NICK HAMPSHIRE
WITH RICHARD FRANKLIN AND CARL GRAHAM

THE COMMODORE 64 ROMs REVEALED

The Commodore 64 ROMs Revealed

Nick Hampshire
with Richard Franklin
and Carl Graham

COLLINS
8 Grafton Street, London W1

Collins Professional and Technical Books
William Collins Sons & Co. Ltd
8 Grafton Street, London W1X 3LA

First published in Great Britain by
Collins Professional and Technical Books 1985

British Library Cataloguing in Publication Data
Hampshire, Nick
Commodore 64 ROMs revealed.
1. Commodore 64 (Computer) 2. Machine codes
(Electronic computers)
I. Title
001.64'24 QA76.8.C64

ISBN 0-00-383087-X

Typeset by V & M Graphics Ltd, Aylesbury, Bucks
Printed and bound in Great Britain by
Mackays of Chatham, Kent

Contents

Preface

The Commodore 64 ROMs Revealed is the first of a six part series covering all topics concerning the use of the Commodore 64. This book gives a detailed insight into the workings of the two ROMs in the Commodore 64 that provide the machine's normal functioning.

The major part of the book is given to a fully commented source code version of these ROMs and will assist machine code programmers in getting the most out of the existing firmware for their own programs. The rest of the book contains useful information concerning the variables used by the firmware, and key entry points into the firmware.

The fully documented source code was effectively produced in six stages from the ROMs to the listing provided:

1. Disassembling the ROMs to disk files. As the full disassembly could not be loaded into memory at one time, logical split points were first ascertained.
2. Labelling all branches, jumps, and JSRs within the disassembled files. This was done using a simple two pass program written in Basic to replace any branch etc. addresses with labels.
3. Converting most of the storage addresses to variable names. The variable names are the ones that have appeared in Commodore's own published documentation.
4. Documenting and separating the tables from the true code. This was done laboriously line by line and tables were separated and formatted into source code form.
5. Testing assembly to make sure that there were no errors in the source files. When the files were assembled correctly the object file was then loaded into memory and verified against the resident ROMs. At this point it should be mentioned that we discovered the modifications that have been made to the Kernal in the most recent Commodore 64s. These have been listed, but due to the number of machines sold with the old ROM we did not change the source code.
6. Final assembly. This is what appears in this book.

Additional information about how to use the routines in the Basic and Kernal

ROMs can be found in Volume 2 (*Advanced Commodore 64 Basic Revealed*) and Volume 3 (*The Commodore 64 Kernal and Hardware Revealed*), both published by Collins.

Machine Differences

The code listed in this book was taken from a Commodore 64 produced at the beginning of 1984. This version had been available since the launch of the Commodore 64 until around mid 1984. From this date Commodore produced a version that has corrected a certain bug in the screen editor. In this new version the Basic ROM is exactly the same but there have been a few alterations to the Kernal ROM. They are as follows:

- A subroutine has been inserted at $E4D3 which reads as follows:

```
E4D3  STA $A9
E4D5  LDA #$01
E4D7  STA $AB
E4D9  RTS
```

  This used to be just the value $AA throughout.
- At $E4DA one command has been changed. This is used to clear a screen line and put the colour to the colour RAM. Previously this value was taken from the background colour location $D021; this has now been changed to the current colour code $0286, i.e. there is no need to poke the colour when pokeing to the screen to see the character – it will appear in the cursor colour.
- A subroutine has been changed at $E57C which reads as follows:

```
E57C  JSR $E9F0
E57F  LDA #$27
E581  INX
E582  LDY $D9,X
E584  BMI $E58C
E586  CLC
E587  ADC #$28
E589  INX
E58A  BPL $E582
E58C  STA $D5
E58E  JMP $EA24

E591  CPX $C9
E593  BE0 $E598
E595  JMP $E6ED
E598  RTS
E599  NOP
```

- The command at location $E621 has been changed from JSR $E6ED to JSR $E591.
- The routine at $EAØ7 has been modified to read as follows:

```
EA07  JSR $E4DA
EA0A  LDA #$20
EA0C  STA ($D1),Y
EA0E  DEY
EA0F  BPL $EA07
EA11  RTS
EA12  NOP
```

- The commands at locations $EF94 and $EF96 used to read:

```
EF94  STA $A9
EF96  RTS
```

They now read:

```
EF94  JMP $E4D3
```

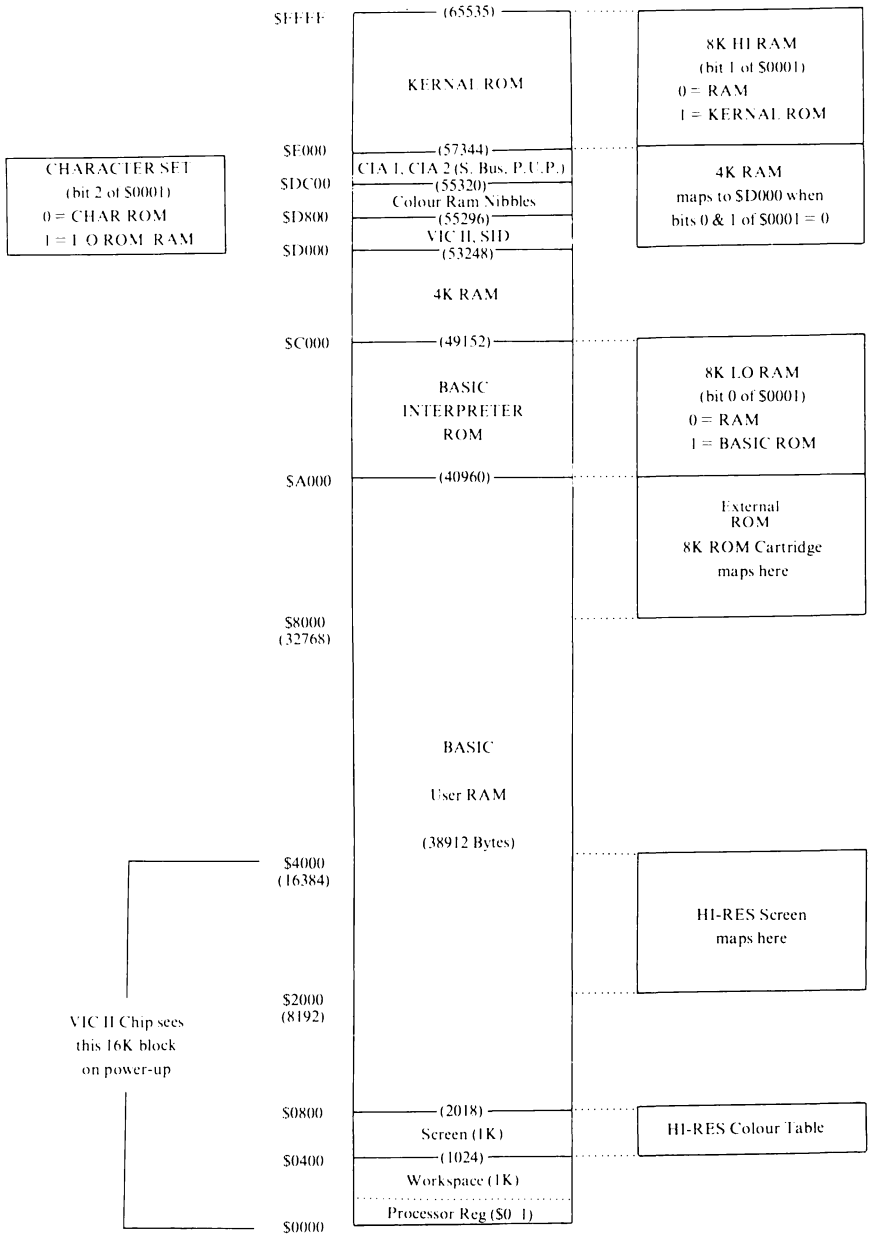
- The last modifications to the Kernal ROMs are:
  Location $E4AC has been changed from $5C to $81
  Location $FF8Ø has been changed from $ØØ to $Ø3

We believe that these last two are version bytes only.

Chapter One

Commodore 64 Memory Architecture Map

The following diagram shows where the different sections of ROM and RAM are to be found on the Commodore 64.



Chapter Two

Operating System Variable Storage Area

In this section the locations used by the Basic and Kernal ROMs for storing their variables have been listed and a brief explanation of what they hold is included.

Address (Hex)	(Dec)	*Location use*
0000	0	6510 DDR register
0001	1	6510 I/O memory and tape control
0003–0004	3–4	Float-Fix vector
0005–0006	5–6	Fix-Float vector
0007	7	Search character
0008	8	Scan quotes garbage collect flag
0009	9	TAB column save
000A	10	Load/verify flag
000B	11	Input buffer pointer/# of subscripts
000C	12	Default DIM flag
000D	13	Variable type: string or numeric
000E	14	Numeric type: integer or real
000F	15	DATA scan/LIST quote/memory flag
0010	16	Subscript/FNx flag
0011	17	INPUT/GET/READ flag
0012	18	ATN sign/Comparison eval flag
0013	19	INPUT prompt flag
0014–0015	20–21	Integer value (line # etc.)
0016	22	Pointer to string stack
0017–0018	23–24	Last temp string vector
0019–0021	25–33	Temporary string stack
0022–0025	34–37	Utility pointer area
0026–002A	38–42	Product area for multiplication
002B–002C	43–44	Pointer: start-of-Basic
002D–002E	45–46	Pointer: start-of-variables
002F–0030	47–48	Pointer: start-of-arrays
0031–0032	49–50	Pointer: end-of-arrays
0033–0034	51–52	Pointer: string storage
0035–0036	53–54	Utility string pointer
0037–0038	55–56	Pointer to top of Basic memory
0039–003A	57–58	Current Basic line number

003B-003C	59-60	Previous Basic line number
003D-003E	61-62	Pointer to statement for CONT
003F-0040	63-64	Current DATA line number
0041-0042	65-66	Current DATA address
0043-0044	67-68	Input vector
0045-0046	69-70	Current variable name
0047-0048	71-72	Pointer to current variable
0049-004A	73-74	Pointer to FOR/NEXT variable
004B-004C	75-76	Y-save; op-save; Basic pointer save
004D	77	Comparison symbol accumulator
004E-0053	78-83	Misc numeric work area
0054-0056	84-86	Jump vector for functions
0057-0060	87-96	Floating accumulators #3 & #4 (packed)
0061	97	FAC#1 exponent
0062-0065	98-101	FAC#1 mantissa
0066	102	FAC#1 sign
0067	103	Series evaluation constant pointer
0068	104	FAC#1 overflow
0069-006E	105	FAC#2 exponent
006A-006D	106-109	FAC#2 mantissa
006E	110	FAC#2 sign
006F	111	Sign comparison FAC#1 vs FAC#2
0070	112	FAC#1 rounding
0071-0072	113-114	Cassette buff length/series pointer
0073-008A	115-138	CHRGET subroutine
007A-007B	122-123	Basic text pointer
008B-008F	139-143	RND seed value (packed)
0090	144	Status word ST
0091	145	Stop key flag
0092	146	Temporary
0093	147	Load/verify flag
0094	148	Serial buffered char flag
0095	149	Serial buffered character
0096	150	Cassette sync number
0097	151	Temp for Basic
0098	152	Index to logical file
0099	153	Input device number
009A	154	Output device number
009B	155	Cassette parity
009C	156	Cassette dipole switch
009D	157	OS message flag
009E	158	Tape pass1 error log
009F	159	Tape pass2 error log corrected
00A0-00A2	160-162	Jiffy clock
00A3	163	Serial bit count/EOI flag

00A4	164	Cycle count
00A5	165	Cassette sync countdown
00A6	166	Cassette buffer pointer
00A7	167	Tape short count lsb/ RS232 input bit
00A8	168	Tape read error/ RS232 input bit count
00A9	169	Tape reading zeros/ RS232 start bit flag
00AA	170	Tape read mode/ RS232 input byte buffer
00AB	171	Tape short count msb/ RS232 input parity
00AC-00AD	172-173	Pointer: tape load address
00AE-00AF	174-175	Pointer: tape end address
00B0-00B1	176-177	Tape timing constants
00B2-00B3	178-179	Pointer: start of tape buffer
00B4	180	Tape timer flag/ RS232 output bit count
00B5	181	Tape EOT/ RS232 output bit
00B6	182	Tape read error/ RS232 output byte buffer
00B7	183	Length of filename
00B8	184	Current logical file number
00B9	185	Current secondary address
00BA	186	Current device number
00BB-00BC	187-188	Address of current filename
00BD	189	Tape write byte/ RS232 output parity
00BE	190	Tape read block count
00BF	191	Serial word buffer
00C0	192	Tape motor interlock
00C1-00C2	193-194	I/O start address
00C3-00C4	195-196	Kernal setup pointer
00C5	197	Last key pressed
00C6	198	Number of characters in keyboard buffer
00C7	199	Screen reverse flag
00C8	200	Pointer: end of line for input
00C9-00CA	201-202	Input cursor log (row column)
00CB	203	Current key pressed
00CC	204	Cursor flash flag
00CD	205	Cursor blink countdown
00CE	206	Character under cursor
00CF	207	On/ off blink flag
00D0	208	Screen/ keyboard input flag
00D1-00D2	209-210	Pointer: cursor position
00D3	211	Pointer: cursor column
00D4	212	Quote switch
00D5	213	40/ 80 current line length
00D6	214	Cursor row
00D7	215	Last inkey/ checksum/ buffer
00D8	216	Number of inserts outstanding
00D9-00F2	217-242	Screen line link flags

09F3-00F4	243-244	Pointer: colour memory
00F5-00F6	245-246	Pointer: keyboard table
00F7-00F8	247-248	RS232 input buffer pointer
00F9-00FA	249-250	RS232 output buffer pointer
00FB-00FE	251-254	Free zero page
00FF-010A	255-266	Float - ASCII work area
0100-013E	256-318	Tape error log
0100-01FF	256-511	Processor stack
0200-0258	512-600	Basic input buffer
0259-0262	601-610	Logical file table
0263-026C	611-620	Device number table
026D-0276	621-630	Secondary address table
0277-0280	631-640	Keyboard buffer
0281-0282	641-642	Pointer: start of RAM
0283-0284	643-644	Pointer: end of RAM
0285	645	Serial timeout flag
0286	646	Current colour code
0287	647	Colour under cursor
0288	648	Screen address msb
0289	649	Maximum size of keyboard buffer
028A	650	Key repeat flag
028B	651	Repeat speed counter
028C	652	Repeat delay counter
028D	653	Current shift combination
028E	654	Last shift pattern
028F-0290	655-656	Keyboard table setup pointer
0291	657	Keyboard shift mode
0292	658	Scroll enable flag
0293	659	RS232 control register
0294	660	RS232 command register
0295-0296	661-662	RS232 bit timing
0297	663	RS232 status byte
0298	664	RS232 number of bits to send
0299-029A	665-666	RS-232 baud rate
029B	667	RS232 input pointer to end
029C	668	RS232 input pointer to start
029D	669	RS232 output pointer to start
029E	670	RS232 output pointer to end
029F-02A0	671-672	IRQ save during tape I/O
02A1	673	NMI interrupt control
02A2	674	Timer A control log
02A3	675	Interrupt log
02A4	676	Timer A enable flag
02A5	677	Screen row marker
02C0-02FE	704-766	Sprite 11 block/free programming memory

The following section of memory is basically divided into three sections:

1. Basic interpreter vectors:

$0300 - Send error messages	(default = $E38B)
$0302 - Basic warm start	(default = $A483)
$0304 - Crunch text to tokens	(default = $A57C)
$0306 - Convert tokens to text	(default = $A71A)
$0308 - Start new Basic code	(default = $A7E4)
$030A - Perform arithmetic function	(default = $AE86)

Register parsing for SYS command:

$030C - .A register
$030D - .X register
$030E - .Y register
$030F - Processor status register

USR function jump:

$0310 - JMP followed by address in vector form
Default is JMP $B248

2. Kernal vectors:

$0314 - IRQ vector	(default = $EA31)
$0316 - Indirect on BRK instruction	(default = $FE66)
$0318 - NMI vector	(default = $FE47)
$031A - Open vector	(default = $F34A)
$031C - Close vector	(default = $F291)
$031E - Set input device vector	(default = $F20E)
$0320 - Set output device vector	(default = $F250)
$0322 - Restore I/O vector	(default = $F333)
$0324 - Input vector	(default = $F157)
$0326 - Output vector	(default = $F1CA)
$0328 - Test STOP vector	(default = $F6ED)
$032A - Get vector	(default = $F13E)
$032C - Abort I/O vector	(default = $F32F)
$032E - STOP/RESTORE vector	(default = $FE66)
$0330 - Load vector	(default = $F4A5)
$0332 - Save vector	(default = $F5ED)

3. Cassette buffer: $033C–$03FB - This is the buffer for reading from and writing to the cassette drive. The list of address contents continues as below:

0340–037E	832–894	(Sprite 13)
0380–03BE	896–958	(Sprite 14)
03C0–03FE	960–1022	(Sprite 15)

Chapter Three

Commodore 64 ROM Memory Map: Key Entry Points

In this section, the most useful entry points to the ROM routines have been listed with a brief description of what the routines do. For further information refer to the documented code to be found on the Commodore 64.

Entry (hex)	*Description of operation*
A000	ROM control vectors
A00C	Keyword action vectors
A052	Function vectors
A080	Operator vectors
A09E	Keywords
A19E	Error messages
A328	Error message vectors
A365	Misc messages
A38A	Scan stack for FOR/GOSUB activity
A3B8	Move a chunk of memory
A3FB	Check for space in stack
A408	Check for space in memory for program line or variables
A435	'out of memory'
A437	Send error message store in .X
A469	BREAK entry
A474	Output 'READY.' and then warm start
A480	Basic warm start
A49C	Insert or remove line of Basic
A533	Re-chain Basic program
A560	Input a line into input buffer
A579	Convert input line to tokenised form
A613	Find address of program line
A642	Perform 'NEW'
A65E	Perform 'CLR'
A68E	Set charget pointer to start of Basic program
A69C	Perform 'LIST'
A717	Convert tokenised line to text and print it

Entry (hex)	*Description of operation*
A742	Perform 'FOR'
A7AE	Main interpreter loop
A7ED	Execute Basic statement
A81D	Perform 'RESTORE'
A82C	Check STOP key and 'BREAK' if so
A82F	Perform 'STOP'
A831	Perform 'END'
A857	Perform 'CONT'
A871	Perform 'RUN'
A883	Perform 'GOSUB'
A8A0	Perform 'GOTO'
A8D2	Perform 'RETURN'
A8F8	Perform 'DATA'
A906	Find next statement
A928	Perform 'IF'
A93B	Perform 'REM'
A94B	Perform 'ON'
A96B	Convert ASCII # to 2 byte format
A9A5	Perform 'LET'
AA80	Perform 'PRINT#'
AA86	Perform 'CMD'
AAA0	Perform 'PRINT'
AAF8	Perform 'TAB' & 'SPC'
AB1E	Output string in (Y,A)
AB3B	Print format character
AB4D	Bad input
AB7B	Perform 'GET'
ABA5	Perform 'INPUT#'
ABBF	Perform 'INPUT'
ABF9	Prompt and input line

AC06 Perform 'READ'
ACFC Input error messages
AD1E Perform 'NEXT'
AD78 Type match check

AD9E Evaluate expression
AEA8 Constant 'PI'
AEF1 Evaluate within brackets
AEF7 Scan past ')'
AEFA Scan past '('
AEFD Scan Past ','
AEFF Comma
AF08 Syntax error
AF14 Check range

AF28 Search for variable
AF84 Get time into FAC#1
AFA7 Setup 'FN' reference
AFE6 Perform 'OR'
AFE9 Perform 'AND'
B016 Perform comparison
B02E Compare strings
B081 Perform 'DIM'
B08B Locate variable

B113 Check on alphabetic
B11D Create variable
B194 Calculate array pointer
B1A5 Constant '32768'
B1B2 Convert float to fix
B1D1 Set up array dimension
B245 'bad subscript'
B248 'illegal quantity'

B34C Compute array size
B37D Perform 'FRE'
B391 Fix-float
B39E Perform 'POS'
B3A6 Test for direct mode
B3B3 Perform 'DEF FN'
B3E1 Check 'FN' syntax
B3F4 Perform 'FN'
B465 Perform 'STR$'
B475 Calculate string vector
B487 Scan and set up string
B4F4 Allocate space for string

B526 Garbage collect
B5BD Check salvageability
B606 Collect string
B63D Concatenate strings (+)

B67A Save string details
B6A3 Discard unwanted string
B6DB Clean descriptor stack
B6EC Perform 'CHR$'
B700 Perform 'LEFT$'
B72C Perform 'RIGHT$'
B737 Perform 'MID$'
B761 Pull string parameters
B77C Perform 'LEN'
B782 Exit string-mode
B78B Perform 'ASC'
B79B Get 1 byte parameter into .X
B7AD Perform 'VAL'
B7EB Get parameters for 'POKE' & 'WAIT'
B7F7 Convert FAC#1 to 2 byte integer
B80D Perform 'PEEK'
B824 Perform 'POKE'
B82D Perform 'WAIT'
B849 Add 0.5
B850 Subtract-from
B853 Perform '–'
B86A Perform '+'
B947 Complement FAC#1
B97E 'overflow'
B983 Multiply by zero byte
B9BC Constants for 'LOG'
B9EA Perform 'LOG'
BA2B Perform '*'
BA59 Multiply-a-bit
BA8C Unpack memory to FAC#2
BAB7 Adjust FAC#1 #2
BAD4 Underflow overflow
BAE2 Multiply by 10
BAF9 Constant 10
BAFE Divide by 10
BB12 Perform '/'
BBA2 Unpack memory to FAC#1
BBC7 FAC#1 to memory
BBD4 Pack FAC#1 to memory
BBFC FAC#2 to FAC#1
BC0C FAC#1 to FAC#2
BC1B Round FAC#1
BC2B Get sign of FAC#1
BC39 Perform 'SGN'
BC58 Perform 'ABS'
BC5B Compare memory to FAC#1

BC9B Float-fixed
BCCC Perform 'INT'
BCF3 Convert ASCII to floating point
BD7E Get ASCII digit
BDC2 Print 'IN #'
BDCD Print 2 byte integer
BDDD Convert floating point number to ASCII
BF16 Decimal constants
BF3A TI constants
BF71 Perform 'SQR'
BF7B Perform 'power'
BFB4 Invert sign of FAC#1
BFED Perform 'EXP'
E043 Series eval 1
E059 Series eval 2
E08D Constants for 'RND'
E097 Perform 'RND'
E0F9 Error handler for I/O

Basic-Kernal link:
E10C OUTPUT
E112 INPUT
E118 SET OUTPUT DEVICE
E11E SET INPUT DEVICE
E124 GET A CHARACTER

E12A Perform 'SYS'
E156 Perform 'SAVE'
E165 Perform 'VERIFY'
E168 Perform 'LOAD'
E1BE Perform 'OPEN'
E1C7 Perform 'CLOSE'
E1D4 Get parameters for 'LOAD', 'VERIFY' & 'SAVE'
E206 Check default parameters
E20E Check for comma
E219 Get parameters for 'OPEN' & 'CLOSE'
E264 Perform 'COS'
E26B Perform 'SIN'
E2B4 Perform 'TAN'
E30E Perform 'ATN'
E37B Basic warm restart
E394 Basic cold start
E3A2 Charget for zero page
E3BF Initialise Basic
E422 Output power-up message
E447 Page 3 vectors
E453 Initialise vectors
E45F Power-up message
E4E0 Wait for commodore key or 8.5 secs
E4EC RS232 Baud rate table
E500 (FFF3) Get address of 6526
E505 (FFED) Get screen size
E50A (FFF0) Read/set cursor position
E518 Initialise screen & keyboard
E544 Clear screen
E566 Home cursor
E56C Set screen pointers
E5A0 Set I/O defaults
E5B4 Input from keyboard until carriage return
E632 Input from screen
E684 Quote test
E691 Setup screen print
E6B6 Advance cursor
E6ED Retreat cursor
E701 Back into previous line
E716 Output to screen
E87C Go to next line
E891 Perform <return>
E8A1 Check line decrement
E8B3 Check line increment
E8CB Set colour code
E8DA Colour code table
E8EA Scroll screen
E956 Open up line on screen
E9C8 Move a screen line
E9E0 Synchronise colour transfer
E9F0 Set start-of-line
E9FF Clear screen line
EA13 Wait for cursor and put character on screen
EA24 Synchronise colour pointer
EA31 Interrupts - clock etc
EA87 (FF9F) General keyboard scan
EB79 Table select vectors for keyboard
EB81 1 unshifted keys
EBC2 2 shifted
EC03 3 Commodore key
EC44 Upper/lower shift select

EC4F Set graphics/text mode
EC78 4 control key
ECB9 Initial VIC chip settings
ECE7 LOAD(cr)RUN(cr)
ECF0 Screen line lsbs
ED09 (FFB4) 'Talk'
ED0C (FFB1) 'Listen'
ED40 Send to serial bus
EDB2 Serial timeout
EDB9 (FF93) 'Listen SA'
EDBE Clear ATN
EDC7 (FF96) 'Talk SA'
EDCC Wait for clock
EDDD (FFA8) Send to serial
EDEF (FFAB) 'Untalk'
EDFE (FFAE) 'Unlisten'
EE13 (FFA5) Receive from serial
EE85 Serial clock on
EE8E Serial clock off
EE97 Serial output '1'
EEA0 Serial output '0'
EEA9 Get serial in & clock
EEB3 Delay 1 ms
EEBB Send over RS232
EF06 Send new RS232 byte
EF2E No-DSR error
EF31 No-CTS error
EF3B Disable timer
EF4A Compute bit count
EF59 Receive from RS232
EF7E Setup to receive
EFC5 Receive parity error
EFCA Receive overflow
EFCD Receive break
EFD0 Framing error
EFE1 Submit to RS232
F00D No-DSR error
F017 Send to RS232 buffer
F04D Input from RS232
F086 GET from RS232
F0A4 Check serial bus idle
F0BD I/O error messages
F12B Print message if required
F13E (FFE4) 'Get'
F14E ..from RS232
F157 (FFCF) 'Input'
F199 Get.. tape/serial/RS232
F1CA (FFD2) 'Output'
F1DD ..to tape
F20E (FFC6) Set input
F250 (FFC9) Set output
F291 (FFC3) 'Close'
F30F Find file
F31F Set file values
F32F (FFE7) Close all files
F333 (FFCC) Reset default I/O
F34A (FFC0) 'Open'
F3D5 Send SA
F409 Open RS232
F49E (FFD5) 'Load'
F5AF 'searching'
F5C1 Print filename
F5D2 'loading/verifying'
F5DD (FFD8) 'Save'
F68F Print 'saving'
F69B (FFEA) Increment clock
F6BC Log PIA key reading
F6DD (FFDE) Get time
F6E4 (FFDB) Set time
F6ED (FFE1) Test stop key
F6FB Output error messages
F72C Find any tape header
F76A Write tape header
F7D0 Get buffer address
F7D7 Set buffer start/end pointers
F7EA Find specific header
F80D Bump tape pointer
F817 'press play..'
F82E Check tape status
F838 'press record..'
F841 Initiate tape read
F864 Initiate tape write
F875 Common tape code
F8D0 Check tape stop
F8E2 Set read timing
F92C Tape read subroutines
FA60 Store tape chars
FB8E Reset pointer
FB97 New character setup
FBA6 Tape write subroutines
FBC8 Write data to tape
FBCD IRQ entry point
FC57 Write tape leader
FC93 Restore normal IRQ
FCB8 Set IRQ vector
FCCA Kill tape motor
FCD1 Check r/w pointer
FCDB Bump r/w pointer

FCE2	Cold start entry
FD02	Check for cartridge
FD10	8-ROM mask
FD15	(FF8A) Kernal reset
FD1A	(FF8D) Kernal move
FD30	Vectors
FD50	(FF87) Initialise system constants
FD9B	IRQ vectors
FDA3	(FF84) Initialise I/O
FDDD	Enable timer
FDF9	(FFBD) Set filename data
FE00	(FFBA) Set file data
FE07	(FFB7) Get status
FE18	Flag status
FE1C	Set status
FE21	(FFA2) Set timeout
FE25	(FF99) Read/set top of memory
FE27	Read top of memory
FE2D	Set top of memory
FE34	(FF9C) Read/set bottom of memory
FE43	NMI entry point
FE66	Warm start
FEB6	Reset IRQ & exit
FEBC	Interrupt exit
FEC2	RS232 timing table
FED6	NMI RS232 in
FF07	NMI RS232 out
FF43	Fake IRQ
FF48	IRQ entry point
FF81	Jump table ()
FFFA	Hardware vectors

Chapter Four

Annotated Assembly of Commodore 64 ROM Routines

Variable Declarations

The variable declarations section has been produced to make it easier for you to follow the code without having to refer back to the memory maps earlier in this book. The variable names used are based on the names used by Commodore in their published documentation. Where no variable name was supplied, we have created our own.

```
LOC   CODE        LINE

0000                    * =$03
0003                ;
0003                ;BASIC VARIABLES
0003                ;
0003                ADRAY1 *=*+2          ;FLOAT-INT VECTOR
0005                ADRAY2 *=*+2          ;INT-FLOAT VECTOR
0007                CHARAC *=*+1          ;SEARCH CHARACTER
0008                ENDCHR *=*+1          ;SCAN UNTIL FLAG
0009                TRMPOS *=*+1          ;COLUMN FROM LAST TAB
000A                VERCKB *=*+1          ;LOAD/VERIFY FLAG
000B                COUNTB *=*+1          ;INPUT BUFFER POINTER/#SUBSCRIPTS
000C                DIMFLG *=*+1          ;DEFAULT ARRAY DIMS
000D                VALTYP *=*+1          ;DATA TYPE (STRING/NUMERIC)
000E                INTFLG *=*+1          ;DATA TYPE (INT/FLOAT)
000F                DATSCN                ;DATA SCAN FLAG/LIST QUOTE
000F                GARBFL *=*+1          ;GARBAGE COLLECT FLAG
0010                SUBFLG *=*+1          ;SUBSCRIPT REF FLAG
0011                INPFLG *=*+1          ;INPUT/GET/READ FLAG
0012                TANSGN *=*+1          ;TAN SIGN FLAG
0013                INPRMT *=*+1          ;INPUT PROMPT FLAG
0014                LINNUM *=*+2          ;INTEGER VALUE
0016                TEMPPT *=*+1          ;TEMP STRING STACK POINTER
0017                LASTPT *=*+2          ;LAST TEMP STRING ADDRESS
0019                TEMPST *=*+9          ;TEMPORARY STRING STACK
0022                INDEX  *=*+4          ;UTILITY POINTER AREA
0026                RESHO  *=*+5          ;MULTIPLY PRODUCT AREA
002B                TXTTAB *=*+2          ;START OF BASIC POINTER
002D                VARTAB *=*+2          ;START OF VARIABLES POINTER
002F                ARYTAB *=*+2          ;START OF ARRAYS POINTER
0031                STREND *=*+2          ;END OF ARRAYS POINTER
0033                FRETOP *=*+2          ;STRING STORAGE POINTER
0035                FREPSC *=*+2          ;UTILITY STRING POINTER
0037                MEMTOP *=*+2          ;TOP OF MEMORY POINTER
0039                CURLIN *=*+2          ;CURRENT BASIC LINE NUMBER
003B                OLDLIN *=*+2          ;PREVIOUS BASIC LINE NUMBER
003D                OLDTXT *=*+2          ;CONT POINTER
```

```
LOC   CODE        LINE

003F              DATLIN *=*+2          ;CURRENT DATA LINE NUMBER
0041              DATPTR *=*+2          ;CURRENT DATA ITEM POINTER
0043              INPPTR *=*+2          ;INPUT VECTOR
0045              VARNAM *=*+2          ;CURRENT VARIABLE NAME
0047              VARPNT *=*+2          ;POINTER TO CURRENT VARIABLE
0049              FORPNT *=*+2          ;VARIABLE FOR FOR/NEXT
004B              BASTMP *=*+22         ;TEMP POINTER/DATA AREA
0061              FACEXP *=*+1          ;FAC#1 EXPONENT
0062              FACHO  *=*+4          ;FAC#1 MANTISSA
0066              FACSGN *=*+1          ;FAC#1 SIGN
0067              SGNFLG *=*+1          ;SERIES EVAL POINTER
0068              BITS   *=*+1          ;FAC#1 OVERFLOW
0069              ARGEXP *=*+1          ;FAC#2 EXPONENT
006A              ARGHO  *=*+4          ;FAC#2 MANTISSA
006E              ARGSGN *=*+1          ;FAC#1 SIGN
006F              ARISGN *=*+1          ;SIGN COMPARISON
0070              FACOV  *=*+1          ;FAC#1 UNDERFLOW
0071              FBUFPT *=*+2          ;CASSETTE BUFFER POINTER
0073              CHRGET *=*+6          ;SUBROUTINE:GET NEXT BYTE
0079              CHRGOT *=*+1          ; OF BASIC TEXT
007A              TXTPTR *=*+2          ;POINTER WITHIN SUROUTINE
007C                     *=*+15
008B              RNDX   *=*+5          ;RND SEED
0090              ;
0090              ;KERNAL VARIABLES
0090              ;
0090              STATUS *=*+1          ;I/O OPERATION STATUS BYTE
0091              STKEY  *=*+1          ;STOP KEY FLAG
0092              SVXT   *=*+1          ;TEMPORARY
0093              VERCK  *=*+1          ;LOAD OR VERIFY FLAG
0094              C3PO   *=*+1          ;IEEE BUFFERED CHAR FLAG
0095              BSOUR  *=*+1          ;CHAR BUFFER FOR IEEE
0096              SYNO   *=*+1          ;CASSETTE SYNC #
0097              XSAV   *=*+1          ;TEMP FOR BASIN
0098              LDTND  *=*+1          ;INDEX TO LOGICAL FILE
0099              DFLTN  *=*+1          ;DEFAULT INPUT DEVICE #
009A              DFLTO  *=*+1          ;DEFAULT OUTPUT DEVICE #
009B              PRTY   *=*+1          ;CASSETTE PARITY
009C              DPSW   *=*+1          ;CASSETTE DIPOLE SWITCH
009D              MSGFLG *=*+1          ;OS MESSAGE FLAG
009E              PTR1                  ;CASSETTE ERROR PASS1
009E              T1     *=*+1          ;TEMPORARY 1
009F              TMPC
009F              PTR2                  ;CASSETTE ERROR PASS2
009F              T2     *=*+1          ;TEMPORARY 2
00A0              TIME   *=*+3          ;24 HOUR CLOCK IN 1/60TH SECONDS
00A3              R2D2                  ;SERIAL BUS USAGE
00A3              PCNTR  *=*+1          ;CASSETTE STUFF
00A4              BSOUR1                ;TEMP USED BY SERIAL ROUTINE
00A4              FIRT   *=*+1
00A5              COUNT                 ;TEMP USED BY SERIAL ROUTINE
00A5              CNTDN  *=*+1          ;CASSETTE SYNC COUNTDOWN
00A6              BUFPT  *=*+1          ;CASSETTE BUFFER POINTER
00A7              INBIT                 ;RS-232 RCVR INPUT BIT STORAGE
00A7              SHCNL  *=*+1          ;CASSETTE SHORT COUNT
00A8              BITCI                 ;RS-232 RCVR BIT COUNT IN
00A8              RER    *=*+1          ;CASSETTE READ ERROR
00A9              RINONE                ;RS-232 RCVR FLAG FOR START BIT CHECK
00A9              REZ    *=*+1          ;CASSETTE READING ZEROES
00AA              RIDATA                ;RS-232 RCVR BYTE BUFFER
00AA              RDFLG  *=*+1          ;CASSETTE READ MODE
```

```
LOC   CODE        LINE

00AB              RIPRTY                ;RS-232 RCVR PARITY STORAGE
00AB              SHCNH   *=*+1         ;CASSETTE SHORT CNT
00AC              SAL     *=*+1
00AD              SAH     *=*+1
00AE              EAL     *=*+1
00AF              EAH     *=*+1
00B0              CMPO    *=*+1
00B1              TEMP    *=*+1
00B2              TAPE1   *=*+2         ;ADDRESS OF TAPE BUFFER
00B4              BITTS                 ;RS-232 TRNS BIT COUNT
00B4              SNSW1   *=*+1
00B5              NXTBIT                ;RS-232 TRNS NEXT BIT TO BE SENT
00B5              DIFF    *=*+1
00B6              RODATA                ;RS-232 TRNS BYTE BUFFER
00B6              FRMP    *=*+1
00B7              FNLEN   *=*+1         ;LENGHT CURRENT FILE N STR
00B8              LA      *=*+1         ;CURRENT FILE LOGICAL ADDR
00B9              SA      *=*+1         ;CURRENT FILE 2ND ADDR
00BA              FA      *=*+1         ;CURRENT FILE PRIMARY ADDR
00BB              FNADR   *=*+2         ;ADDR CURRENT FILE NAME STR
00BD              ROPRTY                ;RS-232 TRNS PARITY BUFFER
00BD              OCHAR   *=*+1
00BE              FSBLK   *=*+1         ;CASSETTE READ BLOCK COUNT
00BF              MYCH    *=*+1
00C0              CAS1    *=*+1         ;CASSETTE MANUAL/CONTROLLED SWITCH
00C1              TMPO
00C1              STAL    *=*+1
00C2              STAH    *=*+1
00C3              MEMUSS                ;CASSETTE LOAD TEMPS (2 BYTES)
00C3              TMP2    *=*+2
00C5              ;
00C5              ;VARIABLES FOR SCREEN EDITOR
00C5              ;
00C5              LSTX    *=*+1         ;KEY SCAN INDEX
00C6              NDX     *=*+1         ;INDEX TO KEYBOARD Q
00C7              RVS     *=*+1         ;RVS FIELD ON FLAG
00C8              INDX    *=*+1
00C9              LSXP    *=*+1         ;X POS AT START
00CA              LSTP    *=*+1
00CB              SFDX    *=*+1         ;SHIFT MODE ON PRINT
00CC              BLNSW   *=*+1         ;CURSOR BLINK ENAB
00CD              BLNCT   *=*+1         ;COUNT TO TOGGLE CUR
00CE              GDBLN   *=*+1         ;CHAR BEFORE CURSOR
00CF              BLNON   *=*+1         ;ON/OFF BLINK FLAG
00D0              CRSW    *=*+1         ;INPUT VS GET FLAG
00D1              PNT     *=*+2         ;POINTER TO ROW
00D3              PNTR    *=*+1         ;POINTER TO COLUMN
00D4              QTSW    *=*+1         ;QUOTE SWITCH
00D5              LNMX    *=*+1         ;40/80 MAX POSITION
00D6              TBLX    *=*+1
00D7              DATA    *=*+1
00D8              INSRT   *=*+1         ;INSERT MODE FLAG
00D9              LDTB1   *=*+25        ;40/80 LINE FLAGS
00F2              LINTMP  *=*+1         ;TEMPORARY FOR LINE INDEX
00F3              USER    *=*+2         ;SCREEN EDITOR COLOUR IP
00F5              KEYTAB  *=*+2         ;KEYSCAN TABLE INDIRECT
00F7              ;
00F7              ;RS-232 Z-PAGE
00F7              ;
00F7              RIBUF   *=*+2         ;RS-232 INPUT BUFFER POINTER
00F9              ROBUF   *=*+2         ;RS-232 OUTPUT BUFFER POINTER
```

```
LOC   CODE        LINE

00FB              FREKZP *=*+4          ;FREE KERNAL ZERO PAGE
00FF              BASZPT *=*+1          ;LOCATION ($00FF) USED BY BASIC
0100              ;
0100                     *=$100
0100              BAD    *=*+1
0101                     *=$200
0200              BUF    *=*+89         ;BASIC/MONITOR BUFFER
0259              ;
0259              ;TABLES FOR OPEN FILES
0259              ;
0259              LAT    *=*+10         ;LOGICAL FILE NUMBERS
0263              FAT    *=*+10         ;PRIMARY DEVICE NUMBERS
026D              SAT    *=*+10         ;SECONDARY ADDRESSES
0277              ;
0277              ;SYSTEM STORAGE
0277              ;
0277              KEYD   *=*+10         ;IRQ KEYBOARD BUFFER
0281              MEMSTR *=*+2          ;START OF MEMORY
0283              MEMSIZ *=*+2          ;TOP OF MEMORY
0285              TIMOUT *=*+1          ;IEEE TIMEOUT FLAG
0286              ;
0286              ;SCREEN EDITOR STORAGE
0286              ;
0286              COLOR  *=*+1          ;ACTIVE COLOUR NYBBLE
0287              GDCOL  *=*+1          ;ORIGINAL COLOUR BEFORE CURSOR
0288              HIBASE *=*+1          ;BASE LOCATION OF SCREEN (TOP)
0289              XMAX   *=*+1
028A              RPTFLG *=*+1          ;KEY REPEAT FLAG
028B              KOUNT  *=*+1
028C              DELAY  *=*+1
028D              SHFLAG *=*+1          ;SHIFT FLAG BYTE
028E              LSTSHF *=*+1          ;LAST SHIFT PATTERN
028F              KEYLOG *=*+2          ;INDIRECT FOR KEYBOARD TABLE SETUP
0291              MODE   *=*+1          ;0-PET MODE, 1-CATTACANNA
0292              AUTODN *=*+1          ;AUTO SCROLL DOWN FLAG (=0 ON,<>0 OFF)
0293              ;
0293              ;RS-232 STORAGE
0293              ;
0293              M26CTR *=*+1          ;6526 CONTROL REGISTER
0294              M26CDR *=*+1          ;6526 COMMAND REGISTER
0295              M26AJB *=*+2          ;NON STANDARD (BITTIME/2-100)
0297              RSSTAT *=*+1          ;RS-232 STATUS REGISTER
0298              BITNUM *=*+1          ;NUMBER OF BITS TO SEND (FAST RESPONSE)
0299              BAUDOF *=*+2          ;BAUD RATE FULL BIT TIME
029B              ;
029B              ;RECIEVER STORAGE
029B              ;
029B              RIDBE  *=*+1          ;INPUT BUFFER INDEX TO END
029C              RIDBS  *=*+1          ;INPUT BUFFER POINTER TO START
029D              ;
029D              ;TRANSMITTER STORAGE
029D              ;
029D              RODBS  *=*+1          ;OUTPUT BUFFER INDEX TO START
029E              RODBE  *=*+1          ;OUTPUT BUFFER INDEX TO END
029F              ;
029F              IRQTMP *=*+2          ;HOLDS IRQ DURING TAPE OPS
02A1              ;
02A1              ;
02A1                     *=$0300       ;PROGRAM INDIRECTS(10)
0300                     *=$0300+20    ;KERNAL/OS INDIRECTS (20)
0314              CINV   *=*+2          ;IRQ RAM VECTOR
```

```
LOC   CODE        LINE

0316              CBINV  *=*+2           ;BRK INSTR RAM VECTOR
0318              NMINV  *=*+2           ;NMI RAM VECTOR
031A              IOPEN  *=*+2           ;INDIRECTS FOR CODE
031C              ICLOSE *=*+2           ;CONFORMS TO KERNAL SPEC 8/19/80
031E              ICHKIN *=*+2
0320              ICKOUT *=*+2
0322              ICLRCH *=*+2
0324              IBASIN *=*+2
0326              IBSOUT *=*+2
0328              ISTOP  *=*+2
032A              IGETIN *=*+2
032C              ICLALL *=*+2
032E              USRCMD *=*+2
0330              ILOAD  *=*+2
0332              ISAVE  *=*+2           ;SAVESP
0334              ;
0334                     *=$0300+60
033C              TBUFFR *=*+192         ;CASSETTE DATA BUFFER
03FC              ;
03FC                     *=$0400
0400              CBMSCN *=*+999         ;64 SCREEN
07E7              ;
07E7                     *=$0800
0800              ;RAMLOC
0800              ;
0800                     *=$D000
D000              VICREG *=*+47          ;VIC REGISTERS
D02F              ;
D02F                     *=$D400
D400              SIDREG *=*+29          ;SID REGISTERS
D41D              ;
D41D                     *=$D800
D800              CBMCOL *=*+999         ;64 COLOUR NYBBLES
DBE7              ;
DBE7              ;I/O DEVICES
DBE7              ;
DBE7                     * =$DC00       ;6526 (IRQ)
DC00              COLM                   ;KEYBOARD MATRIX
DC00              D1DPA  *=*+1
DC01              ROWS                   ;KEYBOARD MATRIX
DC01              D1DPB  *=*+1
DC02              D1DDRA *=*+1
DC03              D1DDRB *=*+1
DC04              D1TAL  *=*+1
DC05              D1TAH  *=*+1
DC06              D1TBL  *=*+1
DC07              D1TBH  *=*+1
DC08              D1TOD1 *=*+1
DC09              D1TOD2 *=*+1
DC0A              D1TOD3 *=*+1
DC0B              D1TOD4 *=*+1
DC0C              D1IODB *=*+1
DC0D              D1ICR  *=*+1
DC0E              D1CRA  *=*+1
DC0F              D1CRB  *=*+1
DC10              ;
DC10                     *= $DD00       ;6526 (NMI)
DD00              D2DPA  *=*+1
DD01              D2DPB  *=*+1
DD02              D2DDRA *=*+1
DD03              D2DDRB *=*+1
```

```
LOC   CODE        LINE

DD04              D2TAL   *=*+1
DD05              D2TAH   *=*+1
DD06              D2TBL   *=*+1
DD07              D2TBH   *=*+1
DD08              D2TOD1  *=*+1
DD09              D2TOD2  *=*+1
DD0A              D2TOD3  *=*+1
DD0B              D2TOD4  *=*+1
DD0C              D2IODB  *=*+1
DD0D              D2ICR   *=*+1
DD0E              D2CRA   *=*+1
DD0F              D2CRB   *=*+1
DD10              ;
DD10              ;TAPE BLOCK TYPES
DD10              ;
DD10              EOT     =5              ;END OF TAPE
DD10              BLF     =1              ;BASIC LOAD FILE
DD10              BDF     =2              ;BASIC DATA FILE
DD10              PLF     =3              ;FIXED PROGRAM TYPE
DD10              BDFH    =4              ;BASIC DATA FILE HEADER
DD10              BUFSZ   =192            ;BUFFER SIZE
DD10              ;
DD10              ;TAPE ERROR TYPES
DD10              ;
DD10              SPERR   =16
DD10              CKERR   =32
DD10              SBERR   =4
DD10              LBERR   =8
DD10              ;
DD10              ;SCREEN EDITOR CONSTANTS
DD10              ;
DD10              LLEN    =40             ;SINGLE LINE 40 COLUMNS
DD10              LLEN2   =80             ;DOUBLE LINE 80 COLUMNS
DD10              NLINES  =25             ;25 ROWS ON SCREEN
DD10              BLUE    =6              ;BLUE SCREEN COLOUR
DD10              LTBLUE  =14             ;LT BLUE CHAR COLOUR
DD10              CR      =$D             ;CARRIAGE RETURN
DD10              MAXCHR  =80
DD10              NWRAP   =2
DD10                      .END
```

