0                        ;Zero FACSGN & SGNFLG.
000047                STY       CNTDIGS                   ;ONLY COUNT THE DIGITS AFTER THE DECIMAL POINT
000048                STY       ANYNUM                    ;LOOK FOR ANY DIGIT ANYWHERE.
000049                LDX       #10                       ;ZERO FAC AND ALL THE REST.
000050 FINZLP:        STY       DECCNT,X                  ;ZERO MO AND LO.
000051                DEX                                 ;ZERO TENEXP AND EXPSGN
000052                BPL       FINZLP                    ;ZERO DECCNT, DPTFLG.
000053                BCC       FINDGQ                    ;FLAGS STILL SET FROM CHRGET.
000054                CMP       #$2D                      ;A NEGATIVE SIGN?
000055                BNE       QPLUS                     ;NO, TRY PLUS SIGN.
000056                STX       SGNFLG                    ;IT'S NEGATIVE. (X=$FF).
000057                BEQ       FINC                      ;ALWAYS BRANCHES.
000058 QPLUS:         CMP       #$2B                      ;PLUS SIGN?
000059                BNE       FIN1                      ;YES, SKIP IT.
000060 FINC:          JSR       CHRGET
000061                DEC       CNTDIGS                   ;ENOUGH DIGITS AFTER THE DECIMAL POINT
000062                BEQ       FINEND
000063 FINDGQ:        BCC       FINDIG
000064 FIN1:          CMP       #'.'                      ;THE DP?
000065                BEQ       FINDP                     ;NO KIDDING.
000066                EOR       #'E'                      ;EXPONENT FOLLOWS.
000067                AND       #$DF                      ;KILL $20 BIT SO LOWER=UPPER.
000068                BNE       FINE                      ;NO.
000069                STA       CNTDIGS                   ;AS MANY DIGITS AS YOU WANT AFTER AN 'E'
000070 ;HERE TO CHECK FOR SIGN OF EXP.
000071                LDA       ANYNUM
000072                BEQ       BADNMB
```

```
000073                JSR       CHRGET                    ;YES. GET ANOTHER.
000074                BCC       FNEDG1                    ;It's a digit. (Easier than backing up Pointer)
000075                CMP       #'-'                      ;MINUS?
000076                BEQ       FINEC1                    ;NEGATE.
000077                CMP       #'+'                      ;PLUS?
000078                BEQ       FINEC
000079                BNE       FINEC2
000080 FINEC1:        ROR       EXPSGN                    ;TURN IT ON.
000081 FINEC:         JSR       CHRGET                    ;GET ANOTHER.
000082 FNEDG1:        BCC       FINEDG                    ;IT IS A DIGIT.
000083 FINEC2:        BIT       EXPSGN
000084                BPL       FINE
000085                LDA       #0
000086                SEC
000087                SBC       TENEXP
000088                JMP       FINE1
000089 FINDP:         ROR       DPTFLG
000090                LDA       #10
000091                STA       CNTDIGS
000092                BIT       DPTFLG
000093                BVC       FINC
000094 FINE:          LDA       TENEXP
000095 FINE1          SEC
000096                SBC       DECCNT                    ;GET NUMBER OF PLACES TO SHIFT.
000097                STA       TENEXP
000098                BEQ       FINQNG                    ;NEGATE?
000099                BPL       FINMUL                    ;POSITIVE SO MULTIPLY.
000100 FINDIV:        JSR       DIV10
000101                INC       TENEXP                    ;DONE?
000102                BNE       FINDIV                    ;NO.
000103                BEQ       FINQNG                    ;YES.
000104 FINMUL:        JSR       MUL10
000105                DEC       TENEXP                    ;DONE?
000106                BNE       FINMUL                    ;NO
000107 FINQNG:        LDA       ANYNUM                    ;WERE ANY DIGITS TYPED?
000108                BEQ       BADNMB
000109                LDA       SGNFLG
000110                BMI       NEGXQS                    ;IF POSITE, RETURN.
000111                RTS
000112 NEGXQS:        JMP       NEGOP                     ;OTHERWISE, NEGATE AND RETURN.
000113 BADNMB         LDA       #$FF
000114                STA       ANYNUM
000115                RTS
000116 FINED1:        JSR       CHRGET                    ;SKIP THE REMAINING DIGITS.
000117 FINEND         BCC       FINED1
000118                BCS       FIN1
000119 FINDIG:        PHA
000120                BIT       DPTFLG
000121                BPL       FINDG1
000122                INC       DECCNT
000123 FINDG1:        JSR       MUL10
000124                PLA                                 ;GET IT BACK.
000125                INC       ANYNUM
000126                SEC
000127                SBC       #'0'
000128                JSR       FINLOG                    ;ADD IT IN.
000129                JMP       FINC
000130 FINLOG:        PHA
000131                JSR       MOVAF                     ;SAVE FAC FOR LATER.
000132                PLA
000133                JSR       FLOAT                     ;FLOAT THE VALUE IN ACCA.
000134                LDA       ARGSGN
000135                EOR       FACSGN
000136                STA       ARISGN                    ;RESULTANT SIGN.
000137                LDX       FACEXP                    ;SET SIGNS ON THING TO ADD.
000138                JMP       FADDT                     ;ADD TOGETHER AND RETURN.
000139 ;HERE PACK IN THE NEXT DIGIT OF THE EXPONENT.
000140 ;MULTIPLY THE OLD EXP BY 10 AND ADD IN THE NEXT
000141 ;DIGIT. NOTE: EXP OVERFLOW IS NOT CHECKED FOR.
000142 FINEDG:        LDA       TENEXP                    ;GET EXP SO FAR.
000143                CMP       #$A                       ;WILL RESULT BE .GE. 100?
000144                BCC       MLEX10
000145                LDA       #$64                      ;GET 100.
000146                BIT       EXPSGN
000147                BMI       MLEXMI                    ;IF NEG EXP, NO CHK FOR OVERR.
000148                JMP       OVERR
000149 MLEX10:        ASL       A                         ;MULT BY 2 TWICE
000150                ASL       A
000151                CLC                                 ;POSSIBLE SHIFT OUT OF HIGH.
000152                ADC       TENEXP                    ;LIKE MULTIPLYING BY FIVE.
```

```
000153                ASL       A                         ;AND NOW BY TEN.
000154                CLC
000155                LDY       #0
000156                ADC       (TXTPTR),Y
000157                SEC
000158                SBC       #'0'
000159 MLEXMI:         STA       TENEXP                    ;SAVE RESULT.
000160                JMP       FINEC
000161                SBTL      "FLOATING POINT OUTPUT ROUTINE."
000162 N.0999:         DFB       $91                       ; 99999.9499
000163                DFB       $43
000164                DFB       $4F
000165                DFB       $F9
000166                DFB       $99
000167 N.9999:         DFB       $94                       ; 999999.499
000168                DFB       $74
000169                DFB       $23
000170                DFB       $F8
000171                DFB       $00
000172 N.MIL:          DFB       $94                       ; 10E6
000173                DFB       $74
000174                DFB       $24
000175                DFB       $00
000176                DFB       0
000177 ;ENTRY TO LINPRT.
000178 INPRT:          LDA       #>INTXT
000179                LDY       #<INTXT
000180                LDX       #INTXTB
000181                JSR       STROUTR
000182                LDA       CURLIN+1
000183                LDX       CURLIN
000184 LINPRT:         STA       FACHO
000185                STX       FACHO+1
000186                LDX       #$90                      ;EXPONENT OF 16.
000187                SEC                                 ;NUMBER IS POSITIVE.
000188                JSR       FLOATC
000189                JSR       FOUT
000190                JMP       STROUTR                   ;PRINT AND RETURN.
000191 FOUT:           JSR       ROUNDER
000192                LDY       #1
000193 FOUTC:          EQU       *
000194                LDA       #$2D                      ;PRINT NOTHING IF POSITIVE,
000195                DEY                                 ;NEG SIGN IF NEGATIVE
000196                BIT       FACSGN
000197                BPL       FOUT1.1
000198                INY
000199                STA       FBUFFR-1,Y                ;STORE THE CHARACTER.
000200 FOUT1.1:         STA       FACSGN                    ;MAKE FAC POS FOR QINT.
000201                STY       FBUFPT                    ;SAVE FOR LATER.
000202                INY
000203                LDA       #'0'                      ;GET ZERO TO TYPE IF FAC=0.
000204                LDX       FACEXP
000205                BNE       *+5
000206                JMP       FOUT19
000207                LDA       #0
000208                CPX       #$80                      ;IS NUMBER .LT. 1.0 ?
000209                BEQ       FOUT37                    ;NO.
000210                BCS       FOUT7
000211 FOUT37:         LDA       #>N.MIL
000212                LDY       #<N.MIL                   ;MULTIPLY BY 106.
000213                JSR       FMULT
000214                LDA       #$100-6-0
000215 FOUT7:          STA       DECCNT                    ;SAVE COUNT OR ZERO IT.
000216 FOUT4:          LDA       #>N.9999
000217                LDX       #N.MILB
000218                LDY       #<N.9999
000219                JSR       FCOMP                     ;IS NUMBER .GT. 999999.499 ?
000220 ;OR 999999999.499?
000221                BEQ       BIGGES
000222                BPL       FOUT9                     ;YES. MAKE IT SMALLER.
000223 FOUT3:          LDA       #>N.0999
000224                LDX       #N.MILB
000225                LDY       #<N.0999
000226                JSR       FCOMP                     ;IS NUMBER .GT. 99999.9499 ?
000227 ; OR 99999999.9499?
000228                BEQ       FOUT38
000229                BPL       FOUT5                     ;YES. DONE MULTIPLYING.
000230 FOUT38:         JSR       MUL10                     ;MAKE IT BIGGER.
000231                DEC       DECCNT
000232                BNE       FOUT3                     ;SEE IF THAT DOES IT.
```

```
000233 ;THIS ALWAYS GOES.
000234 FOUT9:          JSR       DIV10                ;MAKE IT SMALLER.
000235                 INC       DECCNT
000236                 BNE       FOUT4                ;SEE IF THAT DOES IT.
000237 ;THIS ALWAYS GOES.
000238 FOUT5:          JSR       FADDH                ;ADD A HALF TO ROUND UP.
000239 BIGGES:         JSR       QINT
000240                 LDX       #1                   ;DECIMAL POINT COUNT.
000241                 LDA       DECCNT
000242                 CLC
000243                 ADC       #0*1+7               ;SHOULD NUMBER BE PRINTED IN E NOTATION?
000244 ;IE, IS NUMBER .LT. .01 ?
000245                 STA       ISARA                ;FOR PRINT USING.
000246                 BMI       FOUTPI               ;YES.
000247                 CMP       #0+$8                ;IS IT .GT. 999999 (999999999)?
000248                 BCS       FOUT6                ;YES. USE E NOTATION.
000249                 ADC       #$100-1              ;NUMBER OF PLACESEFORE DECIMAL POINT.
000250                 TAX                            ;PUT INTO ACCX.
000251                 LDA       #2                   ;NO E NOTATION.
000252 FOUTPI:         SEC
000253 FOUT6:          SBC       #2                   ;EFFECTIVELY ADD 5 TO ORIG EXP.
000254                 STA       TENEXP               ;THAT IS THE EXPONENT TO PRINT.
000255                 STX       DECCNT               ;NUMBER OF DECIMAL PLACES.
000256                 TXA
000257                 BEQ       FOUT39
000258                 BPL       FOUT8                ;SOME PLACES BEFORE DEC PNT.
000259 FOUT39:         LDY       FBUFPT               ;GET POINTER TO OUTPUT.
000260                 LDA       #'.'                 ;PUT IN '.'
000261                 INY
000262                 STA       FBUFFR-1,Y
000263                 TXA
000264                 BEQ       FOUT16
000265                 LDA       #'0'                 ;GET THE ENSUING ZERO.
000266                 INY
000267                 STA       FBUFFR-1,Y
000268 FOUT16:         STY       FBUFPT               ;SAVE FOLATER.
000269 FOUT8:          LDY       #0
000270                 LDX       #$80                 ;FIRST PASS THRU, ACCX HAS MSB SET.
000271 FOUT2:          LDA       FACLO
000272                 CLC
000273                 ADC       FOUTBL+2+1,Y
000274                 STA       FACLO
000275                 LDA       FACMO
000276                 ADC       FOUTBL+1+1,Y
000277                 STA       FACMO
000278                 LDA       FACMOH
000279                 ADC       FOUTBL+1,Y
000280                 STA       FACMOH
000281                 LDA       FACHO
000282                 ADC       FOUTBL,Y
000283                 STA       FACHO
000284                 INX                            ;IT WAS DONE YET ANOTHER TIME.
000285                 BCS       FOUT41
000286                 BPL       FOUT2
000287                 BMI       FOUT40
000288 FOUT41:         BMI       FOUT2
000289 FOUT40:         TXA
000290                 BCC       FOUTYP               ;CAN USE ACCA AS IS.
000291                 EOR       #$FF                 ;FIND 11.-A.
000292                 ADC       #$A                  ;CARRY STILL ON TO COMPLETE NEGATION.
000293 ;AND WILL ALWAYS BE ON AFTER.
000294 FOUTYP:         ADC       #'0'-1               ;GET A CHARACTER TO PRINT.
000295                 INY
000296                 INY
000297                 INY
000298                 INY                            ;BUMP POINTER UP.
000299                 STY       FDECPT
000300                 LDY       FBUFPT
000301                 INY                            ;POINT TO PLACE TO STORE OUTPUT.
000302                 TAX
000303                 AND       #$7F                 ;GET RID OF MSB.
000304                 STA       FBUFFR-1,Y
000305                 DEC       DECCNT
000306                 BNE       STXBUF               ;NOT TIME FOR DP YET.
000307                 LDA       #'.'
000308                 INY
000309                 STA       FBUFFR-1,Y           ;STORE DP.
000310 STXBUF:         STY       FBUFPT               ;STORE PNTR FOR LATER.
000311                 LDY       FDECPT
000312                 TXA                            ;COMPLEMENT ACCX
```

```
000313                  EOR       #$FF                    ;COMPLEMENT ACCA.
000314                  AND       #$80                    ;SAVONLY MSB.
000315                  TAX
000316                  CPY       #FDCEND-FOUTBL
000317                  BNE       FOUT2                   ;CONTINUE WITH OUTPUT.
000318                  LDY       FBUFPT                  ;GET BACK OUTPUT PNTR.
000319 FOUT11:           LDA       FBUFFR-1,Y              ;REMOVE TRAILING ZEROES.
000320                  DEY
000321                  CMP       #'0'
000322                  BEQ       FOUT11
000323                  CMP       #'.'
000324                  BEQ       FOUT12                  ;RUN INTO DP. STOP.
000325                  INY                               ;SOMETHING ELSE. SAVE IT.
000326 FOUT12:           LDA       #$2B
000327                  LDX       TENEXP
000328                  BEQ       FOUT17                  ;NO EXPONENT TO OUTPUT.
000329                  BPL       FOUT14
000330                  LDA       #0
000331                  SEC
000332                  SBC       TENEXP
000333                  TAX
000334                  LDA       #$2D                    ;EXPONENT IS NEGATIVE.
000335 FOUT14:           STA       FBUFFR-1+2,Y            ;STORE SIGN OF EXP
000336                  LDA       #'E'
000337                  STA       FBUFFR-1+1,Y            ;STORE THE 'E' CHARACTER.
000338                  TXA
000339                  LDX       #'0'-1
000340                  SEC
000341 FOUT15:           INX                               ;MOVE CLOSER TO OUTPUT VALUE.
000342                  SBC       #$A                     ;SUBTRACT 10.
000343                  BCS       FOUT15                  ;NOT NEGATIVE YET.
000344                  ADC       #'0'+$A                 ;GET SECOND OUTPUT CHARACTER.
000345                  STA       FBUFFR-1+4,Y            ;STORE HIGH DIGIT.
000346                  TXA
000347                  STA       FBUFFR-1+3,Y            ;STORE  LOW DIGIT.
000348                  LDA       #0                      ;PUT IN TERMINATOR.
000349                  STA       FBUFFR-1+5,Y
000350                  BEQ       FOUT20                  ;RETURN. (ALWAYS BRANCHES).
000351 FOUT19:           STA       FBUFFR-1,Y              ;STORE THE CHARACTER.
000352 FOUT17:           LDA       #0                      ;A TERMINATOR.
000353                  STA       FBUFFR-1+1,Y
000354 FOUT20:           LDA       #>FBUFFR
000355                  LDX       #0
000356                  LDY       #<FBUFFR
000357                  RTS                               ;ALL DONE.
000358 FHALF:            DFB       $80                     ;1/2
000359 ZERO:             DFB       000
000360                  DFB       000
000361                  DFB       000
000362                  DFB       0
000363                  DFB       0,0,0,0                 ;PTRGET POINTS TO ZERO WHEN IT DOESN'T CREATE.
000364 ;POWER OF TEN TABLE
000365 FOUTBL:           EQU       *
000366                  DFB       $FF,$FE,$79,$60         ;-100,000
000367                  DFB       $0,0,$27,$10            ;10,000
000368                  DFB       $FF,$FF,$FC,$18         ;-1,000
000369                  DFB       0,0,0,$64               ;100
000370                  DFB       $FF,$FF,$FF,$F6         ;-10
000371                  DFB       0,0,0,1                 ;1
000372 FDCEND:           EQU       *
000373                  SBTL      "EXPONENTIATION AND SQUARE ROOT FUNCTION."
000374 ;SQUARE ROOT FUNCTION --- SQR(A)
000375 ;USE SQR(X)=X.5
000376 SQR:              JSR       MOVAF                   ;MOVE FAC INTO ARG.
000377                  LDA       #>FHALF
000378                  LDX       #FHALFB
000379                  LDY       #<FHALF
000380                  JSR       MOVFM                   ;PUT MEMORY INTO FAC.
000381 ;LAST THING FETCHED IS FACEXP. INTO ACCX.
000382 ; JMP FPWRT   ;FALL INTO FPW.
000383 ;EXPONENTIATION ---  XY.
000384 ;N.B.  00=1
000385 ;FIRST CHECK IF Y=0. IF SO, THE RESULT IS 1.
000386 ;NEXT CHECK IF X=0. IF SO THE RESULT IS 0.
000387 ;THEN CHECK IF X.GT.0. IF NOT CHECK THAT Y IS AN INTEGER.
000388 ;IF SO, NEGATE X, SO THAT LOG DOESN'T GIVE FCERR.
000389 ;IF X IS NEGATIVE AND Y IS ODD, NEGATE THE RESULT
000390 ;RETURNED BY EXP.
000391 ;TO COMPUTE THE RESULT USE XY=EXP((Y*LOG(X)).
000392 FPWRT:           BEQ       GOTOEXP                 ;IF FAC=0, JUST EXPONENTIATE THAT.
```

```
000393                  LDA       ARGEXP                    ;IS X=0?
000394                  BNE       FPWRT1
000395                  JMP       ZEROF1                    ;ZERO FAC.
000396 FPWRT1:          LDX       #TEMPF3
000397                  LDA       #TEMPF3B
000398                  LDY       #<TEMPF3                  ;SAVE FOR LATER IN A TEMP.
000399                  JSR       MOVMF
000400 ;Y=0 ALREADY. GOOD IN CASE NO ONE CALLS INT.
000401                  LDA       ARGSGN
000402                  BPL       FPWR1                     ;NO PROBLEMS IF X.GT.0.
000403                  JSR       INT                       ;INTEGERIZE THE FAC.
000404                  LDX       #5
000405 FPLP1            LDA       ARG,X
000406                  PHA
000407                  DEX
000408                  BPL       FPLP1
000409                  LDA       #TEMPF3
000410                  LDX       #TEMPF3B
000411                  LDY       #<TEMPF3                  ;GET ADDR OF COMPERAND.
000412                  JSR       FCOMP                     ;EQUAL?
000413                  BNE       FPLP3                     ;LEAVE X NEG. LOG WILL BLOW HIM OUT.
000414 ;A=-1 AND Y IS IRRELEVANT.
000415                  TYA                                 ;NEGATE X. MAKE POSITIVE.
000416                  LDY       INTEGR                    ;GET EVENNESS.
000417 FPLP3:           LDX       #$FA                      ;-6
000418                  STA       KIMY
000419 FPLP2            PLA                                 ;RESTORE ARG (CLOBBERED BY FCOMP).
000420                  STA       ARG+6,X
000421                  INX
000422                  BNE       FPLP2
000423                  LDA       KIMY
000424 FPWR1:           JSR       MOVFA1                    ;ALTERNATE ENTRY POINT.
000425                  TYA
000426                  PHA                                 ;SAVE EVENNESS FOR TER.
000427                  JSR       LOG                       ;FIND LOG.
000428                  LDA       #TEMPF3
000429                  LDY       #<TEMPF3                  ;MULTIPLY FAC TIMES LOG(X).
000430                  JSR       FMULT
000431                  JSR       EXP                       ;EXPONENTIATE THE FAC.
000432                  PLA
000433                  LSR       A                         ;IS IT EVEN?
000434                  BCC       NEGRTS                    ;YES. OR X.GT.0.
000435 ;NEGATE THE NUMBER IN FAC.
000436 NEGOP:           LDA       FACEXP
000437                  BEQ       NEGRTS
000438                  LDA       FACSGN
000439                  EOR       #255
000440                  STA       FACSGN
000441 NEGRTS:          RTS
000442 GOTOEXP:         JMP       EXP
000443
000444 ; ##############################################################################################
000445 ; #   END OF FILE:  B3FINPM.TEXT
000446 ; #   LINES      :  437
000447 ; #   CHARACTERS :  20543
000448 ; ##############################################################################################

+-----------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 448   CHARACTERS: 21095
|
+-----------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3EXPON.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:26 PM
|  Modified: Wednesday, December 31, 1997             4:37:03 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3EXPON.TEXT
000004 ; ###############################################################################################
000005
000006                  PAGE
000007                  SBTL      "EXPONENTIATION FUNCTION."
000008 ;FIRST SAVE THE ORIGINAL ARGUMENT AND MULTIPLY THE FAC BY
000009 ;LOG2(E). THE RESULT IS USED TO DETERMINE IF OVERFLOW
000010 ;WILL OCCUR SINCE EXP)=2(X*LOG2(E)) WHERE
000011 ;LOG2(E)=LOG(E) BASE 2. THEN SAVE THE INTEGER PART OF
000012 ;THIS TO SCALE THE ANSWER AT THE END. SINCE
000013 ;2Y=2INT(Y)*2(Y-INT(Y)) AND 2INT(Y) IS EASY TO COMPUTE.
000014 ;NOW COMPUTE 2(X*LOG2(E)-INT(X*LOG2(E)) BY
000015 ;P(LN(2)*(INT(X*LOG2(E))+1)-X) WHERE P IS AN APPROXIMATION
000016 ;POLYNOMIAL. THE RESULT IS THEN SCALED BY THE POWER OF 2
000017 ;PREVIOUSLY SAVED.
000018 LOGEB2:          DFB       $81                     ;LOG(E) BASE 2.
000019                  DFB       $38
000020                  DFB       $AA
000021                  DFB       $3B,$29
000022 EXPCON:          DFB       7                       ;DEGREE-1
000023                  DFB       $71                     ; .0000214987636
000024                  DFB       $34
000025                  DFB       $58
000026                  DFB       $3E
000027                  DFB       $56
000028                  DFB       $74                     ; .00014352314036
000029                  DFB       $16
000030                  DFB       $7E
000031                  DFB       $B3
000032                  DFB       $1B
000033                  DFB       $77                     ; .0013422634824
000034                  DFB       $2F
000035                  DFB       $EE
000036                  DFB       $E3
000037                  DFB       $85
000038                  DFB       $7A                     ; .0096140170119
000039                  DFB       $1D
000040                  DFB       $84
000041                  DFB       $1C
000042                  DFB       $2A
000043                  DFB       $7C                     ; .055505126860
000044                  DFB       $63
000045                  DFB       $59
000046                  DFB       $58
000047                  DFB       $0A
000048                  DFB       $7E                     ; .24022638462
000049                  DFB       $75
000050                  DFB       $FD
000051                  DFB       $E7
000052                  DFB       $C6
000053                  DFB       $80                     ; .69314718608
000054                  DFB       $31
000055                  DFB       $72
000056                  DFB       $18
000057                  DFB       $10
000058                  DFB       $81                     ; 1.0
000059                  DFB       $00
000060                  DFB       $00
000061                  DFB       $00
000062                  DFB       $00
000063 EXP:             EQU       *
000064                  LDA       #>LOGEB2
000065                  LDY       #<LOGEB2                ;MULTIPLY BY LOG(E) BASE 2.
000066                  JSR       FMULT
000067                  LDA       FACOV
000068                  ADC       #$50
000069                  BCC       STOLD
000070                  JSR       INCRND
000071 STOLD:           STA       OLDOV
000072                  JSR       MOVEF                   ;TO SAVE IN ARG WITHOUT ROUND.
```

```
000073                  LDA       FACEXP
000074                  CMP       #$88                    ;IF ABS(FAC) .GE. 128, TOO BIG.
000075                  BCC       EXP1
000076 GOMLDV:          JSR       MLDVEX                  ;OVERFLOW OR OVERFLOW.
000077 EXP1:            JSR       INT
000078                  LDA       INTEGR                  ;GET LOW PART.
000079                  CLC
000080                  ADC       #$81
000081                  BEQ       GOMLDV                  ;OVERFLOW OR OVERFLOW !!
000082                  SEC
000083                  SBC       #1                      ;SUBTRACT 1.
000084                  PHA                               ;SAVE A WHILE.
000085                  LDX       #4+1                    ;PREP TO SWAP FAC AND ARG.
000086 SWAPLP:          LDA       ARGEXP,X
000087                  LDY       FACEXP,X
000088                  STA       FACEXP,X
000089                  STY       ARGEXP,X
000090                  DEX
000091                  BPL       SWAPLP
000092                  LDA       OLDOV
000093                  STA       FACOV
000094                  JSR       FSUBT
000095                  JSR       NEGOP                   ;NEGATE FAC.
000096                  LDA       #>EXPCON
000097                  LDY       #<EXPCON
000098                  JSR       POLY
000099                  LDA       #0
000100                  STA       ARISGN                  ;MULTIPLY BY POSITIVE 1.0.
000101                  PLA                               ;GET SCALE FACTOR.
000102                  JSR       MLDEXP                  ;MODIFY FACEXP AND CHECK FOR OVERFLOW.
000103                  RTS                               ;HAS TO DO JSR DUE TO PULAS IN MULDIV.
000104                  SBTL      "POLYNOMIAL EVALUATOR, & RND NUM GENERATOR"
000105 ;EVALUATE P(X2)*X
000106 ;POINTER TO DEGREE IS IN Y,A.
000107 ;THE CONSTANTS FOLLOW THE DEGREE.
000108 ;FOR X=FAC, COMPUTE:
000109 ; C0*X+C1*X3+C2*X5+C3*X7+...+C(N)*X(2*N+1)
000110 POLYX:           STA       POLYPT
000111                  STY       POLYPT+1                ;RETAIN POLYNOMIAL POINTER FOR LATER.
000112                  JSR       MOV1F                   ;SAVE FAC IN FACTMP.
000113                  LDA       #TEMPF1
000114                  JSR       FMULT                   ;COMPUTE X2.
000115                  JSR       POLY1                   ;COMPUTE P(X2).
000116                  LDA       #TEMPF1
000117                  LDY       #<TEMPF1
000118                  JMP       FMULT                   ;MULTIPLY BY FAC AGAIN.
000119 ;POLYNOMIAL EVALUATOR.
000120 ;POINTER TO DEGREE IS IN Y,A.
000121 ;COMPUTE:
000122 ; C0+C1*X+C2*X2+C3*X3+C4*X4+...+C(N-1)*X(N-1)+C(N)*XN.
000123 POLY:            STA       POLYPT
000124                  STY       POLYPT+1
000125 POLY1:           JSR       MOV2F                   ;SAVE FAC.
000126                  STY       POLYPTB                 ;BANK # = 0.
000127                  LDA       (POLYPT),Y
000128                  STA       DEGREE
000129                  LDY       POLYPT
000130                  INY
000131                  TYA
000132                  BNE       POLY3
000133                  INC       POLYPT+1
000134 POLY3:           STA       POLYPT
000135                  LDY       POLYPT+1
000136 POLY2:           JSR       FMULT
000137                  LDA       POLYPT
000138                  LDY       POLYPT+1                ;GET CURRENT POINTER.
000139                  CLC
000140                  ADC       #4+1
000141                  BCC       POLY4
000142                  INY
000143 POLY4:           STA       POLYPT
000144                  STY       POLYPT+1
000145                  JSR       FADD                    ;ADD IN CONSTANT.
000146                  LDA       #TEMPF2
000147                  LDY       #<TEMPF2                ;MULTIY THE ORIGINAL FAC.
000148                  DEC       DEGREE                  ;DONE?
000149                  BNE       POLY2
000150                  RTS                               ;YES, RETURN.
000151 ; PSUEDO-RANDOM NUMBER GENERATOR.
000152 ;  If ARG=0, the last random number generated is returned.
```

```
000153 ;  If ARG .LT. 0, a new sequence of random numbers is started
000154 ;    using the argument.
000155 ;  To form the next random number in the sequence, multiply the
000156 ;  previous random number by a random constant and add in another
000157 ;  random constant. Then the High & Low bytes are switched, the
000158 ;  exponent is put where it will be shifted in by RMAL, & the
000159 ;  exponent in FAC is set to $80 so that the result will be less
000160 ;  than 1.  This is then normalized and saved for the next time.
000161 ;  The Hi and Low bytes were switched so there will be a random
000162 ;  chance of getting a number less than or greater than .5.
000163 RMOD.C          DFB       0,0,0,0
000164                 DFB       $7F,$FF,$FF,$FF
000165 RMUL.C          DFB       0,0,0,0
000166                 DFB       0,0,$41,$A7
000167 RSMAL.C         DFB       $62,$00,0,0
000168                 DFB       0
000169 RND             LDA       FACEXP                    ;ARGUMENT OF ZERO?
000170                 BEQ       GETRND
000171                 BIT       FACSGN
000172                 BPL       RNDMAK
000173                 LSR       FACSGN                    ;MAKE POSATIVE.
000174                 AND       #$1F                      ;PUT EXPONENT IN RANGE $80-$9F.
000175                 ORA       #$90
000176                 STA       FACEXP
000177                 JSR       CONV2LNG                  ;MAKE A LONG INT.
000178                 JSR       RNDAT                     ;CRANK IT THROUGH THE GENERATOR.
000179                 JSR       RNDIT
000180 RNDMAK          JSR       RNDIT
000181 GETRND          JSR       RNDONE
000182                 JSR       LMAKFLT
000183                 LDA       #>RSMAL.C
000184                 LDY       #<RSMAL.C
000185                 LDX       #0
000186                 JMP       FMULT
000187 RNDONE          LDA       #RNDX                     ;FETCH THE LAST NUMBER INTO FACT.
000188                 LDY       #<RNDX
000189                 LDX       #RNDXB
000190                 JMP       LDFACT
000191 RNDIT           JSR       RNDONE                    ;CRANK THE OLD NUMBER THROUGH ONCE.
000192 RNDAT           JSR       FACTOARG                  ;CRANK THE FACT THROUGH THE GENERATOR.
000193                 LDA       #>RMUL.C
000194                 LDY       #<RMUL.C
000195                 LDX       #RNDXB
000196                 JSR       LDFACT
000197                 JSR       LMULT
000198                 JSR       FACTOARG
000199                 LDA       #>RMOD.C
000200                 LDY       #<RMOD.C
000201                 LDX       #0
000202                 JSR       LDFACT
000203                 JSR       LREM                      ;DO THE MOD FUNCTION.
000204                 LDA       #RNDX
000205                 LDY       #<RNDX
000206                 LDX       #RNDXB
000207                 JMP       STFACT
000208                 PAGE
000209                 SBTL      "SINE, COSINE AND TANGENT FUNCTIONS."
000210 ;COSINE FUNCTION.
000211 ;USE COS(X)=SIN(X+PI/2)
000212 COS:            LDA       #>PI2
000213                 LDY       #<PI2                     ;PNTR TO PI/2.
000214                 JSR       FADD                      ;ADD IT IN.
000215 ;FALL INTO SIN.
000216 ;SINE FUNCTION.
000217 ;USE IDENTITIES TO GET FAC IN QUADRANTS I OR IV.
000218 ;THE FAC IS DIVIDED BY 2*PI & THE INTEGER PART IS IGNORED
000219 ;BECAUSE SIN(X+PI)=SIN(X). THEN ARGUMENT CAN BE COMPARED
000220 ;WITH PI/2 BY COMPARING THE RESULT OF THE DIVISION
000221 ;WITH PI/2/(2*PI)=1/4.
000222 ;IDENTITIES ARE THEN USED TO GET THE RESULT IN QUADRANTS
000223 ;I OR IV. AN APPROXIMATION POLYNOMIAL IS THEN USED TO
000224 ;COMPUTE SIN(X).
000225 SIN:            JSR       MOVAF
000226                 LDA       #>TWOPI
000227                 LDY       #<TWOPI                   ;GET PNTR TO DIVISOR.
000228                 LDX       ARGSGN                    ;GET SIGN OF RESULT.
000229                 JSR       FDIVF
000230                 JSR       MOVAF                     ;GET RESULT INTO ARG.
000231                 JSR       INT                       ;INTEGERIZE FAC.
000232                 LDA       #0
```

```
000233                  STA     ARISGN                  ;ALWAYS HAVE THE SAME SIGN.
000234                  JSR     FSUBT                   ;KEEP ONLY THE FRACTIONAL PART.
000235                  LDA     #>FR4
000236                  LDY     #<FR4                   ;GET PNTR TO 1/4.
000237                  JSR     FSUB                    ;COMPUTE 1/4-FAC.
000238                  LDA     FACSGN                  ;SAVE SIGN FOR LATER.
000239                  PHA
000240                  BPL     SIN1                    ;FIRST QUADRANT.
000241                  JSR     FADDH                   ;ADD 1/2 TO FAC.
000242                  LDA     FACSGN                  ;SIGN IS NEGATIVE?
000243                  BMI     SIN2
000244                  LDA     TANSGN
000245                  EOR     #255
000246                  STA     TANSGN                  ;QUADRANTS II AND III COME HERE.
000247 SIN1:            JSR     NEGOP                   ;IF POSITIVE, NEGATE IT.
000248 SIN2:            LDA     #>FR4
000249                  LDY     #<FR4                   ;POINTER TO 1/4.
000250                  JSR     FADD                    ;ADD IT IN.
000251                  PLA                             ;GET ORIGINAL QUADRANT.
000252                  BPL     SIN3
000253                  JSR     NEGOP                   ;IF NEGATIVE, NEGATE RESULT.
000254 SIN3:            LDA     #>SINCON
000255                  LDY     #<SINCON
000256                  JMP     POLYX                   ;DO APPROXIMATION POLYNOMIAL.
000257 ;TANGENT FUNCTION.
000258 TAN:             JSR     MOV1F                   ;MOVE FAC INTO TEMPORARY.
000259                  LDA     #0
000260                  STA     TANSGN                  ;REMEMBER WHETHER TO NEGATE.
000261                  JSR     SIN                     ;COMPUTE THE SIN.
000262                  LDX     #TEMPF3
000263                  LDA     #TEMPF3B
000264                  LDY     #<TEMPF3
000265                  JSR     MOVMF                   ;PUT SIGN INTO OTHER TEMP.
000266                  LDA     #TEMPF1
000267                  LDX     #TEMPF3B
000268                  LDY     #<TEMPF1
000269                  JSR     MOVFM                   ;PUT THIS MEMORY LOC INTO FAC.
000270                  LDA     #0
000271                  STA     FACSGN                  ;START OFF POSITIVE.
000272                  LDA     TANSGN
000273                  JSR     COSC                    ;COMPUTE COSINE.
000274                  LDA     #TEMPF3
000275                  LDY     #<TEMPF3                ;ADDRESS OF SINE VALUE.
000276                  JMP     FDIV                    ;DIVIDE SINE BY COSINE AND RETURN.
000277 COSC:            PHA
000278                  JMP     SIN1
000279 PI2:             DFB     $81                     ;PI/2
000280                  DFB     $49
000281                  DFB     $0F
000282                  DFB     $DA
000283                  DFB     $A2
000284 TWOPI:           DFB     $83                     ;2*PI.
000285                  DFB     $49
000286                  DFB     $0F
000287                  DFB     $DA
000288                  DFB     $A2
000289 FR4:             DFB     $7F                     ;1/4
000290                  DFB     $00
000291                  DFB     $00
000292                  DFB     $00
000293                  DFB     0
000294 SINCON:          DFB     5                       ;DEGREE-1.
000295                  DFB     $84                     ; -14.381383816
000296                  DFB     $E6
000297                  DFB     $1A
000298                  DFB     $2D
000299                  DFB     $1B
000300                  DFB     $86                     ; 42.07777095
000301                  DFB     $28
000302                  DFB     $07
000303                  DFB     $FB
000304                  DFB     $F8
000305                  DFB     $87                     ; -76.704133676
000306                  DFB     $99
000307                  DFB     $68
000308                  DFB     $89
000309                  DFB     $01
000310                  DFB     $87                     ; 81.605223690
000311                  DFB     $23
000312                  DFB     $35
```

```
000313                  DFB       $DF
000314                  DFB       $E1
000315                  DFB       $86                     ; -41.34170209
000316                  DFB       $A5
000317                  DFB       $5D
000318                  DFB       $E7
000319                  DFB       $28
000320                  DFB       $83                     ; 6.2831853070
000321                  DFB       $49
000322                  DFB       $0F
000323                  DFB       $DA
000324                  DFB       $A2
000325                  DFB       $A6
000326                  DFB       $D3
000327                  DFB       $C1
000328                  DFB       $C8
000329                  DFB       $D4
000330                  DFB       $C8
000331                  DFB       $D5
000332                  DFB       $C4
000333                  DFB       $CE
000334                  DFB       $CA
000335                  PAGE
000336                  SBTL      "ARCTANGENT FUNCTION."
000337 ;USE IDENTITIES TO GET ARG BETWEEN 0 AND 1 AND THEN USE AN
000338 ;APPROXIMATION POLYNOMIAL TO COMPUTE ARCTAN(X).
000339 ATN:              LDA       FACSGN                  ;WHAT IS SIGN?
000340                  PHA                               ;(MEANWHILE SAVE FOR LATER.)
000341                  BPL       ATN1
000342                  JSR       NEGOP                   ;IF NEGATIVE, NEGATE FAC.
000343 ;USE ARCTAN(X)=-ARCTAN(-X) .
000344 ATN1:            LDA       FACEXP
000345                  PHA                               ;SAVE THIS TOO FOR LATER.
000346                  CMP       #$81                    ;SEE IF FAC .GE. 1.0 .
000347                  BCC       ATN2                    ;IT IS LESS THAN 1.
000348                  LDA       #>FONE
000349                  LDY       #<FONE                  ;GET PNTR TO 1.0 .
000350                  JSR       FDIV                    ;COMPUTE RECROCAL.
000351 ;USE ARCTAN(X)=PI/2-ARCTAN(1/X) .
000352 ATN2:            LDA       #>ATNCON
000353                  LDY       #<ATNCON                ;PNTR TO ARCTAN CONSTANTS.
000354                  JSR       POLYX
000355                  PLA
000356                  CMP       #$81                    ;WAS ORIGINAL ARGUMENT .LT. 1 ?
000357                  BCC       ATN3                    ;YES.
000358                  LDA       #>PI2
000359                  LDY       #<PI2
000360                  JSR       FSUB                    ;SUBTRACT ARCTAGN FROM PI/2.
000361 ATN3:            PLA                               ;WAS ORIGINAL ARGUMENT POSITIVE?
000362                  BPL       ATN4                    ;YES.
000363                  JMP       NEGOP                   ;IF NEGATIVE, NEGATE RESULT.
000364 ATN4:            RTS                               ;ALL DONE.
000365 ATNCON:          DFB       $0B                     ;DEGREE-1.
000366                  DFB       $76                     ; -.0006847939119
000367                  DFB       $B3
000368                  DFB       $83
000369                  DFB       $BD
000370                  DFB       $D3
000371                  DFB       $79                     ; .004850942156
000372                  DFB       $1E
000373                  DFB       $F4
000374                  DFB       $A6
000375                  DFB       $F5
000376                  DFB       $7B                     ; -.01611170184
000377                  DFB       $83
000378                  DFB       $FC
000379                  DFB       $B0
000380                  DFB       $10
000381                  DFB       $7C                     ; .03420963805
000382                  DFB       $0C
000383                  DFB       $1F
000384                  DFB       $67
000385                  DFB       $CA
000386                  DFB       $7C                     ; -.05427913276
000387                  DFB       $DE
000388                  DFB       $53
000389                  DFB       $CB
000390                  DFB       $C1
000391                  DFB       $7D                     ; .07245719654
000392                  DFB       $14
```

```
000393                  DFB       $64
000394                  DFB       $70
000395                  DFB       $4C
000396                  DFB       $7D                     ; -.08980239538
000397                  DFB       $B7
000398                  DFB       $EA
000399                  DFB       $51
000400                  DFB       $7A
000401                  DFB       $7D                     ; .1109324134
000402                  DFB       $63
000403                  DFB       $30
000404                  DFB       $88
000405                  DFB       $7E
000406                  DFB       $7E                     ; -.1428398077
000407                  DFB       $92
000408                  DFB       $44
000409                  DFB       $99
000410                  DFB       $3A
000411                  DFB       $7E                     ; .1999991205
000412                  DFB       $4C
000413                  DFB       $CC
000414                  DFB       $91
000415                  DFB       $C7
000416                  DFB       $7F                     ; -.3333333157
000417                  DFB       $AA
000418                  DFB       $AA
000419                  DFB       $AA
000420                  DFB       $13
000421                  DFB       $81                     ; 1.0
000422                  DFB       $00
000423                  DFB       $00
000424                  DFB       $00
000425                  DFB       $00
000426
000427 ; ###############################################################################################
000428 ; #   END OF FILE:  B3EXPON.TEXT
000429 ; #   LINES      :  420
000430 ; #   CHARACTERS :  17948
000431 ; ###############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 431   CHARACTERS: 18500
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3FREER.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:26 PM
|  Modified: Wednesday, December 31, 1997             4:37:03 PM
|
+------------------------------------------------------------------------------

000001 ; ##############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3FREER.TEXT
000004 ; ##############################################################################################
000005
000006                  SBTL      "General Pointer Maintenance"
000007 TSTFRE           LDX       INDEX
000008                  LDY       INDEX+1
000009                  STX       GRBTOP
000010                  STY       GRBTOP+1
000011                  LDX       INDEXB
000012                  STX       GRBTOPB
000013 ***********************************************
000014 *   ENTRY CONDITIONS                          *
000015 *     A: LENGTH OF NEW ATOM (LOW)             *
000016 *     X: LENGTH OF NEW ATOM (HIGH)            *
000017 *     Y: DON'T CARE                           *
000018 *     GRBTOP: POINTER TO ATOM TO BE ADDED     *
000019 *                                             *
000020 *   EXIT CONDITIONS                           *
000021 *     A: UNKNOWN                              *
000022 *     X: UNKNOWN                              *
000023 *     Y: UNKNOWN                              *
000024 *                                             *
000025 ***********************************************
000026 ;FRESML    EQU   *             ;This used to be here but not referenced
000027                  STA       TEMP
000028                  CLC
000029                  ADC       GRBTOP
000030                  STA       HEADER                    ;HEADER WILL POINT TO STRING INFO SPACE.
000031                  LDA       #0
000032                  ADC       GRBTOP+1
000033                  LDY       GRBTOPB
000034                  JSR       FIXADC
000035                  STA       HEADER+1
000036                  STY       HEADERB
000037                  LDA       TEMP                      ;STUFF LENGTH INTO STRING INFO AREA.
000038                  LDY       #2
000039 FREIT1           STA       (HEADER),Y
000040                  LDA       #0                        ;FOR THE OTHER TWO BYTES.
000041                  DEY
000042                  BPL       FREIT1
000043                  RTS
000044 FIXSBC           BCS       *+4                       ;THIS ROUTINE USED BY PEOPLE SUBTRACTING
000045                  DEY                                 ;MEMORY POINTERS.
000046                  DEY
000047 FIXSB2           CMP       #MINPG                    ;BYTE 2 IS ALWAYS KEEPED IN THE RANGE
000048                  BCS       *+5                       ;MINPG THROUGH MAXPG (2 -- $82)
000049                  ADC       #MAXPG-MINPG              ;SO THAT THE ON-THE-FLY BANK SWITCHING WILL WORK O.K.
000050                  DEY
000051                  CMP       #MAXPG
000052                  BCC       *+5
000053                  SBC       #MAXPG-MINPG
000054                  INY
000055                  SEC
000056                  RTS
000057 FIXADC           BCC       *+4                       ;JUST LIKE FIXSBC ONLY USED FOR ADD OPERATIONS.
000058                  INY
000059                  INY
000060                  JSR       FIXSB2
000061                  CLC
000062                  RTS
000063 FIXYAX           EQU       *                         ;THIS ROUTINE ALLOWS MIXED SUBTRACTION,
000064                  PHP                                 ;SO THAT A REGULAR 16 BIT QUANTITY CAN
                                                                        BE SUBTACTED
000065                  PHA                                 ;FROM A BANK.PAGE.BYTE (MEMORY) POINTER.
000066                  TXA                                 ;Y,A = BANK,PAGE POINTER VALUES.
000067                  ASL       A                         ;X IS PACKED PAGE COUNT TO SUBTRACT.
000068                  BCC       *+3                       ;ON RETURN A IS RESULT OF SBC,
000069                  DEY                                 ;AND Y IS ADJUSTED PROPERLY.
000070                  LSR       A
000071                  STA       CHARAC
```

```
000072                PLA
000073                PLP
000074                SBC       CHARAC                ;CAN'T CLOBBER VITAL INFO.
000075                JMP       FIXSBC
000076 FIXXY          DEY                             ;X.Y=BANK.PAGE.  INC'S Y AND X IF NEEDED.
000077                CPY       #MINPG
000078                BCS       FIXRTSX
000079                LDY       #MAXPG-1
000080                DEX
000081                RTS
000082 FIXYX          EQU       *                     ;THIS ROUTINE JUST ENSURES X.Y ARE
000083                CPY       #MAXPG                ;BANK.PAGE POINTERS WITH VALUES
000084                BCC       FIXRTSX               ;IN THE ACCEPTABLE RANGE.
000085                PHA
000086                TYA
000087                SBC       #MAXPG-MINPG
000088                TAY
000089                PLA
000090                INX
000091 FIXRTSX        RTS
000092 FIXAYX         JSR       FIXSBC                ;ENTRY TO FIXAY WITH X SET TO
000093                PHA                             ;BANK OF SUBTRACTED POINTER.
000094                TYA                             ;DOES THE SUBTRACT AND RETURNS WITH
                                                                  A THE HIGH 8 BITS
000095                STX       CHARAC                ;OF RESULTING WORD.
000096                SBC       CHARAC
000097                TAY
000098                PLA
000099 FIXAY          EQU       *                     ;THIS ROUTINE DOES THE INVERSE OF FIXYAX.
000100                SEC                             ;IT ALLOWS Y.A AS BANK.PAGE POINTERS AND PACKS
000101                SBC       #MINPG                ;A INTO HIGH BYTE OF 16 BIT VALUE.
000102                ASL       A                     ;THUS IF YOU USE FIXSBC TO SUBRACT TWO MEMORY
000103                PHA                             ;POINTERS AND YOU WANT THE DIFFERENCE TO BE
000104                TYA                             ;A 16 BIT (RELATIVE) QUANTITY, JUST LOAD
000105                CMP       #$80                  ;Y.A WITH THE RESULT, JSR FIXAY, AND
000106                ROR       A                     ;CHECK TO MAKE SURE Y ENDS UP <2.
000107                TAY
000108                PLA
000109                ROR       A
000110                ADC       #MINPG                ;CARRY CLEAR.
000111                BCC       *+3
000112                INY
000113                RTS
000114 FIXYA          EQU       *                     ;THIS ROUTINE OPPOSITE OF FIXAY.
000115                PHP                             ;IT UNPACKS A INTO Y.A.
000116                ASL       A                     ;SO YOU CAN MAKE A REGULAR POINTER OUT
000117                PHA
000118                TYA                             ;OF A 16 BIT PACKED VALUE.
000119                ADC       #0
000120                TAY
000121                PLA
000122                LSR       A
000123                CMP       #MINPG
000124                BCS       *+5
000125                ADC       #MAXPG-MINPG
000126                DEY
000127                PLP
000128                RTS
000129                SBTL      "IF...THEN...ELSE"
000130 IF             LDA       #1                    ;ENTRY INTO LEVEL 1
000131                STA       LVLCNT                ;WE ARE STARTING A NEW IF
000132                LDA       #$20
000133                STA       VALTYP                ;MAKE FRMEVL FIGURE OUT VAL. TYPE
000134                JSR       FRMEVL                ;EVALUATE A FORMULA
000135                BIT       VALTYP                ;RESULT CAN NOT BE A STRING TYPE
000136                BPL       *+5
000137                JMP       MISERR
000138                JSR       CHRGOT                ;GET CURRENT CHAR
000139                CMP       #GOTOTK               ;IS IT A GOTO?
000140                BEQ       OKGOTO
000141                LDA       #THENTK               ;NO. IT MUST BE A THEN
000142                JSR       MSTESC
000143 OKGOTO         BIT       VALTYP                ;TYPE BCD
000144                BVC       EXPBYTC               ;NO, CHECK EXPONENT BYTE
000145                LDX       #0
000146                LDA       #>FAC
000147                LDY       #<FAC
000148                JSR       LORALL                ;A=0 IFF FAC=0
000149                BVS       ISATRUE
000150 EXPBYTC        LDA       FACEXP                ;0=FALSE.
```

```
000151 ISATRUE         BNE       DOCOND                  ;TRUE
000152                 LDY       #$FF                    ;FALSE! LOOK FOR AND MATCH NEXT ELSE
000153 ELSE1           INY                               ;POINT TO NEXT CHAR
000154                 LDA       (TXTPTR),Y              ;GET IT
000155                 BNE       NOTEOL                  ;NOT THE END OF ALINE
000156                 JMP       ADDON                   ;END OF LINE?
000157 NOTEOL          CMP       #IFTOKN                 ;IS IT AN IF?
000158                 BEQ       PLSONE
000159                 CMP       #ELSETK                 ;NO. IS IT AN ELSE?
000160                 BNE       ELSE1
000161                 DEC       LVLCNT
000162                 BNE       ELSE1
000163                 JSR       ADDON                   ;POINT TO THE NEXT CHAR
000164                 JSR       CHRGET                  ;IF IT IS A #, THEN GOTO IT
000165                 BCC       DOCOND1                 ;DO A GOTO
000166                 BCS       DOCO
000167 PLSONE          INC       LVLCNT
000168                 BCS       ELSE1                   ;BRANCH ALWAYS TAKEN
000169 DOCOND          LDA       #$0
000170                 STA       LVLCNT                  ;RESET NEST COUNTER
000171                 JSR       CHRGOT                  ;IF A DIGIT, THEN GO TO IT.
000172                 BCS       DOCO                    ;IF C SET, THEN INTERPRET NEW STMNT
000173 DOCOND1         JMP       GOTO
000174 DOCO            PLA                               ;STRIP NEWSTT ADDRESS
000175                 PLA
000176                 JSR       DECTPT                  ;BACK UP A LITTLE SO IT ADVANCES TO THE TOKEN
000177                 JMP       NWSTT
000178 WINDOW          JSR       GETBYT                  ;GET A NUMBER INTO X
000179                 STX       SLEFT
000180                 JSR       CHKCOM                  ;MUST HAVE COMMA.
000181                 JSR       GETBYT                  ;GET Y1.
000182                 STX       SBOTTOM
000183                 LDA       #TOTK                   ;MUST HAVE "TO".
000184                 JSR       MSTESC
000185                 JSR       GETBYT                  ;GET X2.
000186                 STX       SWIDTH
000187                 CPX       SLEFT                   ;MAKE X1<X2.
000188                 BCS       NOTIN
000189                 LDA       SLEFT                   ;SWITCH LEFT WITH WIDTH.
000190                 STX       SLEFT
000191                 STA       SWIDTH
000192 NOTIN           JSR       CHKCOM
000193                 JSR       GETBYT
000194                 STX       STOPS
000195                 CPX       SBOTTOM
000196                 BCC       WINDER                  ;MAKE Y1<Y2.
000197                 LDA       SBOTTOM
000198                 STX       SBOTTOM
000199                 STA       STOPS
000200 WINDER:         LDA       #1                      ;CLEAR WINDOW.
000201                 JSR       PRNACHAR
000202                 LDA       #$1A                    ;GOTO X Y.
000203                 JSR       PRNACHAR
000204                 LDA       SWIDTH
000205                 JSR       DOITOUT
000206                 LDA       SBOTTOM
000207                 JSR       DOITOUT
000208                 LDA       #3                      ;LOWER RIGHT
000209                 JSR       PRNACHAR
000210                 LDA       #$1A
000211                 JSR       PRNACHAR
000212                 LDA       SLEFT
000213                 JSR       DOITOUT
000214                 LDA       STOPS
000215                 JSR       DOITOUT
000216                 LDA       #2
000217                 JSR       PRNACHAR
000218                 LDX       SBOTTOM
000219                 LDA       #25
000220                 JMP       VWINDER                 ;HOP UP INTO THE WINDOW.
000221 DOITOUT:        SEC
000222                 SBC       #$1
000223                 BCS       *+4
000224                 LDA       #0
000225                 JMP       PRNACHAR
000226
000227 ; ##############################################################################################
000228 ; #   END OF FILE:  B3FREER.TEXT
000229 ; #   LINES      :  220
000230 ; #   CHARACTERS :  10485
```

