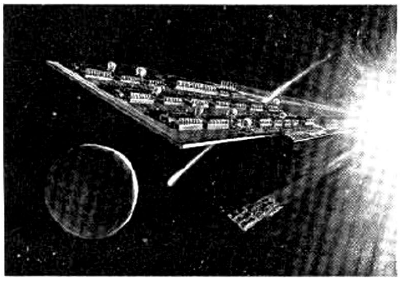
FAST GOTO PROCESSOR



A FAST.GOTO Processor for Applesoft

by Dale Waddell
3648 Greenstone Ct.
Dayton, OH 45430

David H. Bartley had articles in both the Spring 1981 and Summer 1981 issues of APPLE ORCHARD explaining how the GOTO and GOSUB logic within Applesoft works. Then the August 1981 issue of MICRO contained an article called "The Extended Parser for the APPLE II" by Paul R. Wilson. After reading these articles several times, I began to get an idea of another possible way to improve Applesoft's GOTO and GOSUB processing. So, armed with all of this new knowledge, I set out to design a new FAST.GOTO processor. The objective was to limit changes in the Applesoft program to two statements: one statement to activate the FAST.GOTO processor, and another to deactivate it. Then a programmer with very little effort can get the benefits of the processor for new or existing Applesoft programs.

USING FAST.GOTO

To make the changes to an Applesoft program so that it will use the FAST.GOTO processor, do the following:

1. Add the following statement to activate the FAST.GOTO processor: **PRINT CHRS (4)"BRUN FAST.GOTO.OBJ0"**. The processor assumes that any ampersand routines have already been set up, that no string variables have yet been used and that memory locations **$934A-$95FF** are unused, so just to be safe, this should be the first statement in your program.
2. Add statement **& HOME** where the program completes execution to deactivate the FAST.GOTO processor.

WARNING: Any other ampersand routines or CHRGET routine hooks that exist in a program using the FAST.GOTO processor **cannot** use Applesoft tokens **GOTO ($AB)**, **GOSUB ($B0)** or **HOME ($97)**. Be very careful attempting to use Applesoft tokens **ON ($B4)**, **ONERR ($A5)**, **IF ($AD)** or **THEN ($C4)**; see the notes below.

Your program should now work with the FAST.GOTO processor. To get more benefit from the processor you may want to consider doing the following:

1. Change IF statements of the following format
   IF..... THEN nnnn or IF..... GOTO nnnn to the form:
   IF..... THEN GOTO nnnn
2. Renumber the program starting with a low number and increment by a small number, e.g. starting with 2 and incrementing by 2. It is better to have the last line number be a power of 2, because then all index entries will be used. This will probably reduce the size of the program. It also improves the balance of the way each index entry is used within FAST.GOTO processor.

Don't forget to save the new version of your Applesoft program.

To date, the FAST.GOTO processor has been tested with several programs without any problems.

TIMING TESTS

I took a 377 statement program that had already had considerable rework done to make it faster, and ran some timing tests. Below is a summary of these tests.

TEST NUMBER	PROGRAM RENUMBERED	RUN TIME (SEC)	FAST.GOTO PROCESSOR USED
1	FIRST 2,INC 2	1195	NO
2	FIRST 64,INC 128	838	NO
3	FIRST 64,INC 128	590	YES
4	FIRST 2,INC 2	546	YES

PROBLEMS/ALTERNATIVES

The first method I tried was to modify Applesoft's CHRGET routine to recognize GOTO and GOSUB tokens, and have the processor immediately perform the requested function. This proved to be a big failure, because Applesoft also uses the CHRGET routine to skip data in the Applesoft program without executing it.

After several days of thinking about why the above approach failed, I hit on the idea of letting the CHRGET routine change GOTO and GOSUB tokens to Ampersand tokens, and then letting an Ampersand routine restore the original token and perform the GOTO and GOSUB functions. This approach has proved to work very well. Three different problems have been discovered and resolved so far:

A. The next GOTO or GOSUB token found after an ON or ONERR token cannot be handled by the processor, because completely different logic is used by Applesoft to handle ON and ONERR processing.

B. Applesoft allows three different syntax formats for an IF statement which branches to another line if true (see page 76 of the Applesoft Reference Manual):
   1) IF ... THEN GOTO nnnn:
   2) IF ... THEN nnnn:
   3) IF ... GOTO nnnn:
   In case 3) above, a SYNTAX ERROR message is issued if the GOTO token is replaced with an AMPERSAND token. Another test took care of this problem.

C. Applesoft also uses the CHRGET routine to skip the remainder of a statement when the IF expression is false. This means that the previously changed token must be restored before another token is changed to an AMPERSAND.

After my success with using the CHRGET routine to direct an ampersand routine to do the GOTO and GOSUB processing, I began work on another approach for the FAST.GOTO processor. Bartley's normal Amper GOTO/GOSUB routine only improves run time over Applesoft for short forward jumps. His radical Amper GOTO/GOSUB routines are faster than the routine shown in this article, but I believe that they suffer from the third problem outlined above if the line number is located at the beginning of a memory page (low byte of the address is equal to **$00**). The radical approach also modifies the program such that it cannot be edited after it has been executed. Both of Bartley's approaches require that the programmer insert a lot of ampersands into the Applesoft program.

THE APPROACH

The approach selected for the FAST.GOTO processor is to:

1. Let **X** equal the next power of **2** that is equal to or greater than the highest line number divided by **64**.
2. Build an index that contains 33 to 65 entries. Each entry will contain the location of the first line within the **X** range of line numbers. I have tried large indexes with very little improvement in execution time.
3. When attempting to find a line, the routine divides the line number by **X** to determine which index entry should be used. The accumulator and the X register are loaded from the index entry and then the routine jumps into the Applesoft logic to find the line.

Since this FAST.GOTO processor will always perform very fast line number lookups, there is nothing to be gained by placing frequently used routines in the front of the program. REMarks on separate statements will not slow the program down as much as with the method used by Applesoft to find a line. Execution speed doesn't slow down as the program gets larger.