Basic ROM

This section contains the documented machine code routines for the Basic interpreter. The following listing is the output from the Commodore Assembler when assembling our interpretation of a source code for the Basic interpreter of the Commodore 64.

```
LOC   CODE        LINE

0328  A000              ;*********************************************
0329  A000              ;*
0330  A000              ;* BBBB   AAA   SSS  IIIII  CCC
0331  A000              ;* BB  B AA  A SS  S  III  CC  C
0332  A000              ;* BB  B AA  A SS     III  CC
0333  A000              ;* BBBB  AAAAA  SSS   III  CC
0334  A000              ;* BB  B AA  A     S  III  CC
0335  A000              ;* BB  B AA  A SS  S  III  CC  C
0336  A000              ;* BBBB  AA  A  SSS  IIIII  CCC
0337  A000              ;*
0338  A000              ;*************************************************
0339  A000  94 E3             .WOR $E394      ;START VECTOR
0340  A002  7B E3             .WOR $E37B      ;NMI VECTOR
0341  A004  43 42             .BYT 'CBMBASIC'
0342  A00C              ;*********************************************
0343  A00C              ;*START ADDRESS OF BASIC COMMANDS MINUS 1
0344  A00C              ;*********************************************
0345  A00C  30                .BYT $30,$A8,$41,$A7,$1D,$AD,$F7,$A8,$A4,$AB
0345  A00D  A8
0345  A00E  41
0345  A00F  A7
0345  A010  1D
0345  A011  AD
0345  A012  F7
0345  A013  A8
0345  A014  A4
0345  A015  AB
0346  A016  BE                .BYT $BE,$AB,$80,$B0,$05,$AC,$A4,$A9,$9F,$A8
0346  A017  AB
0346  A018  80
0346  A019  B0
0346  A01A  05
0346  A01B  AC
0346  A01C  A4
0346  A01D  A9
```

```
LOC   CODE        LINE

A01E  9F
A01F  A8
A020  70                .BYT $70,$A8,$27,$A9,$1C,$A8,$B2,$A8,$D1,$A8
A021  A8
A022  27
A023  A9
A024  1C
A025  A8
A026  B2
A027  A8
A028  D1
A029  A8
A02A  3A                .BYT $3A,$A9,$2E,$A8,$4A,$A9,$2C,$B8,$67,$E1
A02B  A9
A02C  2E
A02D  A8
A02E  4A
A02F  A9
A030  2C
A031  B8
A032  67
A033  E1
A034  55                .BYT $55,$E1,$64,$E1,$B2,$B3,$23,$B8,$7F,$AA
A035  E1
A036  64
A037  E1
A038  B2
A039  B3
A03A  23
A03B  B8
A03C  7F
A03D  AA
A03E  9F                .BYT $9F,$AA,$56,$A8,$9B,$A6,$5D,$A6,$85,$AA
A03F  AA
A040  56
A041  A8
A042  9B
A043  A6
A044  5D
A045  A6
A046  85
A047  AA
A048  29                .BYT $29,$E1,$BD,$E1,$C6,$E1,$7A,$AB,$41,$A6
A049  E1
A04A  BD
A04B  E1
A04C  C6
A04D  E1
A04E  7A
A04F  AB
A050  41
A051  A6
A052                    ;
A052                    ;***********************************************
A052                    ;*START ADDRESSES OF BASIC FUNCTIONS
A052                    ;***********************************************
A052                    ;
A052  39                .BYT $39,$BC,$CC,$BC,$58,$BC,$10,$03,$7D,$B3
A053  BC
A054  CC
A055  BC
```

```
LOC   CODE        LINE

A056  58
A057  BC
A058  10
A059  03
A05A  7D
A05B  B3
A05C  9E                .BYT $9E,$B3,$71,$BF,$97,$E0,$EA,$B9,$ED,$BF
A05D  B3
A05E  71
A05F  BF
A060  97
A061  E0
A062  EA
A063  B9
A064  ED
A065  BF
A066  64                .BYT $64,$E2,$6B,$E2,$B4,$E2,$0E,$E3,$0D,$B8
A067  E2
A068  6B
A069  E2
A06A  B4
A06B  E2
A06C  0E
A06D  E3
A06E  0D
A06F  B8
A070  7C                .BYT $7C,$B7,$65,$B4,$AD,$B7,$8B,$B7,$EC,$B6
A071  B7
A072  65
A073  B4
A074  AD
A075  B7
A076  8B
A077  B7
A078  EC
A079  B6
A07A  00                .BYT $00,$B7,$2C,$B7,$37,$B7
A07B  B7
A07C  2C
A07D  B7
A07E  37
A07F  B7
A080                ;
A080                ;***********************************************
A080                ;*PRIORITY CODES AND ADDRESSES FOR BASIC OPERATORS
A080                ;***********************************************
A080                ;
A080  79                .BYT $79,$69,$B8,$79,$52,$B8,$7B,$2A,$BA,$7B
A081  69
A082  B8
A083  79
A084  52
A085  B8
A086  7B
A087  2A
A088  BA
A089  7B
A08A  11                .BYT $11,$BB,$7F,$7A,$BF,$50,$E8,$AF,$46,$E5
A08B  BB
A08C  7F
A08D  7A
```

```
LOC   CODE        LINE

A08E  BF
A08F  50
A090  E8
A091  AF
A092  46
A093  E5
A094  AF                .BYT $AF,$7D,$B3,$BF,$5A,$D3,$AE,$64,$15,$B0
A095  7D
A096  B3
A097  BF
A098  5A
A099  D3
A09A  AE
A09B  64
A09C  15
A09D  B0
A09E                    ;
A09E                    ;************************************************
A09E                    ;*BASIC COMMAND WORDS
A09E                    ;************************************************
A09E                    ;
A09E  45 4E             .BYT 'EN',$C4,'FO',$D2,'NEX',$D4,'DAT',$C1
A0A0  C4
A0A1  46 4F
A0A3  D2
A0A4  4E 45 58
A0A7  D4
A0A8  44 41 54
A0AB  C1
A0AC  49 4E             .BYT 'INPUT',$A3,'INPU',$D4,'DI',$CD,'REA',$C4
A0B1  A3
A0B2  49 4E
A0B6  D4
A0B7  44 49
A0B9  CD
A0BA  52 45 41
A0BD  C4
A0BE  4C 45             .BYT 'LE',$D4,'GOT',$CF,'RU',$CE,'I',$C6
A0C0  D4
A0C1  47 4F 54
A0C4  CF
A0C5  52 55
A0C7  CE
A0C8  49
A0C9  C6
A0CA  52 45             .BYT 'RESTOR',$C5,'GOSU',$C2,'RETUR',$CE
A0D0  C5
A0D1  47 4F
A0D5  C2
A0D6  52 45
A0DB  CE
A0DC  52 45             .BYT 'RE',$CD,'STO',$D0,'O',$CE,'WAI',$D4
A0DE  CD
A0DF  53 54 4F
A0E2  D0
A0E3  4F
A0E4  CE
A0E5  57 41 49
A0E8  D4
A0E9  4C 4F 41          .BYT 'LOA',$C4,'SAV',$C5,'VERIF',$D9,'DE',$C6
A0EC  C4
```

```
LOC   CODE        LINE

A0ED  53 41 56
A0F0  C5
A0F1  56 45
A0F6  D9
A0F7  44 45
A0F9  C6
A0FA  50 4F 4B          .BYT 'POK',$C5,'PRINT',$A3,'PRIN',$D4,'CON',$D4
A0FD  C5
A0FE  50 52
A103  A3
A104  50 52
A108  D4
A109  43 4F 4E
A10C  D4
A10D  4C 49 53          .BYT 'LIS',$D4,'CL',$D2,'CM',$C4,'SY',$D3
A110  D4
A111  43 4C
A113  D2
A114  43 4D
A116  C4
A117  53 59
A119  D3
A11A  4F 50 45          .BYT 'OPE',$CE,'CLOS',$C5,'GE',$D4,'NE',$D7
A11D  CE
A11E  43 4C
A122  C5
A123  47 45
A125  D4
A126  4E 45
A128  D7
A129  54 41 42          .BYT 'TAB',$A8,'T',$CF,'F',$CE,'SPC',$A8
A12C  A8
A12D  54
A12E  CF
A12F  46
A130  CE
A131  53 50 43
A134  A8
A135  54 48 45          .BYT 'THE',$CE,'NO',$D4,'STE',$D0,$AB,$AD,$AA,$AF
A138  CE
A139  4E 4F
A13B  D4
A13C  53 54 45
A13F  D0
A140  AB
A141  AD
A142  AA
A143  AF
A144  DE                .BYT $DE,'AN',$C4,'O',$D2,$BE,$BD,$BC,'SG',$CE
A145  41 4E
A147  C4
A148  4F
A149  D2
A14A  BE
A14B  BD
A14C  BC
A14D  53 47
A14F  CE
A150  49 4E             .BYT 'IN',$D4,'AB',$D3,'US',$D2,'FR',$C5,'PO',$D3
A152  D4
A153  41 42
```

```
LOC   CODE        LINE

A155  D3
A156  55 53
A158  D2
A159  46 52
A15B  C5
A15C  50 4F
A15E  D3
A15F  53 51             .BYT 'SQ',$D2,'RN',$C4,'LO',$C7,'EX',$D0,'CO',$D3
A161  D2
A162  52 4E
A164  C4
A165  4C 4F
A167  C7
A168  45 58
A16A  D0
A16B  43 4F
A16D  D3
A16E  53 49             .BYT 'SI',$CE,'TA',$CE,'AT',$CE,'PEE',$CB,'LE',$CE
A170  CE
A171  54 41
A173  CE
A174  41 54
A176  CE
A177  50 45 45
A17A  CB
A17B  4C 45
A17D  CE
A17E  53 54 52          .BYT 'STR',$A4,'VA',$CC,'AS',$C3,'CHR',$A4
A181  A4
A182  56 41
A184  CC
A185  41 53
A187  C3
A188  43 48 52
A18B  A4
A18C  4C 45             .BYT 'LEFT',$A4,'RIGHT',$A4,'MID',$A4,'G',$CF,$00
A190  A4
A191  52 49
A196  A4
A197  4D 49 44
A19A  A4
A19B  47
A19C  CF
A19D  00
A19E                    ;
A19E                    ;************************************************
A19E                    ;*ERROR MESSAGES
A19E                    ;************************************************
A19E                    ;
A19E  54 4F             .BYT 'TOO MANY FILE',$D3,'FILE OPE',$CE
A1AB  D3
A1AC  46 49
A1B4  CE
A1B5  46 49             .BYT 'FILE NOT OPE',$CE,'FILE NOT FOUN',$C4
A1C1  CE
A1C2  46 49
A1CF  C4
A1D0  44 45             .BYT 'DEVICE NOT PRESEN',$D4,'NOT INPUT FIL',$C5
A1E1  D4
A1E2  4E 4F
A1EF  C5
```

```
LOC   CODE        LINE

A1F0  4E 4F             .BYT 'NOT OUTPUT FIL',$C5,'MISSING FILE NAM',$C5
A1FE  C5
A1FF  4D 49
A20F  C5
A210  49 4C             .BYT 'ILLEGAL DEVICE NUMBE',$D2
A224  D2
A225  4E 45             .BYT 'NEXT WITHOUT FO',$D2,'SYNTA',$D8
A234  D2
A235  53 59
A23A  D8
A23B  52 45             .BYT 'RETURN WITHOUT GOSU',$C2,'OUT OF DAT',$C1
A24E  C2
A24F  4F 55
A259  C1
A25A  49 4C             .BYT 'ILLEGAL QUANTIT',$D9,'OVERFLO',$D7
A269  D9
A26A  4F 56
A271  D7
A272  4F 55             .BYT 'OUT OF MEMOR',$D9
A27E  D9
A27F  55 4E             .BYT 'UNDEF',$27,'D STATEMEN',$D4
A284  27
A285  44 20
A28F  D4
A290  42 41             .BYT 'BAD SUBSCRIP',$D4,'REDIM',$27,'D ARRA',$D9
A29C  D4
A29D  52 45
A2A2  27
A2A3  44 20
A2A9  D9
A2AA  44 49             .BYT 'DIVISION BY ZER',$CF,'ILLEGAL DIREC',$D4
A2B9  CF
A2BA  49 4C
A2C7  D4
A2C8  54 59             .BYT 'TYPE MISMATC',$C8,'STRING TOO LON',$C7
A2D4  C8
A2D5  53 54
A2E3  C7
A2E4  46 49             .BYT 'FILE DAT',$C1,'FORMULA TOO COMPLE',$D8
A2EC  C1
A2ED  46 4F
A2FF  D8
A300  43 41 4E          .BYT 'CAN',$27,'T CONTINU',$C5
A303  27
A304  54 20
A30D  C5
A30E  55 4E             .BYT 'UNDEF',$27,'D FUNCTIO',$CE,'VERIF',$D9
A313  27
A314  44 20
A31D  CE
A31E  56 45
A323  D9
A324  4C 4F 41          .BYT 'LOA',$C4
A327  C4
A328              ;
A328              ;***********************************************
A328              ;*ERROR MESSAGE VECTORS
A328              ;***********************************************
A328              ;
A328  9E                .BYT $9E,$A1,$AC,$A1,$B5,$A1,$C2,$A1,$D0,$A1
A329  A1
```

```
LOC   CODE        LINE

A32A  AC
A32B  A1
A32C  B5
A32D  A1
A32E  C2
A32F  A1
A330  D0
A331  A1
A332  E2                .BYT $E2,$A1,$F0,$A1,$FF,$A1,$10,$A2,$25,$A2
A333  A1
A334  F0
A335  A1
A336  FF
A337  A1
A338  10
A339  A2
A33A  25
A33B  A2
A33C  35                .BYT $35,$A2,$3B,$A2,$4F,$A2,$5A,$A2,$6A,$A2
A33D  A2
A33E  3B
A33F  A2
A340  4F
A341  A2
A342  5A
A343  A2
A344  6A
A345  A2
A346  72                .BYT $72,$A2,$7F,$A2,$90,$A2,$9D,$A2,$AA,$A2
A347  A2
A348  7F
A349  A2
A34A  90
A34B  A2
A34C  9D
A34D  A2
A34E  AA
A34F  A2
A350  BA                .BYT $BA,$A2,$C8,$A2,$D5,$A2,$E4,$A2,$ED,$A2
A351  A2
A352  C8
A353  A2
A354  D5
A355  A2
A356  E4
A357  A2
A358  ED
A359  A2
A35A  00                .BYT $00,$A3,$0E,$A3,$1E,$A3,$24,$A3,$B3,$A3
A35B  A3
A35C  0E
A35D  A3
A35E  1E
A35F  A3
A360  24
A361  A3
A362  B3
A363  A3
A364              ;
A364              ;***********************************************
A364              ;*MISCELLANEOUS MESSAGES
```

```
LOC   CODE        LINE

A364              ;************************************************
A364              ;
A364  0D                  .BYT $0D,'OK',$0D,$00,'  ERROR',$00,' IN ',$00
A365  4F 4B
A367  0D
A368  00
A369  20 20
A370  00
A371  20 49
A375  00
A376  0D                  .BYT $0D,$0A,'READY.',$0D,$0A,$00
A377  0A
A378  52 45
A37E  0D
A37F  0A
A380  00
A381  0D                  .BYT $0D,$0A,'BREAK',$00,$A0
A382  0A
A383  42 52
A388  00
A389  A0
A38A                      .END
A38A                      .LIB B1
A38A              ;************************************************
A38A              ;*PEEK STACK FOR 'FOR' OR 'GOSUB' ACTIVITY.
A38A              ;*THIS ROUTINE IS CALLED BY NEXT AND
A38A              ;*RETURN, IF AN ACTIVE FOR IS NOT FOUND
A38A              ;*THEN THIS ROUTINE SETS THE Z FLAG = 0
A38A              ;*AND JUMPS TO THE ERROR ROUTINE. IF FOR
A38A              ;*IS FOUND THEN THE LOOP VARIABLE IS
A38A              ;*CHECKED. IF NOT 'FOR' THEN THE NEXT STACK
A38A              ;*VALUE IS RETURNED IN .A.
A38A              ;************************************************
A38A  BA          L0        TSX
A38B  E8                    INX
A38C  E8                    INX
A38D  E8                    INX
A38E  E8                    INX
A38F  BD 01 01    L90       LDA BAD+1,X
A392  C9 81                 CMP #$81        ;FOR CODE
A394  D0 21                 BNE L3
A396  A5 4A                 LDA FORPNT+1    ;VAR POINTER+1
A398  D0 0A                 BNE L4
A39A  BD 02 01              LDA BAD+2,X     ;COMPARE VARIABLE NAME
A39D  85 49                 STA FORPNT      ;VAR POINTER
A39F  BD 03 01              LDA BAD+3,X
A3A2  85 4A                 STA FORPNT+1    ;VAR POINTER +1
A3A4  DD 03 01    L4        CMP BAD+3,X
A3A7  D0 07                 BNE L2
A3A9  A5 49                 LDA FORPNT      ;VAR POINTER
A3AB  DD 02 01              CMP BAD+2,X
A3AE  F0 07                 BEQ L3
A3B0  8A          L2        TXA
A3B1  18                    CLC
A3B2  69 12                 ADC #$12        ;INCREASE STACK POINTER BY 18
A3B4  AA                    TAX
A3B5  D0 D8                 BNE L90         ;CHECK NEXT LOOP
A3B7  60          L3        RTS
A3B8              ;************************************************
A3B8              ;*OPEN UP SPACE IN MEMORY.
A3B8              ;*ALLOWS THE MERGING OF A NEW BASIC LINE INTO A
```

```
LOC   CODE         LINE

A3B8               ;*PROGRAM BY MOVING THE ENTIRE PROGRAM STORED
A3B8               ;*ABOVE THAT LINE UP ONE BYTE IN MEMORY FOR
A3B8               ;*EVERY CHARACTER ENTERED, A SET OF SEVEN
A3B8               ;*VARIABLES ARE USED TO DO THIS, THEY ARE:
A3B8               ;*$5F-$60  - BOTTOM OF MEMORY TO BE MOVED
A3B8               ;*$5A-$5B  - START OF MEMORY TO BE MOVED
A3B8               ;*$58-$59  - START OF MEMORY TO BE MOVED TO
A3B8               ;*$22 - TEMPORARY PARAMETER
A3B8               ;*********************************************
A3B8  20 08 A4     L1      JSR L92          ;CHECK FOR FREE MEMORY
A3BB  85 31                STA STREND      ;END ARRAYS
A3BD  84 32                STY STREND+1    ;END ARRAYS +1
A3BF  38           L39     SEC
A3C0  A5 5A                LDA BASTMP+15
A3C2  E5 5F                SBC FACEXP-2
A3C4  85 22                STA INDEX
A3C6  A8                   TAY
A3C7  A5 5B                LDA BASTMP+16
A3C9  E5 60                SBC FACEXP-1
A3CB  AA                   TAX
A3CC  E8                   INX
A3CD  98                   TYA
A3CE  F0 23                BEQ L8
A3D0  A5 5A                LDA BASTMP+15
A3D2  38                   SEC
A3D3  E5 22                SBC INDEX
A3D5  85 5A                STA BASTMP+15
A3D7  B0 03                BCS L428
A3D9  C6 5B                DEC BASTMP+16
A3DB  38                   SEC
A3DC  A5 58        L428    LDA BASTMP+13
A3DE  E5 22                SBC INDEX
A3E0  85 58                STA BASTMP+13
A3E2  B0 08                BCS L9
A3E4  C6 59                DEC BASTMP+14
A3E6  90 04                BCC L9
A3E8  B1 5A        L7      LDA (BASTMP+15),Y
A3EA  91 58                STA (BASTMP+13),Y
A3EC  88           L9      DEY
A3ED  D0 F9                BNE L7
A3EF  B1 5A                LDA (BASTMP+15),Y
A3F1  91 58                STA (BASTMP+13),Y
A3F3  C6 5B        L8      DEC BASTMP+16
A3F5  C6 59                DEC BASTMP+14
A3F7  CA                   DEX
A3F8  D0 F2                BNE L9
A3FA  60                   RTS
A3FB               ;*********************************************
A3FB               ;*TEST: STACK TOO DEEP?
A3FB               ;*TEST WHETHER A SPECIFIED NUMBER OF BYTES CAN
A3FB               ;*BE PLACED ON THE STACK. THE NUMBER OF BYTES
A3FB               ;*IS STORED IN THE ACCUMULATOR, IT SHOULD BE
A3FB               ;*NOTED THAT THIS VALUE MUST BE HALF THE NUMBER
A3FB               ;*OF BYTES TO BE PLACED ON THE STACK. IF THERE
A3FB               ;*IS INSUFFICIENT SPACE THEN AN 'OUT OF MEMORY'
A3FB               ;*ERROR IS GENERATED.
A3FB               ;*********************************************
A3FB  0A           L6      ASL A
A3FC  69 3E                ADC #$3E
A3FE  B0 35                BCS L11          ;DISPLAY'OUT OF MEMORY'
A400  85 22                STA INDEX
```

```
LOC   CODE        LINE

A402  BA                TSX
A403  E4 22             CPX INDEX
A405  90 2E             BCC L11         ;DISPLAY'OUT OF MEMORY'
A407  60                RTS
A408                ;***********************************************
A408                ;*CHECK AVAILABLE MEMORY.
A408                ;*THIS CHECKS TO SEE IF THERE IS SUFFICIENT
A408                ;*SPACE FOR THE STORAGE OF STRINGS AND VARIABLES
A408                ;*THIS IS DONE BY COMPARING THE STRING POINTER
A408                ;*VARIABLES AGAINST THE TOP OF VARIABLES ADDRESS
A408                ;*STORED IN A (HIGH) AND Y(LOW). IF THERE IS
A408                ;*INSUFFICIENT SPACE THEN THE GARBAGE COLLECT
A408                ;*ROUTINE IS CALLED TO REMOVE UNWANTED STRINGS
A408                ;*AND THEREBY FREE MEMORY SPACE, IF AFTER THIS
A408                ;*THERE IS STILL INSUFFICIENT FREE MEMORY THEN
A408                ;*AN 'OUT OF MEMORY' ERROR IS GENERATED.
A408                ;***********************************************
A408  C4 34        L92     CPY FRETOP+1     ;STRING LOW +1   SAME?
A40A  90 08             BCC L15          ;SPACE AVAILABLE
A40C  D0 04             BNE L5           ;GOT TO OPEN
A40E  C5 33             CMP FRETOP       ;STRING LOW SAME?
A410  90 22             BCC L15          ;SPACE AVAILABLE
A412  48            L5      PHA
A413  A2 09             LDX #$09
A415  98                TYA
A416  48            L12     PHA
A417  B5 57             LDA BASTMP+12,X ;SAVE REGISTERS
A419  CA                DEX
A41A  10 FA             BPL L12
A41C  20 26 B5          JSR L406        ;GARBAGE COLLECT
A41F  A2 F7             LDX #$F7
A421  68            L13     PLA
A422  95 61             STA FACEXP,X    ;RESTORE REGISTERS
A424  E8                INX
A425  30 FA             BMI L13
A427  68                PLA
A428  A8                TAY
A429  68                PLA
A42A  C4 34             CPY FRETOP+1
A42C  90 06             BCC L15         ;SPACE AVAILABLE
A42E  D0 05             BNE L11         ;DISPLAY'OUT OF MEMORY'
A430  C5 33             CMP FRETOP
A432  B0 01             BCS L11         ;DISPLAY'OUT OF MEMORY'
A434  60            L15     RTS
A435                ;***********************************************
A435                ;*ERROR NUMBER FOR OUT OF MEMORY
A435                ;***********************************************
A435  A2 10         L11     LDX #$10
A437                ;***********************************************
A437                ;*ERROR MESSAGE HANDLING ROUTINE.
A437                ;*THIS ROUTINE OUTPUTS AN ERROR MESSAGE FROM THE
A437                ;*TABLE OF ERROR MESSAGES, THE MESSAGE NUMBER IS
A437                ;*STORED IN THE X INDEX REGISTER. THE ERROR
A437                ;*MESSAGE IS OUTPUT TO THE CURRENTLY OPEN OUTPUT
A437                ;*DEVICE(DEFAULT SCREEN). THIS IS A USEFUL WAY
A437                ;*OF GENERATING ERROR MESSAGES IN A USER PROGRAM
A437                ;*THOUGH UNLESS ONE REPLACES THE ERROR MESSAGE
A437                ;*TABLE ONE IS CONFINED TO THE STANDARD SET OF
A437                ;*MESSAGES.
A437                ;***********************************************
A437  6C 00 03     L10     JMP ($0300)     ;JUMP INDIRECT TO ERROR VECTOR
```

```
LOC   CODE        LINE

A43A  8A          L47     TXA
A43B  0A                  ASL A
A43C  AA                  TAX
A43D  BD 26 A3            LDA $A326,X     ;MESSAGE ADDRESS LO
A440  85 22               STA INDEX
A442  BD 27 A3            LDA $A327,X     ;MESSAGE ADDRESS HI
A445  85 23               STA INDEX+1
A447  20 CC FF            JSR L674        ;RESTORE ACTIVE I/O CHANNEL
A44A  A9 00               LDA #$00
A44C  85 13               STA INPRMT      ;RESTORE CURRENT I/O FLAG
A44E  20 D7 AA            JSR L48         ;OUTPUT (CR) AND (LF)
A451  20 45 AB            JSR L202        ;OUTPUT (?)
A454  A0 00               LDY #$00
A456  B1 22       L661    LDA (INDEX),Y   ;TEXT OF ERROR MESSAGE
A458  48                  PHA
A459  29 7F               AND #$7F
A45B  20 47 AB            JSR L18         ;OUTPUT ERROR MESSAGE
A45E  C8                  INY
A45F  68                  PLA
A460  10 F4               BPL L661
A462  20 7A A6            JSR L636        ;INITIALISE BASIC POINTERS, STOP CONT
A465  A9 69               LDA #$69        ;A/Y ERROR POINTERS
A467  A0 A3               LDY #$A3
A469  20 1E AB    L20     JSR L198         ;STRING OUTPUT
A46C  A4 3A               LDY CURLIN+1    ;PROGRAM MODE
A46E  C8                  INY
A46F  F0 03               BEQ L120        ;DIRECT MODE
A471  20 C2 BD            JSR L555        ;OUTPUT 'IN (LINE #)'
A474  A9 76       L120    LDA #$76         ;SET POINTER TO 'READY'
A476  A0 A3               LDY #$A3
A478  20 1E AB            JSR L198        ;STRING OUTPUT
A47B  A9 80               LDA #$80
A47D  20 90 FF            JSR L1216       ;SET DIRECT MODE FLAG
A480              ;*********************************************
A480              ;*WARM START; WAIT FOR BASIC COMMAND.
A480              ;*PRINTS READY ON THE OUTPUT DEVICE  AND THEN
A480              ;*WAITS FOR AN INPUT FROM THE KEYBOARD. THE
A480              ;*KEYBOARD INPUT ROUTINE STARTS AT $A483 AND
A480              ;*CALLS THE ROUTINE AT$A560 WHICH INPUTS A LINE
A480              ;*INTO THE 80 CHARACTER INPUT BUFFER AND ON THE
A480              ;*RETURN KEY BEING PRESSED PLACES A ZERO TERMINATOR
A480              ;*AS THE LAST CHARACTER. THE LENGTH OF THE INPUT
A480              ;*LINE IS STORED IN LOCATION $0B. THE CONTENTS OF
A480              ;*THE INPUT BUFFER ARE READ BY 'CHARGET' WHICH
A480              ;*TESTS IF THE FIRST CHARACTER IN THE BUFFER IS
A480              ;*NUMERIC IF SO THEN IT BRANCHES TO THE ROUTINE
A480              ;*AT $A49C. IF NOT THEN THE COMMAND IS IN THE
A480              ;*DIRECT MODE, IT IS TOKENISED AND RUN BY THE
A480              ;*ROUTINES AT $A579 AND $A7E1.
A480              ;*********************************************
A480  6C 02 03    L23     JMP ($0302)      ;WARM START JUMP VECTOR
A483  20 60 A5            JSR L41         ;GET BASIC LINE INTO INPUT BUFFER
A486  86 7A               STX TXTPTR      ;CHARGET BASIC POINTERS
A488  84 7B               STY TXTPTR+1
A48A  20 73 00            JSR CHRGET      ;JUMP TO 'CHARGET'
A48D  AA                  TAX
A48E  F0 F0               BEQ L23         ;BUFFER EMPTY DO AGAIN
A490  A2 FF               LDX #$FF
A492  86 3A               STX CURLIN+1    ;FLAG LINE # FOR DIRECT MODE
A494  90 06               BCC L27         ;HANDLE NEW BASIC INPUT
A496  20 79 A5            JSR L45         ;CRUNCH KEYWORDS TO TOKENS
```

```
LOC   CODE        LINE

A499  4C E1 A7          JMP L104        ;EXECUTE BASIC STATEMENT
A49C                ;*********************************************
A49C                ;*HANDLE NEW BASIC LINE INPUT.
A49C                ;*THIS FIRST CHECKS THE LINE NUMBER, IF IT EXISTS
A49C                ;*THEN IT REPLACES IT WITH THE NEW LINE, IF NOT
A49C                ;*THEN IT INSERTS THE LINE INTO THE BASIC
A49C                ;*PROGRAM. IF THE LINE EXISTS IT IS FIRST ERASED
A49C                ;*THEN THE NEW LINE INSERTED BY OPENING UP
A49C                ;*SPACE IN MEMORY USING ROUTINE $A3B8.
A49C                ;*********************************************
A49C  20 6B A9      L27     JSR L144        ;CHANGE LINE NUMBER TO ADDRESS POINTER
A49F  20 79 A5              JSR L45         ;CRUNCH KEYWORDS TO TOKENS
A4A2  84 0B                 STY COUNTB      ;POINT TO INPUT BUFFER
A4A4  20 13 A6              JSR L57         ;CALCULATE LINE ADDRESS
A4A7  90 44                 BCC L35
A4A9                ;*********************************************
A4A9                ;*ERASE PROGRAM LINE.
A4A9                ;*********************************************
A4A9  A0 01                 LDY #$01        ;SET POINTER
A4AB  B1 5F                 LDA (FACEXP-2),Y ;GET NEXT LINE LINK ADDRESS MSB
A4AD  85 23                 STA INDEX+1     ;SAVE TEMPORARY
A4AF  A5 2D                 LDA VARTAB      ;GET POINTER TO START OF VARIABLES LSB
A4B1  85 22                 STA INDEX       ;SAVE TEMPORARY
A4B3  A5 60                 LDA FACEXP-1    ;GET LINK ADDRESS POINTER MSB
A4B5  85 25                 STA INDEX+3     ;STORE TEMPORARY
A4B7  A5 5F                 LDA FACEXP-2    ;GET LINK ADDRESS POINTER LSB
A4B9  88                    DEY
A4BA  F1 5F                 SBC (FACEXP-2),Y ; GET NEXT LINE LINK ADDRESS LSB
A4BC                                        ;AND SUBTRACT FROM POINTER LSB
A4BC  18                    CLC
A4BD  65 2D                 ADC VARTAB      ;ADD TO START OF VARIABLES ADDRESS
A4BF  85 2D                 STA VARTAB      ;STORE IN START OF VARIABLES ADDRESS LSB
A4C1  85 24                 STA INDEX+2     ;STORE IN TEMPORARY
A4C3  A5 2E                 LDA VARTAB+1    ;GET START OF VARIABLES ADDRESS MSB
A4C5  69 FF                 ADC #$FF        ;IF CARRY CLEAR SUBTRACT 1
A4C7  85 2E                 STA VARTAB+1    ;STORE IN START OF VARS ADDRESS MSB
A4C9  E5 60                 SBC FACEXP-1    ;SUBTRACT FROM LINK ADDRESS MSB
A4CB  AA                    TAX             ;AND TRANSFER TO .X
A4CC  38                    SEC
A4CD  A5 5F                 LDA FACEXP-2    ;GET LINK ADDRESS LSB
A4CF  E5 2D                 SBC VARTAB      ;SUBTRACT FROM START OF VARS POINTER LSB
A4D1  A8                    TAY             ;TRANSFER TO .Y
A4D2  B0 03                 BCS L28
A4D4  E8                    INX
A4D5  C6 25                 DEC INDEX+3     ;DECREMENT LINK ADDRESS TEMPORARY
A4D7  18            L28     CLC
A4D8  65 22                 ADC INDEX       ;ADD POINTER TO START OF VARS LSB
A4DA  90 03                 BCC L34
A4DC  C6 23                 DEC INDEX+1     ;IF CARRY SET DECREMENT NEXT LINE LINK
A4DE  18                    CLC
A4DF  B1 22         L34     LDA (INDEX),Y
A4E1  91 24                 STA (INDEX+2),Y
A4E3  C8                    INY
A4E4  D0 F9                 BNE L34
A4E6  E6 23                 INC INDEX+1
A4E8  E6 25                 INC INDEX+3
A4EA  CA                    DEX
A4EB  D0 F2                 BNE L34
A4ED                ;*********************************************
A4ED                ;*INSERT PROGRAM LINE.
A4ED                ;*********************************************
```

```
LOC   CODE        LINE

A4ED  20 59 A6    L35    JSR L672        ;RESET CHARGET TO START OF PROG AND CLR
A4F0  20 33 A5           JSR L635       ;REBUILD CHAINING
A4F3  AD 00 02           LDA BUF
A4F6  F0 88              BEQ L23        ;IF 0 THEN WARM START
A4F8  18                 CLC
A4F9  A5 2D              LDA VARTAB     ;GET POINTER TO START OF VARS LSB
A4FB  85 5A              STA BASTMP+15  ;SAVE
A4FD  65 0B              ADC COUNTB     ;ADD TO INPUT BUFFER POINTER
A4FF  85 58              STA BASTMP+13  ;STORE
A501  A4 2E              LDY VARTAB+1   ;GET POINTER TO START OF VARS MSB
A503  84 5B              STY BASTMP+16  ;STORE
A505  90 01              BCC L33        ;IF NO CARRY BRANCH
A507  C8                 INY            ;INCREMENT MSB
A508  84 59       L33    STY BASTMP+14   ;AND SAVE
A50A  20 B8 A3           JSR L1         ;OPEN UP SPACE IN MEMORY
A50D  A5 14              LDA LINNUM     ;GET LINE NUMBER LSB
A50F  A4 15              LDY LINNUM+1   ;MSB
A511  8D FE 01           STA BUF-2      ;SAVE
A514  8C FF 01           STY BUF-1
A517  A5 31              LDA STREND     ;GET POINTER TO END OF ARRAYS LSB
A519  A4 32              LDY STREND+1   ;MSB
A51B  85 2D              STA VARTAB     ;SAVE IN POINTER TO START OF VARS LSB
A51D  84 2E              STY VARTAB+1   ;MSB
A51F  A4 0B              LDY COUNTB     ;GET INPUT BUFFER POINTER
A521  88                 DEY
A522  B9 FC 01    L38    LDA BUF-4,Y
A525  91 5F              STA (FACEXP-2),Y
A527  88                 DEY
A528  10 F8              BPL L38
A52A  20 59 A6    L40    JSR L672        ;RESET CHARGET TO START OF PROG AND CLR
A52D  20 33 A5           JSR L635       ;REBUILD CHAINING
A530  4C 80 A4           JMP L23        ;WARM START
A533              ;*********************************************
A533              ;*REBUILD CHAINING OF BASIC LINES.
A533              ;*AFTER INSERTING A NEW LINE OF BASIC THIS
A533              ;*ROUTINE RECALCULATES THE LINK ADDRESS BETWEEN
A533              ;*EACH LINE. IT DOES THIS BY SEARCHING FOR THE
A533              ;*ZERO BYTE END OF LINE TERMINATORS TO DETERMINE
A533              ;*LINE LENGTH AND NEW LINK ADDRESS.
A533              ;*********************************************
A533  A5 2B       L635   LDA TXTTAB      ;GET START OF PROG LSB
A535  A4 2C              LDY TXTTAB+1   ;MSB
A537  85 22              STA INDEX      ;SAVE
A539  84 23              STY INDEX+1
A53B  18                 CLC
A53C  A0 01       L37    LDY #$01        ;SET POINTER
A53E  B1 22              LDA (INDEX),Y
A540  F0 1D              BEQ L42        ;END OF PROGRAM
A542  A0 04              LDY #$04
A544  C8          L43    INY
A545  B1 22              LDA (INDEX),Y  ;FIND END OF LINE
A547  D0 FB              BNE L43        ;NOT YET
A549  C8                 INY
A54A  98                 TYA
A54B  65 22              ADC INDEX      ;CALCULATE ADDRESS
A54D  AA                 TAX            ;OF START OF NEXT LINE LSB
A54E  A0 00              LDY #$00
A550  91 22              STA (INDEX),Y  ;STORE IN NEXT LINE
A552  A5 23              LDA INDEX+1    ;GET POINTER LSB
A554  69 00              ADC #$00       ;CALCULATE ADDRESS OF START
A556  C8                 INY            ; OF NEXT LINE MSB
```

```
LOC   CODE        LINE

A557  91 22              STA (INDEX),Y  ;STORE IN NEXT LINE POINTER MSB
A559  86 22              STX INDEX      ;SET POINTER TO NEXT
A55B  85 23              STA INDEX+1    ;LINE
A55D  90 DD              BCC L37        ;AND DO
A55F  60           L42   RTS
A560               ;*********************************************
A560               ;*RECEIVE LINE FROM KEYBOARD.
A560               ;*CHARACTERS ARE INPUT FROM THE KEYBOARD VIA
A560               ;*THE KEYBOARD BUFFER AND PLACED IN CONSECUTIVE
A560               ;*LOCATIONS IN THE INPUT BUFFER WHICH IS 88
A560               ;*CHARACTERS LONG AND STARTS AT LOCATION $0200.
A560               ;*THE ROUTINE IS TERMINATED WHEN THE RETURN KEY
A560               ;*IS PRESSED, AND A TERMINATING ZERO PUT AS THE
A560               ;*LAST CHARACTER IN THE BUFFER. THE NUMBER OF
A560               ;*CHARACTERS STORED IN THE BUFFER IS FOUND IN
A560               ;*LOCATION $0B. IF AN ATTEMPT IS MADE TO PUT
A560               ;*MORE THAN 80 CHARACTERS INTO THE BUFFER THEN
A560               ;*AN ERROR IS GENERATED.
A560               ;*********************************************
A560  A2 00        L41   LDX #$00         ;SET BUFFER POINTER TO ZERO
A562  20 12 E1     L26   JSR L203         ;GET A CHARACTER
A565  C9 0D              CMP #$0D         ;IS IT RETURN?
A567  F0 0D              BEQ L46          ;YES THEN EXIT ROUTINE
A569  9D 00 02           STA BUF,X        ;STORE CHARACTER IN BUFFER
A56C  E8                 INX              ;NEXT CHARACTER
A56D  E0 59              CPX #BASTMP+14   ;MORE THAN 88 CHARACTERS?
A56F  90 F1              BCC L26          ;NO GET NEXT CHARACTER
A571  A2 17              LDX #$17         ;SET ERROR MESSAGE POINTER
A573  4C 37 A4           JMP L10          ;OUTPUT ERROR MESSAGE
A576  4C CA AA     L46   JMP L199         ;OUTPUT CARRAGE RETURN
A579                     .END
A579                     .LIB B2
A579               ;*************************************
A579               ;*CRUNCH KEYWORDS INTO BASIC TOKENS.
A579               ;*THIS PARSES THE CONTENTS OF THE BASIC
A579               ;*INPUT BUFFER LOOKING FOR KEYWORDS STORED
A579               ;*IN THE KEYWORD TABLE. ON FINDING A
A579               ;*KEYWORD IT IS CONVERTED INTO A SINGLE
A579               ;*BYTE TOKEN, THIS COMPACTS THE STORAGE OF
A579               ;*A BASIC PROGRAM. CHARACTERS INSIDE " "
A579               ;*ARE NOT CHECKED FOR KEYWORDS. THE
A579               ;*POINTER TO THE TABLE OF KEYWORDS IS
A579               ;*STORED IN LOCATION $0B AND LINE NUMBER
A579               ;*IS STORED IN $14 AND $15.
A579               ;*************************************
A579  6C 04 03     L45   JMP ($0304)      ;INDIRECT JUMP TO CRUNCH VECTOR
A57C  A6 7A              LDX TXTPTR       ;X = LSB CHARGET POINTER TO BUFFER
A57E  A0 04              LDY #$04         ;SET Y TO OFFSET ON SAVE
A580  84 0F              STY DATSCN
A582  BD 00 02     L29   LDA BUF,X        ;GET CHARACTER FROM BUFFER
A585  10 07              BPL L51          ;SHIFTED CHARACTER TRAP
A587  C9 FF              CMP #$FF         ;IS IT  PI ?
A589  F0 3E              BEQ L63          ;SAVE BASIC CODE
A58B  E8                 INX
A58C  D0 F4              BNE L29          ;NEXT CHARACTER
A58E  C9 20        L51   CMP #$20         ;SPACE CHARACTER ?
A590  F0 37              BEQ L63          ;SAVE BASIC CODE
A592  85 08              STA ENDCHR       ;STORE CHARACTER IN SCAN QUOTES FLAG
A594  C9 22              CMP #$22         ;QUOTES CHARACTER?
A596  F0 56              BEQ L60          ;COPY IN QUOTES TEXT
A598  24 0F              BIT DATSCN       ;IS BIT 2 SET PUT BIT 6 IN V REG
```

```
LOC   CODE        LINE

A59A  70 2D             BVS L63         ;IS V REG SET-CHECK ALPHA AND SAVE
A59C  C9 3F             CMP #$3F        ;QUESTION MARK CHARACTER?
A59E  D0 04             BNE L49         ;IF NOT SKIP
A5A0  A9 99             LDA #$99        ;SET TO PRINT TOKEN
A5A2  D0 25             BNE L63         ;SAVE BASIC CODE
A5A4  C9 30       L49   CMP #$30         ;CHECK FOR ZERO CHARACTER
A5A6  90 04             BCC L53         ;SMALLER?
A5A8  C9 3C             CMP #$3C        ;CHARACTER IS '<'
A5AA  90 1D             BCC L63         ;SAVE BASIC CODE
A5AC  84 71       L53   STY FBUFPT       ;STORE POINTER TO BUFFER SAVE LOC.
A5AE  A0 00             LDY #$00
A5B0  84 0B             STY COUNTB      ;SET INPUT BUFFER POINTER TO ZERO
A5B2  88                DEY
A5B3  86 7A             STX TXTPTR      ;RESTORE CHARGET POINTER LSB
A5B5  CA                DEX
A5B6  C8          L54   INY
A5B7  E8                INX
A5B8  BD 00 02    L55   LDA BUF,X        ;GET CHARACTER FROM BUFFER
A5BB  38                SEC
A5BC  F9 9E A0          SBC $A09E,Y     ;COMPARE WITH BASIC WORD LIST
A5BF  F0 F5             BEQ L54         ;IF SO GET NEXT CHARACTER AND DO AGAIN
A5C1  C9 80             CMP #$80        ;LAST CHARACTER ?
A5C3  D0 30             BNE L52
A5C5  05 0B             ORA COUNTB      ;FOUND THEN SET FLAG
A5C7  A4 71       L62   LDY FBUFPT       ;RESTORE Y
A5C9  E8          L63   INX              ;START OF SAVE BASIC CODE ROUTINE
A5CA  C8                INY
A5CB  99 FB 01          STA BUF-5,Y     ;SAVE CHARACTER TO BUFFER
A5CE  B9 FB 01          LDA BUF-5,Y     ;SET STATUS REGISTER FLAGS
A5D1  F0 36             BEQ L61         ;IF FINISHED THEN END
A5D3  38                SEC
A5D4  E9 3A             SBC #$3A        ;CHECK FOR COLON CHARACTER
A5D6  F0 04             BEQ L50         ;IF EQUAL GO CHECK FOR REM
A5D8  C9 49             CMP #$49        ;IS CHARACTER A DATA TOKEN
A5DA  D0 02             BNE L58
A5DC  85 0F       L50   STA DATSCN       ;IF SO THEN SET DATA SCAN FLAG
A5DE  38          L58   SEC
A5DF  E9 55             SBC #$55        ;IS CHARACTER A REM TOKEN
A5E1  D0 9F             BNE L29         ;IF NOT GET NEXT CHARACTER
A5E3  85 08             STA ENDCHR      ;SET SCAN QUOTES FLAG IF REM
A5E5  BD 00 02    L59   LDA BUF,X        ;GET CHARACTER FROM BUFFER
A5E8  F0 DF             BEQ L63         ;SAVE BASIC CODE
A5EA  C5 08             CMP ENDCHR      ;CHECK FOR END
A5EC  F0 DB             BEQ L63         ;SAVE BASIC CODE
A5EE  C8          L60   INY
A5EF  99 FB 01          STA BUF-5,Y     ;SAVE TO INPUT BUFFER
A5F2  E8                INX
A5F3  D0 F0             BNE L59         ;DO NEXT CHARACTER TILL END
A5F5  A6 7A       L52   LDX TXTPTR       ;SET X TO LSB CHARGET BUFFER POINTER
A5F7  E6 0B             INC COUNTB      ;INCREMENT INPUT BUFFER POINTER
A5F9  C8          L56   INY
A5FA  B9 9D A0          LDA $A09D,Y
A5FD  10 FA             BPL L56         ;IF A LESS THAN 127 GET NEXT CHAR
A5FF  B9 9E A0          LDA $A09E,Y     ;COMPARE WITH KEYWORD LIST
A602  D0 B4             BNE L55
A604  BD 00 02          LDA BUF,X
A607  10 BE             BPL L62
A609  99 FD 01    L61   STA BUF-3,Y
A60C  C6 7B             DEC TXTPTR+1    ;DECREMENT CHARGET BUFFER POINTER
A60E  A9 FF             LDA #$FF
A610  85 7A             STA TXTPTR      ;SET BUFFER POINTER TO $FF
```

```
LOC   CODE        LINE

A612  60            RTS
A613              ;***************************************
A613              ;*SEARCH BASIC FOR GIVEN LINE NUMBER.
A613              ;*THE STARTING ADDRESS OF A BASIC LINE IS
A613              ;*FOUND BY THIS ROUTINE. THE DESIRED LINE
A613              ;*NUMBER IS STORED IN LOCATIONS $14 AND $15.
A613              ;*IF THE LINE NUMBER IS FOUND THEN THE
A613              ;*LOCATION OF THE START OF ITS LINK ADDRESS
A613              ;*IS STORED IN LOCATIONS $5F AND $60. IF
A613              ;*THE DESIRED LINE NUMBER DOES EXIST THEN
A613              ;*THE ROUTINE EXITS WITH THE CARRY FLAG
A613              ;*CLEAR.
A613              ;***************************************
A613  A5 2B       L57   LDA TXTTAB      ;START OF BASIC LSB
A615  A6 2C             LDX TXTTAB+1    ;START OF BASIC MSB
A617  A0 01       L32   LDY #$01
A619  85 5F             STA FACEXP-2    ;SAVE START OF BASIC POINTER IN
A61B  86 60             STX FACEXP-1    ;WORK AREA
A61D  B1 5F             LDA (FACEXP-2),Y
A61F  F0 1F             BEQ L67         ;NOT FOUND CLEAR CARRY AND END
A621  C8                INY
A622  C8                INY
A623  A5 15             LDA LINNUM+1    ;SOUGHT LINE NUMBER HIGH BYTE
A625  D1 5F             CMP (FACEXP-2),Y ;COMPARE WITH CURRENT
A627  90 18             BCC L64         ;IF SMALLER THEN END
A629  F0 03             BEQ L68         ;IF SAME THEN CHECK LOW BYTE
A62B  88                DEY
A62C  D0 09             BNE L66
A62E  A5 14       L68   LDA LINNUM      ;SOUGHT LINE NUMBER LOW BYTE
A630  88                DEY
A631  D1 5F             CMP (FACEXP-2),Y ;COMPARE WITH CURRENT
A633  90 0C             BCC L64         ;IF SMALLER END
A635  F0 0A             BEQ L64         ;IF EQUAL END
A637  88          L66   DEY
A638  B1 5F             LDA (FACEXP-2),Y ;STORE IN X INDEX
A63A  AA                TAX
A63B  88                DEY
A63C  B1 5F             LDA (FACEXP-2),Y ;STORE IN ACCUMULATOR
A63E  B0 D7             BCS L32
A640  18          L67   CLC             ;SET CARRY CLEAR IF NOT FOUND
A641  60          L64   RTS
A642              ;***************************************
A642              ;*PERFORM NEW.
A642              ;*THIS PLACES ZERO INTO THE FIRST TWO BYTES
A642              ;*OF BASIC RAM PROGRAM STORAGE. THE END
A642              ;*OF BASIC POINTERS $2D - $2E ARE THEN
A642              ;*LOADED WITH THE ADDRESS OF THE START OF
A642              ;*BASIC STORAGE AREA + 2 BYTES. FINALLY THE
A642              ;*ROUTINE AT $A68E SETS THE CHARGET POINTERS
A642              ;*$7A - $7B TO POINT TO THE START OF BASIC
A642              ;*STORAGE -1. THE CLR ROUTINE IS THEN
A642              ;*ENTERED.
A642              ;***************************************
A642  D0 FD       NEW   BNE L64         ;IF NOT NULL OR : SYNTAX ERROR
A644  A9 00       L65   LDA #$00
A646  A8                TAY
A647  91 2B             STA (TXTTAB),Y ;SET FIRST 2 BYTES OF BASIC TO 0
A649  C8                INY
A64A  91 2B             STA (TXTTAB),Y
A64C  A5 2B             LDA TXTTAB
A64E  18                CLC
```

```
LOC   CODE        LINE

A64F  69 02             ADC #$02        ;BOTTOM OF VARIABLES = BOTTOM OF
A651  85 2D             STA VARTAB      ;BASIC +2
A653  A5 2C             LDA TXTTAB+1
A655  69 00             ADC #$00
A657  85 2E             STA VARTAB+1
A659  20 8E A6    L672  JSR L70          ;RESET CHARGET TO START OF BASIC
A65C  A9 00             LDA #$00        ;PROGRAM AND SET Z FLAG FOR CLR
A65E              ;****************************************
A65E              ;*PERFORM CLR.
A65E              ;*THE CLR FUNCTION SETS STRING POINTERS
A65E              ;*$33-$34  EQUAL TO THE TOP OF MEMORY
A65E              ;*POINTERS $37-$38 AND START OF ARRAYS
A65E              ;*$2F-$30 EQUAL TO THE START OF VARIABLES
A65E              ;*$2D-$2E THUS ERASING ALL VARIABLE
A65E              ;*STORAGE POINTERS. THE I/O POINTERS ARE
A65E              ;*RETURNED TO DEFAULT VALUES AND THE
A65E              ;*STACK POINTER RESET TO REMOVE UNWANTED
A65E              ;*STACK VARIABLES. PERFORMS RESTORE AND
A65E              ;*BLOCKS CONT COMMAND.
A65E              ;****************************************
A65E  D0 2D       CLR   BNE L21
A660  20 E7 FF    L36   JSR L625         ;CLOSE ALL I/O CHANNELS
A663  A5 37       L123  LDA MEMTOP       ;LIMIT OF MEMORY LO
A665  A4 38             LDY MEMTOP+1    ;LIMIT OF MEMORY HI
A667  85 33             STA FRETOP      ;STRING STORAGE LO
A669  84 34             STY FRETOP+1    ;STRING STORAGE HI
A66B  A5 2D             LDA VARTAB      ;START OF VARIABLES LO
A66D  A4 2E             LDY VARTAB+1    ;START OF VARIABLES HI
A66F  85 2F             STA ARYTAB      ;START OF ARRAYS LO
A671  84 30             STY ARYTAB+1    ;START OF ARRAYS HI
A673  85 31             STA STREND      ;END OF ARRAYS LO
A675  84 32             STY STREND+1    ;END OF ARRAYS HI
A677  20 1D A8    L618  JSR L110         ;RESTORE COMMAND
A67A  A2 19       L636  LDX #$19
A67C  86 16             STX TEMPPT      ;RESET SUBSCRIPT FLAG
A67E  68                PLA
A67F  A8                TAY
A680  68                PLA
A681  A2 FA             LDX #$FA
A683  9A                TXS             ;INITIALISE STACK POINTER
A684  48                PHA
A685  98                TYA
A686  48                PHA
A687  A9 00             LDA #$00
A689  85 3E             STA OLDTXT+1    ;BLOCK POINTER FOR CONT
A68B  85 10             STA SUBFLG
A68D  60          L21   RTS
A68E              ;****************************************
A68E              ;*RESET BASIC EXECUTION TO START.
A68E              ;*RESETS THE CHARGET POINTERS TO POINT TO THE
A68E              ;*START OF A BASIC PROGRAM, BY ADDING $FFFF
A68E              ;*TO THE START OF BASIC POINTER IN $2B-$2C
A68E              ;*AND STORING THE RESULT IN $7A-$7B.
A68E              ;****************************************
A68E  18          L70   CLC
A68F  A5 2B             LDA TXTTAB      ;BASIC START
A691  69 FF             ADC #$FF        ;MINUS 1
A693  85 7A             STA TXTPTR
A695  A5 2C             LDA TXTTAB+1
A697  69 FF             ADC #$FF
A699  85 7B             STA TXTPTR+1
```

```
LOC   CODE         LINE

A69B  60              RTS
A69C                ;****************************************
A69C                ;*PERFORM LIST.
A69C                ;*THE ROUTINE FIRST CHECKS AND SETS UP THE
A69C                ;*PARAMETERS, CONVERTING THE LINE NUMBER
A69C                ;*FROM FLOATING POINT INTO A MEMORY ADDRESS
A69C                ;*FOR THE START OF THE LINK ADDRESS OF THE
A69C                ;*BASIC LINE IN MEMORY. THE START ADDRESS OF
A69C                ;*THE LOWEST LINE NUMBER IS STORED IN LOCATION
A69C                ;*$5F-$60 AND THE HIGHEST LINE NUMBER IN
A69C                ;*$14-$15. IF NO PARAMETERS ARE GIVEN IN THE
A69C                ;*COMMAND THEN THE LOWEST START ADDRESS
A69C                ;*DEFAULTS TO $0801 AND THE HIGHEST TO $FFFF.
A69C                ;TWO IMPORTANT AND USEFUL SUBROUTINES ARE
A69C                ;*USED: $A6C9 LISTS A LINE OF BASIC POINTED
A69C                ;*TO BY $14-$15 TO THE OUTPUT DEVICE AND
A69C                ;*$A717 WHICH CONVERTS A TOKEN VALUE STORED
A69C                ;*IN THE ACCUMULATOR INTO THE BASIC KEYWORD.
A69C                ;****************************************
A69C  90 06         LIST   BCC L69          ;LINE NUMBER PRESENT
A69E  F0 04                BEQ L69          ;NO LINE RANGE
A6A0  C9 AB                CMP #$AB         ;CHECK FOR '-' SIGN
A6A2  D0 E9                BNE L21          ;CAUSE SYNTAX ERROR
A6A4  20 8B A9      L69    JSR L144          ;GET LINE NUMBER
A6A7  20 13 A6             JSR L57          ;FIND LINE NUMBER ADDRESS
A6AA  20 79 00             JSR CHRGOT       ;GET CURRENT CHARACTER
A6AD  F0 0C                BEQ L73          ;ONLY ONE RANGE SPECIFIED
A6AF  C9 AB                CMP #$AB         ;CHECK FOR '-' SIGN
A6B1  D0 8E                BNE L64          ;CAUSE SYNTAX ERROR
A6B3  20 73 00             JSR CHRGET       ;GET NEXT CHARACTER
A6B6  20 8B A9             JSR L144         ;GET LINE NUMBER
A6B9  D0 86                BNE L64          ;CAUSE SYNTAX ERROR
A6BB  68            L73    PLA               ;REMOVE STACK RETURN ADDRESS
A6BC  68                   PLA
A6BD  A5 14                LDA LINNUM
A6BF  05 15                ORA LINNUM+1     ;CHECK FOR LINE NUMBER ZERO
A6C1  D0 06                BNE L74          ;LIST LINE
A6C3  A9 FF                LDA #$FF         ;SET MAXIMUM LINE #
A6C5  85 14                STA LINNUM       ;END LINE NUMBER LO
A6C7  85 15                STA LINNUM+1     ;END LINE NUMBER HI
A6C9  A0 01         L74    LDY #$01          ;LINE LIST ENTRY POINT
A6CB  84 0F                STY DATSCN       ;SET QUOTES FLAG TO 1
A6CD  B1 5F                LDA (FACEXP-2),Y ;LINK ADDRESS HI
A6CF  F0 43                BEQ L81          ;IF 0 END AND GOTO WARM START
A6D1  20 2C A8             JSR L108         ;TEST FOR STOP KEY
A6D4  20 D7 AA             JSR L48          ;NEW LINE PRINT CR
A6D7  C8                   INY
A6D8  B1 5F                LDA (FACEXP-2),Y ;LINE NUMBER LO
A6DA  AA                   TAX
A6DB  C8                   INY
A6DC  B1 5F                LDA (FACEXP-2),Y ;LINE NUMBER HI
A6DE  C5 15                CMP LINNUM+1     ;CHECK IF LAST LINE # IN RANGE
A6E0  D0 04                BNE L75          ;MSB DIFFERENT
A6E2  E4 14                CPX LINNUM       ;CHECK IF LAST LINE # IN RANGE
A6E4  F0 02                BEQ L78
A6E6  B0 2C         L75    BCS L81           ;IF BIGGER GOTO WARM START
A6E8  84 49         L78    STY FORPNT        ;SAVE LINE NUMBER INDEX
A6EA  20 CD BD             JSR L24          ;PRINT LINE # X=LO,A=HI
A6ED  A9 20                LDA #$20         ;SPACE FIRST CHARACTER
A6EF  A4 49         L79    LDY FORPNT        ;RESTORE LINE NUMBER INDEX
A6F1  29 7F                AND #$7F         ;CLEAR BIT 7
```

```
LOC   CODE        LINE

A6F3  20 47 AB    L88     JSR L18          ;PRINT CHARACTER
A6F6  C9 22               CMP #$22         ;IS IT A '"' CHARACTER
A6F8  D0 06               BNE L84          ;NO
A6FA  A5 0F               LDA DATSCN       ;GET QUOTES FLAG
A6FC  49 FF               EOR #$FF         ;TOGGLE
A6FE  85 0F               STA DATSCN       ;RESTORE QUOTES FLAG
A700  C8          L84     INY              ;IF LINE TOO LONG THEN
A701  F0 11               BEQ L81          ;EXIT TO WARM START
A703  B1 5F               LDA (FACEXP-2),Y ;GET CHARACTER
A705  D0 10               BNE L76          ;CONVERT TOKEN TO KEYWORD
A707  A8                  TAY
A708  B1 5F               LDA (FACEXP-2),Y ;START ADDRESS OF NEXT LINE
A70A  AA                  TAX
A70B  C8                  INY
A70C  B1 5F               LDA (FACEXP-2),Y
A70E  86 5F               STX FACEXP-2     ;ADDRESS OF NEXT LINE LO
A710  85 60               STA FACEXP-1     ;ADDRESS OF NEXT LINE HI
A712  D0 B5               BNE L74          ;DO NEXT LINE
A714  4C 86 E3    L81     JMP L659         ;JUMP TO WARM START
A717              ;****************************************
A717              ;*CONVERT BASIC TOKENS TO KEYWORDS.
A717              ;*TAKES THE VALUE STORED IN THE ACCUMULATOR
A717              ;*CHECKS FOR VALID TOKEN AND PI ALSO
A717              ;*CHECKS IF QUOTES ARE OPEN OR CLOSED AND
A717              ;*THEN CONVERTS TOKEN TO COMMAND KEYWORD.
A717              ;****************************************
A717  6C 06 03    L76     JMP ($0306)      ;INDIRECT TOKEN TO TEXT VECTOR
A71A  10 D7               BPL L88          ;NOT A TOKEN, PRINT AND GET NEXT
A71C  C9 FF               CMP #$FF         ;IS IT PI?
A71E  F0 D3               BEQ L88          ;YES THEN PRINT AND GET NEXT
A720  24 0F               BIT DATSCN       ;CHECK FOR QUOTES
A722  30 CF               BMI L88          ;QUOTES OPEN THEN PRINT TEXT & NEXT
A724  38                  SEC
A725  E9 7F               SBC #$7F         ;UNSET HIGH BIT
A727  AA                  TAX
A728  84 49               STY FORPNT       ;SAVE Y
A72A  A0 FF               LDY #$FF
A72C  CA          L82     DEX
A72D  F0 08               BEQ L86          ;KEYWORD FOUND
A72F  C8          L87     INY
A730  B9 9E A0            LDA $A09E,Y      ;GET CHARACTER FROM KEYWORD TABLE
A733  10 FA               BPL L87          ;DO UNTIL END OF WORD
A735  30 F5               BMI L82          ;FOUND END OF WORD TRY NEXT
A737  C8          L86     INY
A738  B9 9E A0            LDA $A09E,Y      ;GET CHARACTER FROM KEYWORD TABLE
A73B  30 B2               BMI L79          ;OUTPUT LAST CHARACTER
A73D  20 47 AB            JSR L18          ;OUTPUT CHARACTER
A740  D0 F5               BNE L86          ;DO NEXT CHARACTER
A742              ;****************************************
A742              ;*PERFORM FOR-NEXT FUNCTION.
A742              ;*THIS ASSIGNS 18 BYTES ON THE STACK FOR
A742              ;*AN ACTIVE FOR LOOP, HAVING CHECKED THAT
A742              ;*THERE IS SPACE ON THE STACK (THIS ACCOUNTS
A742              ;*FOR THE MAXIMUM NESTING LEVEL OF 10 LOOPS).
A742              ;*THE FORMAT OF THE STACK ENTRY FOR AN
A742              ;*ACTIVE FOR IS:
A742              ;*STACK ADDRESS  1 LOOP RETURN ADDRESS LO
A742              ;*               2 LOOP RETURN ADDRESS HI
A742              ;*               3 RETURN LINE # HI
A742              ;*               4 RETURN LINE # LO
A742              ;*               5 'TO' VALUE IN
```

```
LOC   CODE        LINE

A742              ;*               6 FLOATING POINT
A742              ;*               7 NOTATION
A742              ;*               8   "
A742              ;*               9   "
A742              ;*              10 SIGN OF STEP
A742              ;*              11 'STEP' VALUE IN
A742              ;*              12 FLOATING POINT
A742              ;*              13 NOTATION
A742              ;*              14   "
A742              ;*              15   "
A742              ;*              16 VARIABLE ADDRESS HI
A742              ;*              17 VARIABLE ADDRESS LO
A742              ;*              18 TERMINATOR VALUE $81
A742              ;*****************************************
A742  A9 80       FOR     LDA #$80          ;BLOCK INTEGER
A744  85 10               STA SUBFLG
A746  20 A5 A9            JSR L146          ;PERFORM 'LET' FOR 'FOR' VARIABLE
A749  20 8A A3            JSR L0            ;FIND ACTIVE 'FOR-NEXT' ON STACK
A74C  D0 05               BNE L85           ;NOT FOUND
A74E  8A                  TXA
A74F  69 0F               ADC #$0F          ;INCREASE STACK POINTER BY 16
A751  AA                  TAX
A752  9A                  TXS
A753  68          L85     PLA               ;GET RETURN ADDRESS FROM STACK
A754  68                  PLA
A755  A9 09               LDA #$09
A757  20 FB A3            JSR L6            ;TEST IF STACK TOO DEEP
A75A  20 06 A9            JSR L131          ;SCAN FOR NEXT STATEMENT
A75D  18                  CLC
A75E  98                  TYA
A75F  65 7A               ADC TXTPTR        ;SET CHARGET PROGRAM POINTERS TO
A761  48                  PHA               ;NEXT COMMAND AND PUSH VALUES ONTO
A762  A5 7B               LDA TXTPTR+1      ;THE STACK
A764  69 00               ADC #$00
A766  48                  PHA
A767  A5 3A               LDA CURLIN+1      ;SAVE LINE NUMBER ON STACK
A769  48                  PHA
A76A  A5 39               LDA CURLIN
A76C  48                  PHA
A76D  A9 A4               LDA #$A4          ;'TO'COMMAND TOKEN
A76F  20 FF AE            JSR L242          ;COMPARE WITH CODE
A772  20 8D AD            JSR L96           ;CHECK VARIABLE IS NUMERIC
A775  20 8A AD            JSR L253          ;GET NUMBER FROM FAC
A778  A5 66               LDA FACSGN        ;CHECK ON SIGN
A77A  09 7F               ORA #$7F          ;IS IT NEGATIVE
A77C  25 62               AND FACHO
A77E  85 62               STA FACHO         ;SET MANTISSA
A780  A9 8B               LDA #$8B          ;SAVE RETURN ADDRESS LO
A782  A0 A7               LDY #$A7          ;SAVE RETURN ADDRESS HI
A784  85 22               STA INDEX
A786  84 23               STY INDEX+1
A788  4C 43 AE            JMP L101          ;PLACE 'TO'VALUE ON STACK
A78B  A9 BC               LDA #$BC          ;LOAD CONSTANTS
A78D  A0 B9               LDY #$B9          ;TO PUT DEFAULT STEP
A78F  20 A2 BB            JSR L496          ;1 IN FAC
A792  20 79 00            JSR CHRGOT        ;CHECK IF STEP PRESENT
A795  C9 A9               CMP #$A9          ;CHECK STEP TOKEN
A797  D0 06               BNE L91           ;IF NOT DEFAULT
A799  20 73 00            JSR CHRGET        ;GET NEXT CHARACTER
A79C  20 8A AD            JSR L253          ;GET VALUE INTO FAC
A79F  20 2B BC    L91     JSR L597          ;GET SIGN
```

```
LOC   CODE        LINE

A7A2  20 38 AE          JSR L275        ;PUTS STEP AND SIGN ON STACK
A7A5  A5 4A             LDA FORPNT+1    ;PUSH VARIABLE
A7A7  48                PHA             ;ADDRESS ONTO
A7A8  A5 49             LDA FORPNT      ;STACK
A7AA  48                PHA
A7AB  A9 81             LDA #$81        ;PUSH 'FOR' TERMINATOR
A7AD  48                PHA             ;TOKEN ONTO STACK
A7AE                ;***************************************
A7AE                ;*MAIN BASIC CONTROL LOOP.
A7AE                ;*THIS SEQUENCE OF ROUTINES CONTROLS THE
A7AE                ;*EXECUTION OF A BASIC PROGRAM, IT HAS THE
A7AE                ;*FOLLOWING SEQUENCE OF OPERATIONS:
A7AE                ;*1... CHECK FOR STOP KEY, IF PRESSED THEN
A7AE                ;*      EXIT LOOP.
A7AE                ;*2... CHECK FOR END OF A BASIC LINE OR A
A7AE                ;*      BASIC PROGRAM(TERMINATOR 0=END OF
A7AE                ;*      LINE, AND 00= END OF PROGRAM).
A7AE                ;*3... PUT THE NEXT CHARACTER OF THE BASIC
A7AE                ;*      PROGRAM INTO ACCUMULATOR USING THE
A7AE                ;*      CHARGET ROUTINE.
A7AE                ;*4... RUN THE NEXT ROUTINE AND RETURN TO
A7AE                ;*      $A7EA AND REPEAT LOOP.
A7AE                ;***************************************
A7AE  20 2C A8    L99   JSR L108         ;CHECK 'STOP' KEY
A7B1  A5 7A             LDA TXTPTR      ;GET PROGRAM POINTER LO
A7B3  A4 7B             LDY TXTPTR+1    ;GET PROGRAM POINTER HI
A7B5  C0 02             CPY #$02        ;DIRECT MODE?
A7B7  EA                NOP
A7B8  F0 04             BEQ L107        ;IF YES THEN JUMP
A7BA  85 3D             STA OLDTXT      ;SAVE PROGRAM POINTERS IN
A7BC  84 3E             STY OLDTXT+1    ;THE 'CONT' POINTER
A7BE  A0 00       L107  LDY #$00
A7C0  B1 7A             LDA (TXTPTR),Y ;GET CURRENT CHARACTER
A7C2  D0 43             BNE L109        ;END OF LINE?
A7C4  A0 02             LDY #$02
A7C6  B1 7A             LDA (TXTPTR),Y ;END OF PROGRAM?
A7C8  18                CLC             ;SET FLAG FOR END
A7C9  D0 03             BNE L102        ;NOT END
A7CB  4C 4B A8          JMP L118        ;EXECUTE END STATEMENT
A7CE  C8          L102  INY
A7CF  B1 7A             LDA (TXTPTR),Y ;GET LINE # LO
A7D1  85 39             STA CURLIN      ;STORE IN CURRENT LINE# LO
A7D3  C8                INY
A7D4  B1 7A             LDA (TXTPTR),Y ;GET LINE # HI
A7D6  85 3A             STA CURLIN+1    ;STORE IN CURRENT LINE # HI
A7D8  98                TYA
A7D9  65 7A             ADC TXTPTR      ;SET CHARGET PROGRAM POINTERS
A7DB  85 7A             STA TXTPTR      ;TO LINE
A7DD  90 02             BCC L104        ;IF GREATER THAN $FF THEN
A7DF  E6 7B             INC TXTPTR+1    ;BUMP HI BYTE
A7E1  6C 08 03    L104  JMP ($0308)      ;EXECUTE BASIC STATEMENT
A7E4  20 73 00          JSR CHRGET      ;GET NEXT CHARACTER
A7E7  20 ED A7          JSR L30         ;EXECUTE STATEMENT
A7EA  4C AE A7          JMP L99         ;JUMP TO START OF LOOP
A7ED                ;***************************************
A7ED                ;*EXECUTE BASIC STATEMENT.
A7ED                ;*THE CHARACTER OBTAINED BY CHARGET IN THE
A7ED                ;*PREVIOUS ROUTINE IS IN THE ACCUMULATOR
A7ED                ;*AND IS FIRST CHECKED TO SEE IF IT IS A
A7ED                ;*SEPARATING COLON BETWEEN BASIC STATEMENTS,
A7ED                ;*IF SO THEN THE ROUTINE RETURNS TO THE
```

```
LOC   CODE        LINE

A7ED              ;*BASIC CONTROL LOOP AT $A7AE. IF A TOKEN
A7ED              ;*IS NOT THE FIRST CHARACTER FOUND IN A
A7ED              ;*STATEMENT THEN IT DEFAULTS TO A 'LET'
A7ED              ;*COMMAND. WHEN FOUND A TOKEN IS CHECKED
A7ED              ;*TO ENSURE THAT IT IS CONTAINED WITHIN
A7ED              ;*THE KEYWORD TABLE. THE TOKEN VALUE IS
A7ED              ;*USED AS A POINTER INTO THE KEYWORD TABLE
A7ED              ;*BY SUBTRACTING $80 FROM THE TOKEN VALUE
A7ED              ;*THEN DOUBLING IT. THIS VALUE CAN THEN BE
A7ED              ;*USED AS AN OFFSET POINTER TO OBTAIN THE
A7ED              ;*KEYWORD ADDRESS FROM THE ADDRESS TABLE
A7ED              ;*(STARTING AT $A00C). THE TWO BYTE ADDRESS
A7ED              ;*THUS OBTAINED IS PUSHED ONTO THE STACK.
A7ED              ;*THE BASIC STATEMENT IS EXECUTED WHEN
A7ED              ;*CHARGET IS NEXT CALLED IN THE MAIN CONTROL
A7ED              ;*LOOP, THE 'RTS' AT THE END OF CHARGET RETURNING
A7ED              ;*TO THE BASIC ROUTINE RATHER THAN ITS NORMAL
A7ED              ;*RETURN ADDRESS. IN SO DOING THE ACCUMULATOR
A7ED              ;*WILL HOLD THE NEXT CHARACTER OF THE BASIC
A7ED              ;*STATEMENT, THIS IS OFTEN A PARAMETER AND
A7ED              ;*SHOULD NOT BE IGNORED WHEN USING THE
A7ED              ;*FUNCTION ROUTINES IN THE BASIC INTERPRETER.
A7ED              ;****************************************
A7ED  F0 3C       L30    BEQ L114          ;IF END OF LINE THEN END
A7EF  E9 80       L106   SBC #$80          ;CHECK FOR TOKEN AT START
A7F1  90 11              BCC L143          ;DEFAULT TO 'LET' COMMAND
A7F3  C9 23              CMP #$23
A7F5  B0 17              BCS L112          ;FUNCTION TOKEN OR GOTO
A7F7  0A                 ASL A             ;BASIC CODE TIMES 2
A7F8  A8                 TAY
A7F9  B9 0D A0           LDA $A00D,Y       ;GET COMMAND ADDRESS HI
A7FC  48                 PHA               ;PUT ON STACK
A7FD  B9 0C A0           LDA $A00C,Y       ;GET COMMAND ADDRESS LO
A800  48                 PHA               ;PUSH TO STACK
A801  4C 73 00           JMP CHRGET        ;GET NEXT CHARACTER AND EXECUTE
A804  4C A5 A9    L143   JMP L146           ;JUMP TO 'LET' COMMAND
A807  C9 3A       L109   CMP #$3A          ;IS IT A COLON
A809  F0 D6              BEQ L104          ;RETURN TO CONTROL LOOP
A80B  4C 08 AF    L103   JMP L94           ;GENERATE SYNTAX ERROR
A80E  C9 4B       L112   CMP #$4B          ;CHECK FOR 'GO'TOKEN (-$80)
A810  D0 F9              BNE L103          ;IF NO THEN SYNTAX ERROR
A812  20 73 00           JSR CHRGET        ;GET NEXT CHARACTER
A815  A9 A4              LDA #$A4          ;CHECK FOR 'TO'TOKEN (-$80)
A817  20 FF AE           JSR L242          ;CHARACTER CHECK
A81A  4C A0 A8           JMP L124          ;JUMP TO 'GOTO'COMMAND
A81D              ;****************************************
A81D              ;*PERFORM RESTORE COMMAND.
A81D              ;*SETS THE DATA STATEMENT POINTER TO THE
A81D              ;*START OF BASIC PROGRAM STORAGE ($0800).
A81D              ;*THIS POINTER IS STORED IN LOCATIONS
A81D              ;*$41-$42.
A81D              ;****************************************
A81D  38          L110   SEC
A81E  A5 2B              LDA TXTTAB        ;START OF BASIC LO
A820  E9 01              SBC #$01          ;LESS 1
A822  A4 2C              LDY TXTTAB+1      ;START OF BASIC HI
A824  B0 01              BCS L72           ;DECREMENT IF CARRY SET
A826  88                 DEY
A827  85 41       L72    STA DATPTR        ;STORE IN DATA ADDRESS LO
A829  84 42              STY DATPTR+1      ;STORE IN DATA ADDRESS HI
A82B  60          L114   RTS
```

```
LOC   CODE        LINE

A82C                      .END
A82C                      .LIB B3
A82C                ;*************************************
A82C                ;*PERFORM 'STOP' OR 'END'.
A82C                ;*THIS ROUTINE IS CALLED BY EITHER THE
A82C                ;*'STOP'KEY DETECT ROUTINE ($FFE1), THE
A82C                ;*ROUTINE AT $A7BE WHICH DETECTS THE
A82C                ;*TERMINATING DOUBLE ZERO BYTES OF A BASIC
A82C                ;*PROGRAM OR BY THE KEYWORD 'END'. WHICH
A82C                ;*ACTION IS PERFORMED IS DETERMINED BY THE
A82C                ;*ROUTINE TESTING THE STATE OF THE Z AND
A82C                ;*CARRY FLAGS IN THE PROCESSOR STATUS
A82C                ;*REGISTER. IF CARRY AND Z ARE BOTH SET
A82C                ;*THEN A 'STOP' BREAK IS INITIATED. 'END'
A82C                ;*IS PERFORMED IF CARRY IS CLEAR. NOTE
A82C                ;*THAT THIS ROUTINE IN THE STOP MODE DOES
A82C                ;*NOT CHANGE ANY OF THE POINTERS, THESE
A82C                ;*CAN THEREFORE BE USED BY THE 'CONT'
A82C                ;*COMMAND TO RESTORE THE RUNNING OF THE
A82C                ;*BASIC PROGRAM.
A82C                ;*************************************
A82C  20 E1 FF      L108    JSR L309          ;TEST FOR STOP KEY
A82F  B0 01                 BCS L77           ;IF STOP MAKE SURE CARRY SET
A831  18                    CLC               ;IF END CLEAR CARRY FLAG
A832  D0 3C         L77     BNE L121
A834  A5 7A                 LDA TXTPTR        ;GET CHARGET PROGRAM POINTER LO
A836  A4 7B                 LDY TXTPTR+1      ;GET CHARGET PROGRAM POINTER HI
A838  A6 3A                 LDX CURLIN+1      ;CURRENT BASIC LINE# HI
A83A  E8                    INX               ;IS IT DIRECT MODE?
A83B  F0 0C                 BEQ L116          ;YES
A83D  85 3D                 STA OLDTXT        ;SAVE PROG POINTER LO IN CONT POINTER
A83F  84 3E                 STY OLDTXT+1      ;SAVE PROG POINTER HI IN CONT POINTER
A841  A5 39                 LDA CURLIN        ;GET CURRENT BASIC LINE # LO
A843  A4 3A                 LDY CURLIN+1      ;GET CURRENT BASIC LINE # HI
A845  85 3B                 STA OLDLIN        ;SAVE IN PREVIOUS LINE # LO
A847  84 3C                 STY OLDLIN+1      ;SAVE IN PREVIOUS LINE # HI
A849  68            L116    PLA               ;GET RETURN ADDRESS FROM STACK
A84A  68                    PLA
A84B  A9 81         L118    LDA #$81          ;GET POINTER TO 'BREAK'
A84D  A0 A3                 LDY #$A3          ;MESSAGE
A84F  90 03                 BCC L105          ;IS IT 'END' (CARRY CLEAR)
A851  4C 69 A4              JMP L20           ;NO THEN OUTPUT 'BREAK IN ....'
A854  4C 86 E3      L105    JMP L659          ;JUMP TO BASIC WARM START
A857                ;*************************************
A857                ;*PERFORM CONT COMMAND.
A857                ;*THIS RESTORES THE LINE ADDRESS POINTER IN
A857                ;*CHARGOT AT LOCATIONS $7A AND $7B USING THE
A857                ;*CONTENTS OF THE POINTER TO BASIC STATEMENT
A857                ;*FOR CONT VARIABLES AT LOCATIONS $3D-$3E.
A857                ;*IT ALSO SETS THE CURRENT LINE NUMBER
A857                ;*VARIABLE IN $39-$3A EQUAL TO THE PREVIOUS
A857                ;*LINE NUMBER IN $3B-$3C. IF HOWEVER THE
A857                ;*CONTENTS OF $3E IS ZERO THEN A 'CANT
A857                ;*CONTINUE ERROR' IS GENERATED.
A857                ;*************************************
A857  D0 17         CONT    BNE L121          ;IF NULL OR : SYNTAX ERROR
A859  A2 1A                 LDX #$1A          ;SET X TO ERROR NUMBER
A85B  A4 3E                 LDY OLDTXT+1      ;IF ZERO CONT BLOCKED
A85D  D0 03                 BNE L119          ;JUMP TO DO CONT
A85F  4C 37 A4              JMP L10           ;OUTPUT ERROR MESSAGE
A862  A5 3D         L119    LDA OLDTXT        ;BASIC STATEMENT POINTER LO
```

```
LOC   CODE        LINE

A864  85 7A             STA TXTPTR      ;STORE IN CHARGET PROGRAM POINTER LO
A866  84 7B             STY TXTPTR+1    ;STORE IN CHARGET PROGRAM POINTER HI
A868  A5 3B             LDA OLDLIN      ;PREVIOUS LINE # LO
A86A  A4 3C             LDY OLDLIN+1    ;PREVIOUS LINE # HI
A86C  85 39             STA CURLIN      ;SAVE IN CURRENT LINE # LO
A86E  84 3A             STY CURLIN+1    ;SAVE IN CURRENT LINE # HI
A870  60          L121  RTS
A871              ;****************************************
A871              ;*PERFORM RUN COMMAND.
A871              ;*IF 'RUN' IS FOLLOWED BY A LINE NUMBER THEN
A871              ;*'RUN' CALLS THE 'CLR' ROUTINE TO CLEAR THE
A871              ;*CONTENTS OF VARIABLES AND STACK THEN JUMPS
A871              ;*TO THE 'GOTO' ROUTINE. IF 'RUN' IS NOT
A871              ;*FOLLOWED BY A LINE NUMBER THEN THE CHARGET
A871              ;*POINTERS AT $7A-$7B ARE SET TO THE START OF
A871              ;*BASIC PROGRAM STORAGE, THE 'CLR' ROUTINE
A871              ;*IS CALLED AND THE RUN INITIATED WITH A
A871              ;*RETURN TO THE MAIN BASIC CONTROL LOOP.
A871              ;****************************************
A871  08          RUN   PHP
A872  A9 00             LDA #$00
A874  20 90 FF          JSR L1216       ;SET STATUS FOR PROGRAM MODE
A877  28                PLP
A878  D0 03             BNE L117        ;GET LINE NUMBER
A87A  4C 59 A6          JMP L672        ;SET PROGRAM POINTER
A87D  20 60 A6    L117  JSR L36          ;DO 'CLR'COMMAND
A880  4C 97 A8          JMP L122        ;JUMP TO 'GOTO'COMMAND
A883              ;****************************************
A883              ;*PERFORM 'GOSUB' COMMAND.
A883              ;*THIS ROUTINE PUSHES THE SEVEN BYTES OF
A883              ;*DATA REQUIRED FOR A GOSUB ONTO THE STACK
A883              ;*HAVING FIRST CHECKED THAT THERE IS SPACE
A883              ;*ON THE STACK, IF NOT THEN AN 'OUT OF
A883              ;*MEMORY ERROR' IS GENERATED. THE FORMAT OF
A883              ;*A STACK ENTRY FOR AN ACTIVE GOSUB IS:
A883              ;*STACK ADDRESS  1.. $A7 CONSTANT
A883              ;*               2.. $E9 CONSTANT
A883              ;*               3.. RETURN ADDRESS HI
A883              ;*               4.. RETURN ADDRESS LO
A883              ;*               5.. LINE # LO
A883              ;*               6.. LINE # HI
A883              ;*               7.. $8D 'GOSUB' CODE
A883              ;****************************************
A883  A9 03       GOSUB LDA #$03
A885  20 FB A3          JSR L6          ;STACK TOO DEEP?
A888  A5 7B             LDA TXTPTR+1    ;CHARGET POINTER HI
A88A  48                PHA             ;PUSH TO STACK
A88B  A5 7A             LDA TXTPTR      ;CHARGET POINTER LO
A88D  48                PHA             ;PUSH TO STACK
A88E  A5 3A             LDA CURLIN+1    ;LINE NUMBER HI
A890  48                PHA             ;PUSH TO STACK
A891  A5 39             LDA CURLIN      ;LINE NUMBER LO
A893  48                PHA             ;PUSH TO STACK
A894  A9 8D             LDA #$8D        ;GOSUB TOKEN CODE
A896  48                PHA             ;PUSH TO STACK
A897  20 79 00    L122  JSR CHRGOT       ;CHARGOT GET LAST CHARACTER
A89A  20 A0 A8          JSR L124        ;PERFORM THE GOTO ROUTINE
A89D  4C AE A7          JMP L99         ;JUMP TO BASIC CONTROL LOOP
A8A0              ;****************************************
A8A0              ;*PERFORM 'GO-TO'COMMAND.
A8A0              :*THE LINE NUMBER USED IN THE 'GOTO' OR
```

```
LOC   CODE        LINE

A8A0              ;*'GOSUB' IS FIRST FETCHED AND STORED IN
A8A0              ;*LOCATIONS $14-$15. THE LINE NUMBER
A8A0              ;*CORRESPONDING TO THIS IS THEN SOUGHT,
A8A0              ;*WHEN FOUND IT LOADS THE LINE ADDRESS
A8A0              ;*POINTERS $7A-$7B. IF THE LINE NUMBER IS
A8A0              ;*NOT FOUND THEN AN 'UNDEFINED STATEMENT'
A8A0              ;*ERROR IS GENERATED.
A8A0              ;**************************************
A8A0  20 6B A9    L124    JSR L144        ;PUT LINE # IN $14-$15
A8A3  20 09 A9            JSR L93         ;GET START NEXT LINE
A8A6  38                  SEC
A8A7  A5 39               LDA CURLIN      ;CURRENT BASIC LINE # LO
A8A9  E5 14               SBC LINNUM      ;SMALLER THAN CHARGET POINTER LO?
A8AB  A5 3A               LDA CURLIN+1    ;CURRENT BASIC LINE # HI
A8AD  E5 15               SBC LINNUM+1    ;SMALLER THAN CHARGET POINTER HI?
A8AF  B0 0B               BCS L113        ;NEW LINE# LARGER THAN CURRENT LINE#
A8B1  98                  TYA
A8B2  38                  SEC
A8B3  65 7A               ADC TXTPTR      ;IF SMALLER SEARCH STARTING FROM
A8B5  A6 7B               LDX TXTPTR+1    ;CURRENT LINE  NUMBER
A8B7  90 07               BCC L126
A8B9  E8                  INX
A8BA  B0 04               BCS L126
A8BC  A5 2B       L113    LDA TXTTAB       ;LOOK FOR PROGRAM START
A8BE  A6 2C               LDX TXTTAB+1
A8C0  20 17 A6    L126    JSR L32          ;SEARCH BASIC FOR LINE #
A8C3  90 1E               BCC L129        ;NOT FOUND THEN 'UNDEFINED' ERROR
A8C5  A5 5F               LDA FACEXP-2
A8C7  E9 01               SBC #$01
A8C9  85 7A               STA TXTPTR      ;SET CHARGET POINTER TO NEW LINE
A8CB  A5 60               LDA FACEXP-1
A8CD  E9 00               SBC #$00
A8CF  85 7B               STA TXTPTR+1
A8D1  60          L127    RTS
A8D2              ;**************************************
A8D2              ;*PERFORM 'RETURN' COMMAND.
A8D2              ;*THIS CHECKS FOR THE 'GOSUB' POINTER ($8D)
A8D2              ;*ON THE STACK BY CALLING ROUTINE $A38A
A8D2              ;*THIS SEARCHES FOR 'FOR' ENTRIES ON THE
A8D2              ;*STACK WHICH ARE THEN SKIPPED AND THE
A8D2              ;*NEXT STACK ENTRY CHECKED FOR A
A8D2              ;*'GOSUB' IF FOUND THEN ALL HIGHER STACK
A8D2              ;*ENTRIES ARE ERASED AND THE POINTERS TO
A8D2              ;*THE 'GOSUB' CALLING ROUTINE RECOVERED.
A8D2              ;*IF NO GOSUB POINTER IS FOUND THEN A
A8D2              ;*'RETURN WITHOUT GOSUB ERROR' IS CREATED.
A8D2              ;*THE ORIGINAL LINE NUMBER IS STORED IN
A8D2              ;*THE POINTERS $39-$3A. CHARGET IS RESET
A8D2              ;*USING THE RETURN ADDRESS POINTERS FROM
A8D2              ;*THE STACK.
A8D2              ;**************************************
A8D2  D0 FD       RETURN BNE L127          ;IF NOT NULL OR : SYNTAX ERROR
A8D4  A9 FF               LDA #$FF        ;MASK 'FOR' OPERATION
A8D6  85 4A               STA FORPNT+1
A8D8  20 8A A3            JSR L0          ;PEEK STACK FOR 'FOR' ACTIVITY
A8DB  9A                  TXS             ;SET NEW STACK POINTER
A8DC  C9 8D               CMP #$8D        ;GOSUB STACK IDENTIFIER CODE
A8DE  F0 0B               BEQ L141        ;FOUND GOSUB
A8E0  A2 0C               LDX #$0C
A8E2  2C                  .BYT $2C
A8E3  A2 11       L129    LDX #$11         ;SET 'UNDEFINED' ERROR #
```

```
LOC   CODE        LINE

A8E5  4C 37 A4            JMP L10         ;PRINT ERROR MESSAGE
A8E8  4C 08 AF    L128    JMP L94          ;PRINT 'SYNTAX ERROR'
A8EB  68          L141    PLA              ;RECOVER 'GOSUB'POINTERS FROM STACK
A8EC  68                  PLA
A8ED  85 39               STA CURLIN      ;LINE NUMBER LO
A8EF  68                  PLA
A8F0  85 3A               STA CURLIN+1    ;LINE # HI
A8F2  68                  PLA
A8F3  85 7A               STA TXTPTR      ;CHARGET PROGRAM POINTER LO
A8F5  68                  PLA
A8F6  85 7B               STA TXTPTR+1    ;CHARGET PROGRAM POINTER HI
A8F8              ;***************************************
A8F8              ;*PERFORM DATA COMMAND.
A8F8              ;*THIS IS PART OF THE 'RETURN' ROUTINE AND
A8F8              ;*IS USED TO SEARCH FOR THE NEXT BASIC
A8F8              ;*STATEMENT FOLLOWING A 'DATA' STATEMENT,
A8F8              ;*THEREBY IGNORING THE DATA FOLLOWING THE