```
000231 ; ##############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 231   CHARACTERS: 11037
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "LONGINT.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:37 PM
|  Modified: Wednesday, December 31, 1997             4:37:14 PM
|
+------------------------------------------------------------------------------

000001 ; ###########################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  LONGINT.TEXT
000004 ; ###########################################################################################
000005
000006                  SBTL      "LONG INTEGERS"
000007 * LINP.   LONG INTEGER INPUT ROUTINE.
000008 LINP             LDA       #$40
000009                  STA       VALTYP                    ;OUR RESULT WILL BE A LONG INTEGER.
000010                  LDA       #0
000011                  STA       KIMY                      ;SIGN STARTS HERE.
000012                  LDX       #7                        ;8 BYTES OF FAC TO BE ZERO.
000013 LZFAC1           STA       FAC,X
000014                  STA       ARG,X
000015                  DEX
000016                  BPL       LZFAC1
000017                  JSR       CHRGOT
000018                  BCC       LISNUM                    ;CC=NUMERIC.
000019                  CMP       #'+'
000020                  BEQ       LTRYN2                    ;+ O.K.
000021                  CMP       #'-'
000022                  BNE       LTRYDOT                   ;RETURN POINTING TO NON-NUMERIC.
000023                  LDA       #$FF
000024                  EOR       KIMY                      ;HIGH BIT SET IF MINUS.
000025                  STA       KIMY
000026                  JMP       LTRYN2                    ;-- TURNS OUT + THIS WAY.
000027 LISNUM           AND       #$F                       ;MAKE BINARY.
000028                  STA       YSAVE                     ;TEMP
000029                  JSR       LTIMES10                  ;MULTIPLY FAC BY 10.
000030                  JSR       LZIPARG                   ;ZERO OUT ARG (AND PART OF RES).
000031                  LDA       YSAVE
000032                  STA       ARG+7
000033                  JSR       LADDPTR
000034 LTRYN2           JSR       CHRGET                    ;NEXT GUY ALSO A NUM?
000035                  BCC       LISNUM
000036 LTRYDOT          CMP       #'.'                      ;ROUND AFTER A PERIOD.
000037                  BNE       LNUMDON
000038                  JSR       CHRGET
000039                  BCS       LNUMDON
000040                  CMP       #'5'                      ;ROUND UP?
000041                  BCC       LNUMSCN                   ;NO.
000042                  BNE       LROUNDUP                  ;YES, ROUND UP.
000043 LTRYN3           JSR       CHRGET                    ;GET AN OTHER BYTE.
000044                  BCS       LRNDEVN                   ;EXACTLY .50000... SO ROUND EVEN.
000045                  CMP       #'1'
000046                  BCC       LTRYN3                    ;GOT AN OTHER TRAILING ZERO.
000047 LROUNDUP         JSR       LINCFAC                   ;INC FAC.
000048 LNUMSCN          JSR       CHRGET                    ;NEXT CHAR.
000049                  BCC       LNUMSCN                   ;SKIP IT IF NUMERIC.
000050                  BCS       LNUMDON                   ;END ON NON-NUMERIC.
000051 LRNDEVN          LDA       FAC+7
000052                  AND       #1
000053                  BEQ       LNUMDON
000054                  JSR       LINCFAC
000055 LNUMDON          LDA       KIMY
000056                  BPL       LDONE
000057 LTWSCOMP         LDX       #7
000058 LTWSONE          LDA       FAC,X                     ;GET A BYTE.
000059                  EOR       #$FF                      ;MAKE EOR OF FAC.
000060                  STA       FAC,X
000061                  DEX
000062                  BPL       LTWSONE
000063 LINCFAC          LDX       #7
000064 LTWSCO2          INC       FAC,X
000065                  BNE       LDONE
000066                  DEX
000067                  BNE       LTWSCO2
000068                  INC       FAC
000069                  AND       #$7F
000070                  BNE       LDONE
000071 LOVINP           JMP       LOVERR
000072 LDONE            RTS                                 ;ALL DONE
```

```
000073 LTIMES10        EQU     *                       ;MULTIPLY FAC * 10.
000074                 JSR     LPTRS                   ;SET UP PTR1,PTR2,PTR3.
000075                 LDA     #>FAC
000076                 JSR     LSHFTL                  ;SHIFT FAC LEFT ONE.
000077                 BMI     LOVINP
000078                 JSR     FACTOARG
000079                 JSR     LSHFTL                  ;LEFT AGAIN.
000080                 BMI     LOVINP
000081                 JSR     LSHFTL
000082                 BMI     LOVINP
000083                 JSR     LADDPTR
000084                 RTS
000085 FACTOARG        PHA                             ;SAVE A.
000086                 LDX     #8
000087 FAC2AR2         LDA     FAC-1,X
000088                 STA     ARG-1,X
000089                 DEX
000090                 BNE     FAC2AR2
000091                 PLA
000092                 RTS
000093 LSGNPOS         LDY     #0
000094                 LDA     (PTR1),Y
000095                 EOR     (PTR2),Y
000096                 STA     INPFLG
000097                 LDA     (PTR1),Y
000098                 BPL     LSGNP2
000099                 LDA     PTR1
000100                 JSR     TWOSCOMP                ;TWOSCOMP OF (A.Y).
000101 LSGNP2          LDY     #0
000102                 LDA     (PTR2),Y
000103                 BPL     LSGNP3
000104                 LDA     PTR2
000105                 JSR     TWOSCOMP                ;CLOBBERS Y.
000106 LSGNP3          RTS
000107 *
000108 LRESPOS         LDA     RES-1                   ;OVERFLOW OF RES USED BY LMULT.
000109                 BNE     LOVERR                  ;RES SHOULD BE POSATIVE OR OVERFLOWED.
000110 LRESDIV         LDA     #>RES
000111                 STA     PTR3
000112 LRESDV          LDA     #>FAC
000113                 STA     PTR1                    ;EVERYTHING ENDS UP IN FAC.
000114                 LDA     RES
000115                 BMI     LOVERR
000116                 LDA     INPFLG                  ;ASSUMES Y=0
000117                 BPL     LRESP2
000118                 LDA     PTR3
000119                 JSR     TWOSCOMP
000120 LRESP2          EQU     *
000121                 LDY     #7
000122 LRES2F2         LDA     (PTR3),Y
000123                 STA     (PTR1),Y
000124                 DEY
000125                 BPL     LRES2F2
000126                 RTS
000127 *
000128 * UTILITIES
000129 LZIPRES         LDA     #0                      ;RES-1....RES+7=0
000130                 LDX     #8
000131 LZIP2           STA     RES-1,X
000132                 DEX
000133                 BPL     LZIP2
000134                 RTS
000135 STFACT          STA     INDEX
000136                 STX     INDEXB
000137                 STY     INDEX+1
000138                 LDY     #7
000139 STFAC2          LDA     FAC,Y
000140                 STA     (INDEX),Y
000141                 DEY
000142                 BPL     STFAC2
000143                 RTS
000144 LPTRS           EQU     *                       ;SET UP DEFAULT POIINTERS.
000145                 LDY     #0                      ;SETS OPERATION TO
000146                 STY     PTR1+1                  ;HAVE OPERANDS FAC,ARG
000147                 STY     PTR2+1                  ;AND RESULT IN FAC.
000148                 STY     PTR3+1
000149                 STY     PTR1B
000150                 STY     PTR2B
000151                 STY     PTR3B
000152                 LDA     #>FAC                   ;FAC=ARG+FAC
```

```
000153                  STA       PTR1                    ;ASSUMES FAC,ARG,RES IN PAGE 0.
000154                  STA       PTR3
000155                  LDA       #>ARG
000156                  STA       PTR2
000157                  RTS
000158 LADD             EQU       *
000159 * ADD FAC TO ARG WITH RESULT IN FAC.
000160                  JSR       LPTRS
000161 *LADDPTR.  (PTR3)=(PTR1)+(PTR2)
000162 LADDPTR         LDY       #7                      ;8 BYTES.
000163                  CLC
000164 LADD2            LDA       (PTR1),Y
000165                  ADC       (PTR2),Y
000166                  STA       (PTR3),Y
000167                  DEY
000168                  BPL       LADD2
000169                  BVS       LOVERR                  ;OVERFLOW IF V SET.
000170                  RTS
000171 LOVERR           LDX       #ERROV
000172                  JMP       ERROR                   ;OVERFLOW ERROR.
000173 * LONG SUBTRACT. FAC=ARG-FAC.
000174 LSUBB            JSR       LPTRS
000175 LSUBER           LDY       #7
000176                  SEC
000177 LSUB2            LDA       (PTR2),Y
000178                  SBC       (PTR1),Y
000179                  STA       (PTR3),Y
000180                  DEY
000181                  BPL       LSUB2
000182                  RTS
000183 *LSUB.  SUBTRACT FAC FROM ARG GIVING FAC.
000184 LSUB             JSR       LSUBB
000185                  BVS       LOVERR
000186                  RTS
000187 * TWOSCOMP.  MAKES (A.Y)=-(A.Y).
000188 TWOSCOMP         EQU       *
000189                  TAX
000190                  LDY       #7
000191 LEORIT           LDA       0,X
000192                  EOR       #$FF
000193                  STA       0,X
000194                  INX
000195                  DEY
000196                  BPL       LEORIT
000197 * INC (INDEX)
000198                  LDY       #7
000199 INCPTR2          DEX
000200                  INC       0,X
000201                  BNE       INCRT2
000202                  DEY
000203                  BPL       INCPTR2
000204 INCRT2           RTS
000205 *
000206 * LMULT.  MULTIPLY FAC BY ARG GIVING FAC.
000207 *
000208 LMULT            JSR       LPTRS                   ;SETS PRT1+1, PTR2+1, PTR3+1=0.
000209                  JSR       LSGNPOS                 ;MAKE SURE FAC,ARG POSATIVE.
000210                  LDA       #>FAC+7
000211                  STA       PTR1
000212                  LDA       #>ARG+7
000213                  STA       PTR2
000214                  LDA       #>RES+6                 ;MULTIPLY BYTE BY BYTE, STARTING
000215                  STA       PTR3                    ;AT FAC+7,ARG+7, PUTTING RESULT AT RES+6.
000216                  JSR       LZIPRES                 ;RESULT STARTS AT ZERO.
000217 LMULT1           LDY       #0
000218                  LDA       (PTR1),Y
000219                  BEQ       LMULT3                  ;IF BYTE IS ZERO THEN SKIP ROW.
000220 LMULT2           LDA       (PTR2),Y                ;IF BYTE IS ZERO THEN SKIP COLUMN.
000221                  BEQ       *+5
000222                  JSR       LMULTBYT                ;RETURNS WITH Y=0
000223                  DEC       PTR3                    ;RESULT SHOULD GO ONE TO THE LEFT NEXT TIME.
000224                  DEC       PTR2                    ;MULTIPLICAND POINTER.
000225                  LDA       PTR2
000226                  CMP       #ARG                    ;DONE IT ALL?
000227                  BCS       LMULT2
000228                  LDA       #>ARG+7
000229                  STA       PTR2                    ;RETURN COLUMN POINTER FOR NEXT ROW.
000230                  LDA       PTR3                    ;RETURN RESULT POINTER.
000231                  CLC
000232                  ADC       #8
```

```
000233                  STA     PTR3
000234 LMULT3           DEC     PTR3                    ;RESULT ONE LESS FOR EACH ROW.
000235                  DEC     PTR1                    ;ROW POINTER.
000236                  LDA     PTR1
000237                  CMP     #FAC
000238                  BCS     LMULT1                  ;DO NEXT ROW.
000239                  JMP     LRESPOS                 ;GIVE RESULT THE RIGHT SIGN AND PUT IN FAC.
000240 LMULTBYT         EQU     *                       ;MULTIPLIES BYTE AT (PTR1) BY (PTR2)
000241                  STY     INDEX                   ;AND PUTS THE RESULT IN (PTR3),(PTR3)+1.
000242                  STY     INDEX+1                 ;NEVER STORING BELOW RES.
000243                  LDA     (PTR1),Y                ;Y ASSUMED TO BE ZERO.
000244                  STA     KIMY                    ;TEMP.
000245                  LDX     #8
000246 LMULTB2          ROR     KIMY
000247                  BCC     LMULTB3
000248                  LDA     (PTR2),Y
000249                  CLC
000250                  ADC     INDEX
000251                  STA     INDEX
000252 LMULTB3          ROR     INDEX
000253                  ROR     INDEX+1
000254                  DEX
000255                  BNE     LMULTB2
000256                  LDA     PTR3
000257                  CMP     #RES-1
000258                  BCS     *+5                     ;DON'T STORE BELOW RES-1.
000259 LMULTOV          JMP     LOVERR
000260                  LDX     PTR3
000261                  LDA     INDEX+1
000262                  CLC
000263                  ADC     1,X
000264                  STA     1,X
000265                  LDA     INDEX                   ;Y ALMOST ALWAYS ZERO.
000266                  ADC     0,X
000267                  STA     0,X
000268                  BCC     LMULTBR
000269 LMULTB4          DEX
000270                  CPX     #RES-1
000271                  BCC     LMULTOV
000272                  INC     0,X
000273                  BEQ     LMULTB4
000274 LMULTBR          RTS
000275 *SHIFTING ROUTINES.
000276 LSHFTL           CLC                             ;SHIFT LEFT (A.Y) ONE BIT.
000277 LSHFTLC          TAX                             ;ENTRY FOR ROL.
000278                  LDY     #8
000279 LSHFT2           ROL     7,X
000280                  DEX
000281                  DEY
000282                  BNE     LSHFT2
000283                  LDY     8,X
000284                  RTS
000285 LSHFTR           EQU     *                       ;SHIFT (A.Y) RIGHT ONE BIT.
000286                  CLC
000287                  TAX
000288                  LDY     #8
000289 LSHFTR2          ROR     0,X
000290                  INX
000291                  DEY
000292                  BNE     LSHFTR2
000293                  RTS
000294 LSHLEIGT         EQU     *                       ;SHIFTS FAC LEFT 8 BITS.
000295                  LDX     #0                      ;RETURNS WITH X=0.
000296 LSHLE2           LDA     FAC+1,X
000297                  STA     FAC,X
000298                  INX
000299                  CPX     #7
000300                  BCC     LSHLE2
000301                  LDX     #0                      ;FOR DIVIDE?
000302                  STX     FAC+7                   ;ZERO LOW BYTE.
000303                  RTS
000304 LDIVER           EQU     *                       ;THIS DOES THE BASIC DIVIDE OPERATION FOR
000305                  JSR     LPTRS                   ;LDIV AND LREM.  SET UP THE POINTERS.
000306                  JSR     LZIPRES                 ;RESULT INTO RES. STARTS AT 0.
000307                  JSR     LSGNPOS                 ;MAKE OPERANDS POSATIVE.
000308                  JSR     LNRMFAC                 ;NORMALIZE FAC. (HIGH BIT OFF, BIT 6 ON).
000309                  LDX     KIMY                    ;NUMBER OF BITS LNRMFAC SHIFTED FAC.
000310                  STX     YSAVE                   ;WORK WITH THAT VALUE.
000311                  LDA     #>ARG
000312                  STA     PTR3
```

```
000313                  INC       YSAVE                    ;THIS WILL GET DECRIMENTED WITH EACH SHIFT.
000314                  BNE       LDIVCOM                  ;ALWAYS.
000315 LDECYSAV         EQU       *
000316                  LDA       RES-1
000317                  BPL       LRESOK                   ;DON'T SHIFT RES, UNLESS NON-ZERO.
000318                  LDA       #>RES                    ;SO LEADING ZEROS ARE DONE FASTER.
000319                  JSR       LSHFTLC                  ;SHIFT CARRY INTO LOW BIT.
000320 LRESOK           LDA       #>ARG
000321                  JSR       LSHFTL                   ;ASL ARG.
000322 LDIVCOM          JSR       LCOMP                    ;COMARE ARG TO FAC.
000323                  DEC       YSAVE
000324                  BMI       LGOTRES                  ;DONE?
000325                  BCC       LDECYSAV                 ;CARRY CLEAR-- ARG LESS.
000326                  JSR       LSUBER                   ;SUBTRACT ARG FROM FAC.
000327                  LDA       #$80                     ;SET "DONE A SUBTRACT" FLAG.
000328                  STA       RES-1
000329 ;CARRY STILL SET FROM SUB.
000330                  BCS       LDECYSAV                 ;ALWAYS.
000331 LGOTRES          RTS
000332 * HERE WHEN RESULT OF DIVIDE IN RES.
000333 *  REM IS IN ARG SHIFTED LEFT (KIMY) TIMES.
000334 LDIVT            JSR       LDIVER
000335 ;ALREADY GOT THE ROUND BIT IN CARRY.
000336                  BCC       LNOROUND
000337                  JSR       LSUBER
000338                  LDA       #>ARG
000339                  LDX       #0
000340                  LDY       #<ARG
000341                  JSR       LORALL                   ;OR ALL BYTES TOGETHER.
000342                  STA       YSAVE
000343                  LDA       RES+7
000344                  AND       #$1
000345                  ORA       YSAVE
000346                  BEQ       LNOROUND
000347                  LDX       #8
000348 LINCBYT          INC       RES-1,X
000349                  BNE       LNOROUND
000350                  DEX
000351                  BNE       LINCBYT
000352 * IF IT FALLS THROUGH HERE AN ERROR WILL EVENTUALLY RESULT.
000353 LNOROUND         JMP       LRESDIV
000354 LNRMFAC          LDY       #0
000355                  STY       KIMY
000356 LDIVE2           LDA       FAC
000357                  BNE       LDIVE3
000358                  JSR       LSHLEIGT                 ;SHIFT FAC LEFT 8.
000359                  LDA       KIMY
000360                  CLC
000361                  ADC       #8
000362                  STA       KIMY
000363                  CMP       #64                      ;FAC WAS ZERO?
000364                  BCC       LDIVE2
000365                  JMP       DV0ERR                   ;YES, DIVIDE BY ZERO.
000366 LDIVE3           LDA       #>FAC
000367                  BIT       FAC
000368                  BMI       LDIVTOFAR
000369                  BVS       LDIVOK
000370                  JSR       LSHFTL                   ;SHIFT FAC LEFT ONE.
000371                  INC       KIMY
000372                  BNE       LDIVE3                   ;ALWAYS
000373 LDIVTOFAR        JSR       LSHFTR
000374                  DEC       KIMY
000375 LDIVOK           RTS
000376 LCOMP            EQU       *                        ;COMPARE ARG TO FAC.
000377                  LDX       #0
000378 LCMP2            LDA       ARG,X                    ;LDA ARG
000379                  CMP       FAC,X                    ;CMP FAC.
000380                  BNE       LCMP3                    ;RETURN.
000381                  INX
000382                  CPX       #8
000383                  BCC       LCMP2
000384 LCMP3            RTS                                ;RETURNS WITH C,Z FLAGS SET PROPERLY.
000385 LDIV             EQU       *                        ;LONG DIVIDE.
000386                  JSR       LDIVER                   ;DO THE DIVIDE.
000387                  JMP       LRESDIV                  ;PUT RESULT INTO FAC.
000388 LREM             JSR       LDIVER                   ;DO DIVIDE OPERATION.
000389                  LDY       #0
000390                  LDA       #>ARG
000391 LREM2            JSR       LSHFTR
000392                  DEC       KIMY                     ;KIMY SET FROM LNRMFAC.
```

```
000393                  BPL       LREM2
000394                  STA       PTR3                    ;PTR3=ARG
000395                  JMP       LRESDV                  ;GOT THE RESULT.
000396 LORALL            STA       INDEX                   ;RETURN A ZERO IFF (A.Y) IS ALL ZERO.
000397                  STX       INDEXB
000398                  STY       INDEX+1
000399                  LDA       #0
000400                  LDY       #7
000401 LORAL2            ORA       (INDEX),Y
000402                  BNE       LGOTOR
000403                  DEY
000404                  BPL       LORAL2
000405                  TAY                               ;SET Z FLAG.
000406 LGOTOR            RTS
000407 LUNPACK           EQU       *                       ;PUTS UNPACKBCD(FAC) IN NUMSTR.
000408 *SETS HIGH BIT OF FACSGNN IF NEGATIVE.
000409                  LDA       FACEXP
000410                  CMP       #$80
000411                  LDA       #20
000412                  ROL       A                       ;HIGH BIT OF FACEXP NOW LOW BIT OF A.
000413                  STA       ISARA                   ;NOW LOOKS LIKE BCD EXPONENT.
000414                  LDA       FAC                     ;HIGH BYTE.
000415                  PHA
000416                  BPL       *+5                     ;DO TWOSCOMP IF NEGATIVE.
000417                  JSR       LTWSCOMP                ;TWOS COMP OF FAC INTO FAC.
000418                  LDA       #>FAC
000419                  LDX       #0
000420                  LDY       #<FAC
000421                  JSR       LORALL
000422                  BEQ       *+5
000423                  JSR       LNRMFAC                 ;NORMALIZE THE BEAST.
000424                  LDA       #64                     ;MAXIMUM SHIFTT.
000425                  SEC
000426                  SBC       KIMY                    ;# OF LEFT SHIFTS DONE BY LNRMFAC.
000427                  STA       KIMY
000428                  JSR       LZIPARG                 ;ZERO OUT ARG.
000429 LCONVBCD          LDA       #>FAC
000430                  JSR       LSHFTL                  ;CARRY = HIGH BIT.
000431                  LDX       #10
000432                  SED                               ;DECIMAL MODE!
000433 LADDBCD           LDA       ARG-1,X                 ;DOUBLE ARG AND ADD IN CARRY.
000434                  ADC       ARG-1,X
000435                  STA       ARG-1,X
000436                  DEX
000437                  BNE       LADDBCD
000438                  CLD                               ;BETTER CLEAR THAT SUCCER!
000439                  DEC       KIMY                    ;DONE ONE BIT, ALL DONE?
000440                  BNE       LCONVBCD
000441                  LDX       #10
000442 LABCD2            LDA       ARG-1,X                 ;MOVE THE RESULT INTO FACT.
000443                  STA       FACT-1,X                ;SO UUNPACK WILL WORK CORRECTLY.
000444                  DEX
000445                  BNE       LABCD2
000446                  JSR       UUNPACK
000447                  PLA
000448                  STA       FACSGN                  ;HIGH BIT ON IF # WAS NEG.
000449                  RTS
000450 LZIPARG           LDA       #0                      ;ZERO OUT 10 BYTES OF ARG.
000451                  LDX       #10                     ;ARG.....ARG+9 (REALLY RES+1).
000452 LZIPAR2           STA       ARG-1,X
000453                  DEX
000454                  BNE       LZIPAR2
000455                  RTS
000456 LOUT              EQU       *                       ;LONG INT. OUTPUT ROUTINE.
000457                  JSR       LUNPACK                 ;UNPACK TO BCD IN NUMSTR.
000458                  LDX       #0
000459                  LDA       #'0'
000460                  STA       NUMSTR                  ;AT LEAST ONE NUMBER.
000461                  LSR       ISARA                   ;NOW HAS # OF DIGITS.
000462                  LDY       #1
000463 LLOOKLUP          INY
000464                  DEC       ISARA                   ;DONE LAST DIGIT?
000465                  BMI       LISZERO                 ;YES, ALL DONE.
000466                  LDA       NUMSTR,Y                ;GET A BCD BYTE.
000467                  CPX       #0
000468                  BEQ       LOUT3
000469 LOUT5             ORA       #'0'
000470                  STA       NUMSTR,X
000471                  INX
000472                  BNE       LLOOKLUP
```

```
000473 LOUT3           STA       KIMY
000474                 BIT       FACSGN
000475                 BPL       LLOKLUP
000476                 LDA       #'-'
000477                 STA       NUMSTR,X
000478                 INX
000479 LLOKLUP         LDA       KIMY
000480                 BNE       LOUT5                ;ALWAYS.
000481 LISZERO         CPX       #1                   ;AT LEAST ONE BYTE TO BE OUTPUT.
000482                 BCS       *+4
000483                 LDX       #1
000484                 STX       LENUM
000485                 LDA       #0
000486                 STA       NUMSTR,X
000487                 RTS                            ;ALWAYS END WITH A NULL.
000488 LONGST0         LDA       #0                   ;STORE ZERO IN FAC.
000489                 LDX       #7
000490 LONGST          STA       FAC,X
000491                 DEX
000492                 BPL       LONGST
000493                 RTS
000494 LONGST1         EQU       *                    ;STORE 1 IN FAC.
000495                 JSR       LONGST0
000496                 LDA       #1
000497                 STA       FAC+7
000498                 RTS
000499 DMOVFM          EQU       *
000500 LDFACT          STA       INDEX                ;STORE (A.Y) IN FAC.
000501                 STX       INDEXB
000502                 STY       INDEX+1
000503                 LDY       #7
000504 LDFAC2          LDA       (INDEX),Y
000505                 STA       FAC,Y
000506                 DEY
000507                 BPL       LDFAC2
000508                 RTS
000509 CONV2FLT        BIT       VALTYP               ;CONVERT TO FLOAT.
000510                 BMI       STR2FLT
000511                 BVC       CONV2RTS
000512 * CONVERT FROM LONG INTEGER TO FLOAT.
000513 LMAKFLT         LDA       FAC
000514                 PHA
000515                 BPL       *+5
000516                 JSR       LTWSCOMP             ;CONVERT TO PLUS.
000517                 LDA       #>FAC
000518                 LDX       #0
000519                 LDY       #<FAC                ;IN THIS CASE 0.
000520                 JSR       LORALL               ;WAS ALL OF FAC ZERO?
000521                 BNE       *+6                  ;YES, GIVE HIM ZERO.
000522                 PLA                            ;CLEAN UP STACK FOR DONN
000523                 JMP       GIVE0
000524                 JSR       LNRMFAC              ;BIT 6 OF FAC NOW ON.
000525                 JSR       LSHFTL               ;BIT 7 ON, KIMY=# OF LEFT SHIFTS.
000526                 LDA       FAC+4
000527                 STA       FACOV                ;MIGHT AS WELL PUT THE BITS IN.
000528                 LDA       FAC+3
000529                 STA       FACLO
000530                 LDA       FAC+2
000531                 STA       FACMO
000532                 LDA       FAC+1
000533                 STA       FACMOH
000534                 LDA       FAC
000535                 STA       FACHO
000536                 LDA       #$80+$3F             ;MAX EXPONENT WE COULD GET.
000537                 SEC
000538                 SBC       KIMY
000539                 STA       FACEXP               ;NOW HAS THE CORRECT EXPONENT.
000540                 PLA
000541                 STA       FACSGN
000542                 LDA       #0
000543                 STA       VALTYP               ;RESULT IS ZERO.
000544 CONV2RTS        RTS                            ;ALL DONE.
000545 STR2FLT         JMP       VAL                  ;THAT WAS EASY!
000546 CONV2LNG        EQU       *                    ;CONVERT FAC TO LONG INTEGER (ROUNDS).
000547                 BIT       VALTYP               ;WHAT DO WE NEED TO CONVERT?
000548                 BMI       STR2LNG              ;STRING!
000549                 BVS       CONV2RT2             ;STARTED OUT LONG!
000550 * CONVERT FROM FLOATING POINT TO LONG INT.
000551                 JSR       QINTRN               ;ADD .5 AND TRUNCATE.
000552                 JSR       LZIPARG              ;ARG STARTS AT ZERO.
```

```
000553                LDA       FACSGN
000554                PHA
000555                LDA       FACEXP
000556                SEC
000557                SBC       #$80                ;GET # BITS TO SHIFT.
000558                BMI       GTFACNA
000559                STA       YSAVE
000560                CMP       #$40                ;CAN WE REALLY FIT IT IN?
000561                BCC       *+5                 ;YES
000562                JMP       OVERR               ;NO --- OVERFLOW.
000563                LDA       #>ARG
000564 PASSABIT        JSR       ASLFAC
000565                JSR       LSHFTLC             ;SHIFT ARG LEFT WITH CARRY.
000566                DEC       YSAVE
000567                BNE       PASSABIT
000568 GTFACNA         LDA       #$40
000569                STA       VALTYP              ;RESULT IS LONG INT.
000570                LDA       #>ARG
000571                LDX       #0
000572                JSR       LDFACT              ;MOV ARG TO FAC.
000573                PLA
000574                BPL       *+5
000575                JSR       LTWSCOMP
000576                RTS
000577 ASLFAC          EQU       *                   ;SHIFTS FLOATING FAC LEFT BY ONE.
000578                ASL       FACOV
000579                ROL       FACLO
000580                ROL       FACMO
000581                ROL       FACMOH
000582                ROL       FACHO
000583 CONV2RT2        RTS
000584 STR2LNG         LDA       #>LINP              ;WASN'T THAT EASY
000585                LDY       #<LINP
000586                JMP       VALSTR              ;JUST LIKE THE VAL FUNCTION.  ALMOST.
000587 CONV2INT        JSR       CONV2FLT            ;JUST LIKE CONVERTING TO FLOAT.
000588 QINTRN          JSR       FADDH               ;WITH A ROUND ON THE END.
000589                JMP       INT
000590 CONV2STR        EQU       *                   ;CONVERT THE FAC TO A STRING.
000591                BIT       VALTYP              ;WHAT WAS THE BEAST?
000592                BMI       CONV2RT2            ;STRING!
000593                BVS       CONWASL             ;LONG!
000594                JMP       STRS                ;SAME AS STR$ IN THIS CASE.
000595 CONWASL         JSR       LOUT                ;OUTPUT THE # INTTO THE BUFFER.
000596                LDA       #>NUMSTR
000597                LDX       #NUMSTRB
000598                LDY       #<NUMSTR
000599                JMP       STRLIT              ;MAKE THIS SUCCER A STRING.
000600 LAND            LDA       #>FAC               ;LOGICAL AND FOR LONG INT.
000601                LDX       #0
000602                LDY       #<FAC
000603                JSR       LORALL              ;WAS FAC ZERO?
000604                BNE       *+3
000605                RTS
000606                LDA       #>ARG
000607                LDX       #0
000608                LDY       #<ARG
000609                JSR       LORALL
000610 LGIVM1          BEQ       *+5
000611                JMP       LONGST1
000612 LGIVM0          JMP       LONGST0
000613 LONGOR          LDA       #>FAC
000614                LDY       #<FAC               ;IN THIS CASE ZERO.
000615                JSR       LORALL
000616                STA       KIMY                ;0 IFF FAC WAS ALL ZERO.
000617                LDA       #>ARG               ;Y STILL 0="<ARG".
000618                LDY       #<ARG
000619                LDX       #0
000620                JSR       LORALL              ;ARG ZERO?
000621                ORA       KIMY
000622                BEQ       LGIVM0
000623                JMP       LONGST1             ;RESULT 1
000624 LDOCOMP         EQU       *                   ;RETURNS WITH 1 OR ZERO BASED ON COMPARE.
000625                LDA       ARG
000626                CMP       FAC
000627                BMI       LISL
000628                BNE       LISG                ;SIGNED NUMBERS
000629                LDX       #1
000630                JSR       LCMP2               ;COMPARE THE REST
000631                JSR       LCOMP               ;LDA ARG, CMP FAC.
000632                BEQ       LISEQ
```

```
000633                BCS       LISG
000634 LISL           LDA       #4                      ;BIT ON IF <.
000635                DFB       44
000636 LISEQ          LDA       #2                      ;BIT ON IF =.
000637                DFB       44
000638 LISG           LDA       #1                      ;BIT ON IF >.
000639                AND       DOMASK
000640                BNE       LGIVM1                  ;GOT A MATCH! RETURN WITH A 1.
000641                BEQ       LGIVM0                  ;NO MATCH. ALWAYS. RETURN WITH 0.
000642
000643 ; ###############################################################################################
000644 ; #   END OF FILE:  LONGINT.TEXT
000645 ; #   LINES      :  636
000646 ; #   CHARACTERS :  27520
000647 ; ###############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 647   CHARACTERS: 28072
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3DMPYT.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:25 PM
|  Modified: Wednesday, December 31, 1997             4:37:02 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3DMPYT.TEXT
000004 ; ###############################################################################################
000005
000006                  SBTL      "DEC-HEX Converter" "
000007 DECER            JSR       LEN1                        ;0 VALTYP AND GET LENGTH OF STRING IN Y.
000008                  LDA       #0                          ;Put an Integer 0 in FAC.
000009                  STA       FACMO
000010                  STA       FACLO
000011                  LDX       #4
000012                  LDA       #$30                        ;PUT '0' IN MY WORK SPACE.
000013 STARTAT0         STA       RES-1,X
000014                  DEX
000015                  BNE       STARTAT0
000016                  LDX       #4                          ;PUT LAST 4 BYTES OF STRING IN WORKSPACE.
000017 PUTINUM          TYA
000018                  BEQ       GOTANUM
000019                  DEY
000020                  LDA       (INDEX1),Y
000021                  STA       RES-1,X
000022                  DEX
000023                  BNE       PUTINUM
000024 GOTANUM          LDY       #4
000025                  LDX       #2
000026 GETAHEX          LDA       RES-1,Y
000027                  JSR       DNNIB                       ;CONVERT NEXT LOWEST BYTE TO BINARY.
000028                  ORA       FACMO-1,X                   ;OR IT INTO FAC.
000029                  STA       FACMO-1,X
000030                  LDA       RES-1,Y                     ;GET NEXT BYTE.
000031                  JSR       DNNIB                       ;CONVERT TO BINARY.
000032                  ASL       A                           ;SHIFT LEFT 4.
000033                  ASL       A
000034                  ASL       A
000035                  ASL       A
000036                  ORA       FACMO-1,X
000037                  STA       FACMO-1,X
000038                  DEX
000039                  BNE       GETAHEX
000040                  LDA       FACMO
000041                  LDY       FACLO
000042                  JSR       GIVAYF
000043                  JMP       LEN0
000044 DNNIB            DEY                                   ;CONVERT BYTE IN FROM HEX TO BIN.
000045                  CMP       #'Z'+1
000046                  BCC       *+4
000047                  SBC       #$20                        ;MAKE LOWER=UPPER CASE.
000048                  SEC
000049                  SBC       #'0'
000050                  BCC       QNERR                       ;VALUE MUST BE BETWEEN 0 AND F.
000051                  CMP       #10
000052                  BCC       DECDN
000053                  SBC       #7
000054                  CMP       #$10                        ;GREATER THAN "F"?
000055                  BCS       QNERR                       ;YES, ILLEGAL QUANTITY.
000056                  CMP       #10                         ;LESS THAN "A"?
000057                  BCC       QNERR                       ;YES, ILLEGAL QUANTITY.
000058 DECDN            RTS
000059 HEXS             JSR       GETADR                      ;FORM AN ADDRESS FROM THE ARGUMENT
000060                  LDA       #4                          ;GET 4 BYTES OF
000061                  JSR       STRSPA                      ;STRING SPACE.
000062                  LDY       #3                          ;RESULT IS 4 BYTES.
000063                  LDX       #2                          ;CONVERT 2 BYTES.
000064 MAKHEX           LDA       FACMO-1,X                   ;GET A BYTE.
000065                  PHA                                   ;SAVE IT.
000066                  AND       #$0F                        ;KILL HIGH NIBBLE.
000067                  JSR       DONIB                       ;MAKE LOW NIBBLE HEX.
000068                  STA       (DSCTMP+1),Y                ;PUT IN STRING BUFFER.
000069                  DEY
000070                  PLA                                   ;GET BYTE BACK.
000071                  LSR       A                           ;KILL LOW NIBBLE.
000072                  LSR       A
```

```
000073                  LSR       A
000074                  LSR       A
000075                  JSR       DONIB
000076                  STA       (DSCTMP+1),Y
000077                  DEY
000078                  DEX
000079                  BNE       MAKHEX
000080                  JMP       PUTNEW                    ;GOT THE NEW STRING.
000081 DONIB            ORA       #$30
000082                  CMP       #$3A
000083                  BCC       GOTNIB
000084                  ADC       #6
000085 GOTNIB           RTS
000086 QNERR            LDX       #ERRFC
000087                  JMP       ERROR
000088                  SBTL      "SWAP CODE"
000089 ; Procedure: SWAP
000090 ;  This code exchanges the values of two variables.  This is an
000091 ;  extremely useful function in the case of Swapping strings as no
000092 ;  intermediate storage is required.  SWAP works with all types of
000093 ;  variables.  Types must be the same for both or a TYPE MISMATCH
000094 ;  ERROR will result.
000095 ;
000096 ;  SYNTAX:  SWAP A,B
000097 ;
000098 ;  The string descriptor:
000099 ;     DESRC    NAME    TYPE   STRNG   2 BYTE
000100 ;      LEN                     LEN    OFFSET
000101 ;
000102 ;  On entry: TXTPTR points past the SWAP token.
000103 ;  On Exit:  TXTPTR points to the end of statement terminator.
000104 ;           All Registers used.
000105 SWAP             JSR       PTRGET                    ;GET THE POINTER TO THE FIRST VARIABLE.
000106                  PHA                                 ;SAVE IT (LOW BYTE).
000107                  TYA
000108                  PHA                                 ;HIGH BYTE.
000109                  LDA       VARPNTB
000110                  PHA
000111                  LDA       ISARA                     ;IS IT AN ARRAY?
000112                  PHA
000113                  TXA                                 ;PTRGET SETS X TO VALTYP.
000114                  PHA                                 ;SAVE VALTYP
000115                  LDA       INTFLG
000116                  PHA                                 ;SAVE INTFLG
000117                  JSR       CHKCOM                    ;CHECK FOR PROPER SYNTAX: SWAP A,B.
000118                  JSR       PTRGET                    ;GET THE POINTER TO THE NEXT VAR.
000119                  PLA                                 ;COMPARE THE INTFLG OF BOTH VARS.
000120                  CMP       INTFLG
000121                  BEQ       SWAP1                     ;KEEP GOING IF THEY ARE =.
000122 MISERR           JMP       CHKERR                    ;TYPE MISMATCH!!
000123 SWAP1            PLA
000124                  CMP       VALTYP                    ;CHECK IF TYPES ARE THE SAME.
000125                  BNE       MISERR
000126                  PLA
000127                  STA       TEMP                      ;SIMPLE OR ARRAY INDICATOR
000128                  PLA
000129                  STA       INDEXB
000130                  PLA
000131                  STA       INDEX1+1                  ;PUT THE 1ST VAR POINTER IN INDEX1.
000132                  PLA
000133                  STA       INDEX1
000134                  LDY       VARNAM                    ;FORTUNATELY PTRGET PUT THE LEN HERE
000135                  DEY
000136 TRNSFR           LDA       (INDEX1),Y                ;GET SOMETHING FROM FIRST GUY.
000137                  TAX                                 ;SAVE IT TILL LATER
000138                  LDA       (VARPNT),Y                ;GET SOMETHING FROM SECOND GUY.
000139                  STA       (INDEX1),Y                ;PUT IN THE FIRST GUY'S PLACE.
000140                  TXA                                 ;RETRIEVE FIRST GUY'S DATA
000141                  STA       (VARPNT),Y                ;PUT IT IN SECOND GUY'S PLACE.
000142                  DEY
000143                  BPL       TRNSFR                    ;LOOP FOR MORE IF STILL POSITIVE.
000144                  LDX       VALTYP                    ;$FF FOR STRINGS
000145                  INX
000146                  BNE       SWPRTS
000147 *-THE REST IS FOR STRING SWAP
000148                  INY
000149                  LDA       (INDEX),Y                 ;LEN OF FIRST STRING
000150                  BNE       ONEISGD
000151                  LDA       (VARPNT),Y                ;LEN OF SECOND STRING
000152                  BEQ       SWPRTS                    ;BOTH NULLS - WE'RE DONE
```

```
000153 *--HERE WHEN 2ND IS NULL BUT NOT FIRST
000154                 LDA       VARPNT
000155                 PHA
000156                 LDA       VARPNT+1                ;SAVE POINTER TO THE
000157                 PHA                               ;  NOT NULL STRING
000158                 LDA       VARPNTB
000159                 PHA
000160                 LDA       ISARA                   ;  AND VARIABLE TYPE
000161                 PHA
000162                 JSR       TRNS                    ;SWAP 'VARPNT' & 'INDEX'
000163                 JMP       MAKNUL                  ;& GO SWAP A NULL
000164 ONEISGD          LDA       (VARPNT),Y
000165                 BNE       OLDSWAP                 ;NEITHER ARE NULLS, DO REGULAR SWAP
000166                 LDA       INDEX                   ;SAVE POINTER OF NOT NULL STRING
000167                 PHA
000168                 LDA       INDEX+1
000169                 PHA
000170                 LDA       INDEXB
000171                 PHA
000172                 LDA       TEMP                    ;  AND VARIABLE TYPE
000173                 PHA
000174 MAKNUL           JSR       INCNDX                  ;MOVE 'INDEX' TO BACKPOINTER
000175                 LDA       INDEX
000176                 STA       HIGHDS                  ;SET UIP HIGHDS FOR ROUTINE
000177                 LDA       INDEX+1                 ; TO FIX BACKPOINTER
000178                 STA       HIGHDS+1
000179                 LDA       INDEXB
000180                 STA       HIGHDSB
000181                 PLA
000182                 STA       ISARA
000183                 PLA
000184                 STA       FORPNTB
000185                 PLA                               ;SET UP FORPNT TO POINT TO
000186                 STA       FORPNT+1                ;  NOT NULL STRING
000187                 PLA
000188                 STA       FORPNT
000189                 JMP       FIXBAK                  ;GO FIX BACKPOINTER
000190 OLDSWAP          JSR       INCNDX                  ;GIVEN INDEX POINTING TO DESCRIPTOR THIS
000191 ;   ROUTINE MAKES INDEX POINT TO INFO BYTES OF THE STRING.
000192                 JSR       TRNS                    ;XFR INDEX TO VARPNT
000193                 JSR       INCNDX                  ;GET POINTER INTO SECOND STRING'S INFO BYTES.
000194                 LDY       #INFOSIZ-1
000195                 STY       VALTYP                  ;Y=1
000196                 BNE       TRNSFR                  ;ALWAYS
000197 SWPRTS           RTS
000198 TRNS             LDX       #1
000199                 LDA       INDEX,X                 ;SWAP INDEX WITH VARPNT.
000200                 LDY       VARPNT,X
000201                 STY       INDEX,X                 ;WITH STRINGS, THE INFO BYTES MUST
                                                                 ALSO BE SWAPED.
000202                 STA       VARPNT,X                ;SO WE MUST GET POINTERS TO BOTH INFO BYTES.
000203                 DEX
000204                 BPL       *-9
000205                 LDA       INDEXB
000206                 LDY       VARPNTB
000207                 STY       INDEXB
000208                 STA       VARPNTB
000209                 RTS
000210 INCNDX           JSR       NOTNOW                  ;MAKES INDEX POINT TO ACTUAL STRING.
000211                 CLC
000212                 ADC       INDEX                   ;ADD LENGTH OF STRING TO POINTER TO STRING
000213                 STA       INDEX                   ;TO GET POINTER TO INFO BYTES.
000214                 BCC       INCNDRTS
000215                 INC       INDEX+1
000216                 LDA       INDEX+1
000217                 CMP       #MAXPG                  ;ALL THIS BECAUSE OF SARA'S BANK SWITCHING.
000218                 BCC       INCNDRTS
000219                 SBC       #MAXPG-MINPG
000220                 INC       INDEXB
000221                 STA       INDEX+1
000222 INCNDRTS         RTS
000223
000224 ; ##############################################################################################
000225 ; #   END OF FILE:  B3DMPYT.TEXT
000226 ; #   LINES      :  217
000227 ; #   CHARACTERS :  10206
000228 ; ##############################################################################################
```

```
|  THAT'S ALL FOLKS!      LINES: 228   CHARACTERS: 10758
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3DIMNH.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:03:16 PM
|  Modified: Wednesday, December 31, 1997             4:37:01 PM
|
+------------------------------------------------------------------------------

000001 ; ##############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3DIMNH.TEXT
000004 ; ##############################################################################################
000005
000006                  SBTL      "DIMENSION AND VARIABLE SEARCHING."
000007 ; The DIM code puts X nonzero to use as DIMFLG, and then falls into the
000008 ;  variable routine, which looks at DIMFLG in three different points.
000009 ; 1) If an entry is found, DIMFLG being on indicates a Doubly
000010 ;    dimensioned variable.
000011 ; 2) When a new entry is being built, DIMFLG being on indicates that
000012 ;    the indices should be used for the size of each index.
000013 ;    Otherwise the default of 10 is used.
000014 ; 3) When build entry code finishes, indexing will be done only if
000015 ;    DIMFLG is on.
000016 DIM3:            JSR       CHKCOM                    ;Must be a comma
000017 DIM:             TAX                                 ;Set X nonzero so PTRGT1 will
000018                  LDA       #0                        ;  create an array.
000019                  STA       DORES                     ;SO ARRAY WILL BE CREATED
000020                  JSR       PTRGT1
000021                  JSR       CHRGOT                    ;GET LAST CHARACTER.
000022                  BNE       DIM3
000023                  RTS
000024 ;
000025 ; Routine to read variable name at the current Text Pointer Position
000026 ;
000027 ; On Entry: TXTPTR points to a variable name
000028 ;
000029 ; On Exit: VARPNT points to variable's value
000030 ;          TXTPTR points to terminator
000031 ;
000032 ; Note that evaluating in a variable name can cause recursive calls to
000033 ;   PTRGET, and at that point all values must be stored on the Stack.
000034 ;
000035 PTRGET           LDX       #0                        ;CREATE UNKNOWN ARRAYS.
000036 PTREVL           STX       DORES                     ;ALTERNATE ENTRY USED BY EVAL.
000037                  LDX       #$0                       ;TO TELL THAT WE ARE IN A
000038 PTRGT1           STX       DIMFLG                    ;DIM STATEMENT.
000039 PTRGT2           JSR       CHRGOT                    ;GET CURRENT CHARACTER.
000040                  JSR       ISLETC                    ;IF LETTER, CARRY IS SET.
000041                  BCS       PTRGT3
000042 WRDERR           JMP       SNERR                     ;MUST HAVE AN ALPHA! SYNTAX ERROR.
000043 PTRGT3           LDX       #$0
000044                  STX       INTFLG                    ;DEFAULT IS REAL NUMERIC.
000045                  STX       ISARA
000046                  LDA       TXTPTR
000047                  STA       LOWDS
000048                  STA       TMPPTR
000049                  LDA       TXTPTR+1
000050                  STA       LOWDS+1                   ;Save pointer to variable name in
000051                  STA       TMPPTR+1                  ;  LOWDS & TMPPTR.
000052                  LDA       TXTPTRB
000053                  STA       LOWDSB
000054                  STA       TMPPTRB
000055 EATEM            JSR       CHRGET                    ;GET NEXT CHARACTER.
000056                  BCC       EATEM                     ;GOBBLE NUMBERS.
000057                  JSR       ISLETC
000058                  BCS       EATEM                     ;GOBBLE ALPHA'S.
000059                  CMP       #'.'                      ;PERIODS OK IN VARIABLE NAMES.
000060                  BEQ       EATEM
000061                  CMP       #'!'                      ;FIND OUT VARIABLE TYPE.
000062                  BEQ       GOTTYP
000063                  CMP       #'&'+1                    ;IF > THEN IT IS A REAL.
000064                  BCS       REAL
000065                  CMP       #'$'
000066                  BCC       REAL
000067 GOTTYP           ADC       #$DF                      ;CARRY IS SET,RESULT WILL OVERFLOW.
000068                  TAX                                 ;THIS IS NOW AN INDEX.
000069                  BIT       SUBFLG
000070                  BMI       WRDERR                    ;FUNCTIONS ARE ONLY REALS.
000071                  JSR       CHRGET                    ;CHECK NEXT CHAR FOR A '('.
000072 REAL             LDY       VALTAB,X
```

```
000073                STY       VARNAM
000074                PHA
000075                LDA       TXTPTR
000076                SEC
000077                SBC       LOWDS
000078                CMP       #65
000079                BCS       WRDERR                ;BLOW UP. NAME TOO BIG!!
000080                LDY       VALTB2,X
000081                BNE       NTREAL                ;IS NOT A REAL VARIABLE.
000082                ADC       #$1                   ;C IS CLEAR.
000083 NTREAL         STA       FOUR6                 ;NAME LENGTH [+1].
000084                PLA
000085                BIT       SUBFLG
000086                BPL       YSTORE
000087                BVC       YSTORE
000088                LDY       #6
000089                STY       VARNAM                ;LENGTH FOR FN DEF. IS 6.
000090                LDY       #$10                  ;VALTYP FOR FUNCTIONS=$10 (SPECIAL CASE)
000091 YSTORE         STY       VALTYP                ;TYPE BYTE VALUE FROM THE TABLE.
000092                SEC
000093                ORA       SUBFLG                ;ALLOWS ARRAYS ONLY TO GET
000094                SBC       #'('                  ;TO ISARY.
000095                BNE       SPLVAR
000096 GARRAY         JMP       ISARY                 ;WE HAVE AN ARRAY HERE!!
000097 SPLVAR         BIT       SUBFLG
000098                BMI       SIMVAR
000099                BVS       GARRAY                ;GET AN ARRAY (FOR STORE OR RECALL).
000100 SIMVAR         CPY       #$C0                  ;IS IT FILE BUFF TYPE?
000101                BEQ       WRDERR                ;BLOW UP IF SO. FILBUFFS ARE ARRAYS ONLY
000102                LDY       #$0
000103                STY       SUBFLG                ;ALLOWS SUBSCRIPTS AGAIN.
000104                LDA       SMVARS                ;GET THE POINTER TO THE SIMPLE VARIABLES.
000105                LDX       SMVARS+1
000106                LDY       SMVARSB
000107                STY       SRCHPTB
000108 STXLOP         STX       SRCHPT+1              ;SAVE THE CURRENT WORKING POINTER.
000109 SRCHLP         STA       SRCHPT
000110                CMP       STREND                ;CHECK AGAINST END OF VARIABLES STORAGE.
000111                BNE       PNTER3
000112                CPX       STREND+1
000113                BNE       PNTER3
000114                LDY       SRCHPTB
000115                CPY       STRENDB
000116                BEQ       ADDVAR                ;VAR DOESN'T EXIST, SO CREATE IT!!
000117 PNTER3         LDY       #0
000118                LDA       (SRCHPT),Y            ;CURRENT ENTRY LENGTH.
000119                STA       VARNAM+1              ;SAVE IT IN HERE.
000120 COMPAR         LDA       (LOWDS),Y             ;VARIABLE NAME FROM TEXT.
000121                INY
000122                CMP       #'.'
000123                BCC       TYPCHK
000124                CMP       #'A'+$20              ;CHECK FOR LOWER CASE
000125                BCC       NOT2BIG
000126                CMP       #'Z'+1+$20
000127                BCS       TYPCHK
000128                SBC       #$1F                  ;CARRY CLEAR. NOW UPPER CASE.
000129 NOT2BIG        CMP       (SRCHPT),Y
000130                BEQ       COMPAR                ;AS LONG AS THEY MATCH.
000131 TYPCHK         LDA       VALTYP
000132                CMP       (SRCHPT),Y
000133                BNE       GONEXT                ;NOT THIS GUY, GO FURTHER.
000134                CPY       FOUR6                 ;CORRECT LENGTH?
000135                BNE       GONEXT                ;NOT THIS GUY, SORRY.
000136                TYA                             ;CARRY IS SET.
000137 FINUP          ADC       SRCHPT                ;FINAL CALCULATION TO
000138                BCC       CMPLTE                ;POINT TO THE VARIABLE
000139                INX
000140                CPX       #MAXPG
000141                BCC       *+7
000142                LDX       #MINPG
000143                INC       SRCHPTB
000144 CMPLTE         STX       VARPNT+1
000145                LDY       VARPNT+1
000146                STA       VARPNT
000147                LDX       SRCHPTB
000148                STX       VARPNTB
000149 MYMYMY         LDX       VALTYP
000150                CPX       #$80                  ;INTEGER TYPE?
000151                BNE       PTRRTS                ;NO, GOOD.
000152                STX       INTFLG                ;SET INTER FLAG.
```

