AMPER FRE SPLITTER

Amper Fre Splitter

by Gary C. James
1528 W. 130th St.
Brunswick, OH 44212

Have you ever written a high resolution graphics program in Applesoft BASIC, and had to use the Hi-Res graphics page two because your program was too long and overlapped page one? Or possibly you wanted to include a machine language subroutine with your program and could not find a place in memory to locate it. Or worse, the machine language subroutine was not written to be relocatable and had to reside in the same memory locations as your Applesoft program. If you've found yourself in any of these unfortunate situations, **& FRE** is the answer to your problems.

WHY & FRE

& FRE (pronounced Amper Free) is a utility program that allows you to free-up any area of your Apple's memory for whatever reason, even though that area of memory has become unavailable for other uses (because it is being used to contain your Applesoft program). The utility was created to satisfy my need to use the Apple's mixed text and graphics display mode — normally usable only with Hi-Res graphics page one — while running very large Applesoft programs. I was writing an Applesoft-based utility for developing graphics Shape Tables, when it became clear that the program would not fit in the limited memory space below Hi-Res page one. After several weeks spent writing the program and several days typing it in and debugging, I discovered that the program could never run properly because it was too large. The Apple's memory-mapped Hi-Res graphics display overlapped some of the memory it occupied. There I was with a really useful program. Writing it had occupied most of my spare time for almost a month, and all I had to show was a program that kept inviting the Apple's graphics routines out to lunch — on **it**! Well, that was too much to passively accept. I decided that no matter what it took, I was going to get the program running without rewriting it. The result was & FRE.

LOADING AND RUNNING

& FRE is written in assembly language to produce an object code program that operates extremely fast on the Applesoft program to be "treated."

To load and run & FRE you must first have an Apple II, Apple II Plus, Apple IIe, or Franklin ACE with at least 48K of memory and Applesoft. **For instructions on how to enter machine language directly into memory, see the Letters section of this issue.** When you have typed the program into memory, it can be saved on disk with the command:

BSAVE FRE.CODE, A$9102, L$477

Because the routine is located just below the DOS file buffers, if the DOS MAXFILES setting has been changed to a value greater than the default value of three, this can cause DOS to overwrite the routine and destroy it. **Make sure that MAXFILES is set to three.**

A similar fate could result if somehow after & FRE is loaded, the BASIC HIMEM pointer gets altered and allows string variables to overwrite the routine. If you are unsure of the values of these parameters, to be safe, before using & FRE, boot a disk to initialize the DOS and BASIC pointers to their proper values. If the aforementioned precautions have been observed, simply type **BRUN FRE.CODE** to load and initialize the routine. You will know when & FRE is installed when the program title is displayed and the Applesoft prompt character returns to view.

You may now load the BASIC program you want to modify or continue using the Apple as normal, since & FRE waits patiently and is inactive until it is activated. **For reasons that I will describe further on, let me caution you here that the program you intend to modify should only be a LOADed copy of a program that you have saved on disk. Never treat a program with & FRE that does not have an unmodified disk back-up copy that can be used if further program changes are ever needed. And since nobody writes bug-proof code, changes should always be anticipated.**

After you have selected and LOADed the BASIC program to be modified, the first step in using & FRE is to prepare the program by inserting a dummy program line zero. Because there can only be one line zero in a BASIC program, if another line zero already exists, it must be renumbered to allow room for the new line. To create the dummy line zero simply enter

```
0 ::::::::::::::::::::
```

That's a zero followed by exactly twenty-one colons. When you have prepared the program with the new line zero, invoke the & FRE utility by simply entering:

& FRE Address , Length

where Address and Length are decimal numbers that define the starting location and size of the memory area that you want to free-up. For example:

& FRE 8192,8192

would create a Free Zone out of the eight-thousand byte area used by Hi-Res graphics screen number one. This example can be used for running a very large graphics program without fear of overlapping the program into the graphics memory.

When & FRE has finished modifying your program, a message is printed to tell you at what line number the program was split and moved.

However, if your BASIC program was already small enough so as not to interfere with the specified Free Zone, nothing will be moved and the NOT WITHIN RANGE message will be displayed.

I should also mention that & FRE has an additional feature that I call Failsafe. Failsafe is placed in effect during initialization and automatically links & FRE with a previously loaded Ampersand utility. If you enter any **&** command other than "FRE", control will be passed to the previously enabled ampersand command handler.

LISTING/RUNNING

Now that your Applesoft program has been treated by & FRE, if you try to list the program now, immediately following the split, the program shows something other than colons in the dummy line zero. Also, the program will list only up to the line that was **two program lines** ahead of the line number displayed in the message. Don't worry though. The program is supposed to look like this now. Later when the program is run, the POKE commands that mysteriously appeared in line zero will repair the apparently damaged program so that it can be listed in its entirety. **Be aware however, that the newly modified program must NOT be saved to disk after it has been run and repaired.**

The program should be saved to disk — under a **different name** than the original unmodified copy — in its unrepaired form immediately following the & FRE treatment. If you do this, your program will always load and run perfectly with no indication that it has undergone a major transformation. This procedure may seem strange, but it is absolutely essential, since a repaired copy will almost certainly be destroyed by BASIC following a program load.

The type of modification made to your program is one reason why I told you earlier never to treat a program with & FRE unless you have an unmodified back-up copy.

The other reason is that once a program has been modified by & FRE, it becomes a RUN ONLY copy that cannot be edited in any way, including adding or deleting program lines. If you try to edit the program, the BASIC line editor routine gets very confused and will usually destroy your program in the process of editing it.

After modification, any subsequent changes you wish to make should be made to another copy of the original program, and again saved under a name that is different from the original. Saving an & FRE treated program is accomplished in the same manner as for a normal untreated Applesoft program.

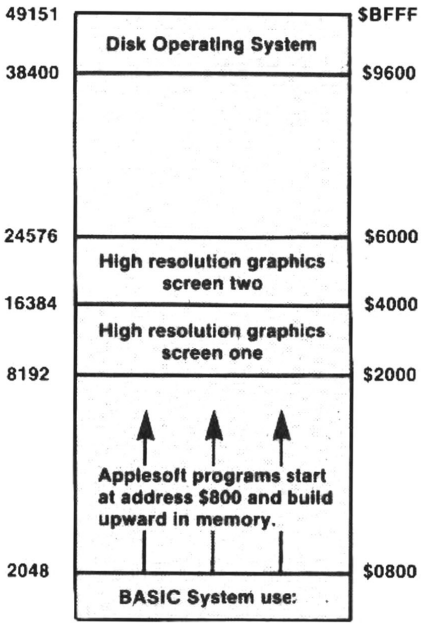
THE PROBLEM

Now that you know how to use & FRE, let me try to explain the technical details that led to its creation.

Normally there are about six thousand bytes between the memory address where Applesoft programs are loaded, and the starting address of Hi-Res graphics page one **(see Figure 1)**. For a Hi-Res graphics program that will use page one, both the program and the numeric variables and arrays it creates must fit into this limited area.