A8F8              ;*DATA STATEMENT.
A8F8              ;***************************************
A8F8  20 06 A9    L130    JSR L131         ;FIND NEXT BASIC STATEMENT
A8FB  98          L218    TYA              ;GET OFFSET
A8FC  18                  CLC
A8FD  65 7A               ADC TXTPTR      ;ADD TO CHARGET PROGRAM POINTER
A8FF  85 7A               STA TXTPTR
A901  90 02               BCC L137
A903  E6 7B               INC TXTPTR+1
A905  60          L137    RTS
A906              ;***************************************
A906              ;*SCAN FOR NEXT BASIC STATEMENT OR LINE.
A906              ;*THE SCAN FOR THE NEXT STATEMENT LOOKS FOR
A906              ;*A COLON SEPARATOR BYTE. THE SCAN FOR THE
A906              ;*NEXT BASIC LINE WHICH STARTS AT $A909
A906              ;*LOOKS FOR A ZERO LINE TERMINATOR BYTE.
A906              ;*ON COMPLETION OF THE ROUTINE THE Y
A906              ;*REGISTER CONTAINS AN OFFSET FROM THE
A906              ;*ADDRESS POINTER IN CHARGET.
A906              ;***************************************
A906  A2 3A       L131    LDX #$3A         ;CHECK FOR : SEPARATOR
A908  2C                  .BYT $2C
A909  A2 00       L93     LDX #$00         ;END OF LINE TERMINATOR ZERO
A90B  86 07               STX CHARAC      ;PUT IN SEARCH CHARACTER
A90D  A0 00               LDY #$00
A90F  84 08               STY ENDCHR      ;INITIALISE OFFSET TO ZERO
A911  A5 08       L125    LDA ENDCHR
A913  A6 07               LDX CHARAC      ;DESIRED CHARACTER?
A915  85 07               STA CHARAC
A917  86 08               STX ENDCHR
A919  B1 7A       L133    LDA (TXTPTR),Y  ;GET CHARACTER
A91B  F0 E8               BEQ L137        ;IF END OF LINE THEN DONE
A91D  C5 08               CMP ENDCHR
A91F  F0 E4               BEQ L137        ;IF END THEN DONE
A921  C8                  INY             ;BUMP POINTER
A922  C9 22               CMP #$22        ;CHECK FOR QUOTES
A924  D0 F3               BNE L133        ;IF NOT DO AGAIN
A926  F0 E9               BEQ L125        ;GET NEXT CHARACTER
A928              ;***************************************
A928              ;*PERFORM 'IF' COMMAND.
A928              ;*THE EXPRESSION FOLLOWING THE 'IF' IS FIRST
A928              ;*EVALUATED BY THE ROUTINE AT $AD9E, THE
A928              ;*RESULT OF THE EVALUATED EXPRESSION BEING
A928              ;*PLACED IN THE FLOATING ACCUMULATOR #1, THE
```

```
LOC   CODE        LINE

A928              ;*EXPONENT VALUE IS ALSO PLACED IN THE
A928              ;*PROCESSOR ACCUMULATOR. IT THEN CHECKS THAT
A928              ;*THE FOLLOWING STATEMENT IS THE TOKEN FOR
A928              ;*EITHER 'GOTO' ($89) OR 'THEN' ($A7), IF
A928              ;*NOT THEN A SYNTAX ERROR IS GENERATED. IF
A928              ;*THE RESULT OF THE EVALUATION IS ZERO THEN
A928              ;*THE EXPONENT IN THE ACCUMULATOR IS SET TO
A928              ;*ZERO, IF THE ACCUMULATOR CONTAINS A ZERO
A928              ;*THEN THE CONDITION IS DEEMED FALSE AND
A928              ;*CONTROL BRANCHES TO THE NEXT LINE, BY ADDING
A928              ;*THE SCAN TO NEXT LINE OFFSET IN THE Y
A928              ;*INDEX REGISTER TO THE CONTENTS OF THE
A928              ;*CHARGET POINTERS $7A-$7B. IF THE ACCUMULATOR
A928              ;*CONTENTS ARE NOT EQUAL TO ZERO THEN THE
A928              ;*CONDITION IS TRUE, AND THE STATEMENT
A928              ;*FOLLOWING THE 'IF' IS EXECUTED, IF A 'GOTO'
A928              ;*OR A 'THEN' FOLLOWED BY A NUMBER, THEN THE
A928              ;*'GOTO' ROUTINE IS EXECUTED, OTHERWISE THE
A928              ;*BASIC VARIABLES ARE ASSIGNED.
A928              ;*************************************
A928  20 9E AD    IF      JSR L256        ;EVALUATE EXPRESSION
A92B  20 79 00            JSR CHRGOT      ;GET LAST CHARACTER
A92E  C9 89               CMP #$89        ;TEST FOR 'GOTO' TOKEN
A930  F0 05               BEQ L132        ;YES
A932  A9 A7               LDA #$A7        ;TEST FOR 'THEN' TOKEN
A934  20 FF AE            JSR L242        ;GET IF TERM
A937  A5 61       L132    LDA FACEXP      ;EXPRESSION TRUE
A939  D0 05               BNE L135        ;NO GOTO NEXT LINE
A93B              ;*************************************
A93B              ;*PERFORM 'REM' COMMAND.
A93B              ;*THE ROUTINE TO PERFORM THE 'REM' COMMAND
A93B              ;*IS PART OF THE 'IF' ROUTINE AND IS THE
A93B              ;*SAME AS THAT USED FOR A CONDITION FALSE,
A93B              ;*IT SKIPS THE REST OF THE LINE BY SETTING
A93B              ;*CHARGET POINTERS $7A-$7B TO THE START OF
A93B              ;*THE NEXT LINE BY ADDING TO THEIR CURRENT
A93B              ;*CONTENTS THE SCAN TO NEXT LINE ($A909)
A93B              ;*OFFSET IN THE Y. INDEX REGISTER.
A93B              ;*************************************
A93B  20 09 A9    REM     JSR L93         ;OFFSET TO NEXT LINE
A93E  F0 BB               BEQ L218        ;POINT TO LAST CHARACTER
A940  20 79 00    L135    JSR CHRGOT      ;GET LAST CHARACTER
A943  B0 03               BCS L136        ;NO DIGIT
A945  4C A0 A8            JMP L124        ;JUMP TO 'GOTO'ROUTINE
A948  4C ED A7    L136    JMP L30         ;EXECUTE BASIC STATEMENT
A94B              ;*************************************
A94B              ;*PERFORM 'ON' COMMAND.
A94B              ;*THIS CHECKS THE VARIABLE TYPE AND EVALUATES
A94B              ;*IT USING THE ROUTINE AT $B79E WHICH RETURNS
A94B              ;*THE VALUE IN LOCATION $65 AND THE X. INDEX
A94B              ;*REGISTER. IT THEN CHECKS WHETHER THE NEXT
A94B              ;*COMMAND FOLLOWING THE 'ON' IS A TOKEN
A94B              ;*FOR 'GOTO' ($89) OR 'GOSUB' ($8D), IF IT
A94B              ;*IS NEITHER OF THESE THEN A 'SYNTAX ERROR'
A94B              ;*IS GENERATED. IT THEN GOES THROUGH A LOOP
A94B              ;*WHICH FIRST DECREMENTS THE VALUE IN
A94B              ;*LOCATION $65 THEN GETS THE FIRST LINE NUMBER
A94B              ;*FROM THE LIST OF LINE NUMBERS FOLLOWING THE
A94B              ;*'GOTO / GOSUB' EXPRESSION AND CHECKS FOR A COMMA
A94B              ;*FOLLOWING IT. THIS LOOP IS THEN REPEATED,
A94B              ;*DECREMENT $65 AND GETS THE NEXT LINE
```

```
LOC   CODE        LINE

A94B              ;*NUMBER AND SO ON UNTIL EITHER THE VALUE IN
A94B              ;*$65 IS ZERO OR THE LINE NUMBERS ARE
A94B              ;*EXHAUSTED. IF THE CONTENTS OF $65 REACH
A94B              ;*ZERO THEN THE NEXT LINE NUMBER TO BE
A94B              ;*ACCESSED IS THE LINE TO WHICH PROGRAM
A94B              ;*CONTROL IS TRANSFERRED. IF THE CONTENTS OF
A94B              ;*$65 IS NOT ZERO AND THERE ARE NO MORE LINE
A94B              ;*NUMBERS THEN THE NEXT STATEMENT IS EXECUTED
A94B              ;*BY DEFAULT.
A94B              ;****************************************
A94B  20 9E B7    ON      JSR L195        ;GET SINGLE BYTE PARAMETER
A94E  48                  PHA             ;SAVE ACCUMULATOR
A94F  C9 8D               CMP #$8D        ;GOSUB TOKEN
A951  F0 04               BEQ L145        ;YES THEN JUMP NEXT
A953  C9 89       L138    CMP #$89         ;GOTO TOKEN
A955  D0 91               BNE L128        ;IF NO THEN SYNTAX ERROR
A957  C6 65       L145    DEC FACHO+3      ;DECREMENT POINTER
A959  D0 04               BNE L140        ;IF ZERO THEN EXIT LOOP
A95B  68                  PLA             ;RETRIEVE ACCUMULATOR
A95C  4C EF A7            JMP L106        ;EXECUTE COMMAND
A95F  20 73 00    L140    JSR CHRGET       ;GET NEXT CHARACTER
A962  20 6B A9            JSR L144        ;GET LINE NUMBER
A965  C9 2C               CMP #$2C        ;IS IT A COMMA?
A967  F0 EE               BEQ L145        ;YES THEN DO AGAIN
A969  68                  PLA             ;NO RESTORE ACCUMULATOR
A96A  60          L142    RTS              ;AND RETURN
A96B              ;****************************************
A96B              ;*GET FIXED POINT NUMBER FROM BASIC.
A96B              ;*ON ENTRY TO THIS ROUTINE THE ACCUMULATOR
A96B              ;*HOLD THE VALUE PREVIOUSLY READ BY CHARGET
A96B              ;*FROM THE BASIC PROGRAM, IF THIS IS NON
A96B              ;*NUMERIC THEN THE ROUTINE EXITS. THE ASCII
A96B              ;*VALUE SO OBTAINED IS CONVERTED INTO A
A96B              ;*NUMERIC VALUE BY SUBTRACTING $2F, THE
A96B              ;*NUMBER IS PLACED IN LOCATION $07. THE
A96B              ;*CONTENTS OF $07 IS ADDED TO THE CONTENTS
A96B              ;*OF LOCATIONS $14-$15, BOTH OF WHICH ARE
A96B              ;*INITIALLY SET TO ZERO. THE NEXT CHARACTER
A96B              ;*IS THEN OBTAINED FROM BASIC AND THE ROUTINE
A96B              ;*REPEATED, THIS TIME THE CONTENTS OF
A96B              ;*LOCATIONS $14-$15 ARE MULTIPLIED BY 10
A96B              ;*USING A SERIES OF SHIFTS AND ROTATIONS.
A96B              ;*THIS PROCESS IS REPEATED UNTIL A NON
A96B              ;*NUMERIC VALUE IS OBTAINED FROM BASIC. THIS
A96B              ;*THEN LEAVES THE INTEGER VALUE IN LOCATIONS
A96B              ;*$14-$15 AND THE CHARGOT POINTER $7A-$7B
A96B              ;*POINTING TO THE NEXT CHARACTER IN BASIC.
A96B              ;****************************************
A96B  A2 00       L144    LDX #$00         ;CLEAR NUMERIC
A96D  86 14               STX LINNUM      ;VALUE STORAGE
A96F  86 15               STX LINNUM+1    ;LOCATIONS
A971  B0 F7       L31     BCS L142         ;EXIT IF CARRY SET
A973  E9 2F               SBC #$2F        ;CONVERT TO NUMERIC BY -$2F
A975  85 07               STA CHARAC      ;SAVE IN SEARCH CHARACTER LOC
A977  A5 15               LDA LINNUM+1
A979  85 22               STA INDEX       ;PUT INTO UTILITY AREA
A97B  C9 19               CMP #$19
A97D  B0 D4               BCS L138        ;< $19 THEN SYNTAX ERROR
A97F  A5 14               LDA LINNUM      ;MULTIPLY BY 10
A981  0A                  ASL A
A982  26 22               ROL INDEX
```

```
LOC   CODE        LINE

A984  0A                ASL A
A985  26 22             ROL INDEX
A987  65 14             ADC LINNUM
A989  85 14             STA LINNUM
A98B  A5 22             LDA INDEX
A98D  65 15             ADC LINNUM+1
A98F  85 15             STA LINNUM+1
A991  06 14             ASL LINNUM
A993  26 15             ROL LINNUM+1
A995  A5 14             LDA LINNUM
A997  65 07             ADC CHARAC
A999  85 14             STA LINNUM
A99B  90 02             BCC L147        ;IF CARRY THEN BUMP MSB
A99D  E6 15             INC LINNUM+1
A99F  20 73 00    L147  JSR CHRGET       ;GET NEXT CHARACTER
A9A2  4C 71 A9          JMP L31         ;DO AGAIN
A9A5                    .END
A9A5                    .LIB B4
A9A5              ;****************************************
A9A5              ;*PERFORM 'LET' FUNCTION.
A9A5              ;*THE VARIABLE DEFINED IN THE 'LET' STATEMENT
A9A5              ;*IS FIRST SEARCHED FOR AMONGST EXISTING
A9A5              ;*BASIC VARIABLES USING THE ROUTINE AT $B08B,
A9A5              ;*IF IT DOES NOT YET EXIST THEN IT IS SET UP.
A9A5              ;*THE VARIABLE POINTER ADDRESS IS STORED IN
A9A5              ;*LOCATIONS $49-$4A. THE ROUTINE THEN CHECKS
A9A5              ;*FOR AN = SIGN (CHARACTER VALUE $B2) IF
A9A5              ;*THIS IS NOT FOUND THEN A SYNTAX ERROR IS
A9A5              ;*GENERATED. THE VALUE, STRING OR EXPRESSION
A9A5              ;*FOLLOWING THE EQUALS SIGN IS THEN EVALUATED
A9A5              ;*AND ASSIGNED TO THE CORRESPONDING VARIABLE
A9A5              ;*POINTED TO BY LOCATIONS $49-$4A. THE
A9A5              ;*FOLLOWING ARE THE START OF THE ROUTINES WHICH
A9A5              ;*ASSIGN THE DIFFERENT VARIABLE TYPES:
A9A5              ;*   $A9C4 - ASSIGN INTEGER VARIABLES
A9A5              ;*   $A9D6 - ASSIGN FLOATING POINT VARIABLES
A9A5              ;*   $AA2C - ASSIGN STRINGS EXCEPT:
A9A5              ;*   $A9D9 - ASSIGN TI$
A9A5              ;*THE ROUTINE WHICH ASSIGNS TI$ USES A ROUTINE
A9A5              ;*AT $AA1D WHICH ADDS AN ASCII DIGIT TO THE
A9A5              ;*CONTENTS OF FLOATING ACCUMULATOR #1, THE
A9A5              ;*DIGIT IS POINTED TO BY $22,Y.
A9A5              ;****************************************
A9A5  20 8B B0    L146  JSR L325          ;LOOK FOR VARIABLE
A9A8  85 49             STA FORPNT      ;SAVE IN FOR/NEXT VARIABLE POINTERS
A9AA  84 4A             STY FORPNT+1
A9AC  A9 B2             LDA #$B2        ;CHECK FOR '=' CHARACTER
A9AE  20 FF AE          JSR L242        ;CHECK ON CHARACTER
A9B1  A5 0E             LDA INTFLG      ;SAVE NUMERIC TYPE FLAG $80=INTEGER
A9B3  48                PHA             ;$00=FLOATING POINT
A9B4  A5 0D             LDA VALTYP      ;SAVE TYPE FLAG $FF= STRING
A9B6  48                PHA             ;$00=NUMERIC
A9B7  20 9E AD          JSR L256        ;EVALUATE EXPRESSION
A9BA  68                PLA
A9BB  2A                ROL A
A9BC  20 90 AD          JSR L312        ;CHECK TYPE
A9BF  D0 18             BNE L151        ;STRING VALUE ASSIGNMENT
A9C1  68                PLA
A9C2  10 12       L89   BPL L238         ;TRANSFER FAC TO VARIABLE
A9C4              ;
A9C4              ;INTEGER VALUE ASSIGNMENT
```

```
LOC   CODE        LINE

A9C4              ;
A9C4  20 1B BC    L516    JSR L51B        ;ROUND FAC#1
A9C7  20 BF B1            JSR L452        ;CHANGE TO INTEGER
A9CA  A0 00               LDY #$00
A9CC  A5 64               LDA FACHO+2     ;GET VALUE LSB
A9CE  91 49               STA (FORPNT),Y  ;AND STORE IN VARIABLE
A9D0  C8                  INY
A9D1  A5 65               LDA FACHO+3     ;GET VALUE MSB
A9D3  91 49               STA (FORPNT),Y  ;AND STORE IN VARIABLE +1
A9D5  60                  RTS
A9D6              ;
A9D6              ;FLOATING POINT VALUE ASSIGNMENT
A9D6              ;
A9D6  4C D0 BB    L238    JMP L607        ;TRANSFER FAC#1 TO VARIABLE
A9D9              ;
A9D9              ;ASSIGN VALUE TO STRING TI$
A9D9              ;
A9D9  68          L151    PLA
A9DA  A4 4A       L150    LDY FORPNT+1    ;GET VARIABLE POINTER MSB
A9DC  C0 BF               CPY #$BF        ;IS IT TI$
A9DE  D0 4C               BNE L166        ;NO
A9E0  20 A6 B6            JSR L451        ;THEN DISCARD UNWANTED STRING
A9E3  C9 06               CMP #$06        ;IS STRING 6 CHARACTERS LONG
A9E5  D0 3D               BNE L158        ;NO THEN 'ILLEGAL QUANTITY' ERROR
A9E7  A0 00               LDY #$00        ;CLEAR VARIABLE STORES
A9E9  84 61               STY FACEXP
A9EB  84 66               STY FACSGN
A9ED  84 71       L235    STY FBUFPT
A9EF  20 1D AA            JSR L161        ;ADD ASCII DIGIT TO FAC #1
A9F2  20 E2 BA            JSR L502        ;MULTIPLY FAC #1 BY 10
A9F5  E6 71               INC FBUFPT      ;INCREMENT POINTER TO NEXT DIGIT
A9F7  A4 71               LDY FBUFPT
A9F9  20 1D AA            JSR L161        ;ADD ASCII DIGIT TO FAC #1
A9FC  20 0C BC            JSR L516        ;COPY FAC #1 TO FAC #2
A9FF  AA                  TAX
AA00  F0 05               BEQ L163        ;IS IT ZERO
AA02  E8                  INX
AA03  8A                  TXA
AA04  20 ED BA            JSR L159        ;FAC#1 = FAC#1+FAC#2
AA07  A4 71       L163    LDY FBUFPT      ;INCREMENT DIGIT COUNTER
AA09  C8                  INY
AA0A  C0 06               CPY #$06        ;IS IT 6 DIGITS?
AA0C  D0 DF               BNE L235        ;NO THEN DO NEXT DIGIT
AA0E  20 E2 BA            JSR L502        ;FAC #1 = FAC #1 * 10
AA11  20 9B BC            JSR L526        ;CONVERT FLOATING TO FIXED
AA14  A6 64               LDX FACHO+2     ;PUT TIME IN VARIABLES
AA16  A4 63               LDY FACHO+1
AA18  A5 65               LDA FACHO+3
AA1A  4C DB FF            JMP L627        ;SET TIME
AA1D              ;
AA1D              ;ADD ASCII DIGIT TO FAC #1
AA1D              ;
AA1D  B1 22       L161    LDA (INDEX),Y   ;GET DIGIT
AA1F  20 80 00            JSR CHRGOT+7    ;CHARGET CHARACTER CHECK
AA22  90 03               BCC L157
AA24  4C 48 B2    L158    JMP L363        ;'ILLEGAL QUANTITY' ERROR
AA27  E9 2F       L157    SBC #$2F        ;CHANGE FROM ASCII TO HEX
AA29  4C 7E BD            JMP L553        ;GET ASCII DIGIT INTO FAC #1
AA2C              ;
AA2C              ;ASSIGN STRING VARIABLE
AA2C              ;
```

```
LOC   CODE        LINE

AA2C  A0 02       L166   LDY #$02         ;SET STRING ADDRESS POINTER
AA2E  B1 64              LDA (FACHO+2),Y ;GET STRING ADDRESS HI
AA30  C5 34              CMP FRETOP+1    ;COMPARE WITH STRING START
AA32  90 17              BCC L170        ;IF LESS THEN STRING IN PROGRAM
AA34  D0 07              BNE L155        ;CHECK FOR START OF VARIABLES
AA36  88                 DEY
AA37  B1 64              LDA (FACHO+2),Y ;GET STRING ADDRESS LO
AA39  C5 33              CMP FRETOP      ;COMPARE WITH STRING START LO
AA3B  90 0E              BCC L170        ;STRING IN PROGRAM
AA3D  A4 65       L155   LDY FACHO+3      ;CHECK FOR START OF VARIABLES
AA3F  C4 2E              CPY VARTAB+1
AA41  90 08              BCC L170        ;IF SMALLER, THEN IN PROGRAM
AA43  D0 0D              BNE L169        ;OTHERWISE CHECK STRING LENGTH
AA45  A5 64              LDA FACHO+2     ;START OF VARS LSB
AA47  C5 2D              CMP VARTAB
AA49  B0 07              BCS L169        ;IF >= THEN CHECK STRING LENGTH
AA4B  A5 64       L170   LDA FACHO+2
AA4D  A4 65              LDY FACHO+3
AA4F  4C 68 AA           JMP L171        ;SAVE TO VARIABLES
AA52  A0 00       L169   LDY #$00
AA54  B1 64              LDA (FACHO+2),Y
AA56  20 75 B4           JSR L304        ;CALCULATE STRING VECTOR
AA59  A5 50              LDA BASTMP+5
AA5B  A4 51              LDY BASTMP+6
AA5D  85 6F              STA ARISGN
AA5F  84 70              STY FACOV
AA61  20 7A B6           JSR L430        ;BUILD STRING INTO MEMORY
AA64  A9 61              LDA #$61
AA66  A0 00              LDY #$00
AA68  85 50       L171   STA BASTMP+5
AA6A  84 51              STY BASTMP+6
AA6C  20 DB B6           JSR L436        ;CLEAN DESCRIPTOR STACK
AA6F  A0 00              LDY #$00
AA71  B1 50              LDA (BASTMP+5),Y ;LENGTH
AA73  91 49              STA (FORPNT),Y ;TO VARIABLE
AA75  C8                 INY             ;POINT TO NEXT
AA76  B1 50              LDA (BASTMP+5),Y ;LOW ADDRESS
AA78  91 49              STA (FORPNT),Y ;TO VARIABLE
AA7A  C8                 INY             ;POINT TO NEXT
AA7B  B1 50              LDA (BASTMP+5),Y ;HIGH ADDRESS
AA7D  91 49              STA (FORPNT),Y ;TO VARIABLE
AA7F  60                 RTS
AA80                  ;***************************************
AA80                  ;*PERFORM 'PRINT#' COMMAND.
AA80                  ;*THIS CONSISTS OF JUST A SUBROUTINE CALL
AA80                  ;*TO $AA86 TO PERFORM THE 'CMD' OPERATION
AA80                  ;*AND A JUMP TO $ABB5 THE END OF THE 'INPUT#'
AA80                  ;*ROUTINE, THIS RESTORES THE DEFAULT I/O
AA80                  ;*AND SETS LOCATION $13 TO ZERO.
AA80                  ;***************************************
AA80  20 86 AA    PRINTH JSR L172         ;DO CMD COMMAND
AA83  4C B5 AB           JMP L209        ;TO INPUT# ROUTINE
AA86                  ;***************************************
AA86                  ;*PERFORM 'CMD' COMMAND.
AA86                  ;*THE PARAMETER FOLLOWING THE 'CMD' COMMAND
AA86                  ;*IS EVALUATED BY THE ROUTINE AT $B79E WHICH
AA86                  ;*GETS A SINGLE BYTE PARAMETER, THE RESULT
AA86                  ;*IS STORED IN THE X. INDEX REGISTER. THE
AA86                  ;*OUTPUT DEVICE IS THEN SET USING THIS VALUE
AA86                  ;*BY THE ROUTINE WITH VECTOR ADDRESS $FFC9
AA86                  ;*AND 'PRINT' PERFORMED.
```

```
LOC   CODE        LINE

AA86              ;*****************************************
AA86  20 9E B7    L172    JSR L195         ;INPUT BYTE PARAMETER
AA89  F0 05               BEQ L176
AA8B  A9 2C               LDA #$2C         ;CHARACTER VALUE FOR COMMA
AA8D  20 FF AE            JSR L242         ;CHECK IF COMMA
AA90  08          L176    PHP
AA91  86 13               STX INPRMT       ;SET CURRENT I/O DEVICE NUMBER
AA93  20 18 E1            JSR L44          ;SET OUTPUT DEVICE
AA96  28                  PLP
AA97  4C A0 AA            JMP L190         ;PERFORM 'PRINT' COMMAND
AA9A              ;*****************************************
AA9A              ;*PERFORM 'PRINT' COMMAND.
AA9A              ;*THERE ARE FOUR DIFFERENT ROUTES WHICH CAN
AA9A              ;*BE TAKEN BY THE 'PRINT' ROUTINE AND THESE
AA9A              ;*DEPEND ON THE CHARACTER OR COMMAND FOLLOWING
AA9A              ;*THE PRINT COMMAND, THESE CAN BE ONE OF -
AA9A              ;*'TAB(', 'SPC(', COMMA, OR SEMICOLON.
AA9A              ;*INTERESTING SUBROUTINES WITHIN THE MAIN
AA9A              ;*'PRINT' ROUTINE ARE:
AA9A              ;*   $AAA4 TEST FOR 'TAB(', BRANCH IF FOUND
AA9A              ;*   $AAA8 TEST FOR 'SPC(', BRANCH IF FOUND
AA9A              ;*   $AAAD TEST FOR COMMA, BRANCH IF FOUND
AA9A              ;*   $AAB1 TEST FOR SEMICOLON, BRANCH IF FOUND
AA9A              ;*   $AABC PRINT NUMERAL AFTER CONVERTING TO ASCII
AA9A              ;*   $AAD7 PRINT CR OR CRLF
AA9A              ;*   $AA9A PRINT STRING
AA9A              ;*IT SHOULD BE NOTED THAT THE OUTPUT DEVICE
AA9A              ;*NUMBER IS STORED IN LOCATION $13. ON
AA9A              ;*COMPLETION THE BUFFER IS RESET AND LOCATION
AA9A              ;*$0200 IS SET TO $00, .X IS $FF AND .Y IS $01.
AA9A              ;*****************************************
AA9A  20 21 AB    L178    JSR L22          ;PRINT STRING FROM Y.A.
AA9D  20 79 00    L186    JSR CHRGOT       ;GET LAST CHARACTER
AAA0  F0 35       L190    BEQ L48          ;IF ZERO DO CARRIAGE RETURN
AAA2  F0 43       L180    BEQ L191         ;EXIT TO SYNTAX ERROR
AAA4  C9 A3               CMP #$A3         ;CHECK FOR 'TAB(' CODE
AAA6  F0 50               BEQ L193         ;DO TAB
AAA8  C9 A6               CMP #$A6         ;CHECK FOR 'SPC(' CODE
AAAA  18                  CLC              ;SET SPC FLAG
AAAB  F0 4B               BEQ L193         ;DO SPC
AAAD  C9 2C               CMP #$2C         ;CHECK FOR ','
AAAF  F0 37               BEQ L182         ;OUTPUT AND GET NEXT CHARACTER
AAB1  C9 3B               CMP #$3B         ;CHECK FOR ';'
AAB3  F0 5E               BEQ L200         ;GET NEXT CHARACTER AND CONTINUE
AAB5  20 9E AD            JSR L256         ;EVALUATE EXPRESSION
AAB8  24 0D               BIT VALTYP       ;FLAG
AABA  30 DE               BMI L178         ;PRINT STRING FROM Y.A.
AABC  20 DD BD            JSR L556         ;CHANGE FAC#1 TO ASCII STRING
AABF  20 87 B4            JSR L440         ;SET UP STRING PARAMETERS
AAC2  20 21 AB            JSR L22          ;POINT TO STRING
AAC5  20 3B AB            JSR L201         ;OUTPUT SPACE CHARACTER
AAC8  D0 D3               BNE L186         ;IF NOT END CONTINUE
AACA  A9 00       L199    LDA #$00         ;SET INPUT BUFFER TO
AACC  9D 00 02            STA BUF,X        ;TERMINATE WITH $00
AACF  A2 FF               LDX #$FF         ;SET X. TO $FF
AAD1  A0 01               LDY #$01         ;SET Y. TO $01
AAD3  A5 13               LDA INPRMT       ;GET OUTPUT DEVICE NUMBER
AAD5  D0 10               BNE L191         ;IF NOT ZERO THEN EXIT
AAD7  A9 0D       L48     LDA #$0D         ;CARRIAGE RETURN CHARACTER
AAD9  20 47 AB            JSR L18          ;OUTPUT
AADC  24 13               BIT INPRMT       ;TEST LOGICAL FILE #
```

```
LOC   CODE        LINE

AADE  10 05             BPL L17         ;IF GREATER THAN 128 NO LINE FEED
AAE0  A9 0A             LDA #$0A        ;CHARACTER FOR LINE FEED
AAE2  20 47 AB          JSR L18         ;OUTPUT
AAE5  49 FF       L17   EOR #$FF
AAE7  60          L191  RTS              ;EXIT
AAE8              ;
AAE8              ;DECIMAL TABULATOR
AAE8              ;
AAE8  38          L182  SEC              ;SET FLAG FOR GET CURSOR
AAE9  20 F0 FF          JSR L808        ;GET CURSOR POSITION
AAEC  98                TYA             ;PUT COLUMN # IN .A
AAED  38                SEC
AAEE  E9 0A       L184  SBC #$0A         ;SUBTRACT 10
AAF0  B0 FC             BCS L184
AAF2  49 FF             EOR #$FF        ;CHECK NOT NEGATIVE
AAF4  69 01             ADC #$01        ;ADD ONE
AAF6  D0 16             BNE L183        ;JUMP INTO TAB
AAF8              ;****************************************
AAF8              ;*PERFORM 'TAB(' OR 'SPC(' COMMAND.
AAF8              ;*WHICH COMMAND IS PERFORMED BY THIS ROUTINE
AAF8              ;*DEPENDS ON THE STATE OF THE CARRY FLAG,
AAF8              ;*IF CARRY =1 THEN A 'TAB' AND IF CARRY
AAF8              ;=0 THEN A 'SPC' COMMAND IS PERFORMED. THE
AAF8              ;*ROUTINE GETS THE CURRENT CURSOR POSITION
AAF8              ;*WHICH IT PLACES IN LOCATION $09 AND THEN
AAF8              ;*COMPARES IT WITH THE BYTE VALUE FOLLOWING
AAF8              ;*THE COMMAND. IF THE VALUE IS LESS THAN THE
AAF8              ;*POSITION THEN THE ROUTINE GETS THE NEXT
AAF8              ;*CHARACTER AND EXITS. IF LARGER THEN THE
AAF8              ;*ROUTINE FILLS IN THE REQUIRED NUMBER OF
AAF8              ;*SPACES. THE SPC COMMAND VARIES FROM THE
AAF8              ;*TAB IN THAT IT DOES NOT DO A COMPARISON
AAF8              ;*AND SUBTRACTION FROM CURRENT CURSOR
AAF8              ;*POSITION.
AAF8              ;****************************************
AAF8  08          L193  PHP              ;SAVE FLAG
AAF9  38                SEC
AAFA  20 F0 FF          JSR L808        ;GET CURSOR POSITION
AAFD  84 09             STY TRMPOS      ;STORE IN TAB COLUMN POINTER
AAFF  20 9B B7          JSR L446        ;GET BYTE VALUE
AB02  C9 29             CMP #$29        ;CHECK FOR CLOSING BRACKET
AB04  D0 59             BNE L206        ;IF NOT THEN SYNTAX ERROR
AB06  28                PLP             ;RETRIEVE FLAG
AB07  90 06             BCC L194        ;JUMP FOR 'SPC('
AB09  8A                TXA             ;PUT TAB VALUE IN ACCUMULATOR
AB0A  E5 09             SBC TRMPOS      ;SUBTRACT FROM CURSOR POSITION
AB0C  90 05             BCC L200        ;IF LESS THEN EXIT
AB0E  AA          L183  TAX              ;PUT DIFFERENCE IN .X
AB0F  E8          L194  INX
AB10  CA          L197  DEX              ;DECREMENT .X FOR SPACE OUTPUT LOOP
AB11  D0 06             BNE L185        ;IF .X NOT ZERO THEN ANOTHER SPACE
AB13  20 73 00    L200  JSR CHRGET       ;GET NEXT CHARACTER
AB16  4C A2 AA          JMP L180        ;AND RETURN TO 'PRINT' ROUTINE
AB19  20 3B AB    L185  JSR L201         ;OUTPUT SPACE CHARACTER
AB1C  D0 F2             BNE L197        ;DO AGAIN
AB1E              ;****************************************
AB1E              ;*PRINT STRING FROM MEMORY.
AB1E              ;*THE STARTING ADDRESS OF THE STRING TO BE
AB1E              ;*PRINTED IS STORED IN THE ACCUMULATOR (LOW
AB1E              ;*ORDER BYTE) AND THE Y INDEX REGISTER (HIGH
AB1E              ;*ORDER BYTE) PRIOR TO ENTRY OF THIS ROUTINE.
```

```
LOC   CODE        LINE

AB1E              ;*CONSECUTIVE CHARACTERS ARE THEN PRINTED
AB1E              ;*THE OUTPUT DEVICE UNTIL A ZERO TERMINATOR
AB1E              ;*BYTE IS ENCOUNTERED.
AB1E              ;**************************************
AB1E  20 87 B4    L198   JSR L440        ;SET UP STRING PARAMETERS
AB21  20 A6 B6    L22    JSR L451        ;DISCARD UNWANTED STRINGS
AB24  AA                 TAX             ;SET STRING LENGTH
AB25  A0 00              LDY #$00        ;INITIALISE STRING POINTER
AB27  E8                 INX
AB28  CA          L181   DEX             ;DECREMENT LOOP COUNTER
AB29  F0 BC              BEQ L191        ;END OF STRING THEN EXIT
AB2B  B1 22              LDA (INDEX),Y   ;GET STRING CHARACTER
AB2D  20 47 AB           JSR L18         ;OUTPUT A CHARACTER
AB30  C8                 INY             ;BUMP POINTER
AB31  C9 0D              CMP #$0D        ;CHECK FOR CARRIAGE RETURN
AB33  D0 F3              BNE L181        ;NO THEN DO NEXT CHARACTER
AB35  20 E5 AA           JSR L17         ;INVERT CODE WITH 'EOR #$FF'
AB38  4C 28 AB           JMP L181        ;AND CONTINUE WITH NEXT CHARACTER
AB3B              ;**************************************
AB3B              ;*PRINT SINGLE FORMAT CHARACTER.
AB3B              ;*THE FORMAT CHARACTER IS EITHER A CURSOR
AB3B              ;*RIGHT OR A SPACE DEPENDING ON THE CONTENTS
AB3B              ;*OF LOCATION $13, IF ZERO THEN CURSOR RIGHT
AB3B              ;*OTHERWISE SPACE.
AB3B              ;**************************************
AB3B  A5 13       L201   LDA INPRMT      ;GET CHARACTER TYPE FLAG
AB3D  F0 03              BEQ L189        ;BRANCH IF CURSOR RIGHT
AB3F  A9 20              LDA #$20        ;SET SPACE CHARACTER
AB41  2C                 .BYT $2C        ;SKIP NEXT
AB42  A9 1D       L189   LDA #$1D        ;SET CURSOR RIGHT CHARACTER
AB44  2C                 .BYT $2C        ;SKIP NEXT
AB45  A9 3F       L202   LDA #$3F        ;SET '?' CHARACTER
AB47  20 0C E1    L18    JSR L619        ;OUTPUT CHARACTER
AB4A  29 FF              AND #$FF        ;SET FLAG
AB4C  60                 RTS
AB4D              ;**************************************
AB4D              ;*HANDLE BAD INPUT DATA.
AB4D              ;*ERROR MESSAGES FOR 'INPUT', 'GET' AND 'READ'
AB4D              ;*ARE GENERATED BY THIS ROUTINE. ON ENTRY
AB4D              ;*LOCATION $11 INDICATES THE COMMAND CALLING
AB4D              ;*THIS ROUTINE, $00 = 'INPUT', $40='GET'
AB4D              ;*AND $98='READ'. 'INPUT' CAN GENERATE ONE OF
AB4D              ;*TWO ERRORS, IF A FILE IS OPEN THEN A 'FILE
AB4D              ;*DATA' ERROR IS GENERATED, IF FILES ARE
AB4D              ;*CLOSED THEN '?REDO FROM START' IS PRINTED
AB4D              ;*ON THE OUTPUT DEVICE AND THE CHARGET
AB4D              ;*POINTERS $7A-$7B ARE LOADED WITH THE START
AB4D              ;*ADDRESS OF THE PREVIOUS LINE.
AB4D              ;**************************************
AB4D  A5 11       L19    LDA INPFLG      ;GET MESSAGE TYPE FLAG
AB4F  F0 11              BEQ L196        ;DO 'INPUT'
AB51  30 04              BMI L240        ;DO 'READ'
AB53  A0 FF              LDY #$FF
AB55  D0 04              BNE L205        ;DO 'GET'
AB57              ;
AB57              ;'READ' ERROR HANDLER
AB57              ;
AB57  A5 3F       L240   LDA DATLIN      ;GET CURRENT DATA LINE # LSB
AB59  A4 40              LDY DATLIN+1    ;MSB
AB5B              ;
AB5B              ;'GET' ERROR HANDLER
```

```
LOC   CODE        LINE


AB5B              ;
AB5B  85 39       L205    STA CURLIN      ;GET CURRENT BASIC LINE # LSB
AB5D  84 3A               STY CURLIN+1    ;MSB
AB5F  4C 08 AF    L206    JMP L94         ;DO SYNTAX ERROR
AB62              ;
AB62              ;'INPUT' ERROR HANDLER
AB62              ;
AB62  A5 13       L196    LDA INPRMT      ;GET INPUT DEVICE #
AB64  F0 05               BEQ L204        ;IF KEYBOARD THEN JUMP
AB66  A2 18               LDX #$18        ;SET ERROR MESSAGE #
AB68  4C 37 A4            JMP L10         ;OUTPUT ERROR MESSAGE
AB6B  A9 0C       L204    LDA #$0C        ;SET POINTERS TO MESSAGE
AB6D  A0 AD               LDY #$AD        ;'REDO FROM START'
AB6F  20 1E AB            JSR L198        ;OUTPUT STRING
AB72  A5 3D               LDA OLDTXT      ;GET PROGRAM POINTERS LSB
AB74  A4 3E               LDY OLDTXT+1    ;MSB
AB76  85 7A               STA TXTPTR      ;SAVE IN CHARGET POINTERS LSB
AB78  84 7B               STY TXTPTR+1    ;MSB
AB7A  60                  RTS
AB7B                      .END
AB7B                      .LIB B5
AB7B              ;***************************************
AB7B              ;*PERFORM 'GET' COMMAND.
AB7B              ;*CHECKS ARE FIRST MADE BY THE ROUTINE TO
AB7B              ;*DETERMINE THE PROGRAM MODE, DIRECT OR
AB7B              ;*PROGRAM, AND WHETHER THE COMMAND IS 'GET'
AB7B              ;*OR 'GET#'. IF THE COMMAND IS 'GET#' THEN
AB7B              ;*IT INPUTS THE FILE NUMBER, CHECKS THAT A
AB7B              ;*COMMA IS PRESENT AND SETS THE REQUIRED
AB7B              ;*DEVICE FOR INPUT. THE INPUT BUFFER $0200
AB7B              ;*IS SET UP TO ACCEPT JUST A SINGLE CHARACTER,
AB7B              ;*THE BUFFER BEING FILLED WITH A NULL BYTE.
AB7B              ;*THE ACCUMULATOR IS LOADED WITH $40 AND
AB7B              ;*THE PROGRAM JUMPS TO THE GET CHARACTER
AB7B              ;*FROM INPUT DEVICE SUBROUTINE WITHIN THE
AB7B              ;*PERFORM 'READ' ROUTINE, THE ENTRY ADDRESS
AB7B              ;*IS $AC0F. THIS ROUTINE FIRST STORES THE
AB7B              ;*ACCUMULATOR IN LOCATION $11 TO IDENTIFY
AB7B              ;*THAT IT IS A GET COMMAND. THE CHARACTER IS
AB7B              ;*THEN OBTAINED FROM THE INPUT DEVICE, THE
AB7B              ;*INPUT CHARACTER BEING STORED IN $0200.
AB7B              ;***************************************
AB7B  20 A6 B3    GET     JSR L450        ;CHECK ON PROGRAM MODE
AB7E  C9 23               CMP #$23        ;CHECK FOR '#' SIGN
AB80  D0 10               BNE L207        ;IF NOT 'GET#' THEN JUMP
AB82  20 73 00            JSR CHRGET      ;GET NEXT CHARACTER
AB85  20 9E B7            JSR L195        ;GET BYTE VALUE
AB88  A9 2C               LDA #$2C        ;IS IT A COMMA
AB8A  20 FF AE            JSR L242        ;CHECK ON CHARACTER
AB8D  86 13               STX INPRMT      ;SET INPUT DEVICE FLAG
AB8F  20 1E E1            JSR L179        ;GETS A CHARACTER
AB92  A2 01       L207    LDX #$01        ;SET POINTERS FOR GET
AB94  A0 02               LDY #$02
AB96  A9 00               LDA #$00
AB98  8D 01 02            STA BUF+1       ; SET TO ONE BYTE BUFFER INPUT
AB9B  A9 40               LDA #$40        ;SET 'GET' FLAG
AB9D  20 0F AC            JSR L219        ;INPUT CHARACTER
ABA0  A6 13               LDX INPRMT      ;GET INPUT DEVICE NUMBER
ABA2  D0 13               BNE L177        ;CLOSE I/O CHANNELS
ABA4  60                  RTS
ABA5              ;***************************************
```

```
LOC   CODE        LINE

ABA5              ;*PERFORM 'INPUT#' COMMAND.
ABA5              ;*THIS GETS THE FILE NUMBER AND CHECKS FOR
ABA5              ;*A FOLLOWING COMMA, SETS THE INPUT DEVICE
ABA5              ;*AND JUMPS TO THE INPUT ROUTINE. HAVING
ABA5              ;*PERFORMED THE INPUT THE DEVICE IS TURNED
ABA5              ;*OFF AND LOCATION $13 IS SET TO ZERO.
ABA5              ;****************************************
ABA5  20 9E B7    INPUTH JSR L195         ;INPUT BYTE PARAMETER
ABA8  A9 2C              LDA #$2C         ;LOAD COMMA CHARACTER INTO .A
ABAA  20 FF AE           JSR L242         ;ROUTINE TO CHECK FOR COMMA
ABAD  86 13              STX INPRMT       ;SET INPUT DEVICE
ABAF  20 1E E1           JSR L179         ;GET A CHARACTER
ABB2  20 CE AB           JSR L212         ;PERFORM INPUT FUNCTION
ABB5  A5 13       L209   LDA INPRMT        ;SET INPUT DEVICE
ABB7  20 CC FF    L177   JSR L674         ;CLEAR I/O CHANNELS
ABBA  A2 00              LDX #$00
ABBC  86 13              STX INPRMT       ;SET INPUT DEVICE TO KEYBOARD
ABBE  60                 RTS
ABBF              ;****************************************
ABBF              ;*PERFORM 'INPUT' COMMAND.
ABBF              ;*THIS FIRST CHECKS FOR A QUOTATION MARK,
ABBF              ;*$22, AS THE NEXT CHARACTER FOLLOWING THE
ABBF              ;*'INPUT' COMMAND. IF ONE IS PRESENT THEN THE
ABBF              ;*STRING WITHIN THE QUOTATION MARKS IS PRINTED
ABBF              ;*ON THE OUTPUT DEVICE. THE INPUT BUFFER AT
ABBF              ;*$0200 IS SET UP TO ACCEPT UP TO 80 CHARACTERS
ABBF              ;*THE STATUS ST IS THEN TESTED (DERIVED FROM
ABBF              ;*THE VALUE IN $13), AND THE ROUTINE BRANCHES
ABBF              ;*TO THE INPUT LINE ROUTINE.
ABBF              ;****************************************
ABBF  C9 22       INPUT  CMP #$22          ;ARE THERE QUOTES
ABC1  D0 0B              BNE L212         ;NO
ABC3  20 BD AE           JSR L283         ;GET STRING IN PROGRAM
ABC6  A9 3B              LDA #$3B         ;SEMICOLON CHARACTER
ABC8  20 FF AE           JSR L242         ;CHECK FOR SEMICOLON
ABCB  20 21 AB           JSR L22          ;OUTPUT STRING
ABCE  20 A6 B3    L212   JSR L450         ;CHECK FOR DIRECT MODE
ABD1  A9 2C              LDA #$2C         ;COMMA CHARACTER
ABD3  8D FF 01           STA BUF-1        ;PUT IN BUFFER START
ABD6  20 F9 AB    L213   JSR L216         ;PROMPT FOR INPUT
ABD9  A5 13              LDA INPRMT       ;GET OUTPUT DEVICE
ABDB  F0 0D              BEQ L220         ;JUMP IF KEYBOARD
ABDD  20 B7 FF           JSR L669         ;GET STATUS
ABE0  29 02              AND #$02         ;CHECK FOR INPUT TIME OUT
ABE2  F0 06              BEQ L220         ;YES
ABE4  20 B5 AB           JSR L209         ;CLEAR CHANNELS AND RESET KEYBOARD
ABE7  4C F8 A8           JMP L130         ;OFFSET TO NEXT STATEMENT
ABEA  AD 00 02    L220   LDA BUF          ;GET FIRST CHARACTER OF BUFFER
ABED  D0 1E              BNE L221         ;DO GET CHARACTER
ABEF  A5 13              LDA INPRMT       ;GET DEVICE NUMBER
ABF1  D0 E3              BNE L213         ;IF NOT KEYBOARD THEN JUMP
ABF3  20 06 A9           JSR L131         ;SCAN FOR NEXT STATEMENT
ABF6  4C FB A8           JMP L218         ;OFFSET TO NEXT STATEMENT
ABF9  A5 13       L216   LDA INPRMT       ;GET DEVICE NUMBER
ABFB  D0 06              BNE L215         ;IF NOT KEYBOARD
ABFD  20 45 AB           JSR L202         ;OUTPUT '?' CHARACTER
AC00  20 3B AB           JSR L201         ;PRINT SPACE CHARACTER
AC03  4C 60 A5    L215   JMP L41          ;RECEIVE LINE INPUT
AC06              ;****************************************
AC06              ;*PERFORM 'READ' COMMAND.
AC06              ;*THIS ROUTINE IS SHARED BY BOTH 'GET' AND
```

```
LOC   CODE        LINE

AC06              ;*'INPUT', THE THREE DIFFERENT FUNCTIONS ARE
AC06              ;*DISTINGUISHED BY THE CONTENTS OF $11. THESE
AC06              ;*VALUES ARE:
AC06              ;* 'GET'       -  $40
AC06              ;* 'INPUT'     -  $00
AC06              ;* 'READ'      -  $98
AC06              ;*THESE ROUTINES SCAN THE INPUT BUFFER OF DATA
AC06              ;*STATEMENTS FOR BLOCKS OF DATA. IN THE CASE
AC06              ;*OF 'GET' A BLOCK OF DATA IS DEFINED AS A
AC06              ;*SINGLE CHARACTER. FOR 'INPUT' A TERMINATING
AC06              ;*CARRIAGE RETURN DEFINES THE INPUT BLOCK,
AC06              ;*AND WITH 'READ' THE SEPARATING COMMA OR
AC06              ;*END OF LINE MARKER FOR THE DATA STATEMENT
AC06              ;*POINTED TO BY $41-$42 DEFINES THE DATA
AC06              ;*BLOCK. THE BLOCK OF DATA FROM WHICHEVER
AC06              ;*SOURCE IS THEN ASSIGNED TO THE VARIABLE IN
AC06              ;*THE COMMAND. OF THE ENTRY POINTS WITHIN
AC06              ;*THIS ROUTINE THE FOLLOWING ARE INTERESTING:
AC06              ;*  $AC0D - 'INPUT' ENTRY POINT
AC06              ;*  $AC0F - 'GET' ENTRY POINT
AC06              ;*  $AC71 - ASSIGN STRING TO STRING VARIABLE
AC06              ;*  $AC89 - ASSIGN NUMERAL TO NUMERIC VARIABLE
AC06              ;*  $ACB8 - USED BY READ TO SCAN FOR DATA
AC06              ;*          STATEMENTS
AC06              ;*  $ACDF - CHECKS FOR TERMINATING ZERO AT END
AC06              ;*          OF BUFFER IF NOT FOUND PRINTS
AC06              ;*          'EXTRA IGNORED' UNLESS THERE IS AN
AC06              ;*          ACTIVE FILE IN WHICH CASE NO WARNING
AC06              ;*          IS GIVEN
AC06              ;****************************************
AC06  A6 41       READ   LDX DATPTR      ;DATA STATEMENT POINTER
AC08  A4 42              LDY DATPTR+1
AC0A  A9 98              LDA #$98       ;SET 'READ' FLAG
AC0C  2C                 .BYT $2C
AC0D              ;
AC0D              ;'INPUT' ENTRY POINT
AC0D              ;
AC0D  A9 00       L221   LDA #$00        ;SET 'INPUT' FLAG
AC0F              ;
AC0F              ;'GET' ENTRY POINT
AC0F              ;
AC0F  85 11       L219   STA INPFLG      ;STORE COMMAND FLAG
AC11  86 43              STX INPPTR     ;SET INPUT VECTORS TO DATA
AC13  84 44              STY INPPTR+1   ;ADDRESS POINTERS
AC15  20 8B B0    L211   JSR L325        ;LOCATE VARIABLE
AC18  85 49              STA FORPNT     ;SAVE LOCATION IN VARIABLE
AC1A  84 4A              STY FORPNT+1   ;POINTERS
AC1C  A5 7A              LDA TXTPTR     ;GET PROGRAM POINTERS
AC1E  A4 7B              LDY TXTPTR+1
AC20  85 4B              STA BASTMP     ;AND SAVE
AC22  84 4C              STY BASTMP+1
AC24  A6 43              LDX INPPTR     ;GET DATA ADDRESS POINTERS AND
AC26  A4 44              LDY INPPTR+1
AC28  86 7A              STX TXTPTR     ;SAVE IN CHARGET PROGRAM POINTERS
AC2A  84 7B              STY TXTPTR+1
AC2C  20 79 00           JSR CHRGOT     ;GET LAST CHARACTER
AC2F  D0 20              BNE L225       ;IF NOT LINE TERMINATOR ZERO THEN
AC31  24 11              BIT INPFLG     ;IS IT 'READ' OR 'GET'
AC33  50 0C              BVC L243       ;YES 'GET' THEN JUMP
AC35  20 24 E1           JSR L210       ;GET A CHARACTER
AC38  8D 00 02           STA BUF        ;AND SAVE IN BUFFER
```

```
LOC   CODE        LINE

AC3B  A2 FF             LDX #$FF        ;SET BUFFER POINTERS
AC3D  A0 01             LDY #$01
AC3F  D0 0C             BNE L227        ;IF NOT TERMINATOR THEN
AC41  30 75       L243  BMI L239         ;SCAN FOR DATA STATEMENT
AC43  A5 13             LDA INPRMT      ;GET DEVICE NUMBER
AC45  D0 03             BNE L223        ;NOT KEYBOARD
AC47  20 45 AB          JSR L202        ;OUTPUT '?'
AC4A  20 F9 AB    L223  JSR L216         ;OUTPUT SECOND '?'
AC4D  86 7A       L227  STX TXTPTR       ;SET CHARGET PROGRAM POINTERS
AC4F  84 7B             STY TXTPTR+1    ;TO BUFFER START
AC51  20 73 00    L225  JSR CHRGET       ;GET NEXT CHARACTER
AC54  24 0D             BIT VALTYP      ;TEST FOR STRING TYPE
AC56  10 31             BPL L232        ;ASSIGN NUMERAL TO NUMERIC VARIABLE
AC58  24 11             BIT INPFLG      ;IS IT 'READ' OR 'GET'
AC5A  50 09             BVC L222        ;YES 'GET' THEN JUMP
AC5C  E8                INX
AC5D  86 7A             STX TXTPTR      ;INCREMENT CHARGET PROGRAM POINTER LSB
AC5F  A9 00             LDA #$00        ;CLEAR SEARCH CHARACTER
AC61  85 07             STA CHARAC      ;FLAG
AC63  F0 0C             BEQ L229        ;JUMP TO ASSIGN STRING VARIABLE ROUTINE
AC65  85 07       L222  STA CHARAC       ;STORE .A IN SEARCH CHARACTER FLAG
AC67  C9 22             CMP #$22        ;IS IT A QUOTES CHARACTER
AC69  F0 07             BEQ L230        ;YES THEN JUMP
AC6B  A9 3A             LDA #$3A        ;':' CHARACTER
AC6D  85 07             STA CHARAC      ;SAVE IN SEARCH CHARACTER FLAG
AC6F  A9 2C             LDA #$2C        ;SET .A TO ',' CHARACTER
AC71              ;
AC71              ;ASSIGN STRING TO STRING VARIABLE
AC71              ;
AC71  18          L229  CLC
AC72  85 08       L230  STA ENDCHR       ;STORE .A IN SCAN QUOTES FLAG
AC74  A5 7A             LDA TXTPTR      ;GET CHARGET PROGRAM POINTERS
AC76  A4 7B             LDY TXTPTR+1
AC78  69 00             ADC #$00        ;IF CARRY SET INCREMENT
AC7A  90 01             BCC L231        ;IF $7A >$FF THEN INCREMENT
AC7C  C8                INY             ;PROGRAM POINTER MSB IN .Y
AC7D  20 8D B4    L231  JSR L188         ;SET UP STRING
AC80  20 E2 B7          JSR L455        ;SET PROGRAM POINTER BACK
AC83  20 DA A9          JSR L150        ;ASSIGN VALUE TO STRING
AC86  4C 91 AC          JMP L228        ;AND CONTINUE
AC89              ;
AC89              ;ASSIGN NUMERAL TO NUMERIC VARIABLE
AC89              ;
AC89  20 F3 BC    L232  JSR L532         ;TRANSFER STRING TO FAC #1
AC8C  A5 0E             LDA INTFLG      ;GET NUMERIC TYPE
AC8E  20 C2 A9          JSR L89         ;BRING FAC#1 TO VARIABLE
AC91  20 79 00    L228  JSR CHRGOT       ;GET LAST CHARACTER
AC94  F0 07             BEQ L236        ;IF ZERO THEN END
AC96  C9 2C             CMP #$2C        ;COMPARE WITH ','
AC98  F0 03             BEQ L236        ;IF COMMA THEN JUMP
AC9A  4C 4D AB          JMP L19         ;OTHERWISE DO BAD INPUT ERROR
AC9D  A5 7A       L236  LDA TXTPTR       ;GET CHARGET PROGRAM POINTERS
AC9F  A4 7B             LDY TXTPTR+1
ACA1  85 43             STA INPPTR      ;STORE IN INPUT VECTOR
ACA3  84 44             STY INPPTR+1
ACA5  A5 4B             LDA BASTMP      ;GET OLD BASIC POINTERS
ACA7  A4 4C             LDY BASTMP+1
ACA9  85 7A             STA TXTPTR      ;AND STORE IN CHARGET POINTERS
ACAB  84 7B             STY TXTPTR+1
ACAD  20 79 00          JSR CHRGOT      ;GET LAST CHARACTER
ACB0  F0 2D             BEQ L244        ;AND CHECK FOR TERMINATING ZERO
```

```
LOC   CODE        LINE

ACB2  20 FD AE            JSR L296        ;CHECK FOR COMMA
ACB5  4C 15 AC            JMP L211        ;AND CONTINUE
ACB8                ;
ACB8                ;SCAN FOR DATA STATEMENTS
ACB8                ;
ACB8  20 06 A9      L239  JSR L131         ;SCAN FOR NEXT STATEMENT
ACBB  C8                  INY
ACBC  AA                  TAX
ACBD  D0 12               BNE L226        ;END OF LINE?
ACBF  A2 0D               LDX #$0D        ;SET ERROR MESSAGE FLAG
ACC1  C8                  INY
ACC2  B1 7A               LDA (TXTPTR),Y ;GET CHARACTER FROM DATA
ACC4  F0 6C               BEQ L249        ;IF 0 THEN 'OUT OF DATA' ERROR
ACC6  C8                  INY
ACC7  B1 7A               LDA (TXTPTR),Y ;GET NEXT CHARACTER FROM DATA
ACC9  85 3F               STA DATLIN      ;SAVE CURRENT DATA LINE # LSB
ACCB  C8                  INY
ACCC  B1 7A               LDA (TXTPTR),Y ;GET CURRENT DATA LINE# AND
ACCE  C8                  INY
ACCF  85 40               STA DATLIN+1    ;SAVE MSB
ACD1  20 FB A8      L226  JSR L218         ;SCAN TO END OF STATEMENT
ACD4  20 79 00            JSR CHRGOT      ;GET LAST CHARACTER
ACD7  AA                  TAX
ACD8  E0 83               CPX #$83        ;IS IT 'DATA' COMMAND TOKEN
ACDA  D0 DC               BNE L239        ;NO THEN LOOK MORE
ACDC  4C 51 AC            JMP L225        ;YES THEN READ DATA
ACDF                ;
ACDF                ;CHECK FOR TERMINATING ZERO
ACDF                ;
ACDF  A5 43         L244  LDA INPPTR       ;GET INPUT VECTOR POINTERS
ACE1  A4 44               LDY INPPTR+1
ACE3  A6 11               LDX INPFLG      ;COMMAND TYPE FLAG
ACE5  10 03               BPL L241        ;IF 'READ' OR 'GET' THEN BRANCH
ACE7  4C 27 A8            JMP L72         ;SAVE CURRENT DATA ADDRESS
ACEA  A0 00         L241  LDY #$00         ;CLEAR POINTER
ACEC  B1 43               LDA (INPPTR),Y ;GET INPUT
ACEE  F0 0B               BEQ L246        ;IF ZERO THEN EXIT
ACF0  A5 13               LDA INPRMT      ;GET I/O FLAG
ACF2  D0 07               BNE L246        ;IF NOT KEYBOARD THEN END
ACF4  A9 FC               LDA #$FC        ;SET UP ERROR MESSAGE POINTERS
ACF6  A0 AC               LDY #$AC        ;TO MESSAGE 'EXTRA IGNORED'
ACF8  4C 1E AB            JMP L198        ;OUTPUT MESSAGE
ACFB  60            L246  RTS
ACFC                ;**************************************
ACFC                ;*ERROR MESSAGE DATA
ACFC                ;**************************************
ACFC  3F 45               .BYT '?EXTRA IGNORED',CR,$00
AD0A  0D
AD0B  00
AD0C  3F 52               .BYT '?REDO FROM START',CR,$00
AD1C  0D
AD1D  00
AD1E                ;**************************************
AD1E                ;*PERFORM 'NEXT' COMMAND.
AD1E                ;*THE FIRST FUNCTION OF THIS ROUTINE IS TO
AD1E                ;*CHECK FOR ANY VARIABLE NAME FOLLOWING THE
AD1E                ;*'NEXT' COMMAND, IF THERE IS NONE THEN
AD1E                ;*LOCATIONS $49-$4A ARE SET TO ZERO. IF A
AD1E                ;*VARIABLE NAME FOLLOWS TH 'NEXT' COMMAND THEN
AD1E                ;*ITS LOCATION IS OBTAINED USING THE ROUTINE
AD1E                ;*AT $B08B, THIS RETURNS THE POINTERS IN THE
```

```
LOC   CODE        LINE

AD1E              ;*ACCUMULATOR (LOW ORDER ADDRESS BYTE) AND THE
AD1E              ;*Y INDEX REGISTER (HIGH ORDER ADDRESS BYTE).
AD1E              ;*THESE VALUES ARE STORED IN THE VARIABLE
AD1E              ;*POINTER $49-$4A. THE STACK IS THEN SEARCHED
AD1E              ;*FOR A MATCHING 'FOR' COMMAND, IF NO VARIABLE
AD1E              ;*IS SPECIFIED THEN THE LAST ENTERED 'FOR'
AD1E              ;*RETURN DATA IS USED, IF THERE IS NO MATCHING
AD1E              ;*RETURN 'FOR' THEN A '?NEXT WITHOUT FOR
AD1E              ;*ERROR' IS GENERATED. THE STEP VALUE IN
AD1E              ;*FLOATING POINT IS MOVED FROM THE STACK TO
AD1E              ;*FLOATING ACCUMULATOR #1 AND ADDED TO THE
AD1E              ;*VARIABLE POINTED TO BY $49-$4A. THIS IS
AD1E              ;*COMPARED WITH THE 'TO' VALUE STORED IN THE
AD1E              ;*STACK AND IF EQUAL EXITS FROM THE 'FOR -
AD1E              ;*NEXT' LOOP. IF NOT EQUAL THEN THE RETURN
AD1E              ;*LINE NUMBER IS RESTORED IN $39-$3A AND THE
AD1E              ;*CHARGET POINTERS $7A-$7B RESET FOR THE
AD1E              ;*'FOR' ENTRY POINT AND A WARM START TO BASIC
AD1E              ;*INITIATED.
AD1E              ;****************************************
AD1E  D0 04       NEXT   BNE L247        ;DOES VARIABLE NAME FOLLOW
AD20  A0 00              LDY #$00
AD22  F0 03              BEQ L248
AD24  20 8B B0    L247   JSR L325        ;LOCATE VARIABLE
AD27  85 49       L248   STA FORPNT      ;SAVE VARIABLE ADDRESS
AD29  84 4A              STY FORPNT+1
AD2B  20 8A A3           JSR L0         ;SCAN STACK FOR 'FOR-NEXT' LOOP
AD2E  F0 05              BEQ L245       ;FOUND IT
AD30  A2 0A              LDX #$0A       ;POINTER TO ERROR MESSAGE
AD32  4C 37 A4    L249   JMP L10         ;OUTPUT 'NEXT WITHOUT FOR' ERROR
AD35  9A          L245   TXS            ;GET STACK POINTER
AD36  8A                 TXA
AD37  18                 CLC
AD38  69 04              ADC #$04       ;POINT TO STEP VARIABLE IN STACK
AD3A  48                 PHA
AD3B  69 06              ADC #$06       ;POINT TO 'TO' VARIABLE IN STACK
AD3D  85 24              STA INDEX+2
AD3F  68                 PLA
AD40  A0 01              LDY #$01       ;SET MSB MEMORY POINTER TO STACK
AD42  20 A2 BB           JSR L496       ;TRANSFER VAR. FROM STACK TO FAC#1
AD45  BA                 TSX            ;TRANSFER STACK POINTER TO .X
AD46  BD 09 01           LDA BAD+9,X    ;AND GET SIGN
AD49  85 66              STA FACSGN     ;STORE IN FAC#1 SIGN
AD4B  A5 49              LDA FORPNT     ;GET 'FOR-NEXT' VARIABLE POINTERS
AD4D  A4 4A              LDY FORPNT+1
AD4F  20 67 B8           JSR L469       ;FAC#1 = FAC#1 + FAC#2(.A,.Y)
AD52  20 D0 BB           JSR L607       ;TRANSFER FAC#1 TO VARIABLE
AD55  A0 01              LDY #$01       ;SET PNTR TO STACK FOR NEXT ROUTINE
AD57  20 5D BC           JSR L316       ;COMPARE FAC#1 TO 'TO' VALUE IN STACK
AD5A                                    ;POINTED TO BY VALUE IN LOCATION $24
AD5A  BA                 TSX
AD5B  38                 SEC
AD5C  FD 09 01           SBC BAD+9,X
AD5F  F0 17              BEQ L254
AD61  BD 0F 01           LDA BAD+15,X   ;GET CURRENT BASIC LINE NUMBER
AD64  85 39              STA CURLIN     ;AND SAVE LSB
AD66  BD 10 01           LDA BAD+16,X
AD69  85 3A              STA CURLIN+1   ;SAVE MSB
AD6B  BD 12 01           LDA BAD+18,X   ;GET LOOP RETURN ADDRESS
AD6E  85 7A              STA TXTPTR     ;AND SAVE LSB
AD70  BD 11 01           LDA BAD+17,X
```

```
LOC   CODE        LINE

AD73  85 7B             STA TXTPTR+1   ;AND SAVE MSB IN CHARGET POINTERS
AD75  4C AE A7    L250  JMP L99         ;GOTO BASIC CONTROL LOOP
AD78              ;**************************************
AD78              ;*TYPE MISMATCH CHECK
AD78              ;**************************************
AD78  8A          L254  TXA
AD79  69 11             ADC #$11       ;DELETE 'FOR-NEXT' FROM STACK
AD7B  AA                TAX
AD7C  9A                TXS
AD7D  20 79 00          JSR CHRGOT     ;GET LAST CHARACTER
AD80  C9 2C             CMP #$2C       ;IS IT A COMMA
AD82  D0 F1             BNE L250       ;NO THEN EXIT TO CONTROL LOOP
AD84  20 73 00          JSR CHRGET     ;GET NEXT CHARACTER
AD87  20 24 AD          JSR L247       ;DO NEXT 'NEXT' VARIABLE
AD8A              ;**************************************
AD8A              ;*GET TERM AND CHECK THAT IT IS NUMERIC
AD8A              ;**************************************
AD8A  20 9E AD    L253  JSR L256        ;EVALUATE TERM
AD8D  18          L96   CLC
AD8E  24                .BYT $24
AD8F  38          L95   SEC
AD90  24 0D       L312  BIT VALTYP      ;TEST VARIABLE TYPE FLAG
AD92  30 03             BMI L257       ;BRANCH IF >127
AD94  B0 03             BCS L255       ;GO TO ERROR
AD96  60          L149  RTS
AD97  B0 FD       L257  BCS L149        ;IF CARRY SET EXIT
AD99  A2 16       L255  LDX #$16        ;SET FLAG FOR ERROR
AD9B  4C 37 A4          JMP L10        ;OUTPUT ERROR MESSAGE
AD9E                    .END
AD9E                    .LIB B6
AD9E              ;**************************************
AD9E              ;*EVALUATE EXPRESSION.
AD9E              ;*A LONG AND VERY IMPORTANT ROUTINE WHICH
AD9E              ;*PARSES ANY EXPRESSION, NUMERIC OR STRING,
AD9E              ;*CHECKING FOR SYNTAX ERRORS AND EVALUATING
AD9E              ;*THE TYPE OF EXPRESSION AND RESULT. THE
AD9E              ;*ROUTINE EVALUATES AN EXPRESSION WHOSE STARTING
AD9E              ;*ADDRESS IS POINTED TO BY THE CHARGET POINTERS
AD9E              ;*$7A-$7B. SINCE THE ROUTINE INVOLVES A LOT
AD9E              ;*OF STACK PROCESSING IT FIRST CHECKS THAT
AD9E              ;*THERE IS SUFFICIENT SPACE (IT SHOULD BE NOTED
AD9E              ;*THAT LONG AND COMPLEX EXPRESSIONS CAN
AD9E              ;*GENERATE AN '?OUT OF MEMORY ERROR' BECAUSE OF
AD9E              ;*INSUFFICIENT STACK SPACE). THE EXPRESSION
AD9E              ;*TYPE IS DETERMINED AND STORED IN LOCATION
AD9E              ;*$0D, A CONTENTS OF $FF=STRING EXPRESSION AND
AD9E              ;*$00=NUMERIC. A SERIES OF ROUTINES THEN
AD9E              ;*EVALUATES THE EXPRESSION AND IF IT IS NUMERIC
AD9E              ;*STORES THE RESULT IN FLOATING ACCUMULATOR #1.
AD9E              ;*IF IT IS A STRING EXPRESSION THEN THE STRING
AD9E              ;*LENGTH IS STORED IN THE ACCUMULATOR AND THE
AD9E              ;*STRING POINTER IS IN LOCATIONS $64 AND $65.
AD9E              ;*THE FOLLOWING ARE THE ENTRY POINTS AND
AD9E              ;*FUNCTIONS OF SOME OF THE ROUTINES USED:
AD9E              ;*  $ADA9 - PUSH .A TO STACK AND RUN ROUTINES
AD9E              ;*  $ADB8 - TEST FOR COMBINATION OF <=> AND
AD9E              ;*          STORE CODE IN $4D
AD9E              ;*  $ADD7 - PROCESS STRING OPERATORS
AD9E              ;*  $AE20 - PUSH ARGUMENT IN FLOATING ACCUMULATOR
AD9E              ;*          #1 ONTO THE STACK. STACK FORMAT AS
AD9E              ;*          FOLLOWS:
```

```
LOC   CODE        LINE

AD9E              ;*          1 .. $AD
AD9E              ;*          2 .. $FA
AD9E              ;*          3 .. OPERATION ADDRESS HI
AD9E              ;*          4 .. OPERATION ADDRESS LO
AD9E              ;*          5 .. SIGN OF VALUE IN FAC #1
AD9E              ;*          6 .. VALUE IN FLOATING
AD9E              ;*          7 .. ACCUMULATOR #1
AD9E              ;*          8 ..     "
AD9E              ;*          9 ..     "
AD9E              ;*         10 .. EXPONENT OF FAC #1
AD9E              ;*         11 .. COMPARE FLAG (FROM LOC $4D)
AD9E              ;*         12 .. OPERATION HIERARCHY
AD9E              ;*         THE OPERATION ADDRESS IS OBTAINED FROM
AD9E              ;*         THE TABLE STARTING AT $A080 THIS TABLE
AD9E              ;*         ALSO CONTAINS THE OPERATION HIERARCHY
AD9E              ;*         THEY ARE STORED AS 3 BYTES -HIERARCHY
AD9E              ;*         IN ONE BYTE AND A TWO BYTE OPERATION
AD9E              ;*         ADDRESS. BYTES ONE AND TWO OF THE STACK
AD9E              ;*         ARE THE EVALUATION RETURN ADDRESS AND
AD9E              ;*         ARE A FIXED VALUE.
AD9E              ;* $AE58 - PUT STACK CONTENTS INTO FLOATING
AD9E              ;*         ACCUMULATOR #2 AND PUT THE EXPONENT IN
AD9E              ;*         THE ACCUMULATOR
AD9E              ;* $AE83 - EVALUATION ROUTINE CHECKS FOR ASCII
AD9E              ;*         NUMERIC STRINGS AND OPERATORS
AD9E              ;* $AEA8 - PI IN FLOATING POINT NOTATION
AD9E              ;* $AEF1 - EVALUATE EXPRESSION WITHIN PARENTHESIS
AD9E              ;* $AEF7 - SYNTAX ERROR IF CHARGET NOT POINT TO')'
AD9E              ;* $AEFA - SYNTAX ERROR IF CHARGET NOT POINT TO'('
AD9E              ;* $AEFD - SYNTAX ERROR IF CHARGET NOT POINT TO','
AD9E              ;* $AEFF - SYNTAX ERROR AND RETURN IF READY
AD9E              ;* $AF08 - SYNTAX ERROR IF CHARGET DOES NOT POINT
AD9E              ;*         TO A BYTE IDENTICAL TO THAT IN .A. IF
AD9E              ;*         IT DOES THEN .A RETURNS WITH THE NEXT
AD9E              ;*         CHARACTER AFTER IT
AD9E              ;****************************************
AD9E  A6 7A       L256   LDX TXTPTR      ;PUT CHARGET POINTER IN .X
ADA0  D0 02              BNE L134       ;IF NOT ZERO, JUMP
ADA2  C6 7B              DEC TXTPTR+1   ;DECREMENT CHARGET MSB POINTER
ADA4  C6 7A       L134   DEC TXTPTR      ;DECREMENT CHARGET LSB
ADA6  A2 00              LDX #$00
ADA8  24                 .BYT $24
ADA9  48          L258   PHA             ;PUSH TO STACK
ADAA  8A                 TXA
ADAB  48                 PHA
ADAC  A9 01              LDA #$01
ADAE  20 FB A3           JSR L6         ;CHECK STACK DEPTH
ADB1  20 83 AE           JSR L277       ;GET ARITHMETIC TERM
ADB4  A9 00              LDA #$00
ADB6  85 4D              STA BASTMP+2   ;SET COMPARISON SYMBOL TO ZERO
ADB8  20 79 00    L276   JSR CHRGOT      ;GET LAST CHARACTER
ADBB  38          L432   SEC
ADBC  E9 B1              SBC #$B1       ;SUBTRACT $B1 FROM CHARACTER VALUE
ADBE  90 17              BCC L262       ;TO MASK OUT CHARACTERS LESS THAN $B1
ADC0  C9 03              CMP #$03       ;CHECK IF GREATER THAN 3 TO MASK OUT
ADC2  B0 13              BCS L262       ;ALL CHARACTERS EXCEPT <=>
ADC4  C9 01              CMP #$01       ;IF <= SET CARRY
ADC6  2A                 ROL A          ;PUT CARRY IN BIT 0
ADC7  49 01              EOR #$01       ;INVERT
ADC9  45 4D              EOR BASTMP+2   ;INVERT COMPARISON SYMBOL
ADCB  C5 4D              CMP BASTMP+2
```

```
LOC   CODE        LINE

ADCD  90 61             BCC L268        ;IF < $AD THEN BRANCH
ADCF  85 4D             STA BASTMP+2
ADD1  20 73 00          JSR CHRGET      ;GET NEXT CHARACTER
ADD4  4C BB AD          JMP L432        ;AND DO AGAIN
ADD7  A6 4D       L262  LDX BASTMP+2     ;NOTE INPUT CHARACTER -$B1 IN .A
ADD9  D0 2C             BNE L297        ;IF COMPARISON SYMBOL THEN BRANCH
ADDB  B0 7B             BCS L97         ;PULL OFF STACK IF FUNCTION OR STRING
ADDD  69 07             ADC #$07        ;CHECK TOKENS FROM + TO OR
ADDF  90 77             BCC L97         ;PULL OFF STACK IF NOT
ADE1  65 0D             ADC VALTYP      ;TYPE FLAG $FF=STRING,$00=NUMERIC
ADE3  D0 03             BNE L260
ADE5  4C 3D B6          JMP L418        ;CONCATENATE IF ZERO
ADE8  69 FF       L260  ADC #$FF
ADEA  85 22             STA INDEX
ADEC  0A                ASL A           ;MULTIPLY $22 BY 3 TO
ADED  65 22             ADC INDEX       ;POINT TO OPERATOR
ADEF  A8                TAY             ;VECTOR TABLE
ADF0  68          L265  PLA
ADF1  D9 80 A0          CMP $A080,Y     ;GET OPERATOR PRIORITY
ADF4  B0 67             BCS L270        ;PULL IF EQUAL
ADF6  20 8D AD          JSR L96         ;GET TERM AND CHECK NUMERIC
ADF9  48          L273  PHA
ADFA  20 20 AE    L274  JSR L269         ;DO PUSH
ADFD                    ;
ADFD                    ;NOTE: THIS IS USED TO PUT THE RETURN ADDRESS ON
ADFD                    ;THE STACK WHICH WILL ENSURE AUTOMATIC RETURN TO
ADFD                    ;THE NEXT INSTRUCTION AFTER DOING THE OPERATION
ADFD                    ;
ADFD  68                PLA
ADFE  A4 4B             LDY BASTMP      ;SAVE .Y
AE00  10 17             BPL L272        ;GET PRIORITY CODE
AE02  AA                TAX
AE03  F0 56             BEQ L264        ;NO HIERARCHY SO END OF EXPRESSION
AE05  D0 5F             BNE L278        ;PULL VALUE OFF STACK
AE07  46 0D       L297  LSR VALTYP       ;SHIFT VARIABLE FLAG
AE09  8A                TXA
AE0A  2A                ROL A
AE0B  A6 7A             LDX TXTPTR      ;PUT POINTER TO PREVIOUS CHARGET
AE0D  D0 02             BNE L263        ;CHARACTER
AE0F  C6 7B             DEC TXTPTR+1
AE11  C6 7A       L263  DEC TXTPTR
AE13  A0 1B             LDY #$1B        ;POINT TO COMPARE IN HIERARCHY TABLE
AE15  85 4D             STA BASTMP+2    ;STORE IN COMPARISON SYMBOL FLAG
AE17  D0 D7             BNE L265
AE19  D9 80 A0    L272  CMP $A080,Y      ;GET PRIORITY CODE
AE1C  B0 48             BCS L278
AE1E  90 D9             BCC L273        ;DO AGAIN
AE20  B9 82 A0    L269  LDA $A082,Y      ;GET OPERATION ADDRESS MSB
AE23  48                PHA             ;PUSH TO STACK
AE24  B9 81 A0          LDA $A081,Y     ;GET OPERATION ADDRESS LSB
AE27  48                PHA             ;PUSH TO STACK
AE28  20 33 AE          JSR L261        ;PUT OPERANDS ON STACK
AE2B  A5 4D             LDA BASTMP+2
AE2D  4C A9 AD          JMP L258        ;RETURN TO LOOP START
AE30  4C 08 AF    L268  JMP L94          ;GOTO SYNTAX ERROR ON .A ROUTINE
AE33  A5 66       L261  LDA FACSGN       ;GET FAC#1 SIGN
AE35  BE 80 A0          LDX $A080,Y     ;GET HIERARCHY
AE38  A8          L275  TAY
AE39  68                PLA
AE3A  85 22             STA INDEX       ;GET RETURN ADDRESS OFF STACK AND
AE3C  E6 22             INC INDEX       ;CHANGE TO JUMP ADDRESS
```

```
LOC   CODE        LINE

AE3E  68                  PLA             ;PUSH TO STACK
AE3F  85 23               STA INDEX+1     ;MSB JUMP ADDRESS
AE41  98                  TYA
AE42  48                  PHA             ;PUSH HIERARCHY TO STACK
AE43  20 1B BC    L101    JSR L518         ;ROUND FAC #1
AE46  A5 65               LDA FACHO+3     ;PUSH FAC#1 TO STACK
AE48  48                  PHA
AE49  A5 64               LDA FACHO+2
AE4B  48                  PHA
AE4C  A5 63               LDA FACHO+1
AE4E  48                  PHA
AE4F  A5 62               LDA FACHO
AE51  48                  PHA
AE52  A5 61               LDA FACEXP      ;FAC#1 EXPONENT
AE54  48                  PHA
AE55  6C 22 00            JMP ($0022)     ;JUMP TO OPERATION
AE58  A0 FF       L97     LDY #$FF
AE5A  68                  PLA             ;GET HIERARCHY
AE5B  F0 23       L264    BEQ L271         ;NO HIERARCHY SO END EXPRESSION
AE5D  C9 64       L270    CMP #$64         ;IS IT A COMPARISON?
AE5F  F0 03               BEQ L267        ;YES
AE61  20 8D AD            JSR L96         ;CHECK ON NUMERIC
AE64  84 4B       L267    STY BASTMP       ;SAVE .Y
AE66  68          L278    PLA
AE67  4A                  LSR A
AE68  85 12               STA TANSGN      ;SET COMPARISON EVALUATION FLAG
AE6A  68                  PLA
AE6B  85 69               STA ARGEXP      ;PULL VAL. OFF STACK AND PUT IN FAC#2
AE6D  68                  PLA
AE6E  85 6A               STA ARGHO
AE70  68                  PLA
AE71  85 6B               STA ARGHO+1
AE73  68                  PLA
AE74  85 6C               STA ARGHO+2
AE76  68                  PLA
AE77  85 6D               STA ARGHO+3
AE79  68                  PLA
AE7A  85 6E               STA ARGSGN      ;SIGN
AE7C  45 66               EOR FACSGN      ;COMPARE FAC#1 SIGN WITH FAC#2 SIGN
AE7E  85 6F               STA ARISGN      ;AND SAVE COMPARISON
AE80  A5 61       L271    LDA FACEXP       ;GET FAC#1 EXPONENT
AE82  60                  RTS
AE83  6C 0A 03    L277    JMP ($030A)      ;GET ARITHMETIC ELEMENT LINK VECTOR
AE86  A9 00               LDA #$00
AE88  85 0D               STA VALTYP      ;SET TYPE FLAG TO NUMERIC
AE8A  20 73 00    L259    JSR CHRGET       ;GET NEXT CHARACTER
AE8D  B0 03               BCS L284        ;BRANCH IF : OR GREATER
AE8F  4C F3 BC    L286    JMP L532         ;PUT STRING INTO FAC #1
AE92  20 13 B1    L284    JSR L338         ;CHECK ALPHANUMERIC
AE95  90 03               BCC L279        ;NO
AE97  4C 28 AF            JMP L298        ;SEARCH FOR VARIABLE
AE9A  C9 FF       L279    CMP #$FF         ;COMPARE WITH $FF
AE9C  D0 0F               BNE L281        ;IF NOT THEN CHECK PARENTHESIS
AE9E              ;AND EVALUATE EXPRESSION WITHIN THEM
AE9E  A9 A8               LDA #$A8        ;POINT TO CONSTANT PI LSB
AEA0  A0 AE               LDY #$AE        ;MSB
AEA2  20 A2 BB            JSR L496        ;TRANSFER PI TO FAC #1
AEA5  4C 73 00            JMP CHRGET      ;GET NEXT CHARACTER
AEA8              ;
AEA8              ;CONSTANT PI IN FLOATING POINT
AEA8              ;
```

```
LOC   CODE        LINE

AEA8  82                .BYT $82,$49,$0F,$DA,$A1
AEA9  49
AEAA  0F
AEAB  DA
AEAC  A1
AEAD  C9 2E       L281   CMP #$2E         ;IS CHARACTER A '.'
AEAF  F0 DE              BEQ L286        ;YES
AEB1  C9 AB              CMP #$AB        ;IS TOKEN A '+'
AEB3  F0 58              BEQ L111        ;YES
AEB5  C9 AA              CMP #$AA        ;IS TOKEN A '*'
AEB7  F0 D1              BEQ L259        ;YES
AEB9  C9 22              CMP #$22
AEBB  D0 0F              BNE L288        ;NO
AEBD  A5 7A       L283   LDA TXTPTR       ;GET CHARGET PROGRAM
AEBF  A4 7B              LDY TXTPTR+1    ;POINTER AND INCREMENT
AEC1  69 00              ADC #$00
AEC3  90 01              BCC L214
AEC5  C8                 INY
AEC6  20 87 B4    L214   JSR L440         ;SET UP STRING POINTERS
AEC9  4C E2 B7           JMP L455        ;SET PROGRAM POINTERS TO STRING
AECC  C9 A8       L288   CMP #$A8         ;IS TOKEN 'NOT'
AECE  D0 13              BNE L287        ;NO
AED0  A0 18              LDY #$18
AED2  D0 3B              BNE L285        ;BRANCH ALWAYS
AED4                     ;****************************************
AED4                     ;*PERFORM 'NOT' OPERATION.
AED4                     ;*DOES A FLOATING POINT TO INTEGER CONVERSION
AED4                     ;*THEN PERFORMS A NOT ON LOCATIONS $64,$65
AED4                     ;*AND THEN REFLOATS THE VALUE INTO FAC#1.
AED4                     ;****************************************
AED4  20 BF B1    NOT    JSR L452         ;CONVERT CONTENTS OF FAC#1 TO INTEGER
AED7  A5 65              LDA FACHO+3     ;INTEGER MSB
AED9  49 FF              EOR #$FF        ;NOT IT
AEDB  A8                 TAY             ;AND PUT IN .Y
AEDC  A5 64              LDA FACHO+2     ;THEN NOT LSB
AEDE  49 FF              EOR #$FF
AEE0                     ;INTEGER VALUE IN .A LOW AND .Y HIGH
AEE0  4C 91 B3           JMP L385        ;CONVERT INTEGER TO FLOATING POINT
AEE3                     ;
AEE3                     ;DO 'FN' FUNCTION
AEE3                     ;
AEE3  C9 A5       L287   CMP #$A5         ;IS IT 'FN' TOKEN
AEE5  D0 03              BNE L289        ;NO
AEE7  4C F4 B3           JMP L387        ;PERFORM 'FN'
AEEA  C9 B4       L289   CMP #$B4         ;IS IT 'SGN' TOKEN
AEEC  90 03              BCC L292        ;IF LESS THEN NO FUNCTION
AEEE  4C A7 AF           JMP L306        ;SET UP FUNCTION REFERENCE
AEF1  20 FA AE    L292   JSR L355         ;SCAN PAST OPEN BRACKETS
AEF4  20 9E AD           JSR L256        ;EVALUATE EXPRESSION ROUTINE
AEF7  A9 29       L294   LDA #$29         ;IS IT ')'
AEF9  2C                 .BYT $2C
AEFA  A9 28       L355   LDA #$28         ;IS IT '('
AEFC  2C                 .BYT $2C
AEFD  A9 2C       L296   LDA #$2C         ;IS IT ','
AEFF  A0 00       L242   LDY #$00
AF01  D1 7A              CMP (TXTPTR),Y ;COMPARE WITH CURRENT
AF03  D0 03              BNE L94
AF05  4C 73 00           JMP CHRGET      ;GET NEXT CHARACTER
AF08  A2 0B       L94    LDX #$0B         ;SET FLAG FOR 'SYNTAX ERROR'
AF0A  4C 37 A4           JMP L10         ;OUTPUT ERROR MESSAGE
AF0D  A0 15       L111   LDY #$15         ;OFFSET FOR OPERATOR EVALUATION
```

```
LOC   CODE        LINE

AF0F  68          L285   PLA
AF10  68                 PLA
AF11  4C FA AD           JMP L274       ;EVALUATE
AF14             ;
AF14             ;CHECK INTEGER WITHIN RANGE +-32768
AF14             ;
AF14  38         L290   SEC
AF15  A5 64              LDA FACHO+2    ;GET INTEGER LSB
AF17  E9 00              SBC #$00
AF19  A5 65              LDA FACHO+3    ;GET INTEGER MSB
AF1B  E9 A0              SBC #$A0
AF1D  90 08              BCC L300       ;IF OK BRANCH
AF1F  A9 A2              LDA #$A2       ;SET INTEGER VALUE TO POINT
AF21  E5 64              SBC FACHO+2    ;TO CHARGET RESTORE
AF23  A9 E3              LDA #$E3       ;ROUTINE ADDRESS
AF25  E5 65              SBC FACHO+3
AF27  60         L300   RTS
AF28                    .END
AF28                    .LIB B7
AF28             ;***************************************
AF28             ;*VARIABLE NAME SETUP.
AF28             ;*THIS FINDS VARIABLES IN RAM AND EVALUATES
AF28             ;*NUMERIC VARIABLES AND SOME STRING VARIABLES.
AF28             ;*THE VARIABLE WHICH IS POINTED TO BY THE
AF28             ;*CHARGET POINTERS $7A,$7B IS SEARCHED FOR BY
AF28             ;*THE ROUTINE $B08B, IF IT IS NOT FOUND THEN
AF28             ;*IT IS CREATED. THE STARTING ADDRESS OF THE
AF28             ;*VARIABLE STORAGE LOCATION IS STORED IN THE
AF28             ;*ACCUMULATOR (LOW ORDER BYTE) AND THE .Y
AF28             ;*INDEX REGISTER (HIGH ORDER BYTE). THE NUMERIC
AF28             ;*VARIABLES ARE CALCULATED AND STORED IN
AF28             ;*FLOATING ACCUMULATOR #1. STRINGS ARE NOT
AF28             ;*PROCESSED EXCEPT FOR TI$ WHICH HAS A NUMERIC
AF28             ;*EQUIVALENT. THE FOLLOWING ARE ENTRY POINTS
AF28             ;*OF SOME OF THE SUBROUTINES IN THIS MAIN
AF28             ;*ROUTINE:
AF28             ;*    $AF48 - READ CLOCK AND SET UP TI$
AF28             ;*    $AF5D - EVALUATE INTEGER VARIABLE INTO FAC#1
AF28             ;*    $AF84 - EVALUATE TI INTO FAC#1
AF28             ;*    $AF9A - EVALUATE ST INTO FAC#1
AF28             ;*    $AFA0 - EVALUATE FLOATING POINT VARIABLE AND
AF28             ;*            PUT INTO FAC#1
AF28             ;*    $AFA7 - SET UP FUNCTION REFERENCES
AF28             ;***************************************
AF28  20 8B B0   L298   JSR L325        ;LOCATE VARIABLE
AF2B  85 64             STA FACHO+2    ;VARIABLE STARTING ADDRESS LSB
AF2D  84 65             STY FACHO+3    ;MSB
AF2F  A6 45             LDX VARNAM     ;CURRENT VARIABLE NAME ADDRESS LSB
AF31  A4 46             LDY VARNAM+1   ;MSB
AF33  A5 0D             LDA VALTYP     ;VARIABLE TYPE FLAG
AF35  F0 26             BEQ L301       ;BRANCH IF NUMERIC
AF37  A9 00             LDA #$00       ;SET ROUNDING BYTE TO
AF39  85 70             STA FACOV      ;ZERO
AF3B  20 14 AF          JSR L290       ;CHECK IN RANGE
AF3E  90 1C             BCC L282       ;IN RANGE
AF40  E0 54             CPX #$54       ;CHECK IF NEXT CHARACTER 'T'
AF42  D0 18             BNE L282       ;NO THEN RTS
AF44  C0 C9             CPY #$C9       ;IS CHARACTER I(STRING)
AF46  D0 14             BNE L282       ;NO THEN RTS
AF48  20 84 AF          JSR L305       ;GET TIME INTO FAC#1
AF4B  84 5E             STY FACEXP-3   ;SET TO ZERO
```

```
LOC   CODE        LINE

AF4D  88                  DEY
AF4E  84 71               STY FBUFPT      ;SET TO $FF
AF50  A0 06               LDY #$06
AF52  84 5D               STY FACEXP-4    ;SET TO $06 FOR TI$ LENGTH
AF54  A0 24               LDY #$24        ;TI TO TI$ CONSTANTS POINTER
AF56  20 68 BE            JSR L574        ;PRODUCE A STRING
AF59  4C 6F B4            JMP L388        ;SETTING STRING POINTERS
AF5C  60            L282  RTS
AF5D  24 0E         L301  BIT INTFLG       ;CHECK FOR INTEGER/F POINT
AF5F  10 0D               BPL L299        ;BRANCH IF NUMERIC
AF61  A0 00               LDY #$00        ;SET POINTER TO ZERO
AF63  B1 64               LDA (FACHO+2),Y ;GET VARIABLE LSB
AF65  AA                  TAX             ;PUT IN .X
AF66  C8                  INY
AF67  B1 64               LDA (FACHO+2),Y ;GET VARIABLE MSB
AF69  A8                  TAY             ;PUT IN .Y
AF6A  8A                  TXA             ;PUT LSB IN .A
AF6B  4C 91 B3            JMP L385        ;SAVE INTEGER AND CONVERT TO F POINT
AF6E  20 14 AF      L299  JSR L290         ;CHECK WITHIN RANGE
AF71  90 2D               BCC L307        ;IN RANGE
AF73  E0 54               CPX #$54        ;COMPARE WITH 'T'