When the program stops running, it is necessary to have a method to stop the FAST.GOTO processor. The method chosen was to place an **& HOME:** statement into the Applesoft program where the FAST.GOTO processor is no longer needed. The ampersand routine recognizes the **& HOME:** and removes all hooks set by the FAST.GOTO processor.

THE AMPERSAND

Since the FAST.GOTO processor uses ampersand routines, provisions have been made for those Applesoft programs that already use ampersand routines. The processor saves the previous ampersand routine address so that the ampersand routine in the FAST.GOTO processor can jump to it if some unexpected data is found. Another problem is due to the fact that some products such as Super Kram also modify the CHRGET routine in Applesoft. Therefore additional logic was added to determine if another routine has already modified the CHRGET routine. If so, then the FAST.GOTO hook is modified to jump to the next hooked location plus 6.

SYSTEM REQUIREMENTS

The enclosed program listing has been ORG'ed to run immediately below $9600. This is the normal HIMEM location for 48K

A FAST.GOTO Processor for Applesoft (Cont.)

systems with DOS 3.3. This article assumes that the reader will be able to save the FAST. GOTO processor onto the same diskette as the Applesoft program using it. The suggested name for the binary program is FAST. GOTO.OBJ0. I used the assembler in the DOS TOOL KIT to assemble this program.

Now that we know why the FAST.GOTO processor was created, let us take a walk through the code and see how it operates. Let's go from top to bottom, examining the more important parts.

HOW IT WORKS

Statement **27** can be changed to cause the processor to be executed from another area. At the end of the program there is a series of EQU's defining where the ORG should be set for some of the more common locations. I suggest that the FAST.GOTO processor be executed immediately below where HIMEM is set. This program is **not** designed to be self-relocating.

Statements **73-78** save the first 3 bytes of Applesoft's CHRGET routine.

Statements **79-88** look for the possibility that CHRGET has already been "hooked" by some other program. If it is already "hooked", then the instruction at statement **277** is modified to jump to the "hooked" location **plus 6**.

Statements **92-110** save any previous ampersand routine address and HIMEM location. Then the CHRGET routine and the ampersand jump instruction are changed to point to routines in the FAST.GOTO processor.

Statements **113-126** determine if **MEMSIZ** is greater than the location where the FAST. GOTO processor is being executed. If so, then both **MEMSIZ** and **FRETOP** are changed so that the processor will be protected.

Statements **130-150** scan the entire Applesoft program to **find the highest line number**. That line number is needed in order to determine the range of line numbers for index entry.

Statements **153-158** use rotate instructions to divide the highest line number by **64**. Remember that **64 is 2 to the 6th power**. The reason for dividing by a power of 2 is so that future divides may be done faster via rotate instructions.

Statements **166-179** use a series of rotate instructions to find a value equal to or greater than the value computed in statements **153-158**. This value will be a power of **2**. Each index point will index this range of line numbers. During these instructions, statement **395** is modified so that the line numbers will be divided by this value.

Statements **184-224** construct the index. Each index entry contains the location of the first line within its range of line numbers.

The part of the program prior to statement **233** is only used to do the setup work so that the FAST.GOTO processor can do its work. As you remember, **MEMSIZ** was set to the same location as statement **253**.

Statements **253-260** replace the first four instructions in the **CHRGET** routine.

Statement **261** tests for an Applesoft token. All Applesoft tokens are greater than **$7F**. If it is not an Applesoft token, a jump occurs to the test for the end of the Applesoft statement.

Statements **262-273** test for the various Applesoft tokens involved in these processes. If any other Applesoft token is encountered, a jump occurs to the exit location.

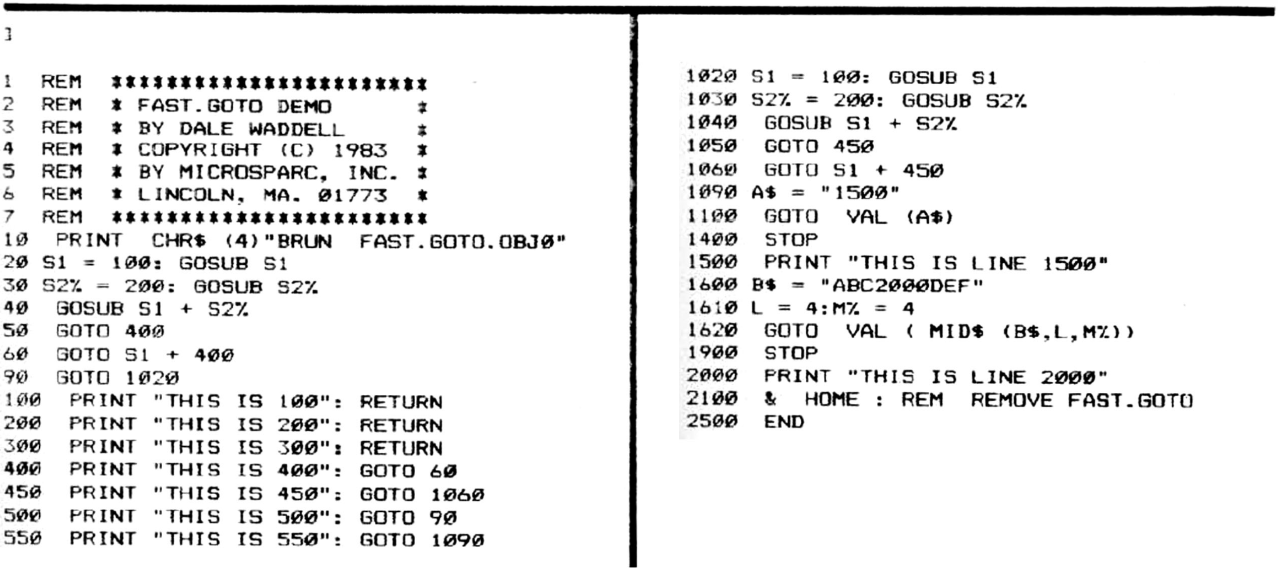
Statement **275** sets **SW** to remember than an **ON**, **ONERR** or **IF** token has been found. This is the manner used to determine when **GOTO** and **GOSUB** tokens cannot be changed to an AMPERSAND token.