Applesoft programs are usually loaded from disk into the Apple's memory starting at address $800. (The dollar sign is used to indicate a number being expressed in hexadecimal form; i.e., base 16.) When the program has been loaded into memory starting at this address, DOS adjusts the BASIC LOMEM pointer to point to the address just beyond the end of the program. Subsequently, running the program will cause any newly created var-

iables to be built up in memory between the starting address pointed to by LOMEM, and the upper memory limit pointed to by HIMEM; the only exception being string variables that start at HIMEM and build downward in memory.



Apple RAM Memory Map (48K)

Figure 1

What this means is that programs that create large arrays or many variables, would be likely to build up data that overlaps into the Hi-Res screen memory, and any data residing there will be destroyed when graphics are displayed. Therefore, if you write graphics programs using Hi-Res page one (the only one that allows direct easy use of a mixed text and graphics mode), your programs are limited to a length of less than six thousand bytes out of the Apple's 48K maximum. Remember also that if you include operating instructions as part of your program, each page of text could contain as many as 960 characters — almost one sixth of the available memory.

The solution to this restriction is to free-up more memory for the program by relocating either the program and variable storage area, or the variable storage area alone.

PUTTING A TWO-POUND PROGRAM INTO A ONE-POUND MEMORY

If the numeric variable storage area is relocated above the graphics screen, the free memory below can be used exclusively for program storage. This technique is highly recommended when using Hi-Res graphics because it allows for larger programs to be loaded, and prevents the Graphics routines from destroying the string and numeric variables that may otherwise grow and cross into the mapped graphics memory.

Relocating the variable storage area is easily accomplished with Applesoft's **LOMEM** command. The LOMEM command sets the contents of the LOMEM pointer to the user-specified address value that is greater than the ending address of the area to be preserved; for Hi-Res graphics programs, that could be the end of Hi-Res page one. After changing LOMEM, the program creates new variables that are built up and down in memory between the LOMEM and HIMEM addresses, thus avoiding the free area **(Figure 2)**.

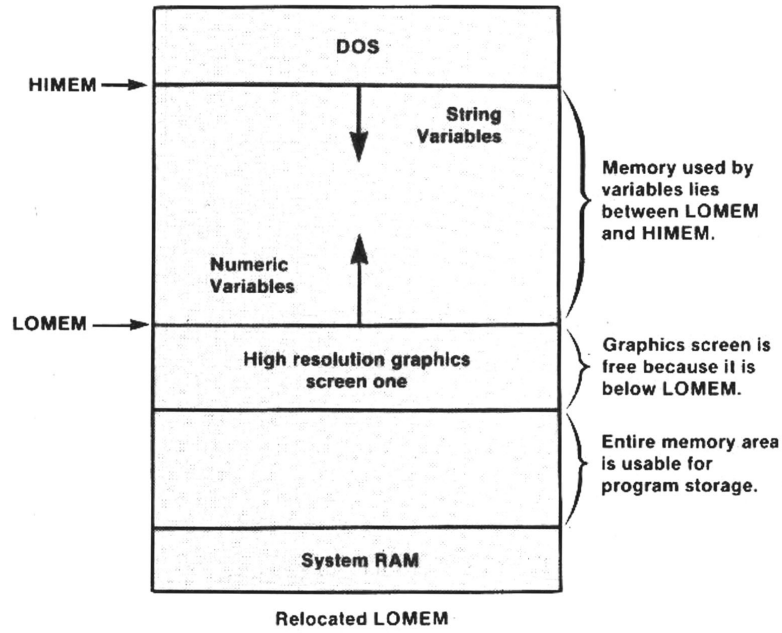


Figure 2

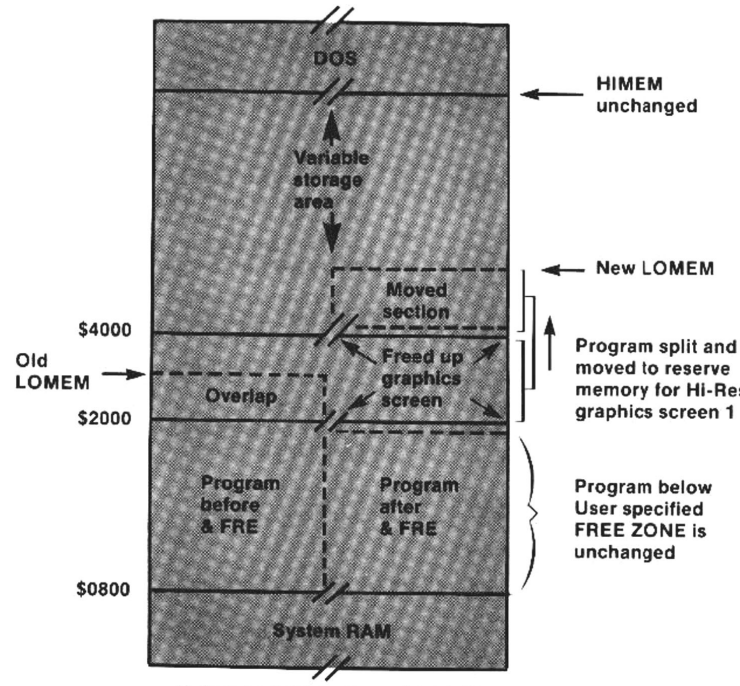
Although this method protects the program from accidental destruction of variables and frees up some memory that may be needed by a larger program, the available memory is still limited to about six thousand bytes.

RELOCATING PROGRAMS

An alternate method to free-up more memory for the program used by many programmers both commercially and at home, is to relocate both the variable storage area and the entire BASIC program. I mentioned earlier that Applesoft programs are usually loaded into memory starting at address $800. This, however, is only the default address established by BASIC during the cold start initialization following a DOS boot. Any address can be substituted and will result in making the programs load into memory at the new address. Placing your program in memory starting just above Hi-Res screen one allows a program that is more than twice as large to be loaded and run.

Relocating the memory address where a BASIC program will be loaded — unlike relocating the variable storage area alone — cannot be accomplished with a simple BASIC command like LOMEM. To do this, the values stored in the page zero, program BEGINNING and END address pointers must be altered to point to the relocation target address, and the target address plus one, respectively. Also, the first two bytes of the target address, and the preceding byte, must be cleared to a zero value. With this done, a DOS LOAD or RUN command will automatically load the disk program file into memory at the new address and adjust LOMEM to a point just beyond the end of the program. Like before, the program's variables will build up and down between LOMEM and HIMEM.

Although this typical approach may provide you with more memory for your program



Before and after views of program modified by & FRE

and may be adequate for many applications, it is very wasteful. The six thousand bytes below Hi-Res page one will be left unused, yet a very large program may still run out of memory.

Therefore, neither of these two popular techniques for overcoming memory utilization conflicts is either totally adequate or designed to make efficient use of the Apple's memory. The only method that is specifically designed to overcome the problem will now be discussed.

SPLITTING YOUR PROGRAMS

By splitting your program into two pieces, it can be made to work around the limitations of memory-mapped graphics. **A large graphics program can be modified to make use of the memory both above and below Hi-Res screen one if it is loaded into memory, then split and moved at the point where it overlaps the screen (Figure 3).**

Even small non-graphic programs can benefit from this technique if they are split and moved to allow a machine language program or subroutine to be loaded into memory that was previously occupied by the program itself.