AF75  D0 1B               BNE L302        ;NO
AF77  C0 49               CPY #$49        ;COMPARE WITH 'I'
AF79  D0 25               BNE L307        ;NO
AF7B  20 84 AF            JSR L305        ;PUT TIME IN FAC#1
AF7E  98                  TYA
AF7F  A2 A0               LDX #$A0        ;SET POINTERS
AF81  4C 4F BC            JMP L557        ;AND DEAL WITH OTHER NUMERICS
AF84                ;
AF84                ;EVALUATE TI, PUT IN FAC#1
AF84                ;
AF84  20 DE FF      L305  JSR L165         ;GET TIME
AF87  86 64               STX FACHO+2     ;PUT JIFFY CLOCK CONTENTS
AF89  84 63               STY FACHO+1     ;INTO FAC#1
AF8B  85 65               STA FACHO+3
AF8D  A0 00               LDY #$00        ;SET LSB TO 0
AF8F  84 62               STY FACHO
AF91  60                  RTS
AF92                ;
AF92                ;EVALUATE ST, PUT IN FAC#1
AF92                ;
AF92  E0 53         L302  CPX #$53         ;COMPARE WITH 'S'
AF94  D0 0A               BNE L307        ;NO
AF96  C0 54               CPY #$54        ;COMPARE WITH 'I'
AF98  D0 06               BNE L307        ;NO
AF9A  20 B7 FF            JSR L669        ;GET STATUS
AF9D  4C 3C BC            JMP L521        ;CHANGE .A TO FLOATING POINT
AFA0                ;
AFA0                ;EVALUATE FLOATING POINT, PUT IN FAC#1
AFA0                ;
AFA0  A5 64         L307  LDA FACHO+2      ;GET VARIABLE ADDRESS LSB
AFA2  A4 65               LDY FACHO+3     ;AND MSB
AFA4  4C A2 BB            JMP L496        ;PUT VARIABLE IN FAC#1
AFA7                ;
AFA7                ;SET UP FUNCTION REFERENCES
AFA7                ;
AFA7  0A            L306  ASL A            ;MULTIPLY .A BY 2
AFA8  48                  PHA             ;AND PUSH TO STACK
AFA9  AA                  TAX             ;AND PUT IN .X
AFAA  20 73 00            JSR CHRGET      ;GET NEXT CHARACTER
AFAD  E0 8F               CPX #$8F
```

```
LOC   CODE        LINE

AFAF  90 20               BCC L295        ;IF FUNCTION THEN EVALUATE
AFB1  20 FA AE            JSR L355        ;CHECK ON BRACKETS OPEN
AFB4  20 9E AD            JSR L256        ;GET TERM
AFB7  20 FD AE            JSR L296        ;CHECK ON BRACKETS CLOSED
AFBA  20 8F AD            JSR L95         ;CHECK ON STRING
AFBD  68                  PLA
AFBE  AA                  TAX             ;GET .X OFF STACK
AFBF  A5 65               LDA FACHO+3     ;PUT STRING DESCRIPTOR
AFC1  48                  PHA             ;ADDRESS
AFC2  A5 64               LDA FACHO+2     ;ONTO THE
AFC4  48                  PHA             ;STACK
AFC5  8A                  TXA             ;RESTORE .X TO
AFC6  48                  PHA             ;THE STACK
AFC7  20 9E B7            JSR L195        ;GET BYTE PARAMETER INTO .X
AFCA  68                  PLA             ;SWAP LAST BYTE
AFCB  A8                  TAY             ;PARAMETER FOR CURRENT
AFCC  8A                  TXA             ;BYTE PARAMETER
AFCD  48                  PHA             ;AND PUT ON STACK
AFCE  4C D6 AF            JMP L311        ;EXECUTE FUNCTION ROUTINE
AFD1  20 F1 AE    L295    JSR L292         ;EVALUATE TERM WITHIN BRACKETS
AFD4  68                  PLA
AFD5  A8                  TAY             ;GET BYTE PARAMETER OFF STACK AND
AFD6  B9 EA 9F    L311    LDA $9FEA,Y      ;USE AS POINTER FOR VECTOR TO
AFD9  85 55               STA $55         ;CALCULATE FUNCTION,
AFDB  B9 EB 9F            LDA $9FEB,Y     ;STORE IN FUNCTION
AFDE  85 56               STA $56         ;JUMP VECTOR
AFE0  20 54 00            JSR $0054       ;EXECUTE FUNCTION
AFE3  4C 8D AD            JMP L96         ;CHECK THAT NUMERIC
AFE6              ;****************************************
AFE6              ;*PERFORM 'OR','AND' OPERATIONS.
AFE6              ;*THIS ONE ROUTINE COVERS BOTH LOGICAL OPERATIONS,
AFE6              ;*A FLAG IN LOCATION $0B DETERMINES WHICH
AFE6              ;*OPERATION IS PERFORMED, $FF='OR' AND '$00'
AFE6              ;*='AND'. THE ENTRY POINT FOR 'OR' IS $AFE6
AFE6              ;*AND 'AND' IS $AFE9. THE ARGUMENTS ARE STORED
AFE6              ;*IN FLOATING ACCUMULATOR #1 AND FLOATING
AFE6              ;*ACCUMULATOR #2, THE RESULT IS STORED IN
AFE6              ;*FLOATING ACCUMULATOR #1.
AFE6              ;****************************************
AFE6  A0 FF       OR      LDY #$FF         ;SET FLAG FOR 'OR' OPERATION
AFE8  2C                  .BYT $2C
AFE9  A0 00       FAND    LDY #$00         ;SET FLAG FOR 'AND' OPERATION
AFEB  84 0B               STY COUNTB      ;AND SAVE
AFED  20 BF B1            JSR L452        ;CHANGE FLOATING TO FIXED POINT
AFF0  A5 64               LDA FACHO+2     ;GET INTEGER LSB
AFF2  45 0B               EOR COUNTB      ;AND EXCLUSIVE OR WITH FLAG VALUE
AFF4  85 07               STA CHARAC      ;STORE
AFF6  A5 65               LDA FACHO+3     ;REPEAT FOR MSB
AFF8  45 0B               EOR COUNTB
AFFA  85 08               STA ENDCHR
AFFC  20 FC BB            JSR L392        ;MOVE FAC#2 TO FAC#1
AFFF  20 BF B1            JSR L452        ;CONVERT FAC#1 TO INTEGER
B002  A5 65               LDA FACHO+3     ;GET INTEGER MSB
B004  45 0B               EOR COUNTB      ;EXCLUSIVE OR WITH FLAG AND
B006  25 08               AND ENDCHR      ;'AND' WITH ARGUMENT 1 MSB THEN
B008  45 0B               EOR COUNTB      ;EXCLUSIVE OR WITH FLAG TO GIVE
B00A  A8                  TAY             ;DESIRED OPERATION AND STORE IN .Y
B00B  A5 64               LDA FACHO+2     ;REPEAT ABOVE FOR LSB
B00D  45 0B               EOR COUNTB
B00F  25 07               AND CHARAC
B011  45 0B               EOR COUNTB
```

```
LOC   CODE        LINE

B013  4C 91 B3          JMP L385        ;CHANGE TO FLOATING POINT
B016                ;****************************************
B016                ;*PERFORM COMPARISONS.
B016                ;*THIS ROUTINE IS IN TWO PARTS, NUMERIC
B016                ;*COMPARISONS AND STRING COMPARISONS. NUMERIC
B016                ;*COMPARISON IS DONE BY A ROUTINE AT LOCATION
B016                ;*$B01B WHICH USES THE VALUES SET UP IN
B016                ;*FLOATING ACCUMULATORS #1 AND #2 BY THE
B016                ;*NUMERIC COMPARISON AT $BC5B THE ARGUMENT
B016                ;*IN FAC#2 IS FIRST CONVERTED TO MEMORY
B016                ;*FORMAT. THE RESULT OF THE COMPARISON IS
B016                ;*STORED IN THE ACCUMULATOR, 0 MEANS THAT THE
B016                ;*VALUES ARE EQUAL, 1 MEANS THAT FAC#1 IS
B016                ;*GREATER THAN FAC#2, AND $FF MEANS THAT
B016                ;*FAC#1 IS LESS THAN FAC#2. THE STRING
B016                ;*COMPARISON ROUTINE STARTS AT $B02E. THE
B016                ;*POINTER TO THE START OF STRING #1 IS STORED
B016                ;*IN $62,$63 AND STRING #2 IN $6C,$6D. THE
B016                ;*.X REGISTER HOLDS THE RESULT OF THE
B016                ;*COMPARISON, .X=0 THEN STRINGS ARE EQUAL,
B016                ;*.X=1 THEN STRING#1 IS GREATER THAN STRING#2,
B016                ;*AND .X=$FF THEN STRING#1 IS LESS THAN
B016                ;*STRING#2. THE FLAG SHOWING WHICH COMPARISON
B016                ;*IS PERFORMED IS STORED IN $0D, $FF=
B016                ;*NUMERIC AND $00=STRING.
B016                ;****************************************
B016  20 90 AD      COMP    JSR L312        ;CHECK ON COMPARISON VARIABLE TYPE
B019  B0 13             BCS L313        ;IF STRING BRANCH
B01B  A5 6E             LDA ARGSGN      ;PUT ARGUMENT IN FAC#2
B01D  09 7F             ORA #$7F        ;INTO MEMORY FORMAT
B01F  25 6A             AND ARGHO
B021  85 6A             STA ARGHO
B023  A9 69             LDA #$69        ;SET UP MEMORY ADDRESS OF
B025  A0 00             LDY #$00        ;ARGUMENT IN FAC#2 THEN
B027  20 5B BC          JSR L308        ;COMPARE WITH CONTENTS OF FAC#1
B02A  AA                TAX
B02B  4C 61 B9          JMP L322        ;PUT THE RESULT IN FAC#1
B02E                ;
B02E                ;STRING COMPARISON
B02E                ;
B02E  A9 00         L313    LDA #$00        ;SET TYPE FLAG TO
B030  85 0D             STA VALTYP      ;ZERO FOR STRINGS
B032  C6 4D             DEC BASTMP+2    ;COMPARISON SYMBOL ACCUMULATOR -1
B034  20 A6 B6          JSR L451        ;GET STRING POINTERS
B037  85 61             STA FACEXP      ;STRING LENGTH
B039  86 62             STX FACHO       ;STRING ADDRESS LSB
B03B  84 63             STY FACHO+1     ;MSB
B03D  A5 6C             LDA ARGHO+2     ;POINTER TO SECOND STRING LSB
B03F  A4 6D             LDY ARGHO+3     ;MSB
B041  20 AA B6          JSR L156        ;GET SECOND STRING POINTERS
B044  86 6C             STX ARGHO+2     ;SECOND STRING ADDRESS LSB
B046  84 6D             STY ARGHO+3     ;MSB
B048  AA                TAX
B049  38                SEC
B04A  E5 61             SBC FACEXP      ;COMPARE STRING LENGTHS
B04C  F0 08             BEQ L315        ;BRANCH IF EQUAL LENGTH
B04E  A9 01             LDA #$01        ;SET FLAG FOR SECOND STRING SHORTER
B050  90 04             BCC L315        ;BRANCH IF SECOND STRING SHORTER
B052  A6 61             LDX FACEXP      ;SET INDEX TO FIRST STRING LENGTH
B054  A9 FF             LDA #$FF        ;SET FLAG FOR FIRST STRING SHORTER
B056  85 66         L315    STA FACSGN      ;SET FAC#1 SIGN TO LENGTH COMPARISON
```

```
LOC   CODE        LINE

B058  A0 FF             LDY #$FF
B05A  E8                INX             ;SET LENGTH TO LENGTH+1
B05B  C8          L319  INY              ;.Y=.Y+1  BUMP MEMORY POINTER
B05C  CA                DEX             ;.X=.X-1 DECREMENT STRING LENGTH
B05D  D0 07             BNE L317        ;BRANCH IF NOT END OF STRING
B05F  A6 66             LDX FACSGN      ;PUT LENGTH COMPARISON FLAG IN .X
B061  30 0F       L322  BMI L320         ;BRANCH IF FIRST STRING SHORTER
B063  18                CLC
B064  90 0C             BCC L320        ;BRANCH ALWAYS
B066  B1 6C       L317  LDA (ARGHO+2),Y ;COMPARE STRINGS CHARACTER
B068  D1 62             CMP (FACHO),Y   ;BY CHARACTER
B06A  F0 EF             BEQ L319        ;IF SAME GO BACK AND DO AGAIN
B06C  A2 FF             LDX #$FF        ;SET FLAG IF STRING#2 > STRING#1
B06E  B0 02             BCS L320        ;BRANCH IF #2 > #1
B070  A2 01             LDX #$01        ;SET FLAG IF STRING#1 < STRING#2
B072  E8          L320  INX
B073  8A                TXA
B074  2A                ROL A           ;MULTIPLY FLAG BY 2 PLUS CARRY
B075  25 12             AND TANSGN      ;COMPARISON EVALUATION FLAG
B077  F0 02             BEQ L321        ;IF NO BIT MATCH THEN BRANCH
B079  A9 FF             LDA #$FF
B07B  4C 3C BC    L321  JMP L521         ;PUT RESULT IN FAC#1
B07E                    ;***************************************
B07E                    ;*PERFORM 'DIM' COMMAND.
B07E                    ;*THE PRESENCE OF A VARIABLE OF THE SAME NAME
B07E                    ;*IS FIRST CHECKED, USING THE ROUTINE AT
B07E                    ;*$B090. IF ONE IS NOT FOUND THEN THE ROUTINE
B07E                    ;*SETS UP AN ARRAY WITH THE VARIABLE NAME AND
B07E                    ;*NUMBER OF ELEMENTS SPECIFIED IN THE 'DIM'
B07E                    ;*STATEMENT. IT CHECKS TO SEE IF CHARGET POINTS
B07E                    ;*TO A COMMA AS THE NEXT CHARACTER, IF SO THEN
B07E                    ;*THE ROUTINE LOOPS BACK AND REPEATS THE PROCEES
B07E                    ;*FOR THE NEXT SPECIFIED ARRAY.
B07E                    ;***************************************
B07E  20 FD AE    L323  JSR L296         ;CHECK ON COMMA FOR NEXT ARRAY
B081  AA          DIM   TAX
B082  20 90 B0          JSR L148        ;DIMENSION VARIABLE
B085  20 79 00          JSR CHRGOT      ;GET LAST CHARACTER
B088  D0 F4             BNE L323        ;IF NOT END CHECK FOR NEXT VARIABLE
B08A  60                RTS             ;EXIT
B08B                    ;***************************************
B08B                    ;*SEARCH FOR VARIABLE.
B08B                    ;*THIS IMPORTANT ROUTINE IS CALLED BY MANY
B08B                    ;*OTHER ROUTINES IN THE INTERPRETER. THE FIRST
B08B                    ;*FUNCTION OF THE ROUTINE IS TO VALIDATE THE
B08B                    ;*VARIABLE NAME, THE FIRST CHARACTER MUST BE
B08B                    ;*ALPHA THOUGH THE SECOND CAN BE EITHER ALPHA
B08B                    ;*OR NUMERIC, THE VARIABLE TYPE IS ALSO DETERMINED
B08B                    ;*AND THE FLAG IN $0D IS SET ACCORDINGLY. IF
B08B                    ;*THE VARIABLE IS NUMERIC THEN $0D =$00 AND IF
B08B                    ;*STRING THEN =$FF. THE NUMERIC TYPE FLAG IN
B08B                    ;*$0E IS SET TO $00 IF FLOATING POINT AND TO
B08B                    ;*$80 IF INTEGER. IF THE VARIABLE NAME IS
B08B                    ;*FOLLOWED BY A LEFT BRACKET THEN THE ROUTINE
B08B                    ;*BRANCHES TO $B1D1 WHICH FINDS OR MAKES AN
B08B                    ;*ARRAY. THE VARIABLE NAME IS STORED IN
B08B                    ;*LOCATIONS $45,$46. HAVING VERIFIED THE VARIABLE
B08B                    ;*NAME AND DETERMINED ITS TYPE THE ROUTINE
B08B                    ;*SEARCHES FOR THE VARIABLE IN THE SECTION OF
B08B                    ;*RAM ALLOCATED TO VARIABLE STORAGE, THIS IS
B08B                    ;*IMMEDIATELY ABOVE BASIC. IF FOUND THEN THE
```

```
LOC   CODE        LINE

B08B              ;*VARIABLE ADDRESS POINTER IS RETURNED IN
B08B              ;*LOCATIONS $5F,$60. IF THE VARIABLE IS NOT
B08B              ;*FOUND THEN THE ROUTINE BRANCHES TO $B11D
B08B              ;*WHERE A NEW BASIC VARIABLE IS CREATED.
B08B              ;**********************************
B08B  A2 00       L325   LDX #$00        ;START OF DIMENSION VARIABLE ROUTINE
B08D  20 79 00           JSR CHRGOT      ;GET LAST CHARACTER
B090  86 0C       L148   STX DIMFLG       ;SET 'DIM' FLAG
B092  85 45       L324   STA VARNAM       ;STORE IN CURRENT VARIABLE NAME
B094  20 79 00           JSR CHRGOT      ;GET LAST CHARACTER
B097  20 13 B1           JSR L338        ;CHECK ALPHABETIC
B09A  B0 03              BCS L333        ;YES
B09C  4C 08 AF    L389   JMP L94         ;OTHERWISE OUTPUT 'SYNTAX ERROR'
B09F  A2 00       L333   LDX #$00
B0A1  86 0D              STX VALTYP      ;SET VARIABLE TYPE TO NUMERIC
B0A3  86 0E              STX INTFLG      ;SET NUMERIC TYPE TO FLOATING POINT
B0A5  20 73 00           JSR CHRGET      ;GET NEXT CHARACTER
B0A8  90 05              BCC L326        ;NUMBER?
B0AA  20 13 B1           JSR L338        ;CHECK ALPHABETIC
B0AD  90 0B              BCC L329        ;NO
B0AF  AA          L326   TAX             ;SECOND LETTER OF NAME
B0B0  20 73 00    L327   JSR CHRGET      ;GET NEXT CHARACTER
B0B3  90 FB              BCC L327        ;GET NEXT UNTIL NOT NUMBER
B0B5  20 13 B1           JSR L338        ;CHECK ALPHABETIC
B0B8  B0 F6              BCS L327        ;NO THEN GET NEXT CHARACTER
B0BA  C9 24       L329   CMP #$24        ;IS CHARACTER '$'
B0BC  D0 06              BNE L328        ;NO
B0BE  A9 FF              LDA #$FF        ;SET TYPE FLAG TO
B0C0  85 0D              STA VALTYP      ;STRING
B0C2  D0 10              BNE L330        ;AND JUMP NEXT
B0C4  C9 25       L328   CMP #$25        ;IS CHARACTER '%'
B0C6  D0 13              BNE L331        ;NO
B0C8  A5 10              LDA SUBFLG      ;IS INTEGER ALOWED
B0CA  D0 D0              BNE L389        ;NO THEN 'SYNTAX ERROR'
B0CC  A9 80              LDA #$80        ;SET NUMERIC TYPE FLAG
B0CE  85 0E              STA INTFLG      ;TO INTEGER
B0D0  05 45              ORA VARNAM      ;SET BIT 7 OF VARIABLE NAME
B0D2  85 45              STA VARNAM
B0D4  8A          L330   TXA
B0D5  09 80              ORA #$80        ;REPEAT FOR SECOND LETTER
B0D7  AA                 TAX             ;OF NAME
B0D8  20 73 00           JSR CHRGET      ;GET NEXT CHARACTER
B0DB  86 46       L331   STX VARNAM+1     ;SAVE SECOND CHARACTER OF NAME
B0DD  38                 SEC
B0DE  05 10              ORA SUBFLG      ;OR CHARACTER WITH SUBSCRIPT FLAG
B0E0  E9 28              SBC #$28        ;THEN CHECK FOR '(' CHARACTER
B0E2  D0 03              BNE L332        ;NO
B0E4  4C D1 B1           JMP L352        ;SET UP ARRAY
B0E7  A0 00       L332   LDY #$00        ;SET SUBSCRIPT FLAG TO ZERO
B0E9  84 10              STY SUBFLG
B0EB  A5 2D              LDA VARTAB      ;GET POINTER TO START OF VARIABLES LSB
B0ED  A6 2E              LDX VARTAB+1    ;MSB
B0EF  86 60       L334   STX FACEXP-1     ;SAVE IN TEMP LOCATION
B0F1  85 5F       L341   STA FACEXP-2
B0F3  E4 30              CPX ARYTAB+1    ;COMPARE WITH START OF ARRAYS
B0F5  D0 04              BNE L340        ;NOT SAME
B0F7  C5 2F              CMP ARYTAB      ;COMPARE LSB
B0F9  F0 22              BEQ L342        ;JUMP IF SAME
B0FB  A5 45       L340   LDA VARNAM      ;GET CURRENT VARIABLE NAME
B0FD  D1 5F              CMP (FACEXP-2),Y ;COMPARE WITH VARIABLES IN MEMORY
B0FF  D0 08              BNE L336        ;FIRST CHARACTERS NOT SAME
```

```
LOC   CODE        LINE

B101  A5 46             LDA VARNAM+1    ;CHECK SECOND CHARACTER
B103  C8                INY
B104  D1 5F             CMP (FACEXP-2),Y
B106  F0 7D             BEQ L348        ;YES SAME
B108  88                DEY             ;RESTORE .Y TO DO AGAIN
B109  18          L336  CLC
B10A  A5 5F             LDA FACEXP-2    ;ADD 7 TO VARIABLE START TO
B10C  69 07             ADC #$07        ;POINT TO NEXT VARIABLE NAME
B10E  90 E1             BCC L341        ;AND CHECK NEXT
B110  E8                INX             ;IF CARRY SET INCREMENT POINTER MSB
B111  D0 DC             BNE L334        ;AND DO NEXT VARIABLE
B113                    .END
B113                    .LIB B8
B113              ;***************************************
B113              ;*CHECK IF CHARACTER IS ALPHABETIC.
B113              ;*THE CHARACTER TO BE CHECKED IS STORED IN
B113              ;*THE ACCUMULATOR. IF THE CHARACTER IS NOT
B113              ;*ALPHABETIC THEN THE CARRY FLAG IS CLEARED
B113              ;*AND IF IT IS THEN CARRY IS SET.
B113              ;***************************************
B113  C9 41       L338  CMP #$41          ;IS CHARACTER 'A' OR LARGER
B115  90 05             BCC L280        ;BRANCH OUT WITH CARRY CLEAR IF LESS
B117  E9 5B             SBC #$5B        ;IS CHARACTER GREATER THAN 'Z'
B119  38                SEC
B11A  E9 A5             SBC #$A5        ;SET CARRY IF IN RANGE
B11C  60          L280  RTS
B11D              ;***************************************
B11D              ;*CREATE NEW VARIABLE.
B11D              ;*A NEW NON ARRAY VARIABLE IS SET UP BY THIS
B11D              ;*ROUTINE. IT FIRST OPENS UP THE VARIABLE
B11D              ;*STORAGE AREA BY SEVEN BYTES TO ACCOMMODATE
B11D              ;*THE NEW VARIABLE. THIS IS DONE BY MOVING
B11D              ;*THE ENTIRE CONTENTS OF THE ARRAY STORAGE
B11D              ;*AREA WHICH LIES ABOVE THAT OF THE SIMPLE
B11D              ;*VARIABLES UP SEVEN BYTES AND ALSO CHANGING
B11D              ;*THE ARRAY POINTERS (DEFINING ALL VARIABLES
B11D              ;*AT THE BEGINNING OF A PROGRAM IMPROVES THE
B11D              ;*PROGRAM SPEED SINCE THE RELOCATION OF THE
B11D              ;*ARRAY VARIABLE STORAGE AREA IS A RELATIVELY
B11D              ;*LONG PROCESS). THE SIMPLE VARIABLE IS THEN
B11D              ;*SET UP WITH THE FIRST TWO BYTES CONTAINING
B11D              ;*THE VARIABLE NAME AND THE OTHER FIVE BYTES
B11D              ;*WHICH CONTAIN THE VALUE OR IN THE CASE OF
B11D              ;*A STRING VARIABLE THE LOCATION POINTERS ARE
B11D              ;*ALL SET TO ZERO. ON EXIT FROM THIS ROUTINE
B11D              ;*LOCATIONS $47,$48 CONTAIN THE STARTING
B11D              ;*ADDRESS OF THE VARIABLE VALUE OR POINTER
B11D              ;*AND LOCATIONS $5F,$60 THE START ADDRESS OF
B11D              ;*THE VARIABLE.
B11D              ;***************************************
B11D  68          L342  PLA               ;CHECK CALLING ROUTINE ADDRESS
B11E  48                PHA             ;ON STACK
B11F  C9 2A             CMP #$2A        ;WAS LSB $2A
B121  D0 05             BNE L345        ;NO
B123  A9 13       L337  LDA #$13          ;SET POINTER TO CONSTANT
B125  A0 BF             LDY #$BF        ;TABLE
B127  60                RTS             ;EXIT
B128  A5 45       L345  LDA VARNAM        ;GET CURRENT VARIABLE NAME CHAR 1
B12A  A4 46             LDY VARNAM+1    ;CHAR 2
B12C  C9 54             CMP #$54        ;IS CHAR#1 = 'T'
B12E  D0 0B             BNE L347        ;NO
```

```
LOC   CODE        LINE

B130  C0 C9             CPY #$C9        ;IS CHAR#2 = 'I$'
B132  F0 EF             BEQ L337        ;YES
B134  C0 49             CPY #$49        ;IS CHAR#2 = 'I'
B136  D0 03             BNE L347        ;NO
B138  4C 08 AF  L343    JMP L94          ;OUTPUT 'SYNTAX ERROR'
B13B  C9 53     L347    CMP #$53         ;IS CHAR#1 = 'S'
B13D  D0 04             BNE L344        ;NO
B13F  C0 54             CPY #$54        ;IS CHAR #2 = 'T'
B141  F0 F5             BEQ L343        ;YES, OUTPUT 'SYNTAX ERROR'
B143  A5 2F     L344    LDA ARYTAB       ;GET POINTER TO START OF
B145  A4 30             LDY ARYTAB+1    ;ARRAYS
B147  85 5F             STA FACEXP-2    ;AND SAVE IN TEMPORARY LOCATION TO
B149  84 60             STY FACEXP-1    ;GIVE OLD BLOCK START
B14B  A5 31             LDA STREND      ;GET POINTER TO END
B14D  A4 32             LDY STREND+1    ;OF ARRAYS
B14F  85 5A             STA BASTMP+15   ;AND SAVE IN TEMPORARY LOCATION TO
B151  84 5B             STY BASTMP+16   ;GIVE OLD BLOCK START +1
B153  18                CLC
B154  69 07             ADC #$07        ;ADD 7 TO GET NEW VARIABLE
B156  90 01             BCC L346
B158  C8                INY             ;IF CARRY SET INCREMENT MSB
B159  85 58     L346    STA BASTMP+13    ;THEN SAVE IN NEW POINTER
B15B  84 59             STY BASTMP+14   ;TO GIVE NEW BLOCK END
B15D  20 B8 A3          JSR L1          ;OPEN UP MEMORY USING ABOVE PARS.
B160  A5 58             LDA BASTMP+13   ;THEN SAVE NEW
B162  A4 59             LDY BASTMP+14   ;START OF
B164  C8                INY             ;ARRAYS POINTER
B165  85 2F             STA ARYTAB      ;PLUS ONE
B167  84 30             STY ARYTAB+1
B169  A0 00             LDY #$00        ;SET INDEX POINTER TO ZERO
B16B  A5 45             LDA VARNAM      ;GET FIRST CHARACTER OF VARIABLE NAME
B16D  91 5F             STA (FACEXP-2),Y ;AND SAVE IN OPENED UP VARIABLE
B16F  C8                INY             ;STORAGE AREA
B170  A5 46             LDA VARNAM+1    ;SECOND CHARACTER OF VARIABLE NAME
B172  91 5F             STA (FACEXP-2),Y ;AND SAVE
B174  A9 00             LDA #$00        ;THEN SET THE NEXT FIVE BYTES TO
B176  C8                INY             ;ZERO FOR STORAGE OF THE VALUE
B177  91 5F             STA (FACEXP-2),Y
B179  C8                INY
B17A  91 5F             STA (FACEXP-2),Y
B17C  C8                INY
B17D  91 5F             STA (FACEXP-2),Y
B17F  C8                INY
B180  91 5F             STA (FACEXP-2),Y
B182  C8                INY
B183  91 5F             STA (FACEXP-2),Y
B185  A5 5F     L348    LDA FACEXP-2     ;SET OLD BLOCK START TO
B187  18                CLC
B188  69 02             ADC #$02        ;OLD BLOCK START PLUS 2
B18A  A4 60             LDY FACEXP-1
B18C  90 01             BCC L339
B18E  C8                INY
B18F  85 47     L339    STA VARPNT       ;AND STORE IN POINTER TO ARRAY
B191  84 48             STY VARPNT+1    ;VARIABLE
B193  60                RTS
B194            ;****************************************
B194            ;*SET UP ARRAY POINTER.
B194            ;*SPACE WITHIN THE ARRAY STORAGE AREA IS
B194            ;*ALLOCATED BY THIS ROUTINE FOR THE ARRAY
B194            ;*POINTERS. TO DO THIS THE NUMBER OF ARRAY
B194            ;*SUBSCRIPTS STORED IN $0B IS MULTIPLIED BY
```

```
LOC   CODE        LINE

B194              ;*TWO AND THEN INCREMENTED BY FIVE AND THIS
B194              ;*VALUE STORED IN LOCATIONS $58,$59. IT IS
B194              ;*USED BY THE ROUTINE AT $B1D1 TO SET UP
B194              ;*AN ARRAY.
B194              ;**************************************
B194  A5 0B       L349   LDA COUNTB      ;GET NUMBER OF 'DIM' ELEMENTS
B196  0A                 ASL A           ;MULTIPLY BY TWO
B197  69 05              ADC #$05        ;AND ADD FIVE
B199  65 5F              ADC FACEXP-2    ;ADD TO $5F AND $60
B19B  A4 60              LDY FACEXP-1
B19D  90 01              BCC L362
B19F  C8                 INY
B1A0  85 58       L362   STA BASTMP+13   ;AND STORE IN RESULT POINTER
B1A2  84 59              STY BASTMP+14
B1A4  60                 RTS
B1A5              ;**************************************
B1A5              ;*-32768 IN FLOATING POINT FORMAT
B1A5              ;**************************************
B1A5  90                 .BYT $90,$80,$00,$00,$00
B1A6  80
B1A7  00
B1A8  00
B1A9  00
B1AA              ;**************************************
B1AA              ;*CONVERT FLOATING POINT NUMBER TO INTEGER
B1AA              ;*AND PUT IN .A,.Y REGISTERS.
B1AA              ;**************************************
B1AA  20 BF B1    FTOI   JSR L452
B1AD  A5 64              LDA FACHO+2
B1AF  A4 65              LDY FACHO+3
B1B1  60                 RTS
B1B2              ;**************************************
B1B2              ;*EVALUATE INTEGER EXPRESSION.
B1B2              ;*THIS TAKES AN EXPRESSION FROM BASIC AND
B1B2              ;*EVALUATES IT. IF THE RESULT IS LESS THAN
B1B2              ;*32768 IT IS CONVERTED INTO A TWO BYTE
B1B2              ;*NUMBER WHICH IS STORED IN LOCATIONS $64
B1B2              ;*AND $65 OF FAC#1.
B1B2              ;**************************************
B1B2  20 73 00    L350   JSR CHRGET      ;GET NEXT CHARACTER
B1B5  20 9E AD           JSR L256        ;EVALUATE EXPRESSION
B1B8  20 8D AD    L353   JSR L96          ;CHECK IF NUMERIC
B1BB  A5 66              LDA FACSGN      ;GET SIGN BYTE
B1BD  30 0D              BMI L153        ;IF NEGATIVE THEN 'ILLEGAL QUANTITY'
B1BF  A5 61       L452   LDA FACEXP      ;GET EXPONENT
B1C1  C9 90              CMP #$90        ;IS IT GREATER THAN 32768
B1C3  90 09              BCC L351        ;NO
B1C5  A9 A5              LDA #$A5        ;POINT TO CONSTANT -32768
B1C7  A0 B1              LDY #$B1
B1C9  20 5B BC           JSR L308        ;COMPARE FAC#1 WITH -32768
B1CC  D0 7A       L153   BNE L363        ;IF NOT, 'ILLEGAL QUANTITY ERROR'
B1CE  4C 9B BC    L351   JMP L526        ;CHANGE FLOATING POINT TO INTEGER
B1D1              ;**************************************
B1D1              ;*FIND OR MAKE AN ARRAY.
B1D1              ;*THIS IS A LONG AND COMPLEX ROUTINE WHICH IS
B1D1              ;*PRINCIPALLY CALLED BY THE ROUTINE AT $B08B.
B1D1              ;*THIS FIRST CHECKS FOR THE EXISTENCE OF THE
B1D1              ;*ARRAY VARIABLE AND OBTAINS ITS SUBSCRIPT(S).
B1D1              ;*EACH SUBSCRIPT IS STORED ON THE STACK AS FOUR
B1D1              ;*BYTES, THE FIRST TWO ARE THE NAME FLAGS AND
B1D1              ;*THE LAST TWO THE SUBSCRIPT VALUE (THIS IS
```

```
LOC   CODE        LINE

B1D1              ;*CONVERTED FROM FLOATING POINT TO INTEGER BY
B1D1              ;*THE ROUTINE AT $B1B2). THIS IS USED TO
B1D1              ;*CALCULATE THE ARRAY ELEMENT ADDRESS USING
B1D1              ;*THE COMPUTE ARRAY SUBSCRIPT SIZE ROUTINE AT
B1D1              ;*$B34C, WHICH RETURNS THE SUBSCRIPT SIZE IN
B1D1              ;*THE .X (LOW) AND .Y (HIGH) INDEX REGISTERS.
B1D1              ;*THE ARRAY ELEMENT ADDRESS IS RETURNED STORED
B1D1              ;*IN $47,$48. IF THE ARRAY IS NOT FOUND THEN
B1D1              ;*SUBROUTINE $B261 SETS UP AN ARRAY USING A
B1D1              ;*DEFAULT ARRAY 'DIM' VALUE OF 10. THE FIRST
B1D1              ;*ARRAY ELEMENT ADDRESS FOR THE NEW ARRAY IS
B1D1              ;*RETURNED IN $47,$48.
B1D1              ;**************************************
B1D1  A5 0C       L352   LDA DIMFLG      ;DEFAULT 'DIM' FLAG
B1D3  05 0E              ORA INTFLG      ;OR'ED WITH THE NUMERIC TYPE FLAG
B1D5  48                 PHA             ;AND PUSHED ONTO THE STACK
B1D6  A5 0D              LDA VALTYP      ;GET VARIABLE TYPE FLAG
B1D8  48                 PHA             ;AND PUSH TO STACK
B1D9  A0 00              LDY #$00        ;PUSH A ZERO ONTO STACK
B1DB  98          L335   TYA
B1DC  48                 PHA
B1DD  A5 46              LDA VARNAM+1    ;GET FIRST CHARACTER IN VARIABLE NAME
B1DF  48                 PHA             ;AND PUSH TO STACK
B1E0  A5 45              LDA VARNAM      ;SECOND CHARACTER OF VARIABLE NAME
B1E2  48                 PHA
B1E3  20 B2 B1           JSR L350        ;EVALUATE INTEGER EXPRESSION
B1E6  68                 PLA
B1E7  85 45              STA VARNAM      ;GET SECOND CHARACTER OF VARIABLE
B1E9  68                 PLA             ;NAME AND
B1EA  85 46              STA VARNAM+1    ;FIRST CHARACTER
B1EC  68                 PLA
B1ED  A8                 TAY             ;PUT ZERO IN .Y REGISTER
B1EE  BA                 TSX             ;TRANSFER STACK POINTER TO .X
B1EF  BD 02 01           LDA BAD+2,X     ;GET FLAG #1 AND PUSH TO
B1F2  48                 PHA             ;TOP OF STACK
B1F3  BD 01 01           LDA BAD+1,X     ;GET FLAG #2 AND PUSH
B1F6  48                 PHA             ;TO TOP OF STACK
B1F7  A5 64              LDA FACHO+2     ;GET INTEGER LSB VALUE AND
B1F9  9D 02 01           STA BAD+2,X     ;REPLACE FLAG #1
B1FC  A5 65              LDA FACHO+3     ;GET INTEGER MSB VALUE AND
B1FE  9D 01 01           STA BAD+1,X     ;REPLACE FLAG #2
B201  C8                 INY             ;INCREMENT .Y (SUBSCRIPT POINTER)
B202  20 79 00           JSR CHRGOT      ;GET LAST CHARACTER
B205  C9 2C              CMP #$2C        ;IS IT A COMMA
B207  F0 D2              BEQ L335        ;YES
B209  84 0B              STY COUNTB      ;STORE .Y IN SUBSCRIPT POINTER
B20B  20 F7 AE           JSR L294        ;CHECK IF BRACKETS CLOSED
B20E  68                 PLA
B20F  85 0D              STA VALTYP      ;GET BACK VARIABLE TYPE FLAG
B211  68                 PLA
B212  85 0E              STA INTFLG      ;GET BACK NUMERIC TYPE FLAG
B214  29 7F              AND #$7F        ;AND WITH $7F THEN
B216  85 0C              STA DIMFLG      ;STORE IN DEFAULT 'DIM' FLAG
B218  A6 2F              LDX ARYTAB      ;GET POINTER TO START OF ARRAYS
B21A  A5 30              LDA ARYTAB+1
B21C  86 5F       L354   STX FACEXP-2     ;AND STORE IN TEMPORARY LOCATION
B21E  85 60              STA FACEXP-1
B220  C5 32              CMP STREND+1    ;CHECK FOR TOP OF VARIABLES MSB
B222  D0 04              BNE L360        ;LESS
B224  E4 31              CPX STREND      ;CHECK FOR TOP OF VARIABLES LSB
B226  F0 39              BEQ L359        ;GOTO ARRAY NOT FOUND ROUTINE
```

```
LOC   CODE        LINE

B228  A0 00       L360   LDY #$00         ;INITIALISE POINTER TO ZERO
B22A  B1 5F              LDA (FACEXP-2),Y ;GET VARIABLE NAME FROM LISTING AND
B22C  C8                 INY
B22D  C5 45              CMP VARNAM       ;COMPARE WITH SOUGHT NAME
B22F  D0 06              BNE L356         ;NO MATCH ON FIRST CHARACTER
B231  A5 46              LDA VARNAM+1     ;COMPARE SECOND CHARACTER
B233  D1 5F              CMP (FACEXP-2),Y
B235  F0 16              BEQ L361         ;VARIABLE FOUND
B237  C8          L356   INY
B238  B1 5F              LDA (FACEXP-2),Y ;GET FIELD LENGTH
B23A  18                 CLC
B23B  65 5F              ADC FACEXP-2     ; AND ADD TO POINTERS
B23D  AA                 TAX
B23E  C8                 INY
B23F  B1 5F              LDA (FACEXP-2),Y ;REPEAT FOR MSB
B241  65 60              ADC FACEXP-1
B243  90 D7              BCC L354         ;CONTINUE SEARCHING
B245  A2 12       L358   LDX #$12          ;'BAD SUBSCRIPT' ERROR POINTER
B247  2C                 .BYT $2C
B248  A2 0E       L363   LDX #$0E          ;'ILLEGAL QUANTITY' ERROR POINTER
B24A  4C 37 A4    L167   JMP L10           ;PRINT ERROR MESSAGE
B24D  A2 13       L361   LDX #$13          ;'REDIM'D ARRAY' ERROR POINTER
B24F  A5 0C              LDA DIMFLG       ;GET 'DIM' SUBSCRIPT FLAG
B251  D0 F7              BNE L167         ;IF NOT ZERO THEN ERROR MESSAGE
B253  20 94 B1           JSR L349         ;ARRAY POINTER SUBROUTINE
B256  A5 0B              LDA COUNTB       ;GET NUMBER OF SUBSCRIPTS
B258  A0 04              LDY #$04
B25A  D1 5F              CMP (FACEXP-2),Y ;COMPARE WITH 'DIM' VALUE
B25C  D0 E7              BNE L358         ;NO THEN 'BAD SUBSCRIPT' ERROR
B25E  4C EA B2           JMP L372         ;SEARCH FOR ELEMENT
B261              ;
B261              ;ARRAY VARIABLE NOT FOUND THEN SET UP (DIM=10)
B261              ;
B261  20 94 B1    L359   JSR L349          ;SET UP ARRAY POINTER
B264  20 08 A4           JSR L92          ;CHECK ON FREE MEMORY
B267  A0 00              LDY #$00
B269  84 72              STY FBUFPT+1     ;SET TO ZERO
B26B  A2 05              LDX #$05         ;SET DEFAULT LENGTH
B26D  A5 45              LDA VARNAM       ;GET FIRST CHARACTER OF NAME
B26F  91 5F              STA (FACEXP-2),Y ;AND PUT IN ARRAY STORAGE
B271  10 01              BPL L357         ;NOT INTEGER
B273  CA                 DEX              ;DECREMENT LENGTH POINTER
B274  C8          L357   INY
B275  A5 46              LDA VARNAM+1     ;GET SECOND CHARACTER OF NAME
B277  91 5F              STA (FACEXP-2),Y ;PUT IN ARRAY STORAGE
B279  10 02              BPL L365
B27B  CA                 DEX
B27C  CA                 DEX
B27D  86 71       L365   STX FBUFPT       ;SET VARIABLE LENGTH
B27F  A5 0B              LDA COUNTB       ;GET SUBSCRIPT NUMBER
B281  C8                 INY
B282  C8                 INY              ;BUMP POINTER BY THREE LOCATIONS
B283  C8                 INY
B284  91 5F              STA (FACEXP-2),Y ;AND SAVE
B286  A2 0B       L366   LDX #COUNTB      ;SET X TO 11, DEFAULT 'DIM' VALUES
B288  A9 00              LDA #$00
B28A  24 0C              BIT DIMFLG
B28C  50 08              BVC L369         ;TEST IF DIMENSIONS DEFINED
B28E  68                 PLA              ;GET DIMENSION FROM STACK
B28F  18                 CLC
B290  69 01              ADC #$01         ;AND ADD ONE
```

```
LOC   CODE        LINE

B292  AA                  TAX
B293  68                  PLA
B294  69 00               ADC #$00        ;ADD CARRY IF OVER 255
B296  C8            L369  INY
B297  91 5F               STA (FACEXP-2),Y ;SAVE MSB
B299  C8                  INY
B29A  8A                  TXA
B29B  91 5F               STA (FACEXP-2),Y  ;SAVE LSB
B29D  20 4C B3            JSR L374        ;COMPUTE ARRAY SIZE TO GET FREE MEM.
B2A0  86 71               STX FBUFPT      ;SAVE VARIABLE END POINTER LSB
B2A2  85 72               STA FBUFPT+1    ;MSB
B2A4  A4 22               LDY INDEX
B2A6  C6 0B               DEC COUNTB      ;DECREMENT NUMBER OF SUBSCRIPTS
B2A8  D0 DC               BNE L366        ;AND REPEAT IF FURTHER DIMENSIONS
B2AA  65 59               ADC BASTMP+14   ;ADD FIELD LENGTH TO START ADDRESS
B2AC  B0 5D               BCS L376        ;'OUT OF MEMORY' ERROR
B2AE  85 59               STA BASTMP+14   ;OTHERWISE STORE
B2B0  A8                  TAY
B2B1  8A                  TXA
B2B2  65 58               ADC BASTMP+13   ;ADD LSB OF FIELD LENGTH
B2B4  90 03               BCC L367        ; TO START ADDRESS LSB
B2B6  C8                  INY
B2B7  F0 52               BEQ L376        ;'OUT OF MEMORY' ERROR
B2B9  20 08 A4      L367  JSR L92          ;CHECK ON FREE MEMORY SPACE
B2BC  85 31               STA STREND      ;SET POINTER TO END OF ARRAYS
B2BE  84 32               STY STREND+1
B2C0  A9 00               LDA #$00        ;FILL ARRAY WITH ZEROS
B2C2  E6 72               INC FBUFPT+1
B2C4  A4 71               LDY FBUFPT      ;WITH FIELD LENGTH VALUE AS POINTER
B2C6  F0 05               BEQ L373
B2C8  88            L371  DEY
B2C9  91 58               STA (BASTMP+13),Y ;AND STORE IN ARRAY AREA
B2CB  D0 FB               BNE L371        ;DO AGAIN UNTIL .Y=0
B2CD  C6 59         L373  DEC BASTMP+14    ;DEC MSB OF START ADDRESS + LENGTH
B2CF  C6 72               DEC FBUFPT+1    ;DECREMENT LENGTH MSB
B2D1  D0 F5               BNE L371        ;CUNTINUE UNTIL END OF ARRAY
B2D3  E6 59               INC BASTMP+14   ;RESTORE ADDRESS + LENGTH ADDRESS MSB
B2D5  38                  SEC
B2D6  A5 31               LDA STREND      ;GET POINTER TO END OF ARRAYS LSB
B2D8  E5 5F               SBC FACEXP-2    ;SUBTRACT FROM TEMPORARY
B2DA  A0 02               LDY #$02
B2DC  91 5F               STA (FACEXP-2),Y ;SAVE IN ARRAY LENGTH LOW
B2DE  A5 32               LDA STREND+1    ;REPEAT FOR LSB
B2E0  C8                  INY
B2E1  E5 60               SBC FACEXP-1
B2E3  91 5F               STA (FACEXP-2),Y ;SAVE IN ARRAY LENGTH HIGH
B2E5  A5 0C               LDA DIMFLG      ;GET 'DIM' FLAG
B2E7  D0 62               BNE L381        ;IF NOT ZERO THEN END
B2E9                ;
B2E9                ;FIND ELEMENT
B2E9                ;
B2E9  C8                  INY
B2EA  B1 5F         L372  LDA (FACEXP-2),Y ;GET NUMBER OF DIMENSIONS
B2EC  85 0B               STA COUNTB      ;AND SAVE
B2EE  A9 00               LDA #$00        ;CLEAR VARIABLE END POINTERS
B2F0  85 71               STA FBUFPT
B2F2  85 72         L364  STA FBUFPT+1
B2F4  C8                  INY
B2F5  68                  PLA             ;GET INDEX FROM STACK
B2F6  AA                  TAX             ;THEN STORE IN .X AND
B2F7  85 64               STA FACHO+2     ;MEMORY
```

```
LOC   CODE        LINE

B2F9  68                  PLA             ;GET INDEX MSB
B2FA  85 65               STA FACHO+3     ;AND SAVE
B2FC  D1 5F               CMP (FACEXP-2),Y ;COMPARE WITH VALUE IN ARRAY
B2FE  90 0E               BCC L370        ;SMALLER THEN CALCULATE ADDRESS
B300  D0 06               BNE L379        ;LARGER THEN 'BAD SUBSCRIPT' ERROR
B302  C8                  INY
B303  8A                  TXA
B304  D1 5F               CMP (FACEXP-2),Y ;IF EQUAL COMPARE LSB
B306  90 07               BCC L375        ;IF SMALLER THEN CONTINUE
B308  4C 45 B2    L379   JMP L358         ;GENERATE 'BAD SUBSCRIPT' ERROR
B30B  4C 35 A4    L376   JMP L11          ;GENERATE 'OUT OF MEMORY' ERROR
B30E              ;
B30E              ;CALCULATE ARRAY ELEMENT ADDRESS
B30E              ;
B30E  C8          L370   INY
B30F  A5 72       L375   LDA FBUFPT+1     ;GET END OF VARIABLE POINTER MSB
B311  05 71               ORA FBUFPT      ;OR WITH LSB
B313  18                  CLC
B314  F0 0A               BEQ L377
B316  20 4C B3            JSR L374        ;COMPUTE ARRAY SIZE
B319  8A                  TXA
B31A  65 64               ADC FACHO+2     ;AND ADD TO INDEX LSB
B31C  AA                  TAX
B31D  98                  TYA             ;GET MSB
B31E  A4 22               LDY INDEX
B320  65 65       L377   ADC FACHO+3      ;AND ADD TO INDEX MSB
B322  86 71               STX FBUFPT      ;SET NEW VARIABLE POINTER LSB
B324  C6 0B               DEC COUNTB      ;DECREMENT NUMBER OF DIMENSIONS
B326  D0 CA               BNE L364        ;GO BACK AND DO NEXT
B328  85 72               STA FBUFPT+1    ;SET NEW VARIABLE POINTER MSB
B32A  A2 05               LDX #$05        ;SET DEFAULT VARIABLE LENGTH TO 5
B32C  A5 45               LDA VARNAM      ;GET FIRST LETTER OF NAME
B32E  10 01               BPL L378
B330  CA                  DEX
B331  A5 46       L378   LDA VARNAM+1     ;GET SECOND CHARACTER OF NAME
B333  10 02               BPL L380
B335  CA                  DEX
B336  CA                  DEX
B337  86 28       L380   STX RESHO+2      ;LENGTH OF VARIABLE (5,3 OR 2 BYTES)
B339  A9 00               LDA #$00
B33B  20 55 B3            JSR L368        ;CALCULATE OFFSET OF ARRAY
B33E  8A                  TXA
B33F  65 58               ADC BASTMP+13
B341  85 47               STA VARPNT      ;STORE CURRENT VARIABLE ADDRESS LSB
B343  98                  TYA
B344  65 59               ADC BASTMP+14
B346  85 48               STA VARPNT+1    ;STORE CURRENT VARIABLE ADDRESS MSB
B348  A8                  TAY             ;PUT MSB OF VARIABLE ADDRESS IN .Y
B349  A5 47               LDA VARPNT      ;AND LSB IN .A
B34B  60          L381   RTS
B34C                      .END
B34C                      .LIB B9
B34C              ;
B34C              ;COMPUTE ARRAY SIZE
B34C              ;
B34C  84 22       L374   STY INDEX        ;SAVE .Y POINTER
B34E  B1 5F               LDA (FACEXP-2),Y
B350  85 28               STA RESHO+2     ;SAVE NUMBER OF ARRAY ELEMENTS LO
B352  88                  DEY
B353  B1 5F               LDA (FACEXP-2),Y
B355  85 29       L368   STA RESHO+3      ;SAVE NUMBER OF ARRAY ELEMENTS HI
```

```
LOC   CODE        LINE

B357  A9 10             LDA #$10
B359  85 5D             STA BASTMP+18  ;PUT 16 IN $5D
B35B  A2 00             LDX #$00       ;CLEAR .X AND .Y
B35D  A0 00             LDY #$00
B35F  8A          L382  TXA             ;MULTIPLY CONTENTS OF .X (LOW) AND
B360  0A                ASL A          ;.Y (HIGH) AS A 16 BIT VALUE BY 2
B361  AA                TAX
B362  98                TYA
B363  2A                ROL A
B364  A8                TAY
B365  B0 A4             BCS L376       ;CALCULATE ADDRESS OF ARRAY ELEMENT
B367  06 71             ASL FBUFPT     ;MULTIPLY VARIABLE END POINTER BY 2
B369  26 72             ROL FBUFPT+1
B36B  90 0B             BCC L384       ;CHECK IF $5D >0 THEN DO AGAIN
B36D  18                CLC
B36E  8A                TXA
B36F  65 28             ADC RESHO+2
B371  AA                TAX            ;PUT ARRAY SIZE LSB IN .X
B372  98                TYA
B373  65 29             ADC RESHO+3
B375  A8                TAY            ;PUT ARRAY SIZE MSB IN .Y
B376  B0 93             BCS L376       ;DO 'OUT OF MEMORY ERROR'
B378  C6 5D       L384  DEC BASTMP+18
B37A  D0 E3             BNE L382       ;DO AGAIN IF $5D NOT ZERO
B37C  60                RTS
B37D              ;*****************************************
B37D              ;*PERFORM 'FRE' FUNCTION.
B37D              ;*THIS DISCARDS ALL UNWANTED STRINGS, DOES A
B37D              ;*GARBAGE COLLECT AND THEN CALCULATES THE
B37D              ;*AMOUNT OF FREE MEMORY AVAILABLE. THIS IS
B37D              ;*RETURNED AS AN INTEGER TWO BYTE VALUE
B37D              ;*STORED IN $62 (LO) AND $63 (HI).
B37D              ;*****************************************
B37D  A5 0D       FRE   LDA VALTYP       ;GET VARIABLE TYPE FLAG
B37F  F0 03             BEQ L383       ;BRANCH IF NUMERIC
B381  20 A6 B6          JSR L451       ;DISCARD UNWANTED STRING
B384  20 26 B5    L383  JSR L406        ;DO GARBAGE COLLECT
B387  38                SEC
B388  A5 33             LDA FRETOP     ;GET POINTER TO STRING STORAGE LSB
B38A  E5 31             SBC STREND     ;AND SUBTRACT FROM END OF ARRAYS LSB
B38C  A8                TAY            ;STORE RESULT IN .Y
B38D  A5 34             LDA FRETOP+1   ;REPEAT FOR MSB
B38F  E5 32             SBC STREND+1
B391              ;*****************************************
B391              ;*CONVERT FIXED TO FLOATING.
B391              ;*THE TWO BYTE INTEGER VALUE IS HELD IN .A(HI)
B391              ;*AND .Y(LO). THE ROUTINE CONVERTS THESE INTO
B391              ;*FLOATING POINT FORM IN FAC#1.
B391              ;*****************************************
B391  A2 00       L385  LDX #$00
B393  86 0D             STX VALTYP     ;SET VARIABLE TYPE FLAG TO NUMERIC
B395  85 62             STA FACHO      ;SAVE RESULT LSB
B397  84 63             STY FACHO+1    ;MSB
B399  A2 90             LDX #$90
B39B  4C 44 BC          JMP L310       ;CONVERT TO FLOATING POINT
B39E              ;*****************************************
B39E              ;*PERFORM 'POS' OPERATION.
B39E              ;*THE POSITION OF THE CURSOR ON THE LINE IS
B39E              ;*OBTAINED USING ROUTINE $FFF0 A ZERO IS PUT
B39E              ;*INTO THE ACCUMULATOR AND THE ROUTINE AT
B39E              ;*$B391 CALLED TO PUT THE VALUE IN FAC#1.
```

```
LOC   CODE        LINE

B39E              ;*****************************************
B39E  38          POS    SEC
B39F  20 F0 FF           JSR L800        ;GET CURSOR POSITION
B3A2  A9 00       L291   LDA #$00
B3A4  F0 EB              BEQ L385        ;CONVERT FIXED VALUE TO FLOATING
B3A6              ;*****************************************
B3A6              ;*CHECK NOT IN 'DIRECT' MODE.
B3A6              ;*THIS CHECKS TO SEE IF THE COMMAND WAS ENTERED
B3A6              ;*FROM THE KEYBOARD IN THE 'DIRECT MODE' (WITHOUT
B3A6              ;*A LINE NUMBER). IT DOES THIS BY LOOKING AT
B3A6              ;*LOCATION $3A WHICH IS THE HIGH ORDER BYTE OF
B3A6              ;*THE CURRENT LINE NUMBER, IF THIS HAS A VALUE OF
B3A6              ;*$FF THEN THE COMMAND WAS DIRECT.
B3A6              ;*****************************************
B3A6  A6 3A       L450   LDX CURLIN+1     ;GET CURRENT LINE# MSB
B3A8  E8                 INX              ;INCREMENT TO CONVERT $FF TO $00
B3A9  D0 A0              BNE L381         ;NOT DIRECT MODE THEN RTS
B3AB  A2 15              LDX #$15         ;ERROR NUMBER FOR'ILLEGAL QUANTITY'
B3AD  2C                 .BYT $2C
B3AE  A2 1B       L208   LDX #$1B         ;ERROR NUMBER FOR'UNDEF'D FUNCTION'
B3B0  4C 37 A4           JMP L10          ;OUTPUT ERROR MESSAGE
B3B3              ;*****************************************
B3B3              ;*PERFORM 'DEF' OPERATION.
B3B3              ;*A SYNTAX CHECK IS FIRST CARRIED OUT USING
B3B3              ;*THE ROUTINE AT $B3E1. THE MODE OF OPERATION
B3B3              ;*IS THEN CHECKED TO MAKE SURE THAT IT IS IN
B3B3              ;*PROGRAM MODE, AND THEN SEARCHES FOR THE
B3B3              ;*PRESENCE OF A LEFT BRACKET. IF FOUND THEN
B3B3              ;*THE FOLLOWING VARIABLE IS LOCATED IN MEMORY
B3B3              ;*USING ROUTINE $B08B. A RIGHT BRACKET IS THEN
B3B3              ;*CHECKED FOR AND THE NEXT CHARACTER IN BASIC
B3B3              ;*TESTED TO MAKE SURE THAT IT IS AN =. THE
B3B3              ;*FIVE BYTES OF DATA OBTAINED ARE THEN PUSHED
B3B3              ;*ONTO THE STACK IN THE FOLLOWING FORMAT:
B3B3              ;*  1 .. FUNCTION TOKEN OF FIRST CHARACTER IN
B3B3              ;*       VARIABLE NAME
B3B3              ;*  2 .. VARIABLE ADDRESS POINTER FROM
B3B3              ;*  3 .. LOCATIONS $47,$48
B3B3              ;*  4 .. POINTER TO BASIC FOR CHARGET FROM
B3B3              ;*  5 .. LOCATIONS $7A,$7B
B3B3              ;*****************************************
B3B3  20 E1 B3    DEF    JSR L390         ;CHECK FN SYNTAX
B3B6  20 A6 B3           JSR L450         ;CHECK IF DIRECT MODE
B3B9  20 FA AE           JSR L355         ;CHECK IF BRACKETS OPEN
B3BC  A9 80              LDA #$80         ;SET FNX FLAG TO $80 WHICH BLOCKS
B3BE  85 10              STA SUBFLG       ;INTEGER VARIABLE
B3C0  20 8B B0           JSR L325         ;SEARCH FOR VARIABLE
B3C3  20 8D AD           JSR L96          ;CHECK ON NUMERIC
B3C6  20 F7 AE           JSR L294         ;CHECK IF BRACKETS CLOSED
B3C9  A9 B2              LDA #$B2         ;CBM ASCII CODE FOR '='
B3CB  20 FF AE           JSR L242         ;CHECK FOR '='
B3CE  48                 PHA              ;PUSH FIRST CHARACTER TOKEN TO STACK
B3CF  A5 48              LDA VARPNT+1     ;GET VARIABLE ADDRESS POINTER
B3D1  48                 PHA              ;MSB AND PUSH TO STACK
B3D2  A5 47              LDA VARPNT       ;GET VARIABLE ADDRESS POINTER
B3D4  48                 PHA              ;LSB AND PUSH TO STACK
B3D5  A5 7B              LDA TXTPTR+1     ;GET CHARGET POINTER MSB
B3D7  48                 PHA              ;AND PUSH TO STACK
B3D8  A5 7A              LDA TXTPTR       ;GET CHARGET POINTER LSB
B3DA  48                 PHA              ;AND PUSH TO STACK
B3DB  20 F8 A8           JSR L130         ;POINT TO NEXT PROGRAM STATEMENT
```

```
LOC   CODE        LINE

B3DE  4C 4F B4          JMP L393        ;GET FN VARIABLE FROM STACK
B3E1                ;***************************************
B3E1                ;*CHECK FN SYNTAX.
B3E1                ;*THIS IS CALLED BY THE ROUTINES AT $B3B3
B3E1                ;*AND $B3F4 AND FIRST CHECKS FOR THE FN TOKEN
B3E1                ;* - $A5. THEN SETS THE FUNCTION FLAG IN
B3E1                ;*LOCATION $10 WITH THE 'OR' OF THE FUNCTION
B3E1                ;*NAME AND $80. IF THE FUNCTION EXISTS IT IS
B3E1                ;*SEARCHED FOR, IF NOT THEN IT IS SET UP, AND
B3E1                ;*FINALLY CHECKS THAT THE VALUE IS NUMERIC.
B3E1                ;***************************************
B3E1  A9 A5         L390    LDA #$A5        ;GET FN TOKEN
B3E3  20 FF AE              JSR L242        ;CHECK ON FN CODE
B3E6  09 80                 ORA #$80        ;OR WITH $30 AND
B3E8  85 10                 STA SUBFLG      ;STORE IN FN FLAG
B3EA  20 92 B0              JSR L324        ;LOCATE VARIABLE
B3ED  85 4E                 STA BASTMP+3    ;STORE FN VARIABLE POINTERS
B3EF  84 4F                 STY BASTMP+4
B3F1  4C 8D AD              JMP L96         ;CHECK IF NUMERIC
B3F4                ;***************************************
B3F4                ;*EVALUATE FNX.
B3F4                ;*THE ROUTINE AT $B3E1 IS FIRST CALLED TO
B3F4                ;*CHECK THE SYNTAX AND GET THE VARIABLE
B3F4                ;*ADDRESS. THE EXPRESSION IS EVALUATED AND
B3F4                ;*THE RESULT STORED IN FAC #1. THE DATA
B3F4                ;*PLACED ON THE STACK BY THE 'DEF' ROUTINE
B3F4                ;*IS RECOVERED AND STORED IN RAM MEMORY AT
B3F4                ;*A LOCATION POINTED TO BY THE VALUES IN
B3F4                ;*LOCATIONS $4E,$4F.
B3F4                ;***************************************
B3F4  20 E1 B3      L387    JSR L390        ;CHECK FN SYNTAX
B3F7  A5 4F                 LDA BASTMP+4    ;GET FN VARIABLE POINTER MSB AND
B3F9  48                    PHA             ;PUSH TO STACK
B3FA  A5 4E                 LDA BASTMP+3    ;GET LSB
B3FC  48                    PHA
B3FD  20 F1 AE              JSR L292        ;EVALUATE EXPRESSION WITHIN BRACKETS
B400  20 8D AD              JSR L96         ;CHECK IF NUMERIC
B403  68                    PLA             ;RESTORE FN VARIABLE POINTERS
B404  85 4E                 STA BASTMP+3    ;LSB
B406  68                    PLA
B407  85 4F                 STA BASTMP+4    ;MSB
B409  A0 02                 LDY #$02        ;SET POINTER TO SECOND BYTE AND
B40B  B1 4E                 LDA (BASTMP+3),Y ;GET VARIABLE ADDRESS LSB
B40D  85 47                 STA VARPNT      ;STORE IN MEMORY AND
B40F  AA                    TAX             ;PUT IN .X REGISTER
B410  C8                    INY
B411  B1 4E                 LDA (BASTMP+3),Y ;GET VARIABLE ADDRESS MSB
B413  F0 99                 BEQ L208        ;GIVE 'UNDEF'D FUNCTION' ERROR
B415  85 48                 STA VARPNT+1    ;STORE
B417  C8                    INY             ;SET TO THREE BYTES
B418  B1 47         L293    LDA (VARPNT),Y  ;GET VARIABLE
B41A  48                    PHA             ;AND PUSH TO STACK
B41B  88                    DEY             ;POINT TO NEXT
B41C  10 FA                 BPL L293        ;IF MORE BYTES DO AGAIN
B41E  A4 48                 LDY VARPNT+1    ;PUT VARIABLE ADDRESS MSB INTO .Y
B420  20 D4 BB              JSR L154        ;LSB IN .X PUT FAC#1 INTO FN VARIABLE
B423  A5 7B                 LDA TXTPTR+1    ;GET CHARGET POINTER MSB
B425  48                    PHA             ;PUSH TO STACK
B426  A5 7A                 LDA TXTPTR      ;REPEAT FOR LSB
B428  48                    PHA
B429  B1 4E                 LDA (BASTMP+3),Y ;GET NEW CHARGET POINTER
```

```
LOC   CODE        LINE

B42B  85 7A             STA TXTPTR      ;AND REPLACE OLD LSB
B42D  C8                INY             ;DO AGAIN FOR MSB
B42E  B1 4E             LDA (BASTMP+3),Y
B430  85 7B             STA TXTPTR+1
B432  A5 48             LDA VARPNT+1    ;GET CURRENT VARIABLE ADDRESS MSB
B434  48                PHA             ;PUSH TO STACK
B435  A5 47             LDA VARPNT      ;REPEAT FOR LSB
B437  48                PHA
B438  20 8A AD          JSR L253        ;GET TERM AND CHECK NUMERIC
B43B  68                PLA             ;PULL CURRENT VARIABLE ADDRESS LSB
B43C  85 4E             STA BASTMP+3    ;FROM STACK AND STORE
B43E  68                PLA             ;REPEAT FOR MSB
B43F  85 4F             STA BASTMP+4
B441  20 79 00          JSR CHRGOT      ;GET LAST CHARACTER
B444  F0 03             BEQ L391        ;NO MORE CHARACTERS
B446  4C 08 AF          JMP L94         ;OUTPUT 'SYNTAX ERROR'
B449  68          L391  PLA              ;RESTORE CHARGET PROGRAM POINTER LSB
B44A  85 7A             STA TXTPTR      ;AND STORE IN CHARGET
B44C  68                PLA             ;REPEAT FOR MSB
B44D  85 7B             STA TXTPTR+1
B44F  A0 00       L393  LDY #$00         ;SET POINTER TO ZERO
B451  68                PLA             ;PULL FN VARIABLE OFF STACK AND STORE
B452  91 4E             STA (BASTMP+3),Y ;THE FIVE BYTES IN MEMORY POINTED
B454  68                PLA             ;TO BY ADDRESS IN LOCATIONS $4E,$4F
B455  C8                INY
B456  91 4E             STA (BASTMP+3),Y
B458  68                PLA
B459  C8                INY
B45A  91 4E             STA (BASTMP+3),Y
B45C  68                PLA
B45D  C8                INY
B45E  91 4E             STA (BASTMP+3),Y
B460  68                PLA
B461  C8                INY
B462  91 4E             STA (BASTMP+3),Y
B464  60                RTS
B465              ;***************************************
B465              ;*PERFORM 'STR$' OPERATION.
B465              ;*THE ROUTINE FIRST CHECKS THAT THERE IS A
B465              ;*NUMERIC EVALUATION TO THE ARGUMENT. THE
B465              ;*ARGUMENT IS STORED IN FAC#1. THIS IS CONVERTED
B465              ;*INTO AN ASCII STRING STARTING AT LOCATION
B465              ;*$0100 BY THE ROUTINE AT $BDDF. THE STRING
B465              ;*AND ITS RELATED POINTERS ARE THEN SET UP IN
B465              ;*MEMORY BY THE ROUTINE $B487.
B465              ;***************************************
B465  20 8D AD    STR   JSR L96          ;CHECK ON NUMERIC
B468  A0 00             LDY #$00
B46A  20 DF BD          JSR L187        ;CONVERT FAC#1
B46D  68                PLA
B46E  68                PLA
B46F  A9 FF       L388  LDA #$FF         ;SET POINTER TO START OF STRING
B471  A0 00             LDY #$00        ;STORAGE BUFFER TO $00FF
B473  F0 12             BEQ L440        ;SET UP STRING
B475              ;***************************************
B475              ;*CALCULATE STRING VECTOR.
B475              ;*ON ENTRY THE STRING LENGTH IS HELD IN THE
B475              ;*ACCUMULATOR, THE ROUTINE THEN CALCULATES
B475              ;*THE AREA OF RAM ALLOCATED TO THE STRING.
B475              ;*ON EXIT THE STRING LENGTH IS HELD IN .A AND
B475              ;*LOCATION $61 AND THE ADDRESS POINTER IN .X
```

```
LOC   CODE        LINE

B475              ;*AND $62 (LSB), .Y AND $63 (MSB).
B475              ;**************************************
B475  A6 64       L304   LDX FACHO+2     ;TRANSFER CONTENTS OF LOCATIONS
B477  A4 65              LDY FACHO+3     ;$64 AND $65 TO THE POINTER TO
B479  86 50              STX BASTMP+5    ;STRING DESCRIPTOR LOCATIONS
B47B  84 51              STY BASTMP+6
B47D  20 F4 B4    L173   JSR L404        ;MAKE ROOM FOR STRING
B480  86 62              STX FACHO       ;SAVE ADDRESS POINTER LSB
B482  84 63              STY FACHO+1     ;MSB
B484  85 61              STA FACEXP      ;SAVE LENGTH OF STRING
B486  60                 RTS
B487              ;**************************************
B487              ;*SCAN AND SET UP STRING.
B487              ;*THIS ROUTINE CREATES SPACE AT THE TOP OF
B487              ;*RAM FOR A STRING, PUTS IT THERE AND SETS
B487              ;*THE POINTERS. ON ENTRY THE STARTING LOCATION
B487              ;*OF THE STRING IS STORED IN THE ACCUMULATOR
B487              ;*(LSB) AND .Y (MSB). THIS STARTING ADDRESS
B487              ;*COULD BE THE INPUT BUFFER AT $0100 IN WHICH
B487              ;*CASE IT WOULD HAVE A ZERO TERMINATOR, OR A
B487              ;*STRING WITHIN QUOTES IN A BASIC PROGRAM.
B487              ;*THE STRING ORIGIN IS DETERMINED BY THE
B487              ;*FLAGS IN LOCATIONS $07,$08. ON EXIT THE
B487              ;*STRING LENGTH IS STORED IN $61, AND THE
B487              ;*ADDRESS POINTER IN $62 (LSB) AND $63 (MSB).
B487              ;**************************************
B487  A2 22       L440   LDX #$22        ;INITIALISE QUOTE FLAGS
B489  86 07              STX CHARAC
B48B  86 08              STX ENDCHR
B48D  85 6F       L188   STA ARISGN      ;SAVE START ADDRESS OF STRING LSB
B48F  84 70              STY FACOV       ;MSB
B491  85 62              STA FACHO       ;SAVE START ADDRESS OF STRING LSB
B493  84 63              STY FACHO+1     ;MSB
B495  A0 FF              LDY #$FF        ;SET INDEX TO START OF STRING -1
B497  C8          L233   INY
B498  B1 6F              LDA (ARISGN),Y  ;NEXT CHARACTER OF STRING
B49A  F0 0C              BEQ L397        ;IF CHARACTER IS NULL, END OF LINE
B49C  C5 07              CMP CHARAC      ;COMPARE WITH CONTENTS OF $07
B49E  F0 04              BEQ L398        ;SAME?
B4A0  C5 08              CMP ENDCHR      ;COMPARE WITH CONTENTS OF $08
B4A2  D0 F3              BNE L233        ;NOT SAME THEN DO NEXT CHARACTER
B4A4  C9 22       L398   CMP #$22        ;COMPARE WITH QUOTES CHARACTER
B4A6  F0 01              BEQ L396        ;IF QUOTES THEN JUMP
B4A8  18          L397   CLC
B4A9  84 61       L396   STY FACEXP      ;SAVE STRING LENGTH
B4AB  98                 TYA
B4AC  65 6F              ADC ARISGN      ;ADD STRING LENGTH TO START ADDRESS
B4AE  85 71              STA FBUFPT      ;AND SAVE IN END OF STRING ADDRESS LO
B4B0  A6 70              LDX FACOV       ;GET STRING START MSB
B4B2  90 01              BCC L399
B4B4  E8                 INX
B4B5  86 72       L399   STX FBUFPT+1    ;SAVE END OF STRING ADDRESS HIGH
B4B7  A5 70              LDA FACOV       ;GET START ADDRESS HIGH
B4B9  F0 04              BEQ L400        ;BRANCH IF ZERO (NO STRING)
B4BB  C9 02              CMP #$02        ;COMPARE WITH 2 (IN INPUT BUFFER)
B4BD  D0 0B              BNE L401        ;STRING IN MEMORY
B4BF  98          L400   TYA
B4C0  20 75 B4           JSR L304        ;CALCULATE STRING VECTOR
B4C3  A6 6F              LDX ARISGN      ;GET STRING START LSB
B4C5  A4 70              LDY FACOV       ;GET STRING START MSB
B4C7  20 88 B6           JSR L174        ;TRANSFER STRING
```

```
LOC   CODE        LINE

B4CA  A6 16       L401   LDX TEMPPT      ;GET TEMPORARY STRING STACK POINTER
B4CC  E0 22              CPX #$22        ;STRING STACK FULL
B4CE  D0 05              BNE L408        ;NO
B4D0  A2 19              LDX #$19        ;# FOR 'FORMULA TOO COMPLEX'
B4D2  4C 37 A4    L402   JMP L10          ;OUTPUT ERROR MESSAGE
B4D5  A5 61       L408   LDA FACEXP       ;GET STRING LENGTH
B4D7  95 00              STA $00,X       ;AND STORE IN STRING STACK
B4D9  A5 62              LDA FACHO       ;GET STRING ADDRESS LSB
B4DB  95 01              STA $01,X       ;SAVE TO STRING STACK
B4DD  A5 63              LDA FACHO+1     ;GET STRING ADDRESS LSB
B4DF  95 02              STA $02,X       ;SAVE TO STRING STACK
B4E1  A0 00              LDY #$00
B4E3  86 64              STX FACHO+2     ;SAVE STRING STACK POINTER
B4E5  84 65              STY FACHO+3     ;CLEAR LOCATION
B4E7  84 70              STY FACOV       ;CLEAR LOCATION
B4E9  88                 DEY             ;SET .Y TO $FF
B4EA  84 0D              STY VALTYP      ;SET VARIABLE TYPE FLAG TO STRING
B4EC  86 17              STX LASTPT      ;SET LAST TEMP STRING VECTOR LSB
B4EE  E8                 INX             ;INCREASE INDEX BY 3
B4EF  E8                 INX
B4F0  E8                 INX
B4F1  86 16              STX TEMPPT      ;SAVE AS NEW INDEX
B4F3  60                 RTS
B4F4                ;***************************************
B4F4                ;*ALLOCATE SPACE FOR STRING.
B4F4                ;*ON ENTRY, THE LENGTH OF THE STRING IS
B4F4                ;*HELD IN THE ACCUMULATOR. THE STRING
B4F4                ;*POINTER TO THE BOTTOM OF THE STRING
B4F4                ;*STORAGE AREA IS THEN DECREMENTED BY THIS
B4F4                ;*AMOUNT AND THE RESULT STORED, LOW ORDER
B4F4                ;*BYTE IN $33 AND $35, HIGH ORDER BYTE IN
B4F4                ;*$34 AND $36. IF THE RESULT IS LESS THAN
B4F4                ;*THE CONTENTS OF THE END OF ARRAY POINTER
B4F4                ;*IN $31,$32 THEN THE 'GARBAGE COLLECT'
B4F4                ;*ROUTINE IS CALLED.
B4F4                ;***************************************
B4F4  46 0F       L404   LSR GARBFL       ;SET MEMORY FLAG FOR GARBAGE COLLECT
B4F6  48          L395   PHA              ;SAVE LENGTH
B4F7  49 FF              EOR #$FF        ;'NOT' THE VALUE FOR ADDITION TO
B4F9  38                 SEC             ;THE STRING POINTER (SUBTRACT)
B4FA  65 33              ADC FRETOP      ;STORE LO BYTE
B4FC  A4 34              LDY FRETOP+1    ;GET HIGH BYTE OF POINTER
B4FE  B0 01              BCS L409        ;NO CARRY FROM LO BYTE CALCULATION
B500  88                 DEY             ;ELSE DECREMENT HI BYTE
B501  C4 32       L409   CPY STREND+1     ;HAVE WE REACHED END OF ARRAYS?
B503  90 11              BCC L407        ;YES, GARBAGE COLLECT
B505  D0 04              BNE L405        ;NOT EQUAL, STILL ENOUGH MEMORY
B507  C5 31              CMP STREND      ;END OF ARRAYS?
B509  90 0B              BCC L407        ;YES, GARBAGE COLLECT
B50B  85 33       L405   STA FRETOP      ;SAVE LO BYTE
B50D  84 34              STY FRETOP+1    ;SAVE HI BYTE
B50F  85 35              STA FREPSC      ;SAVE LO BYTE (UTILITY STR POINTER)
B511  84 36              STY FREPSC+1    ;SAVE HI BYTE (UTILITY STR POINTER)
B513  AA                 TAX
B514  68                 PLA
B515  60                 RTS             ;ALLOCATED O.K.
B516                ;
B516                ;DO GARBAGE COLLECT AND REPEAT ALLOCATE IF
B516                ;GARBAGE COLLECT NOT ALREADY DONE, ELSE
B516                ;SEND 'OUT OF MEMORY ERROR'
B516                ;
```

```
LOC   CODE        LINE

B516  A2 10       L407   LDX #$10        ;'OUT OF MEMORY'
B518  A5 0F              LDA GARBFL      ;GET MEMORY FLAG FOR GARBAGE COLLECT
B51A  30 B6              BMI L402        ;ALREADY DONE GARBAGE COLLECT, ERROR
B51C  20 26 B5           JSR L406        ;GARBAGE COLLECT
B51F  A9 80              LDA #$80        ;SET MEMORY FLAG TO SAY THAT
B521  85 0F              STA GARBFL      ;A GARBAGE COLLECT HAS BEEN DONE
B523  68                 PLA             ;RESTORE STRING LENGTH
B524  D0 D0              BNE L395        ;TRY TO ALLOCATE AGAIN
B526                     .END
B526                     .LIB B10
B526                ;**************************************
B526                ;*GARBAGE COLLECTION.
B526                ;*GARBAGE COLLECT IS CALLED WHEN THERE IS
B526                ;*INSUFFICIENT FREE RAM SPACE TO STORE NEW
B526                ;*VARIABLES, EITHER STRING, NUMERIC, OR
B526                ;*ARRAYS. IT SOLVES THIS PROBLEM BY CLEARING
B526                ;*ALL UNUSED STRINGS AT THE TOP OF MEMORY.
B526                ;**************************************
B526  A6 37       L406   LDX MEMTOP       ;GET TOP OF MEMORY POINTER LO
B528  A5 38              LDA MEMTOP+1    ;AND HI
B52A  86 33       L14    STX FRETOP       ;STORE IN STRING POINTER LO
B52C  85 34              STA FRETOP+1    ;AND HI
B52E  A0 00              LDY #$00
B530  84 4F              STY BASTMP+4    ;SET TEMPORARY POINTER TO ZERO
B532  84 4E              STY BASTMP+3
B534  A5 31              LDA STREND      ;GET END OF ARRAYS POINTER LO
B536  A6 32              LDX STREND+1    ;AND HI
B538  85 5F              STA FACEXP-2    ;STORE IN TEMPORARY POINTER
B53A  86 60              STX FACEXP-1
B53C  A9 19              LDA #$19        ;STACK FOR TEMPORARY STRINGS AT
B53E  A2 00              LDX #$00        ;$0019
B540  85 22              STA INDEX       ;INTO UTILITY POINTER
B542  86 23              STX INDEX+1
B544  C5 16       L429   CMP TEMPPT       ;ROOM IN STACK?
B546  F0 05              BEQ L412        ;NO
B548  20 C7 B5           JSR L415
B54B  F0 F7              BEQ L429
B54D  A9 07       L412   LDA #$07
B54F  85 53              STA BASTMP+8
B551  A5 2D              LDA VARTAB      ;GET START OF VARIABLES LSB
B553  A6 2E              LDX VARTAB+1    ;MSB
B555  85 22              STA INDEX       ;STORE IN POINTER TO UTILITY AREA LSB
B557  86 23              STX INDEX+1     ;MSB
B559  E4 30       L410   CPX ARYTAB+1     ;REACHED START OF ARRAYS?
B55B  D0 04              BNE L416        ;NO
B55D  C5 2F              CMP ARYTAB
B55F  F0 05              BEQ L413        ;YES
B561  20 BD B5    L416   JSR L421         ;CHECK SALVAGEABILITY
B564  F0 F3              BEQ L410
B566  85 58       L413   STA BASTMP+13
B568  86 59              STX BASTMP+14
B56A  A9 03              LDA #$03
B56C  85 53              STA BASTMP+8
B56E  A5 58       L414   LDA BASTMP+13    ;GET TEMP ADDRESS LSB
B570  A6 59              LDX BASTMP+14   ;MSB
B572  E4 32       L419   CPX STREND+1     ;REACHED END OF ARRAYS?
B574  D0 07              BNE L422        ;NO
B576  C5 31              CMP STREND
B578  D0 03              BNE L422        ;NO
B57A  4C 06 B6           JMP L427        ;DONE
B57D  85 22       L422   STA INDEX        :STORE IN UTILITY POINTER LSB
```

```
LOC   CODE        LINE

B57F  86 23             STX INDEX+1     ;MSB
B581  A0 00             LDY #$00
B583  B1 22             LDA (INDEX),Y   ;GET NAME LSB
B585  AA                TAX             ;TO .X
B586  C8                INY
B587  B1 22             LDA (INDEX),Y   ;GET NAME MSB
B589  08                PHP             ;SAVE STATUS
B58A  C8                INY
B58B  B1 22             LDA (INDEX),Y   ;ADD LENGTH OF ARRAY TO
B58D  65 58             ADC BASTMP+13   ;POINTER IN $58,$59
B58F  85 58             STA BASTMP+13
B591  C8                INY
B592  B1 22             LDA (INDEX),Y
B594  65 59             ADC BASTMP+14
B596  85 59             STA BASTMP+14
B598  28                PLP             ;GET STATUS AFTER SECOND BYTE OF NAME
B599  10 03             BPL L414        ;<128, NOT STRING ARRAY
B59B  8A                TXA             ;GET NAME LSB
B59C  30 D0             BMI L414        ;>128, NOT STRING ARRAY
B59E  C8                INY
B59F  B1 22             LDA (INDEX),Y   ;GET NUMBER OF DIMENSIONS
B5A1  A0 00             LDY #$00
B5A3  0A                ASL A           ;TIMES 2
B5A4  69 05             ADC #$05        ;FIND START OF STRING POINTERS
B5A6  65 22             ADC INDEX       ;AFTER THE DIMENSIONS
B5A8  85 22             STA INDEX       ;IN $22,$23
B5AA  90 02             BCC L417
B5AC  E6 23             INC INDEX+1
B5AE  A6 23       L417  LDX INDEX+1      ;REACHED END OF ARRAY?
B5B0  E4 59       L420  CPX BASTMP+14
B5B2  D0 04             BNE L423        ;NO
B5B4  C5 58             CMP BASTMP+13
B5B6  F0 BA             BEQ L419        ;YES, DO NEXT ARRAY
B5B8  20 C7 B5    L423  JSR L415         ;GET STRING ENTRY
B5BB  F0 F3             BEQ L420        ;ALWAYS
B5BD              ;
B5BD              ;GET A STRING ENTRY FROM THE VARIABLES
B5BD              ;
B5BD  B1 22       L421  LDA (INDEX),Y    ;GET NAME LSB
B5BF  30 35             BMI L426        ;NOT STRING
B5C1  C8                INY
B5C2  B1 22             LDA (INDEX),Y   ;GET NAME MSB
B5C4  10 30             BPL L426        ;NOT STRING
B5C6  C8                INY
B5C7  B1 22       L415  LDA (INDEX),Y    ;GET STRING LENGTH
B5C9  F0 2B             BEQ L426        ;ZERO LENGTH
B5CB  C8                INY
B5CC  B1 22             LDA (INDEX),Y   ;GET STRING ADDRESS LSB
B5CE  AA                TAX             ;INTO .X
B5CF  C8                INY
B5D0  B1 22             LDA (INDEX),Y   ;GET STRING ADDRESS MSB
B5D2  C5 34             CMP FRETOP+1    ;EQUAL TO STRING POINTER?
B5D4  90 06             BCC L411        ;NO, LOWER
B5D6  D0 1E             BNE L426        ;NO, MUST BE HIGHER
B5D8  E4 33             CPX FRETOP
B5DA  B0 1A             BCS L426        ;IS HIGHER OR EQUAL TO STRING POINTER
B5DC  C5 60       L411  CMP FACEXP-1     ;EQUAL TO TEMP POINTER?
B5DE  90 16             BCC L426        ;NO, IS LOWER
B5E0  D0 04             BNE L425        ;NO, MUST BE HIGHER
B5E2  E4 5F             CPX FACEXP-2
B5E4  90 10             BCC L426        ;IS LOWER THAN TEMP POINTER
```

```
LOC   CODE        LINE

B5E6  86 5F       L425   STX FACEXP-2    ;TEMP POINTER = ADDRESS OF STRING
B5E8  85 60              STA FACEXP-1
B5EA  A5 22              LDA INDEX       ;GET UTILITY POINTER LSB
B5EC  A6 23              LDX INDEX+1     ;MSB
B5EE  85 4E              STA BASTMP+3    ;STORE IN ANOTHER UTILITY POINTER LSB
B5F0  86 4F              STX BASTMP+4    ;MSB
B5F2  A5 53              LDA BASTMP+8    ;GET ENTRY LENGTH
B5F4  85 55              STA BASTMP+10   ;STORE IT
B5F6  A5 53       L426   LDA BASTMP+8     ;GET LENGTH
B5F8  18                 CLC
B5F9  65 22              ADC INDEX       ;ADD TO UTILITY POINTER
B5FB  85 22              STA INDEX
B5FD  90 02              BCC L424
B5FF  E6 23              INC INDEX+1
B601  A6 23       L424   LDX INDEX+1      ;GET UTILITY POINTER MSB
B603  A0 00              LDY #$00
B605  60                 RTS             ;RETURN FROM GET ENTRY
B606  A5 4F       L427   LDA BASTMP+4     ;DOES (BASTMP+3)=$0000?
B608  05 4E              ORA BASTMP+3
B60A  F0 F5              BEQ L424        ;YES, ALL DONE
B60C  A5 55              LDA BASTMP+10   ;GET TEMP
B60E  29 04              AND #$04
B610  4A                 LSR A
B611  A8                 TAY
B612  85 55              STA BASTMP+10
B614  B1 4E              LDA (BASTMP+3),Y
B616  65 5F              ADC FACEXP-2
B618  85 5A              STA BASTMP+15
B61A  A5 60              LDA FACEXP-1
B61C  69 00              ADC #$00
B61E  85 5B              STA BASTMP+16
B620  A5 33              LDA FRETOP      ;GET STRING POINTER LSB
B622  A6 34              LDX FRETOP+1    ;MSB
B624  85 58              STA BASTMP+13   ;STORE IN A TEMP LOCATION LSB
B626  86 59              STX BASTMP+14   ;MSB
B628  20 BF A3           JSR L39         ;MOVE MEMORY FROM ($5A) TO ($58)
B62B  A4 55              LDY BASTMP+10
B62D  C8                 INY
B62E  A5 58              LDA BASTMP+13
B630  91 4E              STA (BASTMP+3),Y
B632  AA                 TAX
B633  E6 59              INC BASTMP+14
B635  A5 59              LDA BASTMP+14
B637  C8                 INY
B638  91 4E              STA (BASTMP+3),Y
B63A  4C 2A B5           JMP L14         ;REPEAT UNTIL DONE
B63D              ;***************************************
B63D              ;*CONCATENATE STRING.
B63D              ;*ON ENTRY THE POINTER TO THE ADDRESS OF THE
B63D              ;*FIRST STRING IS STORED IN LOCATIONS $64,$65,
B63D              ;*THAT OF THE SECOND STRING IS OBTAINED BY
B63D              ;*THE EVALUATION ROUTINE. THE LENGTH OF THE
B63D              ;*TWO STRINGS IS OBTAINED AND ADDED WITH A
B63D              ;*CHECK TO ENSURE THAT IT IS NOT TOO LONG.
B63D              ;*FROM THIS DATA A NEW STRING IS CREATED AND
B63D              ;*STORED AT THE BOTTOM OF THE STRING STORAGE
B63D              ;*AREA, ITS POINTERS ARE STORED IN THE VARIABLE
B63D              ;*STORAGE AREA AND IN THE RELEVANT ZERO PAGE
B63D              ;*LOCATIONS.
B63D              ;***************************************
B63D  A5 65       L418   LDA FACHO+3      ;GET ADDRESS OF FIRST STRING MSB
```

```
LOC   CODE        LINE

B63F  48                  PHA             ;PUSH IT AWAY
B640  A5 64               LDA FACHO+2     ;LSB
B642  48                  PHA             ;PUSH IT AWAY
B643  20 83 AE            JSR L277        ;GET ADDRESS OF SECOND STRING
B646  20 8F AD            JSR L95         ;CHECK ON STRING TYPE
B649  68                  PLA             ;GET FIRST STRING ADDRESS LSB
B64A  85 6F               STA ARISGN
B64C  68                  PLA             ;MSB
B64D  85 70               STA FACOV
B64F  A0 00               LDY #$00
B651  B1 6F               LDA (ARISGN),Y ;GET LENGTH OF FIRST STRING
B653  18                  CLC
B654  71 64               ADC (FACHO+2),Y ;ADD TO LENGTH OF SECOND STRING