Statement **276** will either clear or set **SW**.

Statement **277** either returns to the 4th instruction in **CHRGET** or to another "hooked" routine plus 6.

Statements **281-285** check for the end of an Applesoft statement. If so, then **SW** is cleared to indicate that it's OK to change the next GOTO or GOSUB token to an AMPERSAND token.

Statements **287-290** "remember" the current token and then test to see if **SW** is on. If **SW** is on, the token cannot be changed to an Ampersand token, so **SW** is cleared and the next routine processes the current token.



```
]

1  REM  ***********************
2  REM  * FAST.GOTO DEMO      *
3  REM  * BY DALE WADDELL     *
4  REM  * COPYRIGHT (C) 1983  *
5  REM  * BY MICROSPARC, INC. *
6  REM  * LINCOLN, MA. Ø1773  *
7  REM  ***********************
1Ø  PRINT  CHR$ (4)"BRUN  FAST.GOTO.OBJØ"
2Ø S1 = 1ØØ: GOSUB S1
3Ø S2% = 2ØØ: GOSUB S2%
4Ø  GOSUB S1 + S2%
5Ø  GOTO 4ØØ
6Ø  GOTO S1 + 4ØØ
9Ø  GOTO 1Ø2Ø
1ØØ  PRINT "THIS IS 1ØØ": RETURN
2ØØ  PRINT "THIS IS 2ØØ": RETURN
3ØØ  PRINT "THIS IS 3ØØ": RETURN
4ØØ  PRINT "THIS IS 4ØØ": GOTO 6Ø
45Ø  PRINT "THIS IS 45Ø": GOTO 1Ø6Ø
5ØØ  PRINT "THIS IS 5ØØ": GOTO 9Ø
55Ø  PRINT "THIS IS 55Ø": GOTO 1Ø9Ø
1Ø2Ø S1 = 1ØØ: GOSUB S1
1Ø3Ø S2% = 2ØØ: GOSUB S2%
1Ø4Ø  GOSUB S1 + S2%
1Ø5Ø  GOTO 45Ø
1Ø6Ø  GOTO S1 + 45Ø
1Ø9Ø A$ = "15ØØ"
11ØØ  GOTO  VAL (A$)
14ØØ  STOP
15ØØ  PRINT "THIS IS LINE 15ØØ"
16ØØ B$ = "ABC2ØØØDEF"
161Ø L = 4:M% = 4
162Ø  GOTO  VAL ( MID$ (B$,L,M%))
19ØØ  STOP
2ØØØ  PRINT "THIS IS LINE 2ØØØ"
21ØØ  &  HOME : REM  REMOVE FAST.GOTO
25ØØ  END
```

A FAST.GOTO Processor for Applesoft (Cont.)

Statements **291-298** check to see if the previous token that was changed to an AMPERSAND token has been restored; if not, it is now restored. This is because at any given time only one token can be changed to an Ampersand token.

Statements **304-312** save the location of the current token. Then the current token is changed to an AMPERSAND token before exiting the CHRGET routine.

Statements **314-315** is a subroutine used to change and restore GOTO and GOSUB tokens. Statement **314** was modified by statements **304-307**.

Statements **326-329** test to see if a GOTO or GOSUB token has been changed by the CHRGET routine. If so, a jump occurs to the routine at statement **359**.

Statements **330-332** test for the **& HOME** condition; if not, control passes to the next amperand routine.

Statements **333-353** restore, if necessary, the previous GOTO or GOSUB token and the original value in **MEMSIZ**. Then the hooks for **CHRGET** and ampersand routines are removed. Note that **FRETOP** is not restored. The next time "garbage collection" is done, **FRETOP** will be reset.

Statement **354** jumps to the next ampersand routine. This instruction was modified during the housekeeping process.

Statements **359-368** check to determine if the AMPERSAND token that caused the ampersand routine to get control is the same one used to replace a GOTO or GOSUB token. If not, control passes to statement **330** and other ampersands are processed.

Statement **369** restores the GOTO or GOSUB token.

Statements **371-372** test for a GOTO token; if so, control passes to statement **389**.

Statements **374-387** are copied from Applesoft, because statement **386** uses the GOTO logic supplied in this program.

Statements **389-390** clear the indications that a token was changed to an AMPERSAND token.

Statement **391** goes to the CHRGET routine to "re-get" the current character. CHRGET will clear the carry indicator if the character is numeric.

Statement **392** tests to see if the character is numeric; if not, control passes to statement **411** to process the named line number. This feature was added to allow named GOTO and GOSUB statements. For more information, check the article "Using Named GOSUB and GOTO Statements in Applesoft Basic" by M. R. Smith in the May 1981 issue of COMPUTE.

Statement **393** uses an Applesoft routine to normalize the line number.

Statements **395-406** use rotate instructions to divide the requested line number by the value determined during housekeeping.