```
000153                  LDX       #$0                     ;YES, CLEAR VALTYP.
000154                  STX       VALTYP
000155 PTRRTS           EQU       *
000156                  RTS
000157 GONEXT           CLC
000158                  LDA       VARNAM+1                ;THIS CODE FIXES
000159                  ADC       SRCHPT                  ;THE SEARCH POINTER
000160                  BCC       SRCHLP                  ;TO LOOK AT THE NEXT
000161                  INX                               ;VARIABLE NAME
000162                  CPX       #MAXPG
000163                  BCC       STXLOP
000164                  LDX       #MINPG
000165                  INC       SRCHPTB
000166                  BCS       STXLOP                  ;ALWAYS TAKEN.
000167 ; TEST FOR A LETTER. / CARRY OFF= NOT A LETTER.
000168 ;     CARRY ON= A LETTER.
000169 ISLETC:          CMP       #'A'
000170                  BCC       ISLRTS                  ;IF LESS THAN 'A', RET.
000171                  SBC       #'Z'+1
000172                  SEC
000173                  SBC       #$100-'Z'-1             ;RESET CARRY IF A .GT. 'Z'.
000174                  BCS       ISLRTS
000175                  CMP       #'A'+$20
000176                  BCC       ISLRTS
000177                  SBC       #'Z'+$21
000178                  SEC
000179                  SBC       #$100-'Z'-$21
000180 ISLRTS:          RTS                               ;RETURN TO CALLER.
000181 ;
000182 ; A looked for variable does not exist in the space.  Add it to
000183 ;   the variable tables.
000184 ADDVAR           SEC
000185                  LDA       FOUR6                   ;FETCH NAME LENGTH +1
000186                  STA       VARNAM+1                ;SAVE IT.
000187                  ADC       VARNAM                  ;C IS SET. ADD [# BYTES FOR TYPE] +1.
000188                  LDY       STREND+1
000189                  LDX       STRENDB
000190                  PHA                               ;TOTAL BYTES THIS ENTRY.
000191                  ADC       STREND                  ;CARRY IS CLEAR.
000192                  BCC       GOARND
000193                  INY                               ;STREND is crossing page bounds
000194                  CPY       #MAXPG                  ;Page <82?
000195                  BCC       GOARND
000196                  LDY       #MINPG                  ;NO! Page wraps to 2 and
000197                  INX                               ;  kick bank indicator up 1.
000198 GOARND           JSR       REASON                  ;FIND OUT IF THERE IS ENOUGH ROOM.
000199                  STY       STREND+1
000200                  STX       STRENDB
000201                  STA       STREND                  ;SAVE THE NEW 'END OF STORAGE' POINTER.
000202                  PLA                               ;RETRIEVE THE TOTAL ENTRY LENGTH.
000203                  LDY       #$0
000204                  STA       (SRCHPT),Y              ;ENTRY LENGTH.
000205                  TAY
000206                  DEY                               ;POINT TO THE ACTUAL LAST BYTE OF SPACE.
000207                  LDX       VARNAM                  ;TYPE LENGTH
000208                  LDA       #$0
000209 TILOOP           STA       (SRCHPT),Y              ;LOOP TO SET VARIABLE TO ZERO.
000210                  DEY
000211                  DEX                               ;TYPE LENGTH = TYPE LENGTH - 1.
000212                  BNE       TILOOP
000213                  LDA       VALTYP
000214                  STA       (SRCHPT),Y              ;STORE THE TYPE BYTE.
000215 TGHTLP           DEY                               ;BEGGINING OF NAME TRANSFER.
000216                  DEY
000217                  BMI       ALDONE                  ;GOES WHEN NAME IS TRANSFERED.
000218                  LDA       (LOWDS),Y               ;This loop upshifts the variable
000219                  CMP       #'A'+$20                ;  name as it transfers it from the
000220                  BCC       *+4                     ;  program to the space allocated for
000221                  SBC       #$20                    ;  it in the table.
000222                  INY
000223                  STA       (SRCHPT),Y
000224                  BPL       TGHTLP                  ;ALWAYS TAKEN.
000225 ALDONE           LDX       SRCHPT+1                ;GET THE POINTER INTO MEMORY.
000226                  LDA       VARNAM+1                ;GET THE # OF [NAME BYTES +1.] +1.
000227                  SEC                               ;SO WE INDEX TO THE FIRST BYTE
000228                  JMP       FINUP                   ;OF THE ACTUAL VARIABLE,
000229 ; VALTAB: table of # of bytes needed in descriptor (excluding link & name)
000230 ; VALTB2: table of variable TYPE bytes (same order as VALTAB)
000231 VALTAB          DFB       4,1,0,10                ;REAL,!,(FILLER),DBL PR
000232                  DFB       3,2,8                   ;STRING,INT,LONG INT
```

```
000233 VALTB2          DFB       0,192,0,64              ;(DITTO ABOVE)
000234                 DFB       255,128,64
000235 PUSHF           TAY                               ;GET POINTER INTO STACK.
000236                 PLA
000237                 STA       INDEX1
000238                 PLA
000239                 STA       INDEX1+1
000240                 INC       INDEX1
000241                 BNE       *+4
000242                 INC       INDEX1+1
000243                 TYA
000244 ;STORE FAC ON STACK UNPACKED.
000245                 PHA                               ;START WITH SIGN SET UP.
000246 FORPSH          JSR       ROUND                   ;PUT ROUNDED FAC ON STACK.
000247                 LDA       FACLO
000248                 PHA
000249                 LDA       FACMO
000250                 PHA
000251                 LDA       FACMOH
000252                 PHA
000253                 LDA       FACHO
000254                 PHA
000255                 LDA       FACEXP
000256                 PHA
000257                 JMP       (INDEX1)                ;RETURN.
000258                 PAGE
000259                 SBTL      "MULTIPLE DIMENSION CODE."
000260 FMAPTR:         LDA       COUNT
000261                 ASL       A
000262 FMPTR1          SEC
000263 FMPTR2          ADC       VARNAM+1                ;POINT TO ENTRIES, C IS SET.
000264                 ADC       LOWTR
000265                 LDY       LOWTR+1
000266                 LDX       LOWTRB
000267                 BCC       JSRGM
000268                 INY
000269                 CPY       #MAXPG
000270                 BCC       JSRGM
000271                 LDY       #MINPG
000272                 INX
000273 JSRGM:          STA       ARYPNT
000274                 STY       ARYPNT+1
000275                 STX       ARYPNTB
000276                 RTS
000277 N32768:         DFB       144,128,0,0             ;-32768.
000278                 DFB       0
000279 ; INTIDX READS A FORMULA FROM THE CURRENT POSITION AND
000280 ; TURNS IT INTO A POSITIVE INTEGER
000281 ; LEAVING THE RESULIN FACMO&LO. NEGATIVE ARGUMENTS
000282 ; ARE NOT ALLOWED.
000283 INTIDX:         JSR       CHRGET
000284                 LDA       TMPPTR                  ;SAVE THE ARRAY NAME POINTER FOR RECURSION.
000285                 PHA
000286                 LDA       TMPPTR+1
000287                 PHA
000288                 LDA       TMPPTRB
000289                 PHA
000290                 LDA       DORES
000291                 PHA
000292                 JSR       FRMNUM                  ;GET A NUMBER
000293                 PLA
000294                 STA       DORES
000295                 PLA
000296                 STA       TMPPTRB
000297                 PLA                               ;GET THE ARRAY NAME POINTER FOR PROSPERITY.
000298                 STA       TMPPTR+1
000299                 PLA
000300                 STA       TMPPTR
000301 POSINT:         LDA       FACSGN
000302                 BMI       NONONO                  ;IF NEGATIVE, BLOW HIM OUT.
000303 AYINT:          LDA       FACEXP
000304                 CMP       #144                    ;FAC .GT. 32767?
000305                 BCC       QINTGO
000306                 LDA       #>N32768
000307                 LDX       #0
000308                 LDY       #<N32768                ;GET ADDR OF -32768.
000309                 JSR       FCOMP                   ;SEE IF FAC=Y,A.
000310 NONONO:         BEQ       QINTGO                  ;FAC IS OK.
000311                 JMP       FCERR                   ;NO, FAC IS TOO BIG
000312 QINTGO:         EQU       *
```

```
000313                JSR       FADDH                    ;ADD .5 AND TRUNCATE.
000314                JMP       QINT                     ;GO TO QINT AND SHOVE IT.
000315 ; ISARY BUILDS OR SEARCHES FOR AN ARRAY
000316 ; AND IF NOT IN A DIM STMNT, INDEXES TO THE DESIRED ELEMENT.
000317 ISARY          LDA       FOUR6
000318                CLC
000319                ADC       #$2                      ; ADJUST THE NAME LENGTH
000320                STA       VARNAM+1                 ;TO BE THE TRUE LENGTH + 3 (FOR
                                                                    TYPE & LEN BYTES).
000321                LDA       SUBFLG
000322                BNE       STRTSRCH                 ;FOR STORE, RECALL LIKE FUNCTIONS.
000323                LDA       DIMFLG
000324                ORA       INTFLG
000325                PHA                                ;SAVE DIMFLG FOR RECURSION.
000326                TYA
000327                PHA                                ;SAVE VALTYP FOR RECURSION.
000328                LDY       #0                       ;SET NUMBER OF DIMENSIONS TO ZERO.
000329 INDLOP:         TYA                                ;SAVE NUMBER OF DIMS.
000330                PHA
000331                LDA       VARNAM+1
000332                PHA
000333                LDA       VARNAM
000334                PHA                                ;SAVE LOOKS.
000335                JSR       INTIDX                   ;EVALUATE INDICE INTO FACMO&LO.
000336                PLA
000337                STA       VARNAM
000338                PLA
000339                STA       VARNAM+1                 ;GET BACK ALL... WE'RE HOME.
000340                PLA                                ;(# OF DIMS).
000341                TAY
000342                TSX
000343                LDA       258,X
000344                PHA                                ;PUSH DIMFLG AND VALTYP FURTHER.
000345                LDA       257,X
000346                PHA
000347                LDA       INDICE                   ;PUT INDICE ONTO STACK.
000348                STA       258,X                    ;UNDER DIMFLG AND VALTYP.
000349                LDA       INDICE+1
000350                STA       257,X
000351                INY                                ;INCREMENT # OF DIMS.
000352                JSR       CHRGOT                   ;GET TERMINATING CHARACTER.
000353                CMP       #44                      ;A COMMA?
000354                BEQ       INDLOP                   ;YES.
000355                STY       COUNT                    AVE COUNT OF DIMS.
000356                JSR       CHKCLS                   ;MUST BE CLOSED PAREN.
000357                PLA
000358                STA       VALTYP                   ;GET VALTYP AND
000359                PLA
000360                STA       INTFLG
000361                AND       #127
000362                STA       DIMFLG                   ;DIMFLG OFF STACK.
000363 STRTSRCH       LDA       ARYTAB+1
000364                LDX       ARYTAB
000365                LDY       ARYTABB
000366 LOPFDA         STA       LOWTR+1                  ;INITIALIZE LOWTR TO POINT TO
000367                STX       LOWTR                    ;THE ARRAY TABLE.
000368                STY       LOWTRB
000369                CPY       VARTABB
000370                BNE       LOPFDV
000371                CMP       VARTAB+1
000372                BNE       LOPFDV
000373                CPX       VARTAB
000374                BEQ       NOTFFD                   ;A FINE THING!! NO ARRAY!!
000375 LOPFDV         LDY       #$1                      ;POINT TO THE NAME IN PROGRAM TEXT.
000376 NMLOOP         DEY
000377                LDA       (TMPPTR),Y               ;GET THE CURRENT NAME CHAR.
000378                INY
000379                INY
000380                CMP       #'A'+$20
000381                BCC       NOT2SML
000382                CMP       #'Z'+$21
000383                BCS       CHKTYP
000384                SBC       #$1F                     ;CARRY CLEAR.
000385 NOT2SML        CMP       (LOWTR),Y                ;IS IT = TO THE CURR CHAR?
000386                BEQ       NMLOOP                   ;IF YES, THEN LOOK AT SOME MORE.
000387 CHKTYP         LDA       VALTYP
000388                CMP       (LOWTR),Y                ;DO TYPE BYTES MATCH?
000389                BNE       NOGOT                    ;WE DON'T HAVE IT YET.
000390                INY
000391                CPY       VARNAM+1                 ;ARE THE NAMES = IN LENGTH?
```

```
000392                  BEQ       GOTARY                    ;IF YES, THEN WE FOUND IT!!
000393 NOGOT            LDY       #$0
000394                  LDA       (LOWTR),Y                 ;ADD THE LENGTH OF THIS ENTRY.
000395                  CLC                                 ;TO LOWTR, THAT IS. IT WILL
000396                  ADC       LOWTR                     ;POINT TO THE NEXT ENTRY.
000397                  TAX
000398                  INY
000399                  LDA       (LOWTR),Y
000400                  ADC       LOWTR+1
000401                  LDY       LOWTRB
000402                  JSR       FIXADC
000403                  JMP       LOPFDA                    ;ALWAYS GOES.
000404 BSERR            LDX       #ERRBS                    ;BAD SUBSCRIPT ERROR.
000405                  DFB       44                        ;A 2 BYTE SKIP.
000406 FCERR            LDX       #ERRFC                    ;TOO BIG. A FUNCTION CALL ERROR.
000407 ERRGO3           JMP       ERROR
000408 GOTARY           LDX       #ERRDD                    ;PERHAPS A RE-DEMENSIONED ERROR.
000409                  LDA       DIMFLG                    ;TEST THE DIM FLAG.
000410                  BNE       ERRGO3
000411                  LDA       SUBFLG
000412                  BEQ       GOGETM
000413                  SEC                                 ;EXIT IF IN STORE OR RECALL.
000414                  RTS                                 ;THIS IS CHEAP ASS FIX. (FAKES OUT DIM CODE).
000415 GOGETM           CLC
000416                  LDA       #$0                       ;GET THE COUNT OF DIMS.
000417                  JSR       FMPTR2                    ;POINT TO THE DIM COUNT BYTE.
000418                  LDA       COUNT
000419                  LDY       #0
000420                  CMP       (ARYPNT),Y                ;ARE THEY = ?
000421                  BNE       BSERR                     ;NO, BAD SUBSCRIPT ERROR.
000422                  JMP       GETDEF                    ;GO CALCULATE POINTER TO VARIABLE.
000423 ;HERE WHEN VARIABLE IS NOT FOUND IN THE ARRAY TABLE.
000424 ;BUILDING AN ENTRY.
000425 ;PUT DOWN THE DESCRIPTOR.
000426 ;SETUP # OF DIMS.
000427 ;MAKE SURE THERE IS ROOM FOR THE NEW ENTRY.
000428 ; REMEMBER 'VARPNT'.
000429 ; TALLY=4.
000430 ; SKIP 2 LOCS FOR LATER FILL IN OF SIZE.
000431 ; LOOP: GET AN INDICE
000432 ; PUT DOWN NUMBER+1 AND INCREMENT VARPTR.
000433 ; TALLY=TALLY*NUMBER+1.
000434 ; DECREMENT NUMBER-DIMS.
000435 ; BNE LOOP
000436 ; CALL 'REASON' WITH Y,A REFLECTING LAST LOC OF VARIABLE.
000437 ; UPDATE STREND.
000438 ; ZERO ALL.
000439 ; MAKE TALLY INCLUDE MAXDIMS AND DESCRIPTOR.
000440 ; PUT DOWN TALLY.
000441 ; IF CALLED BY DIMENSION, RETURN.
000442 ; OTHERWISE INDEX INTO THE VARIABLE AS IF IT
000443 ;   WERE FOUND ON THE INITIAL SEARCH.
000444 NOTFFD:          LDA       DORES                     ;SHOULD WE CREATE THE ARRAY?
000445                  BEQ       NOTFF2                    ;YES.
000446                  LDA       #>ZERO                    ;NO, POINT TO A ZERO LOCATION.
000447                  LDY       #<ZERO
000448                  STA       VARPNT
000449                  STY       VARPNT+1
000450 NOTFONE          PLA
000451                  PLA                                 ;PULL OFF MAX INDICE.
000452                  DEC       COUNT
000453                  BNE       NOTFONE
000454                  LDA       #0
000455                  JMP       DIMRTS2
000456 NOTFF2           LDA       COUNT
000457                  STA       INDEX+1                   ;PUT INDICE COUNT IN A TEMP.
000458                  LDY       #$0
000459                  STY       CURTOL+1                  ;ZERO OUT THE TOTAL STORAGE COUNTER.
000460                  LDY       VARNAM
000461                  STY       CURTOL                    ;CURTOL=# BYTES/ELEMENT.
000462                  TSX                                 ;INDEX FOR STACK DATA (THE NEW MAX INDICES).
000463 SIZLOP           LDA       #$0                       ;ASSUME THIS IS A DEFAULT DECLERATION.
000464                  LDY       #$A                       ;I.E. NEVER DIMMED, AND NEVER USED BEFORE.
000465                  BIT       DIMFLG
000466                  BVC       NOTDIM                    ;BRANCH IF NOT IN A DIM STATEMENT.
000467                  INX
000468                  LDA       $100,X                    ;FETCH MAX CURRENT INDICE LOW.
000469                  INX
000470                  TAY
000471                  LDA       $100,X                    ;MAX CURRENT INDICE HIGH BYTE.
```

```
000472 NOTDIM          STX       INDEX                    ;SAVE X-REG STACK INDEX.
000473                 TAX
000474                 INY                                ;ADD 1 TO THE INDICE
000475                 BNE       NOVFLW
000476                 INX                                ;OVERFLOW INTO THE HIGH BYTE.
000477 NOVFLW          JSR       UMULT                    ;MULTIPLY THE INDICE * CURTOL.
000478                 STX       CURTOL
000479                 STY       CURTOL+1                 ;SAVE NEW CURTOL VALUE.
000480                 LDX       INDEX                    ;RESTORE LAST STACK POINTER VALUE.
000481                 DEC       INDEX+1                  ;HAVE WE RETRIEVED ALL NEW DIMS?
000482                 BNE       SIZLOP                   ;NO, SO GET SOME MORE.
000483                 LDA       COUNT                    ;GET THE # OF DIMS * 2.
000484                 ASL       A
000485                 SEC                                ;ADD [COUNT * 2] + [NAMESIZE + 3] + 1.
000486                 ADC       VARNAM+1                 ;CARRY IS SET TO REFLECT THE DIM BYTE.
000487                 ADC       CURTOL                   ;FINAL LOW ORDER BYTE OF STORAGE
                                                                 REQUIREMENT.(YEAH!)
000488                 STA       CURTOL
000489                 BCC       GOODIE                   ;CARRY CLEAR MEANS ALL OK.
000490                 INY                                ;Carry was set.  Bump high order Byte.
000491                 BNE       ALGOOD                   ;IF OVERFLOW, THEN OUT OF MEMORY.
000492                 JMP       OMERR                    ;BLOW UP ! NO ROOM LEFT!!!!
000493 ALGOOD          CLC
000494                 STY       CURTOL+1                 ;CURTOL IS NOW THE TRUE STORAGE
                                                                 NEEDED FOR THE ENTRY.
000495 GOODIE          LDA       STREND
000496                 STA       HIGHTR                   ;HIGH END SOURCE TO MOVE.
000497                 ADC       CURTOL                   ;DETERMINE DESTINATION HIGH END POINTER.
000498                 STA       HIGHDS                   ;DESTINATION ADDRESS.
000499                 LDA       STREND+1
000500                 STA       HIGHTR+1
000501                 ADC       CURTOL+1
000502                 LDY       STRENDB
000503                 STY       HIGHTRB
000504                 JSR       FIXADC
000505                 STA       HIGHDS+1
000506                 TYA
000507                 STA       HIGHDSB
000508                 TAX
000509                 LDA       HIGHDS                   ;MOVE THE SIMPLE VARIABLE TABLE UP.
000510                 LDY       HIGHDS+1
000511                 JSR       BLTU                     ;STREND IS AUTOMATICALLY UPDATED FOR US.
000512
000513 ; ###########################################################################################
000514 ; #   END OF FILE:  B3DIMNH.TEXT
000515 ; #   LINES      :  506
000516 ; #   CHARACTERS :  24794
000517 ; ###########################################################################################

+-----------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 517   CHARACTERS: 25346
|
+-----------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3UDEFI.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:32 PM
|  Modified: Wednesday, December 31, 1997             4:37:09 PM
|
+------------------------------------------------------------------------------

000001 ; ##############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3UDEFI.TEXT
000004 ; ##############################################################################################
000005
000006                 LDX       HIGHDS+1
000007                 LDY       HIGHDSB
000008                 LDA       HIGHDS                    ;CALCULATE THE NEW SIMPLE
000009                 STA       SMVARS                    ;PUT THE NEW SMVARS POINTER
000010                 STX       SMVARS+1                  ;IN ITS PLACE.
000011                 STY       SMVARSB
000012                 LDA       COUNT
000013                 STA       INDEX+1                   ;SAVE THE DIM COUNT IN A TEMP.
000014                 LDA       #$0
000015                 JSR       FMPTR1                    ;POINT TO THE WHERE THE INDICES GO.
000016                 LDY       #$0
000017 INDLP1          LDX       #$B                       ;SET A,X=11 IN CASE OF NO DIM.
000018                 LDA       #$0                       ;THIS IS THE DEFAULT MAX INDICE SIZE.
000019                 BIT       DIMFLG
000020                 BVC       NTDIMD                    ;IF V CLEAR, THEN NOT IN A DIM STATEMENT.
000021                 PLA                                 ;GET LOW PART OF CURRENT MAX INDICE.
000022                 CLC
000023                 ADC       #$1                       ;ADD 1 TO IT.
000024                 TAX
000025                 PLA                                 ;GET HIGH PART.
000026                 ADC       #$0                       ;INDICE IS NOW INCREMENTED BY 1.
000027 NTDIMD          STA       (ARYPNT),Y                ;STORE HIGH PART OF INDICE.
000028                 INY
000029                 TXA
000030                 STA       (ARYPNT),Y                ;STORE LOW PART OF INDICE.
000031                 INY                                 ;POINT TO NEXT GUY.
000032                 DEC       INDEX+1
000033                 BNE       INDLP1                    ;GO BACK FOR MORE INDICES.
000034                 LDY       #$0
000035                 LDA       CURTOL
000036                 STA       (LOWTR),Y
000037                 LDA       CURTOL+1
000038                 INY
000039                 STA       (LOWTR),Y                 ;WE JUST STORED THE ENTRY LENTGTH.
000040                 LDX       VARNAM+1
000041                 DEX
000042                 DEX                                 ;MAKE INDEX CONTAIN THE TRUE
000043                 STX       INDEX                     ;LENGTH OF THE VARIABLE NAME
                                                                    [+1 FOR THE TYPE BYTE]
000044 NAMLOP          DEY
000045                 LDA       (TMPPTR),Y                ;GET THE NAME FROM PROGRAM TEXT.
000046                 INY
000047                 INY                                 ;POINT TO CURRENT BYTE IN STORAGE.
000048                 CMP       #'A'+$20
000049                 BCC       *+4
000050                 SBC       #$20
000051                 STA       (LOWTR),Y                 ;STORE THE NAME IN THE TABLE.
000052                 CPY       INDEX                     ;ARE WE DONE WITH THE NAME?
000053                 BNE       NAMLOP
000054                 LDA       VALTYP
000055                 INY
000056                 STA       (LOWTR),Y                 ;STORE THE TYPE BYTE.
000057                 INY
000058                 LDA       COUNT
000059                 STA       (LOWTR),Y                 ;STORE THE # OF DIMS HERE.
000060                 JSR       FMAPTR                    ;POINT TO THE VALUES.
000061                 LDX       VARTAB+1                  ;GET THE VARTAB POINTER HIGH BYTE.
000062                 LDY       ARYPNT
000063 ZERITA          LDA       #$0
000064                 STA       ARYPNT                    ;INDEX INTO THE ARRAY AND CLEAR IT OUT.
000065                 STA       (ARYPNT),Y                ;STORE A ZERO EVERYWHERE.
000066                 INY                                 ;POINT TO NEXT LOCATION.
000067                 BNE       CHKUPR
000068                 INC       ARYPNT+1                  ;CROSSED A PAGE BOUNDRY.
000069                 LDA       ARYPNT+1
000070                 CMP       #MAXPG
000071                 BCC       CHKUPR
```

```
000072                  SBC       #MAXPG-MINPG
000073                  STA       ARYPNT+1
000074                  INC       ARYPNTB
000075 CHKUPR           CPX       ARYPNT+1                ;DO UPPER POINTER BYTES MATCH?
000076                  BNE       ZERITA                  ;NO. SO CLEAR SOME MORE.
000077                  LDA       ARYPNTB
000078                  CMP       VARTABB
000079                  BNE       ZERITA
000080                  CPY       VARTAB                  ;DO LOW ORDERS MATCH?
000081                  BCC       ZERITA                  ;CLEAR SOME MORE.
000082                  LDA       DIMFLG
000083                  BNE       DIMRTS1                 ;ALL DONE IF IN A DIM STATEMENT.
000084 ; AT THIS POINT LOWTR,Y POINTS TO THE INDICES.
000085 ; DIMENSIONS. STRATEGY:
000086 ; NUMDIM=NUMBER OF DIMENSIONS.
000087 ; CURTOL.
000088 ; INLPNM:GET A NEW INDICE.
000089 ; MAKE SURE INDICE IS NOT TOO BIG.
000090 ; MULTIPLY CURTOL BY CURMAX.
000091 ; ADD INDICE TO CURTOL.
000092 ; NUMDIM=NUMDIM-1.
000093 ; BNE INLPNM.
000094 ; USE CURTOL*4 AS OFFSET.
000095 GETDEF           LDA       COUNT                   ;AND IT'S JUST THAT SIMPLE!
000096                  STA       INDEX+1
000097                  LDA       #0
000098                  JSR       FMPTR1                  ;POINT TO THE INDICES.
000099                  LDY       #0                      ;ZERO CURTOL.
000100                  STY       CURTOL
000101                  STY       CURTOL+1
000102 INLPNM           PLA                               ;GET LOW INDICE.
000103                  TAX
000104                  STA       INDICE
000105                  PLA                               ;AND THE HIGH PART
000106                  STA       INDICE+1
000107                  CMP       (ARYPNT),Y              ;COMPARE WITH MAX INDICE.
000108                  BCC       INLPN2
000109                  BNE       BSERR7                  ;IF GREATER, 'BADSUBSCRIPT' ERROR.
000110                  INY
000111                  TXA
000112                  CMP       (ARYPNT),Y
000113                  BCC       INLPN1
000114 BSERR7:          JMP       BSERR
000115 OMERR1:          JMP       OMERR
000116 INLPN2:          INY
000117 INLPN1:          LDA       CURTOL+1                ;DON'T MULTIPLY IF CURTOL=0.
000118                  ORA       CURTOL
000119                  CLC                               ;PREPARE TO GET INDICE BACK.
000120                  BEQ       ADDIND                  ;GET HIGH PART OF INDICE BACK.
000121                  STY       INDEX                   ;SAVE IT FOR LATER.
000122                  LDA       (ARYPNT),Y              ;GET THE MAX INDICE.
000123                  PHA                               ;(FOR THE MULTIPLICATION).
000124                  DEY
000125                  LDA       (ARYPNT),Y              ;HIGH BYTE.
000126                  TAX
000127                  PLA
000128                  TAY                               ;LOW BYTE.
000129                  JSR       UMULT                   ;MULTIPLY CURTOL BY LOWTR,Y,Y+1.
000130                  TXA
000131                  ADC       INDICE                  ;ADD IN INDICE.
000132                  TAX
000133                  TYA
000134                  LDY       INDEX
000135 ADDIND:          ADC       INDICE+1
000136                  INY
000137                  STX       CURTOL
000138                  STA       CURTOL+1
000139                  DEC       INDEX+1                 ;COUNT OF DIMS.
000140                  BNE       INLPNM                  ;YES
000141                  JSR       FMAPTR                  ;POINT TO THE VALUES.
000142                  LDX       #$0
000143                  LDY       VARNAM                  ;MAKE X,Y=#OF BYTES PER ELEMENT.
000144                  JSR       UMULT                   ;DO FINAL MULT. TO TAKE ELEMENT
000145                  TXA                               ;SIZE INTO ACCOUNT.
000146                  ADC       ARYPNT                  ;CARRY IS KNOWN TO BE CLEARED.
000147                  STA       VARPNT                  ;LOW ORDER OF ACTUAL VARIABLE POINTER.
000148                  TYA
000149                  ADC       ARYPNT+1
000150                  LDY       ARYPNTB
000151                  JSR       FIXADC
```

```
000152                  STA       VARPNT+1                  ;HIGH ORDER OF ACTUAL POINTER TO VARIABLE.
000153                  TYA
000154                  ADC       #0
000155 DIMRTS2          STA       VARPNTB
000156                  LDA       #$80
000157                  STA       ISARA
000158                  LDY       VARPNT+1
000159                  LDA       VARPNT                    ;Y,A = VARPNT FOR PROSPERITY.
000160 DIMRTS1          JMP       MYMYMY                    ;GO RETIRE...
000161                  PAGE
000162                  SBTL      "INTEGER ARITHMETIC ROUTINES."
000163 ;Y.X=Y.X * CURTOL .  CUTOL,DECCNT CLOBBERED.
000164 ;UNSIGNED INTEGER MULTPY.
000165 ;THIS IS FOR MULTIPLY DIMENSIONED ARRAYS.
000166 ; X,Y=X,A=CURTOL*LOWTR,Y,Y+1.
000167 UMULT           LDA       #$10                      ;LOOP COUNT = 16 BITS.
000168                  STA       DECCNT
000169                  STX       ADDEND+1                  ;ADDEND,+1 IS THE
000170                  STY       ADDEND                    ;MULTIPLICAND.
000171                  LDX       #$0
000172                  LDY       #$0                       ;Y,X WILL BE THE PRODUCT, SO CLEAR IT OUT.
000173 UMULTC           TXA                                 ;PRODUCT=PRODUCT*2.
000174                  ASL       A
000175                  TAX
000176                  TYA
000177                  ROL       A
000178                  TAY
000179                  BCS       OMERR1                    ;IF C SET, THEN OUT OF MEMORY.
000180                  ASL       CURTOL                    ;SHIFT CURTOL 1 BIT TO
000181                  ROL       CURTOL+1                  ;SEE IF TIME FOR PARTIAL PRODUCT.
000182                  BCC       UMLCNT                    ;IF C CLEAR, THEN NO PARTIAL.
000183                  CLC
000184                  TXA
000185                  ADC       ADDEND                    ;PRODUCT=PRODUCT+PARTIAL PRODUCT.
000186                  TAX
000187                  TYA
000188                  ADC       ADDEND+1
000189                  TAY
000190                  BCS       OMERR1                    ;IF C SET, THEN TOO BIG.
000191 UMLCNT           DEC       DECCNT                    ;SEE IF MORE MULTIPLYING, IF SO THEN
000192                  BNE       UMULTC                    ;BACK FOR MORE.
000193 UMLRTS           RTS                                 ;ALL FINISHED... GO GET STONED.
000194                  PAGE
000195                  SBTL      "FRE FUNCTION AND INTEGER TO FLOATING ROUTINES."
000196 NOFREF:          JSR       LONGST0
000197                  LDA       #0
000198                  STA       GARBFL
000199                  JSR       EXPAND
000200                  JSR       GARBA2
000201                  SEC
000202                  LDA       FRETOP                    ;WE WANT
000203                  SBC       STREND                    ;FRETOP-STREND.
000204                  STA       FAC+7
000205                  LDA       FRETOP+1
000206                  SBC       STREND+1
000207                  LDY       FRETOPB
000208                  LDX       STRENDB
000209                  JSR       FIXAYX
000210                  STA       FAC+6
000211                  STY       FAC+5
000212                  JMP       LMAKFLT
000213 GIVAYF:          LDX       #0
000214                  STX       VALTYP
000215                  STA       FACHO
000216                  STY       FACHO+1
000217                  LDX       #144                      ;SET EXPONENT TO 216.
000218                  JMP       FLOATS                    ;TURN IT TO A FLOATING PNT #.
000219 POS:             JSR       VPOS                      ;READ THE POSITION FROM SOS.
000220                  LDY       CURX
000221 VPOS2            INY                                 ;+1 TO AGREE WITH TAB & HTAB
000222 SNGFLT:          LDA       #0
000223                  SEC
000224                  BEQ       GIVAYF                    ;FLOAT IT.
000225 DOVPOS           JSR       VPOS                      ;READ IT FROM SOS.
000226                  LDY       CURY
000227                  JMP       VPOS2
000228                  PAGE
000229                  SBTL      "SIMPLE-USER-DEFINED-FUNCTION CODE."
000230 ; NOTE ONLY SINGLE ARGUMENTS ARE ALLOWED TO FUNCTIONS
000231 ; AND FUNCTIONS MUST BE OF THE SINGLE LINE FORM:
```

```
000232 ; DEF FNA(X)=X2+X-2
000233 ; NO STRINGS CAN BE INVOLVED WITH THESE FUNCTIONS.
000234 ; IDEA: CREATE A SIMPLE VARIABLE ENTRY
000235 ; WHOSE FIRST CHARACTER HAS THE 200 BIT SET.
000236 ; THE VALUE WILL BE:
000237 ; A TEXT PNTR TO THE FORMULA.
000238 ; A PNTR TO THE ARGUMENT VARIABLE.
000239 ; FUNCTIONAMES CAN BE LIKE 'FNA4'.
000240 ; SUBROUTINE TO SEE IF WE ARE IN DIRECT MODE.
000241 ; AND COMPLAIN IF SO.
000242 ERRDIR:         LDX       CURLIN+1                ;DIR MODE HAS CURLIN=0,255
000243                INX                               ;SO NOW, IS RESULT ZERO?
000244                BNE       UMLRTS                  ;YES.
000245                LDX       #ERRID                  ;INPUT DIRECT ERROR CODE.
000246                DFB       44                      ;SKIP 2 OFFSET.
000247 ERRGUF:        LDX       #ERRUF
000248                JMP       ERROR
000249 DEF:           JSR       GETFNM                  ;GET A PNTR TO THE FUNCTION.
000250                JSR       ERRDIR
000251                JSR       CHKOPN                  ;MUST HAVE '('.
000252                LDA       #128
000253                STA       SUBFLG                  ;PROHIBIT SUBSCRIPTED VARIABLES.
000254                JSR       MYPTRGET                ;GET PNTR TO ARGUMENT.
000255                JSR       PNTREL                  ;MAKE FORPNT RELATIVE.
000256                JSR       CHKNUM                  ;IS IT A NUMBER?
000257                JSR       CHKCLS                  ;MUST HAVE ')'
000258                LDA       #'='
000259                JSR       SYNCHR                  ;MUST HAVE '='.
000260                LDA       FORPNTB
000261                PHA
000262                LDA       FORPNT+1
000263                PHA
000264                LDA       FORPNT
000265                PHA
000266                LDA       TXTPTRB
000267                PHA
000268                LDA       TXTPTR+1
000269                PHA
000270                LDA       TXTPTR
000271                PHA
000272                JSR       DATA
000273                LDX       #5
000274                JMP       DEFFIN
000275 ; SUBROUTINE TO GET A PNTR TO A FUNCTION NAME.
000276 GETFNM:        LDA       #FNTK
000277                JSR       MSTESC                  ;THERE BETTER BE AN ESCAPE TOKEN!
000278 GETFN1         ORA       #128                    ;PUT FUNCTION BIT ON.
000279                STA       SUBFLG
000280                JSR       PTRGT2                  ;GET POINTER TO FUNCTION OR CREATE ANEW.
000281                STA       DEFPNT
000282                STY       DEFPNT+1
000283                LDA       VARPNTB
000284                STA       DEFPNTB
000285                LDX       #$0                     ;FUNC ARGS ARE REALS ONLY.
000286                STX       VALTYP
000287                JMP       CHKNUM                  ;MAKE SURE IT'S NOA STRING AND RETURN.
000288 FNDOER:        JSR       CHRGOT
000289                JSR       GETFN1                  ;GET THE FUNCTION'S NAME.
000290                LDA       DEFPNTB
000291                PHA
000292                LDA       DEFPNT+1
000293                PHA
000294                LDA       DEFPNT
000295                PHA
000296                JSR       PARCHK                  ;EVALUATE PARAMETER.
000297                JSR       CHKNUM
000298                PLA
000299                STA       DEFPNT
000300                PLA
000301                STA       DEFPNT+1
000302                PLA
000303                STA       DEFPNTB
000304                LDY       #3
000305                LDA       (DEFPNT),Y              ;GET POINTER TO VARIABLE.
000306                STA       INDEX                   ;SAVE VARIABLE POINTER.
000307                INY
000308                LDA       (DEFPNT),Y
000309                STA       INDEX+1
000310                INY                               ;SINCE DEF USES ONLY 4.
000311                LDA       (DEFPNT),Y
```

```
000312                  BNE       *+5
000313 NOFUN            JMP       ERRGUF
000314                  STA       INDEXB
000315                  LDX       #INDEX
000316                  JSR       PNTRL1                    ;MAKE INDEX ABSOLUTE AGAIN.
000317                  LDY       #3
000318 DEFSTF:          LDA       (INDEX),Y
000319                  PHA                                 ;PUSH IT ALL ON STACK.
000320                  DEY                                 ;SINCE WE ARE RECURSING MAYBE.
000321                  BPL       DEFSTF
000322                  LDA       INDEXB
000323                  JSR       FOURBYT                   ;PUT CURRENT FAC INTO OUR ARG VARIABLE.
000324                  LDA       TXTPTRB
000325                  PHA
000326                  LDA       TXTPTR+1
000327                  PHA
000328                  LDA       TXTPTR
000329                  PHA                                 ;SAVE TEXT POINTER.
000330                  LDA       (DEFPNT),Y                ;PNTR TO FUNCTION.
000331                  STA       TXTPTR
000332                  INY
000333                  LDA       (DEFPNT),Y
000334                  STA       TXTPTR+1
000335                  INY
000336                  LDA       (DEFPNT),Y
000337                  STA       TXTPTRB
000338                  LDA       INDEXB
000339                  PHA
000340                  LDA       INDEX+1
000341                  PHA
000342                  LDA       INDEX
000343                  PHA                                 ;SAVE VARIABLE POINTER.
000344                  JSR       DECTPT
000345                  JSR       CHRGOT
000346                  CMP       #'='                      ;IS THE FUNCTION DEFINTION STILL THERE?
000347                  BNE       NOFUN                     ;NO, GIVE UNDEFINED FUN ERROR.
000348                  JSR       CHRGET
000349                  JSR       FRMNUM                    ;EVALUATE FORMULA AND CHECK NUMERIC.
000350                  PLA
000351                  STA       DEFPNT
000352                  PLA
000353                  STA       DEFPNT+1
000354                  PLA
000355                  STA       DEFPNTB
000356                  JSR       CHRGOT
000357                  BEQ       *+5
000358                  JMP       SNERR                     ;IT DIDN'T TERMINE. HUH?
000359                  PLA
000360                  STA       TXTPTR
000361                  PLA
000362                  STA       TXTPTR+1                  ;RESTORE TEXT PNTR.
000363                  PLA
000364                  STA       TXTPTRB
000365                  LDX       #3
000366 DEFFIN:          LDY       #$FF
000367 DEFLOP           INY
000368                  PLA                                 ;GET OLD ARG VALUE OFF STACK
000369                  STA       (DEFPNT),Y                ;AND PUT IT BACK IN VARIABLE.
000370                  DEX
000371                  BPL       DEFLOP
000372                  RTS
000373
000374 ; ###############################################################################################
000375 ; #   END OF FILE:  B3UDEFI.TEXT
000376 ; #   LINES      :  367
000377 ; #   CHARACTERS :  16565
000378 ; ###############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 378   CHARACTERS: 17117
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "STRNGSTUF.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:38 PM
|  Modified: Wednesday, December 31, 1997             4:37:15 PM
|
+------------------------------------------------------------------------------

000001 ; ###########################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  STRNGSTUF.TEXT
000004 ; ###########################################################################################
000005
000006                  SBTL      "STRING FUNCTIONS."
000007 ; The STR$ function takes a number and gives a string with the
000008 ;   characters the output of the number would have given.
000009 STRS:            LDY       #0
000010                  JSR       FOUTC                     ;DO ITS OUTPUT.
000011                  LDA       #>LOFBUF
000012                  LDY       #<LOFBUF
000013                  LDX       #0
000014                  BEQ       STRLIT                    ;Scan it and turn it into a String.
000015 ;
000016 ; STRINI gets String space for creation of a string and creates a
000017 ;   descriptor for it in DSCTMP.
000018 STRINI:          LDX       FACMOB
000019                  STX       DSCPNTB
000020                  LDX       FACMO
000021                  LDY       FACMO+1                   ;Get FACMO to store in DSCPNT.
000022 STRIN2           STX       DSCPNT
000023                  STY       DSCPNT+1                  ;Retain the descriptor pointer.
000024 STRSPA:          JSR       GETSPA                    ;Get string space.
000025                  STX       DSCTMP+1
000026                  LDX       FRESPCB
000027                  STX       DSCTMPB
000028                  STY       DSCTMP+1+1                ;Save Location.
000029                  STA       DSCTMP                    ;Save Length.
000030                  TAX                                 ;Save Accumulator for STRCP.
000031                  BEQ       STRRT3
000032                  LDY       #2
000033                  LDA       #0
000034 STRMVLP          STA       (HIGHDS),Y
000035                  DEY
000036                  BNE       STRMVLP                   ;TEMPORARY DESC. CAN'T BE POINTED TO.
000037                  LDA       #TEMPTYP
000038                  STA       (HIGHDS),Y
000039                  TXA                                 ;Restore ACC so STRCP will work.
000040 STRRT3           RTS                                 ;All done.
000041 ;
000042 ; STRLT2 takes the String Literal whose first character is pointed
000043 ;   to by Y,A and builds a descriptor for it.  The descriptor is
000044 ;   initially built in DSCTMP, but PUTNEW transfers it into a Temporary
000045 ;   and leaves a pointer to the Temporary in FACMO & FACLO. The
000046 ;   character, other than zero, that terminates the String, should be
000047 ;   set up in CHARAC and ENDCHR.  If terminator is a Quote, the Quote is
000048 ;   saved.  Leading Quote should be skipped before JSR.
000049 ;
000050 ; On Return, the character after the string literal is pointed to by
000051 ;   STRNG2.
000052 STRLIT:          PHA
000053                  LDA       #34                       ;ASSUME STRING ENDS ON QUOTE.
000054                  STA       CHARAC
000055                  STA       ENDCHR
000056                  PLA
000057 STRLT2:          STA       STRNG1
000058                  STY       STRNG1+1                  ;SAVE POINTER TO STRING.
000059                  STX       STRNG1B
000060                  STX       DSCTMPB
000061                  STA       DSCTMP+1
000062                  STY       DSCTMP+1+1                ;IN CASE NO STRCPY.
000063                  LDY       #$FF                      ;INITIALIZE CHARACTER COUNT.
000064 STRGET:          INY
000065                  LDA       (STRNG1),Y                ;GET CHARACTER.
000066                  BEQ       STRFI1                    ;END ON TERMINATORS
000067                  CMP       CHARAC                    ;THIS TERMINATOR?
000068                  BEQ       STRFIN                    ;YES.
000069                  CMP       ENDCHR
000070                  BNE       STRGET                    ;LOOK FURTHER.
000071 STRFIN:          CMP       #34                       ;QUOTE?
000072                  BEQ       STRFI2
```

```
000073 STRFI1:          CLC                             ;NO, BACK UP.
000074 STRFI2:          STY       DSCTMP                ;RETAIN COUNT.
000075                  TYA
000076                  ADC       STRNG1                ;WISHING TO SET TXTPTR.
000077                  STA       STRNG2
000078                  LDX       STRNG1B
000079                  STX       STRNG2B
000080                  LDX       STRNG1+1
000081                  BCC       STRST2
000082                  INX
000083                  CPX       #MAXPG
000084                  BCC       STRST2
000085                  LDX       #MINPG
000086                  INC       STRNG2B
000087 STRST2:          STX       STRNG2+1
000088 ;   Every string gets copied into string space.
000089 STRCP:           TYA
000090                  JSR       STRINI                ;MUST SAVE A FOR CALL TO MOVSTR.
000091                  LDX       STRNG1
000092                  LDY       STRNG1+1
000093                  STX       INDEX1
000094                  STY       INDEX1+1
000095                  LDX       STRNG1B
000096                  STX       INDEX1B
000097                  JSR       MOVDO                 ;Move string
000098 ; Some String function is returning a result in DSCTMP.  Set up a TEMP
000099 ;   descriptor with DSCTMP in it.  Put a pointer to the descriptor in
000100 ;   FACMO & FACLO and flag the result as type String.
000101 PUTNEW:          LDX       TEMPPT                ;POINTER TO FIRST FREE TEMP.
000102                  CPX       #STRSIZ*NUMTMP+TEMPST
000103                  BNE       PUTNW1
000104                  LDX       #ERRST                ;STRING TEMPORARY ERROR.
000105                  JMP       ERROR                 ;GO TELL HIM.
000106 PUTNW1:          LDA       DSCTMP                ;Get actual string length
000107                  STA       0,X
000108                  SEC
000109                  LDA       HIMEM
000110                  SBC       DSCTMP+1
000111                  STA       1,X
000112                  LDA       HIMEM+1
000113                  SBC       DSCTMP+2
000114                  LDY       HIMEMB
000115                  STX       KIMY
000116                  LDX       DSCTMPB
000117                  JSR       FIXAYX
000118                  LDX       KIMY
000119                  STA       2,X
000120                  LDY       #0
000121                  STX       FACMO
000122                  STY       FACMO+1
000123                  STY       FACMOB
000124                  DEY
000125                  STY       VALTYP                ;TYPE IS 'STRING'.
000126                  STX       LASTPT                ;SET POINTER TO LAST-USED TEMP.
000127                  INX
000128                  INX
000129                  INX                             ;POINT FURTHER.
000130                  STX       TEMPPT                ;SAVE POINTER TO NEXT TEMP IF ANY.
000131                  RTS                             ;ALL DONE.
000132 PREFIXS          EQU       *                     ;THIS CODE RETURNS THE PREFIX AS A STRING.
000133                  BRK
000134                  DFB       GETPREF               ;GET PREFIX
000135                  DW        PREFTAB
000136                  BNE       JSERROR
000137                  LDA       CATBUF+1              ;THIS IS WHERE SOS RETURNS THE PREFIX.
000138 GOTITNOW         TAY                             ;LENGTH IN A
000139                  LDA       #>CATBUF+2
000140                  STA       STRNG1
000141                  LDA       #<CATBUF+2            ;STRNG1 POINTS TO CATBUF+2.
000142                  STA       STRNG1+1
000143                  LDA       #0
000144                  STA       STRNG1B
000145                  JMP       STRCP                 ;PART OF STRLIT.
000146 TIMES            EQU       *                     ;RETURNS THE TIME AS A STRING.
000147                  LDA       #':'                  ;DIGIT SEPARATOR.
000148                  LDY       #9                    ;INDEX INTO TIME FROM SOS.
000149                  BNE       DATE01                ;ALWAYS.
000150 DATES            EQU       *                     ;RETURNS THE DATE AS A STRING.
000151                  LDA       #'/'                  ;DIGIT SEPARATOR.
000152                  LDY       #2                    ;INDEX FOR DATE.
```

```
000153 DATE01          STA       YSAVE
000154                 BRK
000155                 DFB       GETCLOK                 ;GET THE TIME FROM SOS.
000156                 DW        DATETAB
000157                 LDA       #3
000158                 STA       KIMY                    ;3 FIELDS BETWEEN /'S OR :'S.
000159                 LDX       #0
000160 DATE02          LDA       CATBUF+2,Y              ;GET A BYTE OF TIME.
000161                 STA       CATBUF+2,X              ;SLAP IT INTO SPOT FOR STRING.
000162                 INX
000163                 INY
000164                 LDA       CATBUF+2,Y
000165                 STA       CATBUF+2,X              ;THIS MAY LOOK SILLY SO FAR, BUT IT WORKS.
000166                 INX
000167                 INY
000168                 LDA       YSAVE                   ;GET THE DELIMETER.
000169                 STA       CATBUF+2,X              ;HERE X GETS AHEAD OF Y.
000170                 INX
000171                 DEC       KIMY                    ;DONE YET?
000172                 BNE       DATE02
000173                 LDA       #8                      ;LENGTH OF ACTUAL STRING TO RETURN.
000174                 BNE       GOTITNOW                ;SAME AS PREFIX$.
000175 PREFIXSET       JSR       CHKEQL                  ;SET PREFIX CODE
000176                 JSR       GETNAME                 ;GET NEXT STRING AS IF IT IS A PATHNAME.
000177                 BRK
000178                 DFB       SETPREF                 ;SET PREFIX.
000179                 DW        PREFTB2
000180                 BNE       *+6
000181                 JSR       FILSOS                  ;New SOS prefix...refill SOSPATH
000182                 RTS                               ;ALL DONE.
000183 JSERROR         JMP       SERROR
000184 PROGPFXS        EQU       *                       ;Return PROG Prefix as a string
000185                 LDY       PROGPATH                ;Get length
000186                 LDA       #>PROGPATH+1            ;Set STRNG1 pointer to PROGPATH+1
000187                 STA       STRNG1
000188                 LDA       #<PROGPATH+1
000189                 STA       STRNG1+1
000190                 LDA       #0
000191                 STA       STRNG1B                 ;Set the Bank too
000192                 JMP       STRCP                   ;Do the string copy to String Space
000193 PROGPFX         EQU       *                       ;Set PROGPATH to a given string
000194                 JSR       CHKEQL                  ;SET PREFIX CODE
000195                 JSR       GETNAME                 ;GET NEXT STRING AS IF IT IS A PATHNAME
000196                 LDY       NAMBUF                  ;Get length
000197                 LDA       NAMBUF,Y                ;Check if last char is a '/'
000198                 CMP       #'/'
000199                 BEQ       PROGPFX1
000200                 INY                               ;Length = length + 1
000201                 LDA       #'/'
000202                 STA       NAMBUF,Y                ;Add a '/' at the end if needed
000203                 STY       NAMBUF
000204 PROGPFX1        EQU       *
000205                 JSR       FILPROG                 ;It's ok so put it in PROGPATH
000206                 JSR       CNVTPFX1
000207                 JSR       SETSOS                  ;Reset prefix to SOSPATH
000208                 RTS
000209 *
000210 * INSTR.
000211 *
000212 * FIND STRING WITHIN A STRING.
000213 *
000214 INSTR           PLA                               ;STARTING POSITION.
000215                 STA       TEMP
000216                 JSR       CHKCLS                  ;CLOSE PAREN
000217                 PLA
000218                 STA       INDEX                   ;STRING TO FIND.
000219                 STA       FACMO
000220                 PLA
000221                 STA       INDEX+1
000222                 STA       FACMO+1
000223                 PLA
000224                 STA       INDEXB
000225                 STA       FACMOB
000226                 PLA
000227                 STA       ARGMO                   ;SOURCE STRING.
000228                 PLA
000229                 STA       ARGMO+1
000230                 PLA
000231                 STA       ARGMOB
000232                 JSR       NOTNOW                  ;INDEX=(INDEX)
```