B656  90 05               BCC L266        ;<256
B658  A2 17               LDX #$17        ;'STRING TOO LONG'
B65A  4C 37 A4            JMP L10         ;SEND ERROR
B65D  20 75 B4    L266    JSR L304         ;GET ADDRESS FOR NEW STRING
B660  20 7A B6            JSR L430        ;TRANSFER FIRST STRING INTO MEMORY
B663  A5 50               LDA BASTMP+5    ;GET POINTER TO SECOND STRING LSB
B665  A4 51               LDY BASTMP+6    ;MSB
B667  20 AA B6            JSR L156        ;DISCARD UNWANTED STRING
B66A  20 8C B6            JSR L403        ;STORE SECOND STRING AT END OF FIRST
B66D  A5 6F               LDA ARISGN      ;GET ADDRESS OF FIRST STRING LSB
B66F  A4 70               LDY FACOV       ;MSB
B671  20 AA B6            JSR L156        ;DISCARD UNWANTED STRING
B674  20 CA B4            JSR L401        ;PUT NEW STRING DECRIPTOR IN STACK
B677  4C B8 AD            JMP L276        ;BACK TO EVALUATION ROUTINE
B67A              ;****************************************
B67A              ;*STORE STRING.
B67A              ;*THERE ARE TWO ENTRY POINTS TO THIS ROUTINE,
B67A              ;*BOTH STORE STRING AT THE TOP OF MEMORY
B67A              ;*STRING STORAGE AREA, THE CHOICE OF WHICH
B67A              ;*ENTRY POINT TO USE DEPENDS ON THE INPUT
B67A              ;*VARIABLES:
B67A              ;*     $B67A  THE VARIABLE INPUT IS STORED IN
B67A              ;*            $6F,$70 AND POINTS TO THE BYTE
B67A              ;*            AFTER THE VARIABLE NAME.
B67A              ;*     $B688  THE ACCUMULATOR HOLDS THE STRING
B67A              ;*            LENGTH, THE .X, .Y REGISTERS HOLD
B67A              ;*            THE ADDRESS OF THE STRING.
B67A              ;****************************************
B67A  A0 00       L430    LDY #$00
B67C  B1 6F               LDA (ARISGN),Y ;GET STRING LENGTH
B67E  48                  PHA             ;PUSH IT TO STACK
B67F  C8                  INY
B680  B1 6F               LDA (ARISGN),Y ;GET STRING ADDRESS LSB
B682  AA                  TAX             ;INTO .X
B683  C8                  INY
B684  B1 6F               LDA (ARISGN),Y ;GET STRING ADDRESS MSB
B686  A8                  TAY             ;INTO .Y
B687  68                  PLA             ;STRING LENGTH
B688  86 22       L174    STX INDEX        ;STORE IN UTILITY POINTER LSB
B68A  84 23               STY INDEX+1     ;MSB
B68C  A8          L403    TAY
B68D  F0 0A               BEQ L434        ;LENGTH=0
B68F  48                  PHA             ;STORE LENGTH
B690  88          L431    DEY
B691  B1 22               LDA (INDEX),Y  ;GET BYTE OF STRING
B693  91 35               STA (FREPSC),Y ;STORE IT
B695  98                  TYA
B696  D0 F8               BNE L431        ;REPEAT UNTIL STRING DONE
```

```
LOC   CODE        LINE

B698  68          L434    PLA             ;GET STRING LENGTH
B699  18                  CLC
B69A  65 35               ADC FREPSC      ;ADD TO UTILITY STRING POINTER
B69C  85 35               STA FREPSC
B69E  90 02               BCC L433
B6A0  E6 36               INC FREPSC+1
B6A2  60          L433    RTS              ;TRANSFER DONE
B6A3              ;***************************************
B6A3              ;*DISCARD UNWANTED STRING.
B6A3              ;*THIS CLEARS THE STRING POINTED TO BY
B6A3              ;*LOCATIONS $64,$65, MOVES THE BOTTOM OF
B6A3              ;*STRING POINTERS UP BY THE STRING LENGTH
B6A3              ;*SO THAT A NEW STRING WILL OVERWRITE IT.
B6A3              ;*THIS ROUTINE IS USED TO REMOVE LAST ENTERED
B6A3              ;*STRING ONLY. ON EXIT THE LOCATIONS $22,$23
B6A3              ;*POINT TO THE STRING REMOVED.
B6A3              ;***************************************
B6A3  20 8F AD    L435    JSR L95          ;CHECK ON VARIABLE TYPE
B6A6  A5 64       L451    LDA FACHO+2      ;GET STRING ENTRY LSB
B6A8  A4 65               LDY FACHO+3     ;MSB
B6AA  85 22       L156    STA INDEX        ;STORE STRING ENTRY LSB
B6AC  84 23               STY INDEX+1     ;MSB
B6AE  20 DB B6            JSR L436        ;CLEAN DESCRIPTOR STACK
B6B1  08                  PHP             ;SAVE STATUS
B6B2  A0 00               LDY #$00
B6B4  B1 22               LDA (INDEX),Y   ;GET STRING LENGTH
B6B6  48                  PHA             ;PUSH IT TO STACK
B6B7  C8                  INY
B6B8  B1 22               LDA (INDEX),Y   ;GET STRING ADDRESS LSB
B6BA  AA                  TAX             ;STORE IN .X
B6BB  C8                  INY
B6BC  B1 22               LDA (INDEX),Y   ;GET STRING ADDRESS MSB
B6BE  A8                  TAY             ;STORE IN .Y
B6BF  68                  PLA             ;GET LENGTH INTO .A
B6C0  28                  PLP             ;GET STATUS
B6C1  D0 13               BNE L437        ;NOT FROM STRING STACK
B6C3  C4 34               CPY FRETOP+1    ;EQUAL TO STRING POINTER?
B6C5  D0 0F               BNE L437        ;NO, IGNORE
B6C7  E4 33               CPX FRETOP
B6C9  D0 0B               BNE L437        ;NO, IGNORE
B6CB  48                  PHA             ;STORE LENGTH
B6CC  18                  CLC
B6CD  65 33               ADC FRETOP      ;ADD STRING POINTER
B6CF  85 33               STA FRETOP
B6D1  90 02               BCC L318
B6D3  E6 34               INC FRETOP+1
B6D5  68          L318    PLA              ;GET LENGTH
B6D6  86 22       L437    STX INDEX        ;STORE STRING ADDRESS LSB
B6D8  84 23               STY INDEX+1     ;MSB
B6DA  60                  RTS             ;DISCARD DONE
B6DB              ;***************************************
B6DB              ;*CLEAN DESCRIPTOR STACK.
B6DB              ;*THIS RESETS THE POINTERS TO THE DESCRIPTOR
B6DB              ;*STACK IN LOCATIONS $16,$17,$18.
B6DB              ;***************************************
B6DB  C4 18       L436    CPY LASTPT+1     ;IS STRING IN DESCRIPTOR STACK?
B6DD  D0 0C               BNE L175        ;NO
B6DF  C5 17               CMP LASTPT
B6E1  D0 08               BNE L175        ;NO
B6E3  85 16               STA TEMPPT      ;REMOVE STACK ENTR
B6E5  E9 03               SBC #$03
```

```
LOC   CODE        LINE

B6E7  85 17             STA LASTPT
B6E9  A0 00             LDY #$00        ;EXIT WITH Z FLAG SET
B6EB  60          L175  RTS
B6EC              ;***************************************
B6EC              ;*PERFORM CHR$.
B6EC              ;*THE SINGLE BYTE PARAMETER IS INPUT AND
B6EC              ;*EVALUATED BY THE ROUTINE AT $B7A1. A SINGLE
B6EC              ;*CHARACTER STRING SPACE IS ALLOCATED, THE
B6EC              ;*CHARACTERS STORED AND THE POINTERS SET UP.
B6EC              ;***************************************
B6EC  20 A1 B7    CHR   JSR L139          ;GET SINGLE BYTE PARAMETER
B6EF  8A                TXA
B6F0  48                PHA             ;PUSH IT TO THE STACK
B6F1  A9 01             LDA #$01
B6F3  20 7D B4          JSR L173        ;OPEN UP 1 BYTE FOR STRING
B6F6  68                PLA
B6F7  A0 00             LDY #$00        ;STORE THE BYTE IN MEMORY ALLOCATED
B6F9  91 62             STA (FACHO),Y
B6FB  68                PLA             ;REMOVE RETURN ADDRESS
B6FC  68                PLA
B6FD  4C CA B4          JMP L401        ;PUT STRING INTO STRING STACK
B700              ;***************************************
B700              ;*PERFORM LEFT$.
B700              ;*THE STRING PARAMETER DATA IS FIRST PULLED
B700              ;*FROM THE STACK BY THE ROUTINE AT $B761.
B700              ;*THE Y INDEX REGISTER CONTAINS THE STRING
B700              ;*LENGTH. THE BULK OF THIS ROUTINE FROM $B706
B700              ;*ON IS SHARED WITH THE MID$ AND RIGHT$ ROUTINES.
B700              ;*THIS INVOLVES CREATING THE SUBSTRING, STORING
B700              ;*IT IN MEMORY AND SETTING UP THE NECESSARY
B700              ;*POINTERS.
B700              ;***************************************
B700  20 61 B7    LEFT  JSR L445          ;GET STRING PARAMTERS FROM STACK
B703  D1 50             CMP (BASTMP+5),Y ;COMPARE LEN WITH LEFT$ PARAMETER
B705  98                TYA
B706  90 04       L438  BCC L444          ;LEN>LEFT$ PARAMETER
B708  B1 50             LDA (BASTMP+5),Y ;GET LENGTH OF STRING
B70A  AA                TAX             ;TO .X
B70B  98                TYA             ;STORE ADDRESS LSB
B70C  48          L444  PHA              ;TO PROCESSOR STACK
B70D  8A          L442  TXA              ;LENGTH
B70E  48          L447  PHA              ;TO PROCESSOR STACK
B70F  20 7D B4          JSR L173        ;ALLOCATE SPACE
B712  A5 50             LDA BASTMP+5    ;GET POINTER TO DESCRIPTOR LSB
B714  A4 51             LDY BASTMP+6    ;MSB
B716  20 AA B6          JSR L156        ;DISCARD UNWANTED STRING
B719  68                PLA             ;GET ADDRESS MSB
B71A  A8                TAY             ;INTO .Y
B71B  68                PLA             ;LSB
B71C  18                CLC
B71D  65 22             ADC INDEX       ;ADD TO UTILITY POINTER
B71F  85 22             STA INDEX       ;AND STORE
B721  90 02             BCC L448
B723  E6 23             INC INDEX+1     ;INCREMENT MSB IF CARRY SET
B725  98          L448  TYA
B726  20 8C B6          JSR L403        ;MOVE STRING INTO MEMORY
B729  4C CA B4          JMP L401        ;BRING DESCRIPTOR TO STRING STACK
B72C              ;***************************************
B72C              ;*PERFORM 'RIGHT$'.
B72C              ;*PULLS PARAMETER DATA OFF THE STACK AND SETS
B72C              ;*THE STRING POSITION POINTER BEFORE JUMPING
```

```
LOC   CODE        LINE

B72C              ;*TO THE ROUTINE IN 'LEFT$' AT $B706.
B72C              ;**************************************
B72C  20 61 B7    RIGHT  JSR L445         ;GET STRING PARAMETER FROM STACK
B72F  18                 CLC
B730  F1 50              SBC (BASTMP+5),Y ;SET NEW STRING POSITION POINTER
B732  49 FF              EOR #$FF        ;TO FIRST ELEMENTS # IN OLD STRING
B734  4C 06 B7           JMP L438        ;CONTINUE AS 'LEFT $'
B737              ;**************************************
B737              ;*PERFORM 'MID$'.
B737              ;*THIS CHECKS THE SYNTAX AND PULLS THE
B737              ;*PARAMETERS FROM THE STACK BEFORE JUMPING
B737              ;*INTO THE LEFT$ ROUTINE AT $B70E.
B737              ;**************************************
B737  A9 FF       MID    LDA #$FF
B739  85 65              STA FACHO+3
B73B  20 79 00           JSR CHRGOT      ;GET LAST CHARACTER
B73E  C9 29              CMP #$29        ;IS IT ')'
B740  F0 06              BEQ L443        ;YES
B742  20 FD AE           JSR L296        ;CHECK ON COMMA
B745  20 9E B7           JSR L195        ;INPUT BYTE PARAMETER
B748  20 61 B7    L443   JSR L445         ;PULL STRING PARAMETER
B74B  F0 4B              BEQ L449        ;FIRST PARAMETER 0 -'ILLEGAL QUANTITY'
B74D  CA                 DEX
B74E  8A                 TXA
B74F  48                 PHA             ;FIRST ELEMENT NUMBER IN OLD STRING
B750  18                 CLC
B751  A2 00              LDX #$00
B753  F1 50              SBC (BASTMP+5),Y ;LENGTH OF OLD STRING
B755  B0 B6              BCS L442
B757  49 FF              EOR #$FF        ;NEW STRING LENGTH
B759  C5 65              CMP FACHO+3
B75B  90 B1              BCC L447        ;CONTINUE WITH 'LEFT$'
B75D  A5 65              LDA FACHO+3
B75F  B0 AD              BCS L447        ;CONTINUE WITH 'LEFT$'
B761              ;**************************************
B761              ;*PULL STRING DATA.
B761              ;*EACH OF THE STRING FUNCTIONS CALLS THIS
B761              ;*ROUTINE. IT FIRST CHECKS FOR A RIGHT BRACKET
B761              ;*THEN PULLS THE STRING ADDRESS POINTER FROM
B761              ;*STACK AND STORES IT IN LOCATIONS $50,$51.
B761              ;*THE STRING LENGTH IS OBTAINED FROM THE STACK
B761              ;*AND STORED IN THE ACCUMULATOR. THE CONTENTS
B761              ;*OF LOCATION $55 AND .Y ARE PUT BACK ONTO THE
B761              ;*STACK FROM WHICH THEY WERE PULLED AT THE
B761              ;*BEGINNING OF THE ROUTINE.
B761              ;**************************************
B761  20 F7 AE    L445   JSR L294         ;CHECK BRACKETS CLOSED
B764  68                 PLA
B765  A8                 TAY             ;GET CALL ADDRESS OFF STACK
B766  68                 PLA
B767  85 55              STA BASTMP+10
B769  68                 PLA
B76A  68                 PLA
B76B  68                 PLA
B76C  AA                 TAX             ;GET FIRST PARAMETER
B76D  68                 PLA
B76E  85 50              STA BASTMP+5    ;ADDRESS OF STRING DESCRIPTOR LSB
B770  68                 PLA
B771  85 51              STA BASTMP+6    ;MSB
B773  A5 55              LDA BASTMP+10   ;RESTORE CALL ADDRESS
B775  48                 PHA
```

```
LOC   CODE        LINE

B776  98                TYA
B777  48                PHA
B778  A0 00             LDY #$00        ;SET .Y TO ZERO
B77A  8A                TXA             ;AND PUT LENGTH IN ACCUMULATOR
B77B  60                RTS
B77C                  ;***************************************
B77C                  ;*PERFORM 'LEN'.
B77C                  ;*CALLS THE ROUTINE AT $B782 TO OBTAIN THE
B77C                  ;*STRING LENGTH WHICH IS RETURNED IN .A AND
B77C                  ;*.Y. IT THEN JUMPS TO $B3A7 WHERE THE CONTENTS
B77C                  ;*OF .A AND .Y ARE CONVERTED TO A FLOATING
B77C                  ;*POINT VALUE IN FAC#1.
B77C                  ;***************************************
B77C  20 82 B7     LEN    JSR L441         ;GET STRING LENGTH
B77F  4C A2 B3            JMP L291         ;PUT IN FAC#1
B782                  ;***************************************
B782                  ;*SWITCH STRING TO NUMERIC.
B782                  ;*THIS CHECKS FOR A STRING, AND SETS UP THE
B782                  ;*POINTERS USING THE ROUTINE AT $B6A3, SETS
B782                  ;*THE FLAG IN LOCATIONS $0D TO ZERO AND LOADS
B782                  ;*THE ACCUMULATOR AND Y INDEX REGISTERS WITH
B782                  ;*THE STRING LENGTH.
B782                  ;***************************************
B782  20 A3 B6     L441   JSR L435         ;GET STRING LENGTH IN .A
B785  A2 00              LDX #$00
B787  86 0D              STX VALTYP       ;SET TYPE FLAG TO NUMERIC
B789  A8                 TAY              ;LENGTH IN .Y
B78A  60                 RTS
B78B                  ;***************************************
B78B                  ;*PERFORM 'ASC'.
B78B                  ;*THE ROUTINE AT $B782 IS FIRST CALLED AND
B78B                  ;*ANY STRING OF LENGTH ZERO REJECTED WITH AN
B78B                  ;*'ILLEGAL QUANTITY' ERROR. THE .Y INDEX
B78B                  ;*REGISTER IS LOADED WITH THE CONTENTS OF THE
B78B                  ;*LOCATION POINTED TO BY $22 WHICH IS THEN
B78B                  ;*CONVERTED INTO A FLOATING POINT NUMBER AND
B78B                  ;*STORED IN FAC#1.
B78B                  ;***************************************
B78B  20 82 B7     ASC    JSR L441         ;GET STRING POINTER INTO $22,$23
B78E  F0 08               BEQ L449        ;IF ZERO, 'ILLEGAL QUANTITY'
B790  A0 00               LDY #$00
B792  B1 22               LDA (INDEX),Y   ;GET FIRST CHARACTER
B794  A8                  TAY             ;PUT ASCII VALUE IN .Y AND
B795  4C A2 B3            JMP L291        ;CONVERT TO FLOATING POINT VALUE
B798  4C 48 B2     L449   JMP L363         ;'ILLEGAL QUANTITY' ERROR
B79B                  ;***************************************
B79B                  ;*GET SINGLE BYTE PARAMETER.
B79B                  ;*THE EXPRESSION POINTED TO BY CHARGET $7A,
B79B                  ;*$7B IS EVALUATED AND ITS RANGE CHECKED. THE
B79B                  ;*RESULT IS STORED IN .X AND $65.
B79B                  ;***************************************
B79B  20 73 00     L446   JSR CHRGET       ;GET NEXT CHARACTER
B79E  20 8A AD     L195   JSR L253         ;PUT VALUE INTO FAC#1
B7A1  20 B8 B1     L139   JSR L353         ;CONVERT TO INTEGER
B7A4  A6 64               LDX FACHO+2     ;INTEGER HIGH BYTE
B7A6  D0 F0               BNE L449        ;IF NOT ZERO THEN 'ILLEGAL QUANTITY'
B7A8  A6 65               LDX FACHO+3     ;PUT INTEGER LOW BYTE IN .X
B7AA  4C 79 00            JMP CHRGOT      ;GET LAST CHARACTER
B7AD                  ;***************************************
B7AD                  ;*PERFORM 'VAL'.
B7AD                  ;*THE STRING POINTED TO BY CHARGET IS LOCATED
```

```
LOC   CODE        LINE

B7AD              ;*AND CONVERTED TO A FLOATING POINT NUMBER
B7AD              ;*BY THE ROUTINE AT $BCF3, THE RESULT IS
B7AD              ;*STORED IN FAC#1.
B7AD              ;**************************************
B7AD  20 82 B7    VAL     JSR L441        ;GET STRING LENGTH AND ADDRESS
B7B0  D0 03               BNE L439        ;IF LENGTH GREATER THAN ZERO
B7B2  4C F7 B8            JMP L476        ;PUT ZERO IN FAC#1
B7B5  A6 7A       L439    LDX TXTPTR      ;GET CHARGET PROGRAM POINTERS
B7B7  A4 7B               LDY TXTPTR+1
B7B9  86 71               STX FBUFPT      ;AND SAVE
B7BB  84 72               STY FBUFPT+1
B7BD  A6 22               LDX INDEX       ;GET STRING POINTER LSB AND
B7BF  86 7A               STX TXTPTR      ;PUT IN PROGRAM POINTER LSB
B7C1  18                  CLC
B7C2  65 22               ADC INDEX       ;CALCULATE STRING END
B7C4  85 24               STA INDEX+2
B7C6  A6 23               LDX INDEX+1     ;GET STRING POINTER MSB AND
B7C8  86 7B               STX TXTPTR+1    ;PUT IN PROGRAM POINTER MSB
B7CA  90 01               BCC L453
B7CC  E8                  INX
B7CD  86 25       L453    STX INDEX+3     ;STRING END MSB
B7CF  A0 00               LDY #$00        ;INITIALISE STRING POINTER
B7D1  B1 24               LDA (INDEX+2),Y ;GET FIRST BYTE OF STRING
B7D3  48                  PHA             ;SAVE ON STACK
B7D4  98                  TYA
B7D5  91 24               STA (INDEX+2),Y ;AND CLEAR FIRST BYTE TO ZERO
B7D7  20 79 00            JSR CHRGOT      ;GET LAST CHARACTER
B7DA  20 F3 BC            JSR L532        ;CONVERT STRING TO FLOATING POINT
B7DD  68                  PLA             ;RESTORE FIRST BYTE
B7DE  A0 00               LDY #$00
B7E0  91 24               STA (INDEX+2),Y ;AND PUT BACK IN STRING
B7E2  A6 71       L455    LDX FBUFPT      ;PUT BACK CHARGET PROGRAM
B7E4  A4 72               LDY FBUFPT+1    ;POINTER VALUES
B7E6  86 7A               STX TXTPTR
B7E8  84 7B               STY TXTPTR+1
B7EA  60                  RTS
B7EB                      .END
B7EB                      .LIB B11
B7EB              ;**************************************
B7EB              ;*GET TWO PARAMETERS FOR 'POKE' OR 'WAIT'.
B7EB              ;*THE PARAMETERS FOR THESE TWO COMMANDS ARE
B7EB              ;*INPUT, CHECKED AND EVALUATED. THE FIRST
B7EB              ;*NUMERIC VALUE IS CONVERTED TO A TWO BYTE
B7EB              ;*INTEGER AND STORED IN FAC#1 BY ROUTINE
B7EB              ;*$B7F7. THE NEXT CHARACTER IS THEN CHECKED
B7EB              ;*AS A COMMA AND THE SECOND PARAMETER PLACED
B7EB              ;*IN THE .X INDEX REGISTER BY ROUTINE $B79E.
B7EB              ;**************************************
B7EB  20 8A AD    L234    JSR L253        ;GET TWO BYTE NUMERIC VALUE
B7EE  20 F7 B7            JSR L459        ;CHANGE FAC#1 TO 16 BIT VALUE
B7F1  20 FD AE    L457    JSR L296        ;CHECK FOR A COMMA
B7F4  4C 9E B7            JMP L195        ;GET SECOND PARAMETER INTO .X
B7F7              ;**************************************
B7F7              ;*CONVERT FLOATING TO FIXED POINT.
B7F7              ;*THE FLOATING POINT VALUE IN FAC#1 IS CONVERTED
B7F7              ;*TO A TWO BYTE INTEGER (PROVIDING IT IS
B7F7              ;*WITHIN THE RANGE 0-65535), THE RESULT IS
B7F7              ;*STORED IN $14 (LOW),$15 (HIGH).
B7F7              ;**************************************
B7F7  A5 66       L459    LDA FACSGN      ;GET FAC#1 SIGN
B7F9  30 9D               BMI L449        ;IF NEGATIVE 'ILLEGAL QUANTITY'
```

```
LOC   CODE        LINE

B7FB  A5 61             LDA FACEXP      ;GET EXPONENT
B7FD  C9 91             CMP #$91        ;COMPARE WITH 65536
B7FF  B0 97             BCS L449        ;LARGER THEN 'ILLEGAL QUANTITY'
B801  20 9B BC          JSR L526        ;FLOAT TO FIXED POINT CONVERSION
B804  A5 64             LDA FACHO+2     ;GET VALUE
B806  A4 65             LDY FACHO+3
B808  84 14             STY LINNUM      ;SAVE LOW
B80A  85 15             STA LINNUM+1    ;SAVE HIGH
B80C  60                RTS
B80D              ;***************************************
B80D              ;*PERFORM 'PEEK'.
B80D              ;*THE MEMORY ADDRESS PARAMETER HAS PREVIOUSLY
B80D              ;*BEEN OBTAINED USING ROUTINE $B7F7. THE
B80D              ;*PARAMETER IS THUS STORED AS A TWO BYTE
B80D              ;*INTEGER IN LOCATIONS $14,$15. THE RESULT
B80D              ;*IS PUT IN THE .Y INDEX REGISTER. THIS IS
B80D              ;*THEN CONVERTED TO FLOATING POINT FORM IN
B80D              ;*FAC#1 BY ROUTINE $B3A2.
B80D              ;***************************************
B80D  A5 15       PEEK    LDA LINNUM+1     ;GET ADDRESS MSB AND
B80F  48                  PHA              ;PUSH TO STACK
B810  A5 14               LDA LINNUM       ;REPEAT FOR LSB
B812  48                  PHA
B813  20 F7 B7            JSR L459         ;CONVERT FAC#1 TO 16 BIT VALUE
B816  A0 00               LDY #$00
B818  B1 14               LDA (LINNUM),Y ;GET PEEK VALUE
B81A  A8                  TAY              ;PUT IN .Y
B81B  68                  PLA              ;GET ADDRESS OFF STACK
B81C  85 14               STA LINNUM       ;AND SAVE
B81E  68                  PLA
B81F  85 15               STA LINNUM+1
B821  4C A2 B3            JMP L291         ;CONVERT .Y TO FLOATING POINT FORMAT
B824              ;***************************************
B824              ;*PERFORM 'POKE'.
B824              ;*USING THE ROUTINE AT $B7EB THE TWO PARAMETERS
B824              ;*ARE EVALUATED, THE SECOND OF THE TWO IS
B824              ;*STORED IN THE ACCUMULATOR. THIS VALUE IS
B824              ;*THEN STORED IN RAM AT THE ADDRESS POINTED
B824              ;*TO BY THE FIRST PARAMETER WHICH IS STORED
B824              ;*IN LOCATIONS $14,$15.
B824              ;***************************************
B824  20 EB B7    POKE    JSR L234         ;GET POKE ADDRESS AND VALUE
B827  8A                  TXA              ;PUT VALUE INTO ACCUMULATOR
B828  A0 00               LDY #$00
B82A  91 14               STA (LINNUM),Y ;AND STORE IN MEMORY
B82C  60                  RTS
B82D              ;***************************************
B82D              ;*PERFORM 'WAIT'.
B82D              ;*THE TWO PARAMETERS ARE OBTAINED USING THE
B82D              ;*ROUTINE AT $B7EB, THIS LEAVES THE ADDRESS
B82D              ;*PARAMETER IN LOCATIONS $14,$15 AND THE
B82D              ;*SECOND PARAMETER IN THE .X INDEX REGISTER.
B82D              ;*THIS SECOND PARAMETER IS STORED IN $49, THE
B82D              ;*OPTIONAL THIRD PARAMETER IS THEN OBTAINED
B82D              ;*BY ROUTINE $B7F1 AND STORED IN $4A, IF THERE
B82D              ;*IS NO THIRD PARAMETER THEN $4A IS SET TO
B82D              ;*ZERO. THE ROUTINE THEN PERFORMS A LOOP WHICH
B82D              ;*CONTINUES UNTIL THE VALUE AT THE LOCATION
B82D              ;*POINTED TO BY THE ADDRESS PARAMETER, EXCLUSIVE
B82D              ;*'OR'ED WITH THE THIRD PARAMETER AND 'AND'ED
B82D              ;*WITH THE SECOND PARAMETER IS NOT EQUAL TO
```

```
LOC   CODE        LINE

B82D              ;*ZERO.
B82D              ;***************************************
B82D  20 EB B7    WAIT    JSR L234         ;GET ADDRESS AND VALUE
B830  86 49               STX FORPNT      ;STORE SECOND PARAMETER VALUE
B832  A2 00               LDX #$00        ;SET DEFAULT FOR THIRD PARAMETER
B834  20 79 00            JSR CHRGOT      ;GET LAST CHARACTER
B837  F0 03               BEQ L456        ;NO THIRD PARAMETER
B839  20 F1 B7            JSR L457        ;CHECK COMMA AND GET 3RD PARAMETER
B83C  86 4A       L456    STX FORPNT+1     ;STORE 3RD PARAMETER
B83E  A0 00               LDY #$00
B840  B1 14       L458    LDA (LINNUM),Y  ;GET VALUE IN WAIT LOCATION
B842  45 4A               EOR FORPNT+1    ;DO LOGICAL OPERATIONS USING 2ND AND
B844  25 49               AND FORPNT      ;3RD PARAMETERS
B846  F0 F8               BEQ L458        ;IF NOT ZERO DO LOOP AGAIN
B848  60          L460    RTS
B849              ;***************************************
B849              ;*ADD 0.5
B849              ;*0.5 IN FLOATING POINT FORM IS ADDED TO THE
B849              ;*CONTENTS OF FAC#1, THIS IS USED IN ROUNDING.
B849              ;***************************************
B849  A9 11       L466    LDA #$11         ;POINTER TO CONSTANT 0.5 IN
B84B  A0 BF               LDY #$BF        ;FLOATING POINT FORMAT
B84D  4C 67 B8            JMP L469        ;FAC#1 = FAC#1 + FAC#2(.A,.Y)
B850              ;***************************************
B850              ;*PERFORM SUBTRACTION.
B850              ;*THE CONTENTS OF FAC#1 ARE SUBTRACTED FROM
B850              ;*FAC#2 AND THE RESULT STORED IN FAC#1. THERE
B850              ;*ARE TWO ENTRY POINTS TO THIS ROUTINE, $B850
B850              ;*FIRST LOADS FAC#2 WITH A FIVE BYTE VALUE
B850              ;*STORED IN MEMORY POINTED TO BY .A (LOW) AND
B850              ;*.Y (HIGH). THE OTHER ENTRY POINT $B853
B850              ;*ASSUMES THAT THE TWO VALUES ARE ALREADY
B850              ;*LOADED INTO THE TWO FLOATING ACCUMULATORS.
B850              ;*THE RESULT IS STORED IN FAC#1.
B850              ;***************************************
B850              ;
B850              ;FAC#1=CONSTANT (.A,.Y) - FAC#1
B850              ;
B850  20 8C BA    L570    JSR L492
B853              ;
B853              ;FAC#1=FAC#2 - FAC#1
B853              ;
B853  A5 66       L489    LDA FACSGN       ;GET SIGN OF FAC#1
B855  49 FF               EOR #$FF        ;CHANGE
B857  85 66               STA FACSGN      ;AND RETURN NEW SIGN OF FAC#1
B859  45 6E               EOR ARGSGN
B85B  85 6F               STA ARISGN      ;ADJUST ARGUMENT SIGN
B85D  A5 61               LDA FACEXP      ;GET FAC#1 EXPONENT
B85F  4C 6A B8            JMP L251        ;DO ADDITION FAC#1=FAC#1+ARG
B862              ;
B862              ;ADJUST EXPONENT OF FAC#1+ARGUMENT
B862              ;
B862  20 99 B9    L604    JSR L482
B865  90 3C               BCC L468
B867              ;***************************************
B867              ;*PERFORM ADDITION.
B867              ;*THE CONTENTS OF FAC#1 IS ADDED TO FAC#2
B867              ;*AND THE RESULT STORED IN FAC#1. THERE ARE
B867              ;*TWO ENTRY POINTS TO THIS ROUTINE, THE FIRST
B867              ;*AT $B867 LOADS A 5 BYTE CONSTANT FROM
B867              ;*MEMORY INTO FAC#2 AND ADDS IT TO FAC#1. THE
```

```
LOC   CODE        LINE

B867              ;*SECOND AT $B86A ASSUMES THAT THE TWO
B867              ;*FLOATING POINT NUMBERS ARE ALREADY LOADED
B867              ;*INTO THE TWO FLOATING ACCUMULATORS. THE
B867              ;*RESULT IS STORED IN FAC#1.
B867              ;**************************************
B867              ;
B867              ;FAC#1=CONSTANT (.A,.Y)+FAC#1
B867              ;
B867  20 8C BA    L469   JSR L492        ;CONSTANT TO FAC#2
B86A              ;
B86A              ;FAC#1=FAC#1+FAC#2
B86A              ;
B86A  D0 03       L251   BNE L462        ;IS FAC#1 ZERO?
B86C  4C FC BB           JMP L392        ;MOVE FAC#2 TO FAC#1
B86F  A6 70       L462   LDX FACOV       ;GET FAC#1 LOW ORDER ROUNDING
B871  86 56              STX BASTMP+11   ;AND STORE
B873  A2 69              LDX #$69
B875  A5 69              LDA ARGEXP      ;GET FAC#2 EXPONENT
B877  A8          L465   TAY             ;TRANSFER TO .Y
B878  F0 CE              BEQ L460        ;IF ZERO EXIT
B87A  38                 SEC
B87B  E5 61              SBC FACEXP      ;GET DIFFERENCE BETWEEN TWO EXPONENTS
B87D  F0 24              BEQ L468        ;IF SAME JUMP
B87F  90 12              BCC L506        ;IS EXPONENT FAC#2 < EXPONENT FAC#1
B881  84 61              STY FACEXP      ;FAC#1 EXPONENT = FAC#2 EXPONENT
B883  A4 6E              LDY ARGSGN      ;GET FAC#2 SIGN
B885  84 66              STY FACSGN      ;STORE IN FAC#1 SIGN
B887  49 FF              EOR #$FF        ;INVERT SIGN AND
B889  69 00              ADC #$00        ;ADD CARRY (SET)
B88B  A0 00              LDY #$00
B88D  84 56              STY BASTMP+11   ;CLEAR FAC#1 ROUNDING FLAG TEMPORARY
B88F  A2 61              LDX #FACEXP
B891  D0 04              BNE L467        ;JUMP ALWAYS
B893  A0 00       L506   LDY #$00
B895  84 70              STY FACOV       ;CLEAR FAC#1 ROUNDING FLAG
B897  C9 F9       L467   CMP #$F9
B899  30 C7              BMI L604        ;ADJUST EXPONENT OF FAC+ARG
B89B  A8                 TAY
B89C  A5 70              LDA FACOV       ;GET LOW ORDER ROUNDING FLAG
B89E  56 01              LSR $01,X       ;DIVIDE HIGH ORDER BYTE OF FAC BY 2
B8A0  20 B0 B9           JSR L484        ;DIVIDE MANTISSA BY 2^.Y
B8A3  24 6F       L468   BIT ARISGN      ;TEST SIGN COMPARISON FLAG
B8A5  10 57              BPL L504        ;BOTH SIGNS THE SAME
B8A7  A0 61              LDY #FACEXP
B8A9  E0 69              CPX #$69        ;DECIDE WHICH ACCUMULATOR IS ARGUMENT
B8AB  F0 02              BEQ L464        ;FOR A SUBTRACTION
B8AD  A0 69              LDY #$69
B8AF  38          L464   SEC             ;SUBTRACT NEGATIVE SIGNED FAC FROM
B8B0  49 FF              EOR #$FF        ;POSITIVE SIGNED FAC AND STORE IN
B8B2  65 56              ADC BASTMP+11   ;FAC#1
B8B4  85 70              STA FACOV
B8B6  B9 04 00           LDA $0004,Y
B8B9  F5 04              SBC $04,X
B8BB  85 65              STA FACHO+3
B8BD  B9 03 00           LDA $0003,Y
B8C0  F5 03              SBC $03,X
B8C2  85 64              STA FACHO+2
B8C4  B9 02 00           LDA $0002,Y
B8C7  F5 02              SBC $02,X
B8C9  85 63              STA FACHO+1
B8CB  B9 01 00           LDA $0001,Y
```

```
LOC   CODE        LINE

B8CE  F5 01             SBC $01,X
B8D0  85 62             STA FACHO
B8D2  B0 03       L472  BCS L522         ;RESULT IS POSITIVE?
B8D4  20 47 B9          JSR L479        ;COMPLEMENT FAC#1
B8D7  A0 00       L522  LDY #$00
B8D9  98                TYA
B8DA  18                CLC
B8DB  A6 62       L473  LDX FACHO         ;GET HIGH ORDER BYTE OF FAC#1
B8DD  D0 4A             BNE L478        ;IF NOT ZERO JUMP
B8DF  A6 63             LDX FACHO+1     ;OTHERWISE MULTIPLY FAC#1 BY 256
B8E1  86 62             STX FACHO
B8E3  A6 64             LDX FACHO+2
B8E5  86 63             STX FACHO+1
B8E7  A6 65             LDX FACHO+3
B8E9  86 64             STX FACHO+2
B8EB  A6 70             LDX FACOV
B8ED  86 65             STX FACHO+3
B8EF  84 70             STY FACOV
B8F1  69 08             ADC #$08        ;ADD 8 TO ACCUMULATOR
B8F3  C9 20             CMP #$20        ;IS .A $20
B8F5  D0 E4             BNE L473        ;NO THEN SHIFT AGAIN WITH LOW ORDER 0
B8F7  A9 00       L476  LDA #$00
B8F9  85 61       L454  STA FACEXP        ;CLEAR EXPONENT FAC#1
B8FB  85 66       L590  STA FACSGN        ;CLEAR SIGN FAC#1
B8FD  60                RTS             ;EXIT FROM ADDITION
B8FE  65 56       L504  ADC BASTMP+11     ;ADD TWO FLOATING ACCUMULATORS
B900  85 70             STA FACOV       ;TOGETHER AND STORE RESULT IN FAC#1
B902  A5 65             LDA FACHO+3
B904  65 6D             ADC ARGHO+3
B906  85 65             STA FACHO+3
B908  A5 64             LDA FACHO+2
B90A  65 6C             ADC ARGHO+2
B90C  85 64             STA FACHO+2
B90E  A5 63             LDA FACHO+1
B910  65 6B             ADC ARGHO+1
B912  85 63             STA FACHO+1
B914  A5 62             LDA FACHO
B916  65 6A             ADC ARGHO
B918  85 62             STA FACHO
B91A  4C 36 B9          JMP L475        ;CORRECT EXPONENT
B91D                                    ;AND TEST FOR OVERFLOW
B91D  69 01       L471  ADC #$01          ;INCREMENT EXPONENT AND
B91F  06 70             ASL FACOV       ;MOVE FAC#1 LEFT 1 BIT
B921  26 65             ROL FACHO+3
B923  26 64             ROL FACHO+2
B925  26 63             ROL FACHO+1
B927  26 62             ROL FACHO
B929  10 F2       L478  BPL L471          ;IF HIGH BIT NOT SET REPEAT
B92B  38                SEC
B92C  E5 61             SBC FACEXP      ;TEST EXPONENT OF FAC#1
B92E  B0 C7             BCS L476        ;CLEAR EXPONENT AND EXIT
B930  49 FF             EOR #$FF        ;TWO'S COMPLEMENT EXPONENT AND
B932  69 01             ADC #$01
B934  85 61             STA FACEXP      ;STORE IT
B936  90 0E       L475  BCC L520          ;BRANCH TO EXIT
B938  E6 61       L477  INC FACEXP        ;INCREMENT EXPONENT
B93A  F0 42             BEQ L481        ;OUTPUT 'OVERFLOW' ERROR
B93C  66 62             ROR FACHO       ;MOVE CONTENTS OF FAC#1 DOWN ONE
B93E  66 63             ROR FACHO+1     ;BIT
B940  66 64             ROR FACHO+2
B942  66 65             ROR FACHO+3
```

```
LOC   CODE        LINE

B944  66 70             ROR FACOV
B946  60          L520  RTS               ;EXIT
B947              ;**************************************
B947              ;*COMPLEMENT FAC#1.
B947              ;*THIS ROUTINE REPLACES THE CONTENTS OF FAC#1
B947              ;*BY ITS TWOS COMPLEMENT - THIS MEANS THAT
B947              ;*ALL THE ZEROS ARE CONVERTED TO ONES AND
B947              ;*VICE VERSA THEN ONE IS ADDED TO THE RESULT.
B947              ;**************************************
B947  A5 66       L479  LDA FACSGN        ;COMPLEMENT SIGN OF FAC#1
B949  49 FF             EOR #$FF
B94B  85 66             STA FACSGN
B94D  A5 62       L474  LDA FACHO         ;COMPLEMENT FAC#1 HIGH BYTE
B94F  49 FF             EOR #$FF
B951  85 62             STA FACHO
B953  A5 63             LDA FACHO+1
B955  49 FF             EOR #$FF
B957  85 63             STA FACHO+1
B959  A5 64             LDA FACHO+2
B95B  49 FF             EOR #$FF
B95D  85 64             STA FACHO+2
B95F  A5 65             LDA FACHO+3
B961  49 FF             EOR #$FF
B963  85 65             STA FACHO+3
B965  A5 70             LDA FACOV       ;COMPLEMENT FAC#1 LOW ORDER ROUNDING
B967  49 FF             EOR #$FF
B969  85 70             STA FACOV
B96B  E6 70             INC FACOV       ;ADD 1 TO MANTISSA
B96D  D0 0E             BNE L519        ;IF NO CARRY THEN EXIT
B96F  E6 65       L530  INC FACHO+3      ;INCREMENT LOW BYTE OF MANTISSA
B971  D0 0A             BNE L519        ;IF NO CARRY EXIT
B973  E6 64             INC FACHO+2     ;INCREMENT SECOND BYTE UP OF MANTISSA
B975  D0 06             BNE L519        ;IF NO CARRY EXIT
B977  E6 63             INC FACHO+1     ;INCREMENT THIRD BYTE UP OF MANTISSA
B979  D0 02             BNE L519        ;IF NO CARRY EXIT
B97B  E6 62             INC FACHO       ;INCREMENT HIGH BYTE OF MANTISSA
B97D  60          L519  RTS               ;EXIT
B97E  A2 0F       L481  LDX #$0F          ;'OVERFLOW' ERROR NUMBER
B980  4C 37 A4          JMP L10         ;OUTPUT ERROR MESSAGE
B983              ;**************************************
B983              ;*MULTIPLY A BYTE.
B983              ;*TAKES THE CONTENTS OF FAC#1 AND MULTIPLIES
B983              ;*IT BY 256 (BYTE OF FAC#1=0).
B983              ;**************************************
B983  A2 25       L480  LDX #$25          ;INDEX TO PRODUCT AREA FOR MULTIPLY
B985  B4 04       L497  LDY $04,X         ;GET LO BYTE
B987  84 70             STY FACOV       ;STORE TO FAC#1 LOW ORDER ROUNDING
B989  B4 03             LDY $03,X       ;GET SECOND BYTE UP
B98B  94 04             STY $04,X       ;STORE TO LOW BYTE
B98D  B4 02             LDY $02,X       ;GET THIRD BYTE UP
B98F  94 03             STY $03,X       ;STORE TO SECOND BYTE
B991  B4 01             LDY $01,X       ;GET HIGH BYTE
B993  94 02             STY $02,X       ;STORE TO THIRD BYTE
B995  A4 68             LDY BITS        ;GET FAC#1 HIGH ORDER OVERFLOW
B997  94 01             STY $01,X       ;STORE TO HIGH BYTE
B999  69 08       L482  ADC #$08          ;INCREASE .A BY 8
B99B  30 E8             BMI L497        ;REPEAT ABOVE UNTIL
B99D  F0 E6             BEQ L497        ;0 < .A < 128
B99F  E9 08             SBC #$08        ;REMOVE FINAL 8 FROM .A
B9A1  A8                TAY             ;STORE IT IN .Y FOR COUNT-DOWN
B9A2  A5 70             LDA FACOV       ;GET LOW ORDER ROUNDING FLAG
```

```
LOC   CODE        LINE

B9A4  B0 14               BCS L470        ;DONE, EXIT
B9A6                ;
B9A6  16 01         L463  ASL $01,X        ;HIGH BYTE TIMES 2
B9A8  90 02               BCC L485        ;IF CARRY SET, INCREMENT
B9AA  F6 01               INC $01,X
B9AC  76 01         L485  ROR $01,X        ;SHIFT FAC RIGHT ONE BIT TO
B9AE  76 01               ROR $01,X       ;ADJUST CONTENTS
B9B0  76 02         L484  ROR $02,X
B9B2  76 03               ROR $03,X
B9B4  76 04               ROR $04,X
B9B6  6A                  ROR A
B9B7  C8                  INY
B9B8  D0 EC               BNE L463
B9BA  18            L470  CLC
B9BB  60                  RTS
B9BC                      .END
B9BC                      .LIB B12
B9BC                ;**************************************
B9BC                ;*CONSTANTS FOR LOG
B9BC                ;   1
B9BC  81                  .BYT $81,$00,$00,$00,$00
B9BD  00
B9BE  00
B9BF  00
B9C0  00
B9C1                ;   COUNTER FOR SERIES CALCULATION.
B9C1  03                  .BYT $03
B9C2                ;   .434255942
B9C2  7F                  .BYT $7F,$5E,$56,$CB,$79
B9C3  5E
B9C4  56
B9C5  CB
B9C6  79
B9C7                ;   .576584541
B9C7  80                  .BYT $80,$13,$9B,$0B,$64
B9C8  13
B9C9  9B
B9CA  0B
B9CB  64
B9CC                ;   .961800759
B9CC  80                  .BYT $80,$76,$38,$93,$16
B9CD  76
B9CE  38
B9CF  93
B9D0  16
B9D1                ;  2.88539007
B9D1  82                  .BYT $82,$38,$AA,$3B,$20
B9D2  38
B9D3  AA
B9D4  3B
B9D5  20
B9D6                ;   .707106781  (1/SQR(2))
B9D6  80                  .BYT $80,$35,$04,$F3,$34
B9D7  35
B9D8  04
B9D9  F3
B9DA  34
B9DB                ;  1.41421356   (SQR(2))
B9DB  81                  .BYT $81,$35,$04,$F3,$34
B9DC  35
B9DD  04
```

```
LOC   CODE        LINE

B9DE  F3
B9DF  34
B9E0              ;  -.5
B9E0  80                .BYT $80,$80,$00,$00,$00
B9E1  80
B9E2  00
B9E3  00
B9E4  00
B9E5              ;    .693147181  (LOG(2))
B9E5  80                .BYT $80,$31,$72,$17,$F8
B9E6  31
B9E7  72
B9E8  17
B9E9  F8
B9EA              ;****************************************
B9EA              ;*PERFORM LOG.
B9EA              ;*THIS PERFORMS THE CALCULATION LOG TO THE
B9EA              ;*BASE E ON A VALUE STORED IN FAC#1 AND PUTS
B9EA              ;*THE RESULT IN FAC#1.
B9EA              ;****************************************
B9EA  20 2B BC    L483    JSR L597        ;GET SIGN OF FAC#1
B9ED  F0 02               BEQ L594        ;ERROR
B9EF  10 03               BPL L486        ;IS O.K.
B9F1  4C 48 B2    L594    JMP L363        ;'ILLEGAL QUANTITY' IF ZERO OR MINUS
B9F4  A5 61       L486    LDA FACEXP      ;GET FAC#1 EXPONENT
B9F6  E9 7F               SBC #$7F        ;UN-COMPLEMENT
B9F8  48                  PHA             ;PUSH TO STACK
B9F9  A9 80               LDA #$80        ;CHANGE VALUE TO WITHIN 0.5 - 1
B9FB  85 61               STA FACEXP
B9FD  A9 D6               LDA #$D6        ;1/SQR(2) CONSTANT LSB
B9FF  A0 B9               LDY #$B9        ;MSB
BA01  20 67 B8            JSR L469        ;ADD TO FAC #1
BA04  A9 DB               LDA #$DB        ;SQR(2) CONSTANT LSB
BA06  A0 B9               LDY #$B9        ;MSB
BA08  20 0F BB            JSR L649        ;FAC#1=(.Y,.A)/FAC#1
BA0B  A9 BC               LDA #$BC        ;1 CONSTANT LSB
BA0D  A0 B9               LDY #$B9        ;MSB
BA0F  20 50 B8            JSR L570        ;FAC#1=(.Y,.A)-FAC#1
BA12  A9 C1               LDA #$C1        ;SERIES EVALUATION COUNTER LSB
BA14  A0 B9               LDY #$B9        ;MSB
BA16  20 43 E0            JSR L603        ;SERIES EVALUATE
BA19  A9 E0               LDA #$E0        ;-0.5 CONSTANT LSB
BA1B  A0 B9               LDY #$B9        ;MSB
BA1D  20 67 B8            JSR L469        ;ADD TO FAC#1
BA20  68                  PLA             ;GET EXPONENT FROM STACK
BA21  20 7E BD            JSR L553        ;FAC#1=FAC#1+.A
BA24  A9 E5               LDA #$E5        ;LOG(2) CONSTANT LSB
BA26  A0 B9               LDY #$B9        ;MSB
BA28              ;****************************************
BA28              ;*PERFORM MULTIPLY.
BA28              ;*THE CONTENTS OF FAC#1 ARE MULTIPLIED BY
BA28              ;*THE CONTENTS OF FAC#2 AND THE RESULT IS
BA28              ;*STORED IN FAC#1. THERE ARE TWO ENTRY POINTS
BA28              ;*TO THIS ROUTINE:
BA28              ;*    $BA28  THE VALUE POINTED TO BY (.Y,.A)
BA28              ;*           IS LOADED INTO FAC#2 AND THEN
BA28              ;*           MULTIPLY IS PERFORMED.
BA28              ;*    $BA2B  FAC#2 HAS ALREADY BEEN LOADED
BA28              ;*           WITH THE ARGUMENT.
BA28              ;****************************************
BA28  20 8C BA    L487    JSR L492        ;MOVE MEMORY TO FAC#2
```

```
LOC   CODE        LINE

BA2B  D0 03             BNE L563        ;IF EXPONENT NOT ZERO
BA2D  4C 8B BA          JMP L498        ;TO AN RTS
BA30  20 B7 BA    L563  JSR L461         ;ADJUST FAC#1/FAC#2
BA33  A9 00             LDA #$00        ;SET PRODUCT AREA TO ZERO
BA35  85 26             STA RESHO
BA37  85 27             STA RESHO+1
BA39  85 28             STA RESHO+2
BA3B  85 29             STA RESHO+3
BA3D  A5 70             LDA FACOV       ;GET LOW ORDER ROUNDING FLAG
BA3F  20 59 BA          JSR L491        ;MULTIPLY BY BYTE
BA42  A5 65             LDA FACHO+3     ;GET MANTISSA LSB
BA44  20 59 BA          JSR L491        ;MULTIPLY BY BYTE
BA47  A5 64             LDA FACHO+2     ;GET SECOND
BA49  20 59 BA          JSR L491        ;MULTIPLY BY BYTE
BA4C  A5 63             LDA FACHO+1     ;GET THIRD
BA4E  20 59 BA          JSR L491        ;MULTIPLY BY BYTE
BA51  A5 62             LDA FACHO       ;GET MSB
BA53  20 5E BA          JSR L494        ;MULTIPLY BY BYTE
BA56  4C 8F BB          JMP L508        ;TRANSFER PRODUCT TO FAC#1 AND ADJUST
BA59              ;
BA59              ;MULTIPLY A BYTE AND STORE THE RESULT IN THE
BA59              ;PRODUCT AREA $26,$27,$28,$29,$2A.
BA59              ;
BA59  D0 03       L491  BNE L494         ;IF BYTE NOT ZERO
BA5B  4C 83 B9          JMP L480        ;MULTIPLY BY ZERO BYTE (256)
BA5E  4A          L494  LSR A            ;BYTE TIMES 2
BA5F  09 80             ORA #$80        ;SET HIGH BIT OF BYTE
BA61  A8          L495  TAY              ;MOVE TO .Y
BA62  90 19             BCC L499        ;DON'T DO IF HIGH BIT WAS ZERO
BA64  18                CLC             ;ADD FAC#2 TO PRODUCT AREA
BA65  A5 29             LDA RESHO+3     ;PRODUCT LSB
BA67  65 6D             ADC ARGHO+3     ;PLUS FAC#2 LSB
BA69  85 29             STA RESHO+3
BA6B  A5 28             LDA RESHO+2
BA6D  65 6C             ADC ARGHO+2
BA6F  85 28             STA RESHO+2
BA71  A5 27             LDA RESHO+1
BA73  65 6B             ADC ARGHO+1
BA75  85 27             STA RESHO+1
BA77  A5 26             LDA RESHO       ;PRODUCT MSB
BA79  65 6A             ADC ARGHO       ;PLUS FAC#2 MSB
BA7B  85 26             STA RESHO
BA7D  66 26       L499  ROR RESHO        ;DIVIDE PRODUCT BY 2
BA7F  66 27             ROR RESHO+1
BA81  66 28             ROR RESHO+2
BA83  66 29             ROR RESHO+3
BA85  66 70             ROR FACOV
BA87  98                TYA
BA88  4A                LSR A           ;SHIFT NEXT BIT TO CARRY
BA89  D0 D6             BNE L495        ;IF NOT ZERO REPEAT
BA8B  60          L498  RTS              ;DONE
BA8C              ;***************************************
BA8C              ;*UNPACK MEMORY TO FAC#2.
BA8C              ;*THIS TAKES THE VALUE STORED AS A FIVE BYTE
BA8C              ;*VARIABLE IN MEMORY AT AN ADDRESS POINTED TO
BA8C              ;*BY .A (LOW ORDER) AND .Y (HI ORDER), UNPACKS
BA8C              ;*THE SIGN BYTE AND STORES THE VALUE IN THE
BA8C              ;*SIX BYTES OF FAC#2.
BA8C              ;***************************************
BA8C  85 22       L492  STA INDEX        ;STORE THE POINTER LSB
BA8E  84 23             STY INDEX+1     ;MSB
```

```
LOC   CODE        LINE

BA90  A0 04             LDY #$04
BA92  B1 22             LDA (INDEX),Y  ;GET FLOATING VALUE LSB
BA94  85 6D             STA ARGHO+3    ;STORE TO FAC#2 LSB
BA96  88                DEY
BA97  B1 22             LDA (INDEX),Y  ;GET SECOND VALUE
BA99  85 6C             STA ARGHO+2    ;STORE TO SECOND FAC#2
BA9B  88                DEY
BA9C  B1 22             LDA (INDEX),Y  ;GET THIRD VALUE
BA9E  85 6B             STA ARGHO+1    ;STORE TO THIRD FAC#2
BAA0  88                DEY
BAA1  B1 22             LDA (INDEX),Y  ;GET VALUE MSB
BAA3  85 6E             STA ARGSGN     ;STORE TO FAC#2 SIGN
BAA5  45 66             EOR FACSGN     ;REMOVE FAC#2 MSB
BAA7  85 6F             STA ARISGN     ;STORE TO SIGN COMPARISON FLAG
BAA9  A5 6E             LDA ARGSGN     ;GET VALUE MSB
BAAB  09 80             ORA #$80       ;REMOVE SIGN
BAAD  85 6A             STA ARGHO      ;STORE TO FAC#2 MSB
BAAF  88                DEY
BAB0  B1 22             LDA (INDEX),Y  ;GET VALUE EXPONENT
BAB2  85 69             STA ARGEXP     ;STORE TO FAC#2 EXPONENT
BAB4  A5 61             LDA FACEXP     ;GET FAC#1 EXPONENT
BAB6  60                RTS            ;EXIT
BAB7            ;****************************************
BAB7            ;*TEST AND ADJUST ACCUMULATORS.
BAB7            ;*THIS CHECKS THE EXPONENT VALUES OF THE TWO
BAB7            ;*FLOATING POINT ACCUMULATORS AND IF THE
BAB7            ;*VALUES ARE TOO LARGE OR SMALL GENERATES
BAB7            ;*AN OVERFLOW ERROR BEFORE SETTING THE
BAB7            ;*CONTENTS OF FAC#1 TO ZERO. IF EITHER
BAB7            ;*ACCUMULATOR IS ZERO THEN BOTH WILL BE SET
BAB7            ;*TO ZERO BY THIS ROUTINE.
BAB7            ;****************************************
BAB7  A5 69     L461    LDA ARGEXP      ;GET FAC#2 EXPONENT
BAB9  F0 1F     L493    BEQ L601        ;CLEAR FAC#1 SIGN & EXPONENT & EXIT
BABB  18                CLC
BABC  65 61             ADC FACEXP     ;ADD TO FAC#1 EXPONENT
BABE  90 04             BCC L606       ;JUMP IF NO CARRY
BAC0  30 1D             BMI L500       ;IF> 127 GENERATE 'OVERFLOW ERROR'
BAC2  18                CLC
BAC3  2C                .BYT $2C
BAC4  10 14     L606    BPL L601        ;CLEAR FAC#1 SIGN & EXPONENT & EXIT
BAC6  69 80             ADC #$80       ;ADD $80 TO FAC#1 EXPONENT AND
BAC8  85 61             STA FACEXP     ;STORE
BACA  D0 03             BNE L501       ;JUMP IF NOT ZERO
BACC  4C FB B8          JMP L590       ;SET SIGN TO ZERO AND EXIT
BACF  A5 6F     L501    LDA ARISGN      ;GET SIGN COMPARISON AND
BAD1  85 66             STA FACSGN     ;STORE TO FAC#1 SIGN
BAD3  60                RTS
BAD4            ;
BAD4            ;UNDERFLOW / OVERFLOW
BAD4            ;
BAD4  A5 66     L503    LDA FACSGN      ;GET FAC#1 SIGN
BAD6  49 FF             EOR #$FF       ;INVERT IT
BAD8  30 05             BMI L500       ;IF >127 'OVERFLOW' ERROR
BADA  68        L601    PLA             ;PULL STACK UP TWO
BADB  68                PLA
BADC  4C F7 B8          JMP L476       ;CLEAR FAC#1 SIGN & EXPONENT & EXIT
BADF  4C 7E B9  L500    JMP L481        ;GENERATE 'OVERFLOW ERROR'
BAE2            ;****************************************
BAE2            ;*MULTIPLY BY 10.
BAE2            ;*THE CONTENTS OF FAC#1 ARE MULTIPLIED BY 10
```

```
LOC   CODE        LINE

BAE2              ;*AND THE RESULT STORED IN FAC#1.
BAE2              ;**************************************
BAE2  20 0C BC    L502   JSR L516         ;ROUND FAC#1 AND TRANSFER TO FAC#2
BAE5  AA                 TAX
BAE6  F0 10              BEQ L162        ;IF FAC#1 = 0 THEN EXIT
BAE8  18                 CLC
BAE9  69 02              ADC #$02        ;ADD 2 TO EXPONENT (MULTIPLY FAC#2 *4)
BAEB  B0 F2              BCS L500        ;IF CARRY SET 'OVERFLOW ERROR'
BAED  A2 00       L159   LDX #$00
BAEF  86 6F              STX ARISGN      ;SET SIGN COMPARASON TO ZERO
BAF1  20 77 B8           JSR L465        ;FAC#1=FAC#1+FAC#2 (MULTIPLY FAC#1 *5)
BAF4  E6 61              INC FACEXP      ;FAC#1 EXPONENT *2 (MULTIPLY FAC# *10)
BAF6  F0 E7              BEQ L500        ;GENERATE 'OVERFLOW ERROR'
BAF8  60          L162   RTS
BAF9              ;**************************************
BAF9              ;*FLOATING POINT CONSTANT 10
BAF9              ;**************************************
BAF9  84                 .BYT $84,$20,$00,$00,$00
BAFA  20
BAFB  00
BAFC  00
BAFD  00
BAFE              ;**************************************
BAFE              ;*DIVIDE BY 10.
BAFE              ;*THE CONTENTS OF FAC#1 ARE DIVIDED BY 10
BAFE              ;*AND THE RESULT STORED IN FAC#1.
BAFE              ;**************************************
BAFE  20 0C BC    L505   JSR L516         ;ROUND FAC#1 AND TRANSFER TO FAC#2
BB01  A9 F9              LDA #$F9        ;POINTER TO CONSTANT 10 LSB.
BB03  A0 BA              LDY #$BA        ;MSB
BB05  A2 00              LDX #$00
BB07  86 6F       L549   STX ARISGN       ;CLEAR SIGN COMPARISON FLAG
BB09  20 A2 BB           JSR L496        ;TRANSFER CONSTANT (.A,.Y) TO FAC#1
BB0C  4C 12 BB           JMP L488        ;FAC#1 = FAC#2 / FAC#1
BB0F              ;**************************************
BB0F              ;*PERFORM DIVIDE BY.
BB0F              ;*THIS DIVIDES FAC#2 BY FAC#1 AND PUTS THE
BB0F              ;*RESULT IN FAC#1. ON ENTRY THE POINTER TO
BB0F              ;*THE 5 BYTE FLOATING POINT VALUE TO BE
BB0F              ;*STORED IN FAC#2 IS CONTAINED IN .A(LOW ORDER)
BB0F              ;*AND .Y(HI ORDER) AND .X IS LOADED WITH THE
BB0F              ;*SIGN COMPARISON BYTE-$6F. THE CONTENTS OF
BB0F              ;*FAC#2 ARE THEN DIVIDED BY THE CONTENTS OF
BB0F              ;*FAC#1, WHICH WERE LOADED PRIOR TO ROUTINE
BB0F              ;*ENTRY, AND THE RESULT STORED IN FAC#1,
BB0F              ;*LEAVING FAC#2 UNCHANGED.
BB0F              ;**************************************
BB0F  20 8C BA    L649   JSR L492         ;TRANSFER MEMORY TO FAC#2
BB12  F0 76       L488   BEQ L512         ;IF FAC#1=0 THEN 'DIVISION BY ZERO'
BB14  20 1B BC           JSR L518        ;ROUND FAC#1
BB17  A9 00              LDA #$00
BB19  38                 SEC
BB1A  E5 61              SBC FACEXP      ;0 - EXPONENT OF FAC#1
BB1C  85 61              STA FACEXP
BB1E  20 B7 BA           JSR L461        ;TEST AND ADJUST FACS
BB21  E6 61              INC FACEXP      ;INCREMENT FAC#1 EXPONENT
BB23  F0 BA              BEQ L500        ;EXPONENT=0, 'OVERFLOW' ERROR
BB25  A2 FC              LDX #$FC        ;POINTER TO FUNCTION REGISTER LSB
BB27  A9 01              LDA #$01        ;MSB
BB29  A4 6A       L507   LDY ARGHO        ;GET FAC#2 MANTISSA MSB
BB2B  C4 62              CPY FACHO       ;SAME AS FAC#1 MANTISSA MSB?
```

```
LOC   CODE        LINE

BB2D  D0 10             BNE L514        ;NO
BB2F  A4 6B             LDY ARGHO+1     ;SAME FOR OTHER 3 BYTES OF MANTISSA
BB31  C4 63             CPY FACHO+1
BB33  D0 0A             BNE L514        ;NO
BB35  A4 6C             LDY ARGHO+2
BB37  C4 64             CPY FACHO+2
BB39  D0 04             BNE L514        ;NO
BB3B  A4 6D             LDY ARGHO+3
BB3D  C4 65             CPY FACHO+3
BB3F  08          L514  PHP              ;SAVE THE STATUS
BB40  2A                ROL A           ;MULTIPLY .A BY 2
BB41  90 09             BCC L509        ;LOST A BIT FROM .A
BB43  E8                INX             ;INCREMENT .X REGISTER
BB44  95 29             STA RESHO+3,X   ;STORE .A
BB46  F0 32             BEQ L513        ;.X=0
BB48  10 34             BPL L511        ;.X<128,MULTIPLY .A BY 64
BB4A  A9 01             LDA #$01        ;SET .A TO 1
BB4C  28          L509  PLP              ;GET STATUS
BB4D  B0 0E             BCS L515        ;CARRY SET, SUBTRACT FAC#1 FROM FAC#2
BB4F  06 6D       L510  ASL ARGHO+3     ;MULTIPLY FAC#2 BY 2
BB51  26 6C             ROL ARGHO+2
BB53  26 6B             ROL ARGHO+1
BB55  26 6A             ROL ARGHO
BB57  B0 E6             BCS L514        ;CARRY SET
BB59  30 CE             BMI L507        ;$6A>127
BB5B  10 E2             BPL L514        ;$6A<128 (JUMP)
BB5D  A8          L515  TAY              ;SAVE .A
BB5E  A5 6D             LDA ARGHO+3     ;FAC#2=FAC#2-FAC#1
BB60  E5 65             SBC FACHO+3
BB62  85 6D             STA ARGHO+3
BB64  A5 6C             LDA ARGHO+2
BB66  E5 64             SBC FACHO+2
BB68  85 6C             STA ARGHO+2
BB6A  A5 6B             LDA ARGHO+1
BB6C  E5 63             SBC FACHO+1
BB6E  85 6B             STA ARGHO+1
BB70  A5 6A             LDA ARGHO
BB72  E5 62             SBC FACHO
BB74  85 6A             STA ARGHO
BB76  98                TYA             ;RESTORE .A
BB77  4C 4F BB          JMP L510        ;CONTINUE WITH DIVISION
BB7A  A9 40       L513  LDA #$40         ;.A=64
BB7C  D0 CE             BNE L509        ;JUMP
BB7E  0A          L511  ASL A            ;MULTIPLY .A BY 64
BB7F  0A                ASL A
BB80  0A                ASL A
BB81  0A                ASL A
BB82  0A                ASL A
BB83  0A                ASL A
BB84  85 70             STA FACOV       ;STORE TO FAC#1 LOW ORDER ROUNDING FLAG
BB86  28                PLP             ;GET STATUS
BB87  4C 8F BB          JMP L508        ;PRODUCT TO FAC#1, ADJUST, AND EXIT
BB8A  A2 14       L512  LDX #$14         ;'DIVISION BY ZERO' ERROR
BB8C  4C 37 A4          JMP L10         ;OUTPUT MESSAGE
BB8F  A5 26       L508  LDA RESHO       ;TRANSFER PRODUCT AREA INTO FAC#1
BB91  85 62             STA FACHO
BB93  A5 27             LDA RESHO+1
BB95  85 63             STA FACHO+1
BB97  A5 28             LDA RESHO+2
BB99  85 64             STA FACHO+2
BB9B  A5 29             LDA RESHO+3
```

```
LOC   CODE        LINE

BB9D  85 65             STA FACHO+3
BB9F  4C D7 B8          JMP L522        ;ADJUST FAC#1
BBA2                ;***************************************
BBA2                ;*UNPACK MEMORY INTO FAC#1.
BBA2                ;*THIS LOADS A VALUE STORED AS A 5 BYTE
BBA2                ;*FLOATING POINT NUMBER, EXTRACTS THE SIGN
BBA2                ;*BYTE AND THEN STORES IT IN THE 6 BYTES
BBA2                ;*OF FAC#1. THE LOCATION OF THE VALUE IN
BBA2                ;*MEMORY IS POINTED TO BY THE CONTENTS OF
BBA2                ;*.A(LOW ORDER) AND .Y(HI ORDER).
BBA2                ;***************************************
BBA2  85 22         L496    STA INDEX       ;STORE FLOATING VALUE POINTER LSB
BBA4  84 23             STY INDEX+1     ;MSB
BBA6  A0 04             LDY #$04
BBA8  B1 22             LDA (INDEX),Y   ;GET VALUE LSB
BBAA  85 65             STA FACHO+3     ;STORE TO FAC#1 LSB
BBAC  88                DEY
BBAD  B1 22             LDA (INDEX),Y   ;GET SECOND VALUE
BBAF  85 64             STA FACHO+2     ;STORE TO FAC#1 SECOND VALUE
BBB1  88                DEY
BBB2  B1 22             LDA (INDEX),Y   ;GET THIRD VALUE
BBB4  85 63             STA FACHO+1     ;STORE TO FAC#1 THIRD VALUE
BBB6  88                DEY
BBB7  B1 22             LDA (INDEX),Y   ;GET VALUE MSB
BBB9  85 66             STA FACSGN      ;STORE TO FAC#1 SIGN
BBBB  09 80             ORA #$80        ;REMOVE SIGN BIT
BBBD  85 62             STA FACHO       ;STORE TO FAC#1 MSB
BBBF  88                DEY
BBC0  B1 22             LDA (INDEX),Y   ;GET VALUE EXPONENT
BBC2  85 61             STA FACEXP      ;STORE TO FAC#1 EXPONENT
BBC4  84 70             STY FACOV       ;STORE ZERO TO FAC#1 LOW ORDER
BBC6  60                RTS
BBC7                ;***************************************
BBC7                ;*PACK FAC#1 TO MEMORY.
BBC7                ;*THIS COMPRESSES THE 6 BYTES OF FAC#1 INTO
BBC7                ;*5 BYTES BY STORING THE SIGN BYTE AS THE
BBC7                ;*MOST SIGNIFICANT BIT OF THE MANTISSA MSB.
BBC7                ;*THESE FIVE BYTES ARE THEN STORED IN A MEMORY
BBC7                ;*LOCATION POINTED TO BY .X(LOW ORDER) AND
BBC7                ;*.Y(HI ORDER). THERE ARE 4 DIFFERENT ENTRY
BBC7                ;*POINTS AS FOLLOWS:
BBC7                ;*  $BBC7  PACK FAC#1 INTO $005C
BBC7                ;*  $BBCA  PACK FAC#1 INTO $0057
BBC7                ;*  $BBD0  PACK FAC#1 INTO CURRENT VARIABLE
BBC7                ;*         ADDRESS ($49)
BBC7                ;*  $BBD4  PACK FAC#1 INTO MEMORY POINTED
BBC7                ;*         TO BY .X(LOW ORDER) AND .Y
BBC7                ;*         (HI ORDER).
BBC7                ;***************************************
BBC7  A2 5C         L98     LDX #$5C        ;SET POINTER TO $005C
BBC9  2C                .BYT $2C        ;SKIP NEXT COMMAND
BBCA  A2 57         L609    LDX #$57        ;SET POINTER TO $0057
BBCC  A0 00             LDY #$00
BBCE  F0 04             BEQ L154        ;JUMP TO PACK
BBD0  A6 49         L607    LDX FORPNT      ;SET POINTER TO CURRENT VARIABLE LSB
BBD2  A4 4A             LDY FORPNT+1    ;MSB
BBD4  20 1B BC      L154    JSR L518        ;ROUND FAC#1
BBD7  86 22             STX INDEX       ;STORE POINTER LSB
BBD9  84 23             STY INDEX+1     ;MSB
BBDB  A0 04             LDY #$04
BBDD  A5 65             LDA FACHO+3     ;GET FAC#1 LSB
```

```
LOC   CODE        LINE

BBDF  91 22             STA (INDEX),Y  ;STORE TO MEMORY
BBE1  88                DEY
BBE2  A5 64             LDA FACHO+2    ;GET FAC#1 SECOND BYTE
BBE4  91 22             STA (INDEX),Y  ;STORE TO MEMORY
BBE6  88                DEY
BBE7  A5 63             LDA FACHO+1    ;GET FAC#1 THIRD BYTE
BBE9  91 22             STA (INDEX),Y  ;STORE TO MEMORY
BBEB  88                DEY
BBEC  A5 66             LDA FACSGN     ;GET FAC#1 SIGN
BBEE  09 7F             ORA #$7F       ;GET RID OF HIGH BIT
BBF0  25 62             AND FACHO      ;AND IN FAC#1 MSB
BBF2  91 22             STA (INDEX),Y  ;STORE IT TO MEMORY
BBF4  88                DEY
BBF5  A5 61             LDA FACEXP     ;GET FAC#1 EXPONENT
BBF7  91 22             STA (INDEX),Y  ;STORE IT TO MEMORY
BBF9  84 70             STY FACOV      ;SET FAC#1 LOW ORDER FLAG TO ZERO
BBFB  60                RTS
BBFC              ;****************************************
BBFC              ;*MOVE FAC#2 TO FAC#1.
BBFC              ;*THIS MOVES THE ENTIRE CONTENTS OF FAC#2
BBFC              ;*INTO FAC#1, LEAVING BOTH CONTAINING THE
BBFC              ;*SAME VALUE. THE ROUNDING BYTE $70 IS
BBFC              ;*ZEROED.
BBFC              ;****************************************
BBFC  A5 6E       L392  LDA ARGSGN       ;GET FAC#2 SIGN
BBFE  85 66       L314  STA FACSGN       ;STORE TO FAC#1 SIGN
BC00  A2 05             LDX #$05         ;LOOP TO COPY THE EXPONENT
BC02  B5 68       L593  LDA BITS,X       ;AND THE MANTISSA FROM FAC#2
BC04  95 60             STA FACEXP-1,X ;TO FAC#1
BC06  CA                DEX
BC07  D0 F9             BNE L593       ;DO NEXT
BC09  86 70             STX FACOV      ;SET FAC#1 ROUNDING BYTE TO ZERO
BC0B  60                RTS
BC0C              ;****************************************
BC0C              ;*MOVE FAC#1 TO FAC#2.
BC0C              ;*THIS MOVES THE ENTIRE CONTENTS OF FAC#1
BC0C              ;*INTO FAC#2, LEAVING BOTH CONTAINING THE
BC0C              ;*SAME VALUE. THE ROUNDING BYTE $70 IS
BC0C              ;*ZEROED.
BC0C              ;****************************************
BC0C  20 1B BC    L516  JSR L518         ;ROUND FAC#1
BC0F  A2 06       L160  LDX #$06         ;LOOP TO COPY 6 BYTES OF
BC11  B5 60       L599  LDA FACEXP-1,X   ;FAC#1 INTO
BC13  95 68             STA BITS,X       ;FAC#2
BC15  CA                DEX
BC16  D0 F9             BNE L599       ;DO NEXT BYTE
BC18  86 70             STX FACOV      ;ZERO ROUNDING FLAG
BC1A  60          L517  RTS
BC1B              ;****************************************
BC1B              ;*ROUND FAC#1.
BC1B              ;*THE EXPONENT OF FAC#1 IN BYTE $61 IS
BC1B              ;*TESTED, IF THE CONTENTS ARE ZERO THE
BC1B              ;*ROUTINE EXITS, IF NOT THEN THE ROUNDING
BC1B              ;*BYTE IN $70 IS MULTIPLIED BY 2 AND THE
BC1B              ;*STATE OF THE CARRY FLAG IS CHECKED. IF
BC1B              ;*CARRY IS CLEAR THEN IT EXITS. OTHERWISE
BC1B              ;*THE FLOATING POINT VALUE IS INCREMENTED
BC1B              ;*BY 1.
BC1B              ;****************************************
BC1B  A5 61       L518  LDA FACEXP       ;GET FAC#1 EXPONENT
BC1D  F0 FB             BEQ L517       ;ZERO, EXIT
```

```
LOC   CODE        LINE

BC1F  06 70             ASL FACOV       ;ROUNDING BYTE TIMES 2
BC21  90 F7             BCC L517        ;CARRY CLEAR, EXIT
BC23  20 6F B9    L152  JSR L530         ;ADD 1 TO FAC#1
BC26  D0 F2             BNE L517        ;NOT ZERO
BC28  4C 38 B9          JMP L477        ;ADJUST FAC#1
BC2B              ;***************************************
BC2B              ;*GET FAC#1 SIGN.
BC2B              ;*THIS FINDS THE SIGN OF THE VALUE IN FAC#1
BC2B              ;*AND STORES IT IN .A.IF .A=$01 THEN THE VALUE
BC2B              ;*IS POSITIVE, $FF IS NEGATIVE, $00 MEANS
BC2B              ;*THAT THE VALUE IS ZERO.
BC2B              ;***************************************
BC2B  A5 61       L597  LDA FACEXP      ;GET FAC#1 EXPONENT
BC2D  F0 09             BEQ L527        ;FAC#1 IS ZERO, EXIT
BC2F  A5 66       L100  LDA FACSGN       ;GET FAC#1 SIGN
BC31  2A          L523  ROL A            ;SHIFT OUT THE HIGH BIT
BC32  A9 FF             LDA #$FF        ;ASSUME NEGATIVE
BC34  B0 02             BCS L527        ;CORRECT IT IS NEGATIVE
BC36  A9 01             LDA #$01        ;ELSE SET FOR POSITIVE
BC38  60          L527  RTS              ;DONE
BC39              ;***************************************
BC39              ;*PERFORM SGN.
BC39              ;*THE ROUTINE TO GET THE SIGN OF FAC#1 IS
BC39              ;*CALLED AND THE SIGN OF THE VALUE IN FAC#1
BC39              ;*IS PUT INTO THE MSB OF FAC#1, $88 INTO THE
BC39              ;*EXPONENT OF FAC#1 AND THE REST OF FAC#1
BC39              ;*IS ZEROED.
BC39              ;***************************************
BC39  20 2B BC    SGN   JSR L597         ;GET SIGN OF FAC#1
BC3C  85 62       L521  STA FACHO        ;STORE TO MSB FAC#1
BC3E  A9 00             LDA #$00        ;SET 2ND BYTE TO ZERO
BC40  85 63             STA FACHO+1
BC42  A2 88             LDX #$88        ;LOAD .X WITH EXPONENT
BC44  A5 62       L310  LDA FACHO        ;GET FAC#1 MSB
BC46  49 FF             EOR #$FF        ;NOT IT
BC48  2A                ROL A          ;SHIFT MSBIT TO CARRY
BC49  A9 00       L386  LDA #$00
BC4B  85 65             STA FACHO+3     ;ZERO REST OF FAC#1 MANTISSA
BC4D  85 64             STA FACHO+2
BC4F  86 61       L557  STX FACEXP       ;STORE EXPONENT
BC51  85 70             STA FACOV       ;ZERO FAC#1 ROUNDING BYTE
BC53  85 66             STA FACSGN      ;ZERO FAC#1 SIGN
BC55  4C D2 B8          JMP L472        ;CORRECT FAC#1 FOR TRUE OR FALSE
BC58              ;***************************************
BC58              ;*PERFORM ABS.
BC58              ;*THIS SIMPLY ENSURES THAT THE SIGN BYTE OF
BC58              ;*FAC#1 $66 ALWAYS CONTAINS A POSITIVE FLAG
BC58              ;*$01.
BC58              ;***************************************
BC58  46 66       ABS   LSR FACSGN       ;SHIFT SIGN BYTE TO REMOVE NEGATIVE
BC5A  60                RTS             ;EXIT
BC5B              ;***************************************
BC5B              ;*COMPARE FAC#1 TO VALUE IN MEMORY.
BC5B              ;*THE VALUE STORED IN FAC#1 IS COMPARED WITH
BC5B              ;*A FIVE BYTE FLOATING POINT VALUE STORED IN
BC5B              ;*MEMORY AT A LOCATION POINTED TO BY THE
BC5B              ;*ACCUMULATOR (LOW ORDER) AND Y INDEX REGISTER
BC5B              ;*(HIGH ORDER). ON EXIT THE ACCUMULATOR
BC5B              ;*CONTAINS THE COMPARISON FLAG: $00 = THAT
BC5B              ;*BOTH VALUES ARE EQUAL, $01 = MEANS THAT FAC#1
BC5B              ;*IS GREATER THAN THE VALUE IN MEMORY AND $FF
```

```
LOC   CODE        LINE

BC5B              ;*= THAT FAC#1 IS LESS THAN THE VALUE IN ·
BC5B              ;*MEMORY.
BC5B              ;***************************************
BC5B  85 24       L308   STA INDEX+2     ;SAVE CONSTANT POINTER LSB
BC5D  84 25       L316   STY INDEX+3     ;MSB
BC5F  A0 00              LDY #$00
BC61  B1 24              LDA (INDEX+2),Y ;GET EXPONENT FROM MEMORY
BC63  C8                 INY
BC64  AA                 TAX             ;PUT EXPONENT IN .X
BC65  F0 C4              BEQ L597        ;IF ZERO THEN GET SIGN OF FAC#1
BC67  B1 24              LDA (INDEX+2),Y ;GET MSB OF MANTISSA IN MEMORY
BC69  45 66              EOR FACSGN      ;IS SIGN SAME
BC6B  30 C2              BMI L100        ;DIFFERENT SIGNS
BC6D  E4 61              CPX FACEXP      ;COMPARE EXPONENTS
BC6F  D0 21              BNE L252        ;DIFFERENT EXPONENTS
BC71  B1 24              LDA (INDEX+2),Y ;GET MSB OF MANTISSA FROM MEMORY
BC73  09 80              ORA #$80        ;GET RID IF SIGN BIT
BC75  C5 62              CMP FACHO       ;COMPARE WITH FAC#1 MSB
BC77  D0 19              BNE L252        ;DIFFERENT
BC79  C8                 INY
BC7A  B1 24              LDA (INDEX+2),Y ;GET BYTE 2 OF MANTISSA FROM MEMORY
BC7C  C5 63              CMP FACHO+1     ;COMPARE WITH FAC#1 BYTE 2
BC7E  D0 12              BNE L252        ;DIFFERENT
BC80  C8                 INY
BC81  B1 24              LDA (INDEX+2),Y ;GET BYTE 3 OF MANTISSA FROM MEMORY
BC83  C5 64              CMP FACHO+2     ;COMPARE WITH FAC#1 BYTE 3
BC85  D0 0B              BNE L252        ;DIFFERENT
BC87  C8                 INY
BC88  A9 7F              LDA #$7F
BC8A  C5 70              CMP FACOV       ;CHECK FOR HIGH BIT IN ROUNDING FLAG
BC8C  B1 24              LDA (INDEX+2),Y ;GET LSB OF MANTISSA FROM MEMORY
BC8E  E5 65              SBC FACHO+3     ;SUBTRACT LSB OF FAC#1 USING CARRY
BC90  F0 28              BEQ L529        ;EQUAL SO EXIT
BC92  A5 66       L252   LDA FACSGN       ;GET SIGN OF FAC#1
BC94  90 02              BCC L524        ;BRANCH IF FAC#1 > THAN MEMORY
BC96  49 FF              EOR #$FF        ;INVERT
BC98  4C 31 BC    L524   JMP L523         ;SET RESULT FLAG
BC9B              ;***************************************
BC9B              ;*FLOATING TO FIXED POINT CONVERSION.
BC9B              ;*THE FLOATING POINT NUMBER IS STORED IN FAC#1
BC9B              ;*AND IS CONVERTED TO A TWO BYTE FIXED POINT
BC9B              ;*NUMBER WHICH IS STORED IN $65(LSB),$66(MSB)
BC9B              ;*(MSB). IF THE VALUE IN FAC#1 IS GREATER
BC9B              ;*THAN 32767 THEN THE OVERFLOW IS STORED IN
BC9B              ;*$68.
BC9B              ;***************************************
BC9B  A5 61       L526   LDA FACEXP       ;GET EXPONENT OF FAC#1
BC9D  F0 4A              BEQ L592        ;BRANCH IF ZERO
BC9F  38                 SEC
BCA0  E9 A0              SBC #$A0
BCA2  24 66              BIT FACSGN      ;TEST SIGN OF FAC#1
BCA4  10 09              BPL L164        ;BRANCH IF POSITIVE
BCA6  AA                 TAX
BCA7  A9 FF              LDA #$FF
BCA9  85 68              STA BITS        ;SET FAC#1 OVERFLOW TO $FF
BCAB  20 4D B9           JSR L474        ;INVERT MANTISSA OF FAC#1
BCAE  8A                 TXA             ;RESTORE EXPONENT
BCAF  A2 61       L164   LDX #FACEXP     ;POINTER TO FAC#1
BCB1  C9 F9              CMP #$F9
BCB3  10 06              BPL L525
BCB5  20 99 B9           JSR L482        ;SHIFT FAC#1 RIGHT
```

```
LOC   CODE        LINE

BCB8  84 68             STY BITS        ;SAVE OVERFLOW
BCBA  60          L529  RTS              ;EXIT
BCBB  A8          L525  TAY
BCBC  A5 66             LDA FACSGN
BCBE  29 80             AND #$80        ;MASK OUT ALL BUT SIGN BIT
BCC0  46 62             LSR FACHO
BCC2  05 62             ORA FACHO
BCC4  85 62             STA FACHO
BCC6  20 B0 B9          JSR L484        ;SHIFT FAC#1 RIGHT BIT BY BIT
BCC9  84 68             STY BITS        ;SAVE OVERFLOW
BCCB  60                RTS
BCCC                ;***************************************
BCCC                ;*PERFORM 'INT'.
BCCC                ;*THE VALUE STORED IN FAC#1 IS ROUNDED DOWN
BCCC                ;*TO THE NEAREST INTEGER BUT IS LEFT IN FULL
BCCC                ;*FLOATING POINT FORM IN FAC#1.
BCCC                ;***************************************
BCCC  A5 61       L531  LDA FACEXP      ;GET EXPONENT FAC#1
BCCE  C9 A0             CMP #$A0        ;IS IT INTEGER
BCD0  B0 20             BCS L528        ;YES THEN EXIT
BCD2  20 9B BC          JSR L526        ;CONVERT FAC#1 TO INTEGER
BCD5  84 70             STY FACOV       ;SAVE LOW ORDER ROUNDING OF FAC#1
BCD7  A5 66             LDA FACSGN      ;GET FAC#1 SIGN +MSB
BCD9  84 66             STY FACSGN
BCDB  49 80             EOR #$80
BCDD  2A                ROL A           ;ROTATE SIGN BIT INTO CARRY
BCDE  A9 A0             LDA #$A0        ;FLAG INTEGER
BCE0  85 61             STA FACEXP      ;IN EXPONENT
BCE2  A5 65             LDA FACHO+3     ;GET LSB
BCE4  85 07             STA CHARAC
BCE6  4C D2 B8          JMP L472        ;ADJUST FAC#1
BCE9  85 62       L592  STA FACHO       ;FILL MANTISSA OF FAC#1 WITH
BCEB  85 63             STA FACHO+1     ;CONTENTS OF ACCUMULATOR
BCED  85 64             STA FACHO+2
BCEF  85 65             STA FACHO+3
BCF1  A8                TAY             ;RETURN ACCUMULATOR VALUE IN .Y
BCF2  60          L528  RTS
BCF3                ;***************************************
BCF3                ;*CONVERT STRING TO FLOATING POINT.
BCF3                ;*THE STRING TO BE CONVERTED IS STORED AT A
BCF3                ;*LOCATION POINTED TO BY CHARGET POINTERS
BCF3                ;*$7A,$7B. THE NUMERIC VALUE STORED IN THE
BCF3                ;*STRING IS CHECKED AND THEN CONVERTED TO
BCF3                ;*FLOATING POINT FORM IN FAC#1. THIS ROUTINE
BCF3                ;*IS USED BY 'VAL'.
BCF3                ;***************************************
BCF3  A0 00       L532  LDY #$00         ;SET TO ZERO
BCF5  A2 0A             LDX #$0A        ;SET .X TO POINTER
BCF7  94 5D       L237  STY FACEXP-4,X  ;CLEAR LOCATIONS $5D TO $66
BCF9  CA                DEX
BCFA  10 FB             BPL L237        ;REPEAT UNTIL DONE
BCFC  90 0F             BCC L536        ;BRANCH IF NEXT CHARACTER IS NUMERIC
BCFE  C9 2D             CMP #$2D        ;IS CHARACTER '-'
BD00  D0 04             BNE L533        ;NO
BD02  86 67             STX SGNFLG      ;SET TO $FF
BD04  F0 04             BEQ L535        ;BRANCH ALWAYS
BD06  C9 2B       L533  CMP #$2B         ;IS CHARACTER '+'
BD08  D0 05             BNE L534        ;NO
BD0A  20 73 00    L535  JSR CHRGET       ;GET NEXT CHARACTER
BD0D  90 5B       L536  BCC L551         ;BRANCH IF CHARACTER NUMERIC
BD0F  C9 2E       L534  CMP #$2E         ;IS CHARACTER '.'
```

```
LOC   CODE        LINE

BD11  F0 2E             BEQ L544        ;YES
BD13  C9 45             CMP #$45        ;IS CHARACTER 'E'
BD15  D0 30             BNE L539        ;NO
BD17  20 73 00          JSR CHRGET      ;GET NEXT CHARACTER
BD1A  90 17             BCC L543        ;BRANCH IF CHARACTER NUMERIC
BD1C  C9 AB             CMP #$AB        ;CHECK IF BASIC TOKEN FOR '-'
BD1E  F0 0E             BEQ L537        ;YES
BD20  C9 2D             CMP #$2D        ;CHECK IF CHARACTER IS '-'
BD22  F0 0A             BEQ L537        ;YES
BD24  C9 AA             CMP #$AA        ;CHECK IF BASIC TOKEN FOR '+'
BD26  F0 08             BEQ L542        ;YES
BD28  C9 2B             CMP #$2B        ;CHECK IF CHARACTER IS '+'
BD2A  F0 04             BEQ L542        ;YES
BD2C  D0 07             BNE L541        ;NOT ANY OF ABOVE THEN ILLEGAL CHAR
BD2E  66 60       L537  ROR FACEXP-1     ;SET BIT 7 IN $60
BD30  20 73 00    L542  JSR CHRGET       ;GET NEXT CHARACTER
BD33  90 5C       L543  BCC L168         ;BRANCH IF NUMERIC TO 'E' ROUTINE
BD35  24 60       L541  BIT FACEXP-1     ;IS BIT 7 SET
BD37  10 0E             BPL L539        ;NO
BD39  A9 00             LDA #$00
BD3B  38                SEC
BD3C  E5 5E             SBC FACEXP-3
BD3E  4C 49 BD          JMP L540
BD41  66 5F       L544  ROR FACEXP-2     ;FOUND DECIMAL POINT