So, for those of you who write large graphics programs or make use of machine language subroutines, efficient utilization of your Apple's memory can only be obtained by sectioning your Applesoft program to fit into the available memory space. Only & FRE can edit your program in this way.

HOW IT WORKS

For those of you who are interested in the inner workings of the utility, the following section will describe the program operation — both in its overall concept and at the assembly source code level.

To start, let's briefly discuss the operation and structure of Applesoft BASIC. You, the programmer, build an Applesoft program by combining, in program lines, sequences of commands that direct the computer to perform a specific task. In order for you to accomplish this in the least difficult way, your program's commands are expressed to the Apple in plain English. Unfortunately, computers have no comprehension of language beyond the simple binary patterns that all programs must ultimately become. This means that part of the time the Apple spends running your program is in translating (interpreting) your English instructions into binary commands that it can then process. It is this necessary interpretive step inherent in BASIC that & FRE takes advantage of in order to modify your program in such a way that it will continue to execute properly.

If you enter the following program line:

10 HOME:PRINT "HELLO"

Applesoft converts this to the following machine readable code sequence:

00 10 08 0A 00 97 3A BA 22 48 45 4C 4C 4F 22 00 00 00

LINE NUMBERS

The first two zeros of the converted line are the single-byte Applesoft End-of-Line delimiter character which separates one program line from another. When Applesoft is interpreting the program at run time, the End-of-Line character flags BASIC to go to the next line to continue running the program. The four hex digits that follow are the program lines' **Link Pointer field**. The Link Pointer contains a 16-bit address that points to the Link Pointer field of the next program line. (In the above example the next line is located at memory address $0810.)

The next section is the **Line Number field**. This is also a 16-bit number, but it holds the value of this particular program line number. Both the Link Pointer and Line Number fields are formatted in the standard 6502 addressing configuration of low byte first, and then the high byte.

PROGRAM LINES

The next series of numbers make up the actual body of the program line. These are the coded results of what you typed into the Apple's keyboard.

The first number, 97, is the tokenized equivalent of the Applesoft HOME command. (The complete list of Applesoft tokens is available on page 121 of the **Applesoft Reference Manual.**) The 3A following it is the 7-bit ASCII number for the COLON command. The number BA, also a token, represents the PRINT command. (PRINT is the only Applesoft command that is also available as a keyboard macro. Instead of typing PRINT, simply type a question mark in its place.) The numbers 22, 48, 45, 4C, 4C, 4F and 22 that follow, are the ASCII numbers that represent the letters, including quotation marks, in "HELLO".

& FRE IN ACTION

Now that you understand the basics of BASIC, let's discuss the operation of & FRE. When & FRE modifies your program, it must do a lot more than just split and move what is present in the specified Free Zone. It must rewrite specific parts of the moved and unmoved sections to fool BASIC into thinking they are still a single contiguous program. In addition, & FRE introduces a protection scheme to guard against the almost certain destruction of your program, after loading, by BASIC's own line editor.

For the following discussion, please refer to the & FRE Assembly Source I sting. Program lines 108 through 148 form the initialization portion of the routine. INIT sets the & FRE internal Fail-safe jump vector equal to the previous **&** command handler address; sets the **&** command vector to & FRE; prints the program title page; adjusts HIMEM to point below & FRE if necessary; and jumps to Applesoft at the warm-start entry point.

Lines 183 through 187 examine the ampersand token and pass control to the Fail-safe routine if the **FRE** token is not present.

Lines 189 through 214 read the & FRE ADDRESS and LENGTH parameters and convert them to integer numbers.

Lines 219 through 250 establish the end of the second line of BASIC + 16 as the lowest allowable start of a Free Zone, and cause the NOT WITHIN RANGE message to be printed if the user-specified Free Zone starting address is lower.

Lines 253 through 265 print the RANGE and ERROR messages.

Lines 271 through 281 print the RANGE message if the specified Free Zone starting address is greater than the end address of the last line of the BASIC program.

Lines 286 through 306 exit the routine and print the ERROR message if line 0 of the target program is missing or improperly formatted.

Lines 317 through 366 scan the BASIC program to find the last program line whose end address, plus 16, is lower than the starting address of the Free Zone. The 16 extra bytes allow a short transparent program line to be appended to the end of the last unmoved line, forcing a programmed link to the moved lines.

Lines 372 through 464 split and move up in memory only the portion of the program that is within and beyond the Free Zone. The program is moved starting from the beginning address of the first program line that extend: into the Free Zone, through the end addres: of the last program line.

Lines 467 through 526 append to the end o the last unmoved line, a transparent progran line that forces a programmed GOTO to th first of the moved lines. The transparent lin has the same Link Field and Line Numbe values as the last unmoved line. The com mand portion of the line is a simple GOT with the moved line's number as the target.

Lines 533 through 573 adjust the las unmoved line's Link Pointer to point to th new address of the next (moved) line, an adjust the Link Pointers of the moved lines t correct for their new memory addresses.

Finally, lines 579 through 677 alter the for matted line 0 of your program by inserting th POKE commands that restore the Link Point er address of the second last unmoved line This is the protection that prevents the Apple soft line editor from seeing the modified pro gram and trying to "fix" it.