```
000233                STA       YSAVE                   ;LENGTH OF STRING TO FIND.
000234                LDY       #0
000235                LDA       (ARGMO),Y               ;LENGTH OF SOURCE STRING.
000236                CMP       YSAVE
000237                BCC       GIVMZERO                ;MUST BE AT LEAST AS BIG.
000238                STA       KIMY                    ;LENGTH OF SOURCE STRING.
000239                INY
000240                LDA       (ARGMO),Y
000241                STA       INDEX2
000242                TAX
000243                INY
000244                LDA       (ARGMO),Y
000245                STA       INDEX2+1
000246                LDA       KIMY
000247                SEC
000248                SBC       YSAVE
000249                STA       KIMY                    ;LAST CHAR TO BE MATCHED.
000250                DEC       TEMP
000251                JSR       RELINX                  ;MAKE INDEX2 ABSOLUTE.
000252                LDA       INDEX2
000253                CLC
000254                ADC       TEMP
000255                STA       INDEX2
000256                BCC       TRYAGM
000257                LDX       INDEX2+1                ;INC INDEX2+1
000258                INX
000259                CPX       #MAXPG
000260                BCC       *+7
000261                LDX       #MINPG
000262                INC       INDEX2B
000263                STX       INDEX2+1
000264 TRYAGM         LDA       KIMY                    ;LAST TO CMP.
000265                CMP       TEMP
000266                BCC       NMTCH2
000267                LDY       #0
000268 INMAT2         LDA       (INDEX2),Y
000269                CMP       (INDEX),Y
000270                BNE       INNOTIT                 ;NO MATCH.
000271                INY                               ;THAT'S ONE CHAR MATCHED.
000272                CPY       YSAVE                   ;LAST CHAR?
000273                BCC       INMAT2                  ;NO, TEST NEXT CARS.
000274                LDY       TEMP                    ;LOAD CHARACTER COUNT.
000275                BCS       GIVMIT                  ;ALWAYS.
000276 INNOTIT        INC       TEMP                    ;POSITION TO BEGIN WITH.
000277                INC       INDEX2                  ;WHERE TO GET THE CARACHTER.
000278                BNE       TRYAGM
000279                INC       INDEX2+1
000280                BNE       TRYAGM                  ;ALWAYS
000281 NMTCH2         EQU       *
000282 GIVMZERO       LDY       #$FF                    ;Y HAS VALUE TO BE RETURNED.
000283 GIVMIT         INY
000284                TYA
000285                PHA
000286                JSR       FRECNOW                 ;FREE THE SOURCE STRING?
000287                LDX       ARGMOB
000288                LDA       ARGMO
000289                LDY       ARGMO+1
000290                JSR       FRETNOW
000291                PLA
000292                TAY
000293                JMP       SNGFLT                  ;GO FLOAT Y.
000294 *
000295 * SUB$
000296 *
000297 * PARTIAL STRING SUBSTITUTION.
000298 *
000299 SUBLEFT        JSR       PTRGET                  ;GOT TO HAVE A VARIABLE.
000300                JSR       CHKSTR
000301                LDA       VARPNT                  ;POINTER TO VARIABLE.
000302                PHA
000303                LDA       VARPNT+1
000304                PHA
000305                LDA       VARPNTB
000306                PHA
000307                LDA       ISARA
000308                PHA
000309                JSR       CHKCOM                  ;MUST HAVE COMMA.
000310                JSR       GETBYT
000311                TXA
000312                PHA
```

```
000313                JSR       CHRGOT                    ;COMMA AFTER SECOND ARG?
000314                CMP       #','
000315                BNE       FAKEIT                    ;NO, ASSUME LENGTH OF REPLACEMENT STRING.
000316                JSR       CHRGET
000317                JSR       GETBYT                    ;GET LENGTH LIMIT.
000318                DFB       44
000319 FAKEIT         LDX       #$FF
000320                TXA
000321                PHA                                 ;LENGTH LIMIT.
000322                JSR       CHKCLS                    ;CLOSE PAREN.
000323                JSR       CHKEQL                    ;EQUALS SIGN.
000324                LDA       #$FF
000325                STA       VALTYP                    ;FORMULA MUST BE STRING.
000326                JSR       FRMEVL
000327                PLA                                 ;LENGTH LIMIT
000328                STA       DELTA
000329                PLA
000330                STA       YSAVE                     ;STARTING POSITION.
000331                BEQ       SUBFUC                    ;ILL. QUAN. ERROR.
000332                DEC       YSAVE
000333                LDY       #0
000334                LDA       (FACMO),Y                 ;LENGTH OF STRING.
000335                CMP       DELTA
000336                BCS       *+4
000337                STA       DELTA
000338                LDY       #0                        ;TEMP NOW HAS LENGTH TO REPLACE.
000339                PLA
000340                STA       ISARA
000341                PLA
000342                STA       FORPNTB
000343                PLA
000344                STA       FORPNT+1                  ;STRING TO CLOBBER.
000345                PLA
000346                STA       FORPNT
000347                LDA       (FORPNT),Y
000348                STA       DELTA+1                   ;LENGTH OF STRING TO CLOBBER.
000349                INY
000350                LDA       (FORPNT),Y
000351                STA       INDEX2
000352                INY
000353                LDA       (FORPNT),Y
000354                STA       INDEX2+1
000355                LDA       YSAVE                     ;POSITION TO START REPLACEMENT.
000356                CLC
000357                ADC       DELTA
000358                BCC       *+5                       ;IS THE RESULTING STRING GOING TO BE
000359 SUBFUC         JMP       FCERR                     ;LONGER THAN 255 CHARS?
000360                CMP       DELTA+1                   ;LONGER THAN THE ORIGINAL?
000361                BCC       GOTITTY                   ;NO, DON'T CREATE A NEW STRING.
000362                BEQ       GOTITTY                   ;STILL O.K.
000363 *THE RESULT IS GOING TO BE BIGGER THAN THE ORIGINAL.
000364 *MOVE THE ORIGINAL TO A NEW SPOT AND SWITCH ALL POINTERS.
000365                LDX       FORPNTB
000366                STX       DSCPNTB
000367                LDX       FORPNT                    ;FOR STRIN2.
000368                LDY       FORPNT+1
000369                JSR       STRIN2                    ;GET A NEW SPOT.
000370                TAY
000371                DEY
000372                LDA       #'                        ' ;INITIALIZE THE NEW STRING TO BLANKS
000373 SUBLNK1        STA       (FRESPC),Y                ;IN CASE OF HOLES.
000374                DEY
000375                BNE       SUBLNK1
000376                STA       (FRESPC),Y
000377                LDA       DELTA                     ;GOT TO SAVE THESE THINGS FROM THE LET CODE.
000378                PHA
000379                LDA       YSAVE
000380                PHA
000381                LDA       FACMO
000382                PHA
000383                LDA       FACMO+1
000384                PHA
000385                LDA       FACMOB
000386                PHA
000387                JSR       COPY.M                    ;OUT WITH THE OLD AND IN WITH THE NEW.
000388                PLA
000389                STA       FACMOB
000390                PLA
000391                STA       FACMO+1
000392                PLA
```

```
000393                  STA       FACMO
000394                  PLA
000395                  STA       YSAVE
000396                  PLA
000397                  STA       DELTA
000398                  LDA       DSCTMP+2
000399                  STA       INDEX2+1
000400                  LDA       DSCTMP+1
000401                  STA       INDEX2
000402                  LDA       DSCTMPB
000403                  STA       INDEX2B
000404                  JMP       GOTABS                    ;SKIP THE RELINX
000405 GOTITTY          JSR       RELINX                    ;MAKE INDEX2 ABSOLUTE.
000406 GOTABS           LDA       YSAVE
000407                  CLC
000408                  ADC       INDEX2
000409                  STA       INDEX2
000410                  BCC       GOTABS1
000411                  LDX       INDEX2+1                  ;INC INDEX2+1
000412                  INX
000413                  CPX       #MAXPG
000414                  BCC       *+7
000415                  LDX       #MINPG
000416                  INC       INDEX2B
000417                  STX       INDEX2+1
000418 GOTABS1          EQU       *                         ;MAKE INDEX ABSOLUTE.
000419                  JSR       NOTFAC
000420                  LDY       #0
000421                  BEQ       SBMV2
000422 SUBMOV           LDA       (INDEX),Y
000423                  STA       (INDEX2),Y
000424                  INY
000425 SBMV2            CPY       DELTA
000426                  BCC       SUBMOV
000427 FRECNOW:         JSR       FREFAC
000428                  JMP       FRENOW
000429 RELINX           SEC
000430                  LDA       HIMEM                     ;MAKE INDEX2 RELATIVE.
000431                  SBC       INDEX2
000432                  STA       INDEX2
000433                  LDA       HIMEM+1
000434                  LDY       HIMEMB
000435                  LDX       INDEX2+1
000436                  JSR       FIXYAX
000437                  STA       INDEX2+1
000438                  TYA
000439                  SBC       #0
000440                  STA       INDEX2B
000441                  RTS
000442 ; Procedure: CHR$(#)
000443 ; Function: Creates a string that contains as its only character the
000444 ;           ASCII equivalent if the integer argument (#).
000445 ; Restriction: The argument must be .GE. 0 and .LT. 256
000446 CHRS:           JSR       CONINT                    ;GET INTEGER IN RANGE.
000447                  TXA
000448                  PHA
000449                  LDA       #1                        ;ONE-CHARACTER STRING.
000450                  JSR       STRSPA                    ;GET SPACE FOR STRING.
000451                  PLA
000452                  LDY       #0
000453                  STA       (DSCTMP+1),Y
000454                  JMP       PUTNEW                    ;SETUP FAC TO POINT TO DESC.
000455 ; Procedure: LEFT$($,#)
000456 ; Function: Takes the leftmost # characters of the string.
000457 ;           If # .GT. length of string, it returns the whole string
000458 LEFTS:          JSR       PREAM                     ;TEST PARAMETERS.
000459                  CMP       (DSCPNT),Y
000460                  TYA
000461 RLEFT:          BCC       RLEFT1
000462                  LDA       (DSCPNT),Y
000463                  TAX                                 ;PUT LENGTH INTO X.
000464                  TYA                                 ;ZERO A THE OFFSET.
000465 RLEFT1:         PHA                                 ;SAVE OFFSET.
000466 RLEFT2:         TXA
000467 RLEFT3:         PHA                                 ;SAVE LENGTH.
000468                  JSR       STRSPA                    ;GET SPACE.
000469                  LDA       DSCPNT
000470                  LDX       DSCPNTB
000471                  LDY       DSCPNT+1
000472                  JSR       NOTNW2
```

```
000473                  PLA
000474                  TAY
000475                  PLA
000476                  CLC
000477                  ADC       INDEX                   ;COMPUTE WHERE TO COPY.
000478                  STA       INDEX
000479                  BCC       PULMOR
000480                  INC       INDEX+1
000481                  LDA       INDEX+1
000482                  CMP       #MAXPG
000483                  BCC       PULMOR
000484                  SBC       #MAXPG-MINPG
000485                  STA       INDEX+1
000486                  INC       INDEXB
000487 PULMOR:           TYA
000488                  JSR       MOVDO                   ;GO MOVE IT.
000489                  LDY       DSCPNT+1                ;HIGH BYTE 0?
000490                  LDX       DSCPNTB                 ;IF NOT, THEN THIS IS NOT A STING TEMP.
000491                  LDA       DSCPNT                  ;GET THE POINTER TO THE CURRENT DESCRIPTOR.
000492                  JSR       FRETNOW                 ;GO FREE THE DESCRIPTOR AND STRING.
000493                  JMP       PUTNEW
000494 RIGHTS:           JSR       PREAM
000495                  CLC                               ;LENGTH DES'D-LENGTH-1.
000496                  SBC       (DSCPNT),Y
000497                  EOR       #255                    ;NEGATE.
000498                  JMP       RLEFT
000499 ; MID ($,#) RETURNS STRING WITH CHARS FROM # POSITION
000500 ; ONWARD. IF # .GT. LEN ($) THEN RETURN NULL STRING.
000501 ; MID ($,#,#) RETURNS STRING WITH CHARACTERS FROM
000502 ; # POSITION FOR #2 CHARS. IF #2 GOES PAST END OF STRING
000503 ; RETURN AS MUCH AS POSSIBLE.
000504 MIDS:            LDA       #255                    ;DEFAULT.
000505                  STA       FACLO                   ;SAVE FOR LAT COMPARE.
000506                  JSR       CHRGOT                  ;GET CURRENT CHARACTER.
000507                  CMP       #41                     ;IS IT A RIGHT PAREN )?
000508                  BEQ       MID2                    ;NO THIRD PARAM.
000509                  JSR       CHKCOM                  ;MUST HAVE COMMA.
000510                  JSR       GETBYT                  ;GET THE LENGTH INTO 'FACLO'.
000511 MID2:            JSR       PREAM                   ;CHECK IT OUT.
000512                  DEX                               ;COMPUTE OFFSET.
000513                  TXA
000514                  PHA                               ;PRSERVE AWHILE.
000515                  CLC
000516                  LDX       #0
000517                  SBC       (DSCPNT),Y              ;GET LENGTH OF WHAT'S LEFT.
000518                  BCS       RLEFT2                  ;GIVE NULL STRING.
000519                  EOR       #255                    ;IN SUB C WAS 0 SO JUST COMPLEMENT.
000520                  CMP       FACLO                   ;GREATER THAN WHAT'S DESIRED?
000521                  BCC       RLEFT3                  ;NO,OPY THAT MUCH.
000522                  LDA       FACLO                   ;GET LENGTH OF WHAT'S DESIRED.
000523                  BCS       RLEFT3                  ;COPY IT.
000524 ; USED BY RIGHT$, LEFT$, MID$ FOR PARAMETER CHECKING & SET
000525 PREAM:            JSR       CHKCLS                  ;PARAM LIST SHOULD END.
000526                  LDA       #$FF
000527                  STA       VALTYP
000528                  PLA                               ;GET THE RETURN ADDRESS INTO
000529                  TAY                               ;JMPER+1,Y
000530                  PLA
000531                  STA       JMPER+1
000532                  PLA                               ;GET LENGTH.
000533                  TAX
000534                  PLA
000535                  STA       DSCPNT
000536                  PLA
000537                  STA       DSCPNT+1
000538                  PLA
000539                  STA       DSCPNTB
000540                  LDA       JMPER+1                 ;PUT RETURN ADDRESS BACK ON
000541                  PHA
000542                  TYA
000543                  PHA
000544                  LDY       #0
000545                  TXA
000546                  BEQ       GOFUC
000547                  RTS
000548 ; The function LEN($) returns the length of the string passed
000549 ;     as an argument.
000550 LEN:              JSR       LEN1
000551                  JSR       SNGFLT
000552 LEN0             LDA       DSCPNT
```

```
000553                LDX       DSCPNTB
000554                LDY       DSCPNT+1
000555 FRETNOW        JSR       FRETMP
000556                JMP       FRENOW                ;Actually free up the string space
                                                          and descriptor..
000557 LEN1:          LDA       FACMO                 ;Pointer to descriptor.
000558                STA       INDEX                 ; NOTNOW needs it in INDEX(+1)(B)
000559                STA       DSCPNT                ;So we can free the temp later
000560                LDA       FACMO+1
000561                STA       INDEX+1
000562                STA       DSCPNT+1
000563                LDA       FACMOB
000564                STA       INDEXB
000565                STA       DSCPNTB
000566                JSR       NOTNOW                ;On rtn, A=length, INDEX points to string
000567                LDX       #0
000568                STX       VALTYP                ;FORCE NUMERIC.
000569                TAY                             ;SET CODES ON LENGTH.
000570                RTS                             ;DONE.
000571 ; The following is the ASC($) function.  It returns an Integer which
000572 ;   is the decimal equivalent.
000573 ASC:           JSR       LEN1
000574                BNE       *+5
000575                JMP       GIVM1                 ;NULL string, return a -1
000576                LDY       #0
000577                LDA       (INDEX1),Y            ;GET CHARACTER.
000578                TAY
000579                JSR       SNGFLT
000580                JMP       LEN0                  ;FREE UP THAT MOTHER NOW.
000581 GOFUC:         JMP       FCERR                 ;YES.
000582 GTBYTC:        JSR       CHRGET
000583 GETBYT:        JSR       FRMNUM                ;READ FORMULA INTO FAC.
000584 CONINT:        JSR       POSINT                ;CONVERT THE FAC TO A SINGLE BYTE INT
000585                LDX       FACMO
000586                BNE       GOFUC                 ;RESULT MUST BE .LE. 255.
000587                LDX       FACLO
000588                JMP       CHRGOT                ;SET CONDITION CODES ON TERMINATOR.
000589 * The VAL function takes a string and turns it into a number by
000590 *   interpreting the ASCII digits, etc.  Except for the problem that a
000591 *   terminator must be supplied by replacing the character beyond the
000592 *   string, VAL is merely a call to FLOATING POINT INPUT (FIN).
000593 VAL:           LDA       #>FIN
000594                LDY       #<FIN
000595 VALSTR         STA       JMPER+1
000596                STY       JMPER+2
000597                JSR       LEN                   ;DO SETUP. SET RESULT=NUMERIC.
000598                BEQ       VALRTS
000599                LDX       TXTPTR
000600                LDY       TXTPTR+1
000601                STX       STRNG2
000602                STY       STRNG2+1              ;SAVE FOR LATER.
000603                LDX       TXTPTRB
000604                STX       STRNG2B
000605                LDX       INDEX1
000606                STX       TXTPTR
000607                LDX       INDEX1B
000608                STX       TXTPTRB
000609                LDY       INDEX1+1
000610                STY       TXTPTR+1
000611                JSR       CHRGOT                ;GET CHARACTER PNT'D TO AND SET FLAGS.
000612                JSR       JMPER
000613 ST2TXT:        LDX       STRNG2
000614                LDY       STRNG2+1
000615                STX       TXTPTR
000616                STY       TXTPTR+1
000617                LDX       STRNG2B
000618                STX       TXTPTRB
000619 VALRTS         RTS                             ;ALL DONE WITH STRINGS.
000620 GIVM1:         LDA       #>CON1M               ;POINT AT '-1' CONSTANT
000621                LDY       #<CON1M
000622                LDX       #CON1MB
000623                JMP       MOVFM
000624 CON1M:         DFB       $81
000625                DFB       $80
000626                DFB       00
000627                DFB       00
000628                DFB       00
000629 GETABYT        JSR       FRMNUM                ;THIS ROUTINE SETS X= SIGNED INT
000630                JSR       AYINT                 ;IN THE RANGE -128 TO 127.
000631                LDA       FACLO
```

```
000632                ROL       A                       ;C=HIGH BIT OF LOW BYTE.
000633                LDA       #0
000634                ADC       FACMO                   ;HIGH BYTE SHOULD BE FF OR 0, SO
000635                BEQ       *+5
000636                JMP       GOFUC                   ;WE SHOULD HAVE 0 NOW.
000637                LDX       FACLO
000638                JMP       CHRGOT                  ;JUST LIKE POSINT.
000639 GETADR:         LDA       FACEXP                  ;EXAMINE EXPONENT.
000640                CMP       #145
000641                BCC       *+5
000642                JMP       FCERR                   ;FUNCTION CALL ERROR.
000643                JSR       QINT                    ;INTEGERIZE IT.
000644                LDA       FACMO
000645                LDY       FACMO+1
000646                STY       POKER
000647                STA       POKER+1
000648                RTS                               ;IT'S DONE !.
000649
000650 ; ###############################################################################################
000651 ; #   END OF FILE:  STRNGSTUF.TEXT
000652 ; #   LINES      :  643
000653 ; #   CHARACTERS :  28681
000654 ; ###############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 654   CHARACTERS: 29237
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "INVOKE.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:36 PM
|  Modified: Wednesday, December 31, 1997             4:37:13 PM
|
+------------------------------------------------------------------------------

000001 ; ###########################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  INVOKE.TEXT
000004 ; ###########################################################################################
000005
000006                  SBTL      "INVOKE"
000007 *
000008 * INVOKE, PERFORM, EXFN
000009 *
000010 *  INVOKE is responsible for LOADing, Relocating, and Linking the
000011 *  Pascal Assembler created Object files.  Upon calling DOINVO,
000012 *  it is assumed that there exists free memory (all in one bank),
000013 *  between the pointers INVTAB and PROCTAB.
000014 *
000015 OFFSST           EQU       FORPNT
000016 PROCPNT          EQU       INPPTR
000017 SWPPNT           EQU       LOWDS
000018 TEMPTR           EQU       HEADER
000019 POINT1           EQU       INDEX1
000020 POINT1B          EQU       POINT1+SYSPAG
000021 POINT2           EQU       INDEX2
000022 POINT2B          EQU       POINT2+SYSPAG
000023 PROCPNTB         EQU       PROCPNT+SYSPAG
000024 OFFSSTB          EQU       OFFSST+SYSPAG
000025 SWPPNTB          EQU       SWPPNT+SYSPAG
000026 * Equates are done this way so that conflicts can be resolved
000027 * by just changing an Equate.
000028 INERROM          JMP       OMERR
000029 DOINVO           EQU       *                       ;FREE MEMORY NOW FROM
000030                  LDA       #0                      ;INVTAB TO PROCTAB.
000031                  STA       HEADERB
000032                  STA       SWPPNTB
000033                  LDA       INVTAB                  ;INVPNT INITIALLY SET TO INVTAB.
000034                  STA       INVPNT
000035                  LDA       INVTAB+1
000036                  STA       INVPNT+1
000037                  LDA       INVTABB
000038                  STA       INVPNTB
000039                  LDA       PROCTAB
000040                  LDX       PROCTAB+1
000041                  STA       PROCPNT
000042                  STX       PROCPNT+1
000043                  LDA       PROCTABB
000044                  STA       PROCPNTB
000045                  STA       OFFSSTB
000046 DOFILE           JSR       CHRGOT                  ;AN OTHER FILE?
000047                  BNE       *+5                     ;DONE YET?
000048                  JMP       DOCRNCH                 ;YES, CRUCH MEMORY BACK TOGETHER AND END.
000049                  LDA       #1
000050                  JSR       OPNPRTB                 ;OPEN THE FILE.
000051                  JSR       CHRGOT                  ;SET FLAGS.
000052                  BEQ       *+5                     ;LAST FILE?
000053                  JSR       CHKCOM                  ;NO, MUST HAVE A COMMA.
000054                  LDA       #>INVPNT                ;READ IN JUST ONE BLOCK.
000055                  STA       SBFPTR                  ;STARTING LOCATION FOR READ.
000056                  LDA       #<INVPNT
000057                  STA       SBFPTR+1
000058                  LDA       #0
000059                  STA       INBYTES
000060                  LDA       #2                      ;READ ONE BLOCK.
000061                  STA       INBYTES+1
000062                  SEC
000063                  LDA       PROCPNT+1
000064                  SBC       INVPNT+1
000065                  CMP       #3                      ;IS THERE GOING TO BE ROOM FOR 1 BLOCK?
000066                  BCC       INERROM                 ;NO, JUMP.
000067                  LDY       #RED                    ;SOS READ.
000068                  JSR       SETGO
000069                  LDY       #7
000070                  LDA       (INVPNT),Y              ;LENGTH OF SOURCE.
000071                  STA       INBYTES+1
000072                  DEY
```

```
000073                LDA       (INVPNT),Y
000074                STA       INBYTES
000075                SEC
000076                LDA       PROCPNT
000077                SBC       INBYTES                 ;READ PROGRAM INTO SPACE
000078                STA       POINT1
000079                LDA       PROCPNT+1
000080                SBC       INBYTES+1
000081                STA       POINT1+1
000082                LDA       INVPNTB
000083                STA       POINT1B
000084                LDA       #>POINT1
000085                STA       SBFPTR
000086                LDA       #<POINT1
000087                STA       SBFPTR+1
000088                LDY       #RED                    ;READ IN PROGRAM BUT NOT LINKER INFO.
000089                JSR       SETGO
000090                LDY       #GTM                    ;GET MARK.
000091                JSR       SETGO
000092                LDA       OUTMRK                  ;LOW BYTE ZERO?
000093                BEQ       MARKLOW
000094                INC       OUTMRK+1                ;NO CARRY OVER.
000095 MARKLOW         LDA       OUTMRK+1                ;THIS CODE MAKES MARK POINT
000096                LSR       A                       ;TO NEXT BLOCK BY MAKING IT DIVISIBLE
000097                ADC       #0                      ;BY 512.
000098                ASL       A
000099                STA       OUTMRK+1+1              ;BECAUSE IT GOES OUT ONE AWAY
                                                                FROM WHERE IT CAME IN.
000100                LDA       #0
000101                STA       OUTMRK+1                ;MARK NOW POINTS TO BLOCK BOUNDRY.
000102                STA       OUTMRK+1+3
000103                STA       BASE
000104                LDY       #STM                    ;SET MARK
000105                JSR       SETGO
000106                SEC
000107                LDA       POINT1                  ;NOW TIME TO LOAD IN THE LINKER INFO.
000108                SBC       INVPNT
000109                STA       INBYTES
000110                LDA       #>INVPNT
000111                STA       SBFPTR
000112                LDA       POINT1+1
000113                SBC       INVPNT+1
000114                STA       INBYTES+1
000115                LDA       #<INVPNT
000116                STA       SBFPTR+1
000117                LDY       #RED                    ;READ TILL EOF INTO INVPNT
000118                JSR       SETGO
000119                JSR       CLSEND                  ;CLOSE THE FILE.
000120                JSR       DEC2PROC                ;DEC PROCPNT BY 2.
000121                LDY       #1
000122                LDA       (PROCPNT),Y
000123                STA       KIMY                    ;# OF PROCEDURES IN THIS FILE.
000124                LDA       PROCPNT                 ;SAVE CURRENT PROCPNT.
000125                PHA
000126                LDA       PROCPNT+1
000127                PHA
000128 DOAPROC         JSR       RELPROC                 ;OFFSST=PROCPNT-(PROCPNT)
000129                LDA       PROCPNT
000130                STA       POINT2                  ;SAVE OLD PROCPNT.
000131                LDA       OFFSST
000132                STA       PROCPNT
000133                LDA       PROCPNT+1
000134                STA       POINT2+1
000135                LDA       OFFSST+1
000136                STA       PROCPNT+1
000137                LDA       PROCPNTB
000138                STA       POINT2B
000139                JSR       RELPROC                 ;OFFSST=PROCPNT-(PROCPNT)
000140                LDY       #0
000141                LDA       OFFSST                  ;POINTER TO ENTRY FOR THIS PROCEDURE.
000142                STA       HIGHTR                  ;POINTER TO BEGINNING OF PROCEDURE.
000143                STA       (POINT2),Y              ;BACK IN PROCEDURE POINTERS TABLE.
000144                INY
000145                LDA       OFFSST+1
000146                CMP       #$82
000147                BCC       *+4
000148                SBC       #$80
000149                STA       HIGHTR+1
000150                STA       (POINT2),Y
000151                JSR       MAKESUR0                ;NEXT WORD MUST BE ZERO.
```

```
000152                LDA       #>POINT1                ;SEGMENT POINTER
000153                STA       SWPPNT                  ;THIS MAKES LINE AT DOARE2 SAME AS "ADC POINT1".
000154                LDA       #<POINT1
000155                STA       SWPPNT+1
000156                JSR       DOAREL                  ;SEGMENT RELATIVE FIXES.
000157                LDA       #<HIGHTR                ;THIS MAKES LINE AT DOARE2 SAME AS "ADC HIGHTR"
000158                STA       SWPPNT+1                ;FOR PROCEDURE RELATIVE RELOCATION!
000159                LDA       #>HIGHTR
000160                STA       SWPPNT
000161                JSR       DOAREL
000162                JSR       MAKESUR0                ;NO INTERPRETER RELATIVE.
000163                DEC       KIMY                    ;DONE WITH ONE PROCEDURE.
000164                BEQ       DOLIB                   ;DONE WITH REGULAR RELOCATION.
000165                LDA       POINT2
000166                STA       PROCPNT
000167                LDA       POINT2+1
000168                STA       PROCPNT+1               ;MOVE ON TO NEXT PROCEDURE.
000169                JMP       DOAPROC
000170 DOLIB           EQU       *                       ;LINKER INFO NOW STARTS AT INVPNT.
000171                PLA                               ;GET BACK OLD PROCPNT.
000172                STA       POINT2+1                ;POINTER TO TABLE OF ADDRESSES.
000173                DEC       POINT2+1                ;SO (POINT2),Y WILL WORK WITH Y NEAR 256.
000174                PLA
000175                STA       POINT2
000176                LDA       INVPNT
000177                STA       PROCPNT                 ;POINTER TO LINKER INFO.
000178                LDA       INVPNT+1
000179                STA       PROCPNT+1
000180                LDY       #8
000181                LDA       (PROCPNT),Y             ;GET THE TYPE OF ENTRY.
000182 DOPAS1          BEQ       DOPAS2                  ;END OF TABLE.
000183                CMP       #$B                     ;PROCEDURE?
000184                BEQ       DOPA11
000185                CMP       #$C                     ;FUNCTION?
000186                BEQ       DOPA11
000187                CMP       #$6                     ;EXTERNAL DEFINITION?
000188                BEQ       DOPA11
000189                CMP       #$2                     ;EXTERNAL REFERENCES?
000190                BEQ       DOPA12
000191                JMP       INVERROR                ;ONLY THOSE THINGS LEAGAL.
000192 DOPA13          LDY       #12
000193                LDA       (PROCPNT),Y             ;FIND # OF REFERENCES.
000194                STA       TEMP
000195                AND       #$F8                    ;COMPUTE # OF 8 WORD BLOCKS USED.
000196                CMP       TEMP
000197                BEQ       *+8                     ;REALLY SHOULD USE LABELS.
000198                ADC       #$8
000199                BCC       *+4
000200                INC       PROCPNT+1
000201                ASL       A
000202                BCC       *+4
000203                INC       PROCPNT+1
000204                JMP       ADDPROC
000205 DOPA12          JSR       DOPA13
000206                JMP       DOPA14
000207 DOPA11          LDY       #10                     ;POINT TO PROC #.
000208                LDA       #0
000209                SEC
000210                SBC       (PROCPNT),Y             ;PROC #
000211                ASL       A                       ;PROC # * -2.
000212                TAY                               ;INDEX INTO ENTRY POINT TABLE.
000213                LDA       (POINT2),Y              ;THIS IS THE ENTRY POINT.
000214                TAX
000215                INY
000216                LDA       (POINT2),Y              ;FOR THAT PROCEDURE.
000217                LDY       #11                     ;REPLACE PROC# WITH IT'S ADDRESS.
000218                STA       (PROCPNT),Y             ;BACK INTO LINKER INFO.
000219                TXA
000220                DEY
000221                STA       (PROCPNT),Y
000222 DOPA14          JSR       NXTPROC                 ;POINT TO NEXT PROC.
000223                BNE       DOPAS1
000224 DOPAS2          LDA       INVPNT                  ;START PASS 2 BY RESETTING POINTER
000225                STA       PROCPNT                 ;TO BEGINNING OF LINKER INFO.
000226                LDA       INVPNT+1
000227                STA       PROCPNT+1
000228 DOPA20          LDY       #8
000229                LDA       (PROCPNT),Y             ;GET TYPE BYTE FOR THIS ENTRY.
000230 DOPA22          BNE       *+5
000231                JMP       DOPAS3                  ;DONE IF ZERO.
```

```
000232                CMP       #2
000233                BNE       DOPA29                ;LOOKING FOR REFERENCES TO RESOLVE.
000234                LDA       PROCPNT
000235                STA       POINT2
000236                LDA       PROCPNT+1
000237                STA       POINT2+1              ;SAVE CURRENT PROCPNT.
000238                LDA       #6
000239                STA       PROCFLG               ;LOOK FOR DEFINITIONS NOW.
000240                JSR       PERFIND
000241                BNE       DOPA25                ;FOUND IT IF NOT ZERO.
000242                JMP       INVERROR
000243 DOPA29          JSR       NXTPROC               ;GO LOOK AT NEXT.
000244                JMP       DOPA22                ;THIS WILL EVEN WORK ON TYPE 2'S.
000245 DOPA25          LDY       #10                   ;GOT A MATCH!
000246                LDA       (PROCPNT),Y           ;ADD OFFSST OF THIS
000247                CLC                             ;LABEL TO BEGINING OF THIS PROCEDURE.
000248                LDY       #12
000249                ADC       (PROCPNT),Y
000250                TAX
000251                DEY
000252                LDA       (PROCPNT),Y
000253                LDY       #13
000254                ADC       (PROCPNT),Y
000255                PHA                             ;NOW THAT WE HAVE THE ADDRESS OF THE .DEF
000256                TXA                             ;WE NEED TO CONVERT IT INTO AN
                                                                OFFSET INTO THE SEGMENT.
000257                SEC                             ;THAT OFFSET WILL THEN BE ADDED TO ALL .REF'S
000258                SBC       POINT1                ;BEGINNING OF SEGMENT.
000259                TAX
000260                PLA
000261                SBC       POINT1+1
000262                LDY       POINT2+1
000263                STA       POINT2+1
000264                STY       PROCPNT+1
000265                LDA       POINT2                ;RESTORE OLD PROCPNT, AND CLOBBER POINT2
000266                STX       POINT2                ;WITH VALUE OF THIS LABLE.
000267                STA       PROCPNT               ;POINTER INTO TYPE 2 (REFERENCE) FIELD.
000268                LDY       #12
000269                LDA       (PROCPNT),Y           ;# OF REFERENCES TO LABLE.
000270                STA       KIMY
000271                TAX
000272                BEQ       DOPA29
000273                JSR       NXTPROC               ;POINT TO TABLE OF REFERENCES.
000274 DOPA27          LDY       #0
000275                CLC
000276                LDA       POINT1                ;ADD REFERENCE OFFSET TO BASE ADDR.
000277                ADC       (PROCPNT),Y
000278                STA       OFFSST
000279                INY
000280                LDA       POINT1+1
000281                ADC       (PROCPNT),Y
000282                STA       OFFSST+1
000283                CLC
000284                DEY
000285                LDA       POINT2                ;GET VALUE OF LABLE.
000286                ADC       (OFFSST),Y
000287                STA       (OFFSST),Y
000288                INY
000289                LDA       POINT2+1
000290                ADC       (OFFSST),Y            ;THE VALUE OF THE LABLE IS ADDED
                                                                INTO THE REFERENCE.
000291                STA       (OFFSST),Y
000292                LDA       #2
000293                JSR       ADDPROC
000294                DEX
000295                BNE       DOPA27
000296                LDA       KIMY                  ;GOT TO SKIP OVER ALL THOSE ENTRYS.
000297                EOR       #$7
000298                CLC
000299                ADC       #1
000300                BCC       *+4
000301                INC       PROCPNT+1
000302                AND       #$7
000303                ASL       A
000304                BCC       *+4
000305                INC       PROCPNT+1
000306                JSR       ADDPROC
000307                JMP       DOPA20
000308 DOPAS3          LDA       INVPNT                ;GOT ALL LINKS RESOLVED!
000309                STA       PROCPNT
```

```
000310                STA       POINT2
000311                LDA       INVPNT+1
000312                STA       PROCPNT+1
000313                STA       POINT2+1
000314                LDY       #8
000315                LDA       (PROCPNT),Y             ;WHAT TYPE OF INFO IS FIRST?
000316 DOPA31          CMP       #$B                     ;.PROC?
000317                BEQ       DOPA37                  ;YES--SAVE A COPY.
000318                CMP       #$C
000319                BEQ       DOPA37                  ;.FUNC'S GET SAVED TOO.
000320                CMP       #$2
000321                BNE       DOPA32                  ;.REFS ARE VARIABLE LENGTH-- TREAT SPECIALLY.
000322                JSR       DOPA13                  ;SKIP OVER THE EXTRA REFERENCES.
000323 DOPA32          JSR       NXTPROC                 ;POINT TO NEXT ENTRY.
000324                BNE       DOPA31                  ;LOOP TILL DONE.
000325                LDA       POINT2                  ;NEW INVPNT AT END OF LINKAGE
000326                STA       INVPNT                  ;INFO.
000327                LDA       POINT2+1
000328                STA       INVPNT+1
000329                LDA       POINT1                  ;NEW PROCPNT IS BEGINING
000330                STA       PROCPNT                 ;OF PROCEDURE CODE.
000331                LDA       POINT1+1
000332                STA       PROCPNT+1
000333                JMP       DOFILE
000334 DOPA37          LDY       #15                     ;MOVE 16 BYTES.
000335 DOPA38          LDA       (PROCPNT),Y
000336                STA       (POINT2),Y              ;TRANSFER A BYTE.
000337                DEY
000338                BPL       DOPA38
000339                LDA       #16
000340                CLC
000341                ADC       POINT2
000342                STA       POINT2
000343                BCC       DOPA32                  ;UPDATE DESTINATION POINTER AND CONTINUE.
000344                INC       POINT2+1
000345                BNE       DOPA32                  ;ALWAYS.
000346 DOCRNCH         EQU       *                       ;MOVE PROCPNT<INVTAB.INVPNT
000347                LDA       PROCPNT
000348                STA       HIGHDS                  ;NOT NEEDED SINCE THEY ARE THE SAME.
000349                LDA       PROCPNT+1               ;FOR THE BLOCK MOVE.
000350                STA       HIGHDS+1
000351                LDA       INVPNT
000352                STA       HIGHTR
000353                LDA       INVPNT+1
000354                STA       HIGHTR+1
000355                LDA       PROCTABB
000356                STA       HIGHDSB
000357                STA       LOWTRB
000358                STA       HIGHTRB
000359                LDA       INVTAB
000360                STA       LOWTR
000361                LDA       INVTAB+1
000362                STA       LOWTR+1
000363                LDA       #2
000364                CLC
000365                ADC       HIGHTR                  ;INCLUDE 2 BYTES OF END MARK.
000366                STA       HIGHTR
000367                BCC       *+4
000368                INC       HIGHTR+1
000369                JSR       BLTUC                   ;SKRUNCH.
000370                LDA       HIGHDS
000371                STA       INVPNT
000372                LDA       HIGHDS+1
000373                STA       INVPNT+1
000374                LDA       HIGHDSB
000375                STA       INVPNTB
000376                RTS
000377 DEC2PROC        LDA       #2
000378                STA       TEMP
000379                LDA       PROCPNT                 ;SUBTRACT A FROM PROCPNT.
000380                SEC
000381                SBC       TEMP
000382                STA       PROCPNT
000383                BCS       *+4
000384                DEC       PROCPNT+1
000385                RTS
000386 MAKESUR0        JSR       DEC2PROC                ;BUMP POINTER.
000387                LDY       #0
000388                LDA       (PROCPNT),Y
000389                BNE       INVERROR
```

```
000390                INY
000391                LDA       (PROCPNT),Y             ;THIS CODE MAKES SURE
000392                BNE       INVERROR                ;NEXT 2 GUYS ARE ZERO.
000393                RTS
000394 INVERROR        LDX       #ERRIN                  ;BAD INVOKE!
000395                JMP       ERROR
000396 DOAREL          JSR       DEC2PROC                ;HANDLES ONE RELATIVE RELOCATION TABLE.
000397                LDY       #1
000398                LDA       (PROCPNT),Y             ;GET COUNT OF SELF REL POINTERS.
000399                STA       TEMPTR+1
000400                DEY
000401                LDA       (PROCPNT),Y
000402                STA       TEMPTR
000403                BNE       DOARE1
000404 DOARE0          DEC       TEMPTR+1
000405                BMI       DOARTS                  ;ALL DONE.
000406 DOARE1          DEC       TEMPTR
000407                JSR       RELPROC                 ;OFFSST=PROCPNT-(PROCPNT)
000408                CLC
000409                LDY       #0
000410                LDA       (OFFSST),Y
000411                ADC       (SWPPNT),Y
000412                STA       (OFFSST),Y
000413                INY
000414                LDA       (OFFSST),Y
000415                ADC       (SWPPNT),Y
000416                CLC
000417                ADC       #$20
000418                STA       (OFFSST),Y
000419                LDA       TEMPTR
000420                BNE       DOARE1                  ;USUALLY GOES.
000421                BEQ       DOARE0                  ;ALWAYS.
000422 DOARTS          RTS                               ;DONE.
000423 RELPROC         JSR       DEC2PROC                ;BUMP PROCPNT
000424                SEC
000425                LDY       #0
000426                LDA       PROCPNT
000427                SBC       (PROCPNT),Y
000428                STA       OFFSST
000429                INY
000430                LDA       PROCPNT+1
000431                SBC       (PROCPNT),Y
000432                STA       OFFSST+1
000433                RTS
000434 ADDPROC         STA       TEMP
000435                CLC
000436                LDA       PROCPNT
000437                ADC       TEMP
000438                STA       PROCPNT
000439                BCC       *+4
000440                INC       PROCPNT+1
000441                RTS
000442 NXTPROC         LDA       #16
000443                JSR       ADDPROC
000444                LDY       #8
000445                LDA       (PROCPNT),Y
000446                RTS
000447 PERFORM         LDA       #$B                     ;LOOKING FOR A PROCEDURE.
000448                STA       PROCFLG
000449                LDA       #0
000450 PERFEXF         STA       NAMPNT
000451                PLA                               ;PULL & SAVE THE
000452                STA       SAFE                    ;  RETURN ADDRESS WHERE
000453                PLA                               ;  IT WILL BE SAFE
000454                STA       SAFE+1
000455                LDA       #0
000456                STA       NPARAMS
000457                STA       NPOINTS
000458                LDA       #'                      '
000459                LDX       NAMPNT
000460                INX
000461                LDY       #8
000462 PERFE1          STA       NAMBUF-1,X
000463                INX
000464                DEY
000465                BNE       PERFE1
000466                LDX       NAMPNT
000467                JSR       CHRGOT
000468                BNE       PERFE3
000469 PERERR:         JMP       SNERR
```

```
000470 PERFE05          JMP       PERFE5
000471 PERFE2           JSR       CHRGET                ;GET FUNCTION NAME.
000472 PERFE3           BEQ       PERFE05               ;LAST BYTE?
000473                  BCC       PERFE35
000474                  JSR       ISLETC
000475                  BCS       PERFE35
000476                  CMP       #'('                  ;PARAMETER LIST COMMING?
000477                  BEQ       PERFE4
000478                  TXA
000479                  BEQ       PERERR
000480                  JMP       PERFE6
000481 PERFE35          INX
000482                  CMP       #'Z'+1
000483                  BCC       *+4
000484                  SBC       #$20
000485                  STA       NAMBUF-1,X            ;BUILD PROCEDURE NAME IN NAMBUF.
000486                  BNE       PERFE2                ;ALWAYS.
000487 ;
000488 ;  INTERPRET PARAMETER LIST
000489 ;
000490 PERFE4           LDA       #$20                  ;DON'T CARE ABOUT VAR TYPE NOW
000491                  STA       VALTYP
000492                  JSR       CHRGET
000493                  CMP       #'@'                  ;ADDRESS PARAMETER?
000494                  BEQ       PERVAL1               ;YES, IT'S OK
000495                  CMP       #'%'                  ;INTEGER?
000496                  BEQ       PERINT1               ;YES, OK
000497                  CMP       #'&'                  ;LONG INTEGER?
000498                  BEQ       PERLONG1              ;YES, OK
000499                  CMP       #'$'                  ;STRING?
000500                  BNE       *+5                   ;NO! OK
000501                  JMP       TMERR                 ;CAN'T PASS STRINGS (OR FUZZ BALLS)
000502                  JSR       DOPAR                 ;SO MUST BE REAL (I HOPE)
000503                  JSR       CONV2FLT
000504                  JSR       ROUNDER
000505                  LDA       FACEXP
000506                  PHA
000507                  LDA       FACSGN                ;PUSH THE FAC.
000508                  ORA       #$7F
000509                  AND       FACHO
000510                  PHA
000511                  LDA       FACMOH
000512                  PHA
000513                  LDA       FACMO
000514                  PHA
000515                  LDA       #2
000516                  BNE       PERVAL2               ;ALWAYS
000517 PERINT1          JSR       CHRGET
000518                  JSR       DOPAR
000519                  JSR       CONV2INT
000520                  JSR       AYINT
000521                  LDA       FACMO                 ;HIGH BYTE FIRST
000522                  PHA
000523                  LDA       FACMO+1
000524                  PHA
000525                  LDA       #1                    ;TOOK ONE WORD.
000526                  BNE       PERVAL2               ;ALWAYS
000527 PERLONG1         JSR       CHRGET
000528                  JSR       DOPAR
000529                  JSR       CONV2LNG
000530                  LDX       #0
000531 PERLNG1          LDA       FAC,X
000532                  PHA
000533                  INX
000534                  CPX       #8
000535                  BCC       PERLNG1
000536                  LDA       #4                    ;FOUR WORDS
000537                  BNE       PERVAL2
000538 PERVAL1          JSR       CHRGET                ;EAT THE @
000539                  JSR       PTRGET                ;GET POINTER
000540                  LDA       VARPNT+1
000541                  PHA
000542                  LDX       NPOINTS
000543                  STA       BANKPNT+1,X
000544                  LDA       VARPNT
000545                  PHA
000546                  STA       BANKPNT,X
000547                  LDA       VARPNTB
000548                  STA       BANKPNTB,X
000549                  INC       NPOINTS
```

```
000550                  INC       NPOINTS
000551                  LDA       #1
000552 PERVAL2          CLC
000553                  ADC       NPARAMS                 ;ADD UP NUMBER OF PARAMETER BYTES PUSHED.
000554                  STA       NPARAMS
000555                  JSR       CHRGOT
000556                  CMP       #')'
000557                  BEQ       PERFE6                  ;MUST END ON CLOSE PAREN.
000558                  CMP       #','
000559                  BEQ       *+5
000560                  JMP       SNERR
000561                  JMP       PERFE4
000562 PERFE6           JSR       CHRGET
000563 PERFE5           LDA       #>NAMBUF                ;POINTER TO PROCEDURE NAME.
000564                  CLC
000565                  ADC       NAMPNT
000566                  STA       POINT2
000567                  LDA       #<NAMBUF
000568                  ADC       #0
000569                  STA       POINT2+1
000570                  LDA       #0
000571                  STA       POINT2B
000572                  JSR       PERFIND                 ;FIND THE ENTRY.
000573                  BEQ       PERFERR
000574                  LDY       #12
000575                  LDA       (PROCPNT),Y             ;# OF PARAMETER WORDS.
000576                  CMP       NPARAMS
000577                  BEQ       PTMOK                   ;Parameter types OK
000578                  JMP       TMERR                   ;Otherwise, TYPE MISMATCH ERROR
000579 PTMOK            DEY
000580                  LDA       (PROCPNT),Y
000581                  STA       JMPER+2                 ;ADDRESS OF ENTRY.
000582                  DEY
000583                  LDA       (PROCPNT),Y
000584                  STA       JMPER+1
000585                  LDA       PROCFLG                 ;GET TYPE OF ROUTINE
000586                  CMP       #$C                     ;IS IT AN EXFN. (OR EXFN%.)?
000587                  BNE       *+6                     ;NO, SKIP TO DO IT
000588                  PHA                               ;OTHERWISE PUSH 4 DUMMY
000589                  PHA                               ;BYTES TO ALLOW ROOM FOR
000590                  PHA                               ;THE RETURNED VALUE
000591                  PHA
000592                  JMP       JUMPDO                  ;Go call the Mach. lang. routine
000593 EXFNS            EQU       *                       ;EXFN%. code starts here
000594                  JSR       PERFEX1
000595                  PLA
000596                  TAY                               ;GET RETURNED VALUE.
000597                  PLA
000598                  JSR       GIVAYF                  ;SLAP VALUE INTO FAC.
000599                  JMP       RESTNAM
000600 PERFEX1          EQU       *
000601                  LDA       #$C
000602                  STA       PROCFLG
000603                  LDA       NAMPNT
000604                  CLC
000605                  ADC       #8
000606                  JMP       PERFEXF
000607 EXFN             EQU       *
000608                  JSR       PERFEX1
000609                  PLA                               ;PULL OF RESULT.
000610                  STA       FACMO
000611                  PLA
000612                  STA       FACMOH
000613                  PLA
000614                  STA       FACSGN
000615                  ORA       #$80
000616                  STA       FACHO
000617                  PLA
000618                  STA       FACEXP
000619                  LDA       #0
000620                  STA       FACLO
000621                  STA       FACOV
000622 RESTNAM          PLA
000623                  PHA
000624                  CMP       #>EVALRET
000625                  BNE       PERFERR
000626                  LDA       NAMPNT
000627                  SEC
000628                  SBC       #8
000629                  STA       NAMPNT
```

```
000630                JSR       CHRGOT                  ;END OF EXPRESSION?
000631                BEQ       *+5                     ;IF NOT, THEN BACK UP TXTPTR ONE.
000632                JMP       DECTPT                  ;BACK UP THE TXTPTR, TO CONTINUE EXPRESSION.
000633                RTS
000634 PERFERR         EQU       *
000635                JMP       ERRGUF
000636 PERFIND         LDA       SEGNUMB
000637                BEQ       PERFI05                 ;NO.
000638                LDA       INVTAB
000639                STA       PROCPNT                 ;ENTRY OF TYPE /KIMY/
000640                LDA       INVTAB+1                ;WITH ENTRY NAME @POINT2.
000641                STA       PROCPNT+1               ;START AT TOP OF TABLE.
000642                LDA       INVTABB
000643                STA       PROCPNTB
000644                LDY       #8
000645                LDA       (PROCPNT),Y             ;GET TYPE OF ENTRY.
000646 PERFI01         BEQ       PERFI05                 ;END OF TABLE.
000647                CMP       #2
000648                BNE       PERFI02
000649                JSR       DOPA13
000650                JMP       PERFI04
000651 PERFI02         CMP       PROCFLG                 ;TYPE OF ENTRY WHAT WE'RE LOOKING FOR?
000652                BNE       PERFI04                 ;NO.
000653 PERFI06         DEY
000654                BMI       PERFI05                 ;ALL MATCHED, THIS IS IT!
000655                LDA       (POINT2),Y              ;DO THE NAMES MATCH?
000656                CMP       (PROCPNT),Y             ;ENTRY NAME, ALL BYTES MUST MATCH.
000657                BEQ       PERFI06                 ;THIS ONE DID, TRY NEXT.
000658 PERFI04         JSR       NXTPROC                 ;DARN.  THIS ONE IS NOT IT!
000659                JMP       PERFI01                 ;WELL, TRY NEXT.
000660 PERFI05         RTS                               ;Z FLAG SET IFF NOT FOUND.
000661 INVERR1         JMP       OMERR
000662
000663 ; ###############################################################################################
000664 ; #   END OF FILE:  INVOKE.TEXT
000665 ; #   LINES      :  656
000666 ; #   CHARACTERS :  28768
000667 ; ###############################################################################################
```

```
+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 667   CHARACTERS: 29318
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "INVOKE1.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:36 PM
|  Modified: Wednesday, December 31, 1997             4:37:14 PM
|
+------------------------------------------------------------------------------