BD43  24 5F             BIT FACEXP-2
BD45  50 C3             BVC L535        ;GET NEXT CHARACTER
BD47  A5 5E       L539  LDA FACEXP-3
BD49  38          L540  SEC
BD4A  E5 5D             SBC FACEXP-4
BD4C  85 5E             STA FACEXP-3
BD4E  F0 12             BEQ L548
BD50  10 09             BPL L550
BD52  20 FE BA    L546  JSR L505         ;FAC#1 = FAC#1 / 10
BD55  E6 5E             INC FACEXP-3    ;LOOP COUNTER
BD57  D0 F9             BNE L546        ;DO AGAIN
BD59  F0 07             BEQ L548        ;JUMP ALWAYS
BD5B  20 E2 BA    L550  JSR L502         ;FAC#1 = FAC#1 * 10
BD5E  C6 5E             DEC FACEXP-3    ;LOOP COUNTER
BD60  D0 F9             BNE L550        ;DO AGAIN
BD62  A5 67       L548  LDA SGNFLG       ;TEST FLAG FOR NEGATIVE
BD64  30 01             BMI L547        ;IF YES INVERT FAC#1 AND EXIT
BD66  60                RTS             ;EXIT
BD67  4C B4 BF    L547  JMP L591         ;INVERT FAC#1 AND EXIT
BD6A  48          L551  PHA             ;SAVE CHARACTER ON STACK
BD6B  24 5F             BIT FACEXP-2    ;HAS A DECIMAL POINT ALREADY
BD6D  10 02             BPL L538        ;BEEN ENCOUNTERED, NO
BD6F  E6 5D             INC FACEXP-4    ;YES THEN INCREMENT DECIMAL PLACES
BD71  20 E2 BA    L538  JSR L502         ;MULTIPLY FAC#1 BY 10
BD74  68                PLA             ;GET CHARACTER OFF STACK
BD75  38                SEC
BD76  E9 30             SBC #$30        ;CONVERT FROM ASCII TO NUMERIC VALUE
BD78  20 7E BD          JSR L553        ;ADD NEXT DIGIT TO FAC#1
BD7B  4C 0A BD          JMP L535        ;REPEAT FOR NEXT CHARACTER
BD7E  48          L553  PHA             ;PUSH VALUE ONTO STACK
BD7F  20 0C BC          JSR L516        ;TRANSFER FAC#1 TO FAC#2
BD82  68                PLA             ;RESTORE DIGIT
BD83  20 3C BC          JSR L521        ;SET UP FAC#1 FOR 1 DIGIT
BD86  A5 6E             LDA ARGSGN      ;GET FAC#2 SIGN
BD88  45 66             EOR FACSGN      ;EXCLUSIVE OR WITH FAC#1 SIGN
BD8A  85 6F             STA ARISGN      ;STORE IN SIGN COMPARISON REGISTER
BD8C  A6 61             LDX FACEXP      ;GET FAC#1 EXPONENT
```

```
LOC   CODE        LINE

BD8E  4C 6A B8          JMP L251        ;PERFORM ADD
BD91                ;ROUTINE TO SET EXPONENT VALUE
BD91  A5 5E         L168  LDA FACEXP-3    ;GET EXPONENT COUNTER
BD93  C9 0A             CMP #$0A        ;COMPARE WITH 10
BD95  90 09             BCC L545        ;BRANCH IF LESS THAN 10
BD97  A9 64             LDA #FACHO+2    ;IF MINUS THEN NEGATIVE EXPONENT
BD99  24 60             BIT FACEXP-1
BD9B  30 11             BMI L554
BD9D  4C 7E B9          JMP L481        ;GENERATE 'OVERFLOW ERROR'
BDA0  0A            L545  ASL A           ;MULTIPLY EXPONENT COUNTER BY 10
BDA1  0A                ASL A
BDA2  18                CLC
BDA3  65 5E             ADC FACEXP-3
BDA5  0A                ASL A
BDA6  18                CLC
BDA7  A0 00             LDY #$00
BDA9  71 7A             ADC (TXTPTR),Y ;ADD ASCII DIGIT
BDAB  38                SEC
BDAC  E9 30             SBC #$30        ;AND REMOVE ASCII OFFSET
BDAE  85 5E         L554  STA FACEXP-3    ;STORE
BDB0  4C 30 BD          JMP L542        ;GET NEXT CHARACTER
BDB3                    .END
BDB3                    .LIB B13
BDB3                ;***************************************
BDB3                ;*CONSTANTS FOR STRING CONVERSION.
BDB3                ;***************************************
BDB3                ;99999999.9
BDB3  9B                .BYT $9B,$3E,$BC,$1F,$FD
BDB4  3E
BDB5  BC
BDB6  1F
BDB7  FD
BDB8                ;999999999
BDB8  9E                .BYT $9E,$6E,$6B,$27,$FD
BDB9  6E
BDBA  6B
BDBB  27
BDBC  FD
BDBD                ;1000000000
BDBD  9E                .BYT $9E,$6E,$6B,$28,$00
BDBE  6E
BDBF  6B
BDC0  28
BDC1  00
BDC2                ;***************************************
BDC2                ;*PRINT 'IN' THEN BASIC LINE NUMBER.
BDC2                ;*THE MESSAGE 'IN' IS PRINTED FOLLOWED BY
BDC2                ;*THE LINE NUMBER OF THE CURRENT BASIC
BDC2                ;*LINE, THIS IS STORED IN LOCATIONS $39,$3A.
BDC2                ;*THE LINE NUMBER IS TRANSFERED TO THE
BDC2                ;*ACCUMULATOR (MSB) AND .X INDEX REGISTER
BDC2                ;*(LSB), THE ROUTINE AT $BDCD THEN PRINTS
BDC2                ;*256*ACCUMULATOR +.X ON THE OUTPUT DEVICE.
BDC2                ;***************************************
BDC2  A9 71         L555  LDA #FBUFPT     ;POINTER TO 'IN' STRING LSB
BDC4  A0 A3             LDY #$A3        ;MSB
BDC6  20 DA BD          JSR L80         ;PRINT STRING
BDC9  A5 3A             LDA CURLIN+1    ;GET CURRENT LINE NUMBER  MSB
BDCB  A6 39             LDX CURLIN      ;LSB
BDCD                ;
BDCD                ;PRINT INTEGER NUMBER IN .A,.X
```

```
LOC   CODE        LINE

BDCD              ;
BDCD  85 62       L24     STA FACHO        ;SAVE NUMBER MSB IN FAC#1 AREA
BDCF  86 63               STX FACHO+1      ;SAVE LSB
BDD1  A2 90               LDX #$90
BDD3  38                  SEC
BDD4  20 49 BC            JSR L386         ;CONVERT INTEGER TO FLOATING POINT
BDD7  20 DF BD            JSR L187         ;CONVERT FAC#1 TO ASCII STRING
BDDA  4C 1E AB    L80     JMP L198         ;PRINT STRING
BDDD              ;***************************************
BDDD              ;*CONVERT FLOATING POINT TO ASCII.
BDDD              ;*THE CONTENTS OF FAC#1 ARE CONVERTED TO AN
BDDD              ;*ASCII STRING STORED IN A BUFFER STARTING
BDDD              ;*AT LOCATION $0100. ON EXIT FROM THE ROUTINE
BDDD              ;*A ZERO TERMINATING BYTE IS PLACED AT THE
BDDD              ;*END OF THE STRING AND THE BUFFER STARTING
BDDD              ;*ADDRESS IS STORED IN THE ACCUMULATOR (LSB)
BDDD              ;*AND .Y INDEX REGISTER (MSB).
BDDD              ;***************************************
BDDD  A0 01       L556    LDY #$01         ;SET POINTER TO BUFFER
BDDF  A9 20       L187    LDA #$20         ;LEADING SPACE FOR POSITIVE VALUES
BDE1  24 66               BIT FACSGN       ;IS NUMBER NEGATIVE
BDE3  10 02               BPL L394         ;NO IT IS POSITIVE
BDE5  A9 2D               LDA #$2D         ;'-' CHARACTER FOR NEGATIVE VALUES
BDE7  99 FF 00    L394    STA BAD-1,Y      ;STORE FIRST CHARACTER IN BUFFER
BDEA  85 66               STA FACSGN       ;STORE SIGN CHARACTER
BDEC  84 71               STY FBUFPT       ;SAVE BUFFER POINTER
BDEE  C8                  INY
BDEF  A9 30               LDA #$30         ;ZERO FOR STRING TERMINATOR
BDF1  A6 61               LDX FACEXP       ;GET EXPONENT OF FAC#1
BDF3  D0 03               BNE L558         ;EXPONENT NOT ZERO
BDF5  4C 04 BF            JMP L586         ;EXPONENT IS ZERO THEN EXIT ROUTINE
BDF8  A9 00       L558    LDA #$00
BDFA  E0 80               CPX #$80         ;COMPARE EXPONENT OF FAC#1
BDFC  F0 02               BEQ L559         ;EXPONENT IS 1
BDFE  B0 09               BCS L561         ;EXPONENT > 1
BE00  A9 BD       L559    LDA #$BD         ;POINTER TO CONSTANT 1000000000 LSB
BE02  A0 BD               LDY #$BD         ;POINTER MSB
BE04  20 28 BA            JSR L487         ;MULTIPLY CONSTANT BY FAC#1
BE07  A9 F7               LDA #$F7
BE09  85 5D       L561    STA FACEXP-4     ;SET POINTER
BE0B  A9 B8       L562    LDA #$B8         ;POINTER TO CONSTANT 999999999 LSB
BE0D  A0 BD               LDY #$BD         ;POINTER MSB
BE0F  20 5B BC            JSR L308         ;COMPARE CONSTANT WITH FAC#1
BE12  F0 1E               BEQ L567         ;EQUAL
BE14  10 12               BPL L566         ;GREATER THAN
BE16  A9 B3       L569    LDA #$B3         ;POINTER TO CONSTANT 99999999.9 LSB
BE18  A0 BD               LDY #$BD         ;POINTER MSB
BE1A  20 5B BC            JSR L308         ;COMPARE CONSTANT WITH FAC#1
BE1D  F0 02               BEQ L568         ;EQUAL
BE1F  10 0E               BPL L565         ;GREATER THAN
BE21  20 E2 BA    L568    JSR L502         ;MULTIPLY FAC#1 BY 10
BE24  C6 5D               DEC FACEXP-4     ;DECREMENT POINTER
BE26  D0 EE               BNE L569         ;IF NOT ZERO DO AGAIN
BE28  20 FE BA    L566    JSR L505         ;DIVIDE FAC#1 BY 10
BE2B  E6 5D               INC FACEXP-4     ;INCREMENT POINTER
BE2D  D0 DC               BNE L562         ;IF NOT ZERO LOOP AGAIN
BE2F  20 49 B8    L565    JSR L466         ;ADD 0.5 TO FAC#1 (ROUNDING)
BE32  20 9B BC    L567    JSR L526         ;CONVERT FLOATING POINT TO INTEGER
BE35  A2 01               LDX #$01
BE37  A5 5D               LDA FACEXP-4     ;GET POINTER
BE39  18                  CLC
```

```
LOC   CODE        LINE

BE3A  69 0A             ADC #$0A
BE3C  30 09             BMI L564        ;IN RANGE
BE3E  C9 0B             CMP #$0B
BE40  B0 06             BCS L571        ;<= 0.1
BE42  69 FF             ADC #$FF
BE44  AA                TAX
BE45  A9 02             LDA #$02
BE47  38          L564  SEC
BE48  E9 02       L571  SBC #$02         ;CHECK IF 'E' VALUE NECESSARY
BE4A  85 5E             STA FACEXP-3    ;SAVE EXPONENT
BE4C  86 5D             STX FACEXP-4    ;SAVE DECIMAL POINT FLAG
BE4E  8A                TXA
BE4F  F0 02             BEQ L572        ;IF <=0, STORE DECIMAL POINT
BE51  10 13             BPL L575        ;CALCULATION OF DIGITS
BE53  A4 71       L572  LDY FBUFPT       ;GET STRING BUFFER POINTER
BE55  A9 2E             LDA #$2E        ;CHARACTER FOR DECIMAL POINT
BE57  C8                INY             ;INCREMENT BUFFER POINTER
BE58  99 FF 00          STA BAD-1,Y     ;STORE DECIMAL POINT IN STRING
BE5B  8A                TXA             ;GET EXPONENT VALUE BACK
BE5C  F0 06             BEQ L573        ;BRANCH IF EXPONENT IS ZERO
BE5E  A9 30             LDA #$30        ;STORE CHARACTER ZERO INTO STRING
BE60  C8                INY             ;INCREMENT BUFFER POINTER
BE61  99 FF 00          STA BAD-1,Y     ;STORE IN BUFFER
BE64  84 71       L573  STY FBUFPT       ;SAVE BUFFER POINTER
BE66                    ;
BE66                    ;CALCULATE SEPERATE DIGITS
BE66                    ;
BE66  A0 00       L575  LDY #$00         ;INITIALISE POINTER TO CONSTANT
BE68  A2 80       L574  LDX #$80
BE6A  A5 65       L303  LDA FACHO+3      ;GET LSB OF FAC#1
BE6C  18                CLC
BE6D  79 19 BF          ADC $BF19,Y     ;AND ADD TO CONSTANT LSB
BE70  85 65             STA FACHO+3     ;STORE IN FAC#1 LSB
BE72  A5 64             LDA FACHO+2
BE74  79 18 BF          ADC $BF18,Y
BE77  85 64             STA FACHO+2
BE79  A5 63             LDA FACHO+1
BE7B  79 17 BF          ADC $BF17,Y
BE7E  85 63             STA FACHO+1
BE80  A5 62             LDA FACHO       ;GET FAC#1 MSB
BE82  79 16 BF          ADC $BF16,Y     ;ADD TO CONSTANT MSB
BE85  85 62             STA FACHO       ;STORE IN FAC#1 MSB
BE87  E8                INX
BE88  B0 04             BCS L577        ;OVERFLOW FROM ADDITION
BE8A  10 DE             BPL L303        ;REPEAT LOOP
BE8C  30 02             BMI L576        ;IF .X >127 AND NO OVERFLOW EXIT LOOP
BE8E  30 DA       L577  BMI L303         ;REPEAT LOOP
BE90  8A          L576  TXA
BE91  90 04             BCC L578
BE93  49 FF             EOR #$FF        ;INVERT AND
BE95  69 0A             ADC #$0A        ;ADD 10
BE97  69 2F       L578  ADC #$2F         ;ADD ASCII OFFSET
BE99  C8                INY             ;INCREMENT POINTER TO NEXT CONSTANT
BE9A  C8                INY
BE9B  C8                INY
BE9C  C8                INY
BE9D  84 47             STY VARPNT      ;AND SAVE POINTER TO CONSTANT TABLE
BE9F  A4 71             LDY FBUFPT      ;GET STRING POINTER
BEA1  C8                INY             ;INCREMENT STRING POINTER
BEA2  AA                TAX
BEA3  29 7F             AND #$7F        ;MASK OUT ANY SIGN ELEMENT
```

```
LOC   CODE        LINE

BEA5  99 FF 00          STA BAD-1,Y     ;AND STORE IN BUFFER
BEA8  C6 5D             DEC FACEXP-4    ;DECREMENT DECIMAL POINT COUNTER
BEAA  D0 06             BNE L579        ;IF NOT ZERO THEN NO DECIMAL POINT
BEAC  A9 2E             LDA #$2E        ;DECIMAL POINT CHARACTER
BEAE  C8                INY             ;INCREMENT BUFFER POINTER
BEAF  99 FF 00          STA BAD-1,Y     ;STORE DECIMAL POINT IN BUFFER
BEB2  84 71       L579  STY FBUFPT       ;SAVE BUFFER POINTER
BEB4  A4 47             LDY VARPNT      ;GET CONSTANT POINTER
BEB6  8A                TXA
BEB7  49 FF             EOR #$FF        ;INVERT
BEB9  29 80             AND #$80        ;AND MASK OFF ALL BUT HIGH BIT
BEBB  AA                TAX
BEBC  C0 24             CPY #$24        ;END OF FIRST CONSTANT TABLE?
BEBE  F0 04             BEQ L580        ;YES
BEC0  C0 3C             CPY #$3C        ;END OF TI$ CONSTANT TABLE
BEC2  D0 A6             BNE L303        ;NO SO REPEAT LOOP
BEC4  A4 71       L580  LDY FBUFPT       ;GET BUFFER POINTER
BEC6  B9 FF 00    L581  LDA BAD-1,Y      ;GET LAST CHARACTER STORED
BEC9  88                DEY             ;POINT TO PREVIOUS BUFFER ENTRY
BECA  C9 30             CMP #$30        ;IS LAST CHARACTER ZERO
BECC  F0 F8             BEQ L581        ;YES THEN REPEAT
BECE  C9 2E             CMP #$2E        ;IS LAST CHARACTER DECIMAL POINT
BED0  F0 01             BEQ L582        ;YES
BED2  C8                INY             ;RESET BUFFER POINTER
BED3  A9 2B       L582  LDA #$2B         ;ASCII '+' CHARACTER
BED5  A6 5E             LDX FACEXP-3    ;GET 'E' FLAG
BED7  F0 2E             BEQ L560        ;IF NO 'E' REQUIRED END
BED9  10 08             BPL L583        ;BRANCH IF POSITIVE
BEDB  A9 00             LDA #$00
BEDD  38                SEC
BEDE  E5 5E             SBC FACEXP-3    ;CONVERT TO POSITIVE NUMBER
BEE0  AA                TAX
BEE1  A9 2D             LDA #$2D        ;ASCII '-' CHARACTER
BEE3  99 01 01    L583  STA BAD+1,Y      ;SAVE IN BUFFER
BEE6  A9 45             LDA #$45        ;ASCII CHARACTER 'E'
BEE8  99 00 01          STA BAD,Y       ;SAVE BEFORE EXPONENT SIGN
BEEB  8A                TXA
BEEC  A2 2F             LDX #$2F        ;'E' TENS DIGIT
BEEE  38                SEC
BEEF  E8          L585  INX              ;INCREMENT TENS DIGIT
BEF0  E9 0A             SBC #$0A        ;SUBTRACT 10 FROM NUMBER
BEF2  B0 FB             BCS L585        ;STILL MORE THAN 10
BEF4  69 3A             ADC #$3A        ;ADD ASCII OFFSET +10 TO UNITS DIGIT
BEF6  99 03 01          STA BAD+3,Y     ;STORE UNITS DIGIT
BEF9  8A                TXA             ;GET TENS DIGIT
BEFA  99 02 01          STA BAD+2,Y     ;STORE TENS DIGIT
BEFD  A9 00             LDA #$00        ;GET STRING TERMINATOR
BEFF  99 04 01          STA BAD+4,Y     ;STORE TERMINATOR
BF02  F0 08             BEQ L584        ;JUMP ALWAYS
BF04  99 FF 00    L586  STA BAD-1,Y      ;STORE ASCII ZERO FOR VALUE =0
BF07  A9 00       L560  LDA #$00         ;GET STRING TERMINATOR
BF09  99 00 01          STA BAD,Y       ;STORE TERMINATOR
BF0C  A9 00       L584  LDA #$00         ;POINTER TO STRING LSB
BF0E  A0 01             LDY #$01        ;POINTER TO STRING MSB
BF10  60                RTS             ;END
BF11              ;****************************************
BF11              ;*CONSTANT 0.5 IN FLOATING POINT FOR 'SQR'
BF11              ;****************************************
BF11  80                .BYT $80,$00,$00,$00,$00
BF12  00
BF13  00
```

```
LOC   CODE        LINE

BF14  00
BF15  00
BF16              ;****************************************
BF16              ;*CONSTANTS FOR FLOATING POINT TO ASCII
BF16              ;*CONVERSION ROUTINE.
BF16              ;****************************************
BF16                                  ;-100,000,000
BF16  FA                    .BYT $FA,$0A,$1F,$00
BF17  0A
BF18  1F
BF19  00
BF1A                                  ;  10,000,000
BF1A  00                    .BYT $00,$98,$96,$80
BF1B  98
BF1C  96
BF1D  80
BF1E                                  ;  -1,000,000
BF1E  FF                    .BYT $FF,$F0,$BD,$C0
BF1F  F0
BF20  BD
BF21  C0
BF22                                  ;      100,000
BF22  00                    .BYT $00,$01,$86,$A0
BF23  01
BF24  86
BF25  A0
BF26                                  ;      -10,000
BF26  FF                    .BYT $FF,$FF,$D8,$F0
BF27  FF
BF28  D8
BF29  F0
BF2A                                  ;        1,000
BF2A  00                    .BYT $00,$00,$03,$E8
BF2B  00
BF2C  03
BF2D  E8
BF2E                                  ;         -100
BF2E  FF                    .BYT $FF,$FF,$FF,$9C
BF2F  FF
BF30  FF
BF31  9C
BF32                                  ;           10
BF32  00                    .BYT $00,$00,$00,$0A
BF33  00
BF34  00
BF35  0A
BF36                                  ;           -1
BF36  FF                    .BYT $FF,$FF,$FF,$FF
BF37  FF
BF38  FF
BF39  FF
BF3A              ;****************************************
BF3A              ;*CONSTANTS TO CHANGE TI TO TI$
BF3A              ;****************************************
BF3A                                  ;-2,160,001
BF3A  FF                    .BYT $FF,$DF,$0A,$80
BF3B  DF
BF3C  0A
BF3D  80
BF3E                                  ;  216,000 (1 HOUR)
BF3E  00                    .BYT $00,$03,$4B,$C0
```

```
LOC   CODE        LINE

BF3F  03
BF40  4B
BF41  C0
BF42                              ;   -36,000
BF42  FF                  .BYT $FF,$FF,$73,$60
BF43  FF
BF44  73
BF45  60
BF46                              ;     3,600 (1 MINUTE)
BF46  00                  .BYT $00,$00,$0E,$10
BF47  00
BF48  0E
BF49  10
BF4A                              ;      -600
BF4A  FF                  .BYT $FF,$FF,$FD,$A8
BF4B  FF
BF4C  FD
BF4D  A8
BF4E                              ;        60 (1 SECOND)
BF4E  00                  .BYT $00,$00,$00,$3C
BF4F  00
BF50  00
BF51  3C
BF52  EC                  .BYT $EC
BF53  AA                  .BYT $AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA
BF54  AA
BF55  AA
BF56  AA
BF57  AA
BF58  AA
BF59  AA
BF5A  AA
BF5B  AA                  .BYT $AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA
BF5C  AA
BF5D  AA
BF5E  AA
BF5F  AA
BF60  AA
BF61  AA
BF62  AA
BF63  AA                  .BYT $AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA
BF64  AA
BF65  AA
BF66  AA
BF67  AA
BF68  AA
BF69  AA
BF6A  AA
BF6B  AA                  .BYT $AA,$AA,$AA,$AA,$AA,$AA
BF6C  AA
BF6D  AA
BF6E  AA
BF6F  AA
BF70  AA
BF71              ;****************************************
BF71              ;*PERFORM SQR.
BF71              ;*THE CONTENTS OF FAC#1 (THE ARGUMENT) ARE
BF71              ;*TRANSFERED TO FAC#2, FAC#1 IS THEN LOADED
BF71              ;*WITH .5 AND THE ROUTINE JUMPS TO THE PERFORM
BF71              ;*POWER ROUTINE AT $BF78. THE RESULT IS STORED
BF71              ;*IN FAC#1.
```

```
LOC   CODE        LINE

BF71              ;****************************************
BF71  20 0C BC    SQR     JSR L516         ;ROUND FAC#1 AND PUT IN FAC#2
BF74  A9 11               LDA #$11         ;POINTER TO CONSTANT 0.5 LSB
BF76  A0 BF               LDY #$BF         ;POINTER MSB
BF78              ;*************************************
BF78              ;*PERFORM POWER FUNCTION.
BF78              ;*THE CONTENTS OF FAC#2 ARE RAISED TO THE
BF78              ;*POWER OF THE VALUE STORED IN FAC#1. BEFORE
BF78              ;*USING THIS ROUTINE FAC#2 MUST BE LOADED.
BF78              ;*IF EITHER VALUE IS ZERO THEN FAC#1 IS
BF78              ;*LOADED WITH EITHER 0 OR 1 DEPENDING ON
BF78              ;*WHICH FAC WAS ZERO. THE EVALUATION IS
BF78              ;*PERFORMED BY SAVING FAC#1 TO ZERO PAGE AND
BF78              ;*THEN MULTIPLYING THE LOGARITHM OF FAC#2 BY
BF78              ;*FAC#1 AND GETTING THE EXPONENT OF THE RESULT.
BF78              ;*THERE ARE TWO ENTRY POINTS TO THIS ROUTINE,
BF78              ;*THE FIRST AT $BF78 RAISES FAC#2 TO THE POWER
BF78              ;*OF A CONSTANT STORED IN MEMORY AND POINTED
BF78              ;*TO BY .A (LSB) AND .Y (MSB). THE SECOND
BF78              ;*ENTRY POINT REQUIRES THE VALUES TO BE IN
BF78              ;*FAC#1 AND FAC#2.
BF78              ;*************************************
BF78  20 A2 BB    POWER   JSR L496         ;PERFORM EXP FUNCTION
BF7B  F0 70               BEQ L595
BF7D  A5 69               LDA ARGEXP       ;GET FAC#2 EXPONENT
BF7F  D0 03               BNE L587         ;NOT ZERO
BF81  4C F9 B8            JMP L454         ;0 IN SIGN & EXPONENT OF FAC#1, EXIT
BF84  A2 4E       L587    LDX #$4E         ;POINTER LSB TO TEMPORARY FAC
BF86  A0 00               LDY #$00         ;MSB TO TEMPORARY FAC
BF88  20 D4 BB            JSR L154         ;TRANSFER FAC#1 TO TEMPORARY FAC
BF8B  A5 6E               LDA ARGSGN       ;GET FAC#2 SIGN
BF8D  10 0F               BPL L589         ;BRANCH IF NUMBER IS POSITIVE
BF8F  20 CC BC            JSR L531         ;PERFORM 'INT' FUNCTION
BF92  A9 4E               LDA #$4E         ;POINTER TO TEMPORARY FAC LSB
BF94  A0 00               LDY #$00         ;MSB
BF96  20 5B BC            JSR L308         ;COMPARE FAC#1 TO TEMPORARY FAC
BF99  D0 03               BNE L589         ;DIFFERENT
BF9B  98                  TYA
BF9C  A4 07               LDY CHARAC
BF9E  20 FE BB    L589    JSR L314         ;TRANSFER FAC#2 TO FAC#1
BFA1  98                  TYA
BFA2  48                  PHA              ;SAVE .Y TO STACK
BFA3  20 EA B9            JSR L483         ;PERFORM LOG OPERATION
BFA6  A9 4E               LDA #$4E         ;POINTER TO TEMPORARY FAC LSB
BFA8  A0 00               LDY #$00         ;MSB
BFAA  20 28 BA            JSR L487         ;MULTIPLY LOG OF FAC#2 WITH FAC#1
BFAD  20 ED BF            JSR L595         ;PERFORM EXP OPERATION
BFB0  68                  PLA
BFB1  4A                  LSR A            ;ROTATE LSB TO CARRY
BFB2  90 0A               BCC L552         ;IF SET BRANCH
BFB4  A5 61       L591    LDA FACEXP       ;GET EXPONENT OF FAC#1
BFB6  F0 06               BEQ L552         ;END IF ZERO
BFB8  A5 66               LDA FACSGN       ;GET SIGN OF FAC#1
BFBA  49 FF               EOR #$FF         ;INVERT IT
BFBC  85 66               STA FACSGN       ;AND STORE NEW SIGN
BFBE  60          L552    RTS
BFBF              ;*************************************
BFBF              ;*CONSTANTS USED BY 'EXP' & 'LOG' ROUTINES.
BFBF              ;*************************************
BFBF              ;1.44269504     (1/LN(2))
BFBF  81                  .BYT $81,$38,$AA,$3B,$29
```

```
LOC   CODE        LINE

BFC0  38
BFC1  AA
BFC2  3B
BFC3  29
BFC4              ;
BFC4  07                .BYT $07
BFC5              ;2.14987637E-5
BFC5  71                .BYT $71,$34,$58,$3E,$56
BFC6  34
BFC7  58
BFC8  3E
BFC9  56
BFCA              ;1.4352314 E-4
BFCA  74                .BYT $74,$16,$7E,$B3,$1B
BFCB  16
BFCC  7E
BFCD  B3
BFCE  1B
BFCF              ;1.34226348E-3
BFCF  77                .BYT $77,$2F,$EE,$E3,$85
BFD0  2F
BFD1  EE
BFD2  E3
BFD3  85
BFD4              ;9.6140117 E-3
BFD4  7A                .BYT $7A,$1D,$84,$1C,$2A
BFD5  1D
BFD6  84
BFD7  1C
BFD8  2A
BFD9              ;0.05505127
BFD9  7C                .BYT $7C,$63,$59,$58,$0A
BFDA  63
BFDB  59
BFDC  58
BFDD  0A
BFDE              ;0.24022639
BFDE  7E                .BYT $7E,$75,$FD,$E7,$C6
BFDF  75
BFE0  FD
BFE1  E7
BFE2  C6
BFE3              ;0.69314719
BFE3  80                .BYT $80,$31,$72,$18,$10
BFE4  31
BFE5  72
BFE6  18
BFE7  10
BFE8              ;1
BFE8  81                .BYT $81,$00,$00,$00,$00
BFE9  00
BFEA  00
BFEB  00
BFEC  00
BFED              ;*****************************************
BFED              ;*PERFORM 'EXP' FUNCTION.
BFED              ;*THE VALUE OF E TO THE POWER OF THE CONTENTS
BFED              ;*OF FAC#1 IS CALCULATED AND THE RESULT
BFED              ;*IS STORED IN FAC#1.
BFED              ;*****************************************
BFED  A9 BF       L595   LDA #$BF         ;POINTER TO CONSTANT 1/LOG(2) LSB
```

```
LOC   CODE        LINE

BFEF  A0 BF             LDY #$BF        ;POINTER MSB
BFF1  20 28 BA          JSR L487        ;MULTIPLY CONSTANT WITH FAC#1
BFF4  A5 70             LDA FACOV       ;GET LOW ORDER ROUNDING BYTE
BFF6  69 50             ADC #$50        ;AND ADD IT TO 80
BFF8  90 03             BCC L588        ;CONTENTS OF $70>=$B0
BFFA  20 23 BC          JSR L152        ;INCREASE MANTISSA OF FAC#1 BY 1
BFFD  4C 00 E0    L588  JMP L596         ;JUMP TO UPPER MEMORY PARTITION
C000                    * =$E000
E000  85 56       L596  STA $56         ;SAVE .A IN FUNCTION JUMP VECTOR
E002  20 0F BC          JSR L160        ;TRANSFER FAC#1 TO FAC#2
E005  A5 61             LDA FACEXP      ;GET FAC#1 EXPONENT
E007  C9 88             CMP #$88        ;LARGER THAN 128?
E009  90 03             BCC L602        ;SMALLER
E00B  20 D4 BA    L598  JSR L503         ;GENERATE 'OVERFLOW' MESSAGE
E00E  20 CC BC    L602  JSR L531         ;PERFORM 'INT' OPERATION
E011  A5 07             LDA CHARAC
E013  18                CLC
E014  69 81             ADC #$81
E016  F0 F3             BEQ L598        ;EQUAL TO 127
E018  38                SEC
E019  E9 01             SBC #$01
E01B  48                PHA
E01C  A2 05             LDX #$05        ;SET POINTER TO # OF FAC LOCATIONS
E01E  B5 69       L600  LDA ARGEXP,X     ;SWITCH FAC#1 TO FAC#2
E020  B4 61             LDY FACEXP,X
E022  95 61             STA FACEXP,X
E024  94 69             STY ARGEXP,X
E026  CA                DEX             ;DECREMENT POINTER
E027  10 F5             BPL L600        ;REPEAT TILL ALL DONE
E029  A5 56             LDA $56         ;GET TEMP FAC LOW ORDER AND
E02B  85 70             STA FACOV       ;STORE TO FAC#1 LOW ORDER
E02D  20 53 B8          JSR L489        ;FAC#2 - FAC#1
E030  20 B4 BF          JSR L591        ;PERFORM NEGATIVE
E033  A9 C4             LDA #$C4        ;POINTER LSB TO CONSTANT 7
E035  A0 BF             LDY #$BF        ;MSB
E037  20 59 E0          JSR L490        ;SERIES EVALUATE 2 ROUTINE
E03A  A9 00             LDA #$00
E03C  85 6F             STA ARISGN      ;CLEAR SIGN COMPARISON FLAG
E03E  68                PLA
E03F  20 B9 BA          JSR L493        ;ADJUST EXPONENTS OF FAC#1 + FAC#2
E042  60                RTS             ;EXIT
E043              ;***************************************
E043              ;*SERIES EVALUATION.
E043              ;*A FUNCTION IS EVALUATED BY THIS ROUTINE
E043              ;*AND THE RESULT STORED IN FAC#1. ON ENTRY
E043              ;*THE ACCUMULATOR (MSB) AND .Y INDEX REGISTER
E043              ;*(LSB) POINT TO THE START ADDRESS OF THE
E043              ;*FIRST VARIABLE TO BE EVALUATED IN THE
E043              ;*SERIES. THIS USES FAC#1 AND WORK SPACE IN
E043              ;*$71AND$72.
E043              ;***************************************
E043              ;
E043              ;SERIES 1 EVALUATION
E043              ;
E043  85 71       L603  STA FBUFPT       ;SAVE ADDRESS POINTER MSB
E045  84 72             STY FBUFPT+1    ;LSB
E047  20 CA BB          JSR L609        ;TRANSFER FAC#1 TO MEMORY AT .A,.Y
E04A  A9 57             LDA #$57        ;POINTER TO TEMPORARY FAC
E04C  20 28 BA          JSR L487        ;MULTIPLY FAC#1 BY TEMPORARY FAC
E04F  20 5D E0          JSR L605        ;PERFORM SERIES 2 EVALUATION
E052  A9 57             LDA #$57        ;POINTER TO TEMPORARY FAC LSB
```

```
LOC   CODE        LINE

E054  A0 00             LDY #$00        ;MSB
E056  4C 28 BA          JMP L487        ;MULTIPLY FAC#1 BY TEMPORARY FAC
E059                    ;
E059                    ;SERIES 2 EVALUATION
E059                    ;
E059  85 71       L490  STA FBUFPT       ;SAVE POINTER TO VALUE LSB
E05B  84 72             STY FBUFPT+1    ;MSB
E05D  20 C7 BB    L605  JSR L98         ;TRANSFER FAC#1 TO TEMPORARY FAC
E060  B1 71             LDA (FBUFPT),Y
E062  85 67             STA SGNFLG      ;SAVE SERIES EVALUATION POINTER
E064  A4 71             LDY FBUFPT      ;GET LOW BYTE POINTER AND
E066  C8                INY             ;INCREMENT
E067  98                TYA             ;TRANSFER TO ACCUMULATOR
E068  D0 02             BNE L608
E06A  E6 72             INC FBUFPT+1    ;INCREMENT HIGH BYTE POINTER
E06C  85 71       L608  STA FBUFPT       ;SAVE LSB POINTER
E06E  A4 72             LDY FBUFPT+1    ;SAVE MSB POINTER
E070  20 28 BA    L610  JSR L487         ;MULTIPLY FAC#1 BY TEMPORARY FAC
E073  A5 71             LDA FBUFPT      ;GET POINTER LSB
E075  A4 72             LDY FBUFPT+1    ;MSB
E077  18                CLC
E078  69 05             ADC #$05        ;ADD 5 TO LSB
E07A  90 01             BCC L612
E07C  C8                INY             ;INCREMENT MSB
E07D  85 71       L612  STA FBUFPT       ;SAVE POINTER LSB
E07F  34 72             STY FBUFPT+1    ;MSB
E081  20 67 B8          JSR L469        ;FAC#1 = FAC#1 + TEMPORARY FAC
E084  A9 5C             LDA #$5C        ;POINTER TO TEMPORARY FAC LSB
E086  A0 00             LDY #$00        ;MSB
E088  C6 67             DEC SGNFLG      ;DECREMENT SERIES EVALUATION POINTER
E08A  D0 E4             BNE L610        ;IF NOT ZERO REPEAT
E08C  60                RTS
E08D                    ;****************************************
E08D                    ;*CONSTANTS FOR 'RND'.
E08D                    ;****************************************
E08D  98                .BYT $98,$35,$44,$7A,$00
E08E  35
E08F  44
E090  7A
E091  00
E092  68                .BYT $68,$28,$B1,$46,$00
E093  28
E094  B1
E095  46
E096  00
E097                    ;****************************************
E097                    ;*PERFORM 'RND'.
E097                    ;*A RANDOM VALUE IS CREATED BY THIS ROUTINE
E097                    ;*AND STORED IN FAC#1. PRIOR TO RUNNING THIS
E097                    ;*ROUTINE FAC#1 CONTAINS A 'SEED' VALUE USED
E097                    ;*TO INITIALISE THE RANDOM NUMBER CALCULATION
E097                    ;*ROUTINE. THE LAST RANDOM NUMBER GENERATED
E097                    ;*IS STORED IN LOCATIONS $8B TO $8F. IF A
E097                    ;*ZERO ARGUMENT IS GIVEN IN THE 'RND' FUNCTION
E097                    ;*THEN THE VALUE IN THE CIA TIMERS IS USED FOR
E097                    ;*THE SEED.
E097                    ;****************************************
E097  20 2B BC    RND   JSR L597         ;GET SIGN
E09A  30 37             BMI L614        ;IS IT NEGATIVE THEN DO SERIES EVAL 1
E09C  D0 20             BNE L611        ;IF NOT ZERO CALCULATE NEW RND
E09E  20 F3 FF          JSR L192        ;GET ADDRESS OF CIA
```

```
LOC   CODE        LINE

E0A1  86 22             STX INDEX       ;STORE CIA ADDRESS LSB IN POINTER
E0A3  84 23             STY INDEX+1     ;MSB
E0A5  A0 04             LDY #$04
E0A7  B1 22             LDA (INDEX),Y   ;GET TIMER 1 LSB
E0A9  85 62             STA FACHO       ;SAVE IN FAC#1 MSB
E0AB  C8                INY             ;INCREMENT POINTER
E0AC  B1 22             LDA (INDEX),Y   ;GET TIMER 1 MSB
E0AE  85 64             STA FACHO+2     ;SAVE IN FAC#1 2ND BYTE
E0B0  A0 08             LDY #$08        ;POINT TO TIMER 2
E0B2  B1 22             LDA (INDEX),Y   ;GET TIMER 2 LSB
E0B4  85 63             STA FACHO+1     ;SAVE IN FAC#1 3RD BYTE
E0B6  C8                INY             ;INCREMENT POINTER
E0B7  B1 22             LDA (INDEX),Y   ;GET TIMER 2 MSB
E0B9  85 65             STA FACHO+3     ;SAVE IN FAC#1 LSB
E0BB  4C E3 E0          JMP L613        ;JUMP TO VALUE OUTPUT
E0BE  A9 8B       L611  LDA #$8B         ;POINTER TO RND SEED LSB
E0C0  A0 00             LDY #$00        ;MSB
E0C2  20 A2 BB          JSR L496        ;TRANSFER TO FAC#1
E0C5  A9 8D             LDA #$8D        ;POINTER TO FIRST RND CONSTANT LSB
E0C7  A0 E0             LDY #$E0        ;MSB
E0C9  20 28 BA          JSR L487        ;FAC#1 =  FAC#1 * FAC#2
E0CC  A9 92             LDA #$92        ;POINTER TO SECOND RND CONSTANT LSB
E0CE  A0 E0             LDY #$E0        ;MSB
E0D0  20 67 B8          JSR L469        ;FAC#1 = FAC#1 + FAC#2
E0D3  A6 65       L614  LDX FACHO+3      ;INVERT HIGH AND LOW BYTES OF FAC#1
E0D5  A5 62             LDA FACHO
E0D7  85 65             STA FACHO+3
E0D9  86 62             STX FACHO
E0DB  A6 63             LDX FACHO+1     ;INVERT BYTES 2 AND 3 OF FAC#1
E0DD  A5 64             LDA FACHO+2
E0DF  85 63             STA FACHO+1
E0E1  86 64             STX FACHO+2
E0E3  A9 00       L613  LDA #$00
E0E5  85 66             STA FACSGN      ;CLEAR SIGN FAC#1
E0E7  A5 61             LDA FACEXP      ;GET EXPONENT
E0E9  85 70             STA FACOV       ;STORE IN LOW ORDER BYTE OF FAC#1
E0EB  A9 80             LDA #$80
E0ED  85 61             STA FACEXP      ;SET EXPONENT TO RANGE 0 TO 1
E0EF  20 D7 B8          JSR L522        ;ADJUST FAC#1
E0F2  A2 8B             LDX #$8B        ;POINTER TO RANDOM SEED LSB
E0F4  A0 00             LDY #$00        ;MSB
E0F6  4C D4 BB    L616  JMP L154         ;ROUND AND TRANSFER FAC#1 TO SEED
E0F9                    ;***************************************
E0F9                    ;*I/O ERROR MESSAGE HANDLER.
E0F9                    ;*THIS ROUTINE IS CALLED BY ALL BASIC I/O
E0F9                    ;*ROUTINES IF THEY HAVE ENCOUNTERED AN ERROR.
E0F9                    ;*ERROR VALUE IS STORED IN ACCUMULATOR.
E0F9                    ;***************************************
E0F9  C9 F0       L654  CMP #$F0
E0FB  D0 07             BNE L621        ;SEND ERROR MESSAGE
E0FD  84 38             STY MEMTOP+1    ;SET TOP OF MEMORY POINTER MSB
E0FF  86 37             STX MEMTOP      ;LSB
E101  4C 63 A6          JMP L123        ;PERFORM 'CLR' OPERATION
E104  AA          L621  TAX              ;TRANSFER ERROR VALUE TO .X
E105  D0 02             BNE L617        ;IF ZERO THEN 'BREAK'
E107  A2 1E             LDX #$1E        ;ERROR MESSAGE NUMBER
E109  4C 37 A4    L617  JMP L10          ;SEND ERROR MESSAGE
E10C                    ;***************************************
E10C                    ;*BASIC I/O ROUTINES.
E10C                    ;*THE FOLLOWING 5 ROUTINES ARE THE LINKS
E10C                    ;*BETWEEN THE BASIC ROM AND THE KERNAL ROM
```

```
LOC   CODE        LINE

E10C              ;*FOR INPUT AND OUTPUT.
E10C              ;***************************************
E10C              ;*OUTPUT.
E10C              ;*THIS ROUTINE OUTPUTS THE CHARACTER STORED
E10C              ;*IN THE ACCUMULATOR TO THE CURRENT OUTPUT
E10C              ;*DEVICE.
E10C              ;***************************************
E10C  20 D2 FF    L619   JSR L622         ;OUTPUT A CHARACTER IN .A
E10F  B0 E8              BCS L654        ;SEND ERROR MESSAGE IF CARRY SET
E111  60                 RTS
E112              ;***************************************
E112              ;*INPUT.
E112              ;*THIS ROUTINE INPUTS A CHARACTER INTO THE
E112              ;*THE ACCUMULATOR FROM THE CURRENT INPUT
E112              ;*DEVICE. IF THE INPUT DEVICE IS KEYBOARD,
E112              ;*THE CURSOR WILL FLASH AND INPUT UNTIL
E112              ;*THE CARRIAGE RETURN IS PRESSED AND .A
E112              ;*WILL HOLD THE 1ST CHARACTER ONLY.
E112              ;***************************************
E112  20 CF FF    L203   JSR L16          ;INPUT A CHARACTER INTO .A
E115  B0 E2              BCS L654        ;SEND ERROR MESSAGE IF CARRY SET
E117  60                 RTS
E118              ;***************************************
E118              ;*SET OUTPUT DEVICE.
E118              ;*THIS ROUTINE MAKES THE CURRENT OUTPUT
E118              ;*DEVICE EQUAL TO THE VALUE IN THE X INDEX
E118              ;*REGISTER.
E118              ;***************************************
E118  20 AD E4    L44    JSR L673         ;SET OUTPUT DEVICE
E11B  B0 DC              BCS L654        ;SEND ERROR MESSAGE IF CARRY SET
E11D  60                 RTS
E11E              ;***************************************
E11E              ;*SET INPUT DEVICE.
E11E              ;*THIS ROUTINE MAKES THE CURRENT INPUT
E11E              ;*DEVICE EQUAL TO THE VALUE IN THE X INDEX
E11E              ;*REGISTER.
E11E              ;***************************************
E11E  20 C6 FF    L179   JSR L639         ;SET INPUT DEVICE
E121  B0 D6              BCS L654        ;SEND ERROR MESSAGE IF CARRY SET
E123  60                 RTS
E124              ;***************************************
E124              ;*GET A CHARACTER.
E124              ;*THIS ROUTINE GETS A CHARACTER FROM THE
E124              ;*CURRENT INPUT DEVICE. THE CHARACTER IS
E124              ;*STORED IN THE ACCUMULATOR. THIS ROUTINE
E124              ;*WILL GET A CHARACTER FROM THE KEYBOARD
E124              ;*WITHOUT FLASHING THE CURSOR.
E124              ;***************************************
E124  20 E4 FF    L210   JSR L115         ;GET A CHARACTER
E127  B0 D0              BCS L654        ;SEND ERROR MESSAGE IF CARRY SET
E129  60                 RTS
E12A                     .END
E12A                     .LIB B14
E12A              ;***************************************
E12A              ;*PERFORM 'SYS'.
E12A              ;*THIS FIRST GETS A TWO BYTE NUMERIC VALUE
E12A              ;*WHICH IS PUT IN LOCATIONS $14,$15, PUSHES
E12A              ;*ITS RETURN ADDRESS TO THE STACK AND PASSES
E12A              ;*ALL PROCESSOR REGISTERS.
E12A              ;***************************************
E12A  20 8A AD    SYS    JSR L253         ; GET NUMERIC VALUE INTO FAC#1
```

```
LOC   CODE        LINE

E12D  20 F7 B7          JSR L459        ;CONVERT TO TWO BYTE FORMAT
E130  A9 E1             LDA #$E1        ;GET RETURN ADDRESS MSB
E132  48                PHA             ;PUSH TO STACK
E133  A9 46             LDA #$46        ;GET RETURN ADDRESS LSB
E135  48                PHA             ;PUSH TO STACK
E136  AD 0F 03          LDA $030F       ;GET STATUS
E139  48                PHA             ;PUSH TO STACK
E13A  AD 0C 03          LDA $030C       ;GET ACCUMULATOR
E13D  AE 0D 03          LDX $030D       ;GET X REGISTER
E140  AC 0E 03          LDY $030E       ;GET Y REGISTER
E143  28                PLP             ;PULL STATUS OFF STACK
E144  6C 14 00          JMP (LINNUM)    ;JUMP INDIRECT TO ADDRESS
E147  08              SYSRTN PHP         ;RETURN FROM SYS, SAVE STATUS
E148  8D 0C 03          STA $030C       ;SAVE ACCUMULATOR
E14B  8E 0D 03          STX $030D       ;SAVE X REGISTER
E14E  8C 0E 03          STY $030E       ;SAVE Y REGISTER
E151  68                PLA             ;PULL STATUS
E152  8D 0F 03          STA $030F       ;AND SAVE
E155  60                RTS
E156                  ;***************************************
E156                  ;*PERFORM 'SAVE'.
E156                  ;*THIS ROUTINE SAVES A BASIC PROGRAM TO DISK
E156                  ;*OR TAPE. THE START ADDRESS OF THE SAVE IS
E156                  ;*IN LOCATIONS $2B,$2C (BOTTOM OF MEMORY) AND
E156                  ;*THE END ADDRESS OF THE SAVE IS IN LOCATIONS
E156                  ;*$2D,$2E (START OF VARIABLES). THE FILE NAME
E156                  ;*AND DEVICE NUMBER ARE OBTAINED BY THE ROUTINE
E156                  ;*AT $E1D4.
E156                  ;***************************************
E156  20 D4 E1        SAVE    JSR L629        ;GET FILE DETAILS
E159  A6 2D                   LDX VARTAB      ;GET END OF SAVE ADDRESS LSB
E15B  A4 2E                   LDY VARTAB+1    ;MSB
E15D  A9 2B                   LDA #$2B        ;INDIRECT POINTER TO START OF SAVE
E15F  20 D8 FF                JSR L628        ;SAVE FILE
E162  B0 95                   BCS L654        ;SEND ERROR MESSAGE IF CARRY SET
E164  60                      RTS
E165                  ;***************************************
E165                  ;*PERFORM 'VERIFY'.
E165                  ;*THIS ROUTINE SETS THE FLAG FOR VERIFY
E165                  ;*AND CONTINUES WITH THE LOAD ROUTINE. AFTER
E165                  ;*THE KERNAL LOAD/VERIFY ROUTINE HAS BEEN
E165                  ;*CALLED, THE STATUS IS CHECKED TO SEE IF
E165                  ;*THE VERIFY WAS CORRECT AND IF SO PRINTS
E165                  ;*'OK' OTHERWISE 'ERROR'.
E165                  ;***************************************
E165  A9 01           VERIFY LDA #$01         ;FLAG FOR VERIFY
E167  2C                     .BYT $2C         ;SKIP NEXT COMMAND
E168                  ;***************************************
E168                  ;*PERFORM 'LOAD'.
E168                  ;*THIS ROUTINE LOADS A FILE INTO THE COMPUTER
E168                  ;*FROM DISK OR TAPE. AFTER LOADING, IF AN
E168                  ;*ERROR HAS OCCURRED THE ERROR MESSAGE IS PRINTED
E168                  ;*OTHERWISE A CHECK ON DIRECT MODE IS MADE.
E168                  ;*IF IN DIRECT MODE, THE VARIABLE POINTERS
E168                  ;*ARE SET TO THE END OF THE PROGRAM, 'READY'
E168                  ;*IS OUTPUT AND 'CLR' IS PERFORMED. IF IN
E168                  ;*PROGRAM MODE, CHARGET IS RESET TO THE BEGINING
E168                  ;*OF THE PROGRAM, THE PROGRAM IS RE-CHAINED
E168                  ;*AND THE BASIC PROGRAM IS EXECUTED.
E168                  ;***************************************
E168  A9 00           LOAD    LDA #$00        ;FLAG LOAD
```

```
LOC   CODE        LINE

E16A  85 0A             STA VERCKB      ;SAVE IN LOAD/VERIFY FLAG
E16C  20 D4 E1          JSR L629        ;GET FILENAME AND DEVICE #
E16F  A5 0A             LDA VERCKB      ;GET LOAD/VERIFY FLAG
E171  A6 2B             LDX TXTTAB      ;GET START OF LOAD/VERIFY LSB
E173  A4 2C             LDY TXTTAB+1    ;MSB
E175  20 D5 FF          JSR L620        ;LOAD/VERIFY
E178  B0 57             BCS L634        ;SEND ERROR MESSAGE IF CARRY SET
E17A  A5 0A             LDA VERCKB      ;WAS IT LOAD?
E17C  F0 17             BEQ L632        ;YES
E17E  A2 1C             LDX #$1C        ;# FOR 'VERIFY ERROR'
E180  20 B7 FF          JSR L669        ;GET STATUS
E183  29 10             AND #$10        ;ISOLATE ERROR
E185  D0 17             BNE L630        ;THERE IS AN ERROR, OUTPUT IT
E187  A5 7A             LDA TXTPTR      ;GET CHARGET POINTER LSB
E189  C9 02             CMP #$02        ;EQUAL TO 2?
E18B  F0 07             BEQ L224        ;YES, THEN EXIT
E18D  A9 64             LDA #$64        ;POINTER TO 'OK' LSB
E18F  A0 A3             LDY #$A3        ;MSB
E191  4C 1E AB          JMP L198        ;OUTPUT MESSAGE
E194  60          L224  RTS
E195  20 B7 FF    L632  JSR L669         ;GET STATUS
E198  29 BF             AND #$BF        ;REMOVE EOI BIT
E19A  F0 05             BEQ L631        ;NO ERROR
E19C  A2 1D             LDX #$1D        ;# FOR 'LOAD ERROR'
E19E  4C 37 A4    L630  JMP L10          ;OUTPUT ERROR MESSAGE
E1A1  A5 7B       L631  LDA TXTPTR+1     ;GET CHARGET POINTER MSB
E1A3  C9 02             CMP #$02        ;DIRECT MODE?
E1A5  D0 0E             BNE L633        ;NO
E1A7  86 2D             STX VARTAB      ;SAVE END OF LOAD IN
E1A9  84 2E             STY VARTAB+1    ;VARIABLE POINTER
E1AB  A9 76             LDA #$76        ;# FOR 'READY.' LSB
E1AD  A0 A3             LDY #$A3        ;MSB
E1AF  20 1E AB          JSR L198        ;OUTPUT 'READY.'
E1B2  4C 2A A5          JMP L40         ;CLR,RE-CHAIN,WARM START
E1B5  20 8E A6    L633  JSR L70          ;SET CHARGET POINTER TO START OF PROG
E1B8  20 33 A5          JSR L635        ;RE-CHAIN LINES
E1BB  4C 77 A6          JMP L618        ;PERFORM RESTORE AND EXECUTE PROGRAM
E1BE              ;****************************************
E1BE              ;*PERFORM 'OPEN'.
E1BE              ;*THIS ROUTINE OPENS A LOGICAL FILE FOR
E1BE              ;*READING OR WRITING.
E1BE              ;****************************************
E1BE  20 19 E2    OPEN  JSR L647         ;GET PARAMETERS FOR OPEN
E1C1  20 C0 FF          JSR L640        ;OPEN FILE
E1C4  B0 0B             BCS L634        ;SEND ERROR MESSAGE IF CARRY SET
E1C6  60                RTS
E1C7              ;****************************************
E1C7              ;*PERFORM 'CLOSE'.
E1C7              ;*THIS ROUTINE CLOSES A LOGICAL FILE THAT HAS
E1C7              ;*ALREADY BEEN OPENED. NO ERROR WILL BE
E1C7              ;*RETURNED IF THE FILE HAS NOT BEEN OPENED.
E1C7              ;****************************************
E1C7  20 19 E2    CLOSE JSR L647         ;GET PARAMETERS FOR CLOSE
E1CA  A5 49             LDA FORFNT      ;GET LOGICAL FILE NUMBER
E1CC  20 C3 FF          JSR L638        ;CLOSE FILE
E1CF  90 C3             BCC L224        ;NO ERROR IF CARRY CLEAR (RTS)
E1D1  4C F9 E0    L634  JMP L654         ;SEND ERROR MESSAGE
E1D4              ;****************************************
E1D4              ;*GET PARAMETERS FOR LOAD/SAVE/VERIFY.
E1D4              ;*THIS ROUTINE IS CALLED TO OBTAIN THE FILENAME
E1D4              ;*AND DEVICE NUMBER FOR THE ABOVE ROUTINES.
```

```
LOC   CODE        LINE

E1D4              ;*IF THERE IS NOTHING FOLLOWING THE COMMAND,
E1D4              ;*THEN IT WILL DEFAULT TO NO FILENAME WITH
E1D4              ;*THE DEVICE SET TO TAPE.
E1D4              ;**************************************
E1D4  A9 00       L629   LDA #$00         ;DEFAULT NAME LENGTH = 0
E1D6  20 BD FF           JSR L641         ;SET NAME DETAILS
E1D9  A2 01              LDX #$01         ;DEFAULT DEVICE = TAPE
E1DB  A0 00              LDY #$00         ;DEFAULT 2ND ADDRESS = 0
E1DD  20 BA FF           JSR L217         ;SET FILE DETAILS
E1E0  20 06 E2           JSR L644         ;GET NEXT CHARACTER FROM INPUT
E1E3  20 57 E2           JSR L648         ;GET THE FILENAME
E1E6  20 06 E2           JSR L644         ;GET NEXT CHARACTER
E1E9  20 00 E2           JSR L626         ;GET DEVICE NUMBER
E1EC  A0 00              LDY #$00
E1EE  86 49              STX FORPNT       ;GET FILE NUMBER
E1F0  20 BA FF           JSR L217         ;SET FILE DETAILS
E1F3  20 06 E2           JSR L644         ;GET NEXT CHARACTER
E1F6  20 00 E2           JSR L626         ;GET SECONDARY ADDRESS
E1F9  8A                 TXA              ;TRANSFER 2ND TO .X
E1FA  AB                 TAY              ;THEN TO .Y
E1FB  A6 49              LDX FORPNT       ;GET FILE NUMBER
E1FD  4C BA FF           JMP L217         ;SET FILE DETAILS
E200              ;
E200              ;GET ONE BYTE DEVICE/SECONDARY ADDRESS VALUE
E200              ;
E200  20 0E E2    L626   JSR L646         ;SCAN PAST ','
E203  4C 9E B7           JMP L195         ;LOAD .X WITH VALUE
E206              ;
E206              ;CHECK FOR FOLLOWING CHARACTERS
E206              ;
E206  20 79 00    L644   JSR CHRGOT       ;CHARGOT
E209  D0 02              BNE L642         ;ANOTHER CHARACTER
E20B  68                 PLA              ;REMOVE RETURN ADDRESS SO THAT
E20C  68                 PLA              ;AN 'RTS' GOES TO MAIN ROUTINE
E20D  60          L642   RTS
E20E              ;
E20E              ;CHECK ON COMMA
E20E              ;
E20E  20 FD AE    L646   JSR L296         ;SCAN PAST COMMA
E211  20 79 00    L645   JSR CHRGOT       ;CHARGOT
E214  D0 F7              BNE L642         ;THERE IS ANOTHER CHARACTER (RTS)
E216  4C 08 AF           JMP L94          ;'SYNTAX ERROR'
E219              ;**************************************
E219              ;*GET PARAMETERS FOR OPEN/CLOSE.
E219              ;*THIS ROUTINE SETS UP THE FILE AND FILENAME
E219              ;*PARAMETERS FOR OPEN/CLOSE. WITH CLOSE, THE
E219              ;*FIRST PARAMETER IS THE ONLY ONE REQUIRED
E219              ;*BUT YOU CAN WRITE IT LIKE THE OPEN COMMAND
E219              ;*TO HELP DE-BUGGING.
E219              ;**************************************
E219  A9 00       L647   LDA #$00         ;DEFAULT FILENAME LENGTH =0
E21B  20 BD FF           JSR L641         ;SET FILENAME DETAILS
E21E  20 11 E2           JSR L645         ;GET NEXT CHARACTER
E221  20 9E B7           JSR L195         ;GET LOGICAL FILENAME INTO .X
E224  86 49              STX FORPNT       ;STORE FILE NUMBER
E226  8A                 TXA              ;TRANSFER TO .A
E227  A2 01              LDX #$01         ;DEFAULT DEVICE = TAPE
E229  A0 00              LDY #$00         ;DEFAULT SECONDARY ADDRESS = 0
E22B  20 BA FF           JSR L217         ;SET FILE DETAILS
E22E  20 06 E2           JSR L644         ;GET NEXT CHARACTER
E231  20 00 E2           JSR L626         ;GET DEVICE NUMBER
```

```
LOC   CODE        LINE

E234  86 4A             STX FORPNT+1    ;STORE DEVICE NUMBER
E236  A0 00             LDY #$00        ;SECONDARY ADDRESS
E238  A5 49             LDA FORPNT      ;FILE NUMBER
E23A  E0 03             CPX #$03        ;OPEN SERIAL OR SCREEN?
E23C  90 01             BCC L637        ;NO
E23E  88                DEY             ;DEFAULT 2ND TO $FF FOR SCREEN/SERIAL
E23F  20 BA FF    L637  JSR L217         ;SET FILE DETAILS
E242  20 06 E2          JSR L644        ;GET NEXT CHARACTER
E245  20 00 E2          JSR L626        ;GET SECONDARY ADDRESS
E248  8A                TXA             ;MOVE SECONDARY ADDRESS TO .A
E249  A8                TAY             ;THEN TO .Y
E24A  A6 4A             LDX FORPNT+1    ;GET DEVICE #
E24C  A5 49             LDA FORPNT      ;GET FILE #
E24E  20 BA FF          JSR L217        ;SET FILE DETAILS
E251  20 06 E2          JSR L644        ;GET NEXT CHARACTER
E254  20 0E E2          JSR L646        ;SCAN PAST ','
E257                    ;
E257                    ;GET FILENAME FOR LOAD/SAVE/VERIFY
E257                    ; OPEN/CLOSE.
E257                    ;
E257  20 9E AD    L648  JSR L256         ;EVALUATE EXPRESSION
E25A  20 A3 B6          JSR L435        ;CHECK ON STRING
E25D  A6 22             LDX INDEX       ;GET ADDRESS OF STRING LSB
E25F  A4 23             LDY INDEX+1     ;MSB (.A HOLDS LENGTH)
E261  4C BD FF          JMP L641        ;SET FILENAME DETAILS
E264                    ;***************************************
E264                    ;*PERFORM 'COS'.
E264                    ;*THE ARGUMENT IN RADIANS IS STORED IN FAC#1
E264                    ;*THIS IS THEN ADDED TO A VALUE OF PI/2 STORED
E264                    ;*IN FAC#2 AND THE RESULT STORED IN FAC#1.
E264                    ;*THE ROUTINE THEN JUMPS TO THE PERFORM 'SIN'
E264                    ;*ROUTINE AT $E26B AND THE RESULT STORED IN
E264                    ;*FAC#1.
E264                    ;***************************************
E264  A9 E0       COS   LDA #$E0         ;POINTER TO CONSTANT PI/2 LSB
E266  A0 E2             LDY #$E2        ;MSB
E268  20 67 B8          JSR L469        ;ADD CONSTANT TO FAC#1 PUT IN FAC#1
E26B                    ;***************************************
E26B                    ;*PERFORM 'SIN'.
E26B                    ;*THE ARGUMENT IN RADIANS IS STORED IN FAC#1
E26B                    ;*IT IS EVALUATED TO GIVE THE SIGN OF THE
E26B                    ;*ANGLE, THIS IS STORED IN FAC#1.
E26B                    ;***************************************
E26B  20 0C BC    L643  JSR L516         ;ROUND FAC#1 AND TRANSFER TO FAC#2
E26E  A9 E5             LDA #$E5        ;POINTER TO CONSTANT PI*2 LSB
E270  A0 E2             LDY #$E2        ;MSB
E272  A6 6E             LDX ARGSGN      ;GET FAC#2 SIGN , AND PUT IN SIGN
E274  20 07 BB          JSR L549        ;FAC#1 = FAC#2 / 2 * PI
E277  20 0C BC          JSR L516        ;ROUND FAC#1 TRANSFER TO FAC#2
E27A  20 CC BC          JSR L531        ;PERFORM 'INT' FUNCTION
E27D  A9 00             LDA #$00
E27F  85 6F             STA ARISGN      ;CLEAR SIGN COMPARISON FLAG
E281  20 53 B8          JSR L489        ;FAC#1 = FAC#2 - FAC#1
E284  A9 EA             LDA #$EA        ;POINTER TO CONSTANT 0.25 LSB
E286  A0 E2             LDY #$E2        ;MSB
E288  20 50 B8          JSR L570        ;FAC#1 = 0.25 - FAC#1
E28B  A5 66             LDA FACSGN      ;GET FAC#1 SIGN
E28D  48                PHA             ;PUSH TO STACK
E28E  10 0D             BPL L653        ;POSITIVE?
E290  20 49 B8          JSR L466        ;FAC#1 = FAC#1 + 0.5
E293  A5 66             LDA FACSGN      ;GET SIGN OF FAC#1
```

```
LOC   CODE        LINE

E295  30 09             BMI L650        ;NEGATIVE?
E297  A5 12             LDA TANSGN      ;GET SIGN FLAG
E299  49 FF             EOR #$FF        ;INVERT
E29B  85 12             STA TANSGN      ;AND SAVE
E29D  20 B4 BF    L653  JSR L591         ;PERFORM NEGATIVE(FAC#1 = - FAC#1)
E2A0  A9 EA       L650  LDA #$EA         ;POINTER TO CONSTANT 0.25 LSB
E2A2  A0 E2             LDY #$E2        ;MSB
E2A4  20 67 B8          JSR L469        ;FAC#1 = FAC#1 + 0.25
E2A7  68                PLA             ;GET FAC#1 SIGN
E2A8  10 03             BPL L651        ;POSITIVE
E2AA  20 B4 BF          JSR L591        ;FAC#1 = - FAC#1
E2AD  A9 EF       L651  LDA #$EF         ;POINTER TO CONSTANT 5 LSB
E2AF  A0 E2             LDY #$E2        ;MSB
E2B1  4C 43 E0          JMP L603        ;PERFORM SERIES EVALUATION 1
E2B4                ;***************************************
E2B4                ;*PERFORM 'TAN'.
E2B4                ;*THE 'TAN' OF THE ARGUMENT IN FAC#1 IS
E2B4                ;*CALCULATED BY DIVIDING THE SINE OF THE
E2B4                ;*VALUE BY THE COSINE USING THE ROUTINES AT
E2B4                ;*$E26B (SIN) AND $E264 (COS). THE RESULT IS
E2B4                ;*STORED IN FAC#1.
E2B4                ;***************************************
E2B4  20 CA BB    TAN   JSR L609         ;PACK FAC#1 TO LOCATION $0057
E2B7  A9 00             LDA #$00
E2B9  85 12             STA TANSGN      ;CLEAR COMPARISON FLAG
E2BB  20 6B E2          JSR L643        ;CALCULATE SINE OF VALUE
E2BE  A2 4E             LDX #$4E        ;POINTER TO SINE STORE LSB
E2C0  A0 00             LDY #$00        ;MSB
E2C2  20 F6 E0          JSR L616        ;ROUND FAC#1 AND PACK TO $004E
E2C5  A9 57             LDA #$57        ;POINTER TO VALUE STORE LSB
E2C7  A0 00             LDY #$00        ;MSB
E2C9  20 A2 BB          JSR L496        ;UNPACK $0057 TO FAC#1
E2CC  A9 00             LDA #$00
E2CE  85 66             STA FACSGN      ;CLEAR SIGN OF FAC#1
E2D0  A5 12             LDA TANSGN      ;GET COMPARISON FLAG
E2D2  20 DC E2          JSR L652        ;CALCULATE COS
E2D5  A9 4E             LDA #$4E        ;POINTER TO SINE OF VALUE LSB
E2D7  A0 00             LDY #$00        ;MSB
E2D9  4C 0F BB          JMP L649        ;FAC#1 = FAC#2 / FAC#1
E2DC  48          L652  PHA              ;PUSH OFF SIGN
E2DD  4C 9D E2          JMP L653        ;CALCULATE COS
E2E0                ;***************************************
E2E0                ;*SIN AND COS CONSTANTS.
E2E0                ;***************************************
E2E0                ;1.57079633 (PI/2)
E2E0  81                .BYT $81,$49,$0F,$DA,$A2
E2E1  49
E2E2  0F
E2E3  DA
E2E4  A2
E2E5                ;6.28318531 (PI*2)
E2E5  83                .BYT $83,$49,$0F,$DA,$A2
E2E6  49
E2E7  0F
E2E8  DA
E2E9  A2
E2EA                ;0.25
E2EA  7F                .BYT $7F,$00,$00,$00,$00
E2EB  00
E2EC  00
E2ED  00
```

```
LOC   CODE        LINE

E2EE  00
E2EF              ;5
E2EF  05                .BYT $05
E2F0              ;-14.38139
E2F0  84                .BYT $84,$E6,$1A,$2D,$1B
E2F1  E6
E2F2  1A
E2F3  2D
E2F4  1B
E2F5              ;42.007797
E2F5  86                .BYT $86,$28,$07,$FB,$F8
E2F6  28
E2F7  07
E2F8  FB
E2F9  F8
E2FA              ;-76.70417
E2FA  87                .BYT $87,$99,$68,$89,$01
E2FB  99
E2FC  68
E2FD  89
E2FE  01
E2FF              ;81.605223
E2FF  87                .BYT $87,$23,$35,$DF,$E1
E300  23
E301  35
E302  DF
E303  E1
E304              ;-41.3417021
E304  86                .BYT $86,$A5,$5D,$E7,$28
E305  A5
E306  5D
E307  E7
E308  28
E309              ;6.28318531 (PI*2)
E309  83                .BYT $83,$49,$0F,$DA,$A2
E30A  49
E30B  0F
E30C  DA
E30D  A2
E30E              ;****************************************
E30E              ;*PERFORM 'ATN'.
E30E              ;*THE ARCTANGENT OF A VALUE STORED IN FAC#1 IS
E30E              ;*CALCULATED AND THE RESULTING ANGLE IN RADIANS
E30E              ;*STORED IN FAC#1.
E30E              ;****************************************
E30E  A5 66       ATN     LDA FACSGN      ;GET SIGN OF FAC#1
E310  48                  PHA             ;PUSH TO STACK
E311  10 03               BPL L655        ;POSITIVE
E313  20 B4 BF            JSR L591        ;FAC#1 = -FAC#1
E316  A5 61       L655    LDA FACEXP      ;GET EXPONENT OF FAC#1
E318  48                  PHA             ;PUSH TO STACK
E319  C9 81               CMP #$81        ;COMPARE NUMBER WITH 1
E31B  90 07               BCC L656        ;LESS THAN 1
E31D  A9 BC               LDA #$BC        ;POINTER TO CONSTANT OF 1 LSB
E31F  A0 B9               LDY #$B9        ;MSB
E321  20 0F BB            JSR L649        ;FAC#1 = 1 / FAC#1
E324  A9 3E       L656    LDA #$3E        ;POINTER TO CONSTANT 11 LSB
E326  A0 E3               LDY #$E3        ;MSB
E328  20 43 E0            JSR L603        ;PERFORM SERIES EVALUATION 1
E32B  68                  PLA             ;PULL EXPONENT OFF STACK
E32C  C9 81               CMP #$81        ;COMPARE WITH 1
```

```
LOC   CODE        LINE

E32E  90 07             BCC L657        ;WAS LESS THAN 1
E330  A9 E0             LDA #$E0        ;POINTER TO PI/2 LSB
E332  A0 E2             LDY #$E2        ;MSB
E334  20 50 B8          JSR L570        ;FAC#1 = PI/2 - FAC#1
E337  68          L657  PLA              ;FULL SIGN
E338  10 03             BPL L658        ;POSITIVE
E33A  4C B4 BF          JMP L591        ;FAC#1 = -FAC#1
E33D  60          L658  RTS              ;END
E33E              ;***************************************
E33E              ;*CONSTANTS FOR ATN EVALUATION
E33E              ;***************************************
E33E              ;11
E33E  0B                .BYT $0B
E33F              ;-6.84793912 E-4
E33F  76                .BYT $76,$B3,$83,$BD,$D3
E340  B3
E341  83
E342  BD
E343  D3
E344              ;4.85094216 E-3
E344  79                .BYT $79,$1E,$F4,$A6,$F5
E345  1E
E346  F4
E347  A6
E348  F5
E349              ;-0.0161117018
E349  7B                .BYT $7B,$83,$FC,$B0,$10
E34A  83
E34B  FC
E34C  B0
E34D  10
E34E              ;0.034209638
E34E  7C                .BYT $7C,$0C,$1F,$67,$CA
E34F  0C
E350  1F
E351  67
E352  CA
E353              ;-0.0542791328
E353  7C                .BYT $7C,$DE,$53,$CB,$C1
E354  DE
E355  53
E356  CB
E357  C1
E358              ;0.0724571965
E358  7D                .BYT $7D,$14,$64,$70,$4C
E359  14
E35A  64
E35B  70
E35C  4C
E35D              ;-0.0898023954
E35D  7D                .BYT $7D,$B7,$EA,$51,$7A
E35E  B7
E35F  EA
E360  51
E361  7A
E362              ;0.110932413
E362  7D                .BYT $7D,$63,$30,$88,$7E
E363  63
E364  30
E365  88
E366  7E
```

```
LOC   CODE        LINE

E367              ;-0.142839808
E367  7E                .BYT $7E,$92,$44,$99,$3A
E368  92
E369  44
E36A  99
E36B  3A
E36C              ;0.19999912
E36C  7E                .BYT $7E,$4C,$CC,$91,$C7
E36D  4C
E36E  CC
E36F  91
E370  C7
E371              ;-0.333333316
E371  7F                .BYT $7F,$AA,$AA,$AA,$13
E372  AA
E373  AA
E374  AA
E375  13
E376              ;1
E376  81                .BYT $81,$00,$00,$00,$00
E377  00
E378  00
E379  00
E37A  00
E37B                    .END
E37B                    .LIB B15
E37B              ;****************************************
E37B              ;*BASIC WARM RESTART.
E37B              ;*THIS ROUTINE IS CALLED BY JMP ($A002) WHEN
E37B              ;*THE STOP KEY IS PRESSED WITH THE RESTORE
E37B              ;*KEY. THE ROUTINE CALLS THE KERNAL CLRCH
E37B              ;*ROUTINE WHICH RESETS DEFAULT I/O, INITIALISES
E37B              ;*BASIC, AND JUMPS TO 'READY' VIA THE ERROR
E37B              ;*MESSAGE LINK.
E37B              ;****************************************
E37B  20 CC FF    WRST    JSR L674        ;RESET DEFAULT I/O
E37E  A9 00               LDA #$00
E380  85 13               STA INPRMT      ;SET BASIC INPUT DEVICE TO KEYBOARD
E382  20 7A A6            JSR L636        ;INITIALISE BASIC SYSTEM
E385  58                  CLI             ;RE-ENABLE INTERRUPTS
E386  A2 80       L659    LDX #$80        ;ERROR > 127
E388              ;****************************************
E388              ;*ERROR MESSAGE LINK.
E388              ;*THIS ROUTINE IS USED TO DETERMINE WHETHER
E388              ;*TO SEND AN ERROR MESSAGE OR PRINT 'READY'.
E388              ;*IF THE ERROR MESSAGE VALUE STORED IN .X
E388              ;*IS GREATER THAN 127 THE 'READY' IS PRINTED
E388              ;*AND THEN BACK TO WARM START. OTHERWISE
E388              ;*THE ERROR MESSAGE CORRESPONDING TO THE
E388              ;*VALUE IN .X IS OUTPUT.
E388              ;****************************************
E388  6C 00 03    EML     JMP ($0300)     ;JUMP TO ERROR ROUTINE
E38B  8A                  TXA             ;GET STATUS OF ERROR NUMBER
E38C  30 03               BMI L83         ;>127, THEN  SEND 'READY'
E38E  4C 3A A4            JMP L47         ;OUTPUT ERROR MESSAGE
E391  4C 74 A4    L83     JMP L120        ;JUMP TO 'READY.'
E394              ;****************************************
E394              ;*BASIC COLD START.
E394              ;*THIS ROUTINE IS CALLED BY JMP ($A000) WHEN
E394              ;*THE MACHINE IS FIRST SWITCHED ON TO
E394              ;*INITIALISE THE BASIC OPERATING SYSTEM.
```

```
LOC   CODE        LINE

E394                ;*****************************************
E394  20 53 E4      COLD    JSR L664         ;INITIALISE BASIC VECTORS
E397  20 BF E3              JSR L666        ;INITIALISE BASIC
E39A  20 22 E4              JSR L671        ;CHECK MEM, OUTPUT POWER-UP MESSAGE
E39D  A2 FB                 LDX #$FB
E39F  9A                    TXS             ;RESET STACK POINTER
E3A0  D0 E4                 BNE L659        ;JUMP TO BASIC WARM RESTART
E3A2                ;*****************************************
E3A2                ;*BASIC CHARGET ROUTINE.
E3A2                ;*THIS ROUTINE IS COPIED DOWN INTO ZERO PAGE
E3A2                ;*BY THE NEXT ROUTINE.
E3A2                ;*****************************************
E3A2  E6 7A         L660    INC TXTPTR       ;INCREMENT BASIC POINTER LSB
E3A4  D0 02                 BNE L667        ;NO OVERFLOW
E3A6  E6 7B                 INC TXTPTR+1    ;INCREMENT MSB
E3A8  AD 60 EA      L667    LDA $EA60        ;GET NEXT BYTE FROM BASIC
E3AB  C9 3A                 CMP #$3A        ;IS IT A COLON?
E3AD  B0 0A                 BCS L665        ;NO, LESS THAN (NUMERIC)
E3AF  C9 20                 CMP #$20        ;IS IT A SPACE?
E3B1  F0 EF                 BEQ L660        ;YES, IGNORE IT AND GET NEXT
E3B3  38                    SEC
E3B4  E9 30                 SBC #$30
E3B6  38                    SEC
E3B7  E9 D0                 SBC #$D0
E3B9  60            L665    RTS
E3BA                ;*****************************************
E3BA                ;*INITIAL RND SEED.
E3BA                ;*THIS VALUE IS COPIED INTO LOCATIONS
E3BA                ;*$8B TO $8F. THE VALUE=.811635157.
E3BA                ;*****************************************
E3BA  80                    .BYT $80,$4F,$C7,$52,$58
E3BB  4F
E3BC  C7
E3BD  52
E3BE  58
E3BF                ;*****************************************
E3BF                ;*INITIALISE BASIC.
E3BF                ;*THIS ROUTINE SETS UP THE USR FUNCTION
E3BF                ;*JUMP, COPIES CHARGET AND RND SEED INTO ZERO
E3BF                ;*PAGE, SETS UP THE FIX-FLOAT AND FLOAT-FIX
E3BF                ;*VECTORS, SETS START AND END OF RAM, AND
E3BF                ;*SETS THE VARIABLE POINTERS.
E3BF                ;*****************************************
E3BF  A9 4C         L666    LDA #$4C         ;VALUE FOR 'JMP'
E3C1  85 54                 STA $54         ;STORE TO FUNCTION VECTOR
E3C3  8D 10 03              STA $0310       ;STORE TO USR VECTOR
E3C6  A9 48                 LDA #$48        ;'ILLEGAL QUANTITY' LSB
E3C8  A0 B2                 LDY #$B2        ;MSB FOR INITIAL USR
E3CA  8D 11 03              STA $0311       ;STORE LSB
E3CD  8C 12 03              STY $0312       ;STORE MSB
E3D0  A9 91                 LDA #$91        ;FIX-FLOAT LSB
E3D2  A0 B3                 LDY #$B3        ;MSB
E3D4  85 05                 STA ADRAY2      ;STORE LSB
E3D6  84 06                 STY ADRAY2+1    ;STORE MSB
E3D8  A9 AA                 LDA #$AA        ;FLOAT-FIX LSB
E3DA  A0 B1                 LDY #$B1        ;MSB
E3DC  85 03                 STA ADRAY1      ;STORE LSB
E3DE  84 04                 STY ADRAY1+1    ;STORE MSB
E3E0  A2 1C                 LDX #$1C        ;LOOP TO COPY CHARGET AND RND SEED
E3E2  BD A2 E3      L663    LDA $E3A2,X      ;GET CHARGET/RND SEED BYTE
E3E5  95 73                 STA $73,X       ;STORE IT TO ZERO PAGE
```

```
LOC   CODE        LINE

E3E7  CA                  DEX             ;DECREMENT COUNTER
E3E8  10 F8               BPL L663        ;REPEAT FOR NEXT BYTE
E3EA  A9 03               LDA #$03
E3EC  85 53               STA $53
E3EE  A9 00               LDA #$00
E3F0  85 68               STA BITS        ;CLEAR FAC#1 OVERFLOW FLAG
E3F2  85 13               STA INPRMT      ;SET INPUT DEVICE TO KEYBOARD
E3F4  85 18               STA LASTPT+1    ;NO TEMPORARY STRING
E3F6  A2 01               LDX #$01
E3F8  8E FD 01            STX BUF-3
E3FB  8E FC 01            STX BUF-4
E3FE  A2 19               LDX #$19
E400  86 16               STX TEMPPT      ;STORE POINTER TO STRING STACK
E402  38                  SEC
E403  20 9C FF            JSR L670        ;READ BOTTOM OF MEMORY
E406  86 2B               STX TXTTAB      ;STORE BOTTOM OF MEMORY LSB
E408  84 2C               STY TXTTAB+1    ;MSB
E40A  38                  SEC
E40B  20 99 FF            JSR L25         ;READ TOP OF MEMORY
E40E  86 37               STX MEMTOP      ;STORE TOP OF MEMORY LSB
E410  84 38               STY MEMTOP+1    ;MSB
E412  86 33               STX FRETOP      ;STORE STRING POINTER LSB
E414  84 34               STY FRETOP+1    ;MSB
E416  A0 00               LDY #$00
E418  98                  TYA
E419  91 2B               STA (TXTTAB),Y ;SET ZERO BYTE BELOW BASIC MEMORY
E41B  E6 2B               INC TXTTAB      ;BOTTOM = BOTTOM + 1
E41D  D0 02               BNE L668
E41F  E6 2C               INC TXTTAB+1
E421  60            L668  RTS              ;INITIALISE DONE
E422                ;***************************************
E422                ;*OUTPUT POWER-UP MESSAGE.
E422                ;*THIS ROUTINE CHECKS ON THE AMMOUNT OF
E422                ;*FREE MEMORY AND OUTPUTS THE POWER-UP
E422                ;*MESSAGE.
E422                ;***************************************
E422  A5 2B         L671  LDA TXTTAB       ;GET BOTTOM OF MEMORY LSB
E424  A4 2C               LDY TXTTAB+1    ;MSB
E426  20 08 A4            JSR L92         ;CHECK FREE RAM
E429  A9 73               LDA #$73        ;POINTER TO POWER-UP MESSAGE LSB
E42B  A0 E4               LDY #$E4        ;MSB
E42D  20 1E AB            JSR L198        ;OUTPUT STRING
E430  A5 37               LDA MEMTOP      ;TOP OF MEMORY LSB
E432  38                  SEC
E433  E5 2B               SBC TXTTAB      ;MINUS BOTTOM OF MEMORY LSB
E435  AA                  TAX             ;STORE TO .X
E436  A5 38               LDA MEMTOP+1    ;TOP OF MEMORY MSB
E438  E5 2C               SBC TXTTAB+1    ;MINUS BOTTOM OF MEMORY MSB
E43A  20 CD BD            JSR L24         ;OUTPUT FREE RAM NUMBER
E43D  A9 60               LDA #$60        ;POINTER TO 'BASIC BYTES FREE' LSB
E43F  A0 E4               LDY #$E4        ;MSB
E441  20 1E AB            JSR L198        ;OUTPUT STRING
E444  4C 44 A6            JMP L65         ;PERFORM 'NEW'
E447                ;***************************************
E447                ;*VECTORS FOR BASIC.
E447                ;***************************************
E447  8B E3               .WOR $E38B      ;ERROR MESSAGE
E449  83 A4               .WOR $A483      ;WARM START
E44B  7C A5               .WOR $A57C      ;CRUNCH TOKENS
E44D  1A A7               .WOR $A71A      ;PRINT TOKENS
E44F  E4 A7               .WOR $A7E4      ;START NEW CODE
```

```
LOC   CODE        LINE

E451  86 AE             .WOR $AE86      ;ARITHMETIC LINK
E453                    ;***************************************
E453                    ;*INITIALISE BASIC VECTORS.
E453                    ;*THIS ROUTINE COPIES THE VECTORS ABOVE INTO
E453                    ;*THE MEMORY FOR THE BASIC OPERATING SYSTEM.
E453                    ;***************************************
E453  A2 0B       L664  LDX #$0B         ;LOOP TO COPY VECTORS
E455  BD 47 E4    L662  LDA $E447,X      ;GET BYTE OF VECTOR
E458  9D 00 03          STA $0300,X     ;STORE VECTOR
E45B  CA                DEX             ;DECREMENT COUNTER
E45C  10 F7             BPL L662        ;DO NEXT BYTE
E45E  60                RTS
E45F                    ;***************************************
E45F                    ;*BASIC POWER-UP MESSAGE.
E45F                    ;*THIS MESSAGE IS PRINTED TO THE TOP OF
E45F                    ;*THE SCREEN WHEN THE COMPUTER IS SWITCHED
E45F                    ;*ON. THE BYTE VALUES $0D ARE CARRIAGE
E45F                    ;*RETURN, THE BYTE $93 IS CLEAR SCREEN.
E45F                    ;***************************************
E45F  00                .BYT $00,' BASIC BYTES FREE',CR,$00
E460  20 42
E471  0D
E472  00
E473  93                .BYT $93,CR,'    **** COMMODORE 64 BASIC V2 ****'
E474  0D
E475  20 20
E498  0D                .BYT CR,CR,' 64K RAM SYSTEM  ',$00,$5C
E499  0D
E49A  20 36
E4AB  00
E4AC  5C
E4AD                    ;***************************************
E4AD                    ;*BASIC SET OUTPUT DEVICE ROUTINE.
E4AD                    ;*THIS ROUTINE IS CALLED BY THE EARLIER I/O
E4AD                    ;*ROUTINE OF THE SAME NAME TO SET UP THE
E4AD                    ;*OUTPUT DEVICE.
E4AD                    ;***************************************
E4AD  48          L673  PHA             ;SAVE VALUE IN .A
E4AE  20 C9 FF          JSR L624        ;SET OUTPUT DEVICE
E4B1  AA                TAX             ;ANY ERROR INTO .X
E4B2  68                PLA             ;RESTORE .A
E4B3  90 01             BCC L623        ;NO ERROR, .A REMAINS
E4B5  8A                TXA             ;MOVE ERROR INTO .A
E4B6  60          L623  RTS
E4B7  AA                .BYT $AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA
E4B8  AA
E4B9  AA
E4BA  AA
E4BB  AA
E4BC  AA
E4BD  AA
E4BE  AA
E4BF  AA                .BYT $AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA
E4C0  AA
E4C1  AA
E4C2  AA
E4C3  AA
E4C4  AA
E4C5  AA
E4C6  AA
E4C7  AA                .BYT $AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA
```

```
LOC   CODE         LINE


E4C8  AA
E4C9  AA
E4CA  AA
E4CB  AA
E4CC  AA
E4CD  AA
E4CE  AA
E4CF  AA                .BYT $AA,$AA,$AA,$AA,$AA,$AA,$AA,$AA
E4D0  AA
E4D1  AA
E4D2  AA
E4D3  AA
E4D4  AA
E4D5  AA
E4D6  AA
E4D7  AA                .BYT $AA,$AA,$AA
E4D8  AA
E4D9  AA
E4DA                ;
E4DA                ;STORE SCREEN COLOUR TO LOCATION AT ($F3)
E4DA                ;
E4DA  AD 21 D0      L675    LDA $D021        ;GET SCREEN COLOUR
E4DD  91 F3                 STA ($F3),Y      ;STORE IT
E4DF  60                    RTS
E4E0                ;***************************************
E4E0                ;*PAUSE FOR COMMODORE KEY OR 8.5 (APPROX)
E4E0                ;*SECONDS.
E4E0                ;***************************************
E4E0  69 02         L806    ADC #$02
E4E2  A4 91         L1103   LDY $91          ;COMMODORE KEY PRESSED?
E4E4  C8                    INY
E4E5  D0 04                 BNE L677         ;YES
E4E7  C5 A1                 CMP $A1          ;TIMER INCREASED BY 2?
E4E9  D0 F7                 BNE L1103        ;NOT YET
E4EB  60            L677    RTS
E4EC                ;***************************************
E4EC                ;*BAUD RATE TABLE FOR RS-232 ROUTINES
E4EC                ;***************************************
E4EC  19 26                 .WOR $2619       ;  50 BAUD
E4EE  44 19                 .WOR $1944       ;  75
E4F0  1A 11                 .WOR $111A       ; 110
E4F2  E8 0D                 .WOR $0DE8       ; 134.5
E4F4  70 0C                 .WOR $0C70       ; 150
E4F6  06 06                 .WOR $0606       ; 300
E4F8  D1 02                 .WOR $02D1       ; 600
E4FA  37 01                 .WOR $0137       ;1200
E4FC  AE 00                 .WOR $00AE       ;1800
E4FE  69 00                 .WOR $0069       ;2400
E500                        .END
```