After your program has been modified by FRE, saving it to disk will also save whatever present in the Free Zone. This is a convenien way to save a machine language routine o Hi-Res picture as part of the BASIC program

```
SOURCE FILE: FRE.S
0000:             1 ;
0000:             2 ;
0000:             3 ********************************
0000:             4 *                              *
0000:             5 *   APPLESOFT  & FRE  UTILITY  *
0000:             6 *                              *
0000:             7 *              BY              *
0000:             8 *                              *
0000:             9 *         GARY C. JAMES        *
0000:            10 *        BRUNSWICK, OHIO       *
0000:            11 *                              *
0000:            12 *  (C) 1983 MICROSPARC, INC.   *
0000:            13 *    DOS TOOLKIT ASSEMBLER     *
0000:            14 ********************************
0000:            15 ;
0000:            16 ;
0000:            17 ;
0000:            18 ;    THE FOLLOWING AMPERSAND INITIATED ROUTINE ALLOWS THE USER TO FREE-UP
0000:            19 ; A SECTION OF MEMORY THAT HAS BECOME UNAVAILABLE DUE TO AN APPLESOFT
0000:            20 ; PROGRAM THAT IS PRESENT IN THE DESIRED AREA OF MEMORY
0000:            21 ;
0000:            22 ;    THE ROUTINE OPERATES BY EXAMINING THE RESIDENT APPLESOFT PROGRAM,
0000:            23 ; THEN MOVES WHAT IS PRESENT WITHIN AND BEYOND THE SPECIFIED
0000:            24 ; FREE ZONE.
0000:            25 ;
0000:            26 ; THE COMMAND FORMAT IS:
0000:            27 ;
0000:            28 ;    & FRE X,Y
0000:            29 ;
0000:            30 ; WHERE X = DECIMAL STARTING ADDRESS OF MEMORY TO FREE-UP, AND Y = DECIMAL
0000:            31 ; LENGTH OF FREE ZONE.
0000:            32 ;
0000:            33 ;
0000:            34 ; TO USE &FRE, YOUR BASIC PROGRAM MUST FIRST BE PROPERLY INITIALIZED.
0000:            35 ; THIS IS DONE BY CREATING A DUMMY PROGRAM LINE ZERO THAT CONTAINS 21 COLONS.
0000:            36 ;
0000:            37 ;         0 :::::::::::::::::::::
0000:            38 ;
0000:            39 ; WHEN THIS IS DONE, RUN &FRE AS DESCRIBED ABOVE TO PRODUCE THE SPLIT PROGRAM
0000:            40 ;
0000:            41 ; NOTE:
0000:            42 ;  THE DUMMY LINE IS USED BY &FRE TO CREATE TWO POKE COMMANDS THAT ARE
0000:            43 ;  NEEDED TO FIX THE PROGRAM SO IT WILL RUN PROPERLY WHEN LOADED FROM DISK.
0000:            44 ;  ALSO, AFTER RUNNING &FRE, THE PROGRAM WILL ONLY LIST UP TO THE 2'ND LAST
0000:            45 ;  UNMOVED PROGRAM LINE.  THIS IS OK!  THE LINE ZERO POKE COMMANDS WILL FIX
0000:            46 ;  THIS WHEN THE PROGRAM IS LATER LOADED AND RUN.
0000:            47 ;  DO NOT RUN THE TREATED PROGRAM BEFORE SAVING IT TO DISK.  THE DISK COPY
0000:            48 ;  MUST NOT '''' BE SAVED AFTER THE POKE COMMANDS HAVE FIXED IT.  TO LOAD
0000:            49 ;  AND RUN PROPERLY, THE PROGRAM MUST BE SAVED TO DISK IMMEDIATELY AFTER
0000:            50 ;  TREATMENT WITH &FRE.
0000:            51 ;
0000:            52 ;
0000:            53 ;
0000:            54 ;
3000:            55 ;
0000:            56 ;
0000:            57 ;  APPLESOFT AND SYSTEM ENTRY POINTS
0000:            58 ;
0000:            59 ;
0067:            60 PRGBEG  EQU  $67        ;PNTR TO BEGINNING OF PROGRAM
0069:            61 VARBGT  EQU  $69        ;PNTR TO VARIABLE STORAGE
0000:            62 OFFSET  EQU  $0B        ;TEMP INDIRECT PNTR
0073:            63 HIMEM   EQU  $73        ;APPLESOFT HIMEM POINTER
0094:            64 HIGHDS  EQU  $94        ;BLK XFER DEST
0096:            65 HIGHTR  EQU  $96        ;BLK XFER END ADR
0098:            66 LOWTR   EQU  $9B        ;BLK XFER BEG ADR
00AF:            67 PRGEND  EQU  $AF        ;PNTR TO END OF PROGRAM
00B1:            68 CHRGET  EQU  $B1        ;INC TXTPTR
00BB:            69 TXTPTR  EQU  $BB        ;APPLESOFT INTERPRETER PNTR
00A0:            70 INTLO   EQU  $A0        ;CONVERTED 2 BYTE INTEGER VALUE
0100:            71 FBUFR   EQU  $100       ;FOUT STRING BUFFER
C051:            72 TEXT    EQU  $C051      ;DISPLAY TEXT MODE
C052:            73 NOMIX   EQU  $C052      ;NOT MIXED MODE
C054:            74 PAGE1   EQU  $C054      ;DISPLAY PRIMARY PAGE
E003:            75 ASWARM  EQU  $E003      ;APPLESOFT WARM START
E10C:            76 AYINT   EQU  $E10C      ;CONVERT FAC TO INT
E2F2:            77 GIVAYF  EQU  $E2F2      ;CONVERT INT TO FAC
ED34:            78 FOUT    EQU  $ED34      ;CONVERT FAC TO ASCII STRING @ $100-$110
D393:            79 BLTU    EQU  $D393      ;MOVE PROGRAM BLOCK
D66C:            80 CLEAR   EQU  $D66C      ;CLEAR VARIABLES AND STACK
DB3A:            81 STROUT  EQU  $DB3A      ;PRINT STRING AT Y,A TILL 0
DD67:            82 FRMNUM  EQU  $DD67      ;CONVERT PROGRAM TEXT TO FAC
DEBE:            83 CHKCOM  EQU  $DEBE      ;GET COMMA FROM TXTPTR
FBDD:            84 BEEP    EQU  $FBDD      ;BEEP THE SPEAKER
FC58:            85 HOME    EQU  $FC58      ;HOME-UP AND CLEAR SCREEN
FD8E:            86 CROUT   EQU  $FD8E      ;ISSUE A 'CR
FDF0:            87 COUT1   EQU  $FDF0      ;PRINT A CHAR
FF58:            88 RTS     EQU  $FF58      ;ADR OF RTS OPCODE
0000:            89 ;
0000:            90 ;
0000:            91 ;
0000:            92 ;
----- NEXT OBJECT FILE NAME IS FRE.CODE
9102:            93         ORG  $9102      ;TO LOCATE PROGRAM JUST BELOW DOS WITH MAXFILES = 3
9102:            94 ;
9102:            95 ;
9102:            96 ; INIT TRANSFERS A PREVIOUSLY DEFINED AMPERSAND HANDLER ADDRESS
9102:            97 ; TO THE " & FRE " FAILSAFE JUMP VECTOR.    FAILSAFE ALLOWS FOR
9102:            98 ; MORE THAN ONE AMPERSAND ROUTINE TO BE OPERATIVE AT THE SAME
9102:            99 ; TIME.    IF THE & COMMAND FAILS RECOGNITION, CONTROL IS
9102:           100 ; PASSED ON TO THE PREVIOUSLY ADDRESSED AMPERSAND HANDLER.
9102:           101 ;  INIT ALSO TESTS THE APPLESOFT HIMEM POINTER AND ADJUSTS IT
9102:           102 ; TO POINT TO JUST BELOW & FRE IF IT CURRENTLY IS SET HIGHER.
```