Statements **407-409** use the value computed in statements **395-406** to control which index entry is used to set the **X** register and the accumulator. Then statement **409** jumps into the middle of the GOTO logic in Applesoft to find the line number requested. After the line number is found Applesoft executes the new line.

```
SOURCE FILE: FAST.GOTO.SRC TOOLKIT
                    1 ************************************************************
0000:               2 *
0000:               3 *  FAST.GOTO
0000:               4 *  BY DALE WADDELL
0000:               5 *
0000:               6 *  COPYRIGHT 1983 BY MICROSPARC, INC.
0000:               7 *
0000:               8 *
0000:               9 *    THIS PROGRAM REQUIRES THAT IT BE BRUN FROM THE APPLESOFT
0000:              10 * PROGRAM BEFORE ANY STRINGS ARE USED.
0000:              11 * SUGGESTED APPLESOFT STATEMENT IS:
0000:              12 *    PRINT CHR$(4)"BRUN FAST.GOTO.OBJ0"
0000:              13 *
0000:              14 * THE REASON THAT THIS PROGRAM MUST BE RUN FROM THE APPLESOFT
0000:              15 * PROGRAM IS SO THAT A TABLE OF LINE LOCATIONS CAN BE
0000:              16 * CONSTRUCTED.
0000:              17 *
0000:              18 * THIS PROGRAM IS ALSO DESIGNED TO WORK WITH APPLESOFT
0000:              19 * PROGRAMS RUNNING WITH SUPER KRAM
0000:              20 *
0000:              21 *
0000:              22 *
0000:              23 *
0000:              24 *
----- NEXT OBJECT FILE NAME IS FAST.GOTO.SRC TOOLKIT.OBJ0
934A:              25          ORG  $934A      ;SET AS NECESSARY
934A:              26 *
934A:              27 *  APPLESOFT TOKENS
934A:              28 *
00AB:              29 TKGOTO  EQU  $AB        ;GOTO
00B0:              30 TKGOSUB EQU  $B0        ;GOSUB
00AF:              31 TKAMPER EQU  $AF        ;AMPER
00B4:              32 TKON    EQU  $B4        ;ON
00A5:              33 TKONERR EQU  $A5        ;ONERR
00AD:              34 TKIF    EQU  $AD        ;IF
00C4:              35 TKTHEN  EQU  $C4        ;THEN
0097:              36 TKHOME  EQU  $97        ;HOME
003A:              37 ENDSTMT EQU  $3A        ;END OF STMT
934A:              38 *
934A:              39 *  6502 OPERATION CODES
934A:              40 *
004C:              41 OPJMP   EQU  $4C        ;JMP
934A:              42 *
934A:              43 * PAGE ZERO INFORMATION
934A:              44 *
0050:              45 LINNUM  EQU  $50        ;GOTO LINE NUMBER
0067:              46 TXTTAB  EQU  $67        ;BEGIN OF PROGRAM POINTER
006F:              47 FRETOP  EQU  $6F        ;TOP OF FREE MEMORY
0073:              48 MEMSIZ  EQU  $73        ;APPLESOFT MEMSIZ
0075:              49 CURLIN  EQU  $75        ;CURRENT LINE NUMBER
009B:              50 WRKPTR  EQU  $9B        ;WRK POINTER DURING HOUSEKEEPING
00B1:              51 CHRGET  EQU  $B1        ;GET NEXT CHAR
00B8:              52 TXTPTR  EQU  $B8        ;TEXT POINTER
934A:              53 *
934A:              54 *  PAGE THREE INFORMATION
934A:              55 *
03F5:              56 AVECTOR EQU  $3F5       ;AMPER VECTOR
934A:              57 *
934A:              58 *  ROUTINES WITHIN APPLESOFT
934A:              59 *
D3D6:              60 STACK   EQU  $D3D6  *   ;CHECK ON STACK POINTER
D7D2:              61 NEWSTT  EQU  $D7D2      ;JUMP TO NORMAL MONITOR GOSUB
D9A6:              62 REMN    EQU  $D9A6      ;Y=BYTES LEFT THIS STATEMENT
DD7B:              63 FRMEVL  EQU  $DD7B      ;PUSH VALUE IN FAC
E752:              64 GETADR  EQU  $E752      ;USE FAC AS GOTO POINTER
D959:              65 WNGOTO  EQU  $D959      ;WITHIN GOTO LOGIC
934A:              66 *          TO FIND REQUESTED LINE
DA0C:              67 LINGET  EQU  $DA0C      ;GET THE LINE NUMBER
934A:              68 *
934A:              69 *  SAVE FIRST 3 BYTES OF CHRGET
934A:              70 *
934A:A5 B2         71 BEGIN   LDA  CHRGET+1
934C:8D F9 94      72         STA  RESTRB2+1
934F:A5 B3         73         LDA  CHRGET+2
9351:8D FD 94      74         STA  RESTRB3+1
9354:A5 B1         75         LDA  CHRGET
9356:8D F5 94      76         STA  RESTRB1+1
9359:C9 4C         77         CMP  #OPJMP     ;CHRGET BEGIN WITH JMP
935B:D0 12         78         BNE  FIX        ;NO-ASSUM STANDARD CHRGET
935D:18            79         CLC             ;YES-ASSUM ALREADY HOOKED
935E:A9 06         80         LDA  #6         ;AND THE FIRST 6 POSITIONS
9360:65 B2         81         ADC  CHRGET+1   ;OF THAT HOOK IS THE
9362:8D 97 94      82         STA  RETURN+1   ;STANDARD CHRGET ROUTINE
9365:A5 B3         83         LDA  CHRGET+2   ;THEREFORE SET THIS
9367:8D 98 94      84         STA  RETURN+2   ;PROGRAM TO RETURN TO
936A:90 03         85         BCC  FIX        ;THE HOOK PLUS 6
936C:EE 98 94      86         INC  RETURN+2
936F:              87 *
936F:              88 * SET CHRGET AND AMPER HOOKS
936F:              89 *
936F:A9 4C         90 FIX     LDA  #OPJMP     ;SET JMP OPCODE
9371:85 B1         91         STA  CHRGET
9373:8D F5 03      92         STA  AVECTOR
9376:AD F6 03      93         LDA  AVECTOR+1 ;SAVE OLD AMPER ADDR
9379:8D 11 95      94         STA  AMPJMP+1
937C:AD F7 03      95         LDA  AVECTOR+2
937F:8D 12 95      96         STA  AMPJMP+2
9382:A9 D2         97         LDA  #>AMPER    ;SET NEW AMPER ADDR
9384:8D F6 03      98         STA  AVECTOR+1
```

A FAST.GOTO Processor for Applesoft (Cont.)

Statements **411-413** make possible the new feature of having named GOTO's and GOSUB's. First the expression is resolved, and then the results are stored in the same place as does the **LINGET** routine in Applesoft. This routine could have been used to resolve all line numbers, but it is considerably slower than the **LINGET** routine. After getting the line number, control passes to statement **389** to resolve the GOTO logic.

Statements **421-436** set aside necessary space for a 65 entry index. The reason for 65 instead of 64 entries is to allow for the possibility of the highest line number being equal to a power of 2. In this case the final results at statement **406** could be as high as 64.

Statements **438-455** contain several equates that determine the value that should be in statement **27**, depending on the location at which the FAST.GOTO processor should execute.