Kernal ROM

This section contains the documented machine code for the input/output and memory control routines of the Commodore 64 (Kernal ROM). The following listing is the output from the Commodore Assembler when assembling our interpretation of a source code for the Kernal routines of the Commodore 64.

NB: There are two versions of this section. If you are unsure of which one you have, use PRINT PEEK(654ø8). If the result is '3' then refer to the Machine Differences section of this book.

```
LOC   CODE        LINE

E500              ;****************************************
E500              ;*                                      *
E500              ;* KK  K EEEEE RRRR  NN  N  AAA  LL     *
E500              ;* KK KK EE    RR  R NNN N AA  A LL     *
E500              ;* KKK   EE    RR  R NNN N AA  A LL     *
E500              ;* KKK   EEEEE RRRR  NNNNN AAAAA LL     *
E500              ;* KK K  EE    RR  R NN NN AA  A LL     *
E500              ;* KK KK EE    RR  R NN NN AA  A LL     *
E500              ;* KK KK EEEEE RR  R NN NN AA  A LLLLL  *
E500              ;*                                      *
E500              ;****************************************
E500                     .END
E500                     .LIB K1
E500              ;
E500              ;RETURN ADDRESS OF 6526
E500              ;
E500  A2 00       L676   LDX #<D1DFA      ;LO BYTE
E502  A0 DC              LDY #>D1DFA     ;HI BYTE
E504  60                 RTS
E505              ;
E505              ;RETURN MAX ROWS/COLS OF SCREEN
E505              ;
E505  A2 28       L1248  LDX #LLEN        ;LINE LENGTH
E507  A0 19              LDY #NLINES     ;# OF LINES
E509  60                 RTS
E50A              ;
E50A              ;****************************************
E50A              ;* READ/PLOT CURSOR POSITION
E50A              ;*
E50A              ;* CARRY CLEAR TO PLOT WITH COLUMN IN .X
E50A              ;* AND ROW IN .Y.
E50A              ;*
E50A              ;* CARRY SET TO READ POSITION. COLUMN
E50A              ;* IS RETURNED IN .X AND ROW IN .Y.
E50A              ;****************************************
E50A              ;
```

```
LOC   CODE        LINE

E50A  B0 07       L1246  BCS L1247
E50C  86 D6              STX TBLX
E50E  84 D3              STY PNTR
E510  20 6C E5           JSR L689
E513  A6 D6       L1247  LDX TBLX
E515  A4 D3              LDY PNTR
E517  60                 RTS
E518              ;
E518              ;INITIALIZE I/O
E518              ;
E518  20 A0 E5    L678   JSR L687         ;SET UP VIC CHIP
E51B  A9 00              LDA #$00        ;PUT IN PET MODE
E51D  8D 91 02           STA MODE
E520  85 CF              STA BLNON       ;NO CHAR FROM SCREEN
E522  A9 48              LDA #<SHFLOG    ;SET SHIFT LOGIC
E524  8D 8F 02           STA KEYLOG      ; INDIRECTS
E527  A9 EB              LDA #>SHFLOG
E529  8D 90 02           STA KEYLOG+1
E52C  A9 0A              LDA #$0A
E52E  8D 89 02           STA XMAX        ;MAX TYPE AHEAD BUFFER SIZE
E531  8D 8C 02           STA DELAY
E534  A9 0E              LDA #LTBLUE
E536  8D 86 02           STA COLOR
E539  A9 04              LDA #$04
E53B  8D 8B 02           STA KOUNT       ;DELAY BETWEEN KEY REPEATS
E53E  A9 0C              LDA #$0C
E540  85 CD              STA BLNCT
E542  85 CC              STA BLNSW
E544  AD 88 02    L1221  LDA HIBASE       ;FILL HI BYTE PTR TABLE
E547  09 80              ORA #$80
E549  A8                 TAY
E54A  A9 00              LDA #$00
E54C  AA                 TAX
E54D  94 D9       L778   STY LDTB1,X
E54F  18                 CLC
E550  69 28              ADC #LLEN
E552  90 01              BCC L682
E554  C8                 INY             ;CARRY BUMP HI BYTE
E555  E8          L682   INX
E556  E0 1A              CPX #NLINES+1   ;DONE # OF LINES?
E558  D0 F3              BNE L778        ;NO
E55A  A9 FF              LDA #$FF        ;TAG END OF LINE TABLE
E55C  95 D9              STA LDTB1,X
E55E  A2 18              LDX #NLINES-1   ;CLEAR FROM BOTTOM LINE UP
E560  20 FF E9    L681   JSR L726         ;SEE SCROLL ROUTINES
E563  CA                 DEX
E564  10 FA              BPL L681
E566              ;
E566              ;HOME FUNCTION
E566              ;
E566  A0 00       L684   LDY #$00
E568  84 D3              STY PNTR        ;LEFT COLUMN
E56A  84 D6              STY TBLX        ;TOP LINE
E56C              ;
E56C              ;MOVE CURSOR TO TBLX,PNTR
E56C              ;
E56C  A6 D6       L689   LDX TBLX        ;GET CURRENT LINE INDEX
E56E  A5 D3              LDA PNTR        ;GET CHARACTER POINTER
E570  B4 D9       L679   LDY LDTB1,X     ;FIND BEGINNING OF LINE
E572  30 08              BMI L686        ;BRANCH IF START FOUND
E574  18                 CLC
```

```
LOC   CODE        LINE

E575  69 28             ADC #LLEN       ;ADJUST POINTER
E577  85 D3             STA PNTR
E579  CA                DEX
E57A  10 F4             BPL L679
E57C  B5 D9       L686  LDA LDTB1,X
E57E  29 03             AND #$03        ;GET RID OF GARBAGE
E580  0D 88 02          ORA HIBASE      ;OR IN HIGH ORDER BITS
E583  85 D2             STA PNT+1
E585  BD F0 EC          LDA $ECF0,X
E588  85 D1             STA PNT
E58A  A9 27             LDA #LLEN-1
E58C  E8                INX
E58D  B4 D9       L685  LDY LDTB1,X
E58F  30 06             BMI L688
E591  18                CLC
E592  69 28             ADC #$28
E594  E8                INX
E595  10 F6             BPL L685
E597  85 D5       L688  STA LNMX
E599  60                RTS
E59A              ;
E59A              ;NMI PANIC ENTRY
E59A              ;
E59A  20 A0 E5    NMIPAN JSR L687         ;FIX 64 SCREEN
E59D  4C 66 E5          JMP L684        ;HOME CURSOR
E5A0  A9 03       L687  LDA #$03         ;RESET DEFAULT I/O
E5A2  85 9A             STA DFLTO
E5A4  A9 00             LDA #$00
E5A6  85 99             STA DFLTN
E5A8              ;
E5A8              ;INIT 64
E5A8              ;
E5A8  A2 2F       INITNM LDX #$2F
E5AA  BD B8 EC    L680  LDA $ECB8,X
E5AD  9D FF CF          STA VICREG-1,X
E5B0  CA                DEX
E5B1  D0 F7             BNE L680
E5B3  60                RTS
E5B4              ;
E5B4              ;***************************************
E5B4              ;* REMOVE A CHARACTER FROM THE KEYBOARD
E5B4              ;* QUEUE.
E5B4              ;*   THIS ROUTINE WILL REMOVE A CHAR FROM
E5B4              ;* THE KEYBOARD QUEUE AND PLACE IT IN .A
E5B4              ;***************************************
E5B4              ;
E5B4  AC 77 02    L690  LDY KEYD         ;GET KEYDOWN
E5B7  A2 00             LDX #$00        ;LOOP TO SHIFT CHARS IN BUFFER
E5B9  BD 78 02    L696  LDA KEYD+1,X     ;GET CHARACTER
E5BC  9D 77 02          STA KEYD,X      ;MOVE IT DOWN 1
E5BF  E8                INX             ;INCREMENT COUNTER
E5C0  E4 C6             CPX NDX         ;DONE # OF CHARS IN BUFFER?
E5C2  D0 F5             BNE L696        ;NOT YET
E5C4  C6 C6             DEC NDX         ;# OF CHARS -1
E5C6  98                TYA             ;CHARACTER IN .A
E5C7  58                CLI             ;ENABLE KEYBOARD
E5C8  18                CLC             ;GOOD RETURN
E5C9  60                RTS
E5CA  20 16 E7    L691  JSR L728         ;OUTPUT CHARACTER IN .A TO SCREEN
E5CD  A5 C6       L699  LDA NDX          ;# OF CHARS
E5CF  85 CC             STA BLNSW       ;BLINK IF NO CHARS
```

```
LOC   CODE        LINE

E5D1  8D 92 02            STA AUTODN      ;TURN ON AUTO SCROLL DOWN
E5D4  F0 F7               BEQ L699        ;NONE
E5D6  78                  SEI
E5D7  A5 CF               LDA BLNON
E5D9  F0 0C               BEQ L693
E5DB  A5 CE               LDA GDBLN
E5DD  AE 87 02            LDX GDCOL       ;RESTORE ORIGINAL COLOUR
E5E0  A0 00               LDY #$00
E5E2  84 CF               STY BLNON
E5E4  20 13 EA            JSR L807        ;RESTORE CHAR
E5E7  20 B4 E5    L693    JSR L690         ;REMOVE CHAR FROM BUFFER
E5EA  C9 83               CMP #$83        ;RUN KEY?
E5EC  D0 10               BNE L698        ;NO
E5EE  A2 09               LDX #$09        ;LOOP TO COPY RUN/STOP INTO BUFFER
E5F0  78                  SEI
E5F1  86 C6               STX NDX
E5F3  BD E6 EC    L694    LDA $ECE6,X      ;GET CHAR
E5F6  9D 76 02            STA KEYD-1,X    ;STORE CHAR
E5F9  CA                  DEX
E5FA  D0 F7               BNE L694
E5FC  F0 CF               BEQ L699        ;ALWAYS, TEST FOR NEXT CHAR
E5FE  C9 0D       L698    CMP #$0D         ;CARRIAGE RETURN?
E600  D0 C8               BNE L691        ;NO, PRINT AND GET NEXT
E602  A4 D5               LDY LNMX
E604  84 D0               STY CRSW
E606  B1 D1       L697    LDA (PNT),Y
E608  C9 20               CMP #$20        ;SPACE CHAR
E60A  D0 03               BNE L701
E60C  88                  DEY
E60D  D0 F7               BNE L697
E60F  C8          L701    INY
E610  84 C8               STY INDX
E612  A0 00               LDY #$00
E614  8C 92 02            STY AUTODN      ;TURN OFF AUTO SCROLL DOWN
E617  84 D3               STY PNTR
E619  84 D4               STY QTSW
E61B  A5 C9               LDA LSXP
E61D  30 1B               BMI L930
E61F  A6 D6               LDX TBLX
E621  20 ED E6            JSR L723        ;FIND 1ST PHYSICAL LINE
E624  E4 C9               CPX LSXP
E626  D0 12               BNE L930
E628  A5 CA               LDA LSTP
E62A  85 D3               STA PNTR
E62C  C5 C8               CMP INDX
E62E  90 0A               BCC L930
E630  B0 2B               BCS L707
E632              ;
E632              ;INPUT A LINE UNTIL CARRIAGE RETURN
E632              ;
E632  98          L700    TYA
E633  48                  PHA
E634  8A                  TXA
E635  48                  PHA
E636  A5 D0               LDA CRSW
E638  F0 93               BEQ L699
E63A  A4 D3       L930    LDY PNTR
E63C  B1 D1               LDA (PNT),Y
E63E  85 D7               STA DATA
E640  29 3F               AND #$3F
E642  06 D7               ASL DATA
```

```
LOC   CODE        LINE

E644  24 D7             BIT DATA
E646  10 02             BPL L702
E648  09 80             ORA #$80
E64A  90 04       L702  BCC L705
E64C  A6 D4             LDX QTSW
E64E  D0 04             BNE L706
E650  70 02       L705  BVS L706
E652  09 40             ORA #$40
E654  E6 D3       L706  INC PNTR
E656  20 84 E6          JSR L712
E659  C4 C8             CPY INDX
E65B  D0 17             BNE L711
E65D  A9 00       L707  LDA #$00
E65F  85 D0             STA CRSW
E661  A9 0D             LDA #$0D
E663  A6 99             LDX DFLTN       ;FIX GETS FROM SCREEN
E665  E0 03             CPX #$03        ;IS IT THE SCREEN?
E667  F0 06             BEQ L704
E669  A6 9A             LDX DFLTO
E66B  E0 03             CPX #$03
E66D  F0 03             BEQ L710
E66F  20 16 E7    L704  JSR L728
E672  A9 0D       L710  LDA #$0D
E674  85 D7       L711  STA DATA
E676  68                PLA
E677  AA                TAX
E678  68                PLA
E679  A8                TAY
E67A  A5 D7             LDA DATA
E67C  C9 DE             CMP #$DE        ;IS IT 'PI'
E67E  D0 02             BNE L709
E680  A9 FF             LDA #$FF
E682  18          L709  CLC
E683  60                RTS
E684              ;
E684  C9 22       L712  CMP #$22
E686  D0 08             BNE L708
E688  A5 D4             LDA QTSW
E68A  49 01             EOR #$01
E68C  85 D4             STA QTSW
E68E  A9 22             LDA #$22
E690  60          L708  RTS
E691              ;
E691  09 40       L713  ORA #$40
E693  A6 C7       L761  LDX RVS
E695  F0 02             BEQ L739
E697  09 80       L737  ORA #$80
E699  A6 D8       L739  LDX INSRT
E69B  F0 02             BEQ L714
E69D  C6 D8             DEC INSRT
E69F  AE 86 02    L714  LDX COLOR        ;PUT COLOUR ON SCREEN
E6A2  20 13 EA          JSR L807
E6A5  20 B6 E6          JSR L717        ;CHECK FOR WRAPAROUND
E6A8  68          L715  PLA
E6A9  A8                TAY
E6AA  A5 D8             LDA INSRT
E6AC  F0 02             BEQ L729
E6AE  46 D4             LSR QTSW
E6B0  68          L729  PLA
E6B1  AA                TAX
E6B2  68                PLA
```

```
LOC   CODE        LINE

E6B3  18          L717    CLC             ;GOOD RETURN
E6B4  58                  CLI
E6B5  60                  RTS
E6B6              ;
E6B6  20 B3 E8    L717    JSR L782         ;MAYBE WE SHOULD INCREMENT
E6B9  E6 D3               INC PNTR        ;BUMP CHARACTER POINTER
E6BB  A5 D5               LDA LNMX
E6BD  C5 D3               CMP PNTR
E6BF  B0 3F               BCS L720        ;LNMX>=PNTR
E6C1  C9 4F               CMP #MAXCHR-1   ;PAST MAX CHARS?
E6C3  F0 32               BEQ L725        ;YES
E6C5  AD 92 02            LDA AUTODN      ;AUTO SCROLL?
E6C8  F0 03               BEQ L716        ;NO
E6CA  4C 67 E9            JMP L768        ;WHICH WAY?
E6CD              ;
E6CD  A6 D6       L716    LDX TBLX
E6CF  E0 19               CPX #NLINES     ;SCROLL DOWN?
E6D1  90 07               BCC L721        ;NO
E6D3  20 EA E8            JSR L787        ;SCROLL UP
E6D6  C6 D6               DEC TBLX        ;ADJUST CURRENT LINE #
E6D8  A6 D6               LDX TBLX
E6DA  16 D9       L721    ASL LDTB1,X      ;WRAP THE LINE
E6DC  56 D9               LSR LDTB1,X
E6DE  E8                  INX
E6DF  B5 D9               LDA LDTB1,X
E6E1  09 80               ORA #$80
E6E3  95 D9               STA LDTB1,X
E6E5  CA                  DEX
E6E6  A5 D5               LDA LNMX
E6E8  18                  CLC
E6E9  69 28               ADC #LLEN
E6EB  85 D5               STA LNMX
E6ED  B5 D9       L723    LDA LDTB1,X      ;FIRST LINE?
E6EF  30 03               BMI L703        ;YES
E6F1  CA                  DEX             ;ELSE BACK 1
E6F2  D0 F9               BNE L723
E6F4  4C F0 E9    L703    JMP L804         ;MAKE SURE PNT IS RIGHT
E6F7              ;
E6F7  C6 D6       L725    DEC TBLX
E6F9  20 7C E8            JSR L777
E6FC  A9 00               LDA #$00
E6FE  85 D3               STA PNTR        ;POINT TO 1ST BYTE
E700  60          L720    RTS
E701              ;
E701  A6 D6       L719    LDX TBLX
E703  D0 06               BNE L742
E705  86 D3               STX PNTR
E707  68                  PLA
E708  68                  PLA
E709  D0 9D               BNE L715
E70B              ;
E70B  CA          L742    DEX
E70C  86 D6               STX TBLX
E70E  20 6C E5            JSR L689
E711  A4 D5               LDY LNMX
E713  84 D3               STY PNTR
E715  60                  RTS
E716              ;
E716              ;PRINT ROUTINE
E716              ;
E716  48          L728    PHA
```

```
LOC   CODE        LINE

E717  85 D7             STA DATA
E719  8A                TXA
E71A  48                PHA
E71B  98                TYA
E71C  48                PHA
E71D  A9 00             LDA #$00
E71F  85 D0             STA CRSW
E721  A4 D3             LDY PNTR
E723  A5 D7             LDA DATA
E725  10 03             BPL L692
E727  4C D4 E7          JMP L753
E72A  C9 0D       L692  CMP #$0D
E72C  D0 03             BNE L730
E72E  4C 91 E8          JMP L780
E731  C9 20       L730  CMP #$20        ;SPACE CHAR
E733  90 10             BCC L736
E735  C9 60             CMP #$60        ;LOWER CASE?
E737  90 04             BCC L732        ;NO
E739  29 DF             AND #$DF        ;YES, MAKE SCREEN LOWER
E73B  D0 02             BNE L735        ;ALWAYS
E73D  29 3F       L732  AND #$3F
E73F  20 84 E6    L735  JSR L712
E742  4C 93 E6          JMP L761
E745  A6 D8       L736  LDX INSRT
E747  F0 03             BEQ L734
E749  4C 97 E6          JMP L737
E74C  C9 14       L734  CMP #$14
E74E  D0 2E             BNE L743
E750  98                TYA
E751  D0 06             BNE L738
E753  20 01 E7          JSR L719
E756  4C 73 E7          JMP L746
E759  20 A1 E8    L738  JSR L733        ;DEC TBLX?
E75C  88                DEY
E75D  84 D3             STY PNTR
E75F  20 24 EA          JSR L811        ;FIX COLOUR PTRS
E762  C8          L741  INY
E763  B1 D1             LDA (PNT),Y
E765  88                DEY
E766  91 D1             STA (PNT),Y
E768  C8                INY
E769  B1 F3             LDA (USER),Y
E76B  88                DEY
E76C  91 F3             STA (USER),Y
E76E  C8                INY
E76F  C4 D5             CPY LNMX
E771  D0 EF             BNE L741
E773  A9 20       L746  LDA #$20
E775  91 D1             STA (PNT),Y
E777  AD 86 02          LDA COLOR
E77A  91 F3             STA (USER),Y
E77C  10 4D             BPL L755
E77E  A6 D4       L743  LDX QTSW
E780  F0 03             BEQ L740
E782  4C 97 E6          JMP L737
E785  C9 12       L740  CMP #$12
E787  D0 02             BNE L748
E789  85 C7             STA RVS
E78B  C9 13       L748  CMP #$13
E78D  D0 03             BNE L749
E78F  20 66 E5          JSR L684
```

```
LOC   CODE        LINE

E792  C9 1D       L749   CMP #$1D
E794  D0 17              BNE L752
E796  C8                 INY
E797  20 B3 E8           JSR L782
E79A  84 D3              STY PNTR
E79C  88                 DEY
E79D  C4 D5              CPY LNMX
E79F  90 09              BCC L754
E7A1  C6 D6              DEC TBLX
E7A3  20 7C E8           JSR L777
E7A6  A0 00              LDY #$00
E7A8  84 D3       L750   STY PNTR
E7AA  4C A8 E6    L754   JMP L715
E7AD  C9 11       L752   CMP #$11
E7AF  D0 1D              BNE L747
E7B1  18                 CLC
E7B2  98                 TYA
E7B3  69 28              ADC #LLEN
E7B5  A8                 TAY
E7B6  E6 D6              INC TBLX
E7B8  C5 D5              CMP LNMX
E7BA  90 EC              BCC L750
E7BC  F0 EA              BEQ L750
E7BE  C6 D6              DEC TBLX
E7C0  E9 28       L751   SBC #LLEN
E7C2  90 04              BCC L756
E7C4  85 D3              STA PNTR
E7C6  D0 F8              BNE L751
E7C8  20 7C E8    L756   JSR L777
E7CB  4C A8 E6    L755   JMP L715
E7CE  20 CB E8    L747   JSR L786        ;CHECK FOR A COLOUR
E7D1  4C 44 EC           JMP L830
E7D4              ;
E7D4  29 7F       L753   AND #$7F
E7D6  C9 7F              CMP #$7F
E7D8  D0 02              BNE L731
E7DA  A9 5E              LDA #$5E
E7DC  C9 20       L731   CMP #$20         ;FUNCTION KEY?
E7DE  90 03              BCC L759        ;YES
E7E0  4C 91 E6           JMP L713
E7E3  C9 0D       L759   CMP #$0D
E7E5  D0 03              BNE L760
E7E7  4C 91 E8           JMP L780
E7EA  A6 D4       L760   LDX QTSW
E7EC  D0 3F              BNE L764
E7EE  C9 14              CMP #$14
E7F0  D0 37              BNE L767
E7F2  A4 D5              LDY LNMX
E7F4  B1 D1              LDA (PNT),Y
E7F6  C9 20              CMP #$20        ;SPACE CHAR
E7F8  D0 04              BNE L762
E7FA  C4 D3              CPY PNTR
E7FC  D0 07              BNE L765
E7FE  C0 4F       L762   CPY #MAXCHR-1
E800  F0 24              BEQ L769        ;EXIT IF LINE TOO LONG
E802  20 65 E9           JSR L803        ;SCROLL DOWN 1
E805  A4 D5       L765   LDY LNMX
E807  20 24 EA           JSR L811
E80A  88          L766   DEY
E80B  B1 D1              LDA (PNT),Y
E80D  C8                 INY
```

```
LOC   CODE        LINE

E80E  91 D1               STA (PNT),Y
E810  88                  DEY
E811  B1 F3               LDA (USER),Y
E813  C8                  INY
E814  91 F3               STA (USER),Y
E816  88                  DEY
E817  C4 D3               CPY PNTR
E819  D0 EF               BNE L766
E81B  A9 20               LDA #$20
E81D  91 D1               STA (PNT),Y
E81F  AD 86 02            LDA COLOR
E822  91 F3               STA (USER),Y
E824  E6 D8               INC INSRT
E826  4C A8 E6    L769    JMP L715
E829  A6 D8       L767    LDX INSRT
E82B  F0 05               BEQ L763
E82D  09 40       L764    ORA #$40
E82F  4C 97 E6            JMP L737
E832  C9 11       L763    CMP #$11
E834  D0 16               BNE L773
E836  A6 D6               LDX TBLX
E838  F0 37               BEQ L775
E83A  C6 D6               DEC TBLX
E83C  A5 D3               LDA PNTR
E83E  38                  SEC
E83F  E9 28               SBC #LLEN
E841  90 04               BCC L770
E843  85 D3               STA PNTR
E845  10 2A               BPL L775
E847  20 6C E5    L770    JSR L689
E84A  D0 25               BNE L775
E84C  C9 12       L773    CMP #$12
E84E  D0 04               BNE L771
E850  A9 00               LDA #$00
E852  85 C7               STA RVS
E854  C9 1D       L771    CMP #$1D
E856  D0 12               BNE L776
E858  98                  TYA
E859  F0 09               BEQ L774
E85B  20 A1 E8            JSR L733
E85E  88                  DEY
E85F  84 D3               STY PNTR
E861  4C A8 E6            JMP L715
E864  20 01 E7    L774    JSR L719
E867  4C A8 E6            JMP L715
E86A  C9 13       L776    CMP #$13
E86C  D0 06               BNE L772
E86E  20 44 E5            JSR L1221
E871  4C A8 E6    L775    JMP L715
E874  09 80       L772    ORA #$80        ;MAKE IT UPPER CASE
E876  20 CB E8            JSR L786        ;TRY FOR COLOUR
E879  4C 4F EC            JMP L758
E87C              ;
E87C  46 C9       L777    LSR LSXP
E87E  A6 D6               LDX TBLX
E880  E8          L727    INX
E881  E0 19               CPX #NLINES     ;OFF BOTTOM?
E883  D0 03               BNE L781        ;NO
E885  20 EA E8            JSR L787        ;YES, SCROLL
E888  B5 D9       L781    LDA LDTB1,X     ;DOUBLE LINE?
E88A  10 F4               BPL L727        ;YES, SCROLL AGAIN
```

```
LOC   CODE        LINE

E88C  86 D6               STX TBLX
E88E  4C 6C E5            JMP L689
E891  A2 00       L780    LDX #$00
E893  86 D8               STX INSRT
E895  86 C7               STX RVS
E897  86 D4               STX QTSW
E899  86 D3               STX PNTR
E89B  20 7C E8            JSR L777
E89E  4C A8 E6            JMP L715
E8A1              ;
E8A1              ;CHECK FOR A DECREMENT TBLX
E8A1              ;
E8A1  A2 02       L733    LDX #NWRAP
E8A3  A9 00               LDA #$00
E8A5  C5 D3       L744    CMP PNTR
E8A7  F0 07               BEQ L783
E8A9  18                  CLC
E8AA  69 28               ADC #LLEN
E8AC  CA                  DEX
E8AD  D0 F6               BNE L744
E8AF  60                  RTS
E8B0              ;
E8B0  C6 D6       L783    DEC TBLX
E8B2  60                  RTS
E8B3              ;
E8B3              ;CHECK FOR INCREMENT TBLX
E8B3              ;
E8B3  A2 02       L782    LDX #NWRAP
E8B5  A9 27               LDA #LLEN-1
E8B7  C5 D3       L718    CMP PNTR
E8B9  F0 07               BEQ L785
E8BB  18                  CLC
E8BC  69 28               ADC #LLEN
E8BE  CA                  DEX
E8BF  D0 F6               BNE L718
E8C1  60                  RTS
E8C2              ;
E8C2  A6 D6       L785    LDX TBLX
E8C4  E0 19               CPX #NLINES
E8C6  F0 02               BEQ L784
E8C8  E6 D6               INC TBLX
E8CA  60          L784    RTS
E8CB              ;
E8CB  A2 0F       L786    LDX #$0F          ;16 COLOURS
E8CD  DD DA E8    L757    CMP COLTAB,X
E8D0  F0 04               BEQ L788
E8D2  CA                  DEX
E8D3  10 F8               BPL L757
E8D5  60                  RTS
E8D6              ;
E8D6  8E 86 02    L788    STX COLOR         ;CHANGE THE COLOUR
E8D9  60                  RTS
E8DA              ;BLK,WHT,RED,CYN,PRPL,GRN,BLU,YEL
E8DA  90          COLTAB  .BYT $90,$05,$1C,$9F,$9C,$1E,$1F,$9E
E8DB  05
E8DC  1C
E8DD  9F
E8DE  9C
E8DF  1E
E8E0  1F
E8E1  9E
```

```
LOC   CODE        LINE

E8E2              ;ORNG,BRWN,LT.RED,DK.GRY
E8E2              ;MD.GRY,LT.GRN,LT.BLU,LT.GRY
E8E2  81                .BYT $81,$95,$96,$97,$98,$99,$9A,$9B
E8E3  95
E8E4  96
E8E5  97
E8E6  98
E8E7  99
E8E8  9A
E8E9  9B
E8EA                    .END
E8EA                    .LIB K2
E8EA              ;
E8EA              ;SCREEN SCROLL ROUTINE
E8EA              ;
E8EA  A5 AC       L787  LDA SAL
E8EC  48                PHA
E8ED  A5 AD             LDA SAH
E8EF  48                PHA
E8F0  A5 AE             LDA EAL
E8F2  48                PHA
E8F3  A5 AF             LDA EAH
E8F5  48                PHA
E8F6              ;
E8F6              ;SCROLL UP
E8F6              ;
E8F6  A2 FF       L724  LDX #$FF
E8F8  C6 D6             DEC TBLX
E8FA  C6 C9             DEC LSXP
E8FC  CE A5 02          DEC $02A5
E8FF  E8          L794  INX               ;GOTO NEXT LINE
E900  20 F0 E9          JSR L804        ;POINT TO 'TO' LINE
E903  E0 18             CPX #NLINES-1   ;DONE?
E905  B0 0C             BCS L791        ;YES
E907  BD F1 EC          LDA $ECF1,X     ;SETUP FROM PNTR
E90A  85 AC             STA SAL
E90C  B5 DA             LDA LDTB1+1,X
E90E  20 C8 E9          JSR L800        ;SCROLL THIS LINE UP 1
E911  30 EC             BMI L794
E913              ;
E913  20 FF E9    L791  JSR L726
E916  A2 00             LDX #$00        ;SCROLL HI BYTE POINTERS
E918  B5 D9       L789  LDA LDTB1,X
E91A  29 7F             AND #$7F
E91C  B4 DA             LDY LDTB1+1,X
E91E  10 02             BPL L793
E920  09 80             ORA #$80
E922  95 D9       L793  STA LDTB1,X
E924  E8                INX
E925  E0 18             CPX #NLINES-1
E927  D0 EF             BNE L789
E929  A5 F1             LDA LDTB1+NLINES-1
E92B  09 80             ORA #$80
E92D  85 F1             STA LDTB1+NLINES-1
E92F  A5 D9             LDA LDTB1       ;DOUBLE LINE?
E931  10 C3             BPL L724        ;YES, SCROLL AGAIN
E933              ;
E933  E6 D6             INC TBLX
E935  EE A5 02          INC $02A5
E938  A9 7F             LDA #$7F        ;CHECK FOR CONTROL KEY
E93A  8D 00 DC          STA COLM        ;DROP LINE 2 ON PORT B
```

```
LOC   CODE        LINE

E93D  AD 01 DC            LDA ROWS
E940  C9 FB               CMP #$FB        ;SLOW SCROLL KEY?(CONTROL)
E942  08                  PHP             ;SAVE STATUS. RESTORE PORT B
E943  A9 7F               LDA #$7F        ;FOR STOP KEY CHECK
E945  8D 00 DC            STA COLM
E948  28                  PLP
E949  D0 0B               BNE L796
E94B                ;
E94B  A0 00               LDY #$00
E94D  EA            L792  NOP              ;DELAY
E94E  CA                  DEX
E94F  D0 FC               BNE L792
E951  88                  DEY
E952  D0 F9               BNE L792
E954  84 C6               STY NDX         ;CLEAR KEY QUEUE BUFFER
E956                ;
E956  A6 D6         L796  LDX TBLX
E958                ;
E958  68            L795  PLA
E959  85 AF               STA EAH
E95B  68                  PLA
E95C  85 AE               STA EAL
E95E  68                  PLA
E95F  85 AD               STA SAH
E961  68                  PLA
E962  85 AC               STA SAL
E964  60                  RTS
E965                ;
E965  A6 D6         L803  LDX TBLX
E967  E8            L768  INX
E968  B5 D9               LDA LDTB1,X     ;FIND LAST DISPLAY LINE
E96A  10 FB               BPL L768
E96C  8E A5 02            STX $02A5       ;FOUND IT
E96F                ;GENERATE A NEW LINE
E96F  E0 18               CPX #NLINES-1   ;IS ONE LINE FROM BOTTOM?
E971  F0 0E               BEQ L722        ;YES, CLEAR LAST
E973  90 0C               BCC L722        ;<NLINES, INSERT LINE
E975  20 EA E8            JSR L787        ;SCROLL EVERYTHING
E978  AE A5 02            LDX $02A5
E97B  CA                  DEX
E97C  C6 D6               DEC TBLX
E97E  4C DA E6            JMP L721
E981  A5 AC         L722  LDA SAL
E983  48                  PHA
E984  A5 AD               LDA SAH
E986  48                  PHA
E987  A5 AE               LDA EAL
E989  48                  PHA
E98A  A5 AF               LDA EAH
E98C  48                  PHA
E98D  A2 19               LDX #NLINES
E98F  CA            L797  DEX
E990  20 F0 E9            JSR L804        ;SET UP TO ADDR
E993  EC A5 02            CPX $02A5
E996  90 0E               BCC L799
E998  F0 0C               BEQ L799        ;BRANCH IF FINISHED
E99A  BD EF EC            LDA $ECEF,X     ;SET FROM ADDR
E99D  85 AC               STA SAL
E99F  B5 D8               LDA LDTB1-1,X
E9A1  20 C8 E9            JSR L800        ;SCROLL THIS LINE DOWN
E9A4  30 E9               BMI L797
```

```
LOC   CODE        LINE

E9A6  20 FF E9    L799   JSR L726
E9A9  A2 17              LDX #NLINES-2
E9AB  EC A5 02    L798   CPX $02A5       ;DONE?
E9AE  90 0F              BCC L801        ;YES
E9B0  B5 DA              LDA LDTB1+1,X
E9B2  29 7F              AND #$7F
E9B4  B4 D9              LDY LDTB1,X     ;WAS IT CONTINUED?
E9B6  10 02              BPL L802        ;YES
E9B8  09 80              ORA #$80
E9BA  95 DA       L802   STA LDTB1+1,X
E9BC  CA                 DEX
E9BD  D0 EC              BNE L798
E9BF  AE A5 02    L801   LDX $02A5
E9C2  20 DA E6           JSR L721
E9C5  4C 58 E9           JMP L795
E9C8              ;
E9C8              ;SCROLL LINE FROM SAL TO PNT
E9C8              ;AND COLOURS FROM EAL TO USER
E9C8              ;
E9C8  29 03       L800   AND #$03         ;CLEAR ANY GARBAGE
E9CA  0D 88 02           ORA HIBASE      ;PUT IN HI ORDER BITS
E9CD  85 AD              STA SAH
E9CF  20 E0 E9           JSR L805        ;COLOUR TO & FROM ADDRS
E9D2  A0 27              LDY #LLEN-1
E9D4  B1 AC       L790   LDA (SAL),Y
E9D6  91 D1              STA (PNT),Y
E9D8  B1 AE              LDA (EAL),Y
E9DA  91 F3              STA (USER),Y
E9DC  88                 DEY
E9DD  10 F5              BPL L790
E9DF  60                 RTS
E9E0              ;
E9E0              ;DO COLOUR TO AND FROM ADDRESSES
E9E0              ;AND CHARACTER TO AND FROM ADDRESSES
E9E0              ;
E9E0  20 24 EA    L805   JSR L811
E9E3  A5 AC              LDA SAL         ;CHARACTER FROM
E9E5  85 AE              STA EAL         ;MAKE COLOUR FROM
E9E7  A5 AD              LDA SAH
E9E9  29 03              AND #$03
E9EB  09 D8              ORA #>CBMCOL
E9ED  85 AF              STA EAH
E9EF  60                 RTS
E9F0              ;
E9F0              ;SET UP PNT FROM .X
E9F0              ;
E9F0  BD F0 EC    L804   LDA $ECF0,X
E9F3  85 D1              STA PNT
E9F5  B5 D9              LDA LDTB1,X
E9F7  29 03              AND #$03
E9F9  0D 88 02           ORA HIBASE
E9FC  85 D2              STA PNT+1
E9FE  60                 RTS
E9FF              ;
E9FF              ;CLEAR THE LINE POINTED TO BY .X
E9FF              ;
E9FF  A0 27       L726   LDY #$27
EA01  20 F0 E9           JSR L804        ;SET UP SCREEN POINTER
EA04  20 24 EA           JSR L811        ;SET UP COLOUR POINTER
EA07  A9 20       L683   LDA #$20         ;SPACE CHAR
EA09  91 D1              STA (PNT),Y
```

```
LOC   CODE        LINE

EA0B  20 DA E4            JSR L675
EA0E  EA                  NOP
EA0F  88                  DEY
EA10  10 F5               BPL L683
EA12  60                  RTS
EA13                ;
EA13                ;PUT A CHAR ON THE SCREEN
EA13                ;
EA13  A8            L807  TAY               ;SAVE CHAR
EA14  A9 02               LDA #$02
EA16  85 CD               STA BLNCT         ;BLINK CURSOR
EA18  20 24 EA            JSR L811          ;SET COLOUR PTR
EA1B  98                  TYA               ;RESTORE COLOUR
EA1C  A4 D3         L695  LDY PNTR           ;GET COLUMN
EA1E  91 D1               STA (PNT),Y       ;CHAR TO SCREEN
EA20  8A                  TXA
EA21  91 F3               STA (USER),Y      ;COLOUR TO SCREEN
EA23  60                  RTS
EA24                ;
EA24  A5 D1         L811  LDA PNT            ;GENERATE COLOUR PTR
EA26  85 F3               STA USER
EA28  A5 D2               LDA PNT+1
EA2A  29 03               AND #$03
EA2C  09 D8               ORA #>CBMCOL      ;64 COLOUR RAM
EA2E  85 F4               STA USER+1
EA30  60                  RTS
EA31                ;
EA31                ;INTERRUPT ROUTINE FOR CLOCK ETC
EA31                ;
EA31  20 EA FF      KEY   JSR L71            ;UPDATE JIFFY CLOCK
EA34  A5 CC               LDA BLNSW         ;BLINKING CURSOR?
EA36  D0 29               BNE L810          ;NO
EA38  C6 CD               DEC BLNCT         ;TIME TO BLINK?
EA3A  D0 25               BNE L810          ;NO
EA3C  A9 14               LDA #$14          ;RESET BLINK COUNTER
EA3E  85 CD               STA BLNCT
EA40  A4 D3               LDY PNTR          ;CURSOR POSITION
EA42  46 CF               LSR BLNON         ;CARRY SET IF ORIG CHAR
EA44  AE 87 02            LDX GDCOL         ;GET CHAR ORIG COLOUR
EA47  B1 D1               LDA (PNT),Y       ;GET CHAR
EA49  B0 11               BCS L745          ;NOT NEEDED
EA4B  E6 CF               INC BLNON         ;SET TO 1
EA4D  85 CE               STA GDBLN         ;SAVE ORIG CHAR
EA4F  20 24 EA            JSR L811
EA52  B1 F3               LDA (USER),Y      ;GET ORIG COLOUR
EA54  8D 87 02            STA GDCOL         ;SAVE IT
EA57  AE 86 02            LDX COLOR         ;BLINK IN THIS COLOUR
EA5A  A5 CE               LDA GDBLN         ;WITH ORIG CHAR
EA5C  49 80         L745  EOR #$80           ;BLINK IT
EA5E  20 1C EA            JSR L695          ;DISPLAY IT
EA61  A5 01         L810  LDA $01            ;GET CASSETTE SWITCH
EA63  29 10               AND #$10          ;SWITCH DOWN?
EA65  F0 0A               BEQ L809          ;YES
EA67  A0 00               LDY #$00
EA69  84 C0               STY CAS1          ;CASSETTE OFF SWITCH
EA6B  A5 01               LDA $01
EA6D  09 20               ORA #$20
EA6F  D0 08               BNE L812          ;MOTOR IS OFF
EA71                ;
EA71  A5 C0         L809  LDA CAS1
EA73  D0 06               BNE L813
```

```
LOC   CODE        LINE

EA75  A5 01             LDA $01
EA77  29 1F             AND #$1F        ;TURN MOTOR OFF
EA79  85 01       L812  STA $01
EA7B  20 87 EA    L813  JSR L814         ;SCAN KEYBOARD
EA7E  AD 0D DC          LDA D1ICR       ;CLEAR INTERRUPT FLAG
EA81  68                PLA             ;RESTORE REGISTERS
EA82  A8                TAY
EA83  68                PLA
EA84  AA                TAX
EA85  68                PLA
EA86  40                RTI             ;EXIT FROM IRQ ROUTINES
EA87              ;
EA87              ;******GENERAL KEYBOARD SCAN******
EA87              ;
EA87  A9 00       L814  LDA #$00
EA89  8D 8D 02          STA SHFLAG
EA8C  A0 40             LDY #$40        ;LAST KEY INDEX
EA8E  84 CB             STY SFDX        ;NULL KEY FOUND
EA90  8D 00 DC          STA COLM        ;RAISE ALL LINES
EA93  AE 01 DC          LDX ROWS        ;CHECK FOR A KEY DOWN
EA96  E0 FF             CPX #$FF        ;KEYS DOWN?
EA98  F0 61             BEQ L824        ;NONE
EA9A  A8                TAY
EA9B  A9 81             LDA #$81        ;SET INDIRECT KEYSCAN
EA9D  85 F5             STA KEYTAB      ;TABLE
EA9F  A9 EB             LDA #$EB
EAA1  85 F6             STA KEYTAB+1
EAA3  A9 FE             LDA #$FE        ;START WITH 1ST COLUMN
EAA5  8D 00 DC          STA COLM
EAA8  A2 08       L815  LDX #$08         ;8 ROW KEYBOARD
EAAA  48                PHA
EAAB  AD 01 DC    L823  LDA ROWS
EAAE  CD 01 DC          CMP ROWS        ;DEBOUNCE KEYBOARD
EAB1  D0 F8             BNE L823
EAB3  4A          L817  LSR A            ;LOOK FOR KEYDOWN
EAB4  B0 16             BCS L820        ;NONE
EAB6  48                PHA
EAB7  B1 F5             LDA (KEYTAB),Y ;GET CHAR CODE
EAB9  C9 05             CMP #$05
EABB  B0 0C             BCS L822        ;NOT SPECIAL KEY
EABD  C9 03             CMP #$03        ;STOP KEY?
EABF  F0 08             BEQ L822        ;YES
EAC1  0D 8D 02          ORA SHFLAG
EAC4  8D 8D 02          STA SHFLAG      ;PUT SHIFT BIT IN FLAG BYTE
EAC7  10 02             BPL L819
EAC9  84 CB       L822  STY SFDX         ;SAVE KEY NUMBER
EACB  68          L819  PLA
EACC  C8          L820  INY
EACD  C0 41             CPY #$41
EACF  B0 0B             BCS L818        ;FINISHED
EAD1  CA                DEX
EAD2  D0 DF             BNE L817
EAD4  38                SEC
EAD5  68                PLA
EAD6  2A                ROL A
EAD7  8D 00 DC          STA COLM        ;NEXT COLUMN
EADA  D0 CC             BNE L815        ;ALWAYS
EADC  68          L818  PLA
EADD  6C 8F 02          JMP ($028F)     ;EVALUATE SHIFT FUNCTIONS
EAE0  A4 CB       L821  LDY SFDX        ;GET KEY INDEX
EAE2  B1 F5             LDA (KEYTAB),Y ;GET CHAR CODE
```

```
LOC   CODE        LINE

EAE4  AA                  TAX             ;SAVE THE CHAR
EAE5  C4 C5               CPY LSTX        ;SAME AS LAST?
EAE7  F0 07               BEQ L832        ;YES
EAE9  A0 10               LDY #$10        ;NO, RESET DELAY
EAEB  8C 8C 02            STY DELAY       ;BEFORE REPEAT
EAEE  D0 36               BNE L828        ;ALWAYS
EAF0  29 7F       L832    AND #$7F         ;UNSHIFT IT
EAF2  2C 8A 02            BIT RPTFLG      ;REPEAT DISABLE?
EAF5  30 16               BMI L816        ;YES
EAF7  70 49               BVS L825
EAF9  C9 7F               CMP #$7F        ;NO KEYS?
EAFB  F0 29       L824    BEQ L828         ;YES, GET OUT
EAFD  C9 14               CMP #$14        ;INSRT/DEL?
EAFF  F0 0C               BEQ L816        ;YES, REPEAT
EB01  C9 20               CMP #$20        ;SPACE?
EB03  F0 08               BEQ L816        ;YES
EB05  C9 1D               CMP #$1D        ;CRSR LEFT/RIGHT?
EB07  F0 04               BEQ L816        ;YES
EB09  C9 11               CMP #$11        ;CRSR UP/DOWN?
EB0B  D0 35               BNE L825        ;NO, EXIT
EB0D  AC 8C 02    L816    LDY DELAY        ;TIME TO REPEAT?
EB10  F0 05               BEQ L826        ;YES
EB12  CE 8C 02            DEC DELAY
EB15  D0 2B               BNE L825
EB17  CE 8B 02    L826    DEC KOUNT        ;NEXT REPEAT?
EB1A  D0 26               BNE L825        ;NO
EB1C  A0 04               LDY #$04        ;YES, RESET CTR
EB1E  8C 8B 02            STY KOUNT
EB21  A4 C6               LDY NDX         ;NO REPEAT IF QUEUE FULL
EB23  88                  DEY
EB24  10 1C               BPL L825
EB26  A4 CB       L828    LDY SFDX         ;GET INDEX OF KEY
EB28  84 C5               STY LSTX        ;SAVE TO KEY FOUND
EB2A  AC 8D 02            LDY SHFLAG      ;UPDATE SHIFT STATUS
EB2D  8C 8E 02            STY LSTSHF
EB30  E0 FF               CPX #$FF        ;NULL OR NO KEY?
EB32  F0 0E               BEQ L825        ;YES
EB34  8A                  TXA             ;NEED .X AS INDEX
EB35  A6 C6               LDX NDX         ;GET # CHARS IN KEY QUEUE
EB37  EC 89 02            CPX XMAX        ;BUFFER FULL?
EB3A  B0 06               BCS L825        ;YES, NO MORE INSERT
EB3C  9D 77 02            STA KEYD,X      ;PUT IN KEY QUEUE
EB3F  E8                  INX
EB40  86 C6               STX NDX         ;NO IN KEY QUEUE
EB42  A9 7F       L825    LDA #$7F        ;SETUP FOR STOP KEY SENSE
EB44  8D 00 DC            STA COLM
EB47  60                  RTS
EB48              ;
EB48              ;SHIFT LOGIC
EB48              ;
EB48  AD 8D 02    SHFLOG  LDA SHFLAG
EB4B  C9 03               CMP #$03
EB4D  D0 15               BNE L827
EB4F  CD 8E 02            CMP LSTSHF
EB52  F0 EE               BEQ L825
EB54  AD 91 02            LDA MODE
EB57  30 1D               BMI L831
EB59  AD 18 D0            LDA VICREG+24
EB5C  49 02               EOR #$02
EB5E  8D 18 D0            STA VICREG+24
EB61  4C 76 EB            JMP L831
```

```
LOC   CODE        LINE

EB64  0A          L827   ASL A
EB65  C9 08               CMP #$08
EB67  90 02               BCC L829
EB69  A9 06               LDA #$06
EB6B  AA          L829   TAX
EB6C  BD 79 EB            LDA $EB79,X
EB6F  85 F5               STA KEYTAB
EB71  BD 7A EB            LDA $EB7A,X
EB74  85 F6               STA KEYTAB+1
EB76  4C E0 EA    L831    JMP L821
EB79                      .END
EB79                      .LIB K3
EB79  81 EB               .WOR MODE1      ;KEYBOARD MODE
EB7B  C2 EB               .WOR MODE2      ; DISPATCH
EB7D  03 EC               .WOR MODE3
EB7F  78 EC               .WOR CONTRL     ;CONTROL KEYS
EB81              ;
EB81              MODE1
EB81              ;DEL,RETURN,RT CRSR,F7,F1,F3,F5,CRSR DWN
EB81  14                  .BYT $14,$0D,$1D,$88,$85,$86,$87,$11
EB82  0D
EB83  1D
EB84  88
EB85  85
EB86  86
EB87  87
EB88  11
EB89              ;3,W,A,4,Z,S,E,L.SHIFT
EB89  33                  .BYT $33,$57,$41,$34,$5A,$53,$45,$01
EB8A  57
EB8B  41
EB8C  34
EB8D  5A
EB8E  53
EB8F  45
EB90  01
EB91              ;5,R,D,6,C,F,T,X
EB91  35                  .BYT $35,$52,$44,$36,$43,$46,$54,$58
EB92  52
EB93  44
EB94  36
EB95  43
EB96  46
EB97  54
EB98  58
EB99              ;7,Y,G,8,B,H,U,V
EB99  37                  .BYT $37,$59,$47,$38,$42,$48,$55,$56
EB9A  59
EB9B  47
EB9C  38
EB9D  42
EB9E  48
EB9F  55
EBA0  56
EBA1              ;9,I,J,0,M,K,O,N
EBA1  39                  .BYT $39,$49,$4A,$30,$4D,$4B,$4F,$4E
EBA2  49
EBA3  4A
EBA4  30
EBA5  4D
EBA6  4B
```

```
LOC   CODE        LINE

EBA7  4F
EBA8  4E
EBA9              ;+,P,L,-,.,:,@,,
EBA9  2B                .BYT $2B,$50,$4C,$2D,$2E,$3A,$40,$2C
EBAA  50
EBAB  4C
EBAC  2D
EBAD  2E
EBAE  3A
EBAF  40
EBB0  2C
EBB1              ;£,*,;,HOME,R.SHIFT,=,↑,/
EBB1  5C                .BYT $5C,$2A,$3B,$13,$01,$3D,$5E,$2F
EBB2  2A
EBB3  3B
EBB4  13
EBB5  01
EBB6  3D
EBB7  5E
EBB8  2F
EBB9              ;1,←,CTRL,2,SPACE,CBM KEY,Q,STOP
EBB9  31                .BYT $31,$5F,$04,$32,$20,$02,$51,$03
EBBA  5F
EBBB  04
EBBC  32
EBBD  20
EBBE  02
EBBF  51
EBC0  03
EBC1  FF                .BYT $FF        ;END OF TABLE NULL
EBC2              MODE2
EBC2              ;SHIFTED   DEL,CR,RIGHT,F7,F1,F3,F5,DWN
EBC2  94                .BYT $94,$8D,$9D,$8C,$89,$8A,$8B,$91
EBC3  8D
EBC4  9D
EBC5  8C
EBC6  89
EBC7  8A
EBC8  8B
EBC9  91
EBCA              ;SHIFTED   3,W,A,4,Z,S,E,L.SHIFT
EBCA  23                .BYT $23,$D7,$C1,$24,$DA,$D3,$C5,$01
EBCB  D7
EBCC  C1
EBCD  24
EBCE  DA
EBCF  D3
EBD0  C5
EBD1  01
EBD2              ;SHIFTED   5,R,D,6,C,F,T,X
EBD2  25                .BYT $25,$D2,$C4,$26,$C3,$C6,$D4,$D8
EBD3  D2
EBD4  C4
EBD5  26
EBD6  C3
EBD7  C6
EBD8  D4
EBD9  D8
EBDA              ;SHIFTED   7,Y,G,8,B,H,U,V
EBDA  27                .BYT $27,$D9,$C7,$28,$C2,$C8,$D5,$D6
EBDB  D9
```

```
LOC   CODE        LINE

EBDC  C7
EBDD  28
EBDE  C2
EBDF  C8
EBE0  D5
EBE1  D6
EBE2              ;SHIFTED   9,I,J,0,M,K,O,N
EBE2  29                .BYT $29,$C9,$CA,$30,$CD,$CB,$CF,$CE
EBE3  C9
EBE4  CA
EBE5  30
EBE6  CD
EBE7  CB
EBE8  CF
EBE9  CE
EBEA              ;SHIFTED   +,P,L,-,.,:,@,,
EBEA  DB                .BYT $DB,$D0,$CC,$DD,$3E,$5B,$BA,$3C
EBEB  D0
EBEC  CC
EBED  DD
EBEE  3E
EBEF  5B
EBF0  BA
EBF1  3C
EBF2              ;SHIFTED   £,*,;,HOME,R.SHIFT,=,↑,/
EBF2  A9                .BYT $A9,$C0,$5D,$93,$01,$3D,$DE,$3F
EBF3  C0
EBF4  5D
EBF5  93
EBF6  01
EBF7  3D
EBF8  DE
EBF9  3F
EBFA              ;SHIFTED   1,←,CTRL,2,SPACE,CBM KEY,Q,STOP
EBFA  21                .BYT $21,$5F,$04,$22,$A0,$02,$D1,$83
EBFB  5F
EBFC  04
EBFD  22
EBFE  A0
EBFF  02
EC00  D1
EC01  83
EC02  FF                .BYT $FF        ;END OF TABLE NULL
EC03              MODE3
EC03              ;LOGO AND   DEL,CR,RIGHT,F7,F1,F3,F5,DWN
EC03  94                .BYT $94,$8D,$9D,$8C,$89,$8A,$8B,$91
EC04  8D
EC05  9D
EC06  8C
EC07  89
EC08  8A
EC09  8B
EC0A  91
EC0B              ;LOGO AND   3(LT.RED),W,A,4(DK.GRY),Z,S,E,L.SHIFT
EC0B  96                .BYT $96,$B3,$B0,$97,$AD,$AE,$B1,$01
EC0C  B3
EC0D  B0
EC0E  97
EC0F  AD
EC10  AE
EC11  B1
```

```
LOC   CODE        LINE

EC12  01
EC13              ;LOGO AND   5(MD.GRY),R,D,6(LT.GRN),C,F,T,X
EC13  98                .BYT $98,$B2,$AC,$99,$BC,$BB,$A3,$BD
EC14  B2
EC15  AC
EC16  99
EC17  BC
EC18  BB
EC19  A3
EC1A  BD
EC1B              ;LOGO AND   7(LT.BLU),Y,G,8(LT.GRY),B,H,U,V
EC1B  9A                .BYT $9A,$B7,$A5,$9B,$BF,$B4,$B8,$BE
EC1C  B7
EC1D  A5
EC1E  9B
EC1F  BF
EC20  B4
EC21  B8
EC22  BE
EC23              ;LOGO AND   9,I,J,0,M,K,O,N
EC23  29                .BYT $29,$A2,$B5,$30,$A7,$A1,$B9,$AA
EC24  A2
EC25  B5
EC26  30
EC27  A7
EC28  A1
EC29  B9
EC2A  AA
EC2B              ;LOGO AND   +,P,L,-,.,:,@,,
EC2B  A6                .BYT $A6,$AF,$B6,$DC,$3E,$5B,$A4,$3C
EC2C  AF
EC2D  B6
EC2E  DC
EC2F  3E
EC30  5B
EC31  A4
EC32  3C
EC33              ;LOGO AND   £,*,;,HOME,R.SHIFT,=,↑,/
EC33  A8                .BYT $A8,$DF,$5D,$93,$01,$3D,$DE,$3F
EC34  DF
EC35  5D
EC36  93
EC37  01
EC38  3D
EC39  DE
EC3A  3F
EC3B              ;LOGO AND   1(ORNG),←,CTRL,2(BRWN),SPACE,,Q,STOP
EC3B  81                .BYT $81,$5F,$04,$95,$A0,$02,$AB,$83
EC3C  5F
EC3D  04
EC3E  95
EC3F  A0
EC40  02
EC41  AB
EC42  83
EC43  FF                .BYT $FF        ;END OF TABLE NULL
EC44              ;
EC44  C9 0E       L830  CMP #$0E         ;LOWER CASE?
EC46  D0 07             BNE L758        ;NO
EC48  AD 18 D0          LDA VICREG+24   ;SET LOWER
EC4B  09 02             ORA #$02
```

```
LOC   CODE        LINE

EC4D  D0 09             BNE L779        ;ALWAYS
EC4F  C9 8E       L758  CMP #$8E         ;UPPER CASE?
EC51  D0 0B             BNE L837        ;NO
EC53  AD 18 D0          LDA VICREG+24   ;SET LOWER
EC56  29 FD             AND #$FD
EC58  8D 18 D0    L779  STA VICREG+24    ;STORE IT
EC5B  4C A8 E6    L833  JMP L715
EC5E              ;
EC5E  C9 08       L837  CMP #$08         ;LOCK CASE TOGGLE?
EC60  D0 07             BNE L834        ;NO
EC62  A9 80             LDA #$80        ;SET LOCK SWITCH
EC64  0D 91 02          ORA MODE
EC67  30 09             BMI L835        ;ALWAYS
EC69  C9 09       L834  CMP #$09         ;UNLOCK CASE TOGGLE?
EC6B  D0 EE             BNE L833        ;NO
EC6D  A9 7F             LDA #$7F        ;UNSET LOCK SWITCH
EC6F  2D 91 02          AND MODE
EC72  8D 91 02    L835  STA MODE         ;STORE IT
EC75  4C A8 E6          JMP L715
EC78              CONTRL
EC78              ;........,
EC78  FF                .BYT $FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF
EC79  FF
EC7A  FF
EC7B  FF
EC7C  FF
EC7D  FF
EC7E  FF
EC7F  FF
EC80              ;CONTROL   3(RED),W,A,4(CYN),Z,S(HOME),E(WHT),
EC80  1C                .BYT $1C,$17,$01,$9F,$1A,$13,$05,$FF
EC81  17
EC82  01
EC83  9F
EC84  1A
EC85  13
EC86  05
EC87  FF
EC88              ;CONTROL   5(PUR),R(RVS),D,6(GRN),C,F,T(DEL),X
EC88  9C                .BYT $9C,$12,$04,$1E,$03,$06,$14,$18
EC89  12
EC8A  04
EC8B  1E
EC8C  03
EC8D  06
EC8E  14
EC8F  18
EC90              ;CONTROL   7(BLU),Y,G,8(YEL),B,H(LOCK),U,V
EC90  1F                .BYT $1F,$19,$07,$9E,$02,$08,$15,$16
EC91  19
EC92  07
EC93  9E
EC94  02
EC95  08
EC96  15
EC97  16
EC98              ;CONTROL   9(RVS),I(UNL),J,0(OFF),M(CR),K,O,N(LOW)
EC98  12                .BYT $12,$09,$0A,$92,$0D,$0B,$0F,$0E
EC99  09
EC9A  0A
EC9B  92
```

```
LOC   CODE        LINE


EC9C  0D
EC9D  0B
EC9E  0F
EC9F  0E
ECA0              ;CONTROL    ,F,L,,,:,@,
ECA0  FF                .BYT $FF,$10,$0C,$FF,$FF,$1B,$00,$FF
ECA1  10
ECA2  0C
ECA3  FF
ECA4  FF
ECA5  1B
ECA6  00
ECA7  FF
ECA8              ;CONTROL    £(RED),,;,(RIGHT),,,=(BLU),↑(GRN),
ECA8  1C                .BYT $1C,$FF,$1D,$FF,$FF,$1F,$1E,$FF
ECA9  FF
ECAA  1D
ECAB  FF
ECAC  FF
ECAD  1F
ECAE  1E
ECAF  FF
ECB0              ;CONTROL    1(BLK),F,,2(WHT),,,Q(DWN),
ECB0  90                .BYT $90,$06,$FF,$05,$FF,$FF,$11,$FF
ECB1  06
ECB2  FF
ECB3  05
ECB4  FF
ECB5  FF
ECB6  11
ECB7  FF
ECB8  FF                .BYT $FF        ;END OF TABLE NULL
ECB9              ;
ECB9              ;INITIAL VIC CHIP SETTINGS
ECB9              ;
ECB9  00                .BYT $00,$00,$00,$00,$00,$00,$00,$00
ECBA  00
ECBB  00
ECBC  00
ECBD  00
ECBE  00
ECBF  00
ECC0  00
ECC1  00                .BYT $00,$00,$00,$00,$00,$00,$00,$00
ECC2  00
ECC3  00
ECC4  00
ECC5  00
ECC6  00
ECC7  00
ECC8  00
ECC9  00                .BYT $00,$9B,$37,$00,$00,$00,$08,$00
ECCA  9B
ECCB  37
ECCC  00
ECCD  00
ECCE  00
ECCF  08
ECD0  00
ECD1  14                .BYT $14,$0F,$00,$00,$00,$00,$00,$00
ECD2  0F
```

```
LOC   CODE        LINE


ECD3  00
ECD4  00
ECD5  00
ECD6  00
ECD7  00
ECD8  00
ECD9  0E                .BYT LTBLUE,BLUE,$01,$02,$03,$04,$00,$01
ECDA  06
ECDB  01
ECDC  02
ECDD  03
ECDE  04
ECDF  00
ECE0  01
ECE1  02                .BYT $02,$03,$04,$05,$06,$07
ECE2  03
ECE3  04
ECE4  05
ECE5  06
ECE6  07
ECE7                ;
ECE7                ;SHIFT STOP KEY
ECE7                ;
ECE7  4C 4F             .BYT 'LOAD',$D
ECEB  0D
ECEC  52 55 4E          .BYT 'RUN',$D
ECEF  0D
ECF0                ;
ECF0                ;SCREEN LINES LO BYTE TABLE
ECF0                ;
ECF0  00                .BYT $00,$28,$50,$78,$A0,$C8,$F0,$18
ECF1  28
ECF2  50
ECF3  78
ECF4  A0
ECF5  C8
ECF6  F0
ECF7  18
ECF8  40                .BYT $40,$68,$90,$B8,$E0,$08,$30,$58
ECF9  68
ECFA  90
ECFB  B8
ECFC  E0
ECFD  08
ECFE  30
ECFF  58
ED00  80                .BYT $80,$A8,$D0,$F8,$20,$48,$70,$98
ED01  A8
ED02  D0
ED03  F8
ED04  20
ED05  48
ED06  70
ED07  98
ED08  C0                .BYT $C0
ED09                    .END
ED09                    .LIB K4
ED09                ;***************************************
ED09                ;*COMMAND SERIAL BUS TO TALK.
ED09                ;*THE ACCUMULATOR MUST BE LOADED WITH THE
ED09                ;*DEVICE NUMBER THAT YOU WISH TO TALK.
```

```
LOC   CODE        LINE

ED09              ;*****************************************
ED09  09 40       L836   ORA #$40         ;MAKE ADDR TALK
ED0B  2C                 .BYT $2C         ;SKIP NEXT COMMAND
ED0C              ;*****************************************
ED0C              ;*COMMAND SERIAL BUS TO LISTEN.
ED0C              ;*TO USE THIS ROUTINE, THE ACCUMULATOR MUST
ED0C              ;*FIRST BE LOADED WITH THE DEVICE NUMBER THAT
ED0C              ;*YOU WISH TO LISTEN (RECEIVE DATA).
ED0C              ;*****************************************
ED0C  09 20       L966   ORA #$20         ;MAKE ADDR LISTEN
ED0E  20 A4 F0           JSR L921         ;PROTECT FROM RS323 NMI
ED11  48          L980   PHA
ED12  24 94              BIT C3PO         ;CHAR IN BUFFER?
ED14  10 0A              BPL L864         ;NO
ED16              ;
ED16              ;SEND BUFFERED CHAR
ED16              ;
ED16  38                 SEC              ;SET EOI FLAG
ED17  66 A3              ROR R2D2
ED19  20 40 ED           JSR L859         ;SEND LAST CHAR
ED1C  46 94              LSR C3PO         ;BUFFER CLEAR
ED1E  46 A3              LSR R2D2         ;CLEAR EOI FLAG
ED20  68          L864   PLA               ;TALK/LISTEN ADDR
ED21  85 95              STA BSOUR
ED23  78                 SEI
ED24  20 97 EE           JSR L844
ED27  C9 3F              CMP #$3F         ;CLKHI ONLY ON UNLISTEN
ED29  D0 03              BNE L839
ED2B  20 85 EE           JSR L875
ED2E  AD 00 DD    L839   LDA D2DPA        ;ASSERT ATTENTION
ED31  09 08              ORA #$08
ED33  8D 00 DD           STA D2DPA
ED36              ;
ED36  78          L842   SEI
ED37  20 8E EE           JSR L843         ;SET CLOCK LINE LOW
ED3A  20 97 EE           JSR L844
ED3D  20 B3 EE           JSR L846         ;DELAY 1 MS
ED40  78          L859   SEI               ;DISABLE IRQ
ED41  20 97 EE           JSR L844         ;MAKE SURE DATA IS RELEASED
ED44  20 A9 EE           JSR L854         ;DATA SHOULD BE LOW
ED47  B0 64              BCS L856
ED49  20 85 EE           JSR L875         ;CLOCK LINE HI
ED4C  24 A3              BIT R2D2         ;EOI FLAG TEST
ED4E  10 0A              BPL L850
ED50              ;DO EOI
ED50  20 A9 EE    L840   JSR L854          ;WAIT FOR DATA HI
ED53  90 FB              BCC L840
ED55  20 A9 EE    L849   JSR L854          ;WAIT FOR DATA LO
ED58  B0 FB              BCS L849
ED5A  20 A9 EE    L850   JSR L854          ;WAIT FOR DATA HI
ED5D  90 FB              BCC L850
ED5F  20 8E EE           JSR L843         ;SET CLOCK LO
ED62              ;
ED62              ;SET TO SEND DATA
ED62              ;
ED62  A9 08              LDA #$08         ;COUNT 8 BITS
ED64  85 A5              STA COUNT
ED66  AD 00 DD    L848   LDA D2DPA         ;DEBOUNCE BUS
ED69  CD 00 DD           CMP D2DPA
ED6C  D0 FB              BNE L848
ED6E  0A                 ASL A
```

```
LOC   CODE        LINE

ED6F  90 3F             BCC L847        ;DATA MUST BE HI
ED71  66 95             ROR BSOUR       ;NEXT BIT INTO CARRY
ED73  B0 05             BCS L851
ED75  20 A0 EE          JSR L841
ED78  D0 03             BNE L853
ED7A  20 97 EE    L851  JSR L844
ED7D  20 85 EE    L853  JSR L875         ;CLOCK HI
ED80  EA                NOP
ED81  EA                NOP
ED82  EA                NOP
ED83  EA                NOP
ED84  AD 00 DD          LDA D2DPA
ED87  29 DF             AND #$DF        ;DATA HI
ED89  09 10             ORA #$10        ;CLOCK LO
ED8B  8D 00 DD          STA D2DPA
ED8E  C6 A5             DEC COUNT
ED90  D0 D4             BNE L848
ED92  A9 04             LDA #$04        ;SET TIMER FOR 1 MS
ED94  8D 07 DC          STA D1TBH
ED97  A9 19             LDA #$19
ED99  8D 0F DC          STA D1CRB
ED9C  AD 0D DC          LDA D1ICR
ED9F  AD 0D DC    L855  LDA D1ICR
EDA2  29 02             AND #$02
EDA4  D0 0A             BNE L847
EDA6  20 A9 EE          JSR L854
EDA9  B0 F4             BCS L855
EDAB  58                CLI             ;ENABLE IRQ
EDAC  60                RTS
EDAD  A9 80       L856  LDA #$80         ;DEVICE NOT PRESENT
EDAF  2C                .BYT $2C
EDB0  A9 03       L847  LDA #$03         ;FRAMING ERROR
EDB2  20 1C FE    L852  JSR L1217        ;SEND MESSAGE
EDB5  58                CLI             ;ENABLE IRQ
EDB6  18                CLC
EDB7  90 4A             BCC L1004       ;ALWAYS
EDB9              ;**************************************
EDB9              ;*SEND SECONDARY ADDRESS AFTER LISTEN.
EDB9              ;*THIS ROUTINE IS USED TO SEND A SECONDARY
EDB9              ;*ADDRESS AFTER A CALL TO THE LISTEN COMMAND.
EDB9              ;**************************************
EDB9  85 95       L871  STA BSOUR         ;BUFFER CHAR
EDBB  20 36 ED          JSR L842        ;SEND IT
EDBE              ;
EDBE              ;RELEASE ATTENTION
EDBE              ;
EDBE  AD 00 DD    L983  LDA D2DPA
EDC1  29 F7             AND #$F7
EDC3  8D 00 DD          STA D2DPA       ;RELEASE
EDC6  60                RTS
EDC7              ;**************************************
EDC7              ;*SEND TALK SA.
EDC7              ;*THIS ROUTINE IS USED TO SEND A SECONDARY
EDC7              ;*ADDRESS TO A DEVICE THAT HAS ALREADY BEEN
EDC7              ;*COMMANDED TO TALK.
EDC7              ;**************************************
EDC7  85 95       L860  STA BSOUR         ;BUFFER CHAR
EDC9  20 36 ED          JSR L842        ;SEND SA
EDCC              ;
EDCC              ;SHIFT OVER TO LISTENER
EDCC              ;
```

```
LOC   CODE        LINE

EDCC  78          L970   SEI              ;DISABLE IRQ
EDCD  20 A0 EE           JSR L841        ;DATA LINE LO
EDD0  20 BE ED           JSR L983
EDD3  20 85 EE           JSR L875        ;CLOCK LINE HI
EDD6  20 A9 EE    L968   JSR L854         ;WAIT FOR CLOCK LO
EDD9  30 FB              BMI L968
EDDB  58                 CLI             ;DONE
EDDC  60                 RTS
EDDD              ;
EDDD              ;BUFFERED OUTPUT TO SERIAL BUS
EDDD              ;
EDDD  24 94       L861   BIT C3PO         ;BUFFERED CHAR?
EDDF  30 05              BMI L949        ;YES, SEND LAST
EDE1  38                 SEC             ;NO
EDE2  66 94              ROR C3PO        ;SET BUFFERED CHAR FLAG
EDE4  D0 05              BNE L862        ;ALWAYS
EDE6              ;
EDE6  48          L949   PHA              ;SAVE CURRENT CHAR
EDE7  20 40 ED           JSR L859        ;SEND LAST CHAR
EDEA  68                 PLA             ;RESTORE CURRENT
EDEB  85 95       L862   STA BSOUR        ;BUFFER IT
EDED  18                 CLC             ;GOOD EXIT
EDEE  60                 RTS
EDEF              ;****************************************
EDEF              ;*SEND UNTALK.
EDEF              ;*THIS ROUTINE SENDS AN 'UNTALK' TO THE SERIAL
EDEF              ;*BUS. IT WILL TELL ALL DEVICES IN TALK
EDEF              ;*MODE TO STOP TALKING (SENDING DATA).
EDEF              ;****************************************
EDEF  78          L863   SEI
EDF0  20 BE EE           JSR L843
EDF3  AD 00 DD           LDA D2DPA       ;PULL ATN
EDF6  09 08              ORA #$08
EDF8  8D 00 DD           STA D2DPA
EDFB  A9 5F              LDA #$5F        ;UNTALK
EDFD  2C                 .BYT $2C        ;SKIP NEXT COMMAND
EDFE              ;****************************************
EDFE              ;*SEND UNLISTEN.
EDFE              ;*THIS ROUTINE SENDS AN 'UNLISTEN' TO
EDFE              ;*THE SERIAL BUS. IT WILL TELL ALL DEVICES
EDFE              ;*IN LISTEN MODE TO STOP LISTENING.
EDFE              ;****************************************
EDFE  A9 3F       L1006  LDA #$3F         ;UNLISTEN COMMAND
EE00  20 11 ED           JSR L980        ;SEND
EE03              ;
EE03              ;RELEASE ALL LINES
EE03              ;
EE03  20 BE ED    L1004  JSR L983         ;RELEASE ATN
EE06              ;
EE06              ;DELAY THEN RELEASE CLOCK AND DATA
EE06              ;
EE06  8A          L858   TXA              ;DELAY APPROX 60 MICRO SECS
EE07  A2 0A              LDX #$0A
EE09  CA          L876   DEX
EE0A  D0 FD              BNE L876
EE0C  AA                 TAX
EE0D  20 85 EE           JSR L875
EE10  4C 97 EE           JMP L844
EE13              ;
EE13              ;INPUT A BYTE FROM SERIAL BUS
EE13              ;
```

```
LOC   CODE        LINE

EE13  78          L865   SEI              ;DISABLE IRQ
EE14  A9 00              LDA #$00         ;SET EOI/ERROR FLAG
EE16  85 A5              STA COUNT
EE18  20 85 EE           JSR L875         ;RELEASE CLOCK LINE
EE1B  20 A9 EE    L943   JSR L854          ;WAIT FOR CLOCK HI
EE1E  10 FB              BPL L943
EE20  A9 01       L866   LDA #$01          ;SET TIMER B FOR 256 US
EE22  8D 07 DC           STA D1TBH
EE25  A9 19              LDA #$19
EE27  8D 0F DC           STA D1CRB
EE2A  20 97 EE           JSR L844
EE2D  AD 0D DC           LDA D1ICR
EE30  AD 0D DC    L872   LDA D1ICR
EE33  29 02              AND #$02         ;CHECK THE TIMER
EE35  D0 07              BNE L868         ;RAN OUT
EE37  20 A9 EE           JSR L854         ;CHECK THE CLOCK LINE
EE3A  30 F4              BMI L872         ;NOT YET
EE3C  10 18              BPL L870         ;YES
EE3E              ;
EE3E
EE3E  A5 A5       L868   LDA COUNT         ;CHECK FOR ERROR
EE40  F0 05              BEQ L867
EE42  A9 02              LDA #$02
EE44  4C B2 ED           JMP L852         ;ST=2, READ TIMEOUT
EE47              ;
EE47              ;TIMER RAN OUT, DO AN EOI
EE47              ;
EE47  20 A0 EE    L867   JSR L841          ;DATA LINE LO
EE4A  20 85 EE           JSR L875         ;DELAY, SET DATA HI
EE4D  A9 40              LDA #$40
EE4F  20 1C FE           JSR L1217        ;OR AN EOI BIT INTO ST
EE52  E6 A5              INC COUNT AND AGAIN FOR ERROR CHECK
EE54  D0 CA              BNE L866
EE56              ;
EE56              ;BYTE TRANSFER
EE56              ;
EE56  A9 08       L870   LDA #$08          ;SET UP COUNTER
EE58  85 A5              STA COUNT
EE5A  AD 00 DD    L869   LDA D2DPA         ;WAIT FOR CLOCK HI
EE5D  CD 00 DD           CMP D2DPA        ;DEBOUNCE
EE60  D0 F8              BNE L869
EE62  0A                 ASL A
EE63  10 F5              BPL L869
EE65  66 A4              ROR BSOUR1       ;ROTATE DATA IN
EE67  AD 00 DD    L873   LDA D2DPA         ;WAIT FOR CLOCK LO
EE6A  CD 00 DD           CMP D2DPA        ;DEBOUNCE
EE6D  D0 F8              BNE L873
EE6F  0A                 ASL A
EE70  30 F5              BMI L873
EE72  C6 A5              DEC COUNT
EE74  D0 E4              BNE L869         ;MORE BITS
EE76  20 A0 EE           JSR L841         ;DATA LO
EE79  24 90              BIT STATUS       ;CHECK FOR EOI
EE7B  50 03              BVC L874         ;NONE
EE7D  20 06 EE           JSR L858         ;DELAY AND DATA HI
EE80  A5 A4       L874   LDA BSOUR1
EE82  58                 CLI              ;ENABLE IRQ
EE83  18                 CLC              ;GOOD EXIT
EE84  60                 RTS
EE85              ;
EE85              ;SET CLOCK LINE HI (INVERTED)
```

```
LOC   CODE        LINE

EE85                ;
EE85  AD 00 DD      L875    LDA D2DPA
EE88  29 EF                 AND #$EF
EE8A  8D 00 DD              STA D2DPA
EE8D  60                    RTS
EE8E                ;
EE8E                ;SET CLOCK LINE LO (INVERTED)
EE8E                ;
EE8E  AD 00 DD      L843    LDA D2DPA
EE91  09 10                 ORA #$10
EE93  8D 00 DD              STA D2DPA
EE96  60                    RTS
EE97                ;
EE97                ;SET DATA LINE HI (INVERTED)
EE97                ;
EE97  AD 00 DD      L844    LDA D2DPA
EE9A  29 DF                 AND #$DF
EE9C  8D 00 DD              STA D2DPA
EE9F  60                    RTS
EEA0                ;
EEA0                ;SET DATA LINE LO (INVERTED)
EEA0                ;
EEA0  AD 00 DD      L841    LDA D2DPA
EEA3  09 20                 ORA #$20
EEA5  8D 00 DD              STA D2DPA
EEA8  60                    RTS
EEA9                ;
EEA9                ;DEBOUNCE THE PIA
EEA9                ;
EEA9  AD 00 DD      L854    LDA D2DPA
EEAC  CD 00 DD              CMP D2DPA
EEAF  D0 F8                 BNE L854
EEB1  0A                    ASL A
EEB2  60                    RTS
EEB3                ;
EEB3                ;DELAY 1 MS
EEB3                ;
EEB3  8A            L846    TXA
EEB4  A2 B8                 LDX #$B8
EEB6  CA            L845    DEX
EEB7  D0 FD                 BNE L845
EEB9  AA                    TAX
EEBA  60                    RTS
EEBB                        .END
EEBB                        .LIB K5
EEBB                ;
EEBB                ;***************************************
EEBB                ;* RS-232 TRANSMIT ROUTINES.
EEBB                ;***************************************
EEBB                ;
EEBB  A5 B4         L877    LDA BITTS       ;CHECK FOR PLACE IN BYTE
EEBD  F0 47                 BEQ L879        ;DONE, START NEXT
EEBF  30 3F                 BMI L883        ;DOING STOP BITS
EEC1  46 B6                 LSR RODATA      ;SHIFT DATA INTO CARRY
EEC3  A2 00                 LDX #$00        ;PREPARE FOR A ZERO
EEC5  90 01                 BCC L1227       ;YES, A ZERO
EEC7  CA                    DEX             ;NO, MAKE $FF
EEC8  8A            L1227   TXA             ;READY TO SEND
EEC9  45 BD                 EOR ROPRTY      ;CALC INTO PARITY
EECB  85 BD                 STA ROPRTY
EECD  C6 B4                 DEC BITTS       ;BIT COUNT DOWN
```

```
LOC   CODE        LINE

EECF  F0 06             BEQ L886        ;WANT A PARITY INSTEAD
EED1  8A          L880  TXA              ;CALC BIT WHOLE TO SEND
EED2  29 04             AND #$04
EED4  85 B5             STA NXTBIT      ;SEND
EED6  60                RTS
EED7              ;
EED7              ;CALCULATE PARITY, NXTBIT=0 UPON ENTRY
EED7              ;
EED7  A9 20       L886  LDA #$20         ;CHECK 6526 REG BITS
EED9  2C 94 02          BIT M26CDR
EEDC  F0 14             BEQ L885        ;NO PARITY, SEND A STOP
EEDE  30 1C             BMI L884        ;NOT REAL PARITY
EEE0  70 14             BVS L882        ;EVEN PARITY
EEE2  A5 BD             LDA ROPRTY      ;CALC ODD PARITY
EEE4  D0 01             BNE L887        ;CORRECT GUESS
EEE6  CA          L881  DEX              ;WRONG, IT'S A ONE
EEE7  C6 B4       L887  DEC BITTS        ;ONE STOP BIT ALWAYS
EEE9  AD 93 02          LDA M26CTR      ;CHECK # OF STOP BITS
EEEC  10 E3             BPL L880        ;ONE
EEEE  C6 B4             DEC BITTS       ;TWO
EEF0  D0 DF             BNE L880
EEF2  E6 B4       L885  INC BITTS        ;COUNTS AS ONE STOP BIT
EEF4  D0 F0             BNE L881        ;JUMP TO FLIP ONE
EEF6  A5 BD       L882  LDA ROPRTY       ;EVEN PARITY?
EEF8  F0 ED             BEQ L887        ;YES, EXIT
EEFA  D0 EA             BNE L881        ;NO, FLIP & EXIT
EEFC              ;
EEFC  70 E9       L884  BVS L887         ;WANTED SPACE
EEFE  50 E6             BVC L881        ;WANTED MARK
EF00              ;
EF00              ;STOP BITS
EF00              ;
EF00  E6 B4       L883  INC BITTS        ;STOP BIT COUNT DOWN
EF02  A2 FF             LDX #$FF        ;SEND STOP BIT
EF04  D0 CB             BNE L880        ;JUMP TO EXIT
EF06              ;
EF06              ;ENTRY TO START BYTE TRANSMIT
EF06              ;
EF06  AD 94 02    L879  LDA M26CDR       ;CHECK FOR 3/X LINE
EF09  4A                LSR A
EF0A  90 07             BCC L878        ;3 LINE, NO CHECK
EF0C  2C 01 DD          BIT D2DPB       ;CHECK FOR:
EF0F  10 1D             BPL L888        ; DSR ERROR
EF11  50 1E             BVC L889        ; CTS ERROR
EF13              ;
EF13              ;SET UP TO SEND NEXT BYTE
EF13              ;
EF13  A9 00       L878  LDA #$00
EF15  85 BD             STA ROPRTY      ;ZERO PARITY
EF17  85 B5             STA NXTBIT      ;SEND START BIT
EF19  AE 98 02          LDX BITNUM      ;GET # OF BITS
EF1C  86 B4             STX BITTS       ;BITTS=# OF BITS+1
EF1E  AC 9D 02          LDY RODBS       ;CHECK BUFFER POINTERS
EF21  CC 9E 02          CPY RODBE
EF24  F0 13             BEQ L890        ;ALL DONE
EF26  B1 F9             LDA (ROBUF),Y   ;GET DATA
EF28  85 B6             STA RODATA      ;INTO BYTE BUFFER
EF2A  EE 9D 02          INC RODBS       ;POINT TO NEXT
EF2D  60                RTS
EF2E              ;
EF2E              ;SET ERRORS
```

```
LOC   CODE        LINE

EF2E                ;
EF2E  A9 40         L888   LDA #$40        ;DSR GONE ERROR
EF30  2C                  .BYT $2C
EF31  A9 10         L889   LDA #$10        ;CTS GONE ERROR
EF33  0D 97 02             ORA RSSTAT
EF36  8D 97 02             STA RSSTAT
EF39                ;
EF39                ;ERRORS TURN OF T1
EF39                ;
EF39  A9 01         L890   LDA #$01        ;KILL T1 NMI
EF3B  8D 0D DD      L891   STA D21CR
EF3E  4D A1 02             EOR $02A1
EF41  09 80                ORA #$80
EF43  8D A1 02             STA $02A1
EF46  8D 0D DD             STA D2ICR
EF49  60                   RTS
EF4A                ;
EF4A                ;CALC # OF BITS TO BE SENT
EF4A                ; RETURNS # OF BITS+1
EF4A                ;
EF4A  A2 09         L899   LDX #$09        ;CALC WORD LENGTH
EF4C  A9 20                LDA #$20
EF4E  2C 93 02             BIT M26CTR
EF51  F0 01                BEQ L1037
EF53  CA                   DEX             ;BIT 5 HI IS A 7 OR 5
EF54  50 02         L1037  BVC L892
EF56  CA                   DEX             ;BIT 6 HI IS A 6 OR 5
EF57  CA                   DEX
EF58  60            L892   RTS
EF59                       .END
EF59                       .LIB K6
EF59                ;
EF59                ;**************************************
EF59                ;* RS-232 RECEIVER ROUTINES
EF59                ;**************************************
EF59                ;
EF59  A6 A9         L893   LDX RINONE       ;START BIT?
EF5B  D0 33                BNE L900         ;YES
EF5D  C6 A8                DEC BITCI        ;CHECK POS IN INPUT
EF5F  F0 36                BEQ L894         ;HAVE A FULL BYTE
EF61  30 0D                BMI L904         ;GETTING STOP BITS
EF63                ;
EF63                ;CALC PARITY
EF63                ;
EF63  A5 A7                LDA INBIT        ;GET DATA UP
EF65  45 AB                EOR RIPRTY       ;CALC PARITY
EF67  85 AB                STA RIPRTY
EF69                ;
EF69                ;SHIFT DATA BIT IN
EF69                ;
EF69  46 A7                LSR INBIT        ;IN BIT POS 0
EF6B  66 AA                ROR RIDATA       ;C INTO DATA