continued on next page

```
9102:           103 ;
9102:           104 ;
9102:           105 ;
9102:           106 ;
9102:           107 ;
9102:AD F6 03   108 INIT   LDA  $3F6
9105:8D 3F 92   109        STA  JMPAMP+1
9108:AD F7 03   110        LDA  $3F7
910B:8D 40 92   111        STA  JMPAMP+2  ;XFER OLD '&' ADR TO FAILSAFE VECTOR
910E:           112 ;
910E:A9 4C      113        LDA  #$4C
9110:8D F5 03   114        STA  $3F5      ;JMP OPCODE
9113:A9 36      115        LDA  #>START
9115:8D F6 03   116        STA  $3F6      ;PROG LO ADR BYTE
9118:A9 92      117        LDA  #<START
911A:8D F7 03   118        STA  $3F7      ;PROG HI ADR BYTE
911D:           119 ;
911D:AD 51 C0   120        LDA  TEXT
9120:AD 52 C0   121        LDA  NOMIX
9123:AD 54 C0   122        LDA  PAGE1
9126:20 58 FC   123        JSR  HOME
9129:20 8E FD   124        JSR  CROUT     ;HOME & CLEAR SCREEN
912C:20 8E FD   125        JSR  CROUT
912F:           126 ;
912F:A9 5C      127        LDA  #>MSG1
9131:A0 91      128        LDY  #<MSG1
9133:20 3A DB   129        JSR  STROUT    ;PRINT LINE ONE
9136:           130 ;
9136:20 8E FD   131        JSR  CROUT
9139:A9 85      132        LDA  #>MSG2
913B:A0 91      133        LDY  #<MSG2
913D:20 3A DB   134        JSR  STROUT    ;PRINT LINE TWO
9140:20 8E FD   135        JSR  CROUT
9143:20 8E FD   136        JSR  CROUT
9146:           137 ;
9146:38         138        SEC
9147:A9 00      139        LDA  #>INIT-2  ;COMPARE THE CURRENT SETTING OF APPLESOFT'S HIMEM
9149:E5 73      140        SBC  HIMEM     ;POINTER WITH THE STARTING ADDRESS OF THE & FRE
914B:A9 91      141        LDA  #<INIT    ;PROGRAM.  IF HIMEM IS ALREADY SET LOWER, THEN DO
914D:E5 74      142        SBC  HIMEM+1   ;NOTHING.   OTHERWISE ADJUST HIMEM TO POINT BELOW
914F:B0 08      143        BGE  NOSET     ;THE & FRE PROGRAM.
9151:A9 00      144        LDA  #>INIT-2
9153:85 73      145        STA  HIMEM     ;RESET HIMEM POINTER TO BELOW &FRE
9155:A9 91      146        LDA  #<INIT
9157:85 74      147        STA  HIMEM+1
9159:4C 03 E0   148 NOSET  JMP  ASWARM    ;GOTO APPLESOFT, INITIALIZATION COMPLETE.
915C:           149 ;
915C:           150 ;
915C:C1 D0 D8   151 MSG1   ASC            "APPLESOFT '& FRE' PROGRAM SPLITTER REL-1"
915F:CC C5 D3
9162:CF C6 D4
9165:A0 A7 A6
9168:A0 C6 D2
916B:C5 A7 A0
916E:D0 D2 CF
9171:C7 D2 C1
9174:CD A0 D3
9177:D0 CC C9
917A:D4 D4 C5
917D:D2 A0 D2
9180:C5 CC AD
9183:B1
9184:00         152        DFB  0
9185:C2 D9 A0   153 MSG2   ASC            "BY GARY JAMES - (C) 1983 MICROSPARC INC."
9188:C7 C1 D2
918B:D9 A0 CA
918E:C1 CD C5
9191:D3 A0 AD
9194:A0 A8 C3
9197:A9 A0 B1
919A:B9 B0 B3
919D:A0 CD C9
91A0:C3 D2 CF
91A3:D3 D8 C1
91A6:D2 C3 A0
91A9:C9 CE C3
91AC:AE
91AD:00         154        DFB  0
91AE:BE BF BE   155 MSG3   ASC            "????? NOT WITHIN RANGE <<<<<"
91B1:BE BE A0
91B4:CE CF D4
91B7:A0 D7 C9
91BA:D4 C8 C9
91BD:CE A0 D2
91C0:C1 CE C7
91C3:C5 A0 BC
91C6:BC BC BC
91C9:BC
91CA:00         156        DFB  0
91CB:D0 D2 CF   157 MSG4   ASC            "PROGRAM SPLIT AND MOVED AT LINE "
91CE:C7 D2 C1
91D1:CD A0 D3
91D4:D0 CC C9
91D7:D4 A0 C1
91DA:CE C4 A0
91DD:CD CF D6
91E0:C5 D4 A0
91E3:C1 D4 A0
91E6:CC C9 CE
91E9:C5 A0
```

```
91EB:00         158          DFB  0
91EC:BE BE A0   159 MSG5     ASC        "»»  LINE 0 OMITTED OR INCORRECT ««"
91EF:A0 CC C9
91F2:CE C5 A0
91F5:B0 A0 CF
91F8:CD C9 D4
91FB:D4 C5 C4
91FE:A0 CF D2
9201:A0 C9 CE
9204:C3 CF D2
9207:D2 C5 C3
920A:D4 A0 BC
920D:BC
920E:00         160          DFB  0
920F:           161          MSB  OFF
920F:00 00      162 LINE0ST  DW   0          ;FOR LINE #0
9211:3A 3A 3A   163          ASC        ":::::::::::::::::::::::"
9214:3A 3A 3A
9217:3A 3A 3A
921A:3A 3A 3A
921D:3A 3A 3A
9220:3A 3A 3A
9223:3A 3A 3A
9226:           164          MSB  ON
9226:           165 ;
9226:           166 ;
9226:           167 ;
9226:           168 ; TEMPORARY STORAGE REGISTERS USED BY ' & FRE '.
9226:           169 ;
9226:           170 ;
9226:           171 ADRLO    DS   2          ;TEMP START ADR
9228:           172 LENLO    DS   2          ;TEMP LENGTH
922A:           173 FREND    DS   2          ;TEMP FRE END ADR
922C:           174 LASTLN   DS   2          ;LAST UNMOVED LINE POINTER
922E:           175 TEMP1    DS   2          ;GENERAL PURPOSE STORAGE
9230:           176 TEMP2    DS   2          ;HOLDS FIRST FREE ZONE LINE NO.
9232:           177 TEMP3    DS   2          ;LAST LINES NUMBER
9234:           178 EOFLIN   DS   2          ;END OF FIRST LINE ADDRESS
9236:           179 ;
9236:           180 ;
9236:           181 ;
9236:           182 ;
9236:A9 D6      183 START    LDA  #$D6       ;GET " FRE " TOKEN
9238:A0 00      184          LDY  #0
923A:D1 B8      185          CMP  (TXTPTR),Y CHK IF FRE COMMAND
923C:F0 03      186          BEQ  OK
923E:4C 58 FF   187 JMPAMP   JMP  RTS        ;FAILSAFE JUMP VECTOR
9241:           188 ;
9241:20 B1 00   189 OK       JSR  CHRGET     ;POINT TXTPTR TO ADR
9244:20 67 DD   190          JSR  FRMNUM     ;GET START ADR TO FAC
9247:20 0C E1   191          JSR  AYINT      ;CONVERT IT TO INT
924A:38         192          SEC
924B:A5 A1      193          LDA  INTLO+1
924D:E9 10      194          SBC  #16        ;SUBTRACT 16 FROM ADR TO ALLOW
924F:8D 26 92   195          STA  ADRLO      ;ROOM FOR THE APPENDED " GOTO "
9252:A5 A0      196          LDA  INTLO
9254:E9 00      197          SBC  #0
9256:8D 27 92   198          STA  ADRLO+1
9259:20 BE DE   199          JSR  CHKCOM     ;CHK FOR COMMA DELIMITER
925C:20 67 DD   200          JSR  FRMNUM     ;GET LENGTH TO FAC
925F:20 0C E1   201          JSR  AYINT      ;CONVERT IT TO INT