While you are all "pumped up", why not give the FAST.GOTO processor a chance to speed up your favorite Applesoft programs? Included with this article is a small Applesoft program showing how named GOTO's and GOSUB's can be used. Also included is an assembly listing of the FAST.GOTO processor.

```
9387:A9 94       99          LDA  #<AMPER
9389:8D F7 Ø3   1ØØ          STA  AVECTOR+2
938C:A5 73      1Ø1          LDA  MEMSIZ    ;SAVE
938E:8D ED 94   1Ø2          STA  RSTHIM1+1 ;CURRENT
9391:A5 74      1Ø3          LDA  MEMSIZ+1  ;MEMSIZ
9393:8D F1 94   1Ø4          STA  RSTHIM2+1 ;VALUE
9396:A9 64      1Ø5          LDA  #>ENTRY   ;SET
9398:85 B2      1Ø6          STA  CHRGET+1  ;CHRGET
939A:A9 94      1Ø7          LDA  #<ENTRY   ;HOOK
939C:85 B3      1Ø8          STA  CHRGET+2  ;LOCATION
939E:           1Ø9 *
939E:           11Ø *   CHANGE MEMSIZ AND FRETOP IF > LOCATION OF ENTRY
939E:A9 94      111          LDA  #<ENTRY-1
93AØ:C5 74      112          CMP  MEMSIZ+1
93A2:9Ø ØA      113          BCC  FIX1Ø
93A4:FØ Ø2      114          BEQ  FIX5
93A6:BØ 12      115          BCS  FIX15     ;DONT CHANGE
93A8:A9 63      116 FIX5     LDA  #>ENTRY-1
93AA:C5 73      117          CMP  MEMSIZ
93AC:BØ ØC      118          BCS  FIX15     ;DONT CHANGE
93AE:A9 94      119 FIX1Ø    LDA  #<ENTRY-1
93BØ:85 74      12Ø          STA  MEMSIZ+1
93B2:85 7Ø      121          STA  FRETOP+1
93B4:A9 63      122          LDA  #>ENTRY-1
93B6:85 73      123          STA  MEMSIZ
93B8:85 6F      124          STA  FRETOP
93BA:           125 *
93BA:           126 * FIND HIGHEST LINE NUMBER IN THE APPLESOFT PROGRAM
93BA:           127 *
93BA:AØ Ø1      128 FIX15    LDY  #1
93BC:A5 67      129          LDA  TXTTAB    ;BEGIN OF APPLESOFT PROGRAM
93BE:A6 68      13Ø          LDX  TXTTAB+1
93CØ:85 9B      131 LNCT     STA  WRKPTR
93C2:86 9C      132          STX  WRKPTR+1
93C4:B1 9B      133          LDA  (WRKPTR),Y ;END OF PROGRAM
93C6:FØ 17      134          BEQ  LNCT3     ;YES
93C8:C8         135          INY
93C9:B1 9B      136          LDA  (WRKPTR),Y ;SAVE
93CB:8D 62 94   137          STA  NBRLN     ;CURRENT
93CE:C8         138          INY            ;LINE
93CF:B1 9B      139          LDA  (WRKPTR),Y ;NUMBER
93D1:8D 63 94   14Ø          STA  NBRLN+1
93D4:88         141          DEY
93D5:88         142          DEY
93D6:B1 9B      143          LDA  (WRKPTR),Y ;POINT
93D8:AA         144          TAX            ;TO
93D9:88         145          DEY            ;NEXT
93DA:B1 9B      146          LDA  (WRKPTR),Y ;LINE
93DC:C8         147          INY
93DD:DØ E1      148          BNE  LNCT      ;CONT LOOKING FOR END OF PGM
93DF:           149 *
93DF:           15Ø *
93DF:AØ Ø6      151 LNCT3    LDY  #6        ;DIVIDE
93E1:18         152 LNCT4    CLC            ;HIGHEST
93E2:6E 63 94   153          ROR  NBRLN+1   ;LINE
93E5:6E 62 94   154          ROR  NBRLN     ;NUMBER
93E8:88         155          DEY            ;BY
93E9:DØ F6      156          BNE  LNCT4     ;64
93EB:           157 *
93EB:           158 * FIND LINE NUMBER RANGE PER TABLE ENTRY.  THE RANGE
93EB:           159 * WILL BE STORED IN INSTRUCTIONS INC AND INC2.
93EB:           16Ø * DURING SAME PROCESS DETERMINE HOW MANY TIMES A
93EB:           161 * LINE NUMBER MUST BE DIVIDED BY 2 TO CONVERT TO
93EB:           162 * AN INDEX WITHIN THE TABLE.
93EB:           163 *
93EB:18         164 SETDIV   CLC            ;DETERMINE
93EC:2E 1D 94   165          ROL  INC+1     ;NEXT
93EF:2E 25 94   166          ROL  INC2+1    ;POWER
93F2:EE 59 95   167          INC  DIV+1     ;OF
93F5:AD 25 94   168          LDA  INC2+1    ;TWO
93F8:CD 63 94   169          CMP  NBRLN+1
93FB:9Ø EE      17Ø          BCC  SETDIV
93FD:FØ Ø2      171          BEQ  SETDIV2
93FF:BØ ØB      172          BCS  HAVEDIV
94Ø1:AD 1D 94   173 SETDIV2 LDA  INC+1
94Ø4:CD 62 94   174          CMP  NBRLN
94Ø7:9Ø E2      175          BCC  SETDIV
94Ø9:A5 67      176 HAVEDIV LDA  TXTTAB
94ØB:A6 68      177          LDX  TXTTAB+1
94ØD:           178 *
94ØD:           179 * CONSTRUCT A TABLE OF LOCATIONS WHERE EACH
94ØD:           18Ø * RANGE OF LINE NUMBERS BEGIN.
94ØD:           181 *
94ØD:AC 7D 95   182 BLD1     LDY  LNXDSP