000001 ; ###########################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  INVOKE1.TEXT
000004 ; ###########################################################################################
000005
000006 INVOKE          EQU       *                       ;INVOKE MACHINE LANGUAGE UTILITIES.
000007                 LDA       #0                      ;Set COMMAND call flag
000008                 STA       CMDFLG
000009                 JSR       SETPROG
000010                 LDA       #0
000011                 JSR       SCRUNCH                 ;CRUNCH UP MEMORY SO THERE'S ROOM.
000012                 LDA       SEGNUMB                 ;DO WE CURRENTLY HAVE ANY UTILITIES?
000013                 BEQ       INVOK1                  ;NO, SKIP THE RELEASE OF MEMORY.
000014                 BRK
000015                 DFB       MRLS                    ;RELEASE SEGMENT.
000016                 DW        SEGTAB1
000017 INVOK1          JSR       CHRGOT                  ;ANY FILES TO INVOKE?
000018                 BNE       INVOK3
000019                 JMP       INVEXP2
000020 INVOK3          LDA       #0
000021                 STA       BANKPNT
000022                 STA       BANKPNT+1
000023                 JSR       SVTXT                   ;OPEN THE FILE, AND CLOSE IT AGAIN.
000024 INVOK4          LDA       #1                      ;SO SOS BUFFERS WILL GET OUT OF
                                                             THE WAY OF MY SEGMENT.
000025                 LDX       #PCODTYP                ;OPEN UP A PCODE FILE.
000026                 JSR       OPENIT
000027                 LDA       FEOF                    ;GOT EOF FROM OPNPRTB
000028                 CLC
000029                 ADC       BANKPNT+1
000030                 STA       BANKPNT+1
000031                 LDA       FEOF+1
000032                 ADC       BANKPNT
000033                 BCS       INVERR1
000034                 SEC
000035                 SBC       #2
000036                 BCC       INVERR1                 ;AT LEAST 2 WASTED PAGES.
000037                 STA       BANKPNT
000038                 JSR       CLSEND                  ;CLOSE IT AGAIN.
000039                 JSR       CHRGOT
000040                 BEQ       *+8                     ;DON'T YOU HATE THESE?
000041                 JSR       CHKCOM                  ;COMMA COMES NEXT.
000042                 JMP       INVOK4
000043                 LDX       BANKPNT
000044                 INX
000045                 BEQ       INVERR1
000046                 STX       PGCNT2
000047                 LDX       #0
000048                 STX       PGCNT2+1
000049                 JSR       RSTTXT                  ;RETORE TXTPTR.
000050                 BRK
000051                 DFB       MFND                    ;FIND A SEGMENT OF MINIMAL SIZE (IN THIS BANK).
000052                 DW        SEGTAB2
000053                 BNE       INVERR1                 ;INVOKE ERRROR.
000054                 LDA       SEGNUM2
000055                 STA       SEGNUMB
000056                 STA       SEGNUM3                 ;SO CHANGE AND REMOVE SEG. KNOW IT.
000057                 LDA       BASPTR
000058                 ORA       #$80
000059                 STA       INVBNK
000060                 TAY
000061                 LDA       BASPTR+1                ;TRY TO EAT UP THE WHOLE BANK.
000062                 SEC
000063                 SBC       #$20
000064                 JSR       FIXSBC
000065                 STA       INVTAB+1
000066                 STY       INVTABB
000067                 STY       PROCTABB
000068                 LDA       LIMPTR
000069                 ORA       #$80
000070                 TAY
000071                 LDA       LIMPTR+1                ;SET UP PROCTAB FROM LIMPTR.
```

```
000072                  SEC
000073                  SBC     #$20                    ;MEMORY POINTERS RETURNED BY SOS MUST BE FIXED.
000074                  JSR     FIXSBC
000075                  CPY     INVTABB
000076                  BEQ     *+5
000077                  CLC
000078                  ADC     #MAXPG-MINPG
000079                  STA     PROCTAB+1
000080                  LDA     #0
000081                  STA     PROCTAB                 ;SET UP MEMORY LIMITS.
000082                  STA     INVTAB
000083                  JSR     DOINVO                  ;DO THE ACTUAL INVOKING.
000084                  LDA     INVPNT                  ;HOW MANY PAGES ARE LEFT OVER?
000085                  CMP     INVTAB
000086                  LDA     INVPNT+1
000087                  SBC     INVTAB+1
000088                  AND     #$7F
000089                  BEQ     INVEXP1
000090                  STA     PGECNT                  ;SCRUNCH UP THIS SEGMENT AS FAR AS POSSIBLE.
000091                  LDA     INVTAB+1
000092                  CLC
000093                  ADC     PGECNT                  ;FIX UP POINTER.  IT MAY END UP INVPNT ANYWAY.
000094                  STA     INVTAB+1
000095                  LDA     #0
000096                  STA     PGECNT+1
000097                  LDA     INVPNT
000098                  STA     INVTAB
000099                  BRK
000100                  DFB     MCHG
000101                  DW      SEGTAB3
000102 INVEXP1          LDA     #0                      ;ALL FILES ARE NOW IN.
000103                  JMP     EXPAND                  ;ENPAND THE MAIN MODULE BACK AND END.
000104 INVEXP2          LDA     #0
000105                  STA     SEGNUMB                 ;NOTHING TO INVOKE SO DON'T REQUEST A SEG.
000106                  BEQ     INVEXP1                 ;ALWAYS.
000107 TRYSEG           BRK
000108                  DFB     MFND                    ;FIND SEG FOR BASIC PROG., DATA.
000109                  DW      SEGTAB7
000110                  BNE     TRYSEG                  ;SHOULD WORK THE SECOND TIME...
000111                  LDA     SEGNUM7                 ;WHEN CALLED WITH A REASONABLE SIZE.
000112                  STA     SEGNUM5
000113                  STA     SEGNUM6                 ;FOR EXPAND AND SCRUNCH.
000114                  LDY     #0
000115                  STY     MEMSIZ
000116                  INY
000117                  STY     RAMLOC
000118                  LDA     BASBNK                  ;BASE SEGMENT BANK
000119                  ORA     #$80
000120                  TAY
000121                  LDA     BASBNK+1                ;BASE SEGMENT PAGE
000122                  SEC
000123                  SBC     #$20
000124                  JSR     FIXSB2                  ;INSURE IN RANGE OF 2-$82
000125                  STA     RAMLOC+1
000126                  STY     RAMLOCB
000127                  LDA     LIMBNK                  ;LIMIT SEGMENT BANK
000128                  ORA     #$80
000129                  TAY
000130                  LDA     LIMBNK+1                ;LIMIT SEGMENT PAGE
000131                  SEC
000132                  SBC     #$1F
000133                  JSR     FIXSB2
000134                  STA     MEMSIZ+1
000135                  STY     MEMSIZB
000136                  RTS
000137 SEGTAB1          DFB     1
000138 SEGNUMB          DFB     0
000139 SEGTAB2          DFB     6
000140                  DFB     0
000141                  DFB     $12
000142 PGCNT2           DW      1
000143 BASPTR           DW      0
000144 LIMPTR           DW      0
000145 SEGNUM2          DFB     0
000146 SEGTAB3          DFB     3
000147 SEGNUM3          DFB     0
000148                  DFB     0                       ;MOVES BASE-PTR UP.
000149 PGECNT           DW      0
000150 SEGTAB4          DFB     4                       ;LOCK DOWN PAGE 0,1 IN BANK ZERO BECAUSE
000151                  DFB     0                       ;THEY CAN'T BE VIRTUALLY ADDRESSED.
```

```
000152                DFB       $20
000153                DFB       0
000154                DFB       $21
000155                DFB       $13
000156 SEGNUM4        DFB       0
000157 SEGTAB5        DFB       3
000158 SEGNUM5        DFB       $11
000159                DFB       3                       ;MOVE LIMIT DOWN.
000160 SEGSIZ5        DW        0
000161 SEGTAB6        DFB       3
000162 SEGNUM6        DFB       $11
000163                DFB       2
000164 SEGSIZ6        DW        0
000165 SEGTAB7        DFB       6                       ;6 PARAMETERS
000166                DFB       2                       ;SRCM MODE 2
000167                DFB       $11                     ;GIVE SEGMENT ID #$11
000168                DFB       $FF                     ;PAGE COUNT (IF ERR, SOS STUFFS SIZE OF
000169                DFB       $FF                     ;  LARGEST AVAILABLE PGS ON FIRST TIME THRU)
000170 BASBNK         DW        0                       ;WHEN SUCCESSFUL, HOLDS BASE SEG BANK, PAGE
000171 LIMBNK         DW        0                       ;WHEN SUCCESSFUL, HOLDS LIMIT SEG BANK, PAGE
000172 SEGNUM7        DFB       0                       ;  AND SEGMENT NUMBER
000173 PREFTAB        DFB       2
000174                DW        CATBUF+1
000175                DFB       >BUF-CATBUF-4
000176 DATETAB        DFB       1
000177                DW        CATBUF+2
000178 PREFTB2        DFB       1
000179                DW        NAMBUF
000180 PREFTB3        DFB       2
000181                DW        NAMBUF
000182                DFB       128
000183                SBTL      "EXPAND, SCRUNCH"
000184 *
000185 * SCRUNCH
000186 *  COMPACTIFIES THE USER-MEMORY BY N PAGES,
000187 *  WHERE N= VALUE IN A REG. UPON CALL.
000188 *  IF A = 0 THEN SCRUNCH ALL THE WAY.
000189 *
000190 SCRUNCH         PHA
000191                JSR       GARBA2                  ;GARBAGE COLLECT.
000192                PLA
000193                TAX                               ;SAVE VALUE.
000194                LDA       FRETOP                  ;COMPUTE MAX # OF PAGES
000195                CMP       STREND                  ;THAT WE COULD POSSIBLY
000196                LDA       FRETOP+1                ;CRUNCH.
000197                SBC       STREND+1
000198                STA       INDEX2                  ;INDEX2=# OF PAGES TO CRUNCH (MAX).
000199                LDA       FRETOPB
000200                SBC       STRENDB
000201                STA       INDEX2+1
000202                ASL       INDEX2                  ;FIX THE FACT THAT 32K BANKS EXIST.
000203                LSR       INDEX2+1                ;BY SHIFTING LOW BIT OF HIGH BYTE
000204                ROR       INDEX2                  ;INTO HIGH BIT OF LOW BYTE.
000205                TXA                               ;HOW MANY PAGES?
000206                BEQ       SCRUN1                  ;0 MEANS CRUNCH AS MUCH AS POSSIBLE.
000207                LDX       INDEX2+1                ;MORE THAN 256 PAGES?
000208                BNE       SCRUN0                  ;YES, DO # SPECIFIED.
000209                CMP       INDEX2
000210                BCS       SCRUN1                  ;IF N LOOKS BIGGER THAN MAX, JUST DO MAX.
000211 SCRUN0         STA       INDEX2                  ;SAVE NEW # PAGES TO MOVE.
000212                LDA       #0
000213                STA       INDEX2+1
000214 SCRUN1         EQU       *                       ;INDEX NOW # OF PAGES TO SCRUNCH.
000215                LDA       INDEX2
000216                ORA       INDEX2+1                ;CAN ONLY MOVE 0?
000217                BNE       *+5                     ;CAN'T SCRUNCH ANY MORE--
000218                JMP       OMERR                   ;OUT OF MEMORY ERROR.
000219                LDA       INDEX2
000220                ASL       A                       ;HEADER = INDEX2 EXCEPT THAT THE HIGH BIT OF
000221                TAX                               ;INDEX2 IS SHIFTED INTO HEADER+1
000222                LDA       INDEX2+1                ;IN OTHER WORDS, HEADER LOOKS LIKE
000223                ROL       A                       ;A PAGE--BANK PAIR EQUAL TO INDEX2'S 16 BITS.
000224                STA       HEADER+1
000225                TXA
000226                LSR       A
000227                STA       HEADER
000228                LDA       FRETOP
000229                STA       INDEX1
000230                STA       LOWTR
000231                LDA       FRETOP+1
```

```
000232                STA       INDEX1+1
000233                LDY       FRETOPB
000234                STY       INDEX1B
000235                SEC
000236                SBC       HEADER
000237                JSR       FIXSBC
000238                STA       LOWTR+1                 ;OH, WHAT I'D GIVE FOR A 68000!!
000239                TYA
000240                SBC       HEADER+1
000241                STA       LOWTRB
000242                LDA       STREND
000243                PHA
000244                LDA       STREND+1                ;GOT TO SAVE STREND.
000245                PHA
000246                LDA       STRENDB
000247                PHA
000248                LDA       HIMEM
000249                STA       STREND
000250                LDA       HIMEM+1
000251                STA       STREND+1
000252                LDA       HIMEMB
000253                STA       STRENDB
000254                LDA       #0
000255                STA       DELTA                   ;ZERO OUT DELTA
000256                STA       DELTA+1                 ;SO POINTERS DON'T GET UPDATED
000257                STA       DELTAB                  ;BY THIS MOVE.
000258                JSR       MVDWN                   ;MOVE IT TO A LOWER ADDRESS.
000259                PLA
000260                STA       STRENDB
000261                PLA
000262                STA       STREND+1
000263                PLA
000264                STA       STREND
000265                LDA       LOWTR                   ;FIX FRETOP
000266                STA       FRETOP
000267                LDA       LOWTR+1
000268                STA       FRETOP+1
000269                LDA       LOWTRB
000270                STA       FRETOPB
000271                LDA       HIMEM+1
000272                LDY       HIMEMB
000273                SEC
000274                SBC       HEADER
000275                JSR       FIXSBC                  ;I WISH I WERE A MOTOROLA 68000,
000276                STA       HIMEM+1                 ;YES THAT IS WHAT I'D TRUELY LIKE TO BE,
000277                TYA                               ;CUZ IF I WERE A M. 68000.
000278                SBC       HEADER+1                ;EVERYONE WOULD LOVE TO PROGRAM ME!
000279                STA       HIMEMB
000280                LDA       INDEX2
000281                STA       SEGSIZ5
000282                LDA       INDEX2+1
000283                STA       SEGSIZ5+1
000284                BRK
000285                DFB       MCHG                    ;CHANGE SEG.
000286                DW        SEGTAB5
000287                BEQ       *+5
000288                JMP       SERROR
000289                RTS
000290 EXPAND          EQU       *
000291                LDY       #0                      ;DOES THE OPPOSITE OF SCRUNCH.
000292                TAX                               ;MAX?
000293                BNE       EXPAN0
000294                LDY       #15                     ;SOMETHING BIG.
000295                LDA       #255                    ;CLOSE ENOUGH.
000296 EXPAN0          STA       SEGSIZ6
000297                STY       SEGSIZ6+1
000298 EXPAN1          EQU       *
000299                BRK
000300                DFB       MCHG
000301                DW        SEGTAB6
000302                BNE       EXPAN1                  ;CRUDE BUT EFFECTIVE.  IF IT CAN'T BE DONE,
000303                LDA       SEGSIZ6
000304                ORA       SEGSIZ6+1
000305                BNE       EXPAN2
000306                TYA                               ;Y NON-ZERO IF CALLED WITH ARGUMENT OF ZERO.
000307                BNE       *+5
000308                JMP       OMERR                   ;TRIED TO EXPAND BUT NO MEMORY ERROR.
000309                RTS
000310 EXPAN2          LDA       HIMEM                   ;DO HOWEVER MUCH YOU CAN.  HANG ON AN ERROR.
000311                STA       HIGHTR
```

```
000312                STA       HIGHDS
000313                CLC
000314                LDA       HIMEM+1                 ;SET UP POINTERS FOR THE BLOCK MOVE.
000315                STA       HIGHTR+1
000316                LDY       HIMEMB
000317                STY       HIGHTRB
000318                ADC       SEGSIZ6
000319                JSR       FIXADC
000320                STA       HIMEM+1
000321                STA       HIGHDS+1
000322                TYA
000323                ADC       SEGSIZ6+1
000324                ADC       SEGSIZ6+1               ;YES TWO!  SINCE BANKS ARE ONLY 32K.
000325                STA       HIMEMB
000326                STA       HIGHDSB
000327                LDA       FRETOP
000328                STA       LOWTR
000329                LDA       FRETOP+1
000330                STA       LOWTR+1
000331                LDA       FRETOPB
000332                STA       LOWTRB
000333                JSR       BLTUC                   ;DO THE MOVE.
000334                LDA       HIGHDS
000335                STA       FRETOP
000336                LDA       HIGHDS+1
000337                STA       FRETOP+1
000338                LDA       HIGHDSB
000339                STA       FRETOPB
000340                RTS
000341
000342 ; ###############################################################################################
000343 ; #   END OF FILE:  INVOKE1.TEXT
000344 ; #   LINES      :  335
000345 ; #   CHARACTERS :  14762
000346 ; ###############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 346   CHARACTERS: 15314
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3PRU1.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:30 PM
|  Modified: Wednesday, December 31, 1997             4:37:07 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3PRU1.TEXT
000004 ; ###############################################################################################
000005
000006                  SBTL      "PRINT USING "
000007                  REP       70
000008 *
000009 *        !! PRINT USING !!
000010 *
000011 *  1.  Locate the IMAGE or SPEC list and be sure it is not a NULL
000012 *      string or a missing statement.  A Base Pointer is set up
000013 *      and its length is determined for all types.
000014 *
000015 *  2.  Locate the beginning of the Expression List.  TXTPTR is set
000016 *      up to point at this.  This also involves Syntaxing the
000017 *      PRINT USING statement.
000018 *
000019 *  3.  The Expression List is then done LEFT to RIGHT.
000020 *    A. The next Expression is evaluated and numerics are converted
000021 *       to 1 digit/byte BCD.
000022 *    B. Specs are then processed until a variable Spec is returned.
000023 *    C. Type match of Spec vs Expr is verified.
000024 *    D. The required NUM to STR edit is done and the String result
000025 *       is output.
000026 *
000027 *  4.  When the Expression List is exhausted, the Spec List is
000028 *      processed to print trailing LITERAL Specs and the trailing
000029 *      CR is sent if no ; ends the statement.
000030 *
000031                  REP       70
000032 DELIM            EQU       FAC+1                     ; SPEC SCANNER OUTPUT
000033 REP              EQU       FAC+2
000034 DIGB4DPT         EQU       FAC+24                    ;# DIGIT B4 DPT
000035 DIGAFDPT         EQU       FAC+25                    ;# DIGITS AFTER DPT
000036 *FACSGN    EQU   FAC+5         ;FOR REF
000037 SAVLEN           EQU       FAC+6                     ; USED IN STRINGSTUFF
000038 SPECTYP          EQU       FAC+7                     ;TYPE OF SPEC
000039 DIGCTR           EQU       FAC+8                     ;INDEX IN BCDSTR
000040 DIGEXP           EQU       FAC+9                     ;EXP OF REQUEST DIGIT
000041 USVSTRL          EQU       FAC+9                     ;STR LEN SAVE BYTE
000042 DPTNDX           EQU       FAC+10                    ;DPT POSITION IN MASK
000043 NMASK            EQU       FAC+11                    ;NUM SUBSPEC DIGIT TYPE
000044 EXPADJ           EQU       FAC+11                    ;ENG ADJUST TEMP
000045 EMASK            EQU       FAC+12                    ;SPEC SYNTAX CONTROL BITS
000046 FLTMSK           EQU       FAC+13                    ;FLOAT SYMS ORDER&SYNTAX BITS
000047 LITRLCH          EQU       FAC+14                    ;STRING STUFF
000048 SPCB4DIG         EQU       FAC+14                    ;ENG NOTATAION PTR
000049 MASKPT           EQU       FRESPC
000050 VSPEC            EQU       FAC+29                    ;INFINITE LOOP PREVENTION FLAG
000051 ENDFLG           EQU       VSPEC
000052 MSKNDX           EQU       FAC+16                    ;INDEX 4 CREATE.. LEN 4 USE
000053 SPCPTR           EQU       FAC+17                    ;PTR TO IMAGE
000054 SPCPTRH          EQU       FAC+17+1
000055 SPCNDX           EQU       FAC+19                    ;CURNT INDEX IN IMAGE STR
000056 SPCLEN           EQU       FAC+20                    ; MAX LEN OF IMAGE STRING
000057 TENZNDX          EQU       FAC+21                    ;10**0 DIGIT INDEX
000058 CMACTR           EQU       FAC+22                    ;MOD 3 CTR FOR , INSERTION
000059 HALF             EQU       FAC+22
000060 *USAVP     EQU   FAC+22        ;STRING P REG SAVE
000061 CMAFILL          EQU       FAC+23                    ;*,SP,Z FILL;MSB IS CMA FLAG
000062 SPCPTRB          EQU       SPCPTR+SYSPAG
000063                  PAGE
000064 ********************************
000065 STRTYP           EQU       $FF
000066 ZTYP             EQU       $41
000067 CMATYP           EQU       $44
000068 DTYP             EQU       $42
000069 MSKLEN           EQU       32                        ;NUMERIC MASK LENGTH
000070 ********************************
000071 PRUSING          JSR       CHRGET                    ;GET 1ST CHAR
000072                  BCC       UIMAGE                    ;PR USING LINENUM FORM
```

```
000073                CMP       #'"'                  ;PR USING "SPEC"; ???
000074                BEQ       ULITRL                ;YES, BUILD PTR&LEN
000075                JSR       ISLETC                ;A VARIABLE NAME NEXT?
000076                BCC       USNERR
000077                JSR       PTRGET                ;GO GET THE VARIABLE
000078                CPX       #STRTYP               ; LIKE IT MUST BE?
000079                BNE       USNERR
000080                STA       INDEX
000081                STY       INDEX+1               ;VARPNT SAVED AT INDEX.
000082                LDA       VARPNTB
000083                STA       INDEXB
000084                JSR       NOTNOW                ;MAKE INDEX POINT TO THE ACTUAL STRING.
000085                STA       SPCLEN                ;LENGTH OF STRING.
000086                STX       SPCPTR                ;LOW BYTE OF POINTER.
000087                STY       SPCPTRH               ;HIGH BYTE OF POINTER.
000088                LDA       INDEXB
000089                STA       SPCPTRB
000090                JMP       UGOTSPT
000091 ULITRL          LDY       #0                    ; NOW SCAN THE LITERAL
000092 ULITS           JSR       CHRGET                ; GET NEXT CHR
000093                BEQ       USNERR                ;ENDED TO EARLY!!
000094                CPY       #0                    ;1ST TIME THRU?
000095                BNE       ULITX
000096                LDX       TXTPTR
000097                STX       SPCPTR
000098                LDX       TXTPTR+1
000099                STX       SPCPTR+1              ;SAVE BEGIN ADDRESS
000100                LDX       TXTPTRB
000101                STX       SPCPTRB
000102 ULITX           CMP       #'"'
000103                BEQ       ULITE                 ; YES
000104                INY                             ; COUNT THIS ONE
000105                BNE       ULITS                 ;LOOK FURTHER
000106 ULITE           STY       SPCLEN                ;ZERO LENGTH ?
000107                TYA                             ;NULL LITERAL ??
000108                BEQ       USNERR                ;YESSS SERRR YREEE
000109                JSR       CHRGET                ;GET SO I CAN GOT NEXT
000110                BNE       UGOTSPT               ;GO ON IF NOT DELIM!!!
000111 USNERR          JMP       SNERR                 ; OFF TO SYNTAX ERROR
000112 UIMERR          LDX       #ERRUS                ;NOT IMAGE ERROR
000113                JMP       ERROR
000114                PAGE
000115 UIMAGE          EQU       *                     ;GO FIND IMAGE STMT!!
000116                JSR       LINGET                ;CVRT LINNUM TO BIN
000117                LDA       TXTPTR
000118                PHA
000119                LDA       TXTPTR+1
000120                PHA                             ;SAVE PRU PTR
000121                LDA       TXTPTRB
000122                PHA
000123                JSR       GOTOB                 ;GO FIND LINNUM & SET TXTPTR
000124                LDY       #3                    ;MOVE TXT PTR UP
000125                JSR       ADDON                 ; MOVE TXTPTR TO RIGHT PLACE
000126                JSR       CHRGET                ;GET 1ST CHAR IN IMAGE
000127                TAX
000128                CPX       #IMAGETK              ; IS IT IMAGE STMT?
000129                BNE       UIMERR                ;NO, COMPLAIN ABOUT IT
000130                JSR       CHRGET                ;GET NEXT CHR
000131                BEQ       UIMERR                ;NULL IM, COMPLAIN ALSO
000132                LDA       TXTPTR
000133                STA       SPCPTR
000134                LDA       TXTPTR+1
000135                STA       SPCPTR+1              ; SAVE PTR IMAGE
000136                LDA       TXTPTRB
000137                STA       SPCPTRB
000138                JSR       REMN                  ;COUNT TO EOL IN Y
000139                STY       SPCLEN                ;INTO MY DESCRIPTOR
000140                PLA
000141                STA       TXTPTRB
000142                PLA
000143                STA       TXTPTR+1
000144                PLA
000145                STA       TXTPTR
000146 UGOTSPT         JSR       CHRGOT                ;HAVE IMAGE DESCRIPTOR
000147                CMP       #';'                  ;PROPER SYNTAX??
000148                BNE       USNERR                ;NO, MISSING TERMINATOR
000149                LDX       #$FF
000150                STX       SPCNDX                ;CHECK 1ST CHR OF SPEC
000151                JSR       UGETCH                ; GET IMAGE FIRST CHR
000152                BEQ       USNERR                ;DUMMY HAS A , FIRST
```

```
000153                STX       SPCNDX
000154                INX
000155                STX       VSPEC                ;FLAG OFF
000156                PAGE
000157 USKPNXT        JSR       CHRGET
000158                BNE       *+5                  ;DOESN'T MATTER IF IT GOES.
000159 UNXTEXP        JSR       CHRGOT               ;TOP OF MAIN LOOP
000160                BNE       UNXTEXC
000161                LDA       #0
000162                BEQ       UTAILS
000163 UNXTEXC        CMP       #','                 ;ENDED ON COMMA?
000164                BEQ       USKPNXT              ;YES, IGNORE THEM
000165                CMP       #';'                 ;NO CR END?
000166                BNE       UEVAL                ;GO GET EXPR
000167 UTAILS         STA       ENDFLG               ;SAV HOW TO EXIT
000168                TAX
000169                BEQ       *+5                  ;SKIP IF NOT ;
000170                JSR       CHRGET               ; DONE NOW?
000171                BEQ       *+5
000172                JMP       USNERR
000173                LDA       #MSKLEN              ;GET MASK SPACE
000174                JSR       GETSPA               ;ANYWAY
000175                JSR       UDOLITS
000176                JSR       UFRESPC              ;FREE UP TEMP
000177                DEC       SPECTYP              ;NO SPECS?
000178                BEQ       *+5                  ;YES
000179                JMP       UTMERR               ;NO, SPEC WITHOUT EXPR ERR
000180                LDA       ENDFLG               ;ME CUTLASS OR THE PLANK??
000181                BNE       *+5                  ;THE CUTLASS EH
000182                JMP       CRDO                 ;THE PLANK ME BOY
000183                RTS
000184 UEVAL          LDA       SPCNDX
000185                PHA
000186                LDA       SPCLEN
000187                PHA
000188                LDA       SPCPTR
000189                PHA
000190                LDA       SPCPTRH
000191                PHA
000192                LDA       VSPEC
000193                PHA
000194                LDA       #0
000195                PHA
000196                JSR       CHRGOT               ;FORMULA OR SCALE FUNC?
000197                CMP       #SCALETK
000198                BNE       UNOSCALE
000199                JSR       CHRGET               ;EAT IT.
000200                JSR       GETABYT              ;1 BYTE SIGNED VALUE.
000201                PLA                            ;REPLACE SCALE FACTOR OF ZERO WITH NEW ONE.
000202                TXA
000203                PHA
000204                JSR       CHKCOM               ;MUST HAVE COMMA.
000205                LDA       #$20
000206                STA       VALTYP               ;COULD HAVE BEEN CLOBBERED BY GETBYT
000207                JSR       FRMEVL               ;EVALUATE THE BEAST.
000208                JSR       CHKCLS
000209                BEQ       UGOTNUM              ;MUST HAVE A LEAGAL SEPARATOR NEXT.
000210                CMP       #','
000211                BEQ       UGOTNUM
000212                CMP       #';'
000213                BEQ       UGOTNUM
000214                JMP       SNERR
000215 UNOSCALE       LDA       #$20
000216                STA       VALTYP               ;ANY TYPE ALLOWED
000217                JSR       FRMEVL               ;GO GET EXPR
000218 UGOTNUM        BIT       VALTYP               ;WHAT TYPE
000219                BPL       UNTYPE
000220                JSR       NOTFAC               ;MOVE STR PTR TO INDEX
000221                STA       USVSTRL              ;SAVE THE STR LENGTH
000222                JMP       USTKSAV
000223 UDTYPE         EQU       *
000224                JSR       LUNPACK              ;UNPACK FROM LONG INT.
000225                LDA       ISARA                ;GET ADJUSTED EXPONENT.
000226                STA       FACEXP               ;NOW LOOKS LIKE BCD.
000227                JMP       USTKSAV
000228 UNTYPE         BVS       UDTYPE               ;CVRT # TO STRING
000229                LDA       FACSGN               ;THIS CODE MUST CONVERT FAC TO A USABLE
000230                PHA                            ;FORM FOR PRINT USING (SEE UUNPACK).
000231                JSR       FOUT                 ;OUTPUT THE FLOATING POINT # INTO FBUFR.
000232                PLA
```

```
000233                  CMP     #$80                    ;WAS IT NEGATIVE?
000234                  PHP                             ;SIGGN IN CARRY.
000235                  LDA     ISARA                   ;FOUT SET THIS GUY UP TO BE ALMOST THE EXPONENT.
000236                  SEC
000237                  SBC     #1
000238                  PLP
000239                  ROL     A                       ;SIGN INTO LOW BIT.
000240                  STA     FACEXP
000241                  LDY     #2
000242                  LDX     #0
000243 STRBCD0          LDA     FBUFFR,X                ;GGET A DIGGIT.
000244                  CMP     #'.'                    ;SKIP ACROSS PERIODS.
000245                  BEQ     STRBCD1
000246                  CMP     #'-'                    ;SKIP OVER A MINUS.
000247                  BEQ     STRBCD1
000248                  CMP     #':'
000249                  BCS     STRBCD2
000250                  SBC     #'0'-1
000251                  BCC     STRBCD2
000252                  BNE     *+6                     ;CHECK FOR LEADING ZEROES.
000253                  CPY     #2                      ;IS THIS ALSO THE FIRST NUMBER?
000254                  BEQ     STRBCD1                 ;YES, SKIP OVER IT.
000255                  STA     BCDSTR,Y                ;NOW ITS A BCD DIGIT.
000256                  INY
000257 STRBCD1          INX
000258                  BNE     STRBCD0                 ;ALWAYS.
000259 STRBCD2          LDA     #0                      ;THAT MUST BE THE END OF IT.
000260 STRBCD3          STA     BCDSTR,Y                ;GOT TO FILL THE REST WITH 0'S.
000261                  INY
000262                  CPY     #22
000263                  BNE     STRBCD3
000264 USTKSAV          PLA
000265                  STA     ISARA                   ;SCALE FACTOR.
000266                  PLA
000267                  STA     VSPEC
000268                  PLA
000269                  STA     SPCPTRH
000270                  PLA
000271                  STA     SPCPTR
000272                  PLA
000273                  STA     SPCLEN
000274                  PLA
000275                  STA     SPCNDX
000276                  LDA     #MSKLEN                 ;NEED 32 BYTE BUFR
000277                  JSR     GETSPA                  ;FRESPC IS MASK PTR
000278 UNXTSPC          JSR     UDOLITS                 ;GET EXPR SPEC/NONE
000279                  LDX     SPECTYP                 ;WHAT RESULT?
000280                  BEQ     UVSPEC                  ;NUMERIC SPEC
000281                  BMI     UVSPEC                  ;STRING SPEC
000282                  LDA     VSPEC                   ;END OF SPEC LIST!
000283                  BEQ     UTMERR2                 ;NO SPECS TO REUSE!!!
000284                  STA     SPCNDX                  ;RESTART SPECS
000285                  BNE     UNXTSPC                 ;ALWAYS
000286 UVSPEC           LDA     #$FF
000287                  STA     VSPEC                   ;SET FLAG ON
000288                  SEC
000289                  LDA     VALTYP                  ;BACK FROM EVAL
000290                  TAX
000291                  SBC     SPECTYP                 ;SAME TYPES?
000292                  LSR     A                       ;LSB TELLS ME!
000293                  BCC     *+8                     ;YES THEY ARE
000294 UTMERR2          JSR     UFRESPC                 ;FREE MASK.
000295 UTMERR           JMP     TMERR
000296                  TXA                             ;VALTYP
000297                  BPL     *+5
000298                  JMP     USTRVAR                 ;GO DO STRINGS
000299                  JMP     UNUMVAR                 ;GO DO NUMBERS
000300                  PAGE
000301 UDOLITS          LDY     #MSKLEN-1               ;Y=MSKLEN=32
000302                  LDA     #0                      ;INIT TO ALL DIGITS!
000303 UBLKMSK          STA     (MASKPT),Y
000304                  DEY
000305                  BPL     UBLKMSK
000306                  JSR     IMSYNCK                 ; GO SYNTAX & REPACK THE NEXT SPEC
000307                  BIT     SPECTYP
000308                  BMI     ULITOUT                 ;MAYBE LITERAL
000309 UDOXIT           RTS                             ;RETURN SPEC OR NONE!
000310 ULITOUT          BVS     UDOXIT                  ;STRING SPEC
000311 ; LOWTR POINTS TO LITERAL
000312 ; REP IS NUMBER OF TIMES TO SEND
```

```
000313 *
000314 UREPEAT         LDY       #0
000315                 LDX       SAVLEN                ;GET LIT LENGTH
000316 UOUTLP          LDA       (LOWTR),Y
000317                 JSR       OUTDO
000318                 INY
000319                 DEX
000320                 BNE       UOUTLP                ;DO NEXT ONE
000321                 DEC       REP                   ;ANOTHER TIME?
000322                 BNE       UREPEAT               ; YEP
000323                 BEQ       UDOLITS               ;DONE, DO NEXT ONE!
000324                 PAGE
000325 ******************************
000326 USTRVAR         LDA       USVSTRL               ;GET STR LEN
000327                 STA       SAVLEN
000328                 LDA       REP                   ;IS FIELD LEN > STRING LENGTH
000329                 SEC
000330                 SBC       SAVLEN                ; ???
000331                 BEQ       UEXACT                ;JUST FITS SEND IT
000332                 BCS       USFITS                ;YES
000333                 LDX       REP                   ;GET MAX LENGTH
000334                 STX       SAVLEN                ; LIMIT TO FIELD SIZE
000335                 JMP       UEXACT
000336 USFITS          LDY       DELIM                 ;WHAT DOING?
000337                 CPY       #'A'
000338                 BEQ       UEXACT
000339                 CPY       #'A'+$20              ;LOWER CASE?
000340                 BEQ       UEXACT
000341                 CPY       #'C'                  ; CENTER IT?
000342                 BEQ       *+6
000343                 CPY       #'C'+$20              ;LOWER CASE?
000344                 BNE       UEATME
000345                 LDX       SAVLEN
000346                 BEQ       UEATME
000347                 LSR       A
000348 UEATME          TAX                             ;X=LENGTH OF LEADING SPACES
000349                 INX
000350 ULSPCS          DEX                             ; AM I DONE?
000351                 BEQ       UEXACT
000352                 JSR       OUTSPC
000353                 DEC       REP
000354                 BNE       ULSPCS
000355 UEXACT          LDY       #0                    ;STR START
000356                 LDX       SAVLEN                ;GET STRING LENGTH
000357                 BEQ       UNULL                 ;IF NULL STRING, DO SPEC
000358 UEXACT2         LDA       (INDEX),Y             ; GET STRING CHR
000359                 JSR       OUTDO                 ;PRINT IT
000360                 INY
000361                 DEC       REP
000362                 DEX                             ; AM I DONE?
000363                 BNE       UEXACT2               ;ON WITH THE SHOW
000364 UNULL           LDX       REP
000365 UFILL3          BEQ       UFRESTR
000366                 JSR       OUTSPC
000367                 DEX
000368                 JMP       UFILL3
000369 UNUMVAR         JSR       UNUMEDIT              ;DO NUMBERS
000370 UFREMSK         JSR       UFRESPC               ;FREE UP MASK AREA
000371                 JMP       UNXTEXP               ;DO NEXT EXPR
000372                 PAGE
000373 UFRESTR         EQU       *
000374                 JSR       FRECNOW               ;FREE IT IF IT WAS A TEMPORARY STRING
000375                 JMP       UFREMSK               ;NOW MASK TOO
000376 ****************************************
000377 *
000378 * BEGIN SUBROUTINES
000379 *
000380 **********************************
000381 UFRESPC         LDA       #MSKLEN               ;FIXED LENGTH
000382                 LDY       FRESPC+1
000383                 LDX       FRESPC                ;ADRS OF AREA TO FREE
000384                 STX       INDEX                 ;FOR FRESPA.
000385                 STY       INDEX+1
000386                 LDX       FRESPCB
000387                 STX       INDEXB
000388                 JMP       FRESPA
000389 ******************************
000390 UREGET          PLA                             ;CLEAR STK
000391 UGETCHR         JSR       UGETCH
000392                 PHP                             ;SAVE STATUS
```

```
000393                CMP       #'                      ' ;A SPACE?
000394                BEQ       UREGET                  ;YES, IGNORE
000395                PLP                               ;RECOVER STATUS
000396                RTS
000397 ********************************
000398 UGETCH          INC       SPCNDX
000399 IGOTCH          LDY       SPCNDX
000400                LDA       #0                      ;IF END OR OVRFLO
000401                CPY       #$FF                    ; PAST MAX?
000402                BNE       UGETCKL                 ;NOT OVRFLO
000403                DEC       SPCNDX                  ;FOR REUSE
000404                BNE       UGETNE
000405 UGETCKL         CPY       SPCLEN
000406                BCS       UGETNE
000407                LDA       (SPCPTR),Y
000408                CMP       #','                    ;END OF SPEC?
000409                BEQ       UGETEND                 ;YES
000410                CMP       #'A'                    ; IS IT < A ?
000411                BCC       UGETCKD                 ;YES
000412                CMP       #'Z'+1                  ; IS IT A LETTER
000413                BCC       UGETVS                  ;YES SET V ON
000414 UGETCKD         CMP       #':'                    ;NO, IS IT A DIGIT ?
000415                BCS       UGETNE                  ;NO SPECIAL
000416                SEC
000417                SBC       #'0'                    ; TAKE OUT ZERO
000418                SEC
000419                SBC       #$D0                    ;AND IT COMPLEMENT
000420 UGETNE          TAY                               ; EQU ON ZERO
000421 UGETEND         CLV
000422                RTS
000423 UGETVS          BIT       *+4                     ; SET V BIT ON
000424                CMP       #$40                    ;CS,VS,NE
000425                RTS
000426                PAGE
000427 UATOMSK         LDY       MSKNDX
000428                INY
000429                CPY       #MSKLEN                 ;TOO MUCH?
000430                BCS       USYERR                  ;YEP
000431                STA       (MASKPT),Y              ;ADD IT
000432                STY       MSKNDX
000433                RTS
000434 USYERR          JMP       USNERR
000435 ********************************
000436 UDOMASK         PHA                               ;SAVE A
000437                LDY       DIGCTR                  ; ANY TO DO
000438                BEQ       UDONT                   ;EXIT
000439                LDX       #1                      ;ASSUME AFTER
000440                LDA       #4                      ;DPT MASK BIT
000441                BIT       EMASK                   ;DPT YET?
000442                PHP                               ;SAVE Z BIT
000443                BNE       UAFDPT                  ;YES, NO CHECK
000444                LDA       NMASK
000445                CMP       #ZTYP                   ;Z SPEC?
000446                BNE       UNOTZ                   ;NO
000447                LDA       FLTMSK                  ;FLOAT AND Z IS SYNTAX
000448                BNE       USYERR                  ;BAD SPEC!
000449 UNOTZ           DEX                               ;X=0 FOR B4 DPT
000450 UAFDPT          TYA                               ;GET FIELD SIZE
000451                STA       DIGB4DPT,X              ;SAVE FOR EDITOR
000452                LDX       NMASK
000453                BEQ       USYERR                  ;NO DIGITS B4 DPT AND FLOAT SPEC!
000454                CPX       #CMATYP                 ;IS THIS #
000455                BNE       UANYSIZ                 ;YES ANY SIZE
000456                CMP       #5                      ;MIN SIZE IS #,###
000457                BCC       USYERR                  ;FIELD TOO SMALL FOR COMMA INSERT
000458 UANYSIZ         LDA       #0                      ;DIGITS = 0
000459 UDOMSKL         JSR       UATOMSK                 ;STICK IT IN
000460                DEC       DIGCTR
000461                BNE       UDOMSKL
000462                PLP                               ;GET Z BIT
000463                BNE       UDONT
000464                STY       TENZNDX
000465 UDONT           LDY       #2                      ;RESTORE Y
000466                LDA       #0
000467                STA       NMASK                   ;NEW SUB FIELD
000468                PLA
000469                RTS
000470                PAGE
000471                REP       60
000472 *
```

```
000473 *      Spec Syntax Checker
000474 *
000475 * Validates the Syntax of all Specs, Builds Numeric
000476 *   Edit Mask and Pointers to Literal Specs
000477 *
000478                 REP       60
000479 IMSYNCK         EQU       *
000480                 LDY       #1
000481                 STY       SPECTYP                   ; NONE IS DEFAULT
000482                 DEY
000483                 STY       CMAFILL                   ;NO FILLER
000484                 STY       DIGB4DPT
000485                 STY       DIGAFDPT
000486                 STY       FLTMSK                    ;NO FLOAT
000487                 STY       DIGCTR
000488                 STY       EMASK                     ; NO EDITING
000489                 STY       NMASK                     ; NO DIGITS
000490                 DEY                                 ;Y=FF
000491                 STY       DPTNDX                    ;NO DPT
000492                 STY       TENZNDX                   ; NO 10**0 DIGIT
000493                 STY       MSKNDX                    ;INIT AT FF
000494                 JSR       UGETDL                    ; GET NEXT REP & DELIM
000495                 BPL       UNUMTYP                   ;GO DO NUM TYPES
000496                 STA       SPECTYP                   ;SET TYPE
000497                 BIT       SPECTYP                   ;LIT/STR ?
000498                 BVS       UCHEKCM                   ;ALL DONE IF STRING!
000499                 LDA       DELIM
000500                 CMP       #'/'                      ; CR OUT?
000501                 BEQ       UCRLIT
000502                 CMP       #'X'                      ; SPACE OUT?
000503                 BEQ       USPLIT
000504                 CMP       #'X'+$20
000505                 BEQ       USPLIT
000506                 LDA       SPCNDX
000507                 SEC                                 ;PLUS 1
000508                 ADC       SPCPTR
000509                 LDY       SPCPTR+1
000510                 LDX       SPCPTRB
000511                 BCC       *+6
000512                 INY
000513                 JSR       FIXYX
000514                 STA       LOWTR
000515                 STY       LOWTR+1
000516                 STX       LOWTRB
000517                 LDX       #0-1
000518 UQSCAN          INX                                 ;NEXT CHR
000519                 JSR       UGETCH                    ;MOVE PTR ALONG!
000520                 CMP       #0                        ; END OF SPEC?
000521                 BEQ       USNERRL                   ;NOTAIL"
000522                 CMP       #'"'                      ; END YET?
000523                 BNE       UQSCAN
000524                 STX       SAVLEN                    ;RETURN LENGTH OF LITERAL
000525                 JSR       UGETCHR                   ; NEXT AFTR "
000526 UCHEKCM         JSR       IGOTCH                    ;DONE AFTER SPEC?
000527                 BNE       *+3
000528                 RTS
000529 USNERRL         JMP       USNERR
000530 UCRLIT          LDA       #$0A                      ;A LF.
000531                 STA       LITRLCH+1
000532                 LDY       #2
000533                 LDA       #$0D                      ;A CR!!
000534                 BNE       *+6
000535 USPLIT          LDA       #'                        ' ;A SPACE
000536                 LDY       #1
000537                 STA       LITRLCH
000538                 LDX       #<LITRLCH
000539                 LDA       #LITRLCH
000540                 STA       LOWTR
000541                 STX       LOWTR+1
000542                 STY       LOWTRB
000543                 STY       SAVLEN
000544                 BNE       UCHEKCM                   ; DONE NOW?
000545                 PAGE
000546 UNUMTYP         EQU       *
000547                 LDY       #2                        ;A TWO
000548                 CMP       #$02                      ; IS IT * ?
000549                 BNE       UDOLLR
000550                 CPY       REP                       ;** OR 2*?
000551                 BNE       UNSERR                    ;BAD STUFF
000552                 LDA       #$10                      ; ** BIT ON
```

```
000553                STA     FLTMSK                  ;SAY FLOATING
000554                LDA     DELIM                   ; GET AN *
000555                STA     CMAFILL                 ;* FILLER
000556                JSR     UENDERR                 ;EXIT IF END ELSE GETDL
000557                JMP     UDOLLR
000558 UDOLAR          CMP     #$10                    ;$ OR $$ ?
000559                BEQ     *+3
000560                RTS
000561                BIT     EMASK
000562                BNE     UNSERR
000563                ORA     EMASK
000564                STA     EMASK                   ;SET $ BIT ON
000565                CPY     REP                     ;REP =2
000566                BNE     UCK1DL
000567                LDA     FLTMSK
000568                BNE     UNSERR                  ;MA SN BUDDY
000569                BEQ     UDLFLT                  ;FLOAT NOW
000570 UCK1DL          DEY
000571                CPY     REP
000572                BNE     UNSERR                  ;ERROR
000573                LDA     FLTMSK                  ;** PREV?
000574                BEQ     USTDLR                  ;NO
000575 UDLFLT          CMP     #$40                    ;PREV ++/-- ?
000576                LDA     #$20
000577                BCC     *+4                     ;NOT $ AFTER SIGN
000578                ASL     A
000579                ASL     A                       ; CREATE $80
000580                ORA     FLTMSK
000581                STA     FLTMSK
000582                INC     MSKNDX                  ;RESERVE SPC
000583                JMP     UXDLR
000584 USTDLR          LDA     DELIM                   ; $ TO MASK,MASKNDX
000585                JSR     UATOMSK
000586 UXDLR           JMP     UENDERR                 ;NEXT DELIM
000587 UDOLLR          JSR     UDOLAR                  ;$
000588                BIT     UDOLLR                  ;EITHER SIGN?
000589                BEQ     UTRYDL2                 ;NO
000590                ORA     EMASK
000591                STA     EMASK                   ; SET SIGN BITS
000592                CPY     REP                     ;TWO SIGNS
000593                BNE     UCK1SIN                 ;NO
000594                LDA     FLTMSK
000595                BNE     UNSERR                  ;MA BUDDY AGAIN
000596                BEQ     UFLTSIN
000597 UCK1SIN         DEY                             ;Y=1
000598                CPY     REP                     ;1 SIGN ?
000599                BNE     UNSERR
000600                LDA     FLTMSK
000601                BEQ     USETSIN
000602 UFLTSIN         LDA     #$40
000603                ORA     FLTMSK
000604                STA     FLTMSK
000605                INC     MSKNDX                  ;RESERVE SPC
000606                JMP     UXSIN                   ;ALL DONE
000607 UNSERR          JMP     USNERR                  ;SYNTAX IT
000608 USETSIN         LDA     DELIM                   ; GET SIGN
000609                JSR     UATOMSK                 ; PUT +/- INTO MASK
000610 UXSIN           JSR     UENDERR                 ; ERR IF ENDED!
000611 UTRYDL2         JSR     UDOLAR                  ;TRY $ AGAIN
000612 UTRYDIG         CMP     #ZTYP                   ;DIG R 41,42 44
000613                BCC     UBLDMK                  ;NOT DIGIT !
000614                TAX                             ;TYPE TO X
000615                BMI     UNSERR                  ;NO MIX ADEE APPLS N' ORANGES
000616                LDA     #$80
000617                ORA     EMASK                   ;REMEMBER THEM
000618                STA     EMASK                   ;FOR SYNTAX CK
000619                LDA     REP                     ;GET REP
000620                CLC
000621                ADC     DIGCTR                  ;ADD IN DIGITS FROM
000622                STA     DIGCTR                  ;DIGIT LISTS
000623                CPX     NMASK                   ;LOWER CLASS DIGIT?
000624                BCS     *+4                     ;NO, EQ OR GT
000625                LDX     NMASK                   ;GET HIGHER CLASS
000626                LDA     #$04                    ;TEST BIT FOR DPT
000627                BIT     EMASK                   ;DPT BEFORE?
000628                BEQ     UB4DPT                  ;NO SET FILL
000629                LDX     #ZTYP                   ;Z ONLY AFTER DPT
000630                BNE     UXNMASK                 ; AND SET CLASS
000631 UB4DPT          LDA     CMAFILL                 ;FILLING WITH ASTERISKS?
000632                AND     #$7F
```

```
000633                  CMP       #'*'
000634                  BEQ       USETCMA
000635                  LDA       FILLTABL-ZTYP,X
000636 USETCMA           ORA       CMATBL-ZTYP,X           ; ADD FILLER
000637                  STA       CMAFILL                 ;SET THEM ALL
000638 UXNMASK           STX       NMASK                   ;SET DIGIT CLASS
000639 UXMORE            JSR       UCKDEND                 ;ANY MORE ?
000640                  JMP       UTRYDIG                 ;YES IF HERE
000641 UBLDMK            JSR       UDOMASK                 ;PUT DIGITS IN MASK
000642                  CMP       #$04                    ;THE DPT? (.)
000643                  BNE       USGNCK                  ;NO TAIL SIGN?
000644                  BIT       EMASK                   ;DPT BEFORE?
000645                  BNE       UNSERR                  ;YES, TWO IS NO NO
000646                  DEY                               ;Y=1!
000647                  CPY       REP
000648                  BNE       UNSERR                  ;N. IS ALSO
000649                  ORA       EMASK
000650                  STA       EMASK                   ;SET DPT FOUND
000651                  LDA       DELIM
000652                  JSR       UATOMSK
000653                  LDA       MSKNDX
000654                  STA       DPTNDX
000655                  JMP       UXMORE                  ;YES IF RETURNED
000656 USGNCK            BIT       UDOLLR                  ;SIGN AGAIN?
000657                  BEQ       UCKEXP                  ;NOT SIGN
000658                  DEY                               ;Y=1
000659                  CPY       REP
000660                  BNE       USTERR                  ;SYNTAX
000661                  BIT       EMASK                   ;SIGN BEFORE?
000662                  BNE       USTERR                  ;TWO IS NO NO
000663                  ORA       EMASK
000664                  STA       EMASK
000665                  LDA       DELIM                   ;GET SIGN
000666                  JSR       UATOMSK
000667                  JSR       UCKDEND
000668 UCKEXP            CMP       #$08                    ;EEEE ?
000669                  BNE       USTERR                  ;NAUGHTY NAUGHTY!
000670                  INY                               ;Y=3
000671                  CPY       REP
000672                  BEQ       UEEE
000673                  INY
000674                  CPY       REP
000675                  BNE       USTERR                  ;OTHERS ARE SYNTAX
000676 UEEE              ORA       EMASK
000677                  STA       EMASK
000678                  LDA       DELIM
000679 UEEELP            JSR       UATOMSK
000680                  DEC       REP                     ;COUNT DOWN
000681                  BNE       UEEELP                  ;DO IT Y TIMES
000682                  LDA       #$F0                    ;NO FLOATS
000683                  LDX       DIGB4DPT
000684                  BEQ       UENOFLT
000685                  DEX
000686                  BEQ       UENOFLT
000687                  CPX       #2                      ;SIZE =3?
000688                  BNE       USTERR                  ;ALL OTHER BAD
000689                  LDA       #$B0                    ;ALLOW FLOAT SIGNS
000690 UENOFLT           AND       FLTMSK
000691                  BNE       USTERR                  ;BAD COMBO!!
000692                  JSR       UCHEKCM                 ;ENDED NOW?
000693                  JSR       UDIGEND                 ;YES, ANY DIGITS?
000694 ; DIGEND DOESN'T RETURN
000695 UENDERR           JSR       IGOTCH                  ;GET LAST CHR
000696                  BEQ       USTERR
000697                  BNE       UNXTGET                 ;DO NEXT ONE
000698 UCKDEND           JSR       IGOTCH                  ;GET LAST
000699                  BEQ       UDIGEND                 ;ENDED SO CHECK
000700 UNXTGET           JSR       UGETDL
000701                  LDY       #2                      ;EVERYBODY LIKES 2
000702                  RTS
000703 UDIGEND           LDA       EMASK                   ; ANY DIGITS?
000704                  BPL       USTERR                  ;NO DUMMY
000705                  JSR       UDOMASK                 ; TAKE CARE OF TRAIL DIGITS
000706                  LDA       FLTMSK                  ;ANY FLOAT SPECS?
000707                  BEQ       UCKXIT                  ;NO
000708                  LDA       DIGB4DPT                ;ANY DIGIT WITH EM?
000709                  BEQ       USTERR                  ;TURKEY USER
000710 UCKXIT            LDA       #0
000711                  STA       SPECTYP                 ;SAY NUMERIC FIELD
000712                  PLA
```