9262:A5 A1      202          LDA  INTLO+1
9264:8D 28 92   203          STA  LENLO
9267:A5 A0      204          LDA  INTLO
9269:8D 29 92   205          STA  LENLO+1    ;XFER LENGTH TO LENLO
926C:           206 ;

926C:18         207          CLC
926D:AD 28 92   208          LDA  LENLO
9270:6D 26 92   209          ADC  ADRLO
9273:69 10      210          ADC  #16        ;FIX BECAUSE OF PRIOR ADJUSTMENT
9275:8D 2A 92   211          STA  FREND
9278:AD 29 92   212          LDA  LENLO+1
927B:6D 27 92   213          ADC  ADRLO+1
927E:8D 2B 92   214          STA  FREND+1    ;DERIVE FRE ZONE END ADR
9281:           215 ;
9281:           216 ;
9281:           217 ; SET PROGRAM BEGINNING = END OF SECOND LINE
9281:           218 ;
9281:A0 00      219          LDY  #0
9283:A5 67      220          LDA  PRGBEG
9285:85 08      221          STA  OFFSET
9287:A5 68      222          LDA  PRGBEG+1
9289:85 09      223          STA  OFFSET+1   ;OFFSET = LINK FIELD ADR OF 1'ST LINE
928B:B1 08      224          LDA  (OFFSET),Y
928D:8D 34 92   225          STA  EOFLIN
9290:C8         226          INY
9291:B1 08      227          LDA  (OFFSET),Y
9293:8D 35 92   228          STA  EOFLIN+1   ;EOFLIN = NEXT LINE LINK FIELD ADR
9296:           229 ;
9296:A5 08      230          LDA  OFFSET     ;TEMPORARILY SAVE OFFSET VALUE
9298:8D 2E 92   231          STA  TEMP1
929B:A5 09      232          LDA  OFFSET+1
929D:8D 2F 92   233          STA  TEMP1+1
92A0:           234 ;
92A0:           235 ;
92A0:           236 ;  EXIT IF FRE ADR < OR = PROGRAM BEGINNING
92A0:           237 ;
92A0:AD 34 92   238          LDA  EOFLIN     ;GET 2'ND LINES LINK VALUE
92A3:85 08      239          STA  OFFSET     ;XFER IT TO OFFSET
92A5:AD 35 92   240          LDA  EOFLIN+1
92A8:85 09      241          STA  OFFSET+1
92AA:           242 ;
92AA:88         243          DEY             ;SET YREG = 0
92AB:38         244          SEC
92AC:B1 08      245          LDA  (OFFSET),Y ;GET 3'RD LINES LINK FIELD ADR
92AE:ED 26 92   246          SBC  ADRLO      ;COMPARE IT TO THE FREE ZONE START ADR
92B1:C8         247          INY
92B2:B1 08      248          LDA  (OFFSET),Y
92B4:ED 27 92   249          SBC  ADRLO+1
92B7:90 1E      250          BLT  NXTCHK     ;BRA IF FREE ZONE START > END OF 2'ND PROGRAM LINE + 1
92B9:           251 ;
92B9:           252 ;
```

```
92B9:20 BE FD   253 EXIT     JSR  CROUT      ;OTHERWISE PRINT MESSAGE AND EXIT
92BC:A9 AE      254          LDA  #)MSG3
92BE:A0 91      255          LDY  #(MSG3
92C0:20 3A DB   256 EXIT1    JSR  STROUT     ;PRINT RANGE MESSAGE
92C3:20 DD FB   257          JSR  BEEP
92C6:20 DD FB   258          JSR  BEEP
92C9:20 BE FD   259          JSR  CROUT
92CC:60         260          RTS             ;RETURN TO BASIC
92CD:           261 ;
92CD:20 BE FD   262 EXIT2    JSR  CROUT
92D0:A9 EC      263          LDA  #)MSG5
92D2:A0 91      264          LDY  #(MSG5
92D4:4C C0 92   265          JMP  EXIT1      ;PRINT LINE 0 ERROR MSG
92D7:           266 ;
92D7:           267 ;
92D7:           268 ;  EXIT IF PROGRAM END ADR < FRE ADR
92D7:           269 ;
92D7:           270 ;
92D7:AD 2E 92   271 NXTCHK   LDA  TEMP1      ;RESTORE ORIGINAL OFFSET VALUE
92DA:85 08      272          STA  OFFSET
92DC:AD 2F 92   273          LDA  TEMP1+1
92DF:85 09      274          STA  OFFSET+1
92E1:           275 ;
92E1:38         276          SEC
92E2:AD 26 92   277          LDA  ADRLO
92E5:E5 AF      278          SBC  PRGEND
92E7:AD 27 92   279          LDA  ADRLO+1
92EA:E5 B0      280          SBC  PRGEND+1
92EC:B0 CB      281          BGE  EXIT       ;BRA IF FRE ZONE ADR > END ADR
92EE:           282 ;
92EE:           283 ;
92EE:           284 ; EXIT IF PROGRAM LINE NUMBER ZERO IS OMITTED OR INCORRECTLY INITIALIZED
92EE:           285 ;
92EE:18         286          CLC
92EF:A9 01      287          LDA  #1         ;SET OFFSET = LINE 0 LINK ADR + 1
92F1:65 08      288          ADC  OFFSET
92F3:85 08      289          STA  OFFSET
92F5:A9 00      290          LDA  #0
92F7:65 09      291          ADC  OFFSET+1
92F9:85 09      292          STA  OFFSET+1
92FB:           293 ;
92FB:38         294          SEC
92FC:AD 34 92   295          LDA  EOFLIN
92FF:E5 08      296          SBC  OFFSET
9301:E9 02      297          SBC  #2
9303:C9 17      298          CMP  #$17       ;CHECK FOR 21 COLONS IN LINE 0
9305:D0 C4      299          BNE  EXIT2
9307:A8         300          TAY             ;YREG = OFFSET VAL TO LINE 0 LAST BYTE
9308:           301 ;
9308:B1 08      302 LIN0TST  LDA  (OFFSET),Y
930A:D9 0E 92   303          CMP  LINE0ST-1,Y ;CMP LINE 0 BYTE WITH CHECK STRING
930D:D0 BE      304          BNE  EXIT2
930F:88         305          DEY
9310:D0 F6      306          BNE  LIN0TST    ;GO DO NEXT BYTE
9312:           307 ;
9312:           308 ;

9312:           309 ;
9312:           310 ;
9312:           311 ;  START OF ROUTINE THAT RELOCATES THE SECTION OF PROGRAM
9312:           312 ;  THAT OVERLAPS THE " FRE " MEMORY AREA.    THE PROGRAM
9312:           313 ;  IS MOVED FROM THE BEGINNING ADDRESS OF THE LINE NUMBER
9312:           314 ;  THAT IS FOUND EXTENDING INTO THE FRE ZONE.
9312:           315 ;
9312:           316 ;
9312:A5 67      317 MOVEIT   LDA  PRGBEG
9314:8D 2E 92   318          STA  TEMP1
9317:A5 68      319          LDA  PRGBEG+1
9319:8D 2F 92   320          STA  TEMP1+1    ;XFER PROG START ADR -> TEMP1
931C:           321 ;
931C:A0 01      322 FNDLIN   LDY  #1
931E:88         323 OUT1     DEY             ;SET Y = 0
931F:AD 2E 92   324          LDA  TEMP1
9322:8D 2C 92   325          STA  LASTLN
9325:85 08      326          STA  OFFSET
9327:AD 2F 92   327          LDA  TEMP1+1