941Ø:99 7E 95   183          STA  LOADR,Y
9413:48         184          PHA
9414:8A         185          TXA
9415:99 BF 95   186          STA  HIADR,Y
9418:18         187          CLC
9419:AD 6Ø 94   188          LDA  WRKLN
941C:69 Ø1      189 INC      ADC  #1            ;THIS INSTR MODIFIED
941E:8D 6Ø 94   19Ø          STA  WRKLN
9421:AD 61 94   191          LDA  WRKLN+1
9424:69 ØØ      192 INC2     ADC  #Ø            ;THIS INSTR MODIFIED
9426:8D 61 94   193          STA  WRKLN+1
9429:68         194          PLA
942A:85 9B      195 BLD2     STA  WRKPTR
942C:86 9C      196          STX  WRKPTR+1
942E:AØ Ø1      197          LDY  #1
943Ø:B1 9B      198          LDA  (WRKPTR),Y
```

```
9432:FØ 19                BEQ  BLD3      ;END OF PROGRAM
9434:C8                   INY
9435:B1 9B                LDA  (WRKPTR),Y
9437:AA                   TAX
9438:C8                   INY
9439:B1 9B                LDA  (WRKPTR),Y
943B:CD 61 94             CMP  WRKLN+1
943E:FØ 1B                BEQ  BLD5
9440:BØ ØC                BCS  BLD4
9442:AØ Ø1       BLD2A    LDY  #1
9444:B1 9B                LDA  (WRKPTR),Y
9446:AA                   TAX
9447:88                   DEY
9448:B1 9B                LDA  (WRKPTR),Y
944A:4C 2A 94             JMP  BLD2
944D:60          BLD3     RTS
944E:A5 9B       BLD4     LDA  WRKPTR
945Ø:A6 9C                LDX  WRKPTR+1
9452:EE 7D 95             INC  LNXDSP
9455:4C ØD 94             JMP  BLD1
9458:8A          BLD5     TXA
9459:CD 6Ø 94             CMP  WRKLN
945C:BØ FØ                BCS  BLD4
945E:9Ø E2                BCC  BLD2A
946Ø:            *
946Ø:            * WORKAREAS USED DURING HOUSEKEEPING
946Ø:ØØ ØØ       WRKLN    DW   Ø
9462:ØØ ØØ       NBRLN    DW   Ø
                 ************************************************************
9464:            *     END OF HOUSE KEEPING -
9464:            *     HOUSEKEEPING NO LONGER NEEDED
                 ************************************************************
9464:            *
9464:            * THE FOLLOWING ROUTINE BECOMES PART OF THE APPLESOFT
9464:            * PARSER.  IT LOOKS FOR GOTO AND GOSUB TOKENS THAT
9464:            * DO NOT FOLLOW ON OR ONERR TOKENS.  WHEN FOUND
9464:            * THE GOTO OR GOSUB IS SAVED AND THEN REPLACED WITH
9464:            * AN AMPERSAND TOKEN.
9464:            *
9464:            * IN ORDER TO AVOID CAUSING A SYNTAX ERROR, AN EDIT
9464:            * MUST BE DONE TO AVOID CHANGING GOTO AFTER AN IF
9464:            * THAT IS NOT PRECEEDED WITH A THEN.
9464:            *
9464:            * SUGGESTION FOR APPLESOFT PROGRAMMERS:
9464:            *   TO GET MORE BENEFIT FROM FAST.GOTO
9464:            *   REMEMBER TO DO THE FOLLOWING.
9464:            *   WHEN USING GOTO AFTER AN IF, USE
9464:            *      IF .......THEN GOTO ...
9464:            *
9464:            *   IF POSSIBLE USE MULTIPLE IF'S INSTEAD OF
9464:            *      ON .......GOTO OR GOSUB
9464:            *
9464:E6 B8       ENTRY    INC  TXTPTR
9466:DØ Ø2                BNE  ENTRYØ
9468:E6 B9                INC  TXTPTR+1
946A:A5 B8       ENTRYØ   LDA  TXTPTR
946C:8D 75 94             STA  ENTRY1+1
946F:A5 B9                LDA  TXTPTR+1
9471:8D 76 94             STA  ENTRY1+2
9474:AD Ø5 Ø2    ENTRY1   LDA  $205      ;MODIFIED FROM ABOVE
9477:1Ø 2Ø                BPL  NOTOKEN   ;NOT AN APPLESOFT TOKEN
9479:C9 AB       ENTRY2   CMP  #TKGOTO   ;IS IT GOTO?
947B:FØ 26                BEQ  HAVEIT    ;YES, GO PROCESS IT
947D:C9 AD                CMP  #TKIF     ;REMEMBER IF TOKEN
947F:FØ 1Ø                BEQ  SETSW     ;SO CAN LOOK FOR THEN
9481:C9 C4                CMP  #TKTHEN   ;IF HAVE THEN OK TO
9483:FØ 1A                BEQ  CLRIT     ;CHANGE GOTO BEHIND IF
9485:C9 BØ                CMP  #TKGOSUB  ;IS IT GOSUB?
9487:FØ 1A                BEQ  HAVEIT    ;YES
9489:C9 B4                CMP  #TKON     ;ON TOKEN
948B:FØ Ø4                BEQ  SETSW     ;YES
948D:C9 A5                CMP  #TKONERR  ;ONERR TOKEN
948F:DØ Ø5                BNE  RETURN    ;CONT NORMAL PROC
9491:            *
9491:A9 Ø1       SETSW    LDA  #1
9493:8D 7C 95    CLRSW    STA  SW
9496:4C B7 ØØ    RETURN   JMP  CHRGET+6  ;IF CHRGET WAS ALREADY