```
000713                PLA
000714                RTS                              ; TO MAIN LINE
000715 USTERR         JSR       UFRESPC                ;FREE THE SPACE.
000716                JMP       USNERR
000717                PAGE
000718
000719 ; ###############################################################################################
000720 ; #   END OF FILE:  B3PRU1.TEXT
000721 ; #   LINES      :  712
000722 ; #   CHARACTERS :  34842
000723 ; ###############################################################################################
```

```
+-------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 723   CHARACTERS: 35392
|
+-------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "B3PRU2.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:31 PM
|  Modified: Wednesday, December 31, 1997             4:37:08 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  B3PRU2.TEXT
000004 ; ###############################################################################################
000005
000006                  REP       70
000007 *
000008 * GETDL fetches the next monochromatic delimiter sequence
000009 *   from the spec string via the subroutine UGETCH.
000010 *
000011 * It converts the Repeat Factor (REPFAC) into Binary in REP,
000012 *   and compresses consecutive occurances of the same delimiter,
000013 *   incrementing REP to compensate.
000014                  REP       70
000015 * INPUTS DATA ITEMS
000016 *
000017 * SPCNDX    Points to last used char in IMAGE string
000018 *
000019 * SPCPTR    Points to String Base
000020 * SPCLEN    String length (1-255)
000021 *
000022 * OUTPUT DATA ITEMS
000023 *
000024 * DELIM     The actual valid delimiter
000025 *
000026 * REP       The number of them
000027 *
000028                  REP       70
000029 *
000030 * ERROR EXITS
000031 *
000032 * If an Invalid Delinator is found, a SYNTAX Error will occur.
000033 *
000034 * If a repeat factor is >255 or the sum of repeat and
000035 *   consecutive delinators is >255 then an ILLEGAL QUANTITY
000036 *   Error and SYNTAX Error respectively, will occur.
000037 *
000038 * If SPCNDX is at the end of IMAGE when GETDL is called, it
000039 *   will return up 1 level higher via the stack clear
000040 *   of the caller's Return Address.
000041                  REP       70
000042 UGETDL          EQU       *
000043                  LDY       #0
000044                  STY       DELIM
000045                  STY       REP
000046 UCOMA           JSR       UGETCHR                 ; GET A CHAR
000047                  BNE       UGOT1
000048                  CMP       #','                    ; IS IT A COMMA?
000049                  BEQ       UCOMA                   ;YES, IGNORE
000050                  PLA                               ; NO MUST BE EOI
000051                  PLA
000052                  RTS
000053 UGOT1           BCS       UDLCHR                  ;DO DELIM
000054                  LDX       DELIM                   ;GOT A DELIM YET?
000055                  BEQ       UG1STD                  ;NOTHING SO FAR
000056                  CPX       #'9'                    ;ONLY DIGITS SO FAR ?
000057                  BNE       UGVAL                   ;NO, GO VALIDATE
000058 UG1STD          LDY       REP                     ;REP*10>250?
000059                  CPY       #26
000060                  BCC       *+8                     ;NO CHANCE
000061 UIQERR          JSR       UFRESPC
000062                  JMP       FCERR                   ;DUMB USERS!!
000063                  LDX       #'9'
000064                  STX       DELIM                   ;I GOT A DIGIT BEFORE
000065                  TYA
000066                  ASL       A
000067                  ASL       A
000068                  ASL       A                       ;A=REP*8
000069                  ADC       REP
000070                  ADC       REP                     ;A=REP*10
000071                  STA       REP
000072                  LDY       SPCNDX
```

```
000073                  LDA       (SPCPTR),Y              ;ADD THE DIGIT
000074                  AND       #$0F                    ; ZAP ASCII
000075                  ADC       REP                     ;REP*10+DIGIT
000076                  BCS       UIQERR                  ;DIG >5 FOR REP=25
000077                  STA       REP                     ;REP=REP*10+DIGIT
000078 UGNXTC           JSR       UGETCHR                 ;INC & LDA
000079                  BEQ       UGEND                   ;END OF SPEC!
000080                  BNE       UGOT1                   ;PROCESS
000081 UGVAL            DEC       SPCNDX                  ;PICK SAME NXT TIME
000082 UGEND            LDA       DELIM
000083                  CMP       #'Z'+1
000084                  BCC       *+4
000085                  SBC       #$20
000086                  LDY       #DLMCNT-1
000087 UVALOOP          CMP       DLMTBL,Y
000088                  BEQ       UGETND
000089                  DEY
000090                  BPL       UVALOOP                 ; TRY NEXT
000091 UGERRX           JSR       UFRESPC
000092                  JMP       USNERR                  ;BAD DELIM
000093 UGETND           LDA       STYPTBL,Y
000094                  RTS
000095 UDLCHR           LDY       DELIM
000096                  BEQ       U1STDL
000097                  CPY       #'9'                    ;REP FAC ONLY?
000098                  BNE       UDLCK
000099                  LDY       REP                     ;ALL 0 REP?
000100                  BEQ       UIQERR                  ;THE GUY IS NUTS!
000101                  BNE       USETDL
000102 U1STDL           LDY       #1
000103                  STY       REP
000104 USETDL           STA       DELIM                   ; SET DELIM
000105                  CMP       #'"'                    ;LITERAL START?
000106                  BEQ       UGEND                   ;YES, STOP NOW
000107                  BNE       UGNXTC                  ;DO NEXT
000108 UDLCK            CMP       DELIM                   ; SAME DELIM AS LAST?
000109                  BNE       UGVAL                   ;NO, DONE
000110                  CMP       #'"'                    ;" AFTER DELIM="?
000111                  BEQ       UGERRX
000112                  INC       REP
000113                  BNE       UGNXTC                  ;DO NEXT
000114                  BEQ       UGERRX
000115 DLMTBL           EQU       *
000116                  ASC       'AX"+-&$.ECR/#Z*'
000117 STYPTBL          EQU       *
000118 DLMCNT           EQU       STYPTBL-DLMTBL
000119                  DFB       $FF                     ;STRG
000120                  DFB       $80                     ;LIT
000121                  DFB       $80                     ;LIT
000122                  DFB       $21                     ;EDIT
000123                  DFB       $20                     ;EDIT
000124                  DFB       CMATYP                  ;$44
000125                  DFB       $10                     ;EDIT
000126                  DFB       $04                     ;EDIT
000127                  DFB       $08                     ;EDIT
000128                  DFB       $FF                     ;STRG
000129                  DFB       $FF                     ;STRG
000130                  DFB       $80                     ;LIT
000131                  DFB       DTYP                    ;$42
000132                  DFB       ZTYP                    ;$41
000133                  DFB       $02                     ;EDIT
000134 *        FORMAT OF TYPE BYTES
000135 *     BIT 7 6 5 4 3 2 1 0
000136 *         T T S $ E . * +
000137 *
000138 *         WHERE S=SIGN AND +=WHICH SIGN
000139 *         TT=0 1  FOR DIGIT TYPE
000140 *           =0 0  FOR EDITING TYPE
000141 *           =1 0  FOR LITERAL TYPE
000142 *           =1 1  FOR STRING TYPE
000143 *
000144 FILLTABL         ASC       '0   '
000145 CMATBL           DFB       0,0,0,$80
000146                  PAGE
000147 *********************************
000148 * UNPACK BCD MANTISSA
000149 *********************************
000150 * ALL 6502 REGS DESTORYED
000151 *********************************
000152 * OUTPUTS
```

```
000153 * A IS ZERO
000154 * P HAS ZERO FLAG SET
000155 * Y 18  OR $12
000156 * X 10  OR $0A
000157 * OUTPUT FORM IS ALWAYS
000158 * 1122334455667788990
000159 *   @ =$00
000160 ********************************
000161 UUNPACK         EQU       *
000162                 LDY       #2
000163                 LDX       #$FF
000164                 STX       HALF                ; HIGH HALF FIRST
000165                 INX                           ;X=0
000166 UHLOOP          LDA       FACT,X              ; GET HALF
000167                 BIT       HALF                ; WHICH HALF?
000168                 BPL       UPUSELOW            ; LSH
000169                 INC       HALF                ;LOW HALF NEXT
000170                 LSR       A
000171                 LSR       A
000172                 LSR       A
000173                 LSR       A
000174                 BPL       UPUSEA
000175 UPUSELOW        DEC       HALF                ;UPR HALF NEXT
000176                 INX                           ; NEW BYTE TOO
000177                 AND       #$F                 ; MASK TOP
000178 UPUSEA          EQU       *                   ;IN HALF BCD. (NOT ASCII)
000179                 CPY       #3
000180                 BCS       UGTABYT             ;MIDDLE OF #, DON'T SKIP ZEROES.
000181                 CMP       #0                  ;LEADING ZERO?
000182                 BNE       UGTABYT             ;NO, SKIP.
000183                 DEC       ISARA               ;VIRTUAL EXPONENT FOR LONG INTEGER #'S.
000184                 DEC       ISARA
000185                 BNE       UHLOOP              ;ALWAYS.
000186 UGTABYT         STA       BCDSTR,Y            ; PUT IT
000187                 INY                           ; NEXT CHAR
000188                 CPY       #22
000189                 BNE       UHLOOP              ; NO SO LOOP
000190                 LDA       #0
000191                 STA       BCDSTR,Y            ;TRAILING ZERO
000192                 RTS
000193                 PAGE
000194 UNUMEDIT        INC       MSKNDX              ;= LENGTH
000195                 LDA       FACEXP              ;THIS SEPERATES BCD EXP
000196                 CMP       #$80                ;EXTEND SIGN TO C
000197                 ROR       A
000198                 ROR       A                   ;MANT SGN TO BIT 7
000199                 STA       FACSGN              ;SAVE IT
000200                 ROL       A                   ;EXP NOW TRUE SIGNED #
000201                 SEC
000202                 SBC       #1                  ;ADJUST FOR BCD FORM
000203                 CLC
000204                 ADC       ISARA               ;ADD SCALE FACTOR.
000205                 STA       FACEXP              ;-63 -> +64
000206                 CMP       #MINSGN             ;MINEXP
000207                 BCS       UROKNOW
000208                 CMP       #MAXSGN+1
000209                 BCC       UROKNOW
000210                 JSR       UFRESPC
000211                 JMP       FCERR
000212 MINSGN          EQU       $9D
000213 MAXSGN          EQU       $63
000214 UROKNOW         JSR       USGNSCN             ; DO FIXED SIGN
000215                 LDY       #0
000216                 STY       CMACTR
000217                 STY       DIGCTR              ;GET NEXT INIT
000218                 LDA       #$08                ;EEEE BIT
000219                 BIT       EMASK               ;INTEGER OR EE FORMAT?
000220                 BEQ       *+5                 ;INTEGER
000221                 JMP       UEEEDIT
000222                 BIT       CMAFILL             ;CMA INSERTION?
000223                 BPL       UNOCMAS             ;NO EXP OK
000224                 LDA       DIGB4DPT            ;AT LEAST 5 THERE
000225                 AND       #3                  ;IS IT DIVISIBLE BY 4?
000226                 CMP       #1                  ;C=0 IF TRUE
000227                 LDA       DIGB4DPT
000228                 TAX
000229                 BCC       UNOSBC1
000230                 SBC       #1
000231 UNOSBC1         EQU       *
000232                 LSR       A                   ;CALC MAC DIGITS
```

```
000233                LSR       A                       ;WHEN COMMAS USED
000234                STA       DIGB4DPT
000235                TXA                               ;RECOVER ORIGINAL
000236                SEC
000237                SBC       DIGB4DPT                ;COMPENSATE
000238                STA       DIGB4DPT                ;FOR COMMAS
000239 UNOCMAS         EQU       *
000240                LDA       #0-1
000241                SEC
000242                SBC       DIGAFDPT                ;A=EXP OF ROUND DIGIT
000243                TAX
000244                JSR       UROUNDI                 ;INTEGER ROUND
000245                JSR       UEXPLS1                 ;ABS(FACEXP)->TENEXP
000246                LDA       TENEXP
000247                BIT       FACEXP                  ;POSITIVE ?
000248                BMI       UDOLEFT
000249                CMP       DIGB4DPT                ;WILL DIGS LEFT FIT?
000250                BCC       UDOLEFT
000251 UTOOBIG         LDA       #'!'                    ;LOSS OF SIGNIF CHAR
000252                LDY       MSKNDX                  ;GET LENGTH
000253 UEXLOOP         DEY
000254                BMI       UEXDONE
000255                STA       (MASKPT),Y
000256                BNE       UEXLOOP                 ;FILL OUTPUT
000257 UEXDONE         JMP       USENDIT                 ;SHOW IT
000258 UDOLEFT         LDA       DIGB4DPT                ;ANY TO DO?
000259                BNE       *+5                     ;YES
000260                JMP       URIGHTS
000261                LDX       #$FF
000262                LDY       TENZNDX
000263 ULEFTLP         INX                               ;
000264                CPX       DIGB4DPT                ;DONE THEM ALL?
000265                BCS       UDOFLTS                 ;YES
000266                JSR       UGET10X                 ;X =DECIMAL PLACE REQUIRED
000267                BPL       UDOADIG                 ;GOT A DIGIT
000268                TXA                               ;NO DIG. 10**0 PLACE??
000269                BNE       UDOISGN                 ;NO SO EXIT
000270                LDA       #'0'                    ;USE A ZERO!!!!
000271 UDOADIG         BIT       CMAFILL                 ;DOING INSERTION?
000272                BPL       USTALFT                 ;NO
000273                PHA
000274                LDA       CMACTR
000275                CMP       #3                      ;TIME TO INSERT?
000276                BCC       UINCMA
000277                LDA       #','
000278                STA       (MASKPT),Y
000279                DEY
000280                LDA       #0                      ;RESTART CTR
000281                STA       CMACTR
000282 UINCMA          INC       CMACTR
000283                PLA                               ;GET CHR BACK
000284 USTALFT         STA       (MASKPT),Y
000285                DEY
000286                BPL       ULEFTLP                 ;DO NEXT DIGIT
000287 UDOISGN         LDA       #$20                    ;SIGN MASK
000288                BIT       EMASK                   ;DID HE REQUEST A SIGN
000289                BNE       UDOFLTS                 ;YES, SO NOT IMPLIED
000290                BIT       FACSGN                  ;IS IT NEGATIVE ?
000291                BPL       UDOFLTS                 ;NO, SO SKIP IT
000292                TYA                               ;ANY ROOM?
000293                BMI       UTOOBIG                 ;NO SO ERR
000294                LDA       (MASKPT),Y              ;IS A DIGIT AVAILABLE
000295                BNE       UTOOBIG                 ;NOT THAT EITHER
000296                LDA       #'-'                    ;ALL IS OK
000297                STA       (MASKPT),Y              ;PUT IN UNUSED DIGIT
000298                DEY
000299 UDOFLTS         LDA       FLTMSK                  ;ANY TO DO?
000300                AND       #$E0                    ;IGNORE UNUSED BITS
000301                BEQ       UDOFILL                 ;NO SO FILL REST
000302                ASL       A                       ;$ B4 SIGN?
000303                BCC       UNODL1                  ;NOPE
000304                TAX
000305                LDA       #'$'
000306                STA       (MASKPT),Y
000307                DEY
000308                TXA
000309 UNODL1          ASL       A                       ;SIGN?
000310                BCC       UNOSIGN                 ;NO
000311                TAX
000312                LDA       EMASK
```

```
000313                LSR       A                       ;+/- TO CARRY
000314                LDA       #'-'
000315                BIT       FACSGN
000316                BMI       USTORMI
000317                BCC       USKPSI
000318                LDA       #'+'
000319 USTORMI         STA       (MASKPT),Y              ;PUT IT HERE
000320                DEY                               ;LEFT 1 MORE
000321 USKPSI          TXA
000322 UNOSIGN         ASL       A                       ;DO $ AFTER SIGN?
000323                BCC       UDOFILL                 ;NO, FLOATERS DONE
000324                LDA       #'$'
000325                STA       (MASKPT),Y
000326                DEY
000327 UDOFILL         TYA                               ;ANY TO FILL
000328                BMI       URIGHTS                 ;NO ROOM TO FILL
000329                LDA       CMAFILL                 ;GET FILL CHR
000330                AND       #$7F                    ;ZAP CMA BIT
000331                TAX                               ;FILLER IN X
000332 UFILLP          LDA       (MASKPT),Y              ;A DIGIT POSITION
000333                BNE       URIGHTS                 ;NO SO DONE
000334                TXA                               ;GET FILLER
000335                STA       (MASKPT),Y
000336                DEY
000337                BPL       UFILLP
000338 URIGHTS         LDA       DIGAFDPT                ;ANY TO DO?
000339                BEQ       UDOEXP                  ;NO TRY EEEE
000340                LDY       DPTNDX                  ;GET DPT INDEX
000341                INY                               ;POINT TO NEXT
000342                LDX       #0
000343                TXA
000344                SEC
000345                SBC       DIGAFDPT                ;0-DIGAFDPT
000346                STA       DIGAFDPT                ;SAVE AS LIMIT
000347 URITLOP         DEX
000348                CPX       DIGAFDPT
000349                BCC       UDOEXP                  ;ALL DONE
000350                LDA       #8
000351                BIT       EMASK                   ;EXP OR INTEGER
000352                BNE       UEXPTYP
000353                JSR       UGET10X                 ;GET 10**X DIGIT
000354                BPL       UDIGRIT                 ;A DIGIT
000355                LDA       #'0'                    ;USE A ZERO
000356                BPL       UDIGRIT
000357 UEXPTYP         JSR       UGETNSD                 ;GET NEXT SIGNIFICANT DIGIT
000358 UDIGRIT         STA       (MASKPT),Y
000359                INY
000360                JMP       URITLOP                 ;LOOP
000361 UDOEXP          LDA       #8
000362                BIT       EMASK                   ;EEE TO DO?
000363                BEQ       USENDIT                 ;NO
000364                LDY       #0                      ;SCAN WHOLE MASK
000365 UDOEXP1         LDA       (MASKPT),Y              ;IS THIS AN E
000366                AND       #$DF
000367                INY
000368                CMP       #'E'                    ; ???
000369                BNE       UDOEXP1
000370                DEY
000371                STA       (MASKPT),Y
000372                INY
000373                LDA       #'+'
000374                BIT       FACEXP
000375                BPL       *+4
000376                LDA       #'-'
000377                STA       (MASKPT),Y
000378                INY
000379                LDA       TENEXP                  ;GET VALUE
000380                LDX       #0
000381                SEC
000382 USBCLP          SBC       #10
000383                BCC       UGOTHI
000384                INX                               ;HI DIG=HIDIG*10
000385                BNE       USBCLP
000386 UGOTHI          ADC       #10                     ;GET LO DIG BACK
000387                INY                               ;BUMP TO 2ND EXP
000388                CPY       MSKNDX                  ; PAST END?
000389                DEY                               ;BACK TO 1ST
000390                BCC       UDO1STD
000391                CPX       #0
000392                BEQ       UDO2NDD
```

```
000393                  JMP       UTOOBIG                 ;EXP WON'T FIT 3E
000394 UDO1STD          PHA
000395                  TXA
000396                  ORA       #$30
000397                  STA       (MASKPT),Y
000398                  INY
000399                  PLA                               ;GET LO BACK
000400 UDO2NDD          ORA       #$30                    ;MAKE ASCII
000401                  STA       (MASKPT),Y
000402 USENDIT          LDA       #0
000403                  STA       VALTYP
000404                  LDA       MASKPT
000405                  STA       INDEX
000406                  LDA       MASKPT+1
000407                  STA       INDEX+1
000408                  LDA       FRESPCB                 ;THIS IS THE SAME AS MASKPTB
000409                  STA       INDEXB
000410                  LDX       MSKNDX
000411                  JMP       STRPR3
000412                  PAGE
000413 UEEEDIT          LDA       DIGB4DPT
000414                  CLC
000415                  ADC       DIGAFDPT
000416                  TAX                               ;X=ROUND DIGIT
000417                  JSR       UROUNDA                 ;ROUND TO TOTAL PLACES
000418                  JSR       UEXPLS1                 ;ABS(EXP+(C=1)*1) -> TENEXP
000419                  LDX       DIGB4DPT                ;WHAT FORMAT?
000420                  BEQ       UDOSCI0                 ;FMT=(+).D(DDD)(-)EEE(E)
000421                  DEX
000422                  BNE       UENGNOT                 ;DIGB4DPT=3
000423                  LDY       #0
000424                  LDA       (MASKPT),Y              ;GET 1ST CHAR
000425                  BEQ       UDOSCI1
000426                  INY                               ;MUST BE NEXT !!
000427 UDOSCI1          LDA       DIGB4DPT
000428                  BEQ       UDOSCI2
000429                  JSR       UGETNSD
000430                  STA       (MASKPT),Y
000431                  DEC       DIGB4DPT                ;COUNT DOWN
000432                  INY
000433                  BNE       UDOSCI1
000434 UDOSCI0          SEC                               ;FORCE EXP+1
000435                  JSR       UEXPLS1                 ;EXP=EXP+1
000436 UDOSCI2          JMP       URIGHTS                 ;DO REMAINDER
000437 UENGNOT          LDA       FACEXP
000438 UMOD3LP          BPL       UCHKMD3
000439                  CLC
000440                  ADC       #3
000441                  BMI       UMOD3LP
000442 UCHKMD3          CMP       #3
000443                  BCC       UGOTMOD
000444                  SBC       #3                      ;C=1
000445                  BNE       UMOD3LP
000446 UGOTMOD          BIT       FACEXP
000447                  STA       DIGB4DPT                ;SET DIGIT COUNT
000448                  STA       EXPADJ
000449                  LDA       FACEXP
000450                  SEC
000451                  SBC       EXPADJ
000452                  STA       FACEXP
000453                  JSR       UABSEXP
000454                  LDA       #2                      ;CREATE 2-EXPADJ
000455                  SEC                               ;(2,1,0)->(0,1,2)
000456                  SBC       DIGB4DPT
000457                  STA       SPCB4DIG
000458                  INC       DIGB4DPT                ;=(1,2,3)
000459                  LDY       SPCB4DIG                ;GET INDEX
000460                  LDA       FLTMSK
000461                  BEQ       UDOEFIL
000462                  INC       SPCB4DIG                ;POINT 1 MORE
000463                  LDA       EMASK                   ;GET SIGN TYPE
000464                  LSR       A                       ;TO CARRY BIT
000465                  LDA       #'-'
000466                  BIT       FACSGN
000467                  BMI       USTORM
000468                  BCC       UDOEFIL
000469                  LDA       #'+'
000470 USTORM           STA       (MASKPT),Y
000471 UDOEFIL          DEY
000472                  BMI       UDODIGS
```

```
000473                LDA       CMAFILL
000474                AND       #$7F
000475                TAX
000476 UDOEFLL        LDA       (MASKPT),Y
000477                BNE       UDODIGS
000478                TXA                             ;GET FILLER
000479                STA       (MASKPT),Y
000480                BNE       UDOEFLL
000481 UDODIGS        LDY       SPCB4DIG              ;1ST DIG INDEX
000482                JMP       UDOSCI1               ;DO 1,2,3 DIGITS
000483                PAGE
000484 ******************************
000485 * SUBROUTINES
000486 ******************************
000487 * X=REQUEST FOR DIGIT WITH X=EXP
000488 * IE X=3 MEANS GIVE BACK 10**3 DIGIT
000489 UGET10X        JSR       UCVTX                 ;CVT X TO INDEX
000490                BMI       UNODIGT               ;X TOO BIG
000491 UGETITA        CMP       #22                   ; IS IT TOO SMALL?
000492                BCC       UUSEIT                ;NO
000493                LDA       #'0'                  ;RETURN A ZERO
000494 UNODIGT        RTS
000495 UUSEIT         TAX                             ;GET OFFSET IN STRING
000496                LDA       BCDSTR+2,X            ;GET ASCII DIGIT
000497                ORA       #$30                  ;CVT TO ASCII
000498                LDX       DIGEXP                ;RESTORE X
000499                AND       #$7F                  ;RETURN NEG OFF
000500                RTS
000501 UGETNSD        STX       DIGEXP                ;SAVE X
000502                LDA       DIGCTR                ;NEXT DIGIT COUNTER
000503                INC       DIGCTR
000504                JMP       UGETITA
000505 **********************************
000506 USGNSCN        LDA       #$20
000507                BIT       EMASK
000508                BEQ       USGNRTS               ; NO SIGN TO DO!!
000509                LDY       #$FF
000510 USIGNLP        INY
000511                CPY       MSKNDX                ;DONE YET?
000512                BEQ       USGNRTS
000513                LDA       (MASKPT),Y
000514                CMP       #'-'
000515                BEQ       UDOMSIN
000516                CMP       #'+'
000517                BNE       USIGNLP
000518                LDA       #'+'                  ;LOAD DEFAULT
000519                BNE       USTASIN
000520 UDOMSIN        LDA       #'                    ' ;DEFAULT
000521 USTASIN        BIT       FACSGN
000522                BPL       *+4
000523                LDA       #'-'
000524                STA       (MASKPT),Y
000525 USGNRTS        RTS                             ;DONE
000526 ******************************
000527 UEXPLS1        LDA       #1
000528                BCC       UABSEXP
000529                STA       REP
000530                LDA       FACEXP
000531                CLC
000532                ADC       REP                   ;ADJUST EXP
000533                STA       FACEXP                ;GET SMART AND LEAVE ME HERE
000534 UABSEXP        LDA       FACEXP                ;REMAP EXP
000535                BPL       USTATEN               ;POSITIVE?
000536                EOR       #$FF                  ;COMPLEMENT
000537                CLC
000538                ADC       #1                    ;
000539 USTATEN        STA       TENEXP                ;SAVE ABS(FACEXP)
000540                RTS
000541 ***********************************
000542 UCVTX           STX       DIGEXP
000543                LDA       FACEXP
000544                SEC
000545                SBC       DIGEXP
000546                RTS
000547 ****************************************
000548 UROUNDI         JSR       UCVTX
000549                BMI       UROUNDX
000550                CMP       #22                   ;IS IT TOO SMALL
000551                BCS       UROUNDX               ;YES
000552                TAX                             ;GET INDEX
```

```
000553 UROUNDA         LDA       BCDSTR+2,X              ;IN HALF BCD FORM
000554                 CMP       #$05                    ;ROUNDABLE?
000555                 BCC       UROUNDX+1               ;NO
000556                 LDA       #9                      ;THIS BYTE MUST GO TO ZERO IN
                                                                 CASE UROUNDO EXECUTED.
000557                 BCS       UROUNDB
000558 UROUNDL         DEX
000559                 BMI       UROUNDO                 ;OVERFLOW ED!!
000560                 LDA       BCDSTR+2,X
000561 UROUNDB         SED                               ;ADD IN DECIMAL
000562                 ADC       #0                      ;CARRY SET !!
000563                 CLD                               ;OFF AGAIN
000564                 CMP       #$10                    ;TRANSFER TO CARRY
000565                 AND       #$0F                    ;CORRECT FORM
000566                 STA       BCDSTR+2,X
000567                 BCS       UROUNDL                 ;STOP IF NO CARRY
000568 UROUNDX         CLC                               ;MUST FOR NO ACTION CASE
000569                 RTS
000570 UROUNDO         LDA       #1                      ;MUST HAVE BEEN ALL 9'S !
000571                 STA       BCDSTR+2                ;NOW=10000000...
000572                 RTS                               ;CARRY SET SAY EXP 1 TO SMALL
000573
000574 ; ###############################################################################################
000575 ; #   END OF FILE:  B3PRU2.TEXT
000576 ; #   LINES      :  567
000577 ; #   CHARACTERS :  25164
000578 ; ###############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 578   CHARACTERS: 25714
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "DISKSTUF1.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:34 PM
|  Modified: Wednesday, December 31, 1997             4:37:11 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  DISKSTUF1.TEXT
000004 ; ###############################################################################################
000005
000006                 SBTL      "CHARACTER I/O"
000007 * CHARACTER I/O-
000008 *
000009 INALLWD         EQU       255                     ;MAX LINE LENGTH
000010 INPUTLIN:       TXA                               ;PRINT THE PROMPT
000011                 STA       TEMP
000012                 BEQ       ILM                     ;IF NO PROMPT
000013                 JSR       PRNACHAR
000014 ILM:            LDX       INFLNO
000015                 BNE       DSEXEC
000016                 JSR       CTCOFF
000017                 BRK                               ;GET.LINE
000018                 DFB       SRED
000019                 DW        SLINTB
000020                 BNE       FUK1
000021                 STA       TRMPOS                  ;CURSOR NOW AT LEFT.
000022                 JSR       CTCON
000023 IL4             LDX       SNOCHRS                 ;HOW MANY CHARS WERE TYPED?
000024                 LDA       #$5C                    ;MAY WANT TO CANCEL THE LINE
000025                 CPX       #INALLWD                ;TOO MANY CHARS?
000026                 DEX                               ;POINT AT LAST CHAR TYPED
000027                 BCC       CLR2EOL
000028                 JSR       PRNACHAR                ;PRINT THE BACKSLASH
000029                 LDX       TEMP                    ;GET PROMPT CHAR
000030                 JSR       CRDO                    ;ADVANCE TO NEXT LINE
000031                 JMP       INPUTLIN
000032 CLR2EOL         LDA       #31                     ;CLR TO EOL
000033                 JSR       PRNACHAR
000034                 LDA       #$0D                    ;CARRIAGE RETURN.
000035                 JSR       PRNACHAR
000036                 LDA       #$0A                    ;LF.
000037 PRNACHAR        STA       OUTCHAR
000038                 LDA       #3                      ;3 PARMS
000039                 STA       SCHRTB
000040                 BRK
000041                 DFB       SWRT
000042                 DW        SCHRTB
000043                 BNE       FUK1
000044                 LDA       OUTCHAR                 ;RETURN A-REG
000045                 RTS
000046 DSEXEC          LDY       SLINTB+1                ;SAVE CONSOLE REF NUM
000047                 STX       SLINTB+1                ;STUFF IN EXEC'S REF NUM INSTEAD
000048                 BRK
000049                 DFB       SRED
000050                 DW        SLINTB
000051                 STY       SLINTB+1
000052                 BEQ       IL4
000053                 JSR       EXCCLS
000054                 BEQ       ILM
000055 EXCCLS          STX       RWRFNM                  ;MUST CLOSE THE FILE
000056                 PHA                               ;SAVE ERR CODE
000057                 LDA       #0                      ;NO MORE EXECING
000058                 STA       INFLNO
000059                 JSR       CLSEND
000060                 PLA                               ;GET ERR CODE BACK
000061                 CMP       #SEEOF                  ;IF OUT OF DATA, OK
000062                 BEQ       *+5
000063 FUK1            JMP       SERROR
000064                 RTS
000065                 SBTL      "TYP(), REC()" "
000066 TYP:            JSR       CONINT                  ;MAKE AN INTEGER
000067                 JSR       GTFLNO0                 ;GET FCBNDX
000068                 BEQ       NOTOPN                  ;Z BIT SET IF FILE NOT OPEN
000069                 SEC                               ;INDICATE READ ONLY
000070                 ROR       IOFLG
000071                 LDA       FCB+XUID,Y              ;GET TYPE OF FILE
000072                 AND       #$0F                    ;IF INDETERMINATE, BLOW
```

```
000073                  EOR     #UNKNTY                 ;IF AN UNKNOWN TYPE
000074                  BEQ     RTNVAL
000075                  EOR     #UNKNTY
000076                  TAY
000077                  LDA     #8                      ;RETURN 8 FOR TEXT FILES
000078                  CPY     #TXTTYP
000079                  BEQ     RTNVAL
000080                  JSR     PREBIN
000081 TYP1             JSR     GETNDX                  ;CALC PTR INTO FILE BUF.
000082                  LDA     (NDXPTR),Y
000083                  BNE     TYP2
000084                  LDY     FCBNDX
000085                  LDA     FCB+XBUFOFS,Y
000086                  ORA     FCB+XBUFOFS+1,Y
000087                  BEQ     TYP2
000088                  JSR     NXRCD                   ;READ IN NEXT RECORD.
000089                  JMP     TYP1
000090 TYP2             JSR     GETVAL                  ;GIVEN DESCRIPTOR, GIVE TYP FUNCTION
000091                  LDA     TYPFNT,Y
000092 RTNVAL:          TAY
000093                  JMP     SNGFLT
000094 * REC() FUNCTION:
000095 REC:             JSR     CONINT                  ;MAKE IT AN INTEGER
000096                  JSR     GTFLNO0                 ;CALC FCBNDX
000097                  BEQ     NOTOPN                  ;BRANCH IF FILE NOT OPEN
000098                  LDA     FCB+XUID,Y              ;GET FILE TYPE
000099                  AND     #$0F
000100                  CMP     #TXTTYP
000101                  BNE     GTREC
000102                  LDA     FCB+XSEGNM,Y
000103                  BEQ     GTREC                   ;CAT FILES ARE OK
000104                  JSR     GETRN                   ;A TEXTTYPE FILE DOESN'T KEEP TRACK
000105 GTREC            LDA     FCB+XRNUM+1,Y           ;HIGH ORDER
000106                  PHA
000107                  LDA     FCB+XRNUM,Y
000108                  TAY
000109                  PLA
000110                  JMP     GIVAYF                  ;TELL HIM WHERE IT IS.
000111 NOTOPN:          LDA     #SEFNO                  ;FILE WASN'T OPEN, BOZO!
000112                  JMP     SERROR                  ;SOS ERROR
000113 GETRN            LDY     FCBNDX                  ;CALC THE REC NUM FROM THE POS IN THE FILE
000114                  LDA     FCB,Y
000115                  STA     RWRFNM
000116                  LDY     #GTM                    ;GET.MARK
000117                  JSR     SETGO
000118                  LDY     #4                      ;MOVE INTO DVDND
000119 GTRN0            LDA     OUTMRK-1,Y
000120                  STA     DVDND-1,Y
000121                  DEY
000122                  BNE     GTRN0
000123                  BEQ     GTRN3
000124 GETRN1           LDY     #4
000125 GTRN2            LDA     FEOF-1,Y
000126                  STA     DVDND-1,Y
000127                  DEY
000128                  BNE     GTRN2
000129 GTRN3            JSR     DIV
000130                  LDY     FCBNDX
000131                  LDA     QUOTNT                  ;Put Record number into File
000132                  STA     FCB+XRNUM,Y
000133                  LDA     QUOTNT+1
000134                  STA     FCB+XRNUM+1,Y
000135                  RTS
000136                  PAGE
000137                  SBTL    "CLOSE" "
000138 DCLOSE:          BEQ     CLSALL                  ;IF NO FILE SPECIFIED, CLOSE 'EM ALL
000139                  JSR     GTFLNO
000140                  LDA     #0
000141                  STA     SUBFLG
000142                  JSR     CLOSEM                  ;CLOSE JUST THIS FILE
000143 CLSDONE          LDX     #0
000144                  LDA     SUBFLG
000145                  STX     SUBFLG
000146                  BEQ     *+5
000147                  JMP     SERROR
000148                  RTS
000149 CLSALL:          LDA     #0                      ;CLOSE ALL 10 FILES
000150                  STA     SUBFLG
000151                  LDA     #9                      ;CLOSE ALL 10 FILES
000152 CLSA1:           PHA
```

```
000153                  TAX
000154                  JSR       GTFLNO1
000155                  BEQ       CLSAL2
000156                  JSR       CLOSEM                    ;CLOSE THIS GUY,
000157 CLSAL2:          PLA
000158                  SEC
000159                  SBC       #1
000160                  BPL       CLSA1                     ;DO EM ALL
000161                  JMP       CLSDONE
000162 * ROUTINE TO CLOSE ONE FILE. ENTER Y-REG=FCB OFFSET
000163 CLOSEM:          CPX       FILNO+1                   ;IS THIS THE OUTPUT FILE.
000164                  BNE       DNTCLS
000165                  LDA       #$FF
000166                  STA       FILNO+1                   ;YES, DO AN OUTPUT #0.
000167                  STA       FILNO
000168 DNTCLS           STY       YSAVE
000169                  JSR       WRTRCD
000170                  BEQ       *+4
000171                  STA       SUBFLG
000172                  LDY       YSAVE
000173                  LDA       FCB,Y                     ;HAS THE FILE BEEN OPENED?
000174                  BNE       *+5
000175                  JMP       NOTOPN
000176                  LDA       FCB+XSEGNM,Y              ;RELEASE FILE BUFFER
000177                  BEQ       CLOSEM2                   ;IF A CATALOG FILE
000178                  CMP       #$FF
000179                  BEQ       CLOSEM2                   ;TEXT FILES DON'T HAVE MEM ALLOC'D
000180                  STA       SEGNUM
000181                  LDY       #RLS
000182                  JSR       SETGO
000183                  LDY       YSAVE                     ;GET FCBNDX BACK
000184                  LDA       #0
000185                  STA       FCB+XSEGNM,Y
000186 CLOSEM2          LDA       FCB,Y                     ;GET REFNUM
000187                  STA       RWRFNM                    ;TELL SOS WHAT IT IS
000188                  LDX       #FCBLEN                   ;CLEAN OUT THE GUYS FCB...
000189                  LDA       #0
000190 CLS3:            STA       FCB,Y
000191                  INY
000192                  DEX
000193                  BNE       CLS3
000194 CLSEND:          LDY       #CLS
000195                  JMP       SETGO
000196                  SBTL      "FIND FILE ROUTINES" "
000197 * ROUTINE TO GET A FILE NUMBER- #<EXPR> FROM PROGRAM.
000198 * ALTERNATE ENTRY GTFLNO1
000199 *      TO CALC OFFSET INTO FCB GIVEN FILE # IN X-REG
000200 *
000201 * ON EXIT:
000202 *     X=FILE#, A=REF#, Y=FCB INDEX
000203 *     FLAGS SET ON A-REG
000204 *
000205 GTFLNO:          JSR       CHKPND                    ;MUST HAVE #
000206                  JSR       GETBYT
000207 GTFLNO0:        DEX
000208 GTFLNO1:        TXA                                 ;TO A-REG FOR MULTIPLY
000209                  BMI       BADBOY1                   ;OBVIOUSLY A BAD VALUE
000210                  CPX       #10
000211                  BCS       BADBOY1
000212 * NOW MULT A BY FCBLEN TO GET OFFSET
000213                  STX       SVFLNO
000214                  LDY       #FCBLEN-1
000215                  STA       TEMPFOR
000216 FCBMUL:          ADC       TEMPFOR
000217                  DEY
000218                  BNE       FCBMUL
000219                  TAY                                 ;INDEX NOW.
000220                  STY       FCBNDX
000221                  LDA       FCB,Y                     ;GET REFNUM
000222                  STA       RWRFNM                    ;WILL USE IT HERE, SURELY!
000223                  RTS
000224 BADBOY1:         JMP       FCERR
000225 PDLHNDL:         JSR       CONINT                    ;MAP 0->1, 1->2, 2->5, 3->6.
000226                  CPX       #4
000227                  BCS       BADBOY1
000228                  TXA                                 ;GET PDLNUM TO READ-
000229                  PHA
000230                  EOR       #1                        ;COMPLEMENT LAST BIT-
000231                  LSR       A                         ;AND GET IT TO C.
000232                  PLA
```

```
000233                  ROL     A                       ;ROL IT IN.
000234                  STA     JMODE
000235                  TXA
000236                  AND     #1                      ;0 GOT X,1 FOR Y.
000237                  ORA     #2
000238 READJOY          PHA                             ;SAVE FOR A SEC.
000239                  LDY     #PDL                    ;READ IT NOW.
000240                  JSR     SETGO
000241                  PLA
000242                  TAX
000243                  LDA     JMODE+1,X               ;GET VAL.
000244                  TAY
000245                  JMP     SNGFLT                  ;RETURN IT.
000246 BUTTON           JSR     CONINT
000247                  CPX     #4
000248                  BCS     BADBOY1
000249                  TXA
000250                  AND     #$FE
000251                  ASL     A
000252                  STA     JMODE
000253                  TXA
000254                  AND     #01
000255                  JMP     READJOY
000256 *
000257 * SUBROUTINE TO GET A FILE NAME FOR A DISK COMMAND.
000258 * ALTERNATE ENTRY GETNAM2: GET A NAME, BUT PUT IT IN BUF STARTING
000259 * AT POSITION SPECIFIED IN THE X-REG
000260 * ON EXIT: PTHPTR IS SET UP, AS IS NAMBUF WITH THE NAME-
000261 *  FIRST BYTE OF NAMBUF CONTAINS LEN OF STRING, LIKE SOS EXPECTS
000262 *
000263 GETNAME:        LDX     #0                      ;ENTRY 1 (FIRST FILE NAME)
000264 GETNAM2:        LDY     CURLIN+1                ;IMMEDIATE MODE?
000265                  STX     FORPNT                  ;SAVE POSN INTO NOUNSTK
000266                  INY
000267                  BNE     DFRD                    ;NO, A DEFERRED CALL
000268                  JSR     CHRGOT
000269                  CMP     #$80
000270                  BCS     DFRD
000271                  CMP     #$22                    ;IF SO, DO IT TO IT
000272                  BEQ     DFRD
000273                  DEY                             ;NOW Y=$FF, X=POS TO PUT NAME AT
000274 GTNM:            INY
000275                  INX                             ;NEXT POS
000276                  LDA     (TXTPTR),Y              ;MOVE CHAR TO SAFE AREA
000277                  STA     NAMBUF,X
000278                  BEQ     GOTNAM
000279                  CMP     #$3A                    ;COLON OR COMMA ENDS STRING
000280                  BEQ     GOTNAM
000281                  CMP     #'                      ' ;OR SPACE
000282                  BEQ     GOTNAM
000283                  CMP     #$2C
000284                  BNE     GTNM
000285 * WE'VE GOT THE NAME:
000286 GOTNAM:          TYA                             ;LEN OF NAME TO A
000287                  PHA                             ;ADJUST TXTPTR
000288                  JSR     ADDON
000289                  PLA
000290 GOTN2:           LDY     FORPNT                  ;OFFSET TO BEGIN OF NAME
000291 GOTN22           STA     NAMBUF,Y                ;PUT LEN HERE FOR SOS
000292                  CLC                             ;CALC POINTER INTO PTHPTR
000293                  TYA
000294                  ADC     #>NAMBUF
000295                  STA     PTHPTR
000296                  LDA     #0
000297                  ADC     #<NAMBUF
000298                  STA     PTHPTR+1
000299                  JMP     CHRGOT                  ;SKIP OVER TRAILING SPACES.
000300 DFRD:            LDY     #$FF                    ;ALLOW ONLY STRINGS
000301                  STY     VALTYP
000302                  JSR     FRMEVL                  ;EASY AS MUD PIE
000303                  JSR     NOTFAC                  ;GO THROUGH THE STRING MATING RITUAL
000304                  LDX     FORPNT
000305                  PHA
000306                  JSR     GOTN2                   ;DO STUFF
000307                  PLA
000308                  STA     FORPNT
000309                  BEQ     DFRD3
000310                  LDY     #$FF                    ;MOVE STRING TO NOUNSTK
000311 DFRD2:           INY
000312                  INX
```

```
000313                  LDA       (INDEX),Y
000314                  STA       NAMBUF,X
000315                  DEC       FORPNT
000316                  BNE       DFRD2
000317 DFRD3             STX       FORPNT
000318                  JSR       FRECNOW                   ;FINISH THE DIRTY STRINGS AND GO HOME.
000319                  LDX       FORPNT
000320 RTSQ7             RTS
000321 *
000322 * ROUTINE TO GET FILE # AND DO A POSITION OPERATION IF NECESSARY
000323 FILNUM:           JSR       GTFLNO
000324                  BNE       *+5
000325                  JMP       NOTOPN
000326                  JSR       CHRGOT
000327                  CMP       #$2C
000328                  BNE       RTSQ7                     ;NAW, NONE OF THAT.
000329                  JSR       CHKCOM                    ;EAT IT.
000330                  JSR       FRMNUM                    ;GET REC NUM
000331                  JSR       POSINT
000332                  JSR       WRTRCD2
000333                  LDY       FCBNDX                    ;MOVE RECNUM INTO FCB
000334                  LDA       FACLO
000335                  STA       FCB+XRNUM,Y
000336                  LDA       FACMO
000337                  STA       FCB+XRNUM+1,Y
000338 RPOSN:            JSR       POSREC                    ;POSITION TO RECORD FIRST
000339                  LDY       FCBNDX                    ;IF A TEXT TYPE, DON'T DO MORE
000340                  LDA       FCB+XUID,Y
000341                  AND       #$0F
000342                  CMP       #TXTTYP
000343                  BEQ       RDGE
000344                  JSR       SETPARMS
000345                  LDA       FCB+XFLGS,Y
000346                  ASL       A
000347                  BMI       RDGE
000348                  LDY       #RED
000349                  JSR       SETUP
000350                  JSR       GOSOS
000351                  BEQ       RDGE
000352                  CMP       #$4C
000353                  BEQ       READ25                    ;IF NOT AN OUT-OF DATA, BLOW IT
000354                  JMP       SERROR
000355 READ25            JSR       GETNDX                    ;ZERO OUT BUFFER
000356                  LDA       #DDEOR
000357                  LDY       #0
000358                  STA       (NDXPTR),Y
000359 RDGE              RTS
000360 DUMSHIT           JMP       NOTOPN
000361
000362 ; ##############################################################################################
000363 ; #   END OF FILE:  DISKSTUF1.TEXT
000364 ; #   LINES      :  355
000365 ; #   CHARACTERS :  15183
000366 ; ##############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 366   CHARACTERS: 15739
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "DISKSTUF2.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:34 PM
|  Modified: Wednesday, December 31, 1997             4:37:11 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  DISKSTUF2.TEXT
000004 ; ###############################################################################################
000005
000006 *
000007 * ROUTINE TO CALL BEFORE EACH READ OR WRITE - DETERMINES IF THE
000008 * GUY IS DOING THE CORRECT OPERATION ON THE CORRECT TYPE OF FILE
000009 PREBIN:          LDA       #BINTIP
000010                  DFB       44
000011 PRETXT:          LDA       #TXTTYP
000012                  STA       TYPSAV
000013                  LDY       FCBNDX
000014                  LDA       FCB,Y
000015                  BEQ       DUMSHIT
000016                  LDA       FCB+XFLGS,Y               ;IS OPEN COMPLETE YET?
000017                  ASL       A                         ;BIT 6 ON IF NO
000018                  BPL       CKTYP                     ;YES, JUST CHECK OPERATION TYPE
000019                  ASL       A                         ;MAKE 0 STATUS
000020                  STA       FCB+XFLGS,Y
000021                  LDA       FCB+XUID,Y
000022                  AND       #$0F
000023                  CMP       #UNKNTY
000024                  BNE       FINOPN
000025                  JSR       GETNDX                    ;CALC PTR TO FILE BUFFER
000026                  LDA       RWRFNM                    ;SAVE REFNUM
000027                  PHA
000028                  LDA       #>NDXPTR
000029                  STA       PTHPTR
000030                  LDA       #<NDXPTR+1
000031                  STA       PTHPTR+1
000032                  JSR       GETFI                     ;GET FILE INFO
000033                  LDA       FID                       ;HAS THE TYPE ALREADY BEEN SET UNBEKNOWNST TO US?
000034                  CMP       #UNKNTY
000035                  BNE       GTTYP1                    ;IF SO, DON'T RESET IT.
000036 * THE FILE TYPE HAS NOT YET BEEN SET, SO WE CAN NOW SET IT TO
000037 * WHICHEVER TYPE (TEXT OR BINARY) THAT WAS REQUESTED
000038                  JSR       TSTOUT
000039                  LDA       TYPSAV
000040                  STA       FID
000041                  LDY       #SFI
000042                  JSR       SETGO
000043                  LDA       TYPSAV                    ;A TEXT FILE?
000044                  CMP       #TXTTYP
000045                  BNE       GTTYP1
000046                  PLA                                 ;GET REF NUM
000047                  PHA                                 ;KEEP STACK CLEAN
000048                  STA       RWRFNM
000049                  JSR       EXRTS                     ;SET UP IS.NEW.LINE
000050 GTTYP1:          PLA                                 ;GET BACK REFNUM
000051                  STA       RWRFNM
000052                  LDY       FCBNDX
000053                  LDA       FCB+XUID,Y
000054                  AND       #$F0
000055                  ORA       TYPSAV
000056                  STA       FCB+XUID,Y
000057 FINOPN           JSR       RPOSN
000058                  LDY       FCBNDX
000059 *
000060 * NOW CHECK FOR A FILE TYPE MISMATCH
000061 *
000062 CKTYP:           LDA       FCB+XUID,Y                ;GET TYPE OF FILE
000063                  EOR       TYPSAV
000064                  AND       #$0F
000065                  BNE       ERTYP
000066                  RTS
000067 ERTYP:           JMP       CHKERR
000068                  PAGE
000069 * GET THE NECESSARY BUFFER POINTER AND
000070 * PUT THE FILE NAME THERE FOR PREBIN
000071 DOPEN:           JSR       GTFLNO
000072                  BEQ       DOP2                      ;BRANCH IF NOT PREVIOUSLY OPEN
```