932A:8D 2D 92   328          STA  LASTLN+1
932D:85 09      329          STA  OFFSET+1   ;POINT OFFSET & LASTLN TO LINK FIELD OF CURRENT LINE
932F:           330 ;
932F:B1 08      331          LDA  (OFFSET),Y ;GET LINK FIELD LO-BYTE
9331:8D 2E 92   332          STA  TEMP1
9334:C8         333          INY
9335:B1 08      334          LDA  (OFFSET),Y ;GET LINK FIELD HI-BYTE
9337:8D 2F 92   335          STA  TEMP1+1    ;LOAD TEMP1 WITH ADR OF NEXT LINES LINK FIELD
933A:C8         336          INY
933B:B1 08      337          LDA  (OFFSET),Y ;GET LINE NUMBER LO BYTE
933D:8D 32 92   338          STA  TEMP3      ;SAVE IT
9340:C8         339          INY
9341:B1 08      340          LDA  (OFFSET),Y ;GET HI BYTE
9343:8D 33 92   341          STA  TEMP3+1    ;CURRENT LINES NUMBER -> TEMP3
9346:A0 00      342          LDY  #0
9348:           343 ;
9348:AD 2E 92   344          LDA  TEMP1
934B:85 08      345          STA  OFFSET
934D:AD 2F 92   346          LDA  TEMP1+1
9350:85 09      347          STA  OFFSET+1   ;SET OFFSET = NEXT LINES LINK FIELD ADR
9352:           348 ;
9352:B1 08      349          LDA  (OFFSET),Y ;GET 3'RD LINES LINK FIELD LO BYTE
9354:8D 38 92   350          STA  TEMP2
9357:C8         351          INY
9358:B1 08      352          LDA  (OFFSET),Y
935A:8D 31 92   353          STA  TEMP2+1    ;SAVE HI BYTE
935D:           354 ;
935D:38         355          SEC             ;PREPARE FOR COMPARISON OF LINK ADDRESS VALUE
935E:AD 38 92   356          LDA  TEMP2      ;AND STARTING ADDRESS
9361:ED 26 92   357          SBC  ADRLO      ;SUBTRACT LO BYTES
9364:AD 31 92   358          LDA  TEMP2+1
9367:ED 27 92   359          SBC  ADRLO+1    ;SUBTRACT HI BYTES
936A:90 B2      360          BLT  OUT1       ;BRA IF LINK FIELD VALUE
936C:C8         361          INY               ;IS LESS THAN STARTING ADDRESS
936D:B1 08      362          LDA  (OFFSET),Y ;GET LINENO LO-BYTE
936F:8D 30 92   363          STA  TEMP2
9372:C8         364          INY
9373:B1 08      365          LDA  (OFFSET),Y ;GET LINENO HI-BYTE
9375:8D 31 92   366          STA  TEMP2+1
9378:           367 ;
```

```
9378:          368 ; END WITH NEXT LINEND IN TEMP2 ; CURRENT LINEND IN TEMP3  ; LINE ADR IN OFFS
9378:          369 ; AND LAST UNMOVED LINE STARTING ADR IN LASTLN
9378:          370 ;
9378:          371 ;
9378:          372 ; PRESET HIGHTR & LOWTR FOR BLTU
9378:          373 ;
9378:A5 08     374          LDA  OFFSET
937A:85 98     375          STA  LOWTR
937C:A5 09     376          LDA  OFFSET+1
937E:85 9C     377          STA  LOWTR+1   ;LOWTR
9380:          378 ;
9380:A5 AF     379          LDA  PRGEND
9382:85 96     380          STA  HIGHTR
9384:A5 B0     381          LDA  PRGEND+1
9386:85 97     382          STA  HIGHTR+1
9388:          383 ;
9388:          384 ;
9388:          385 ; TEST FOR PROGRAM END INSIDE OR BEYOND FREE ZONE
9388:          386 ; THEN SELECT WHICH TYPE OF BLOCK MOVE TO USE
9388:          387 ;
9388:38        388          SEC
9389:A5 AF     389          LDA  PRGEND
938B:ED 2A 92  390          SBC  FREND
938E:A5 B0     391          LDA  PRGEND+1
9390:ED 2B 92  392          SBC  FREND+1
9393:B0 33     393          BGE  CBA       ;BRA IF PROGRAM END > FREE ZONE END
9395:          394 ;
9395:          395 ; PROGRAM END WITHIN FREE ZONE
9395:          396 ; SET HIGHDS = FREND - ( PRGEND - OFFSET )
9395:          397 ; SET PRGEND = VARIST = HIGHDS+1
9395:          398 ;
9395:38        399          SEC
9396:A5 AF     400          LDA  PRGEND
9398:E5 08     401          SBC  OFFSET
939A:AA        402          TAX
939B:A5 B0     403          LDA  PRGEND+1
939D:E5 09     404          SBC  OFFSET+1
939F:A8        405          TAY
93A0:          406 ;
93A0:18        407          CLC
93A1:8A        408          TXA
93A2:6D 2A 92  409          ADC  FREND
93A5:90 06     410          BCC  NOCRY1
93A7:18        411          CLC
93A8:69 01     412          ADC  #1
93AA:38        413          SEC
93AB:B0 02     414          BCS  CARY1
93AD:69 01     415 NOCRY1   ADC  #1
93AF:85 94     416 CARY1    STA  HIGHDS
93B1:85 69     417          STA  VARIST
93B3:85 AF     418          STA  PRGEND
93B5:AA        419          TAX
93B6:98        420          TYA
93B7:6D 2B 92  421          ADC  FREND+1
93BA:85 95     422          STA  HIGHDS+1
93BC:85 6A     423          STA  VARIST+1
93BE:85 B0     424          STA  PRGEND+1
93C0:A8        425          TAY
93C1:8A        426          TXA
93C2:20 93 D3  427          JSR  BLTU      ;MOVE PROGRAM
93C5:38        428          SEC
93C6:B0 38     429          BCS  DOLINO
93C8:          430 ;
93C8:          431 ;
93C8:          432 ; PROGRAM END BEYOND FREE ZONE END
93C8:          433 ; SET HIGHDS = PRGEND + LENLO,HI
93C8:          434 ; SET PRGEND = VARIST = HIGHDS
93C8:          435 ;
93C8:38        436 CBA      SEC
93C9:AD 2A 92  437          LDA  FREND
93CC:E5 08     438          SBC  OFFSET
93CE:AA        439          TAX
93CF:AD 2B 92  440          LDA  FREND+1
93D2:E5 09     441          SBC  OFFSET+1
93D4:A8        442          TAY
93D5:          443 ;
93D5:18        444          CLC
93D6:8A        445          TXA
93D7:65 AF     446          ADC  PRGEND
93D9:90 06     447          BCC  NOCRY2
93DB:18        448          CLC
93DC:69 01     449          ADC  #1
93DE:38        450          SEC
93DF:B0 02     451          BCS  CARY2
93E1:69 01     452 NOCRY2   ADC  #1
93E3:85 94     453 CARY2    STA  HIGHDS
93E5:85 69     454          STA  VARIST
93E7:85 AF     455          STA  PRGEND
93E9:AA        456          TAX
93EA:98        457          TYA
93EB:65 B0     458          ADC  PRGEND+1