9499:            * HOOKED THIS JMP IS MODIFIED TO JMP TO HOOKED LOCATION
9499:            * PLUS 6.
9499:            *
9499:FØ F8       NOTOKEN  BEQ  CLRSW     ;END, CLEAR SW
949B:C9 3A                CMP  #ENDSTMT  ;END OF STMT?
949D:DØ F7                BNE  RETURN    ;NO
949F:A9 ØØ       CLRIT    LDA  #Ø        ;YES,
94A1:FØ FØ                BEQ  CLRSW     ;GO CLEAR SW
94A3:            *
94A3:8D C3 94    HAVEIT   STA  HAVEIT3+1 ;SAVE TOKEN
94A6:A9 ØØ                LDA  #Ø        ;FROM PARSER
94A8:CD 7C 95             CMP  SW        ;IF SW IS ON, CLEAR IT
94AB:DØ E6                BNE  CLRSW     ;AND PASS TOKEN
94AD:CD D6 94             CMP  AMPER1+1  ;STILL HAVE THE PREVIOUS
94BØ:            *  GOTO/GOSUB TOKEN AS AN AMPERSAND
94BØ:FØ Ø6                BEQ  HAVEIT2   ;NO
94B2:            *
94B2:            *   RESET PREVIOUS GOTO/GOSUB TOKEN
94B2:            *
```

```
94B2:AD D6 94             LDA   AMPER1+1
94B5:20 CE 94             JSR   ENTRY3     ;RESTORE TOKEN
94B8:            *
94B8:            *   RESET DONE
94B8:            *
94B8:            * SAVE TOKEN LOCATION AND TOKEN
94B8:            *
94B8:A5 B8       HAVEIT2  LDA   TXTPTR
94BA:8D CF 94             STA   ENTRY3+1
94BD:A5 B9                LDA   TXTPTR+1
94BF:8D D0 94             STA   ENTRY3+2
94C2:A9 00       HAVEIT3  LDA   #0         ;MODIFIED ABOVE
94C4:8D D6 94             STA   AMPER1+1   ;SAVE TOKEN
94C7:A9 AF                LDA   #TKAMPER   ;CHANGE TO AMPER TOKEN
94C9:20 CE 94             JSR   ENTRY3
94CC:D0 C8                BNE   RETURN

94CE:8D 05 02    ENTRY3   STA   $205       ;MODIFIED ABOVE
94D1:60                   RTS
94D2:            *

94D2:            * AMPER SUBROUTINE
                 ***********************************************************************
94D2:            *
94D2:            *   ONLY HANDLE &'S SET ABOVE AND & HOME FROM THE
94D2:            *   APPLESOFT PROGRAM.  ALL OTHER &'S ARE PASSED TO
94D2:            *   THE NEXT AMPER ROUTINE.
94D2:            *

94D2:8D DC 94    AMPER    STA   AMPOUT+1
94D5:A9 00       AMPER1   LDA   #0         ;CONTAINS AMPEROP
94D7:C9 00                CMP   #0         ;FROM PARSER?
94D9:D0 38                BNE   AMPER2     ;YES
94DB:A9 00       AMPOUT   LDA   #0
94DD:C9 97                CMP   #TKHOME    ;SHOULD FAST.GOTO BE REMOVED?
94DF:D0 2F                BNE   AMPJMP     ;NO
94E1:48                   PHA
94E2:AD D6 94             LDA   AMPER1+1   ;ANY TOKEN TO
94E5:C9 00                CMP   #0         ;BE RESTORED
94E7:F0 03                BEQ   RSTHIM1    ;NO
94E9:20 CE 94             JSR   ENTRY3     ;YES
94EC:A9 00       RSTHIM1  LDA   #0         ;RESTORE
94EE:85 73                STA   MEMSIZ     ;ORIGINAL
94F0:A9 00       RSTHIM2  LDA   #0         ;MEMSIZ
94F2:85 74                STA   MEMSIZ+1   ;VALUE
94F4:A9 00       RESTRB1  LDA   #0         ;RESTORE
94F6:85 B1                STA   CHRGET     ;APPLESOFT
94F8:A9 00       RESTRB2  LDA   #0         ;PARSER
94FA:85 B2                STA   CHRGET+1
94FC:A9 00       RESTRB3  LDA   #0         ;SUBROUTINE
94FE:85 B3                STA   CHRGET+2
9500:AD 11 95             LDA   AMPJMP+1   ;RESTORE
9503:8D F6 03             STA   AVECTOR+1  ;ORIGINAL
9506:AD 12 95             LDA   AMPJMP+2   ;AMPER
9509:8D F7 03             STA   AVECTOR+2  ;VECTOR
950C:68                   PLA
950D:4C B1 00             JMP   CHRGET
9510:4C F5 03    AMPJMP   JMP   $3F5       ;TO NORMAL AMPER ROUTINE
9513:            *
9513:            *  FIRST MAKE SURE AT SAME LOCATION AS THE TOKEN
9513:            * THAT WAS CHANGED TO AN AMPERSAND
9513:            *
9513:A4 B8       AMPER2   LDY   TXTPTR
9515:88                   DEY
9516:CC CF 94             CPY   ENTRY3+1
9519:D0 C0                BNE   AMPOUT     ;NO
951B:A6 B9                LDX   TXTPTR+1
951D:C0 FF                CPY   #$FF       ;HIGH BYTE NEED DEC.
951F:D0 01                BNE   AMPER3     ;NO
9521:CA                   DEX
9522:EC D0 94    AMPER3   CPX   ENTRY3+2
9525:D0 B4                BNE   AMPOUT
9527:20 CE 94             JSR   ENTRY3     ;RESTORE TOKEN
952A:            *
952A:C9 AB                CMP   #TKGOTO    ;IS IT GOTO
952C:F0 1A                BEQ   GOTO       ;YES