```
000073                  STX       XSAV                    ;SAVE FILE #
000074                  JSR       CLOSEM                  ;GO DO IT.
000075                  LDX       XSAV                    ;GET REF # BACK
000076                  JSR       GTFLNO1
000077 DOP2             LDX       #$00                    ;LOOK FOR 'AS' OPTION
000078                  LDA       #ASTKN
000079                  JSR       TRYESC
000080                  BNE       NOTAS                   ;NAW, FORGET IT
000081                  JSR       CHRGET                  ;AS WHAT?
000082                  CMP       #INPTKN                 ;INPUT?
000083                  BNE       AS1
000084                  LDX       #$10                    ;INPUT ONLY
000085 AS1              CMP       #OUTTKN                 ;OUTPUT?
000086                  BNE       AS2
000087                  LDX       #$20                    ;OUTPUT ONLY
000088 AS2              LDA       #EXTKN                  ;EXTENSION?
000089                  JSR       TRYESC
000090                  BNE       AS3
000091                  LDX       #$60                    ;EXTENSION.
000092 AS3:             TXA                               ;AS SOMETHING?
000093                  BEQ       GVERR                   ;NO, HE SCREWED UP.
000094                  JSR       CHRGET
000095 NOTAS            TXA
000096                  LDY       FCBNDX                  ;PUT FILE MODE INTO FCB
000097                  STA       FCB+XUID,Y
000098                  LSR       A
000099                  LSR       A
000100                  LSR       A
000101                  LSR       A
000102                  CMP       #2
000103                  BCC       *+4
000104                  LDA       #$3
000105                  STA       INREQ
000106                  JSR       CHKCOM                  ;SYNTAX IS OPEN #N,<NAME>
000107                  JSR       GETNAME
000108                  JSR       GRECLN                  ;GET REC LEN
000109                  JMP       NOTAS2
000110 GVERR:           JMP       SNERR
000111 GVTT             JSR       CLOSEM2                 ;CLOSE IN SOS&WIPE FCB
000112                  JMP       MSMTCH
000113 * NOW GO OPEN IT & CREATE IT IF NECCESSARY
000114 NOTAS2:          LDA       #UNKNTY                 ;UNKNOWN TYPE OF FILE
000115                  STA       TYPSAV
000116                  JSR       OPNP2
000117                  LDY       FCBNDX
000118                  LDA       #$40                    ;EVEN IF A KNOWN TYPE, MUST READ IN DATA LATER.
000119                  STA       FCB+XFLGS,Y
000120 * FILE HAS BEEN OPENED- NOW SET UP FCB
000121                  LDA       REFOUT                  ;GET REFNUM
000122                  STA       FCB+XRFNM,Y
000123                  LDA       FSTYP                   ;IS IT A DIRECTORY?
000124                  CMP       #$0D
000125                  BCS       GTDIR
000126                  LDA       FID
000127                  CMP       #UNKNTY                 ;ONLY ALLOW UNDESIGNATED FILES,
000128                  BEQ       OPG2
000129                  CMP       #BINTIP                 ;BINARY DATA
000130                  BEQ       OPG2
000131                  CMP       #TXTTYP
000132                  BNE       GVTT
000133 OPG2             AND       #$0F                    ;GET FILE TYPE TO L.O. 4 BITS
000134                  ORA       FCB+XUID,Y
000135                  STA       FCB+XUID,Y
000136                  LDA       FAUX                    ;GET RECLEN FROM DIRECTORY
000137                  STA       FCB+XRECL,Y
000138                  LDA       FAUX+1
000139                  STA       FCB+XRECL+1,Y
000140                  ORA       FAUX
000141                  BNE       RNDYSLAB
000142                  LDA       #2
000143                  STA       FAUX+1
000144                  STA       FCB+XRECL+1,Y
000145 RNDYSLAB         LDA       FID                     ;IF TEXT FILE, DO SPECIAL
000146                  CMP       #TXTTYP
000147                  BEQ       GTTXT
000148 * EVERYTHING READY. GET BUFFER
000149 OPLP:            LDY       FAUX+1
000150                  LDX       FAUX
000151                  STX       DVSR                    ;FOR EXTENSION MODE
000152                  STY       DVSR+1
```

```
000153                BEQ       OPGD1
000154                INY                           ;ODD SIZE RECLEN, GIVE HIM ROOM IN BUFFER
000155 OPGD1          STY       IOPGCN              ;ASK SOS FOR BUFFER
000156                LDA       #0
000157                STA       IOPGCN+1
000158                LDA       #2                  ;FIND ANYTHING ANYWHERE
000159                STA       ISRCHMD
000160                LDA       #18
000161                STA       ISEGID
000162                LDY       #FND                ;FIND A SEGMENT.
000163                JSR       SETUP
000164                JSR       GOSOS
000165                BEQ       OPGOOD
000166                CMP       #$E1                ;SEG REQUEST DENIED?
000167                BNE       OPSHIT
000168                LDY       FAUX+1
000169                LDX       FAUX
000170                BEQ       *+3
000171                INY
000172                TYA
000173                JSR       SCRUNCH
000174                JMP       OPLP
000175 GTDIR:         JMP       SETCAT
000176 OPSHIT         JMP       SERROR
000177 GTTXT          JSR       EXRTS               ;DO STUFF (SET IS.NEW.LINE)
000178                LDA       #$FF                ;SEGNUM OF $FF INDICATES A TEXT FILE
000179                STA       OSEGNM
000180 OPGOOD         LDX       FCBNDX
000181                LDA       BSBNKP
000182                ORA       #$80                ;MAKE A VIRTUAL ADDRESS.
000183                STA       FCB+XBUFPT,X
000184                TAY
000185                LDA       BSBNKP+1            ;GET ACTUAL PAGE NO
000186                SEC
000187                SBC       #$20                ;MAKES A VIRTUAL
000188                STA       FCB+XBUFPT+1,X
000189                JSR       FIXSBC
000190                LDA       OSEGNM              ;SAVE THE SEG NUM FOR CLOSE
000191                STA       FCB+XSEGNM,X
000192                LDA       FCB,X               ;PUT REFNUM BACK
000193                STA       RWRFNM
000194                LDA       FCB+XUID,X          ;IS IT AS EXTENSION?
000195                AND       #$40
000196                BEQ       NOEXT
000197                LDA       FID
000198                CMP       #UNKNTY
000199                BEQ       NOEXT
000200                CMP       #TXTTYP             ;DO TEXT FILES DIF.
000201                PHP
000202                LDA       #0
000203                STA       FCB+XFLGS,X
000204                PLP
000205                BEQ       TXTEXT
000206                JSR       GETRN1
000207                LDA       RMNDR
000208                TAY
000209                ORA       RMNDR+1             ;AT BEG. OF REC?
000210                BEQ       OPGOT
000211                TYA
000212                BNE       *+4
000213                DEC       RMNDR+1             ;SO AS EXTENSION WILL WORK PROPERLY.
000214                DEC       RMNDR
000215 OPGOT          LDA       RMNDR+1
000216                PHA
000217                LDA       RMNDR
000218                PHA
000219                JSR       RPOSN
000220                LDY       FCBNDX              ;PUT INDEX INTO BUFFER
000221                PLA
000222                STA       FCB+XBUFOFS,Y
000223                PLA
000224                STA       FCB+XBUFOFS+1,Y
000225                RTS
000226 NOEXT:         JSR       GETNDX
000227                LDA       OSEGNM              ;TEXT FILE?
000228                BMI       XYZZY
000229                LDA       NAMBUF              ;LEN OF NAME
000230                TAX
000231                INX
000232                LDY       #0
```

```
000233 NOX2            STA       (NDXPTR),Y              ;MOVE NAME TO FILE BUFFER
000234                 INY
000235                 LDA       NAMBUF,Y
000236                 DEX
000237                 BNE       NOX2
000238 XYZZY           RTS
000239 TXTEXT          LDY       #4
000240                 LDA       #0
000241 TE1             STA       DSPLMNT-1,Y
000242                 DEY
000243                 BPL       TE1
000244                 INC       DSPLMNT-1               ;BASE MODE 1.
000245                 LDY       #STM
000246                 JSR       SETGO
000247                 JMP       NOEXT
000248                 PAGE
000249 *
000250 * Routine to do the OPEN portion for INVOKE, LOAD, SAVE, and
000251 *   even OPEN!
000252 *
000253 * On Entry, if ACC=2 then CREATE a new file, else the file should exist.
000254 *
000255 OPNPRTB          STA       INREQ                   ;ACC=1 if called from LOAD, 2 if called from SAVE
000256                 STX       TYPSAV                  ;X = Type of File
000257                 JSR       GETNAME                 ;Get file name & put it in PTHPTR
000258                 JSR       SETPROG
000259 OPNP2           EQU       *
000260                 JSR       GETFISET                ;Set up SOS GET_FILE_INFO Block
000261                 JSR       SETUP                   ;Set up SOS Call
000262                 JSR       GOSOS                   ;Do SOS Call & only give error on Read-Only
000263                 BEQ       OPNP3                   ;OPENed OK
000264                 CMP       #SENBK                  ;SOS Err - Not Block Device
000265                 BNE       OPPP2
000266                 LDA       #TXTTYP                 ;OPEN a Character Device as a TEXT (ASCII) file
000267                 STA       FID
000268                 STA       FSTYP                   ;TELL HIM ITS A TEXT FILE AND OK STORAGE TYPE
000269                 BNE       OPNP3
000270 OPPP2           CMP       #SEFNF                  ;FILE THERE?
000271                 BNE       CMPLAIN
000272                 LDX       INREQ                   ;READ ONLY ACCESS?
000273                 DEX
000274                 BEQ       CMPLAIN
000275                 LDA       TYPSAV
000276                 STA       INFLID
000277                 JSR       CRTDO
000278                 JMP       OPNP2
000279 OPNP3           LDA       #>INREQ
000280                 STA       OPNLST
000281                 LDA       #<INREQ
000282                 STA       OPNLST+1
000283                 LDA       #1                      ;ONLY SPECIFY FILE ID
000284                 STA       OPNLNGTH
000285                 LDY       #OPN
000286                 JSR       SETUP                   ;DOITTOIT
000287                 JSR       GOSOS
000288                 BEQ       OPNRTS                  ;IF IT WORKS, DON'T FIX IT!
000289                 CMP       #SEMEM                  ;OUT OF MEM?
000290                 BNE       CMPLAIN                 ;NO, SOME OTHER ERROR
000291                 LDA       #4
000292                 JSR       SCRUNCH                 ;GIVE SOS SOME MEM
000293                 JMP       OPNP3
000294 OPNRTS          LDA       REFOUT
000295                 STA       RWRFNM                  ;PREPARE FOR EVERYBODY
000296                 RTS
000297 CMPLAIN         JMP       SERROR
000298
000299 ; ###########################################################################################
000300 ; #   END OF FILE:  DISKSTUF2.TEXT
000301 ; #   LINES      :  292
000302 ; #   CHARACTERS :  12434
000303 ; ###########################################################################################
```

```
+-----------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 303   CHARACTERS: 12990
|
+-----------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "DISCMDS.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:33 PM
|  Modified: Wednesday, December 31, 1997             4:37:10 PM
|
+------------------------------------------------------------------------------

000001 ; ###########################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  DISCMDS.TEXT
000004 ; ###########################################################################################
000005
000006                  SBTL      "GENERAL OVERHEAD SUBROUTINES"
000007 GETNDX:          LDX       FCBNDX                      ;GET OFFSET
000008                  LDA       FCB+XBUFOFS,X               ;BUFFER ALWAYS STARTS ON PAGE BOUNDARY
000009                  STA       NDXPTR
000010                  CLC
000011                  LDA       FCB+XBUFPT+1,X              ;GET PAGE ADDRESS
000012                  ADC       FCB+XBUFOFS+1,X
000013                  LDY       FCB+XBUFPT,X                ;GET BANK NUM
000014                  JSR       FIXADC
000015                  STA       NDXPTR+1
000016                  STY       NDXPTRB
000017                  LDY       #0                          ;ALWAYS GIVE HIM Y=0 FOR INDIRECT
000018                  RTS
000019 *
000020 * TEST TO SEE IF ENOUGH ROOM FOR MORE DATA.
000021 *ENTRY: A=# OF BYTES TO GO INTO BUFFER
000022 *  FCBNDX=INDEX INTO FCB
000023 * RETURN: CARRY SET IF IT FITS, CLEAR IF NOT
000024 TSTFTX:          LDA       #1                          ;AHH,PLEASE, JUST ONE MORE BYTE
000025 TSTFIT:          CLC
000026                  STA       LENSAV
000027                  LDY       FCBNDX
000028                  ADC       FCB+XBUFOFS,Y
000029                  STA       TMPPTR
000030                  LDA       FCB+XBUFOFS+1,Y
000031                  LDX       LENSAV                      ;IF A 256-BYTE STRING, DO IT RIGHT
000032                  BNE       *+3
000033                  SEC
000034                  ADC       #0
000035                  STA       TMPPTR+1
000036                  LDA       FCB+XRECL,Y
000037                  CMP       TMPPTR                      ;DON'T DO BANKS, BECAUSE THIS IS JUST
000038                  LDA       FCB+XRECL+1,Y               ;A COMPARE OPERATION
000039                  SBC       TMPPTR+1
000040                  RTS
000041 *
000042 * SUBROUTINE TO UPDATE FILE BUFFER OFFSET. ENTER LENSAV=
000043 *  NUMBER OF BYTES(0 IF 256-CHAR STRING), FCBNDX= INDEX INTO FCB
000044 *
000045 UPOFS:           CLC
000046                  LDA       LENSAV
000047                  LDY       FCBNDX
000048                  ADC       FCB+XBUFOFS,Y
000049                  STA       FCB+XBUFOFS,Y
000050                  LDA       FCB+XBUFOFS+1,Y
000051                  LDX       LENSAV
000052                  BNE       *+3
000053                  SEC
000054                  ADC       #0
000055                  STA       FCB+XBUFOFS+1,Y
000056                  RTS
000057                  PAGE
000058 * MULTIPLY & DIVIDE ROUTINES. NOTE THAT ALL THE REFERENCED
000059 * LOCATIONS MUST BE IN ZERO PAGE FOR THE ADDRESSING TO WORK
000060 * CORRECTLY
000061 MUL:             LDA       #0                          ;CLEAR OUT PARTIAL PRODUCT
000062                  STA       RSLT+2
000063                  STA       RSLT+3
000064                  LDY       #$10                        ;INDEX FOR 16 BITS
000065 MUL2:            LDA       MLTPLR                      ;IS LOW-ORDER BIT SET?
000066                  LSR       A
000067                  BCC       MUL4                        ;NO, DON'T MULT BY THIS BIT
000068                  CLC
000069                  LDX       #$FE
000070 MUL3:            LDA       MLTPLR2,X                   ;THIS IS WHERE ZPAGE,X WRAPAROUND IS
000071                  ADC       MLTPLR2+2,X                 ;DEPENDED UPON
000072                  STA       MLTPLR2,X
```

```
000073                  INX
000074                  BNE       MUL3
000075 MUL4             LDX       #3                    ;SHIFT ONCE NOW
000076 MUL5             ROR       RSLT,X
000077                  DEX
000078                  BPL       MUL5
000079                  DEY
000080                  BNE       MUL2
000081                  RTS
000082 DIV:             LDY       #$10                  ;DVDND/DVSR=>QUOTNT,RMNDR
000083 DIV2:            ASL       QUOTNT
000084                  ROL       QUOTNT+1
000085                  ROL       RMNDR
000086                  ROL       RMNDR+1
000087                  SEC
000088                  LDA       RMNDR
000089                  SBC       DVSR
000090                  TAX
000091                  LDA       RMNDR
000092                  SBC       DVSR
000093                  TAX                             ;STOLEN DIRECTLY FROM THE GOOD OL'
                                                                 APPLE II MONITOR
000094                  LDA       RMNDR+1
000095                  SBC       DVSR+1
000096                  BCC       DIV3
000097                  STX       RMNDR
000098                  STA       RMNDR+1
000099                  INC       QUOTNT                ;SET BIT IN QUOTIENT
000100 DIV3             DEY
000101                  BNE       DIV2
000102                  RTS
000103 *
000104 *  VALTYP DESCRIPTORS
000105 *
000106 VREAL            EQU       $0
000107 VLNT             EQU       $40
000108 VSTR             EQU       $FF
000109 VINT             EQU       $08
000110 *
000111 * DTABLE AND VTABLE FOLLOWING.....
000112 *
000113 DTABLE           DFB       DDINT,DDFP,DDLNT,DDSTR
000114                  DFB       DDMXSTR,DDEOR
000115 VTABLE           DFB       VINT,VREAL,VLNT,$FF
000116                  DFB       VSTR,$01
000117 TBLLEN           EQU       VTABLE-DTABLE-1       ;# OF BYTES - 1 FOR THE TABLE SIZE.
000118 TYPFNT           DFB       2,1,3,4
000119                  DFB       4,5
000120                  SBTL      "READ, WRITE, CHAIN"
000121 *
000122 *               CHAIN
000123 *
000124 CHAIN            JSR       LDRUN                 ;GET READY TO LOAD THE NEXT PROGRAM
000125                  JSR       GARBA2                ;AND MAKE SURE GARBAGE COLLECTION IS DONE
000126 * NOW MOVE THE VARIABLES.
000127                  CLC                             ;CALCULATE HOW FAR TO MOVE -1 EXTRA SO DON'T HIT STRINGS
000128                  LDA       FRETOP
000129                  SBC       STREND
000130                  STA       DELTA
000131                  LDA       FRETOP+1
000132                  SBC       STREND+1
000133                  LDY       FRETOPB
000134                  JSR       FIXSBC
000135                  STA       DELTA+1
000136                  TYA                             ;TRANSFER BANK
000137                  SBC       STRENDB
000138                  STA       DELTAB
000139                  LDA       ARYTAB                ;BEGIN MOVING VARS HERE.
000140                  STA       LOWTR
000141                  LDA       ARYTAB+1
000142                  STA       LOWTR+1
000143                  LDA       ARYTABB
000144                  STA       LOWTRB
000145                  JSR       MVUP                  ;MOVE IT, BOY!
000146                  LDA       ARYTAB                ;SAVE ARYTAB IN THE FAC
000147                  STA       FAC
000148                  LDA       ARYTAB+1
000149                  STA       FAC+1
000150                  LDA       ARYTABB
000151                  STA       FAC+2
```

```
000152 * NOW LOAD THE PROGRAM....
000153                 JSR       DOLD2                    ;LOAD IT...
000154                 JSR       FLOAD                    ;CLEAN UP.
000155 * NOW RESTORE THE OLD VARIABLES
000156                 LDA       ARYTAB                   ;THIS IS WHERE TO MOVE TO.
000157                 STA       LOWTR
000158                 SEC                                ;ALSO CALCULATE DELTA
000159                 SBC       FAC
000160                 STA       DELTA
000161                 LDA       ARYTAB+1
000162                 STA       LOWTR+1
000163                 SBC       FAC+1                    ;FAC+0,1 IS WHERE TO START MOVING DOWN FROM
000164                 LDY       ARYTABB
000165                 STY       LOWTRB
000166                 JSR       FIXSBC
000167                 STA       DELTA+1
000168                 TYA
000169                 SBC       FAC+2
000170                 STA       DELTAB
000171                 LDA       FAC
000172                 STA       ARYTAB
000173                 STA       INDEX1                   ;BEGIN MOVE LOCATION
000174                 LDA       FAC+1
000175                 STA       ARYTAB+1
000176                 STA       INDEX1+1
000177                 LDA       FAC+2
000178                 STA       INDEX1B
000179                 STA       ARYTABB
000180                 JSR       MVDWN                    ;MOVE IT FOLKS!
000181 FRUN:           JSR       STXTPT                   ;RESET TEXT POINTER
000182                 JSR       STKINI                   ;CLEAN UP THE STACK
000183                 LDA       LINNUM                   ;DID HE SPECIFY A LINE?
000184                 ORA       LINNUM+1
000185                 BEQ       *+5
000186                 JSR       LUK4IT                   ;POSITION TO THE LINE
000187                 JMP       NEWSTT                   ;AND GO RUN!
000188 * ROUTINE TO SET UP FOR CHAIN AND RUN
000189 LDRUN:          LDA       #1                       ;ONLY LOAD FILES, DON'T CREATE ANY
000190                 STA       CMDFLG                   ;Set COMMAND call flag
000191                 LDX       #PRGTY
000192                 JSR       OPENIT
000193                 LDA       #1                       ;(For SELECTOR operations,
000194                 STA       RNFLG                    ;  set flag that a program has been run)
000195                 JSR       CHRGOT                   ;WHAT WAS LAST CHAR?
000196                 BEQ       DATSALL                  ;IF A TERMINATOR, OK
000197                 JSR       CHKCOM                   ;MUST HAVE COMMA IF NOT TERMINATOR
000198                 JMP       LINGET                   ;GET THE LINE NUMBER
000199 DATSALL:        LDA       #0                       ;NO LINE SPECIFIED
000200                 STA       LINNUM
000201                 STA       LINNUM+1
000202                 RTS
000203                 PAGE
000204 *
000205 * HERE IS WHAT YOU'VE BEEN WAITING FOR ALL THIS LISTING!!!
000206 *    READ AND WRITE !!!!!
000207 *
000208 DREAD:          SEC
000209                 ROR       IOFLG
000210                 JSR       FILNUM                   ;GET FILE NUMBER AND REC# IF SPECIFIED
000211                 JSR       PREBIN
000212                 JSR       TSTIN                    ;TEST IF A BINARY FILE AND IN INPUT MODE-
000213                 JSR       CHRGOT
000214                 CMP       #$3B
000215                 BNE       RDRTS
000216                 JSR       CHRGET                   ;EAT THE SEMICOLON
000217                 BNE       DRD2                     ;ALWAYS
000218 RDRTS           RTS
000219 DRD1            JSR       CHRGOT                   ;MORE VARS?
000220                 BEQ       RDRTS
000221                 JSR       CHKCOM                   ;SEP'D BY COMMAS
000222 DRD2            JSR       GETNDX                   ;GET PTR INTO FILE BUFFER
000223                 LDA       (NDXPTR),Y               ;GET DESCRIPTOR
000224                 BNE       DRD3                     ;IF NOT AN END-OF-RECORD
000225                 JSR       DRDNXT                   ;READ IN THE NEXT RECORD IF NON-EMPTY.
000226                 JMP       DRD2
000227 ODERR           JMP       CHKEOF
000228 *
000229 * WE HIT AN END OF RECORD MARK. GO TO THE NEXT RECORD IF WE'RE NOT
000230 * AT THE END OF THIS ONE.
000231 DRDNXT          LDY       FCBNDX                   ;IF BUFOFS=0000, GIVE 'OUT-OF-DATA'
```

```
000232                LDA       FCB+XBUFOFS,Y
000233                ORA       FCB+XBUFOFS+1,Y
000234                BEQ       ODERR
000235                JMP       NXRCD                ;GO TO THE NEXT RECORD
000236 DRD3            JSR       GETVAL               ;TYPE OF VAR IN THE FILE
000237                STA       VALTYP
000238                BMI       RDSTRI               ;READ A STRING
000239                PHA
000240                JSR       MYPTRGET
000241                BIT       VALTYP               ;WHAT DOES HE WANT?
000242                BMI       MSMTCH
000243                JSR       NTINT3
000244                TAY                            ;SAVE DSC OF WHAT HE WANTS
000245                PLA                            ;GET TYPE OF DATA WE HAVE
000246                CMP       #VINT
000247                CLC                            ;ASSUM NOT INT
000248                BNE       *+3
000249                SEC
000250                ROR       INTFLG
000251                STA       VALTYP
000252                TYA
000253                PHA                            ;SAVE VALTYP OF VAR READING INTO
000254                LDA       NDXPTR
000255                LDY       NDXPTR+1
000256                LDX       NDXPTRB
000257                CLC
000258                ADC       #1
000259                BCC       *+3
000260                INY
000261                JSR       ISVRET2              ;UNPACK VAL INTO FAC
000262                PLA                            ;SEE WHAT THE USER IS RDING INTO
000263                PHA
000264                CMP       #DDLNT               ;LNG INT?
000265                BNE       NTLNT
000266                JSR       CONV2LNG
000267 DRD5            PLA
000268                JSR       GETVAL
000269                STA       VALTYP
000270                LSR       A
000271                LSR       A
000272                LSR       A
000273                LSR       A                    ;GET INT FLG INTO CARRY
000274                ROR       INTFLG
000275                JSR       LETP2                ;PUT IT IN THE VAR
000276 DRD6            JSR       GETNDX
000277                LDA       (NDXPTR),Y
000278                STA       DSCRPT               ;SAVE THE DESCRIPTOR
000279                JSR       CALCLEN
000280                JSR       UPOFS
000281                CMP       FCB+XRECL+1,Y        ;AT THE END OF THE BUFFER?
000282                BCC       DRD7                 ;NO.
000283                LDA       FCB+XBUFOFS,Y        ;MAYBE, CHECK LOW BYTE.
000284                CMP       FCB+XRECL,Y
000285                BCC       DRD7
000286                JSR       DRDNXT               ;YES, READ THE NEXT BUF.
000287 DRD7            JMP       DRD1                 ;LOOP.
000288 NTLNT:          JSR       CONV2FLT
000289                JMP       DRD5
000290 MSMTCH          JMP       ERTYP
000291 MYPTRGET        JSR       PTRGET               ;SAME AS PTRGET,
000292                STA       FORPNT               ;BUT PUTS PTRS IN FORPNT ALSO
000293                STY       FORPNT+1
000294                LDX       VARPNTB
000295                STX       FORPNTB
000296                RTS
000297 *
000298 * WAS A STRING
000299 *
000300 RDSTRI          JSR       MYPTRGET
000301                BIT       VALTYP               ;ANYTHING ELSE ILLEGAL
000302                BPL       MSMTCH
000303                LDY       #0
000304                LDA       (NDXPTR),Y           ;GET DESCRIPTOR
000305                INY
000306                CMP       #DDMXSTR             ;255-CHAR STRING?
000307                BNE       NTMXSTR
000308                LDA       #$FF
000309                DEY                            ;STR LEN=255
000310                DFB       44
000311 NTMXSTR         LDA       (NDXPTR),Y           ;GET STR LEN
```

```
000312                  PHA                             ;SAVE THE LEN
000313                  INY                             ;COPY CHARS FROM STRING
000314                  TYA
000315                  CLC
000316                  ADC       NDXPTR                ;UPDATES NDXPTR
000317                  STA       STRNG1
000318                  LDA       NDXPTR+1
000319                  ADC       #0
000320                  STA       STRNG1+1
000321                  LDA       NDXPTRB
000322                  STA       STRNG1B
000323                  PLA
000324                  TAY
000325                  JSR       STRFI1
000326                  JSR       INPCOM
000327                  JMP       DRD6
000328 *
000329 * HERE IS THE 'WRITE' OPERATION---
000330 *
000331 DWRITE:          LSR       IOFLG                 ;SPECIFY A WRITE OP.
000332                  JSR       FILNUM                ;GET FILE # AND REC #S
000333                  JSR       PREBIN                ;MAKE SURE A BINARY FILE
000334                  JSR       TSTOUT                ;WITH OUTPUT ALLOWED
000335                  JSR       CHRGOT
000336                  CMP       #$3B
000337                  BNE       WRRTS
000338                  JSR       CHRGET                ;EAT THE SEMICOLON
000339                  BNE       DWR2                  ;ALWAYS
000340 WRRTS:           RTS
000341 DWR1:            JSR       CHRGOT                ;ANY MORE VARS TO WRITE?
000342                  BEQ       WRRTS
000343                  JSR       CHKCOM                ;VARS MUST BE SEPD BY COMMAS
000344 DWR2             JSR       GETXPR                ;GET AN EXPRESSION TO WRITE
000345 DWR3             LDX       FCBNDX
000346                  LDA       FCB+XBUFOFS,X         ;GOT AN EMPTY BUFFER?
000347                  ORA       FCB+XBUFOFS+1,X
000348                  BNE       DWR4
000349                  JSR       POSREC
000350                  JSR       SETPARMS              ;YES, EMPTY BUFFER - WRITE IT
000351                  LDY       #WRT                  ; TO RESERVE THE DISK SPACE.
000352                  JSR       SETGO
000353                  JSR       POSREC                ;REPOSITION.
000354 DWR4             JSR       GETNDX                ;GET NDXPTR TO DATA
000355                  JSR       CALCLEN               ;CALC LEN OF THIS ITEM
000356                  JSR       TSTFIT                ;WILL IT FIT HERE?
000357                  BCS       ITFITS                ;YES.
000358 * SEE IF DATA ITEM WILL FIT IN AN EMPTY RECORD
000359                  LDY       FCBNDX
000360                  LDA       FCB+XRECL+1,Y
000361                  BNE       NXREC                 ;IF A RECLEN>255, YES FOR SURE
000362                  LDA       LENSAV
000363                  BEQ       DNTFIT                ;IF A 256-BYTE STRING, SORRY CHARLIE
000364                  LDA       FCB+XRECL,Y
000365                  CMP       LENSAV                ;ENOUGH ROOM IN THE RECORD
000366                  BCC       DNTFIT                ;NOPE.
000367 NXREC            JSR       NXRCD                 ;GO TO THE NEXT RECORD
000368                  JMP       DWR3                  ;AND LOOP
000369 DNTFIT           JMP       ODERR
000370 * MOVE IT INTO THE DATA BUFFER
000371 ITFITS:          LDY       #0
000372                  LDX       LENSAV
000373                  DEX
000374                  STX       XSAV
000375                  LDA       DSCRPT                ;GET THE DESCRIPTOR
000376                  STA       (NDXPTR),Y            ;PUT IT INTO THE BUFFER
000377                  INY
000378                  DEX
000379                  CMP       #DDSTR                ;IF A LONG STRING, DO SPECIAL
000380                  BNE       ITF1
000381                  STX       XSAV                  ;PUT THE STRING LEN OUT
000382                  TXA
000383                  STA       (NDXPTR),Y            ;SAVE LEN OF STRING
000384                  BEQ       DATDON                ;IF A NULL STRING
000385                  INY
000386 ITF1             LDX       #0                    ;FOR ABS INDIRECT
000387                  LDA       (VARPNT,X)
000388                  STA       (NDXPTR),Y            ;MOVE IT INTO THE FILE BUFFER
000389                  INY
000390                  INC       VARPNT
000391                  BNE       *+4
```

```
000392                  INC       VARPNT+1
000393                  DEC       XSAV
000394                  BNE       ITF1                    ;MOVE ALL THE BYTES
000395 DATDON:           JSR       UPOFS                   ;UPDATE THE OFFSET
000396                  JSR       SETOUT                  ;INDICATE TO WRITE THIS BUFFER
000397                  JSR       TSTFTX                  ;CAN WE FIT THE END OF RECORD MARKER?
000398                  BCC       GODWR1                  ;IF NOT, FORGET IT.
000399                  JSR       GETNDX                  ;YES, PUT IT IN THERE
000400                  LDA       #DDEOR                  ;SHOULD BE A 00
000401                  STA       (NDXPTR),Y              ;BUT DON'T UPDATE XBUFOFS
000402 GODWR1            BIT       VALTYP                  ;WAS IT A STRING?
000403                  BPL       *+5
000404                  JSR       FRECNOW                 ;FREE THE SUCCER.
000405                  JMP       DWR1
000406 SETOUT            LDY       FCBNDX
000407                  LDA       FCB+XFLGS,Y
000408                  ORA       #$80
000409                  STA       FCB+XFLGS,Y
000410                  RTS
000411 * SUBROUTINE GIVEN DSCRPT=DESCRIPTOR, RETURNS LEN IN LENSAV
000412 CALCLEN:          LDA       DSCRPT
000413                  EOR       #DDMXSTR                ;IF A MAX STRING, LEN=00(LO BYTE)
000414                  BEQ       GOTLEN
000415                  EOR       #DDSTR-DDMXSTR
000416                  BNE       CANUM                   ;IT'S A NUMBER OF SOME TYPE
000417                  TAY                               ;A=0 FROM THE EOR SO Y=0
000418                  LDA       (FACMO),Y               ;GET STRING LENGTH
000419                  SEC
000420                  BCS       ADJLEN                  ;BRANCH ALWAYS
000421 CANUM             LDA       DSCRPT                  ;GET THE DESCRIPTOR BACK
000422                  AND       #$0F                    ;STRIP HI BITS
000423                  CLC
000424 ADJLEN            ADC       #1
000425 GOTLEN            STA       LENSAV
000426                  RTS
000427 *
000428 * MAIN SUBROUTINE FOR 'DWRITE'- FIGURES OUT WHAT AN EXPRESSION IS AND
000429 * PACKS IT INTO THE FAC.
000430 *
000431 GETXPR:           JSR       CHRGOT                  ;WHAT IS FIRST CHAR OF EXPRESSION?
000432                  BCC       GETX2                   ;IF A DIGIT, NOT AN INTEGER VAR
000433                  JSR       ISLETC                  ;IS IT A VALID VAR?
000434                  BCC       GETX2                   ;IF NOT, DON'T BOTHER CHKING FOR INTEGER
000435 * MUST CHECK FOR SINGLE INTEGER VARIABLE. IF IT IS, WRITE IT OUT
000436 * AS AN INTEGER
000437                  JSR       SVTXT                   ;SAVE THE TEXT POINTER
000438                  JSR       PTRGET                  ;GET PTR TO VAR
000439                  BIT       INTFLG                  ;IF NOT AN INTEGER
000440                  BPL       NOTINT                  ;THE FUCK IT
000441                  JSR       CHRGOT                  ;WHATS AFTER THE VAR? EITHER EOL OR COMMAS OK
000442                  BEQ       NTINT3                  ;A TERMINATOR
000443                  CMP       #$2C
000444                  BEQ       NTINT3
000445 NOTINT            JSR       RSTTXT                  ;RESET TXTPTR
000446 GETX2             LDA       #$20                    ;TAKE ANY KIND OF EXPRESSION
000447                  STA       VALTYP
000448                  JSR       FRMEVL
000449                  LDA       #<FAC                   ;WILL PROBABLY LOAD FROM FAC
000450                  LDY       #>FAC
000451                  LDX       #0                      ;CURRENT BANK
000452                  BIT       VALTYP
000453                  BPL       GTX3
000454                  JSR       NOTFAC
000455                  LDA       INDEX+1                 ;PNTR TO THE STRING
000456                  LDY       INDEX
000457                  LDX       INDEXB
000458 GTX3              STA       VARPNT+1
000459                  STY       VARPNT
000460                  STX       VARPNTB
000461                  BIT       VALTYP
000462                  BMI       NTINT2                  ;IF A STRING, OK
000463                  BVS       NTINT2                  ;DON'T KNOW ABOUT LONG INTEGERS
000464                  JSR       ROUND                   ;ROUND A REAL NUMBER
000465                  LDA       FACSGN
000466                  ORA       #$7F
000467                  AND       FACHO
000468                  STA       FACHO
000469 NTINT2            LSR       INTFLG                  ;NOT AN INTEGER
000470 NTINT3            LDA       VALTYP
000471 * FALLS INTO THE GETDSC ROUTINE
```

```
000472                  LDX       #0
000473                  BIT       INTFLG                  ;IF AN INTEGER, DONE
000474                  BMI       GOTDSC
000475                  LDX       #TBLLEN
000476 GTDSC1           CMP       VTABLE,X
000477                  BEQ       NTDONE
000478                  DEX
000479                  BPL       GTDSC1
000480                  BMI       ERRMIS                  ;DON'T KNOW WHAT TO DO.
000481 NTDONE           CPX       #4                      ;IS IT A STRING?
000482                  BNE       GOTDSC
000483                  LDY       #0                      ;GET LEN OF STRING
000484                  LDA       (FACMO),Y
000485                  ADC       #0                      ;CARRY IS SET
000486                  BEQ       GOTDSC
000487                  DEX
000488 GOTDSC           LDA       DTABLE,X
000489                  STA       DSCRPT
000490                  RTS
000491 ERRMIS           JMP       ERTYP
000492 * TEST WHETHER INPUT OR OUTPUT IS ALLOWED
000493 TSTIN            LDA       #$10                    ;CHECK READ
000494                  DFB       44
000495 TSTOUT           LDA       #$20                    ;CHECK WRITE
000496                  LDY       FCBNDX
000497                  STA       TEMP
000498                  LDA       FCB+XUID,Y
000499                  AND       #$F0
000500                  BEQ       TSTRTS
000501                  AND       TEMP
000502                  BEQ       BADMD                   ;IF BIT NOT SET, NONO!
000503 TSTRTS           RTS
000504 GETVAL           LDY       #TBLLEN                 ;GIVEN DESCRIPTOR, GET VALTYP
000505 GTVAL1           CMP       DTABLE,Y
000506                  BEQ       GOTVAL
000507                  DEY
000508                  BPL       GTVAL1
000509 BADMD            JMP       ERTYP                   ;TYPE MISMATCH
000510 GOTVAL           LDA       VTABLE,Y
000511                  RTS
000512                  PAGE
000513 *
000514 * HERE IS THE CHARACTER FILE I/O
000515 *
000516 OUTPUT:          JSR       CHKPND                  ;FORMAT IS OUTPUT#<FILNO>
000517                  JSR       GETBYT
000518                  CPX       #11                     ;ONLY FILE #S 0-10
000519                  BCC       *+5
000520                  JMP       FCERR
000521                  DEX
000522                  TXA
000523                  STA       FILNO
000524                  STA       FILNO+1
000525                  BMI       OUTRTS                  ;IF OUT #0, DONE
000526                  JSR       GTFLNO1
000527                  JSR       PRETXT                  ;MAKE SURE A TEXT FILE
000528 OUTRTS           RTS
000529 EXEC             LDA       INFLNO
000530                  BEQ       EXEC2                   ;CLOSE PREVIOUS EXEC FILE.
000531                  STA       RWRFNM
000532                  JSR       CLSEND
000533 EXEC2            LDA       #1                      ;READ ONLY EXEC FILES
000534                  JSR       OPNPRTB
000535                  STA       INFLNO                  ;REF NUMBER GOES HERE
000536                  LDA       FID                     ;OK TYPE OF FILE?
000537                  CMP       #TXTTYP                 ;IF SO, WE'RE DONE
000538                  BEQ       EXRTS
000539                  JSR       CLSEND                  ;CLOSE THE FILE
000540                  STA       INFLNO                  ;RESET INFLNO TO 00
000541                  LDA       FID                     ;IS IT AN UNDETERMINED TYPE?
000542                  CMP       #UNKNTY
000543                  BEQ       OUTRTS
000544                  JMP       CHKERR                  ;TYPE MISMATCH
000545 EXRTS            LDA       RWRFNM                  ;SET IS-NEW-LINE TRUE FOR CRS
000546                  STA       ISNLTB+1
000547                  BRK
000548                  DFB       SNWL
000549                  DW        ISNLTB
000550                  BEQ       OUTRTS
000551 CLTERR           PHA                               ;CLOSE A FILE, THEN GIVE AN ERROR
```

```
000552                JSR       CLSEND
000553                PLA
000554                JMP       SERROR
000555 DSKLIN:         LDX       FILNO                   ;DO A READ ON THIS FILE
000556                JSR       GTFLNO1
000557                LDA       FCB+XSEGNM,Y            ;MAKE SURE A GOOD FILE
000558                BEQ       LINCAT
000559                JSR       PRETXT
000560                JSR       TSTIN
000561                LDA       FCB,Y                   ;PUT REFNUM IN PLACE OF CONSOLES
000562                LDX       SLINTB+1
000563                STA       SLINTB+1
000564                BRK                               ;QUICK DO THE READ
000565                DFB       SRED
000566                DW        SLINTB
000567                STX       SLINTB+1                ;PUT BACK CONSOLE REF NUM
000568                BEQ       DSKLRT
000569 DSKEOF:         CMP       #SEEOF                  ;AN END OF FILE?
000570                BEQ       *+5
000571                JMP       SERROR                  ;NO.
000572 NMOR            JMP       CHKEOF
000573 DSKLRT          LDX       SNOCHRS                 ;TOO LONG A LINE?
000574                LDA       BUF-1,X                 ;SEE IF TERMINATED BY A CR
000575                CMP       #$0D
000576                BEQ       DSKLRT1
000577                INX
000578 DSKLRT1         DEX
000579                JMP       GDBUFS
000580 LINCAT          LDA       VALTYP                  ;PRESERVE JUST IN CASE
000581                PHA
000582                JSR       NCLN                    ;NEXT LINE OF CAT
000583                BCS       NMOR                    ;NO MORE, EOF
000584                PLA
000585                STA       VALTYP
000586                LDX       #0                      ;MOVE LINE IN...
000587 MVCATL          LDA       CATBUF,X
000588                STA       BUF,X
000589                INX
000590                CPX       #79                     ;ENOUGH YET?
000591                BCC       MVCATL
000592                BCS       DSKLRT1
000593
000594 ; ###############################################################################################
000595 ; #   END OF FILE:  DISCMDS.TEXT
000596 ; #   LINES      :  587
000597 ; #   CHARACTERS :  25764
000598 ; ###############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 598   CHARACTERS: 26316
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "FILESTUF.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:35 PM
|  Modified: Wednesday, December 31, 1997             4:37:12 PM
|
+------------------------------------------------------------------------------

000001 ; ###############################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  FILESTUF.TEXT
000004 ; ###############################################################################################
000005
000006                  SBTL      "CREATE" "
000007 * HERE IS THE CREATE OPERATION
000008 *
000009 * FORMAT: CREATE <NAME>, CATALOG|TEXT|DATA [,AEXPR]
000010 *
000011 CREATE:          JSR       GETNAME
000012                  JSR       CHKCOM
000013                  LDY       #0                      ;INSTRTYP OF 0
000014                  LDX       #$FF                    ;NO TYPE ASSOCIATED YET...
000015                  CMP       #CATATK                 ;A DIR FILE?
000016                  BNE       CR1
000017                  LDX       #15                     ;A DIR TYPE
000018                  LDY       #$0D                    ;INSTRTYP OF $0D (SUBDIRECTORY)
000019 CR1:             CMP       #TEXTTK                 ;A TEXT TYPE FILE?
000020                  BNE       CR2
000021                  LDX       #TXTTYP
000022 CR2              CMP       #DATATK                 ;DATA FILE?
000023                  BNE       CR3
000024                  LDX       #BINTIP
000025 CR3              TXA
000026                  BMI       CRSNR                   ;SNEER AT HIM (OR SNERR?)
000027                  STY       INSTRTYP
000028                  STA       INFLID
000029                  JSR       CHRGET                  ;EAT THE TYPE TOKEN
000030                  JSR       GRECLN                  ;IF A RECLEN SPECIFIED, GET IT
000031                  LDA       INSTRTYP                ;IF A CATFILE, RECLEN IS FILE SIZE.
000032                  BEQ       CRTDO
000033                  LDA       INAUXID                 ;MAKE REC LEN THE EOF.
000034                  STA       INEOF
000035                  LDA       INAUXID+1
000036                  STA       INEOF+1
000037                  JMP       CRTDO2
000038 CRSNR            JMP       SNERR
000039 CRTDO:           LDA       #0                      ;USUALLY CREATE A STANDARD FILE
000040                  STA       INSTRTYP
000041                  STA       INEOF
000042                  STA       INEOF+1
000043 CRTDO2           LDA       #>CRTTBL                ;CREATE TABLE
000044                  STA       CRTLST
000045                  LDA       #<CRTTBL
000046                  STA       CRTLST+1
000047                  LDA       #8                      ;ONLY A LITTLE BIT OF STUFF
000048                  STA       INLNGTH
000049                  LDY       #CRT
000050                  STY       INEOF+2
000051                  STY       INEOF+3
000052                  JMP       SETGO
000053                  SBTL      "LOAD, SAVE"
000054 * HERE ARE SAVE & LOAD COMMANDS
000055 *
000056 BYTBLOCK         DFB       $FF,$7F,0,0             ;4 Bytes (for expansion)
000057 *
000058 *   Here is the SAVE routine.
000059 *
000060 SAVE             EQU       *
000061                  LDA       #0                      ;Set the COMMAND flag
000062                  STA       CMDFLG
000063                  JSR       SETPROG
000064                  LDA       #2                      ;CREATE the file if necessary
000065                  LDX       #PRGTY                  ;BASIC Program Type
000066                  JSR       OPENIT                  ;OPEN the file by name
000067                  JSR       PGMLEN                  ;Calculate the Program length into BCDSTR
000068 *
000069 *  Length of PGM is NOT represented in the file anymore.
000070 *    Bytes 0-1: 0,0 always! (This is for FORMAT compatibility w/earlier
000071 *    versions of BASIC. They can't load new PGMs but this BASIC can
000072 *   LOAD old PGMS.
```

```
000073                  LDA     #2                      ;Write 2 bytes of 0 to the file
000074                  STA     INBYTES                 ;  to maintain compatible Format with
000075                  LDA     #0                      ;  older versions on BASIC
000076                  STA     INBYTES+1
000077                  LDA     #>BCDSTR                ;Point SBFPTR to start of BCDSTR
000078                  STA     SBFPTR
000079                  LDA     #<BCDSTR
000080                  STA     SBFPTR+1
000081                  JSR     SVWRT                   ;Write the length to the file
000082                  LDA     TXTTAB
000083                  STA     INDEX                   ;Put TXTTAB into INDEX to use as a moving
000084                  LDA     TXTTAB+1                ;  starting point for the SOS WRITE
000085                  STA     INDEX+1
000086                  LDA     TXTTABB
000087                  STA     INDEXB
000088                  JSR     SAVE1
000089                  LDY     #3
000090                  LDA     #0
000091 DSP0LUP          STA     DSPLMNT,Y
000092                  DEY
000093                  BPL     DSP0LUP
000094                  LDA     #2
000095                  STA     DSPLMNT-1
000096                  LDY     #STE
000097                  JSR     SETGO
000098                  JSR     CLSEND                  ;NOW CLOSE IT
000099                  JSR     SETSOS
000100                  RTS
000101 OPENIT           STX     TEMPFOR                 ;Save File Type
000102                  JSR     OPNPRTB
000103                  LDX     TEMPFOR                 ;Does the file type match
000104                  CPX     FID                     ;  the File ID?
000105                  BEQ     *+5                     ;Yes, skip the JMP & do the RTS
000106                  JMP     NBP                     ;If not, then Not BASIC Program
000107                  RTS
000108 DOLOAD:          LDA     #1                      ;OPEN IT, BUT GIVE AN ERROR IF NOT THERE
000109                  LDX     #PRGTY
000110                  JSR     OPENIT
000111 DOLD2            EQU     *                       ;BASIC PROGRAM TYPE ONLY
000112                  JSR     CLEARONS
000113                  LDA     #>BCDSTR                ;Read 2 dummy bytes from start of pgm file
000114                  STA     SBFPTR
000115                  LDA     #<BCDSTR                ;These will be overwritten unused, but it
000116                  STA     SBFPTR+1                ;  saves the need to SET_MARK to the 3rd byte
000117                  LDA     #0                      ;  of the file where the program actually starts.
000118                  STA     INBYTES+1
000119                  LDA     #2
000120                  STA     INBYTES
000121                  JSR     LDRED
000122                  JSR     LOOP1                   ;Fill BCDSTR+2,+3,+5,+5 from FEOF
000123 *
000124 *    Here is the setting up of ARYTAB(B) and check if Pgm will fit.
000125 *
000126                  LDY     TXTTABB                 ;Get the TXTTAB Bank
000127 SVLP1           LDA     BCDSTR+4                ;Length >64K?
000128                  BEQ     SVLP2
000129                  DEC     BCDSTR+4                ;Yes, shift out 32K bytes
000130                  CLC
000131                  LDA     BCDSTR+3                ; and add them to the hi byte
000132                  ADC     #$80                    ; of the 16 bit portion,
000133                  STA     BCDSTR+3
000134                  INY                             ; then kick up the bank indicator.
000135                  BCC     SVLP1
000136                  INY                             ;(2 more times if we crossed a
000137                  INY                             ;   bank pair boundary.)
000138                  BCS     SVLP1
000139 SVLP2           CLC                             ;OK the lengths are <64K and banks right.
000140                  LDA     BCDSTR+2                ;Now we add in the differentials in the
000141                  ADC     TXTTAB                  ; starting and ending points.
000142                  TAX
000143                  LDA     BCDSTR+3
000144                  ADC     TXTTAB+1
000145                  JSR     FIXADC                  ;Adjust for 2<=page<82
000146                  CPY     HIMEMB                  ;IF NOT IN THE LAST BANK, OK
000147                  BCC     SVLDPP
000148                  BNE     OUTOFM
000149                  CPX     MEMSIZ
000150                  PHA
000151                  SBC     MEMSIZ+1
000152                  PLA
```

```
000153                  BCS       OUTOFM
000154 SVLDPP           STX       ARYTAB                ;Save results in ARYTAB etc.
000155                  STA       ARYTAB+1
000156                  STY       ARYTABB
000157                  JSR       LOOP1                 ;Refill BCDSTR from FEOF
000158                  LDA       TXTTAB
000159                  STA       INDEX                 ;Put TXTTAB into INDEX to use as a moving
000160                  LDA       TXTTAB+1              ;  starting point for the SOS WRITE
000161                  STA       INDEX+1
000162                  LDA       TXTTABB
000163                  STA       INDEXB
000164                  LDA       #1
000165                  STA       BCDSTR                ;This will be used as a calling flag
000166                  JSR       SAVE1
000167                  JMP       CLSEND
000168 NBP:             JSR       CLSEND
000169                  JMP       MSMTCH
000170 SLRTS            RTS
000171 OUTOFM:          JMP       OMERR
000172 LOAD             LDA       #0                    ;Set COMMAND flag
000173                  STA       CMDFLG
000174                  JSR       SETPROG
000175                  JSR       DOLOAD
000176                  JSR       CLEARL                ;CLEARC WITH NO "CLOSE ALL" CALL.
000177                  LSR       TRFLAG                ;NOTRACE.
000178                  JSR       SETSOS
000179                  JMP       MAIN
000180 RDWRT            LDA       BCDSTR                ;It will be 0 if called from SAVE
000181                  BNE       LDRED                 ;      and 1 if called from LOAD
000182 SVWRT            LDY       #WRT
000183                  DFB       44
000184 LDRED            LDY       #RED
000185                  JSR       SETUP
000186                  JSR       GOSOS
000187                  BEQ       SLRTS
000188                  JMP       CLTERR
000189 *
000190 *    If the # of bytes left to read/write is > the # bytes of BYTBLOCK
000191 *      then read/write BYTBLOCK bytes & adjust the # bytes left.
000192 *
000193 SAVE1            EQU       *
000194                  LDA       BCDSTR+4              ;The MSB (since +5 wil always have a 0)
000195                  BEQ       somewhere
000196 LOOPAGIN         LDA       BYTBLOCK
000197                  STA       INBYTES               ;Put the number of bytes into INBYTES
000198                  LDA       BYTBLOCK+1
000199                  STA       INBYTES+1
000200                  LDA       #>INDEX               ;Tell it to start at INDEX
000201                  STA       SBFPTR
000202                  LDA       #<INDEX
000203                  STA       SBFPTR+1
000204                  JSR       RDWRT                 ;READ/WRITE the bytes
000205                  LDA       INDEX                 ;Get INDEX at old starting point
000206                  CLC
000207                  ADC       BYTBLOCK              ;Add # bytes done
000208                  STA       INDEX
000209                  LDA       INDEX+1
000210                  ADC       BYTBLOCK+1
000211                  LDY       INDEXB                ;Get the Bank
000212                  JSR       FIXADC                ;Adjust the bank/page tallys
000213                  STY       INDEXB                ; and save the new starting
000214                  STA       INDEX+1               ; point.
000215                  JSR       DOSUB                 ;Now we have fewer bytes to do
000216                  JMP       SAVE1                 ;Do it again
000217 somewhere        EQU       *                     ;We're here because the bytes left <=64K
000218                  LDA       BYTBLOCK+1
000219                  CMP       BCDSTR+3              ;More than BYTBLOCK bytes left?
000220                  BCC       LOOPAGIN              ;Yes, do a full BYTBLOCK bytes
000221                  LDA       BCDSTR+3              ;No, do the rest of the bytes.
000222                  STA       INBYTES+1
000223                  LDA       BCDSTR+2
000224                  STA       INBYTES
000225                  LDA       #>INDEX               ;Tell it to start at INDEX
000226                  STA       SBFPTR
000227                  LDA       #<INDEX
000228                  STA       SBFPTR+1
000229                  JSR       RDWRT
000230                  RTS
000231 LOOP1            LDY       #3                    ;We'll move the 4 bytes of file length
000232 LOOP1A           LDA       FEOF,Y                ;  from the GET_FILE_INFO results
```