93ED:85 95     459          STA  HIGHDS+1
93EF:85 6A     460          STA  VARIST+1
93F1:85 B0     461          STA  PRGEND+1
93F3:A8        462          TAY
93F4:8A        463          TXA
93F5:20 93 D3  464          JSR  BLTU      ;MOVE PROGRAM
93F8:          465 ;
93F8:          466 ;
93F8:AD 31 92  467 DOLINO   LDA  TEMP2+1   ;TARGET LINE NUMBER HI BYTE
93FB:AC 30 92  468          LDY  TEMP2     ;LO BYTE
93FE:20 F2 E2  469          JSR  GIVAYF    ;CONVERT LINO TO FAC
9401:20 34 ED  470          JSR  FOUT      ;CONVERT FAC TO ASCII STRING
9404:A9 CB     471          LDA  #)MSG4
9406:A0 91     472          LDY  #(MSG4
9408:20 3A DB  473          JSR  STROUT    ;PRINT MOVED LINE STRING
940B:A9 00     474          LDA  #)FBUFR
940D:A0 01     475          LDY  #(FBUFR
940F:20 3A DB  476          JSR  STROUT    ;PRINT LINE NUMBER
9412:20 8E FD  477          JSR  CROUT
9415:          478 ;
9415:AD 2A 92  479          LDA  FREND
9418:18        480          CLC
9419:69 01     481          ADC  #1
941B:8D 2E 92  482          STA  TEMP1
941E:AD 2B 92  483          LDA  FREND+1
9421:69 00     484          ADC  #0
9423:8D 2F 92  485          STA  TEMP1+1   ;LINK FIELD ADR OF FIRST MOVED LINE -> TEMP1
9426:          486 ;
9426:A0 00     487          LDY  #0
9428:AD 2E 92  488          LDA  TEMP1
942B:91 08     489          STA  (OFFSET),Y
942D:C8        490          INY
942E:AD 2F 92  491          LDA  TEMP1+1
9431:91 08     492          STA  (OFFSET),Y ;APPEND LINK FIELD ADR OF MOVED LINE
9433:          493 ;
9433:C8        494          INY
9434:AD 32 92  495          LDA  TEMP3
9437:91 08     496          STA  (OFFSET),Y
9439:C8        497          INY
943A:AD 33 92  498          LDA  TEMP3+1
943D:91 08     499          STA  (OFFSET),Y ;APPEND DUPLICATE LINE NUMBER
943F:C8        500          INY
9440:          501 ;
9440:A9 AB     502          LDA  #$AB      ;"GOTO" TOKEN
9442:91 08     503          STA  (OFFSET),Y ;APPEND "GOTO"
9444:C8        504 GETSTR   INY
9445:B9 FB 00  505          LDA  $FB,Y     ;GET FAC STRING CHAR
9448:91 08     506          STA  (OFFSET),Y ;PUT IT AT EOL
944A:D0 F8     507          BNE  GETSTR    ; GET NEXT CHAR
944C:          508 ;
944C:C8        509          INY
944D:AD 2E 92  510          LDA  TEMP1
9450:91 08     511          STA  (OFFSET),Y
9452:C8        512          INY
9583:A8        616          TAY
9584:AD 2D 92  617          LDA  LASTLN+1
9587:69 00     618          ADC  #0
9589:20 F2 E2  619          JSR  GIVAYF
958C:20 34 ED  620          JSR  FOUT
958F:          621 ;
958F:20 57 95  622          JSR  TTO
9512:A0 01     623          LDY  #1
9514:20 6A 95  624          JSR  SAVEOFF
9517:20 57 95  625          JSR  TTO
951A:20 47 95  626          JSR  DOPOKE
951D:20 64 95  627          JSR  DOCOMMA
9528:          628 ;
9528:AD 2C 92  629          LDA  LASTLN
9523:85 08     630          STA  OFFSET
9525:AD 2D 92  631          LDA  LASTLN+1
9528:85 09     632          STA  OFFSET+1
952A:A0 01     633          LDY  #1
952C:B1 08     634          LDA  (OFFSET),Y
952E:A8        635          TAY
952F:A9 00     636          LDA  #0
9531:20 F2 E2  637          JSR  GIVAYF
9534:20 34 ED  638          JSR  FOUT
9537:A9 00     639          LDA  #0
9539:A0 01     640          LDY  #1
953B:91 08     641          STA  (OFFSET),Y ;CLR LAST LINE LINK HI BYTE
953D:20 57 95  642          JSR  TTO
9540:C8        643          INY
9541:20 4C 95  644          JSR  POCODE
9544:          645 ;
9544:4C 6C D6  646          JMP  CLEAR     ;LET APPLESOFT CLEAN UP REST
9547:          647 ;
9547:A9 B9     648 DOPOKE   LDA  #$B9      ;POKE COMMAND TOKEN
9549:C8        649          INY
954A:91 08     650 POKCODE  STA  (OFFSET),Y
954C:C8        651 POCODE   INY
954D:B9 FE 00  652          LDA  $FE,Y     ;GET STRING CHAR.
9550:D0 F8     653          BNE  POKCODE   ;BRANCH IF NOT DONE
9552:88        654          DEY
9553:20 6A 95  655          JSR  SAVEOFF
9556:60        656          RTS
9557:          657 ;
9557:AD 2E 92  658 TTO      LDA  TEMP1
955A:85 08     659          STA  OFFSET
955C:AD 2F 92  660          LDA  TEMP1+1
955F:85 09     661          STA  OFFSET+1
9561:A0 00     662          LDY  #0
9563:60        663          RTS
9564:          664 ;
9564:A9 2C     665 DOCOMMA  LDA  #$2C      ;ASCII COMMA
9566:C8        666          INY
9567:91 08     667          STA  (OFFSET),Y
9569:60        668          RTS
956A:          669 ;
956A:18        670 SAVEOFF  CLC
956B:98        671          TYA
956C:65 08     672          ADC  OFFSET
956E:8D 2E 92  673          STA  TEMP1
9571:A9 00     674          LDA  #0
9573:65 09     675          ADC  OFFSET+1
9575:8D 2F 92  676          STA  TEMP1+1
9578:60        677          RTS
9579:          678 ;
9579:          679 ;
9579:          680 ;
9579:          681 END      EQU  *
9579:          682 ;
0477:          683 LENGTH   EQU  END-INIT  ;PROGRAM LENGTH
9579:          684 ;
9579:          685 ;

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

```
        KEY PERFECT 4.0
           RUN ON
         FRE.CODE
-------------------------------
    CODE      ADDR# - ADDR#
    ------    -------------
     25D4      9102 - 9151
     263A      9152 - 91A1
     2975      91A2 - 91F1
     2879      91F2 - 9241
     2168      9242 - 9291
     29C9      9292 - 92E1
     2C6C      92E2 - 9331
     2855      9332 - 9381
     2971      9382 - 93D1
     2A82      93D2 - 9421
     273F      9422 - 9471
     2765      9472 - 94C1
     26B9      94C2 - 9511
     24BE      9512 - 9561
     0A72      9562 - 9578
TOTAL PROGRAM CHECK IS : 0477
```

```
APPLE CHECKER

ON: FRE.CODE
TYPE: B

LENGTH: 0477
CHECKSUM: 2F
```