952E:            *  ASSUM GOSUB
952E:A9 03       GOSUB    LDA   #$3        ;NORMAL GOSUB PROCEDURE
9530:20 D6 D3             JSR   STACK      ;RELOCATED FROM $D921
9533:A5 B9                LDA   TXTPTR+1   ;STORE CURRENT TEXT POINTERS
9535:48                   PHA
9536:A5 B8                LDA   TXTPTR
9538:48                   PHA
9539:A5 76                LDA   CURLIN+1   ;STORE CURRENT LINE NUMBER
953B:48                   PHA
953C:A5 75                LDA   CURLIN
953E:48                   PHA
953F:A9 B0                LDA   #TKGOSUB   ;IT NEEDS THIS
9541:48                   PHA
9542:20 48 95             JSR   GOTO       ;DO A GOTO
9545:4C D2 D7             JMP   NEWSTT     ;CONTINUE NORMAL GOSUB
9548:            *
9548:A9 00       GOTO     LDA   #0         ;CLEAR
954A:8D D6 94             STA   AMPER1+1   ;AMPEROP
954D:20 B7 00             JSR   CHRGET+6   ;GET FIRST CHAR OF LINE #
9550:B0 21                BCS   GOTO4      ;JMP IF NOT NUMERIC
9552:20 0C DA             JSR   LINGET     ;GET NUMERIC LINE NUMBER
```

```
9555:20 A6 D9   392 GOTO2   JSR   REMN       ;FIND NEXT STATEMENT
9558:A0 ØØ      393 DIV     LDY   #Ø         ;THIS INSTR MODIFIED
955A:           394 * TO CONTAIN THE NUMBER OF TIMES THAT THE LINE
955A:           395 * NUMBER NEEDS TO BE DIVIDED BY 2.
955A:A5 51      396         LDA   LINNUM+1
955C:8D 7D 95   397         STA   LNXDSP
955F:A5 5Ø      398         LDA   LINNUM
9561:18         399 DIVIDE  CLC
9562:6E 7D 95   4ØØ         ROR   LNXDSP
9565:6A         4Ø1         ROR   A
9566:88         4Ø2         DEY
9567:DØ FB      4Ø3         BNE   DIVIDE
9569:A8         4Ø4         TAY              ;Y NOW = Ø - 63
956A:BE BF 95   4Ø5         LDX   HIADR,Y
956D:B9 7E 95   4Ø6         LDA   LOADR,Y
957Ø:4C 59 D9   4Ø7         JMP   WNGOTO     ;CONTINUE IN GOTO LOGIC


9573:20 7B DD   4Ø9 GOTO4   JSR   FRMEVL     ;EVALUATE NEXT EXPRESSION
9576:20 52 E7   41Ø         JSR   GETADR     ;FIX GOTO LOCATION
9579:4C 55 95   411         JMP   GOTO2



957C:ØØ         414 SW      DFB   Ø          ;IF ON HAVE ON OR ONERR
957D:ØØ         415 LNXDSP  DFB   Ø


957E:           417 *  ALLOW FOR 65 INDEX POINTS
957E:Ø1 Ø1 Ø1   419 LOADR   DFB   1,1,1,1,1,1,1,1,1
9581:Ø1 Ø1 Ø1
9584:Ø1 Ø1 Ø1
9587:Ø1 Ø1 Ø1   42Ø         DFB   1,1,1,1,1,1,1,1,1
958A:Ø1 Ø1 Ø1
958D:Ø1 Ø1 Ø1
959Ø:Ø1 Ø1 Ø1   421         DFB   1,1,1,1,1,1,1,1,1
9593:Ø1 Ø1 Ø1
9596:Ø1 Ø1 Ø1
9599:Ø1 Ø1 Ø1   422         DFB   1,1,1,1,1,1,1,1,1
959C:Ø1 Ø1 Ø1
959F:Ø1 Ø1 Ø1
95A2:Ø1 Ø1 Ø1   423         DFB   1,1,1,1,1,1,1,1,1
95A5:Ø1 Ø1 Ø1
95A8:Ø1 Ø1 Ø1
95AB:Ø1 Ø1 Ø1   424         DFB   1,1,1,1,1,1,1,1,1
95AE:Ø1 Ø1 Ø1
95B1:Ø1 Ø1 Ø1
95B4:Ø1 Ø1 Ø1   425         DFB   1,1,1,1,1,1,1,1,1
95B7:Ø1 Ø1 Ø1
95BA:Ø1 Ø1 Ø1
95BD:Ø1 Ø1      426         DFB   1,1
95BF:Ø8 Ø8 Ø8   427 HIADR   DFB   8,8,8,8,8,8,8,8,8
95C2:Ø8 Ø8 Ø8
95C5:Ø8 Ø8 Ø8
95C8:Ø8 Ø8 Ø8   428         DFB   8,8,8,8,8,8,8,8,8
95CB:Ø8 Ø8 Ø8
95CE:Ø8 Ø8 Ø8
95D1:Ø8 Ø8 Ø8   429         DFB   8,8,8,8,8,8,8,8,8
95D4:Ø8 Ø8 Ø8
95D7:Ø8 Ø8 Ø8
95DA:Ø8 Ø8 Ø8   43Ø         DFB   8,8,8,8,8,8,8,8,8
95DD:Ø8 Ø8 Ø8
95EØ:Ø8 Ø8 Ø8
95E3:Ø8 Ø8 Ø8   431         DFB   8,8,8,8,8,8,8,8,8
95E6:Ø8 Ø8 Ø8
95E9:Ø8 Ø8 Ø8
95EC:Ø8 Ø8 Ø8   432         DFB   8,8,8,8,8,8,8,8,8
95EF:Ø8 Ø8 Ø8
95F2:Ø8 Ø8 Ø8
95F5:Ø8 Ø8 Ø8   433         DFB   8,8,8,8,8,8,8,8,8
95F8:Ø8 Ø8 Ø8
95FB:Ø8 Ø8 Ø8
95FE:Ø8 Ø8      434         DFB   8,8


                436 ***********************************************************
96ØØ:           437 * END OF PROGRAM
                438 ***********************************************************
96ØØ:           439 ENDPGM  EQU   *

Ø19C:           441 SIZE     EQU  ENDPGM-ENTRY ;ACTUAL SPACE REQUIRED
96ØØ:           442 *  AFTER HOUSEKEEPING IS COMPLETE.

Ø2B6:           444 PGMSIZE EQU   ENDPGM-BEGIN ;TRUE PROGRAM SIZE

934A:           446 ORG96ØØ EQU   $96ØØ-PGMSIZE ;NON-SUPER KRAM WITH
96ØØ:           447 *    MAXFILES=3

58FE:           449 ORG5BB4 EQU   $5BB4-PGMSIZE ;SUPER KRAM WITH 5 BUFFERS

BC4A:           451 ORGBFØØ EQU   $BFØØ-PGMSIZE ;WITH DOS MOVED TO RAM BOARD

                453 ***********************************************************

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```