```
000233                  STA       BCDSTR+2,Y                ;  into BCDSTR+2, +3, +4
000234                  DEY
000235                  BNE       LOOP1A
000236                  LDA       FEOF,Y                    ;Subtract 2 from LSB of FEOF since
000237                  SEC                                 ;  FEOF includes the 2 bytes of the
000238                  SBC       #2                        ;  program length.
000239                  STA       BCDSTR+2,Y
000240                  RTS
000241                  SBTL      "RENAME, (UN)LOCK, DELETE" "
000242 RENAME:          JSR       GETNAME                   ;GET FIRST NAME
000243                  LDA       PTHPTR                    ;AND SAVE ITS POINTER
000244                  PHA
000245                  LDA       PTHPTR+1
000246                  PHA
000247                  JSR       CHKCOM
000248                  INX                                 ;STORE NAME INTO NEXT LOC
000249                  JSR       GETNAM2
000250                  LDA       PTHPTR                    ;SECOND NAME IS NEW ONE
000251                  STA       NWPTHNM
000252                  LDA       PTHPTR+1
000253                  STA       NWPTHNM+1
000254                  PLA
000255                  STA       PTHPTR+1
000256                  PLA
000257                  STA       PTHPTR
000258                  LDY       #RNM                      ;RENAME IT NOW
000259                  JMP       SETGO                     ;SET UP & GO
000260 UNLOCK:          LDA       #$C3                      ;ALLOW ANYTHING HE WANTS (KINKY!)
000261                  DFB       44
000262 LOCK:            LDA       #$01                      ;LOCK IT
000263                  STA       FATRB
000264                  JSR       GETNAME
000265                  LDA       #>FATRB
000266                  STA       FLSTPTR
000267                  LDA       #<FATRB
000268                  STA       FLSTPTR+1                 ;POINT AT FILE ATTRIBUTES TABLE
000269                  LDA       #1                        ;ONLY ASINGLE ITEM IN FLIST
000270                  STA       INLNGTH
000271                  LDY       #SFI                      ;SET FILE INFO
000272                  JMP       SETGO
000273 * DELETE:
000274 DDELETE:         JSR       GETNAME                   ;GET THE FILE NAME
000275                  LDY       #DST
000276                  JMP       SETGO
000277 *---------------------------------------------------------------
000278 *
000279 * Routine to calculate Program length
000280 *
000281 *  On Exit:  BCDSTR, +1 will each have 0
000282 *            BCDSTR+2, +3, +4, +5 will have Pgm length
000283 *  Uses:     A,Y
000284 *  Routines: None
000285 *
000286 PGMLEN           EQU       *
000287                  LDY       #5                        ;6 Bytes for PGM length & flags
000288                  LDA       #0                        ;These lines are for cleaning up
000289 PGMLEN1          STA       BCDSTR,Y                  ;  before the fact.
000290                  DEY
000291                  BPL       PGMLEN1
000292                  LDA       ARYTAB                    ;Get end of program (low byte)
000293                  SEC
000294                  SBC       TXTTAB                    ;Subtract start of pgm (low byte)
000295                  STA       BCDSTR+2                  ;Save difference
000296                  LDA       ARYTAB+1                  ;Find difference of Hi bytes
000297                  SBC       TXTTAB+1
000298                  LDY       ARYTABB                   ;Does pgm spill to next Bank(s)?
000299                  CPY       TXTTABB
000300                  BEQ       PGMLEN2                   ;If not, don't adjust length
000301                  CLC
000302                  ADC       #MAXPG-MINPG
000303                  DEY                                 ;Decrement Y since we just crossed
                                                                    a bank boundary
000304 PGMLEN2          STA       BCDSTR+3                  ;Save Hi byte
000305 PGMLEN3          CPY       TXTTABB                   ;Y equal to TXTTAB bank?
000306                  BEQ       SZRTS                     ;If yes, then we're done
000307                  LDA       BCDSTR+3                  ;Get the hi byte back
000308                  CLC
000309                  ADC       #MAXPG-MINPG              ;Add bank size (Member FDIC)
000310                  STA       BCDSTR+3                  ; and resave it
000311                  LDA       BCDSTR+4                  ;If carry is set, then we need to
```

```
000312                ADC       #$0                     ; increment the program size by bank
000313                STA       BCDSTR+4                ; and resave it.
000314                DEY                               ;One less bank in counter.
000315                JMP       PGMLEN3
000316 SZRTS          RTS                               ;All done.  Let's go back!
000317 ;
000318 ;   This routine does a NUMBYTES subtract
000319 ;
000320 DOSUB          LDY       #0                      ;Start loop at 0
000321                SEC
000322                PHP                               ;Dummy Push to secure Stack
000323 LOOP           EQU       *
000324                PLP                               ;Pull Status off stack
000325                LDA       BCDSTR+2,Y
000326                SBC       BYTBLOCK,Y
000327                STA       BCDSTR+2,Y
000328                PHP                               ;Save status of calculation
000329                INY                               ;Increment loop
000330                CPY       #4                      ;At limit?
000331                BNE       LOOP                    ;No, do next byte
000332                PLP                               ;Clean up stack
000333                RTS
000334 ;
000335 ; Routine to Get the Record Length (Default is 512)
000336 GRECLN         LDA       #0                      ;DEFAULT REC LEN OF 512
000337                STA       INAUXID
000338                LDA       #2
000339                STA       INAUXID+1
000340                JSR       CHRGOT                  ;IS THERE A RECLEN SPECIFIED?
000341                BEQ       GRCRTS
000342                JSR       CHKCOM                  ;MUST HAVE A COMMA
000343                JSR       FRMNUM
000344                JSR       POSINT
000345                LDX       FACLO                   ;RECLEN MUST BE >3, <32767
000346                LDA       FACMO
000347                BMI       BUMSIZ
000348                BNE       GRC2
000349                CPX       #3
000350                BCS       GRC2
000351 BUMSIZ         JMP       FCERR
000352 GRC2           STA       INAUXID+1
000353                STX       INAUXID
000354 GRCRTS         RTS
000355 WRTRCD2        JSR       WRTRCD
000356                BNE       *+5
000357 IDNUMR         LDA       #0
000358                RTS
000359                JMP       SERROR
000360 WRTRCD         LDY       FCBNDX                  ;SHOULD WE WRITE THIS?
000361                LDA       FCB+XFLGS,Y
000362                BPL       IDNUMR
000363                LDA       FCB+XBUFOFS+1,Y
000364                PHA
000365                LDA       FCB+XBUFOFS,Y
000366                PHA
000367                ORA       FCB+XBUFOFS+1,Y         ;ANYTHING TO WRITE?
000368                BEQ       IDNUM2                  ;NO, QUIT IT.
000369                JSR       TSTFTX                  ;DO WE HAVE A TRAILING 0 ON THIS RECORD?
000370                PHP                               ;CARRY SET IF SO.
000371                JSR       POSREC                  ;DO POSITION
000372                JSR       SETPARMS
000373                PLP
000374 IDNUM2         PLA
000375                ADC       #0                      ;ADD 1 IF EXTRA NULL TO WRITE.
000376                STA       INBYTES
000377                PLA
000378                ADC       #0
000379                STA       INBYTES+1
000380                LDY       #WRT                    ;AND WRITE THE DATA
000381                JSR       SETUP
000382                JMP       GOSOS
000383 NXRCD:         JSR       WRTRCD2                 ;WRITE IF NEC.
000384                LDX       FCBNDX                  ;INCREMENT RECORD NUMBER
000385                INC       FCB+XRNUM,X
000386                BNE       NXR2
000387                INC       FCB+XRNUM+1,X
000388 NXR2:          JMP       RPOSN
000389 SETPARMS       LDA       #0
000390                LDY       FCBNDX
000391                STA       FCB+XBUFOFS,Y
```

```
000392                  STA       FCB+XBUFOFS+1,Y
000393                  JSR       GETNDX
000394                  LDA       #>NDXPTR
000395                  STA       SBFPTR
000396                  LDA       #<NDXPTR
000397                  STA       SBFPTR+1
000398                  LDY       FCBNDX
000399                  LDA       FCB+XRECL,Y
000400                  STA       INBYTES
000401                  LDA       FCB+XRECL+1,Y
000402                  STA       INBYTES+1
000403                  LDA       FCB+XRFNM,Y
000404                  STA       RWRFNM
000405                  RTS
000406 *
000407 * MISC ROUTINES
000408 *
000409 SERROR:          LDX       #0
000410 FNDAER1          CMP       ERRTBL,X
000411                  BEQ       FOUNAR1
000412                  INX
000413                  INX
000414                  BCS       FNDAER1
000415                  LDX       #SSSSSS
000416                  STA       SOSLOC
000417                  JMP       ERROR
000418 FOUNAR1          INX
000419                  LDA       ERRTBL,X
000420                  TAX
000421                  JMP       ERROR
000422 GETFISET:        LDA       #>FATRB                 ;GET INFO ON THIS FILE.
000423                  STA       FLSTPTR
000424                  LDA       #<FATRB
000425                  STA       FLSTPTR+1
000426                  LDA       #$B                     ;FIND OUT EVERYTHING
000427                  STA       INLNGTH
000428                  LDY       #GFI
000429                  RTS
000430 SETDSP:          LDY       FCBNDX                  ;SET UP DISPLACEMENT FOR REC
000431                  LDA       #0                      ;SPECIFY AT BEGINNING OF RECORD
000432                  STA       FCB+XBUFOFS,Y
000433                  STA       FCB+XBUFOFS+1,Y
000434                  LDA       FCB+XRNUM,Y             ;POSITION TO RNUM*RECLEN
000435                  STA       MLTPLR
000436                  LDA       FCB+XRNUM+1,Y
000437                  STA       MLTPLR+1
000438                  LDA       FCB+XRECL,Y
000439                  STA       MLTPLR2
000440                  LDA       FCB+XRECL+1,Y
000441                  STA       MLTPLR2+1
000442                  JSR       MUL                     ;GET BYTE # IN FILE
000443                  LDY       #4
000444 QTIP:            LDA       RSLT-1,Y                ;MOVE RESULT TO DSPLMNT
000445                  STA       DSPLMNT-1,Y
000446                  DEY
000447                  BNE       QTIP
000448                  STY       BASE                    ;MAKE IT RELATIVE TO BEGINNING OF FILE
000449 PPRTS            RTS
000450 POSREC           JSR       SETDSP                  ;POSITION TO THIS REC
000451                  LDY       #STM
000452                  JSR       SETUP
000453                  JSR       GOSOS                   ;DO THE SET.MARK
000454                  BEQ       PPRTS                   ;EVERYTHING IS OK
000455                  CMP       #SENBK                  ;IF NOT A BLOCK DEV,
000456                  BEQ       PPRTS                   ;DON'T DO IT!
000457                  CMP       #$4D
000458                  BEQ       *+5
000459                  JMP       SERROR
000460 *WANT TO SET MARK BEYOND EOF
000461                  BIT       IOFLG
000462                  BMI       PSRERR
000463                  JSR       TSTOUT                  ;TEST IF WE CAN FIRST
000464                  LDX       FCBNDX                  ;GO TO NEXT REC
000465                  INC       FCB+XRNUM,X
000466                  BNE       *+5
000467                  INC       FCB+XRNUM+1,X
000468                  JSR       SETDSP
000469                  LDY       #STE                    ;SET.EOF
000470                  JSR       SETGO
000471                  LDX       FCBNDX
```

```
000472                  LDA       FCB+XRNUM,X
000473                  BNE       *+5
000474                  DEC       FCB+XRNUM+1,X
000475                  DEC       FCB+XRNUM,X
000476                  JMP       POSREC
000477 PSRERR            JSR       GETNDX
000478                  TYA                                 ;A=0.
000479                  STA       (NDXPTR),Y
000480                  JMP       CHKEOF
000481
000482 ; ###############################################################################################
000483 ; #   END OF FILE:  FILESTUF.TEXT
000484 ; #   LINES      :  475
000485 ; #   CHARACTERS :  21734
000486 ; ###############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 486   CHARACTERS: 22288
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "CATALOG.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:33 PM
|  Modified: Wednesday, December 31, 1997             4:37:10 PM
|
+------------------------------------------------------------------------------

000001 ; ###########################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  CATALOG.TEXT
000004 ; ###########################################################################################
000005
000006                  SBTL      "CATALOG"
000007 CTPL             EQU       NAMBUF
000008 CATALOG:         EQU       *
000009                  BNE       ISANAM
000010                  BRK
000011                  DFB       GETPREF                   ;SOS CALL: GET_PREFIX
000012                  DW        PREFTB3
000013                  LDX       NAMBUF
000014                  LDA       #1
000015                  STA       INREQ                     ;SO JSR OPNP2 WILL WORK.
000016                  LDY       #0
000017                  TXA                                 ;NAME LEN
000018                  JSR       GOTN22
000019                  JSR       OPNP2
000020                  JMP       CG0                       ;SKIP AROUND OPNPRTB
000021 ISANAM:          LDA       #1                        ;ASK FOR READ MODE ONLY
000022                  JSR       OPNPRTB                   ;OPEN EITHER SPECIFIED DIRECTORY
000023 CG0              LDY       #FCBLEN                   ;WILL CLEAR CAT FCB OUT.
000024                  LDA       FSTYP                     ;CHECK STORAGE TYPE FOR DIRS
000025                  CMP       #$0D                      ;PLAIN NAVILLA
000026                  BEQ       CG1
000027                  CMP       #$0F                      ;ROOT DIRECTORY?
                                                                    (WHAT DOES TOM HAVE TO DO WITH IT?)
000028                  BNE       CATERR
000029 CG1              LDA       #0
000030                  STA       CATFCB-1,Y
000031                  DEY
000032                  BNE       CG1
000033                  LDY       #FCBLEN*10                ;GET FCBNDX CHEAPLY
000034                  STY       FCBNDX
000035                  JSR       SETCAT                    ;INITIALIZE
000036                  LDA       RWRFNM                    ;PUT REF NUM INTO FCB
000037                  STA       FCB,Y
000038 CG2              JSR       CRDO
000039                  LDY       #FCBLEN*10
000040                  STY       FCBNDX
000041                  LDA       FCB,Y                     ;GET REF.NUM INTO READ BYTE
000042                  STA       RWRFNM
000043                  JSR       NCLN                      ;NEXT CATALOG LINE
000044                  BCS       CATDON                    ;CARRY SET IF DONE
000045                  LDA       #>CATBUF
000046                  LDY       #<CATBUF
000047                  LDX       #0                        ;IN THE CURRENT BANK
000048                  JSR       STROUT
000049                  JMP       CG2
000050 CATERR:          LDA       #$40                      ;BAD PATH.
000051                  JMP       CLTERR
000052 CATDON:          JSR       CRDO                      ;AN EXTRA LINE AT THE END
000053                  LDY       #FCBLEN*10
000054                  JMP       CLOSEM2
000055 *
000056 * INITIALIZE FCB FOR A CATALOG
000057 *
000058 SETCAT           LDY       FCBNDX
000059                  LDA       #TXTTYP+$10               ;TYPE: TEXT, READ ONLY
000060                  STA       FCB+XUID,Y
000061                  LDA       #0
000062                  STA       FCB+XFLGS,Y
000063                  STA       FCB+XSEGNM,Y
000064                  LDX       FSTYP                     ;A ROOT DIR?
000065                  CPX       #$0F
000066                  BNE       NTROOT                    ;NO
000067 * FOR A ROOT DIR, FAUX CONTAINS TOTAL BLOCK, FBLKS=# USED.
000068                  LDA       FAUX
000069                  STA       FCB+XRECL,Y
000070                  LDA       FAUX+1
000071                  STA       FCB+XRECL+1,Y
```

```
000072                  LDA       FBLKS
000073                  STA       FCB+XBLKS,Y
000074                  LDA       FBLKS+1
000075                  STA       FCB+XBLKS+1,Y
000076 NTROOT:          RTS
000077 NCLN:            LDY       #70                     ;FILL CATBUF WITH SPACES FIRST
000078                  LDA       #$20
000079 NCL1             STA       CATBUF-1,Y
000080                  DEY
000081                  BNE       NCL1
000082                  STY       CATBUF+69
000083                  LDX       FCBNDX
000084                  INC       FCB+XRNUM,X
000085                  BNE       NCL2
000086                  INC       FCB+XRNUM+1,X
000087 NCL2             LDA       FCB+XFLGS,X
000088                  ASL       A                       ;GO TO APPROP ROUTIN
000089                  TAY
000090                  LDA       CTDP+1,Y
000091                  PHA
000092                  LDA       CTDP,Y
000093                  PHA
000094                  RTS
000095 CTDP:            DW        CATL1-1                 ;CAT LINE 1
000096                  DW        CBL-1                   ;BLANK LINE
000097                  DW        CHDG-1                  ;HEADING INFO
000098                  DW        CMP-1                   ;MAIN PART (FILES DISPLAYED)
000099                  DW        CEND-1                  ;ENDING.
000100                  DW        CBL-1                   ;AN EXTRA BLANK LINE
000101 *
000102 * ROUTINE GENERATES THE FIRST LINE OF A CATALOG
000103 *
000104 CATL1:           INC       FCB+XFLGS,X
000105                  LDA       #$2B                    ;LEN OF FIRST READ
000106                  JSR       RDCAT2                  ;READ IT IN.
000107                  LDA       CTPL+4                  ;GET NAME LEN
000108                  AND       #$0F
000109                  TAY
000110 CL12             LDA       CTPL+4,Y
000111                  STA       CATBUF,Y                ;PUT NAME OF DIR INTO CATBUF
000112                  DEY
000113                  BNE       CL12
000114                  LDA       #'('                    ;PUT DATE OF CREATION IN PARENS
000115                  STA       CATBUF+17
000116                  LDA       #')'
000117                  STA       CATBUF+26
000118                  LDA       CTPL+$1D                ;GET DATE DIR WAS CREATED
000119                  LDX       CTPL+$1C
000120                  LDY       #18
000121                  JSR       GENDATE
000122                  LDA       #'V'                    ;VERSION NUM
000123                  STA       CATBUF+28
000124                  LDY       CTPL+$20                ;FROM DIR
000125                  JSR       SNGFLT
000126                  JSR       FOUT                    ;MAKE TO ASCII
000127                  LDY       #$FF
000128 TPL:             INY
000129                  LDA       FBUFFR,Y
000130                  BEQ       TPRTS
000131                  STA       CATBUF+29,Y
000132                  BNE       TPL
000133 CBL:             LDX       FCBNDX
000134                  INC       FCB+XFLGS,X             ;NEXT STAGE OF CATALOG
000135 TPRTS:           LDY       FCBNDX
000136                  LDA       FCB+XFLGS,Y
000137                  CMP       #6                      ;DONE YET?
000138                  RTS
000139 *
000140 * DO THE HEADINGS LINE
000141 CHDG             LDY       #HDGLEN
000142 CHD2             LDA       HDMSG-1,Y
000143                  STA       CATBUF,Y
000144                  DEY
000145                  BNE       CHD2
000146                  BEQ       CBL                     ;ALWAYS
000147 HDMSG:           ASC       " TYPE   BLKS  NAME           MODIFIED"
000148                  ASC       " TIME  CREATED  TIME   EOF"
000149                  DFB       $0A                     ;Leave a Blank Line
000150 HDGLEN           EQU       *-HDMSG
000151 *
```

```
000152 CMP:            JSR       RDCAT                    ;READ THE CATALOG
000153                 BCS       CBL                      ;ALL DONE
000154                 LDA       CTPL                     ;IS NAME LEN=00?
000155                 AND       #$0F                     ;  (Strip off 4 high bits)
000156                 BEQ       CMP                      ;IF SO, GO DO NEXT FILE.
000157                 LDA       CTPL+$1E                 ;Get Access indicator
000158                 AND       #$C3                     ;Mask out all but access bits
000159                 CMP       #1                       ;Is it Locked?
000160                 BNE       NTLKD
000161                 LDA       #'*'                     ;Load LOCKED indicator
000162                 BNE       STUFACS
000163 NTLKD           CMP       #$C3                     ;Unlocked?
000164                 BEQ       ALLOK
000165                 LDA       #'+'                     ;Restricted access but not LOCKED
000166 STUFACS         STA       CATBUF+1
000167 ALLOK           LDA       CTPL                     ;GET NAME LEN
000168                 AND       #$0F
000169                 TAY
000170 CMP1:           LDA       CTPL,Y
000171                 STA       CATBUF+14,Y
000172                 DEY
000173                 BNE       CMP1                     ;TRANSFER FILE NAME TO CATBUF
000174                 LDA       CTPL+$10                 ;GET FILE TYPE
000175                 CMP       #22                      ;22 ALLOWABLE TYPES (0-21) SO FAR
000176                 BCS       FORNFILE                 ;IT'S A FOREIGN FILE
000177                 ASL       A                        ;*2
000178                 ASL       A                        ;*4
000179                 ADC       CTPL+$10                 ;*5
000180                 ADC       CTPL+$10                 ;*6
000181                 TAX                                ;USE AS INDEX
000182 CMP2:           LDA       TYPTB,X
000183                 STA       CATBUF+2,Y
000184                 LDA       #0                       ;ZERO OUT FAC
000185                 STA       FAC,Y
000186                 INX
000187                 INY
000188                 CPY       #6
000189                 BCC       CMP2
000190                 BCS       NOWEOF
000191 *
000192 * DO UNEXPECTED OR OUT-OF-RANGE FILE TYPE
000193 *
000194 FORNFILE        LDX       #$2
000195                 LDA       CTPL+$10                 ;GET FILE TYPE
000196                 AND       #$0F                     ;STRIP HI NIBBLE
000197 FRNLP2          ORA       #$30
000198                 CMP       #$3A                     ;MAKE SURE IT'S A DIGIT
000199                 BCC       FRNLP1
000200                 ADC       #$06                     ;NOW IT'S A HEX CHAR A-F
000201 FRNLP1          STA       FRNTYP+3,X               ;Cover both asses
000202                 STA       PROTYP+3,X
000203                 DEX
000204                 BEQ       FRNFL
000205                 LDA       CTPL+$10
000206                 LSR       A                        ;DO HIGH NIBBLE NOW
000207                 LSR       A
000208                 LSR       A
000209                 LSR       A
000210                 JMP       FRNLP2
000211 FRNFL           LDA       CTPL+$10                 ;GET THE FILE TYPE
000212                 CMP       #$E0                     ;IS IT A PRODOS FILE?
000213                 LDA       PROTYP,Y
000214                 BCS       ISPRO
000215                 LDA       FRNTYP,Y
000216 ISPRO           STA       CATBUF+2,Y
000217                 LDA       #0                       ;ZERO OUT FAC
000218                 STA       FAC,Y
000219                 INY
000220                 CPY       #6
000221                 BCC       FRNFL
000222 NOWEOF          LDA       CTPL+$15                 ;MOVE EOF INTO FAC
000223                 STA       FAC+7
000224                 LDA       CTPL+$16
000225                 STA       FAC+6
000226                 LDA       CTPL+$17
000227                 STA       FAC+5
000228                 JSR       LOUT                     ;OUTPUT IT.
000229                 LDY       #0                       ;TRANSFER NUM INTO CATBUF+50
000230 OEOF            LDA       NUMSTR,Y
000231                 STA       CATBUF+62,Y
```

```
000232                  BEQ       NMREOF
000233                  INY
000234                  BNE       OEOF
000235 NMREOF           LDY       CTPL+$13
000236                  LDA       CTPL+$14
000237                  JSR       GIVAYF
000238                  JSR       FOUT
000239                  LDY       #$FF
000240 CNTLN            INY
000241                  LDA       FBUFFR+1,Y              ;MOVE BLOCK COUNT IN
000242                  BNE       CNTLN                   ;COUNT #OF DIGS
000243                  LDX       #5                      ;5 DIGIT COUNT
000244 MVBCNT           LDA       FBUFFR,Y                ;GET DIG
000245                  STA       CATBUF+8,X
000246                  DEX
000247                  DEY
000248                  BPL       MVBCNT
000249                  LDA       #'0'                    ;FILL REST OF COUNT WITH 0S
000250 ZBC:             DEX
000251                  BMI       CMP3
000252                  STA       CATBUF+9,X
000253                  BPL       ZBC
000254 * NOW PUT IN THE DATES
000255 CMP3:            LDA       CTPL+$22                ;GET DATE LAST MODIFIED
000256                  LDX       CTPL+$21
000257                  LDY       #31
000258                  JSR       GENDATE
000259                  LDA       CTPL+$19
000260                  LDX       CTPL+$18
000261                  LDY       #46
000262                  JSR       GENDATE                 ;CREATE DATE.
000263                  LDA       CTPL+$24                ;GIVE TIME LAST MOD'D
000264                  LDX       CTPL+$23
000265                  LDY       #40
000266                  JSR       GENTIME
000267                  LDA       CTPL+$1B
000268                  LDX       CTPL+$1A                ;GIVE TIME LAST CREATED
000269                  LDY       #55
000270                  JSR       GENTIME
000271                  CLC
000272                  RTS
000273 *
000274 * PRINT THE SUMMING UP INFO, IF POSSIBLE.
000275 *
000276 CEND             LDA       FCB+XRECL,X             ;IS IT A ROOT DIR W/INFO?
000277                  ORA       FCB+XRECL+1,X
000278                  BEQ       CEND2
000279                  LDY       #SUMML                  ;SUMMING MSG LEN.
000280 CEN2:            LDA       SMMSG-1,Y
000281                  STA       CATBUF,Y
000282                  DEY
000283                  BNE       CEN2
000284                  LDA       FCB+XRECL,X             ;GET LOW OF TOTAL BLOCKS
000285                  TAY
000286                  LDA       FCB+XRECL+1,X
000287                  LDX       #53
000288                  JSR       DONUM
000289                  LDX       FCBNDX
000290                  LDA       FCB+XBLKS,X
000291                  TAY
000292                  LDA       FCB+XBLKS+1,X
000293                  LDX       #33
000294                  JSR       DONUM
000295                  LDX       FCBNDX
000296                  LDA       FCB+XRECL,X
000297                  SEC
000298                  SBC       FCB+XBLKS,X
000299                  TAY
000300                  LDA       FCB+XRECL+1,X
000301                  SBC       FCB+XBLKS+1,X
000302                  LDX       #14
000303                  JSR       DONUM
000304                  LDX       FCBNDX
000305                  DEC       FCB+XFLGS,X
000306 CEND2:           INC       FCB+XFLGS,X
000307                  JMP       CBL
000308 FRNTYP           ASC       "TYP=  "
000309 PROTYP           ASC       "PRO=  "
000310 TYPTB:           ASC       "UNKNWN"
000311                  ASC       "BAD   " ;6 CHARS EACH"
```

```
000312                ASC       "PASCOD"
000313                ASC       "PASTXT"
000314                ASC       "TEXT  "
000315                ASC       "PASDTA"
000316                ASC       "BINARY"
000317                ASC       "FONT  "
000318                ASC       "FOTO  "
000319                ASC       "BASIC "
000320                ASC       "DATA  "
000321                ASC       "WPTEXT"
000322                ASC       "SYSTEM"
000323                ASC       "RESERV"
000324                ASC       "RESERV"
000325                ASC       "CAT   "
000326                ASC       "RPSDAT"
000327                ASC       "RPSIDX"
000328                ASC       "AFDISC"
000329                ASC       "ASMOD " "
000330                ASC       "AFRPT "
000331                ASC       "SCNLIB"
000332 *
000333 * ROUTINE TO READ CATALOG INFO FROM SOS
000334 *
000335 RDCAT           LDY       FCBNDX
000336                LDA       FCB,Y                     ;CAT REF NUM
000337                STA       RWRFNM
000338                LDA       FCB+XBUFOFS,Y
000339                CMP       #$FF                      ;END OF ONE DIR BLOCK?
000340                BNE       RDCAT3                    ;NO.
000341                LDA       #5                        ;YES, SKIP JUNK IN BETWEEN DIR BLOCKS
000342                JSR       RDCAT2
000343                BCS       NMCAT                     ;BRANCH IF NO MORE CATALOG
000344 RDCAT3         LDA       #$27
000345 RDCAT2         STA       INBYTES
000346                STA       LENSAV
000347                LDA       #0
000348                STA       INBYTES+1
000349                LDA       #>CTPL
000350                STA       SBFPTR
000351                LDA       #<CTPL
000352                STA       SBFPTR+1
000353                LDY       #RED                      ;READ WITH EOF CHECK
000354                JSR       SETUP
000355                JSR       GOSOS
000356                BEQ       OKCAT
000357                CMP       #SEEOF                    ;WAS IT AN END-OF-FILE ERROR?
000358                BNE       CTR2                      ;NO, BLOW UP
000359 NMCAT:         SEC
000360                RTS
000361 OKCAT:         JMP       UPOFS                     ;CLEAN UP STUFF
000362 CTR2:          JMP       CLTERR
000363                PAGE
000364 *
000365 * HERE IS ROUTINE TO GEN THE DATE FOR EVERY BODY UNDER THE SUN---
000366 *
000367 * ENTER A=HI,X=LO, Y=PT TO PUT DATE IN CATBUF
000368 *  DATE IS(FROM HI TO LO) 7 BITS YEAR, 4 BITS MONTH, 5 BITS DAY
000369 *
000370 GENDATE:       STA       CTPL+1                    ;PRESERVE STUFF
000371                STX       CTPL
000372                STY       CTPL+2
000373                TXA                                 ;COMPUTE DAY
000374                AND       #$1F
000375                TAY                                 ;G2DIGS EXPECTS Y REG
000376                LDA       #3                        ;3 CHAR POSITIONS INTO DATE
000377                JSR       G2DIGS
000378                LDA       CTPL+1                    ;NOW DO THE YEAR
000379                LSR       A
000380                ROR       CTPL                      ;PREP TO DO THE MONTH
000381                TAY
000382                LDA       #6
000383                JSR       G2DIGS
000384                LDA       CTPL                      ;GET THE MONTH
000385                LSR       A
000386                LSR       A
000387                LSR       A
000388                LSR       A
000389                TAY
000390                LDA       #0                        ;AT BEGINNING OF BUF
000391                JSR       G2DIGS
```

```
000392                  LDA       #'/'
000393                  STA       CATBUF+2,Y              ;Y-REG IS SET UP BY G2DIGS
000394                  STA       CATBUF+5,Y
000395                  RTS
000396 *
000397 * GIVE 2 DIGITS SOMEWHERE INTO THE CATBUF
000398 *
000399 G2DIGS           PHA
000400                  JSR       SNGFLT
000401                  JSR       FOUT
000402                  PLA
000403                  CLC                               ;CALC PLACE TO PUT DIGS
000404                  ADC       CTPL+2
000405                  TAY
000406                  LDA       FBUFFR                  ;GET DIG#1
000407                  STA       CATBUF,Y
000408                  STA       CATBUF+1,Y              ;PUT IT BOTH PLACES
000409                  LDA       FBUFFR+1                ;ANOTHER DIG?
000410                  BEQ       AZERH                   ;NO, ZERO FOR HIGH DIGIT
000411                  STA       CATBUF+1,Y
000412                  RTS
000413 AZERH            LDA       #'0'
000414                  STA       CATBUF,Y
000415                  RTS
000416 * ROUTINE TO GENERATE THE TIME FOR THE CATALOG
000417 GENTIME          STX       CTPL                    ;PRESERVE MINUTES
000418                  STY       CTPL+2                  ;POS IN THE LINE
000419                  TAY                               ;PUT HOURS INTO Y-REG
000420                  LDA       #0                      ;POSITION RELATIVE TO (CTPL+2)
000421                  JSR       G2DIGS
000422                  LDA       #$3A                    ;COLON IN THE TIME
000423                  STA       CATBUF+2,Y
000424                  LDY       CTPL
000425                  LDA       #3                      ;NOW PRINT MINUTES
000426                  JMP       G2DIGS
000427 *
000428 * ROUTINE TO OUTPUT A NUMBER TO CATBUF,X
000429 DONUM:           STX       XSAV
000430                  JSR       GIVAYF
000431                  JSR       FOUT
000432                  LDY       #0
000433                  LDX       XSAV                    ;MOVE # IN.
000434 DONUM2:          LDA       FBUFFR,Y
000435                  BEQ       DONUMR
000436                  STA       CATBUF,X
000437                  INY
000438                  INX
000439                  BNE       DONUM2                  ;ALWAYS
000440 DONUMR           RTS
000441 SMMSG:           ASC       'BLOCKS FREE:       BLOCKS USED:       TOTAL BLOCKS:      '
000442 SUMML            EQU       *-SMMSG
000443                  SBTL      "Subroutine Jump Table" "
000444 JMPRTAB          DW        DOPAR
000445                  DW        PTRGET
000446                  DW        MVUP
000447                  DW        MVDWN
000448                  DW        BLTUC
000449                  DW        ERRDIR
000450                  DW        LINGET
000451                  DW        GOTOB
000452                  DW        GETADR
000453                  DW        FNDLNC0
000454                  DW        FNDLIN
000455                  DW        INITCNS
000456                  DW        RESLST-$2000
000457                  DW        NOTNOW
000458                  DW        ERROR
000459                  DW        SERROR
000460                  DW        SCRUNCH
000461                  DW        EXPAND
000462                  DW        FREFAC
000463                  DW        FRENOW
000464                  DW        FRECNOW
000465                  DW        FRESPA
000466                  DW        OPENIT
000467                  DW        GOSOS
000468                  DW        CLSALL
000469                  DW        GIVAYF
000470                  DW        POSINT
000471                  DW        FIN
```

```
000472                  DW        NWSTT
000473                  DW        PTRGT3
000474                  DW        PNTREL                    ;30
000475                  DW        RELPTR2                   ;31
000476                  DW        DATAN
000477                  DW        STRCP
000478                  DW        INPCOM
000479                  DW        LETP2                     ;35
000480                  DW        FOUT                      ;36
000481                  DW        NEWRET
000482                  DW        JUMPDO
000483                  DW        INT
000484                  DW        FBUFFR-$2000
000485                  DW        RESL2-$2000
000486                  DW        SETUP
000487                  DW        SETGO
000488                  DW        CONV2STR
000489                  DW        JMPRTAB
000490
000491 ; ##############################################################################################
000492 ; #   END OF FILE:  CATALOG.TEXT
000493 ; #   LINES      :  484
000494 ; #   CHARACTERS :  19482
000495 ; ##############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 495   CHARACTERS: 20034
|
+-------------------------------------------------------------------------------
```

```
+------------------------------------------------------------------------------
|
|  File    : "BASICEND.TEXT.PRETTY"
|  Created : Tuesday, December 30, 1997               5:14:32 PM
|  Modified: Wednesday, December 31, 1997             4:37:09 PM
|
+------------------------------------------------------------------------------

000001 ; ###########################################################################################
000002 ; #   PROJECT  :  Apple /// Business BASIC 1.3 (6502 Assembly Source Code)
000003 ; #   FILE NAME:  BASICEND.TEXT
000004 ; ###########################################################################################
000005
000006                  SBTL      "SWHGO - JUMPDO" "
000007                  REP       68
000008 *
000009 *               - - - C A U T I O N - - -
000010 *
000011 *  The following section is for jumping to entry points of various
000012 *   routines that MAY be OUTSIDE the realm of the BASIC Interpreter,
000013 *   or being called from the outside world.  If the routine is an
000014 *   EXFN(%). or a PERFORM, it may be not be in BASIC's bank.  Therefore
000015 *   the following code must NOT reside in Bank Switched memory (unless
000016 *   SOS has a utility built in to allow it)!!
000017 *
000018                  REP       68
000019 SWCHGO           PHA                               ;SAVE ACC
000020                  TXA                               ;  AND X-REGISTER
000021                  PHA
000022                  LDA       PASSAREG                ;GET ROUTINE #
000023                  ASL       A                       ;MULTIPLY BY 2 FOR OFFSET INTO TABLE
000024                  TAX                               ;PUT OFFSET INTO X-REG
000025                  LDA       JMPRTAB,X
000026                  STA       JMPER+1                 ;SAVE ROUTINE'S ADDRESS
000027                  LDA       JMPRTAB+1,X             ;  IN JMPER
000028                  STA       JMPER+2
000029                  LDA       BASICBNK
000030                  STA       $FFEF                   ;  (AND THE BANK, TOO)
000031                  CPX       #>SWCHGO-JMPRTAB
000032                  BCS       BCALERR                 ;IS OFFSET OUT OF RANGE OF TABLE?
000033                  PLA                               ;RESTORE X-REG AND ACC
000034                  TAX
000035                  PLA
000036                  JSR       JMPER
000037                  PHA
000038                  LDA       INVBNK
000039                  STA       $FFEF
000040                  PLA
000041                  RTS
000042 BCALERR          JMP       FCERR
000043 JUMPDO           LDA       INVBNK                  ;GET INVOKABLES BANK
000044                  STA       TEMP                    ;  & SAVE IT
000045                  LDA       $FFEF                   ;GET SYSTEM BANK #.
000046                  STA       SAFE+2                  ;  & SAVE THAT TOO
000047                  LDA       TEMP
000048                  STA       $FFEF                   ;SWITCH BANKS.
000049                  LDA       JMPER+2
000050                  AND       #$7F                    ;MASK HIGH BIT OFF
000051                  CLC
000052                  ADC       #$20                    ;$2000 ADJ FOR BANK BOUNDS
000053                  STA       JMPER+2
000054                  JSR       JMPER                   ;THIS DOES A JUMP TO ADDR IN JMPER+1 & +2
000055                  STA       YSAVE
000056                  LDA       SAFE+2                  ;Restore execution bank
000057                  STA       $FFEF
000058                  LDA       SAFE+1                  ;RETURN FOR WHOEVER CALLED US.
000059                  PHA
000060                  LDA       SAFE
000061                  PHA
000062                  LDA       YSAVE
000063                  RTS
000064                  ASC       'TH..TH..TH..THATS ALL, FOLKS!'
000065 ZZZZZ           EQU       *                       ;Last real byte of BASIC
000066                  DO        DEBUG
000067 BASICEND        EQU       $A200+$1559             ;Save $1559 bytes for the Debuggerer
000068                  DS        BASICEND-*
000069                  ELSE
000070 BASICEND        EQU       *                       ;No Debuggerer so extra space saved.
000071                  FIN
000072
```

```
000073 ; ###############################################################################################
000074 ; #   END OF FILE:  BASICEND.TEXT
000075 ; #   LINES      :  66
000076 ; #   CHARACTERS :  3091
000077 ; ###############################################################################################

+-------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 77   CHARACTERS: 3643
|
+-------------------------------------------------------------------------------
```

DTCASMREFORMAT PROGRAM LISTING

```
+--------------------------------------------------------------------------------
|
|  File    : "DTCASMREFORMAT.P"
|  Created : Monday, December 29, 1997               4:18:37 PM
|  Modified: Tuesday, December 30, 1997              6:06:35 PM
|
+--------------------------------------------------------------------------------

000001 { DTCAsmReFormat.p }
000002
000003 { reformat assembly language source listings to look much nicer
000004
000005   apple macintosh mpw shell tool
000006
000007   syntax:  DTCAsmReFormat project-name text-file-1 text-file-2 ...
000008
000009            where project-name is the name of the project that the source files
000010            belong to and which will appear at the top of each reformatted output
000011            file, and text-file-n is the name of an assembly language text file
000012            that needs to be reformatted
000013
000014            note: if project-name starts with "*" then all spaces in the output file
000015                  are replaced by non-breaking spaces, this is done since non-breaking
000016                  spaces cause printing to line up better than regular spaces
000017
000018            output is set of text files with same names as inputted files
000019            but with ".pretty" suffix and each output file begins with the
000020            name of the file and ends with the number of lines and characters
000021            in the file
000022
000023   example: DTCAsmReFormat "My Best Project" Foobar Frodor "Christmas Tree"
000024
000025   david t craig - 29 dec 1997 - 71533.606@compuserve.com }
000026
000027 program reformat_asm_source;
000028
000029 uses Memtypes, Quickdraw, OSIntf, ToolIntf, PackIntf, { Standard Includes}
000030      CursorCtl,              { for the spinning cursor}
000031      Signal,                 { to handle command-period}
000032      PasLibIntf,             { for standard I/O, etc.}
000033      IntEnv;                 { for argV and argC}
000034
000035 {$r+}
000036
000037 const kPgmName       = 'Assembly Language Reformatter';
000038       kPgmVersion    = '1.0.0';
000039       kPgmDate       = '29 December 1997';
000040       kPgmAuthor     = 'David T. Craig -- 71533.606@compuserve.com -- Santa Fe, New Mexico USA';
000041
000042       kComment1      = '*'; { at start of a line }
000043       kComment2      = ';'; { at start or end of a line }
000044
000045       kWidth_Label   = 15;
000046       kWidth_Opcode  =  9;
000047       kWidth_Operand = 25;
000048
000049       kSpace         = chr($20);
000050       kSpaceNoBreak  = chr($ca);
000051       kQuote1        = '"';
000052       kQuote2        = ''';     { ' }
000053
000054       kSuffix        = '.pretty';
000055
000056 type  tStrBig        = string[255];
000057       tStrSmall      = string[131];
000058
000059 var   project        :tStrBig;
000060
000061       argi           : integer;
000062       argn           : tStrBig;
000063       argf           : text;
000064       argm           : tStrBig;
000065       argg           : text;
000066
000067       nobrkspace     : boolean;
```

```
000068
000069      s             : tStrBig;
000070      e             : integer;
000071      cnt_lines     : longint;
000072      cnt_chars     : longint;
000073
000074      plabel        : tStrSmall;
000075      popcode       : tStrSmall;
000076      poperand      : tStrSmall;
000077      pcomment      : tStrSmall;
000078
000079      {-------------------------------------------------------------------------------}
000080
000081      procedure normalize (var _s: tStrBig);
000082
000083      var i: integer;
000084
000085      begin
000086        for i := 1 to length(_s) do
000087          if _s[i] < kSpace then
000088            _s[i] := kSpace;
000089      end;
000090
000091      {-------------------------------------------------------------------------------}
000092
000093      function is_blank_line (_s: tStrBig): boolean;
000094
000095      var blank: boolean;
000096          i    : integer;
000097
000098      begin
000099        blank := true;
000100        normalize(_s);
000101        for i := 1 to length(_s) do
000102          if _s[i] <> kSpace then
000103            blank := false;
000104        is_blank_line := blank;
000105      end;
000106
000107      {-------------------------------------------------------------------------------}
000108
000109      procedure parse_line(_s: tStrBig;
000110                           var _label,_opcode,_operand,_comment: tStrSmall);
000111
000112      var done: boolean;
000113          i,j : integer;
000114          w   : tStrSmall;
000115
000116          procedure get_next_word (var _word: tStrSmall);
000117
000118          var done: boolean;
000119
000120          begin
000121            {
000122            writeln(diagnostic,'"',_s,'"');
000123            writeln(diagnostic,' 123456789-123456789-123456789-123456789-123456789-123456789-');
000124            }
000125
000126            _word := '';
000127            done  := false;
000128
000129            if i <= length(_s) then begin
000130              if _s[i] = kSpace then begin
000131                while not(done) do begin
000132                  _word := concat(_word,_s[i]);
000133                  i := i + 1;
000134                  if i > length(_s) then
000135                    done := true
000136                  else begin
000137                    if _s[i] <> kSpace then
000138                      done := true;
000139                  end;
000140                end;
000141              end else begin
000142                while not(done) do begin
000143                  _word := concat(_word,_s[i]);
000144                  i := i + 1;
000145                  if i > length(_s) then
000146                    done := true
000147                  else begin
```

```
000148                    if _s[i] = kSpace then
000149                      done := true;
000150                  end;
000151                end;
000152              end;
000153            end;
000154
000155            {
000156            writeln(diagnostic,'i = ',i:3,'     WORD = "',_word,'"');
000157            }
000158          end;
000159
000160        begin
000161          _label := '';  _opcode := '';  _operand := '';  _comment := '';
000162
000163          { FRMEV3:   PLA       ;GET CURRENT OP BACK AND LOOP. }
000164          { label     opcode    comment }
000165
000166          { DOIT      BMI      ASTRNG    ;IT'S A STRING. }
000167          { label     opcode   operand   comment }
000168
000169          {   SEC }
000170          {   opcode }
000171
000172          if _s <> '' then begin
000173            if s[1] in [kComment1,kComment2] then begin
000174              _comment := _s
000175            end else begin
000176              i := 1;
000177              get_next_word(w); { "FRMEV3:" or "  " }
000178              if w <> '' then begin
000179                if w[1] <> kSpace then begin
000180                  _label := w;
000181                  get_next_word(w); { "   " }
000182                end;
000183                if w <> '' then begin
000184                  get_next_word(w); { "PLA" }
000185                  if w <> '' then begin
000186                    if w[1] = kComment2 then begin
000187                      for j := i-length(w) to length(_s) do
000188                        _comment := concat(_comment,_s[j]);
000189                    end else begin
000190                      _opcode := w;
000191                      get_next_word(w); { "       " }
000192                      if w <> '' then begin
000193                        get_next_word(w); { ";GET CURRENT OP BACK AND LOOP." }
000194                        if w <> '' then begin
000195                          if w[1] = kComment2 then begin
000196                            for j := i-length(w) to length(_s) do
000197                              _comment := concat(_comment,_s[j]);
000198                          end else begin
000199                            if w[1] in [kQuote1,kQuote2] then begin
000200                              for j := i-length(w) to length(_s) do
000201                                _operand := concat(_operand,_s[j]);
000202                              if not(_operand[length(_operand)] in [kQuote1,kQuote2]) then
000203                                _operand := concat(_operand,w[1]);
000204                            end else begin
000205                              _operand := w;
000206                              get_next_word(w);
000207                              if w <> '' then
000208                                for j := i-length(w) to length(_s) do
000209                                  _comment := concat(_comment,_s[j]);
000210                            end;
000211                          end;
000212                        end;
000213                      end;
000214                    end;
000215                  end;
000216                end;
000217              end;
000218            end;
000219          end;
000220        end;
000221
000222        {-------------------------------------------------------------------------------}
000223
000224        function trim_trailing (_s: tStrBig): tStrBig;
000225
000226        var done: boolean;
000227
```

```
000228      begin
000229        done := false;
000230
000231        while not(done) do begin
000232          if length(_s)=0 then
000233            done := true
000234          else begin
000235            if _s[length(_s)]=kSpace then
000236              delete(_s,length(_s),1)
000237            else
000238              done := true;
000239          end;
000240        end;
000241
000242        trim_trailing := _s;
000243      end;
000244
000245      {-------------------------------------------------------------------------------}
000246
000247      function trim_leading (_s: tStrBig): tStrBig;
000248
000249      var done: boolean;
000250
000251      begin
000252        done := false;
000253
000254        while not(done) do begin
000255          if length(_s)=0 then
000256            done := true
000257          else begin
000258            if _s[1]=kSpace then
000259              delete(_s,1,1)
000260            else
000261              done := true;
000262          end;
000263        end;
000264
000265        trim_leading := _s;
000266      end;
000267
000268      {-------------------------------------------------------------------------------}
000269
000270      function use_nobreak_spaces (_s: tStrBig): tStrBig;
000271
000272      var i: integer;
000273
000274      begin
000275        for i := 1 to length(_s) do
000276          if _s[i] = kSpace then
000277            _s[i] := kSpaceNoBreak;
000278
000279        use_nobreak_spaces := _s;
000280      end;
000281
000282      {-------------------------------------------------------------------------------}
000283
000284      procedure write_info (_project: tStrBig; _header: boolean);
000285
000286      const k = '##########################################';
000287
000288      begin
000289        case _header of
000290          true:
000291            begin
000292              writeln(argg,'; ',k,k);
000293              writeln(argg,'; #   PROJECT  :  ',_project);
000294              writeln(argg,'; #   FILE NAME:  ',argn);
000295              writeln(argg,'; ',k,k);
000296              writeln(argg);
000297            end;
000298          false:
000299            begin
000300              writeln(argg);
000301              writeln(argg,'; ',k,k);
000302              writeln(argg,'; #   END OF FILE:  ',argn);
000303              writeln(argg,'; #   LINES      :  ',cnt_lines:0);
000304              writeln(argg,'; #   CHARACTERS :  ',cnt_chars:0);
000305              writeln(argg,'; ',k,k);
000306            end;
000307        end;
```

```
000308       end;
000309
000310       {-------------------------------------------------------------------------------}
000311
000312 begin
000313   writeln(diagnostic,kPgmName,' ',kPgmVersion,' [',kPgmDate,']');
000314   writeln(diagnostic,'Written by ',kPgmAuthor);
000315   writeln(diagnostic);
000316
000317   if argc < 3 then
000318     writeln(diagnostic,'WARNING: You need to specify at least one text file.  Try again :-)')
000319   else begin
000320     project := argv^[1]^;
000321
000322     nobrkspace := false;
000323     if length(project) > 0 then begin
000324       if project[1] = '*' then begin
000325         nobrkspace := true;
000326         delete(project,1,1);
000327       end;
000328     end;
000329
000330     writeln(diagnostic,'Project Name       : ',project);
000331     writeln(diagnostic,'Non-Breaking Spaces: ',nobrkspace);
000332     writeln(diagnostic);
000333
000334     argi := 2;
000335
000336     while argi<argc do begin
000337       argn := argv^[argi]^;
000338
000339       writeln(diagnostic,'Processing file "',argn,'" ...');
000340
000341       reset(argf,argn); e := ioresult;
000342
000343       if e <> 0 then
000344         writeln(diagnostic,'### ERROR ',e:0,': Opening file "',argn,'" failed.')
000345       else begin
000346         argm := concat(argn,kSuffix);
000347         rewrite(argg,argm); e := ioresult;
000348
000349         if e <> 0 then
000350           writeln(diagnostic,'### ERROR ',e:0,': Creating file "',argm,'" failed.')
000351         else begin
000352           write_info(project,true);
000353
000354           {++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
000355           cnt_lines := 0;
000356           cnt_chars := 0;
000357
000358           while not(eof(argf)) do begin
000359             readln(argf,s); e := ioresult;
000360
000361             cnt_lines := cnt_lines + 1;
000362
000363             if e<>0 then
000364               writeln(diagnostic,'### ERROR ',e:0,' at line ',cnt_lines:0,' "',s,'"')
000365             else begin
000366               normalize(s);
000367
000368               { FRMEV3:   PLA       ;GET CURRENT OP BACK AND LOOP. }
000369               { label     opcode    comment }
000370
000371               {  BMI      ASTRNG    ;IT'S A STRING. }
000372               {  opcode   operand   comment }
000373
000374               { DOIT      BMI      ASTRNG    ;IT'S A STRING. }
000375               { label     opcode   operand   comment }
000376
000377               { ;GET CURRENT OP BACK AND LOOP. }
000378               { comment }
000379
000380               { *GET CURRENT OP BACK AND LOOP. }
000381               { comment }
000382
000383               if is_blank_line(s) then s := ''; { do this in case line has just spaces in it }
000384
000385               parse_line(s,plabel,popcode,poperand,pcomment);
000386
000387               plabel   := trim_leading(plabel  );   plabel   := trim_trailing(plabel  );
```

```
000388              popcode  := trim_leading(popcode );   popcode  := trim_trailing(popcode );
000389              poperand := trim_leading(poperand);   poperand := trim_trailing(poperand);
000390              pcomment := trim_leading(pcomment);   pcomment := trim_trailing(pcomment);
000391
000392              if (plabel<>'') or (popcode<>'') or (poperand<>'') or (pcomment<>'') then begin
000393                {
000394                writeln(diagnostic,'LABEL   = "',plabel,'"');
000395                writeln(diagnostic,'OPCODE  = "',popcode,'"');
000396                writeln(diagnostic,'OPERAND = "',poperand,'"');
000397                writeln(diagnostic,'COMMENT = "',pcomment,'"');
000398                }
000399
000400                if (plabel='') and (popcode='') then
000401                  s := pcomment
000402                else begin
000403                  while length(plabel )<kWidth_Label    do plabel   := concat(plabel  ,kSpace);
000404                  while length(popcode)<kWidth_Opcode   do popcode  := concat(popcode ,kSpace);
000405                  while length(poperand)<kWidth_Operand do poperand := concat(poperand,kSpace);
000406                  s := concat(plabel,kSpace,popcode,kSpace,poperand,kSpace,pcomment);
000407                end;
000408
000409                s := trim_trailing(s);
000410
000411                if nobrkspace then s := use_nobreak_spaces(s);
000412
000413                cnt_chars := cnt_chars + length(s);
000414                writeln(argg,s);
000415              end else begin
000416                cnt_chars := cnt_chars + 1;
000417                writeln(argg);
000418              end;
000419            end;
000420          end; {while}
000421          {++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++}
000422
000423          write_info('',false);
000424
000425          close(argg);
000426        end;
000427
000428        close(argf);
000429      end;
000430
000431      argi := argi + 1;
000432    end; {while}
000433  end;
000434
000435  writeln(diagnostic);
000436  writeln(diagnostic,'That''s all folks!');
000437 end.
000438
000439 { finis }

+--------------------------------------------------------------------------------
|
|  THAT'S ALL FOLKS!      LINES: 439   CHARACTERS: 13956
|
+--------------------------------------------------------------------------------
```

###