EF6D                ;
EF6D                ;EXIT
EF6D                ;
EF6D  60            L1230  RTS
EF6E                ;
EF6E                ;HAVE STOP BIT SO STORE IN BUFFER
EF6E                ;
EF6E  C6 A8         L898   DEC BITCI         ;NO PARITY
EF70  A5 A7         L904   LDA INBIT         ;GET DATA
```

```
LOC   CODE        LINE

EF72  F0 67             BEQ L906        ;0, AN ERROR
EF74  AD 93 02          LDA M26CTR      ;CHECK CORRECT # STOP BITS
EF77  0A                ASL A
EF78  A9 01             LDA #$01
EF7A  65 A8             ADC BITCI
EF7C  D0 EF             BNE L1230       ;NO, EXIT
EF7E              ;
EF7E              ;ENABLE TO RECEIVE A BYTE
EF7E              ;
EF7E  A9 90       L896  LDA #$90         ;ENABLE CB1 FOR NEXT BYTE
EF80  8D 0D DD          STA D2ICR
EF83  0D A1 02          ORA $02A1       ;FLAG FOR START BIT
EF86  8D A1 02          STA $02A1
EF89  85 A9             STA RINONE
EF8B  A9 02             LDA #$02
EF8D  4C 3B EF          JMP L891        ;DISABLE T2
EF90              ;
EF90              ;RECEIVER START BIT CHECK
EF90              ;
EF90  A5 A7       L900  LDA INBIT        ;SPACE?
EF92  D0 EA             BNE L896        ;BAD, TRY AGAIN
EF94  85 A9             STA RINONE      ;GOOD, DISABLE FLAG
EF96  60                RTS             ;EXIT
EF97              ;
EF97              ;PUT DATA IN BUFFER (AT PARITY TIME)
EF97              ;
EF97  AC 9B 02    L894  LDY RIDBE        ;GET END
EF9A  C8                INY
EF9B  CC 9C 02          CPY RIDBS       ;PASSED START?
EF9E  F0 2A             BEQ L905        ;YES, ERROR
EFA0  8C 9B 02          STY RIDBE       ;MOVE RIDBE FORWARD
EFA3  88                DEY
EFA4  A5 AA             LDA RIDATA      ;GET BYTE BUFFER UP
EFA6  AE 98 02          LDX BITNUM      ;SHIFT UNTIL FULL BYTE
EFA9  E0 09       L895  CPX #$09         ;ALWAYS 8 BITS
EFAB  F0 04             BEQ L903
EFAD  4A                LSR A           ;FILL WITH 0'S
EFAE  E8                INX
EFAF  D0 F8             BNE L895
EFB1  91 F7       L903  STA (RIBUF),Y    ;DATA TO PAGE BUFFER
EFB3              ;
EFB3              ;PARITY CHECKING
EFB3              ;
EFB3  A9 20             LDA #$20        ;CHECK 6526 COMMAND REG
EFB5  2C 94 02          BIT M26CDR
EFB8  F0 B4             BEQ L898        ;NO PARITY BIT SO STOP BIT
EFBA  30 B1             BMI L1230       ;NO PARITY CHECK
EFBC              ;
EFBC              ;CHECK CALC PARITY
EFBC              ;
EFBC  A5 A7             LDA INBIT
EFBE  45 AB             EOR RIPRTY      ;PUT IN WITH PARITY
EFC0  F0 03             BEQ L902        ;EVEN PARITY
EFC2  70 A9             BVS L1230       ;ODD, OK SO EXIT
EFC4  2C                .BYT $2C        ;SKIP NEXT COMMAND
EFC5  50 A6       L902  BVC L1230        ;EVEN SO EXIT
EFC7              ;
EFC7              ;ERRORS REPORTED
EFC7              ;
EFC7  A9 01             LDA #$01        ;PARITY ERROR
EFC9  2C                .BYT $2C
```

```
LOC   CODE        LINE

EFCA  A9 04       L905   LDA #$04        ;RECEIVER OVERRUN
EFCC  2C                 .BYT $2C
EFCD  A9 80       L901   LDA #$80        ;BREAK DETECTED
EFCF  2C                 .BYT $2C
EFD0  A9 02       L907   LDA #$02        ;FRAME ERROR
EFD2  0D 97 02           ORA RSSTAT
EFD5  8D 97 02           STA RSSTAT
EFD8  4C 7E EF           JMP L896
EFDB                ;
EFDB                ;CHECK FOR ERRORS
EFDB                ;
EFDB  A5 AA       L906   LDA RIDATA      ;EXPECTING STOP
EFDD  D0 F1              BNE L907       ;FRAME ERROR
EFDF  F0 EC              BEQ L901       ;COULD BE A BREAK
EFE1                ;
EFE1                ;OUTPUT A FILE USING RS-232
EFE1                ;
EFE1  85 9A       L897   STA DFLTO       ;SET DEFAUT OUT
EFE3  AD 94 02           LDA M26CDR     ;CHECK FOR 3/X LINE
EFE6  4A                 LSR A
EFE7  90 29              BCC L909       ;3LINE, NO TURN AROUND
EFE9                ;
EFE9                ;TURN AROUND LOGIC
EFE9                ;
EFE9  A9 02              LDA #$02       ;BIT RTS IS ON
EFEB  2C 01 DD           BIT D2DPB
EFEE  10 1D              BPL L912       ;NO DSR, ERROR
EFF0  D0 20              BNE L909       ;RTS, OUTPUTTING OR FULL DUPLEX
EFF2                ;
EFF2                ;CHECK FOR ACTIVE INPUT
EFF2                ;
EFF2  AD A1 02    L978   LDA $02A1
EFF5  29 02              AND #$02
EFF7  D0 F9              BNE L978       ;HANG UNTIL INPUT DONE
EFF9                ;
EFF9                ;WAIT FOR CTS TO BE OFF
EFF9                ;
EFF9  2C 01 DD    L910   BIT D2DPB
EFFC  70 FB              BVS L910
EFFE                ;
EFFE                ;TURN ON RTS
EFFE                ;
EFFE  AD 01 DD           LDA D2DPB
F001  09 02              ORA #$02
F003  8D 01 DD           STA D2DPB
F006                ;
F006                ;WAIT FOR CTS TO GO ON
F006                ;
F006  2C 01 DD    L911   BIT D2DPB
F009  70 07              BVS L909       ;DONE
F00B  30 F9              BMI L911       ;STILL DSR
F00D  A9 40       L912   LDA #$40        ;DSR ERROR
F00F  8D 97 02           STA RSSTAT
F012  18          L909   CLC
F013  60                 RTS
F014                ;
F014                ;OUTPUT A CHAR RS-232, BUFFER HANDLER
F014                ;
F014  20 28 F0    L908   JSR L955
F017  AC 9E 02    L914   LDY RODBE
F01A  C8                 INY
```

```
LOC   CODE        LINE

F01B  CC 9D 02            CPY RODBS       ;BUFFER FULL?
F01E  F0 F4               BEQ L908        ;YES, HANG
F020  8C 9E 02            STY RODBE       ;NEW START
F023  88                  DEY
F024  A5 9E               LDA T1
F026  91 F9               STA (ROBUF),Y   ;STORE DATA
F028                ;
F028                ;SET UP IF NECESSARY TO OUTPUT
F028                ;
F028  AD A1 02      L955  LDA $02A1        ;T1 NMI ENABLE?
F02B  4A                  LSR A
F02C  B0 1E               BCS L913        ;YES, EXIT
F02E  A9 10               LDA #$10        ;NO, SET UP TIMER A
F030  8D 0E DD            STA D2CRA
F033  AD 99 02            LDA BAUDOF
F036  8D 04 DD            STA D2TAL
F039  AD 9A 02            LDA BAUDOF+1
F03C  8D 05 DD            STA D2TAH
F03F  A9 81               LDA #$81        ;SET UP T1 NMI'S
F041  20 3B EF            JSR L891
F044  20 06 EF            JSR L879
F047  A9 11               LDA #$11
F049  8D 0E DD            STA D2CRA
F04C  60            L913  RTS             ;BEGIN TRANSFER
F04D                      .END
F04D                      .LIB K7
F04D                ;
F04D                ;INPUT A FILE OVER RS-232
F04D                ;
F04D  85 99         L915  STA DFLTN        ;SET DEFAULT INPUT
F04F  AD 94 02            LDA M26CDR      ;CHECK FOR 3/X LINE
F052  4A                  LSR A
F053  90 28               BCC L920        ;3LINE, NO HANDSHAKE
F055  29 08               AND #$08        ;FULL/HALF CHECK
F057  F0 24               BEQ L920        ;FULL, NO HANDSHAKE
F059                ;
F059                ;TURN AROUND LOGIC
F059                ;
F059  A9 02               LDA #$02        ;BIT RTS IS ON
F05B  2C 01 DD            BIT D2DPB
F05E  10 AD               BPL L912        ;NO, DSR ERROR
F060  F0 22               BEQ L916        ;RTS LO, IN CORRECT MODE
F062                ;
F062                ;WAIT FOR ACTIVE OUTPUT TO BE DONE
F062                ;
F062  AD A1 02      L964  LDA $02A1
F065  4A                  LSR A
F066  B0 FA               BCS L964
F068                ;
F068                ;TURN OFF RTS
F068                ;
F068  AD 01 DD            LDA D2DPB
F06B  29 FD               AND #$FD
F06D  8D 01 DD            STA D2DPB
F070                ;
F070                ;WAIT FOR DCD TO GO HI
F070                ;
F070  AD 01 DD      L918  LDA D2DPB
F073  29 04               AND #$04
F075  F0 F9               BEQ L918
F077                ;
```

```
LOC   CODE        LINE

F077              ;ENABLE CB1 FOR RS-232 INPUT
F077              ;
F077  A9 90       L919   LDA #$90
F079  18                 CLC              ;NO ERROR
F07A  4C 3B EF           JMP L891
F07D              ;
F07D              ;IF NOT 3 LINE HALF, SEE IF WE
F07D              ; NEED TO TURN ON CB1
F07D              ;
F07D  AD A1 02    L920   LDA $02A1         ;CB1 OR T2 ACTIVE?
F080  29 12              AND #$12
F082  F0 F3              BEQ L919         ;NO NEED TO TURN ON
F084  18          L916   CLC               ;NO ERROR
F085  60                 RTS
F086              ;
F086              ;INPUT A CHAR RS-232, BUFFER HANDLER
F086              ;
F086  AD 97 02    L917   LDA RSSTAT
F089  AC 9C 02           LDY RIDBS        ;GET LAST BYTE ADDRESS
F08C  CC 9B 02           CPY RIDBE        ;BUFFER EMPTY?
F08F  F0 0B              BEQ L929         ;RETURN IF NO CHAR
F091  29 F7              AND #$F7
F093  8D 97 02           STA RSSTAT
F096  B1 F7              LDA (RIBUF),Y    ;GET LAST CHAR
F098  EE 9C 02           INC RIDBS        ;NEXT POSITION
F09B              ;
F09B              ;RECEIVER ALWAYS RUNS
F09B              ;
F09B  60                 RTS
F09C  09 08       L929   ORA #$08
F09E  8D 97 02           STA RSSTAT
F0A1  A9 00              LDA #$00         ;RETURN A NULL
F0A3  60                 RTS
F0A4              ;
F0A4              ;PROTECT SERIAL/CASSETTE FROM
F0A4              ; RS-232 NMI'S
F0A4              ;
F0A4  48          L921   PHA               ;SAVE .A
F0A5  AD A1 02           LDA $02A1        ;RS-232 ENABLES?
F0A8  F0 11              BEQ L923         ;NO
F0AA  AD A1 02    L838   LDA $02A1         ;WAIT UNTIL DONE
F0AD  29 03              AND #$03         ;WITH T1 & T2
F0AF  D0 F9              BNE L838
F0B1  A9 10              LDA #$10         ;DISABLE CB1
F0B3  8D 0D DD           STA D2ICR
F0B6  A9 00              LDA #$00
F0B8  8D A1 02           STA $02A1
F0BB  68          L923   PLA               ;ALL DONE
F0BC  60                 RTS
F0BD                     .END
F0BD                     .LIB K8
F0BD  0D          MS1    .BYT $D,'I/O ERROR ',$A3
F0BE  49 2F
F0C8  A3
F0C9  0D          MS5    .BYT $D,'SEARCHING',$A0
F0CA  53 45
F0D3  A0
F0D4  46 4F 52    MS6    .BYT 'FOR',$A0
F0D7  A0
F0D8  0D          MS7    .BYT $D,'PRESS PLAY ON TAP',$C5
F0D9  50 52
```

```
LOC   CODE        LINE


F0EA  C5
F0EB  50 52       MS8     .BYT 'PRESS RECORD & PLAY ON TAP',$C5
F105  C5
F106  0D          MS10    .BYT $D,'LOADIN',$C7
F107  4C 4F
F10D  C7
F10E  0D          MS11    .BYT $D,'SAVING',$A0
F10F  53 41
F115  A0
F116  0D          MS21    .BYT $D,'VERIFYIN',$C7
F117  56 45
F11F  C7
F120  0D          MS17    .BYT $D,'FOUND',$A0
F121  46 4F
F126  A0
F127  0D          MS18    .BYT $D,'OK',$8D
F128  4F 4B
F12A  8D
F12B              ;
F12B              ;PRINT MESSAGE TO SCREEN ONLY IF
F12B              ; OUTPUT ENABLED
F12B              ;
F12B  24 9D       L922    BIT MSGFLG      ;PRINTING MESSAGES?
F12D  10 0D             BPL L925        ;NO
F12F  B9 BD F0    L1073   LDA MS1,Y
F132  08                PHP
F133  29 7F             AND #$7F
F135  20 D2 FF          JSR L622
F138  C8                INY
F139  28                PLP
F13A  10 F3             BPL L1073
F13C  18          L925    CLC
F13D  60                RTS
F13E                    .END
F13E                    .LIB K9
F13E              ;
F13E              ;***************************************
F13E              ;* GETIN -- GET CHARACTER FROM CHANNEL.
F13E              ;*       CHANNEL IS DETERMINED BY DFLTN.
F13E              ;* IF DEVICE IS 0, KEYBOARD QUEUE IS
F13E              ;* EXAMINED AND A CHARACTER REMOVED IF
F13E              ;* AVAILABLE.  IF QUEUE IS EMPTY, Z
F13E              ;* FLAG IS RETURNED SET.  DEVICES 1-31
F13E              ;* ADVANCE TO BASIN. THE CHARACTER IS
F13E              ;* RETURNED IN .A. IF ZERO, NULL CHAR.
F13E              ;***************************************
F13E              ;
F13E  A5 99       NGETIN LDA DFLTN        ;CHECK DEVICE
F140  D0 08             BNE L924        ;NOT KEYBOARD
F142  A5 C6             LDA NDX         ;QUEUE INDEX
F144  F0 0F             BEQ L944        ;NOTHING THERE, EXIT
F146  78                SEI
F147  4C B4 E5          JMP L690        ;REMOVE A CHAR
F14A  C9 02       L924    CMP #$02        ;RS-232?
F14C  D0 18             BNE L927        ;NO, USE BASIN
F14E  84 97       L926    STY XSAV        ;SAVE .Y, USED IN RS-232
F150  20 86 F0          JSR L917
F153  A4 97             LDY XSAV        ;RESTORE .Y
F155  18          L944    CLC             ;GOOD RETURN
F156  60                RTS
F157              ;
```

```
LOC   CODE        LINE

F157              ;***************************************
F157              ;* BASIN-- INPUT CHARACTER FROM CHANNEL.
F157              ;*      BASIN DIFFERS FROM GETIN ON KEYBOARD
F157              ;* AND RS-232 ONLY. THE SCREEN EDITOR
F157              ;* MAKES READY AN ENTIRE LINE WHICH IS
F157              ;* PASSED CHARACTER BY CHARACTER UP
F157              ;* TO THE CARRIAGE RETURN. THE CHARACTER
F157              ;* IS RETURNED IN .A. ZERO FOR NULL CHAR
F157              ;* OTHER DEVICES ARE:
F157              ;*       0 --- KEYBOARD
F157              ;*       1 --- CASSETTE
F157              ;*       2 --- RS-232
F157              ;*       3 --- SCREEN
F157              ;*    4-31 --- SERIAL BUS
F157              ;***************************************
F157              ;
F157  A5 99       NBASIN LDA DFLTN       ;CHECK DEVICE
F159  D0 0B              BNE L927        ;NOT KEYBOARD
F15B              ;
F15B              ;INPUT FROM KEYBOARD
F15B              ;
F15B  A5 D3              LDA PNTR        ;SAVE CURRENT:
F15D  85 CA              STA LSTP        ;CURSOR COLUMN,
F15F  A5 D6              LDA TBLX
F161  85 C9              STA LSXP        ;LINE NUMBER
F163  4C 32 E6           JMP L700        ;BLINK CURSOR UNTIL RETURN
F166              ;
F166  C9 03       L927   CMP #$03         ;SCREEN?
F168  D0 09              BNE L928        ;NO
F16A  85 D0              STA CRSW        ;FAKE CARRIAGE RETURN
F16C  A5 D5              LDA LNMX        ;ENDED:
F16E  85 C8              STA INDX        ;ON THIS LINE
F170  4C 32 E6           JMP L700        ;PICK UP CHARACTERS
F173              ;
F173  B0 38       L928   BCS L939         ;DEVICES>3
F175  C9 02              CMP #$02        ;RS-232?
F177  F0 3F              BEQ L942        ;YES
F179              ;
F179              ;INPUT FROM CASSETTE BUFFER
F179              ;
F179  86 97              STX XSAV
F17B  20 99 F1           JSR L935
F17E  B0 16              BCS L936        ;STOP KEY, ERROR
F180  48                 PHA
F181  20 99 F1           JSR L935
F184  B0 0D              BCS L937        ;STOP KEY, ERROR
F186  D0 05              BNE L931        ;NOT EOF
F188  A9 40              LDA #$40        ;TELL USER EOF
F18A  20 1C FE           JSR L1217       ;IN STATUS
F18D  C6 A6       L931   DEC BUFPT
F18F  A6 97              LDX XSAV        ;.X RESERVED
F191  68                 PLA             ;CHARACTER RETURNED
F192  60                 RTS             ;ALL DONE
F193  AA          L937   TAX              ;SAVE ERROR INFO
F194  68                 PLA             ;TOSS DATA
F195  8A                 TXA             ;RESTORE ERROR
F196  A6 97       L936   LDX XSAV         ;RETURN
F198  60                 RTS             ;ERROR RETURN
F199              ;
F199              ;GET A CHARACTER FROM CASSETTE BUFFER
F199              ;
```

```
LOC   CODE        LINE

F199  20 0D F8    L935   JSR L1110       ;BUFFER POINTER WRAP?
F19C  D0 0B              BNE L934       ;NO
F19E  20 41 F8           JSR L1029      ;YES, READ NEXT BLOCK
F1A1  B0 11              BCS L945       ;STOP KEY, ERROR
F1A3  A9 00              LDA #$00       ;POINT TO BEGINNING
F1A5  85 A6              STA BUFPT      ;OF BUFFER
F1A7  F0 F0              BEQ L935       ;ALWAYS
F1A9  B1 B2       L934   LDA (TAPE1),Y   ;GET CHAR FROM BUFFER
F1AB  18                 CLC            ;GOOD RETURN
F1AC  60                 RTS
F1AD              ;
F1AD              ;INPUT FROM SERIAL BUS
F1AD              ;
F1AD  A5 90       L939   LDA STATUS      ;STATUS FROM LAST
F1AF  F0 04              BEQ L941       ;O.K.
F1B1  A9 0D       L932   LDA #$0D        ;BAD, ALL DONE
F1B3  18          L946   CLC             ;VALID DATA
F1B4  60          L945   RTS
F1B5              ;
F1B5  4C 13 EE    L941   JMP L865        ;GOOD, HANDSHAKE
F1B8              ;
F1B8              ;INPUT FROM RS-232
F1B8              ;
F1B8  20 4E F1    L942   JSR L926        ;GET INFO
F1BB  B0 F7              BCS L945       ;ERROR RETURN
F1BD  C9 00              CMP #$00
F1BF  D0 F2              BNE L946       ;WAIT FOR VALID DATA
F1C1  AD 97 02           LDA RSSTAT
F1C4  29 60              AND #$60
F1C6  D0 E9              BNE L932       ;GOOD RETURN
F1C8  F0 EE              BEQ L942       ;ALWAYS
F1CA              ;
F1CA              ;**************************************
F1CA              ;* BSOUT -- OUTPUT CHAR STORED IN .A TO
F1CA              ;* CHANNEL DETERMINED BY VARIABLLE DFLTO:
F1CA              ;*      0 --- INVALID
F1CA              ;*      1 --- CASSETTE
F1CA              ;*      2 --- RS-232
F1CA              ;*      3 --- SCREEN
F1CA              ;*  4-31 --- SERIAL BUS
F1CA              ;**************************************
F1CA              ;
F1CA  48          NBSOUT PHA             ;PRESERVE .A
F1CB  A5 9A              LDA DFLTO      ;CHECK DEVICE
F1CD  C9 03              CMP #$03       ;SCREEN?
F1CF  D0 04              BNE L933       ;NO
F1D1  68                 PLA            ;YES, RESTORE .A
F1D2  4C 16 E7           JMP L728       ;PRINT TO SCREEN
F1D5              ;
F1D5  90 04       L933   BCC L947        ;DEVICE 1 OR 2
F1D7              ;
F1D7              ;PRINT TO SERIAL BUS
F1D7              ;
F1D7  68                 PLA
F1D8  4C DD ED           JMP L861
F1DB              ;
F1DB              ;PRINT TO CASSETTE DEVICES
F1DB              ;
F1DB  4A          L947   LSR A
F1DC  68                 PLA
F1DD  85 9E       L948   STA PTR1
```

```
LOC   CODE        LINE

F1DF              ;
F1DF              ;PRESERVE REGISTERS
F1DF              ;
F1DF  8A                TXA
F1E0  48                PHA
F1E1  98                TYA
F1E2  48                PHA
F1E3  90 23             BCC L954        ;RS-232
F1E5  20 0D F8          JSR L1110       ;CHECK BUFFER POINTER
F1E8  D0 0E             BNE L996        ;NOT END
F1EA  20 64 F8          JSR L1069       ;WRITE FULL BUFFER
F1ED  B0 0E             BCS L956        ;ABORT ON STOP KEY
F1EF              ;
F1EF              ;PUT BUFFER TYPE BYTE
F1EF              ;
F1EF  A9 02             LDA #$02
F1F1  A0 00             LDY #$00
F1F3  91 B2             STA (TAPE1),Y
F1F5              ;
F1F5              ;RESET BUFFER POINTER
F1F5              ;
F1F5  C8                INY             ;.Y=1
F1F6  84 A6             STY BUFPT       ;BUFPT=1
F1F8  A5 9E       L996  LDA PTR1
F1FA  91 B2             STA (TAPE1),Y   ;DATA TO BUFFER
F1FC              ;
F1FC              ;RESTORE REGISTERS
F1FC              ;
F1FC  18          L951  CLC              ;GOOD RETURN
F1FD  68          L956  PLA
F1FE  A8                TAY
F1FF  68                PLA
F200  AA                TAX
F201  A5 9E             LDA PTR1        ;RESTORE .A
F203  90 02             BCC L953        ;NO ERROR
F205  A9 00             LDA #$00        ;STOP ERROR
F207  60          L953  RTS
F208              ;
F208              ;OUTPUT TO RS-232
F208              ;
F208  20 17 F0    L954  JSR L914         ;OUTPUT
F20B  4C FC F1          JMP L951        ;GOOD RETURN
F20E                    .END
F20E                    .LIB K10
F20E              ;
F20E              ;***************************************
F20E              ;* CHKIN -- OPEN CHANNEL FOR INPUT.
F20E              ;*      THE NUMBER OF THE LOGICAL FILE TO
F20E              ;* BE OPENED FOR INPUT IS PASSED IN .X.
F20E              ;* CHKIN SEARCHES THE LOGICAL FILE TO
F20E              ;* LOOK UP DEVICE AND COMMAND INFO.
F20E              ;* ERRORS ARE REPORTED IF THE DEVICE WAS
F20E              ;* NOT OPENED FOR INPUT, (E.G. CASSETTE
F20E              ;* WRITE FILE), OR THE LOGICAL FILE HAS
F20E              ;* NO REFERENCE IN THE TABLES. DEVICE 0,
F20E              ;* (KEYBOARD), AND DEVICE 3 (SCREEN),
F20E              ;* REQUIRE NO TABLE ENTRIES AND ARE
F20E              ;* HANDLED SEPARATELY.
F20E              ;***************************************
F20E              ;
F20E  20 0F F3    NCHKIN JSR L1000        ;FILE OPENED?
```

```
LOC   CODE        LINE

F211  F0 03               BEQ L950        ;YES
F213  4C 01 F7            JMP L1009       ;NO, FILE NOT OPEN
F216  20 1F F3    L950    JSR L1002        ;GET FILE INFO
F219  A5 BA               LDA FA
F21B  F0 16               BEQ L963        ;KEYBOARD
F21D              ;
F21D              ;COULD BE SCREEN, RS-232, OR SERIAL
F21D              ;
F21D  C9 03               CMP #$03        ;SCREEN?
F21F  F0 12               BEQ L963        ;YES, DONE
F221  B0 14               BCS L961        ;SERIAL
F223  C9 02               CMP #$02        ;RS-232?
F225  D0 03               BNE L958        ;NO, MUST BE TAPE
F227  4C 4D F0            JMP L915        ;RS-232
F22A              ;
F22A              ;CHECK FOR INPUT FILE ON TAPE
F22A              ;
F22A  A6 B9       L958    LDX SA           ;CHECK SECONDARY AD
F22C  E0 60               CPX #$60        ;INPUT?
F22E  F0 03               BEQ L963        ;YES
F230  4C 0A F7            JMP L971        ;NO, NOT INPUT FILE
F233  85 99       L963    STA DFLTN        ;SET INPUT
F235  18                  CLC :GOOD RETURN
F236  60                  RTS
F237              ;
F237              ;A SERIAL DEVICE MUST TALK
F237              ;
F237  AA          L961    TAX              ;SAVE DEVICE #
F238  20 09 ED            JSR L836        ;TALK
F23B  A5 B9               LDA SA          ;SECOND?
F23D  10 06               BPL L962        ;YES, SEND IT
F23F  20 CC ED            JSR L970        ;NO, LET GO
F242  4C 48 F2            JMP L967
F245  20 C7 ED    L962    JSR L860         ;SEND TALK SA
F248  8A          L967    TXA
F249  24 90               BIT STATUS      ;DID IT LISTEN?
F24B  10 E6               BPL L963        ;YES
F24D  4C 07 F7            JMP L1026       ;DEVICE NOT PRESENT
F250              ;
F250              ;****************************************
F250              ;* CHKOUT -- OPEN CHANNEL FOR OUTPUT.
F250              ;*      THE NUMBER OF THE LOGICAL FILE TO
F250              ;* BE OPENED FOR OUTPUT IS PASSED IN .X.
F250              ;* CHKOUT SEARCHES THE LOGICAL FILE TO
F250              ;* LOOK UP DEVICE AND COMMAND INFO.
F250              ;* ERRORS ARE REPORTED IF THE DEVICE WAS
F250              ;* NOT OPENED FOR INPUT, (E.G. KEYBOARD)
F250              ;* OR THE LOGICAL FILE HAS NO REFERENCE
F250              ;* IN THE TABLES. DEVICE 3 (SCREEN)
F250              ;* REQUIRES NO TABLE ENTRY AND IS
F250              ;* HANDLED SEPARATELY.
F250              ;****************************************
F250              ;
F250  20 0F F3    NCKOUT JSR L1000         ;FILE IN TABLE;
F253  F0 03               BEQ L969        ;YES
F255  4C 01 F7            JMP L1009       ;NO, FILE NOT OPEN
F258  20 1F F3    L969    JSR L1002        ;GET TABLE INFO
F25B  A5 BA               LDA FA
F25D  D0 03               BNE L979        ;NOT KEYBOARD
F25F  4C 0D F7    L972    JMP L965         ;KEYBOARD, NOT OUTPUT FILE
F262              ;
```

```
LOC   CODE        LINE

F262              ;COULD BE SCREEN, SERIAL,
F262              ; CASSETTE, OR RS-232
F262              ;
F262  C9 03       L979   CMP #$03         ;SCREEN?
F264  F0 0F              BEQ L977         ;YES, DONE
F266  B0 11              BCS L975         ;NO, SERIAL
F268  C9 02              CMP #$02         ;RS-232?
F26A  D0 03              BNE L973         ;NO, MUST BE CASSETTE
F26C  4C E1 EF           JMP L897         ;SET UP FOR RS-232
F26F              ;
F26F              ;CHECK FOR CASSETTE FILE TYPE
F26F              ;
F26F  A6 B9       L973   LDX SA
F271  E0 60              CPX #$60         ;INPUT FILE?
F273  F0 EA              BEQ L972         ;YES, ERROR
F275  85 9A       L977   STA DFLTO         ;SET OUTPUT
F277  18                 CLC              ;GOOD RETURN
F278  60                 RTS
F279              ;
F279              ;SERIAL DEVICES
F279              ;
F279  AA          L975   TAX               ;SAVE DEVICE #
F27A  20 0C ED           JSR L966         ;LISTEN
F27D  A5 B9              LDA SA           ;AND SECOND?
F27F  10 05              BPL L976         ;YES
F281  20 BE ED           JSR L983         ;NO, RELEASE LINES
F284  D0 03              BNE L981         ;ALWAYS
F286  20 B9 ED    L976   JSR L871         ;SEND LISTEN SA
F289  8A          L981   TXA
F28A  24 90              BIT STATUS       ;DID IT LISTEN?
F28C  10 E7              BPL L977         ;YES, FINISH
F28E  4C 07 F7           JMP L1026        ;NO, DEVICE NOT PRESENT
F291                     .END
F291                     .LIB K11
F291              ;
F291              ;***************************************
F291              ;* CLOSE -- CLOSE LOGICAL FILE.
F291              ;*      THE LOGICAL FILE NUMBER OF THE
F291              ;* FILE TO BE CLOSED IS PASSED IN .A.
F291              ;* KEYBOARD, SCREEN, AND FILES NOT OPEN
F291              ;* PASS STRAIGHT THROUGH. TAPE FILES
F291              ;* OPEN FOR WRITE ARE CLOSED BY DUMPING
F291              ;* THE LAST BUFFER AND CONDITIONALLY
F291              ;* WRITING AN END OF TAPE BLOCK. SERIAL
F291              ;* FILES ARE CLOSED BY SENDING A CLOSE
F291              ;* FILE COMMAND IF A SECONDARY ADDRESS
F291              ;* WAS SPECIFIED IN ITS OPEN COMMAND.
F291              ;***************************************
F291              ;
F291  20 14 F3    NCLOSE JSR L957          ;LOOK UP FILE
F294  F0 02              BEQ L982         ;FOUND
F296  18                 CLC              ;ELSE RETURN
F297  60                 RTS
F298  20 1F F3    L982   JSR L1002         ;GET FILE DATA
F29B  8A                 TXA              ;SAVE TABLE INDEX
F29C  48                 PHA
F29D  A5 BA              LDA FA           ;CHECK DEVICE #
F29F  F0 50              BEQ L987         ;KEYBOARD, DONE
F2A1  C9 03              CMP #$03         ;SCREEN?
F2A3  F0 4C              BEQ L987         ;YES, DONE
F2A5  B0 47              BCS L997         ;SERIAL
```

```
LOC   CODE        LINE

F2A7  C9 02               CMP #$02        ;RS-232?
F2A9  D0 1D               BNE L993        ;NO, MUST BE TAPE
F2AB                ;
F2AB                ;REMOVE RS-232 FILE
F2AB                ;
F2AB  68                  PLA
F2AC  20 F2 F2            JSR L986
F2AF                ;
F2AF                ;REMOVE RS-232 NMI'S
F2AF                ; SEND MARK & ALL LINES HI
F2AF                ;
F2AF  20 83 F4            JSR L994
F2B2                ;
F2B2                ;DE-ALLOCATE BUFFERS
F2B2                ;
F2B2  20 27 FE            JSR L1235       ;GET MEMSIZ
F2B5  A5 F8               LDA RIBUF+1     ;CHECK INPUT ALLOCATION
F2B7  F0 01               BEQ L985        ;ALLOCATED
F2B9  C8                  INY
F2BA  A5 FA         L985  LDA ROBUF+1      ;OUTPUT ALLOCATION?
F2BC  F0 01               BEQ L992
F2BE  C8                  INY
F2BF  A9 00         L992  LDA #$00         ;DE-ALLOCATE
F2C1  85 F8               STA RIBUF+1
F2C3  85 FA               STA ROBUF+1
F2C5  4C 7D F4            JMP L1043       ;GO SET NEW TOP
F2C8                ;
F2C8                ;CLOSE CASSETTE FILE
F2C8                ;
F2C8  A5 B9         L993  LDA SA           ;WAS IT READ?
F2CA  29 0F               AND #$0F
F2CC  F0 23               BEQ L987        ;YES
F2CE  20 D0 F7            JSR L1104       ;NO, IT WAS WRITE
F2D1  A9 00               LDA #$00        ;EOF CHAR
F2D3  38                  SEC
F2D4  20 DD F1            JSR L948        ;SEND EOF
F2D7  20 64 F8            JSR L1069       ;CLOSE FILE PATCH
F2DA  90 04               BCC L988        ;NO STOP KEY
F2DC  68                  PLA
F2DD  A9 00               LDA #$00
F2DF  60                  RTS
F2E0  A5 B9         L988  LDA SA
F2E2  C9 62               CMP #$62        ;WRITE END OF TAPE BLOCK?
F2E4  D0 0B               BNE L987        ;NO
F2E6  A9 05               LDA #EOT
F2E8  20 6A F7            JSR L1099       ;YES
F2EB  4C F1 F2            JMP L987
F2EE                ;
F2EE                ;CLOSE A SERIAL FILE
F2EE                ;
F2EE  20 42 F6      L997  JSR L1081
F2F1                ;
F2F1                ;REMOVE FILE ENTRIES FROM TABLES
F2F1                ;
F2F1  68            L987  PLA              ;GET TABLE INDEX
F2F2  AA            L986  TAX
F2F3  C6 98               DEC LDTND
F2F5  E4 98               CPX LDTND       ;IS IT AT END?
F2F7  F0 14               BEQ L989        ;YES, DONE
F2F9  A4 98               LDY LDTND       ;NO, SHIFT LAST ENTRY
F2FB  B9 59 02            LDA LAT,Y       ;INTO DELETED ENTRIES'
```

```
LOC   CODE        LINE

F2FE  9D 59 02          STA LAT,X       ;POSITION
F301  B9 63 02          LDA FAT,Y
F304  9D 63 02          STA FAT,X
F307  B9 6D 02          LDA SAT,Y
F30A  9D 6D 02          STA SAT,X
F30D  18          L989  CLC              ;GOOD EXIT
F30E  60                RTS
F30F              ;
F30F              ;FIND FILE ENTRY
F30F              ;
F30F  A9 00       L1000 LDA #$00
F311  85 90             STA STATUS
F313  8A                TXA
F314  A6 98       L957  LDX LDTND
F316  CA          L984  DEX
F317  30 15             BMI L960
F319  DD 59 02          CMP LAT,X
F31C  D0 F8             BNE L984
F31E  60                RTS
F31F              ;
F31F              ;FETCH TABLE ENTRIES
F31F              ;
F31F  BD 59 02    L1002 LDA LAT,X
F322  85 B8             STA LA
F324  BD 63 02          LDA FAT,X
F327  85 BA             STA FA
F329  BD 6D 02          LDA SAT,X
F32C  85 B9             STA SA
F32E  60          L960  RTS
F32F                    .END
F32F                    .LIB K12
F32F              ;
F32F              ;***************************************
F32F              ;* CLALL -- CLOSE ALL LOGICAL FILES.
F32F              ;*       DELETES ALL TABLE ENTRIES AND
F32F              ;* RESTORES DEFAULT I/O  CHANNELS AND
F32F              ;* CLEARS SERIAL PORT DEVICES
F32F              ;***************************************
F32F              ;
F32F  A9 00       NCLALL LDA #$00
F331  85 98             STA LDTND       ;FORGET ALL FILES
F333              ;
F333              ;***************************************
F333              ;* CLRCH -- CLEAR CHANNELS.
F333              ;*       UNLISTEN OR UNTALK SERIAL
F333              ;* DEVICES, BUT LEAVE OTHERS ALONE.
F333              ;* DEFAULT CHANNELS ARE RESTORED.
F333              ;***************************************
F333              ;
F333  A2 03       NCLRCH LDX #$03
F335  E4 9A             CPX DFLTO       ;OUTPUT CHANNEL SERIAL?
F337  B0 03             BCS L1001       ;NO
F339  20 FE ED          JSR L1006       ;YES, UNLISTEN
F33C  E4 99       L1001 CPX DFLTN        ;INPUT CHANNEL SERIAL?
F33E  B0 03             BCS L1003       ;NO
F340  20 EF ED          JSR L863        ;YES, UNTALK
F343  86 9A       L1003 STX DFLTO        ;OUTPUT CHANNEL=3
F345  A9 00             LDA #$00
F347  85 99             STA DFLTN       ;INPUT CHANNEL=0
F349  60                RTS
F34A                    .END
```

```
LOC   CODE        LINE

F34A                      .LIB K13
F34A                ;
F34A                ;**************************************
F34A                ;* OPEN  -- OPEN A FILE.
F34A                ;*     CREATES AN ENTRY IN THE FILE
F34A                ;* FILE TABLES CONSISTING OF LOGICAL
F34A                ;* FILE NUMBER, DEVICE NUMBER, AND SEC
F34A                ;* ADDRESS NUMBER.
F34A                ;*     ROUTINES SETLFS & SETNAM SHOULD
F34A                ;* BE USED FIRST.
F34A                ;**************************************
F34A                ;
F34A  A6 B8         NOPEN  LDX LA          ;CHECK FILE #
F34C  D0 03                BNE L1005       ;NOT KEYBOARD
F34E  4C 0A F7             JMP L971        ;NOT INPUT FILE
F351  20 0F F3      L1005  JSR L1000       ;ALREADY OPEN?
F354  D0 03                BNE L1007       ;NO
F356  4C FE F6             JMP L1011       ;YES, FILE OPEN
F359  A6 98         L1007  LDX LDTND       ;END OF TABLE?
F35B  E0 0A                CPX #$0A
F35D  90 03                BCC L1008       ;NO
F35F  4C FB F6             JMP L1097       ;YES, TOO MANY FILES
F362  E6 98         L1008  INC LDTND       ;NEW FILE
F364  A5 B8                LDA LA
F366  9D 59 02             STA LAT,X       ;STORE FILE #
F369  A5 B9                LDA SA
F36B  09 60                ORA #$60        ;MAKE SA SERIAL
F36D  85 B9                STA SA
F36F  9D 6D 02             STA SAT,X       ;STORE SA
F372  A5 BA                LDA FA
F374  9D 63 02             STA FAT,X       ;STORE DEVICE #
F377                ;
F377                ;PERFORM DEVICE SPECIFIC OPEN TASKS
F377                ;
F377  F0 5A                BEQ L1030       ;KEYBOARD, DONE
F379  C9 03                CMP #$03        ;SCREEN?
F37B  F0 56                BEQ L1030       ;YES, DONE
F37D  90 05                BCC L1010       ;CASSETTE OR RS-232
F37F  20 D5 F3             JSR L1021       ;OPEN SERIAL
F382  90 4F                BCC L1030       ;ALWAYS, DONE
F384                ;
F384                ;TAPE/RS-232 OPEN
F384                ;
F384  C9 02         L1010  CMP #$02        ;RS-232?
F386  D0 03                BNE L1013       ;NO, MUST BE TAPE
F388  4C 09 F4             JMP L1032       ;DO RS-232
F38B  20 D0 F7      L1013  JSR L1104       ;SEE IF TAPE BUFFER
F38E  B0 03                BCS L1015       ;YES
F390  4C 13 F7             JMP L1049       ;NO, TAPE NOT ALLOCATED
F393  A5 B9         L1015  LDA SA
F395  29 0F                AND #$0F        ;MASK OFF COMMAND
F397  D0 1F                BNE L1023       ;TAPE WRITE
F399                ;
F399                ;OPEN TAPE FILE TO READ
F399                ;
F399  20 17 F8             JSR L938        ;'PRESS PLAY'
F39C  B0 36                BCS L1012       ;STOP KEY PRESSED
F39E  20 AF F5             JSR L1062       ;'SEARCHING'
F3A1  A5 B7                LDA FNLEN       ;FILE NAME = ''
F3A3  F0 0A                BEQ L1028       ;ANY FILE
F3A5  20 EA F7             JSR L1108       ;NAMED FILE
```

```
LOC   CODE        LINE

F3A8  90 18             BCC L1019       ;FOUND IT
F3AA  F0 28             BEQ L1012       ;STOP KEY PRESSED
F3AC  4C 04 F7    L1017 JMP L959         ;FILE NOT FOUND
F3AF  20 2C F7    L1028 JSR L1098        ;GET ANY HEADER
F3B2  F0 20             BEQ L1012       ;STOP KEY PRESSED
F3B4  90 0C             BCC L1019       ;O.K.
F3B6  B0 F4             BCS L1017       ;FILE NOT FOUND
F3B8                    ;
F3B8                    ;OPEN TAPE FILE FOR WRITE
F3B8                    ;
F3B8  20 38 F8    L1023 JSR L1114        ;'PRESS RECORD & PLAY'
F3BB  B0 17             BCS L1012       ;STOP KEY PRESSED
F3BD  A9 04             LDA #HDFH       ;DATA FILE HEADER TYPE
F3BF  20 6A F7          JSR L1099       ;WRITE IT
F3C2                    ;
F3C2                    ;FINISH OPEN FOR TAPE READ/WRITE
F3C2                    ;
F3C2  A9 BF       L1019 LDA #BUFSZ-1    ;ASSUME FORCE READ
F3C4  A4 B9             LDY SA
F3C6  C0 60             CPY #$60        ;READ?
F3C8  F0 07             BEQ L1025       ;YES
F3CA                    ;
F3CA                    ;SET POINTERS FOR BUFFERING DATA
F3CA                    ;
F3CA  A0 00             LDY #$00
F3CC  A9 02             LDA #HDF        ;TYPE FLAG FOR BLOCK
F3CE  91 B2             STA (TAPE1),Y   ;TO BEGIN OF BUFFER
F3D0  98                TYA
F3D1  85 A6       L1025 STA BUFPT        ;POINT TO DATA
F3D3  18          L1030 CLC              ;FLAG GOOD OPEN
F3D4  60          L1012 RTS              ;EXIT
F3D5                    ;
F3D5                    ;OPEN SERIAL
F3D5                    ;
F3D5  A5 B9       L1021 LDA SA
F3D7  30 FA             BMI L1030       ;NO SA, DONE
F3D9  A4 B7             LDY FNLEN
F3DB  F0 F6             BEQ L1030       ;NO FILENAME, DONE
F3DD  A9 00             LDA #$00
F3DF  85 90             STA STATUS
F3E1  A5 BA             LDA FA
F3E3  20 0C ED          JSR L966        ;DEVICE LA TO LISTEN
F3E6  A5 B9             LDA SA
F3E8  09 F0             ORA #$F0
F3EA  20 B9 ED          JSR L871
F3ED  A5 90             LDA STATUS      ;DEVICE THERE?
F3EF  10 05             BPL L1014       ;YES
F3F1  68                PLA             ;NO
F3F2  68                PLA
F3F3  4C 07 F7          JMP L1026       ;DEVICE NOT PRESENT
F3F6  A5 B7       L1014 LDA FNLEN
F3F8  F0 0C             BEQ L1033       ;NO NAME, DONE
F3FA                    ;
F3FA                    ;SEND FILE NAME
F3FA                    ;
F3FA  A0 00             LDY #$00
F3FC  B1 BB       L1031 LDA (FNADR),Y
F3FE  20 DD ED          JSR L861
F401  C8                INY
F402  C4 B7             CPY FNLEN
F404  D0 F6             BNE L1031
```

```
LOC   CODE        LINE

F406  4C 54 F6    L1033  JMP L999         ;UNLISTEN AND RETURN
F409              ;
F409              ;OPEN AN RS-232 OR PARALLEL PORT FILE
F409              ;
F409  20 83 F4    L1032  JSR L994         ;SET UP FOR RS-232
F40C  8C 97 02           STY RSSTAT       ;CLEAR STATUS
F40F  C4 B7       L1016  CPY FNLEN        ;END OF NAME?
F411  F0 0A              BEQ L1036        ;YES
F413  B1 BB              LDA (FNADR),Y    ;MOVE DATA
F415  99 93 02           STA M26CTR,Y     ;TO M26 REGISTERS
F418  C8                 INY
F419  C0 04              CPY #$04         ;ONLY 4 POSSIBLE PARAMS
F41B  D0 F2              BNE L1016
F41D              ;
F41D              ;CALC # OF BITS
F41D              ;
F41D  20 4A EF    L1036  JSR L899
F420  8E 98 02           STX BITNUM
F423              ;
F423              ;CALC BAUD RATE
F423              ;
F423  AD 93 02           LDA M26CTR       ;GET INDEX TO BAUDO
F426  29 0F              AND #$0F
F428  F0 1C              BEQ L1040
F42A  0A                 ASL A            ;SET INDEX FOR 2BYTE TABLE
F42B  AA                 TAX
F42C  AD A6 02           LDA $02A6        ;DECIDE WHICH TABLE
F42F  D0 09              BNE L1035
F431  BC C1 FE           LDY $FEC1,X
F434  BD C0 FE           LDA $FEC0,X
F437  4C 40 F4           JMP L1039
F43A  BC EB E4    L1035  LDY $E4EB,X
F43D  BD EA E4           LDA $E4EA,X
F440  8C 96 02    L1039  STY M26AJB+1     ;STORE BAUD RATE
F443  8D 95 02           STA M26AJB
F446              ;
F446              ;CHECK FOR 3/X LINE RESPONSE
F446              ;
F446  AD 95 02    L1040  LDA M26AJB
F449  0A                 ASL A
F44A  20 2E FF           JSR L1226
F44D  AD 94 02           LDA M26CDR       ;BIT 0 OF M26CDR
F450  4A                 LSR A
F451  90 09              BCC L1038        ;IS 3LINE
F453              ;
F453              ;CHECK FOR XLINE PROPER STATES
F453              ;
F453  AD 01 DD           LDA D2DPB
F456  0A                 ASL A
F457  B0 03              BCS L1038
F459  20 0D F0           JSR L912         ;NO DATA SENT
F45C              ;
F45C              ;SET UP BUFFER POINTERS
F45C              ;
F45C  AD 9B 02    L1038  LDA RIDBE
F45F  8D 9C 02           STA RIDBS
F462  AD 9E 02           LDA RODBE
F465  8D 9D 02           STA RODBS
F468              ;
F468              ;ALLOCATE BUFFERS
F468              ;
```

```
LOC   CODE        LINE

F468  20 27 FE            JSR L1235       ;GET MEMSIZ
F46B  A5 F8               LDA RIBUF+1     ;IN ALLOCATION
F46D  D0 05               BNE L1042       ; ALREADY
F46F  88                  DEY             ;PROTECT 256 BYTES
F470  84 F8               STY RIBUF+1
F472  86 F7               STX RIBUF
F474  A5 FA         L1042 LDA ROBUF+1      ;OUT ALLOCATION
F476  D0 05               BNE L1043       ; ALREADY
F478  88                  DEY             ;PROTECT 256 BYTES
F479  84 FA               STY ROBUF+1
F47B  86 F9               STX ROBUF
F47D  38            L1043 SEC              ;SIGNAL TOP OF MEMORY CHANGE
F47E  A9 F0               LDA #$F0
F480  4C 2D FE            JMP L991        ;TOP CHANGED
F483                ;
F483                ;SET UP DDRB AND CB2 FOR RS-232
F483                ;
F483  A9 7F         L994  LDA #$7F         ;CASS OFF,TR MARK,CB1 HTL
F485  8D 0D DD            STA D2ICR
F488  A9 06               LDA #$06        ;DDRB
F48A  8D 03 DD            STA D2DDRB
F48D  8D 01 DD            STA D2DPB
F490  A9 04               LDA #$04
F492  0D 00 DD            ORA D2DPA
F495  8D 00 DD            STA D2DPA
F498  A0 00               LDY #$00
F49A  8C A1 02            STY $02A1
F49D  60                  RTS
F49E                      .END
F49E                      .LIB K14
F49E                ;
F49E                ;****************************************
F49E                ;* LOAD RAM FUNCTION.
F49E                ;*      LOADS FROM CASSETTE OR SERIAL BUS
F49E                ;* DEVICES >=4 TO 31 AS DETERMINED BY
F49E                ;* CONTENTS OF VARIABLE FA. VERIFY FLAG
F49E                ;* IN .A.
F49E                ;*      ALT LOAD IF SA=0, NORMAL SA=1
F49E                ;*      .X, .Y LOAD ADDRESS IF SA=0
F49E                ;*      .A=0 PERFORMS LOAD,<>0 IS VERIFY.
F49E                ;* HIGH LOAD RETURN IN .X,.Y
F49E                ;* USE SETLFS & SETNAM BEFORE THIS ROUTINE
F49E                ;****************************************
F49E                ;
F49E  86 C3         L990  STX MEMUSS       ;LO ALT START
F4A0  84 C4               STY MEMUSS+1    ;HI ALT START
F4A2  6C 30 03            JMP (ILOAD)
F4A5  85 93         NLOAD STA VERCK        ;STORE VERIFY FLAG
F4A7  A9 00               LDA #$00
F4A9  85 90               STA STATUS
F4AB  A5 BA               LDA FA          ;CHECK DEVICE #
F4AD  D0 03               BNE L1046
F4AF  4C 13 F7      L1241 JMP L1049        ;KEYBOARD, BAD DEVICE
F4B2  C9 03         L1046 CMP #$03         ;SCREEN?
F4B4  F0 F9               BEQ L1241       ;YES
F4B6  90 7B               BCC L1050       ;TAPE
F4B8                ;
F4B8                ;LOAD FROM SERIAL BUS DEVICES
F4B8                ;
F4B8  A4 B7               LDY FNLEN       ;MUST HAVE FILENAME
F4BA  D0 03               BNE L1045       ;O.K.
```

```
LOC   CODE        LINE

F4BC  4C 10 F7            JMP L974        ;MISSING FILENAME
F4BF  A6 B9       L1045   LDX SA
F4C1  20 AF F5            JSR L1062       ;'SEARCHING
F4C4  A9 60               LDA #$60        ;SPECIAL LOAD COMMAND
F4C6  85 B9               STA SA
F4C8  20 D5 F3            JSR L1021       ;OPEN FILE
F4CB  A5 BA               LDA FA
F4CD  20 09 ED            JSR L836        ;TALK, ESTABLISH CHANNEL
F4D0  A5 B9               LDA SA
F4D2  20 C7 ED            JSR L860        ;TELL IT TO LOAD
F4D5  20 13 EE            JSR L865        ;GET FIRST BYTE
F4D8  85 AE               STA EAL
F4DA  A5 90               LDA STATUS      ;ERROR?
F4DC  4A                  LSR A
F4DD  4A                  LSR A
F4DE  B0 50               BCS L1058       ;FILE NOT FOUND
F4E0  20 13 EE            JSR L865
F4E3  85 AF               STA EAH
F4E5  8A                  TXA             ;ORIG SA=0?
F4E6  D0 08               BNE L1048       ;NO
F4E8  A5 C3               LDA MEMUSS      ;YES, SET ALT
F4EA  85 AE               STA EAL         ; LOAD ADDRESS
F4EC  A5 C4               LDA MEMUSS+1
F4EE  85 AF               STA EAH
F4F0  20 D2 F5    L1048   JSR L1070        ;'LOADING'
F4F3  A9 FD       L1051   LDA #$FD         ;MASK OFF TIMEOUT
F4F5  25 90               AND STATUS
F4F7  85 90               STA STATUS
F4F9  20 E1 FF            JSR L309        ;STOP KEY?
F4FC  D0 03               BNE L1055       ;NO
F4FE  4C 33 F6            JMP L1084       ;'BREAK'
F501  20 13 EE    L1055   JSR L865         ;GET BYTE
F504  AA                  TAX
F505  A5 90               LDA STATUS      ;TIMEOUT?
F507  4A                  LSR A
F508  4A                  LSR A
F509  B0 E8               BCS L1051       ;YES, TRY AGAIN
F50B  8A                  TXA
F50C  A4 93               LDY VERCK       ;VERIFY?
F50E  F0 0C               BEQ L1053       ;NO, LOAD IT
F510  A0 00               LDY #$00
F512  D1 AE               CMP (EAL),Y     ;VERIFY IT
F514  F0 08               BEQ L1056       ;O.K.
F516  A9 10               LDA #$PERR      ;NO, VERIFY ERROR
F518  20 1C FE            JSR L1217       ;UPDATE STATUS
F51B  2C                  .BYT $2C        ;SKIP STORE
F51C  91 AE       L1053   STA (EAL),Y
F51E  E6 AE       L1056   INC EAL          ;INCREMENT STORE ADDR
F520  D0 02               BNE L1057
F522  E6 AF               INC EAH
F524  24 90       L1057   BIT STATUS       ;END OF INPUT?
F526  50 CB               BVC L1051       ;NO, CARRY ON
F528  20 EF ED            JSR L863        ;CLOSE CHANNEL
F52B  20 42 F6            JSR L1081       ;CLOSE FILE
F52E  90 79               BCC L1067       ;ALWAYS
F530  4C 04 F7    L1058   JMP L959         ;FILE NOT FOUND
F533                      ;
F533                      ;LOAD FROM TAPE
F533                      ;
F533  4A          L1050   LSR A            ;TAPE?
F534  B0 03               BCS L1047       ;YES
```

```
LOC   CODE        LINE

F536  4C 13 F7            JMP L1049       ;NO, BAD DEVICE
F539  20 D0 F7    L1047   JSR L1104        ;SET TAPE POINTERS
F53C  B0 03               BCS L1060
F53E  4C 13 F7            JMP L1049       ;DEALLOCATED
F541  20 17 F8    L1060   JSR L938         ;'PRESS PLAY ON TAPE'
F544  B0 68               BCS L1059       ;STOP KEY?
F546  20 AF F5            JSR L1062       ;'SEARCHING'
F549  A5 B7       L1061   LDA FNLEN        ;NAME?
F54B  F0 09               BEQ L1066       ;NO, LOAD FIRST PROG
F54D  20 EA F7            JSR L1108       ;YES, FIND A FILE
F550  90 0B               BCC L1063       ;FOUND
F552  F0 5A               BEQ L1059       ;STOP KEY
F554  B0 DA               BCS L1058       ;NO, END OF TAPE
F556  20 2C F7    L1066   JSR L1098        ;FIND ANY HEADER
F559  F0 53               BEQ L1059       ;STOP KEY
F55B  B0 D3               BCS L1058       ;NO HEADER
F55D  A5 90       L1063   LDA STATUS
F55F  29 10               AND #SPERR      ;MUST HAVE GOT HEADER RIGHT
F561  38                  SEC
F562  D0 4A               BNE L1059       ;IS BAD
F564  E0 01               CPX #BLF        ;MOVEABLE?
F566  F0 11               BEQ L1068       ;YES
F568  E0 03               CPX #PLF        ;PROGRAM
F56A  D0 DD               BNE L1061       ;NO, TRY FOR NEXT
F56C  A0 01       L1064   LDY #$01         ;FIXED LOAD
F56E  B1 B2               LDA (TAPE1),Y   ;ADDRESS IN BUFFER
F570  85 C3               STA MEMUSS      ;IS LOAD ADDRESS
F572  C8                  INY
F573  B1 B2               LDA (TAPE1),Y
F575  85 C4               STA MEMUSS+1
F577  B0 04               BCS L1065
F579  A5 B9       L1068   LDA SA           ;MONITOR LOAD?
F57B  D0 EF               BNE L1064       ;YES, FIXED TYPE
F57D  A0 03       L1065   LDY #$03         ;TAPEA-TAPESTA
F57F  B1 B2               LDA (TAPE1),Y
F581  A0 01               LDY #$01
F583  F1 B2               SBC (TAPE1),Y
F585  AA                  TAX             ;LO TO .X
F586  A0 04               LDY #$04
F588  B1 B2               LDA (TAPE1),Y
F58A  A0 02               LDY #$02
F58C  F1 B2               SBC (TAPE1),Y
F58E  A8                  TAY             ;HI TO .Y
F58F  18                  CLC             ;EA=STA+(TAPEA-TAPESTA)
F590  8A                  TXA
F591  65 C3               ADC MEMUSS
F593  85 AE               STA EAL
F595  98                  TYA
F596  65 C4               ADC MEMUSS+1
F598  85 AF               STA EAH
F59A  A5 C3               LDA MEMUSS      ;SET UP START ADDRESS
F59C  85 C1               STA STAL
F59E  A5 C4               LDA MEMUSS+1
F5A0  85 C2               STA STAH
F5A2  20 D2 F5            JSR L1070       ;'LOADING'
F5A5  20 4A F8            JSR L940        ;LOAD TAPE BLOCK
F5A8  24                  .BYT $24        ;SKIP NEXT COMMAND
F5A9  18          L1067   CLC              ;GOOD EXIT
F5AA              ;
F5AA              ;SET UP END ADDRESS
F5AA              ;
```

```
LOC   CODE        LINE

F5AA  A6 AE             LDX EAL
F5AC  A4 AF             LDY EAH
F5AE  60          L1059 RTS
F5AF              ;
F5AF              ;PRINT 'SEARCHING [FOR NAME]'
F5AF              ;
F5AF  A5 9D       L1062 LDA MSGFLG      ;PRINT IT?
F5B1  10 1E             BPL L1071       ;NO
F5B3  A0 0C             LDY #MS5-MS1    ;'SEARCHING'
F5B5  20 2F F1          JSR L1073
F5B8  A5 B7             LDA FNLEN
F5BA  F0 15             BEQ L1071
F5BC  A0 17             LDY #MS6-MS1    ;'FOR'
F5BE  20 2F F1          JSR L1073
F5C1              ;
F5C1              ;PRINT FILE NAME
F5C1              ;
F5C1  A4 B7       L1022 LDY FNLEN       ;NAME?
F5C3  F0 0C             BEQ L1071       ;NO, DONE
F5C5  A0 00             LDY #$00
F5C7  B1 BB       L1091 LDA (FNADR),Y
F5C9  20 D2 FF          JSR L622
F5CC  C8                INY
F5CD  C4 B7             CPY FNLEN
F5CF  D0 F6             BNE L1091
F5D1  60          L1071 RTS
F5D2              ;
F5D2              ;PRINT LOADING/VERIFYING
F5D2              ;
F5D2  A0 49       L1070 LDY #MS10-MS1   ;ASSUME 'LOADING'
F5D4  A5 93             LDA VERCK       ;CHECK FLAG
F5D6  F0 02             BEQ L1052       ;YES, LOADING
F5D8  A0 59             LDY #MS21-MS1   ;'VERIFYING'
F5DA  4C 2B F1    L1052 JMP L922
F5DD                    .END
F5DD                    .LIB K15
F5DD              ;
F5DD              ;**************************************
F5DD              ;* SAVE MEMORY FUNCTION.
F5DD              ;*      SAVES TO CASSETTE OR SERIAL
F5DD              ;* DEVICES >=4 TO 31 AS SELECTED BY
F5DD              ;* VARIABLE FA.
F5DD              ;*      START OF SAVE IS INDIRECT AT .A
F5DD              ;* END OF SAVE IS .X, .Y
F5DD              ;* USE SETLFS & SETNAM BEFORE THIS ROUTINE
F5DD              ;**************************************
F5DD              ;
F5DD  86 AE       L1072 STX EAL
F5DF  84 AF             STY EAH
F5E1  AA                TAX             ;SET UP START
F5E2  B5 00             LDA $00,X
F5E4  85 C1             STA STAL
F5E6  B5 01             LDA $01,X
F5E8  85 C2             STA STAH
F5EA  6C 32 03          JMP (ISAVE)
F5ED  A5 BA       NSAVE LDA FA
F5EF  D0 03             BNE L1075
F5F1  4C 13 F7    L1242 JMP L1049       ;BAD DEVICE
F5F4  C9 03       L1075 CMP #$03        ;SERIAL?
F5F6  F0 F9             BEQ L1242       ;SCREEN, BAD DEVICE
F5F8  90 5F             BCC L1085       ;NO, TAPE
```

```
LOC   CODE        LINE

F5FA  A9 61             LDA #$61        ;YES
F5FC  85 B9             STA SA
F5FE  A4 B7             LDY FNLEN
F600  D0 03             BNE L1074
F602  4C 10 F7          JMP L974        ;MISSING FILE NAME
F605  20 D5 F3    L1074 JSR L1021        ;OPEN
F608  20 8F F6          JSR L1087       ;'SAVING'
F60B  A5 BA             LDA FA
F60D  20 0C ED          JSR L966        ;LISTEN
F610  A5 B9             LDA SA
F612  20 B9 ED          JSR L871        ;LISTEN SA
F615  A0 00             LDY #$00
F617  20 8E FB          JSR L1174
F61A  A5 AC             LDA SAL
F61C  20 DD ED          JSR L861
F61F  A5 AD             LDA SAH
F621  20 DD ED          JSR L861
F624  20 D1 FC    L1077 JSR L1193        ;COMPARE START TO END
F627  B0 16             BCS L1082       ;HAVE REACHED END
F629  B1 AC             LDA (SAL),Y
F62B  20 DD ED          JSR L861
F62E  20 E1 FF          JSR L309        ;STOP KEY?
F631  D0 07             BNE L1054       ;NO
F633  20 42 F6    L1084 JSR L1081        ;YES, CLOSE
F636  A9 00             LDA #$00
F638  38                SEC
F639  60                RTS
F63A              ;
F63A  20 DB FC    L1054 JSR L1080        ;INCREMENT CURRENT ADDR
F63D  D0 E5             BNE L1077
F63F  20 FE ED    L1082 JSR L1006        ;UNLISTEN
F642  24 B9       L1081 BIT SA
F644  30 11             BMI L1034
F646  A5 BA             LDA FA
F648  20 0C ED          JSR L966        ;LISTEN
F64B  A5 B9             LDA SA
F64D  29 EF             AND #$EF
F64F  09 E0             ORA #$E0
F651  20 B9 ED          JSR L871        ;LISTEN SA
F654  20 FE ED    L999  JSR L1006        ;UNLISTEN
F657  18          L1034 CLC             ;GOOD EXIT
F658  60                RTS
F659              ;
F659              ;TAPE SAVE
F659              ;
F659  4A          L1085 LSR A           ;RS-232?
F65A  B0 03             BCS L1076       ;NO, MUST BE TAPE
F65C  4C 13 F7          JMP L1049       ;BAD DEVICE
F65F  20 D0 F7    L1076 JSR L1104        ;GET BUFFER ADDR
F662  90 8D             BCC L1242       ;NOT ALLOCATED
F664  20 38 F8          JSR L1114
F667  B0 25             BCS L1090       ;STOP KEY
F669  20 8F F6          JSR L1087       ;'SAVING'
F66C  A2 03             LDX #PLF        ;DECIDE TYPE TO SAVE
F66E  A5 B9             LDA SA          ;1-PLF, 0-BLF
F670  29 01             AND #$01
F672  D0 02             BNE L1086
F674  A2 01             LDX #BLF
F676  8A          L1086 TXA
F677  20 6A F7          JSR L1099       ;INITIALISE WRITE
F67A  B0 12             BCS L1090       ;STOP KEY
```

```
LOC   CODE        LINE

F67C  20 67 F8            JSR L952        ;WRITE HEADER
F67F  B0 0D               BCS L1090       ;STOP KEY
F681  A5 B9               LDA SA
F683  29 02               AND #$02        ;WRITE END OF TAPE?
F685  F0 06               BEQ L1088       ;NO
F687  A9 05               LDA #$05
F689  20 6A F7            JSR L1099
F68C  24                  .BYT $24        ;SKIP COMMAND
F68D  18            L1088 CLC
F68E  60            L1090 RTS
F68F                ;
F68F                ;PRINT 'SAVING [FILE NAME]'
F68F                ;
F68F  A5 9D         L1087 LDA MSGFLG       ;PRINT IT?
F691  10 FB               BPL L1090       ;NO
F693  A0 51               LDY #MS11-MS1   ;'SAVING'
F695  20 2F F1            JSR L1073
F698  4C C1 F5            JMP L1022       ;SEND FILENAME
F69B                      .END
F69B                      .LIB K16
F69B                ;
F69B                ;***************************************
F69B                ;* UPDATE TIME.
F69B                ;*      THIS ROUTINE UPDATES THE TIME
F69B                ;* EVERY 60TH OF A SECOND. THE STOP KEY
F69B                ;* IS ALSO LOGGED AT THIS POINT.
F69B                ;***************************************
F69B                ;
F69B                ;INTERRUPTS ARE COMING FROM 6526 TIMERS
F69B                ;
F69B  A2 00         L1078 LDX #$00         ;PRE-LOAD FOR LATER
F69D  E6 A2               INC TIME+2      ;INCREMENT THE TIME
F69F  D0 06               BNE L1245       ; REGISTERS
F6A1  E6 A1               INC TIME+1
F6A3  D0 02               BNE L1245
F6A5  E6 A0               INC TIME
F6A7  38            L1245 SEC             ;CHECK FOR 24 HR WRAP
F6A8  A5 A2               LDA TIME+2      ; AROUND
F6AA  E9 01               SBC #$01        ;LO BYTE +1 (CORRECTION)
F6AC  A5 A1               LDA TIME+1
F6AE  E9 1A               SBC #$1A        ;MIDDLE BYTE
F6B0  A5 A0               LDA TIME
F6B2  E9 4F               SBC #$4F        ;HI BYTE
F6B4  90 06               BCC L1092
F6B6  86 A0               STX TIME        ;YES WRAP, SET ALL TIME
F6B8  86 A1               STX TIME+1      ; REGISTERS TO ZERO
F6BA  86 A2               STX TIME+2
F6BC  AD 01 DC      L1092 LDA D1DPB        ;NO WRAP, SET STOP
F6BF  CD 01 DC            CMP D1DPB       ; KEY FLAG & WAIT TO SETTLE
F6C2  D0 F8               BNE L1092
F6C4  AA                  TAX
F6C5  30 13               BMI L1095
F6C7  A2 BD               LDX #$BD
F6C9  8E 00 DC            STX D1DPA
F6CC  AE 01 DC      L1093 LDX D1DPB
F6CF  EC 01 DC            CPX D1DPB
F6D2  D0 F8               BNE L1093
F6D4  8D 00 DC            STA D1DPA
F6D7  E8                  INX
F6D8  D0 02               BNE L1094
F6DA  85 91         L1095 STA STKEY
```

```
LOC   CODE        LINE

F6DC  60          L1094  RTS
F6DD              ;
F6DD              ;**************************************
F6DD              ;* READ TIME.
F6DD              ;*      THIS ROUTINE RETURNS THE TIME
F6DD              ;* VALUE STORED IN .A(LO), .X(NEXT),
F6DD              ;* .Y(HI).
F6DD              ;**************************************
F6DD              ;
F6DD  78          L1096  SEI              ;DISABLE TIME
F6DE  A5 A2              LDA TIME+2      ; INCREMENT
F6E0  A6 A1              LDX TIME+1      ;SET REGISTER WITH TIME
F6E2  A4 A0              LDY TIME
F6E4              ;
F6E4              ;**************************************
F6E4              ;* SET TIME.
F6E4              ;*      THIS ROUTINE SETS THE TIME FROM
F6E4              ;* REGISTERS .A(LO), .X(NEXT), .Y(HI)
F6E4              ;**************************************
F6E4              ;
F6E4  78          L1244  SEI              ;DISABLE TIME
F6E5  85 A2              STA TIME+2      ; INCREMENT
F6E7  86 A1              STX TIME+1      ;STORE TIME FROM REGISTEI
F6E9  84 A0              STY TIME
F6EB  58                 CLI
F6EC  60                 RTS
F6ED                     .END
F6ED                     .LIB K17
F6ED              ;
F6ED              ;**************************************
F6ED              ;* STOP -- CHECK STOP KEY FLAG AND
F6ED              ;* RETURN Z FLAG SET IF FLAG TRUE.
F6ED              ;* ALSO CLOSES ACTIVE CHANNELS AND
F6ED              ;* FLUSHES KEYBOARD QUEUE.
F6ED              ;* ALSO RETURNS KEY DOWNS FROM LAST
F6ED              ;* KEYBOARD ROW IN .A.
F6ED              ;*      SHOULD CALL UPDATE TIME BEFORE
F6ED              ;* THIS.
F6ED              ;**************************************
F6ED              ;
F6ED  A5 91       NSTOP  LDA STKEY        ;VALUE OF LAST ROW
F6EF  C9 7F              CMP #$7F        ;STOP KEY POSITION
F6F1  D0 07              BNE L1243       ;NOT DOWN
F6F3  08                 PHP
F6F4  20 CC FF           JSR L674        ;CLEAR CHANNELS
F6F7  85 C6              STA NDX         ;CLEAR KEY QUEUE
F6F9  28                 PLP
F6FA  60          L1243  RTS
F6FB              ;
F6FB              ;**************************************
F6FB              ;* ERROR HANDLER.
F6FB              ;* PRINTS KERNAL ERROR MESSAGE IF BIT 6
F6FB              ;* OF MSGFLG IS SET. RETURNS WITH ERROR
F6FB              ;* # IN .A AND CARRY SET.
F6FB              ;**************************************
F6FB              ;
F6FB  A9 01       L1097  LDA #$01         ;TOO MANY FILES
F6FD  2C                 .BYT $2C
F6FE  A9 02       L1011  LDA #$02         ;FILE OPEN
F700  2C                 .BYT $2C
F701  A9 03       L1009  LDA #$03         ;FILE NOT OPEN
```

```
LOC   CODE        LINE

F703  2C                  .BYT $2C
F704  A9 04       L959    LDA #$04        ;FILE NOT FOUND
F706  2C                  .BYT $2C
F707  A9 05       L1026   LDA #$05        ;DEVICE NOT PRESENT
F709  2C                  .BYT $2C
F70A  A9 06       L971    LDA #$06        ;NOT INPUT FILE
F70C  2C                  .BYT $2C
F70D  A9 07       L965    LDA #$07        ;NOT OUTPUT FILE
F70F  2C                  .BYT $2C
F710  A9 08       L974    LDA #$08        ;MISSING FILE NAME
F712  2C                  .BYT $2C
F713  A9 09       L1049   LDA #$09        ;BAD DEVICE #
F715  48                  PHA             ;ERROR # ON STACK
F716  20 CC FF            JSR L674        ;RESTORE I/O
F719  A0 00               LDY #MS1-MS1
F71B  24 9D               BIT MSGFLG      ;PRINT ERROR?
F71D  50 0A               BVC L1018       ;NO
F71F  20 2F F1            JSR L1073       ;PRINT 'I/O ERROR #'
F722  68                  PLA
F723  48                  PHA
F724  09 30               ORA #$30        ;MAKE ERROR # ASCII
F726  20 D2 FF            JSR L622        ;PRINT IT
F729  68          L1018   PLA
F72A  38                  SEC
F72B  60                  RTS
F72C                      .END
F72C                      .LIB K18
F72C                ;
F72C                ;***************************************
F72C                ;* FIND ANY TAPE HEADER.
F72C                ;* READS TAPE DEVICE UNTIL ONE OF THE
F72C                ;* FOLOWING BLOCK TYPES IS FOUND: BDFH--
F72C                ;* BASIC DATA FILE HEADER, BLF--BASIC
F72C                ;* LOAD FILE. FOR SUCCESS, CARRY IS
F72C                ;* CLEAR ON RETURN. FOR FAILURE, CARRY
F72C                ;* IS SET ON RETURN. IN ADDITION, .A IS
F72C                ;* 0 IF STOP KEY WAS PRESSED.
F72C                ;***************************************
F72C                ;
F72C  A5 93       L1098   LDA VERCK        ;SAVE OLD VERIFY
F72E  48                  PHA
F72F  20 41 F8            JSR L1029       ;READ TAPE BLOCK
F732  68                  PLA
F733  85 93               STA VERCK       ;RESTORE VERIFY
F735  B0 32               BCS L1101       ;READ TERMINATED
F737  A0 00               LDY #$00
F739  B1 B2               LDA (TAPE1),Y   ;GET HEADER TYPE
F73B  C9 05               CMP #EOT        ;END OF TAPE?
F73D  F0 2A               BEQ L1101       ;YES
F73F  C9 01               CMP #BLF        ;BASIC LOAD FILE?
F741  F0 08               BEQ L1027       ;YES
F743  C9 03               CMP #FLF        ;FIXED LOAD FILE?
F745  F0 04               BEQ L1027       ;YES
F747  C9 04               CMP #BDFH       ;BASIC DATA FILE?
F749  D0 E1               BNE L1098       ;NO, TRY AGAIN
F74B  AA          L1027   TAX              ;FILE TYPE IN .X
F74C  24 9D               BIT MSGFLG      ;PRINT MESSAGE?
F74E  10 17               BPL L1102       ;NO
F750  A0 63               LDY #MS17-MS1   ;'FOUND'
F752  20 2F F1            JSR L1073
F755  A0 05               LDY #$05
```

```
LOC   CODE        LINE

F757  B1 B2       L1100  LDA (TAPE1),Y   ;OUTPUT COMPLETE
F759  20 D2 FF           JSR L622        ;FILE NAME
F75C  C8                 INY
F75D  C0 15              CPY #$15
F75F  D0 F6              BNE L1100
F761  A5 A1              LDA TIME+1      ;WAIT FOR 8.5 SECONDS
F763  20 E0 E4           JSR L806        ; OR FOR THE CBM KEY
F766  EA                 NOP
F767  18          L1102  CLC              ;SUCCESS
F768  88                 DEY
F769  60          L1101  RTS
F76A              ;
F76A              ;***************************************
F76A              ;* WRITE TAPE HEADER.
F76A              ;*      ERROR IF TAPE BUFFER DE-ALLOCATED
F76A              ;* CARRY CLEAR IF O.K.
F76A              ;***************************************
F76A              ;
F76A  85 9E       L1099  STA T1
F76C  20 D0 F7           JSR L1104       ;GET BUFFER ADDRESS
F76F  90 5E              BCC L1106       ;NOT ALLOCATED
F771  A5 C2              LDA STAH        ;PRESERVE START AND END
F773  48                 PHA             ; ADDRESSES
F774  A5 C1              LDA STAL
F776  48                 PHA
F777  A5 AF              LDA EAH
F779  48                 PHA
F77A  A5 AE              LDA EAL
F77C  48                 PHA
F77D  A0 BF              LDY #BUFSZ-1    ;BLANK TAPE BUFFER
F77F  A9 20              LDA #$20        ;SPACE CHARS
F781  91 B2       L998   STA (TAPE1),Y
F783  88                 DEY
F784  D0 FB              BNE L998
F786  A5 9E              LDA T1          ;BLOCK TYPE IN HEADER
F788  91 B2              STA (TAPE1),Y
F78A  C8                 INY
F78B  A5 C1              LDA STAL        ;START ADDRESS IN HEADER
F78D  91 B2              STA (TAPE1),Y
F78F  C8                 INY
F790  A5 C2              LDA STAH
F792  91 B2              STA (TAPE1),Y
F794  C8                 INY             ;END ADDRESS IN HEADER
F795  A5 AE              LDA EAL
F797  91 B2              STA (TAPE1),Y
F799  C8                 INY
F79A  A5 AF              LDA EAH
F79C  91 B2              STA (TAPE1),Y
F79E  C8                 INY             ;FILE NAME IN HEADER
F79F  84 9F              STY T2
F7A1  A0 00              LDY #$00
F7A3  84 9E              STY T1
F7A5  A4 9E       L1105  LDY T1
F7A7  C4 B7              CPY FNLEN
F7A9  F0 0C              BEQ L1107
F7AB  B1 BB              LDA (FNADR),Y
F7AD  A4 9F              LDY T2
F7AF  91 B2              STA (TAPE1),Y
F7B1  E6 9E              INC T1
F7B3  E6 9F              INC T2
F7B5  D0 EE              BNE L1105
```

```
LOC   CODE        LINE

F7B7  20 D7 F7    L1107  JSR L995         ;SET UP START & END
F7BA  A9 69              LDA #$69        ;ADDR OF HEADER & SET
F7BC  85 AB              STA SHCNH       ; TIME FOR LEADER
F7BE  20 6B F8           JSR L1009       ;WRITE HEADER TO TAPE
F7C1  A8                 TAY             ;SAVE ERROR CODE IN .Y
F7C2  68                 PLA             ;RESTORE START & END
F7C3  85 AE              STA EAL         ; ADDRESSES
F7C5  68                 PLA
F7C6  85 AF              STA EAH
F7C8  68                 PLA
F7C9  85 C1              STA STAL
F7CB  68                 PLA
F7CC  85 C2              STA STAH
F7CE  98                 TYA             ;RESTORE ERROR CODE
F7CF  60          L1106  RTS
F7D0              ;
F7D0              ;RETURN BUFFER ADDRESS
F7D0              ;
F7D0  A6 B2       L1104  LDX TAPE1
F7D2  A4 B3              LDY TAPE1+1
F7D4  C0 02              CPY #$02        ;ALLOCATED?
F7D6  60                 RTS             ;CARRY CLEAR, DE-ALLOCATED
F7D7  20 D0 F7    L995   JSR L1104        ;GET PTR TO CASSETTE
F7DA  8A                 TXA
F7DB  85 C1              STA STAL        ;SAVE START LOW
F7DD  18                 CLC
F7DE  69 C0              ADC #BUFSZ      ;COMPUTE POINTER TO END
F7E0  85 AE              STA EAL         ;SAVE END LOW
F7E2  98                 TYA
F7E3  85 C2              STA STAH        ;SAVE START HI
F7E5  69 00              ADC #$00        ;COMPUTE POINTER TO END
F7E7  85 AF              STA EAH         ;SAVE END HIGH
F7E9  60                 RTS
F7EA              ;
F7EA  20 2C F7    L1108  JSR L1098        ;FIND ANY HEADER
F7ED  B0 1D              BCS L1111       ;FAILED
F7EF  A0 05              LDY #$05        ;CHECK NAME
F7F1  84 9F              STY T2          ;OFFSET TO HEADER
F7F3  A0 00              LDY #$00
F7F5  84 9E              STY T1          ;OFFSET TO NAME
F7F7  C4 B7       L1024  CPY FNLEN        ;COMPARE THIS MANY
F7F9  F0 10              BEQ L1112       ;DONE
F7FB  B1 BB              LDA (FNADR),Y
F7FD  A4 9F              LDY T2
F7FF  D1 B2              CMP (TAPE1),Y
F801  D0 E7              BNE L1108       ;WRONG FILE NAME
F803  E6 9E              INC T1
F805  E6 9F              INC T2
F807  A4 9E              LDY T1
F809  D0 EC              BNE L1024       ;ALWAYS
F80B  18          L1112  CLC              ;SUCCESS
F80C  60          L1111  RTS
F80D                     .END
F80D                     .LIB K19
F80D  20 D0 F7    L1110  JSR L1104
F810  E6 A6              INC BUFPT
F812  A4 A6              LDY BUFPT
F814  C0 C0              CPY #BUFSZ
F816  60                 RTS
F817              ;
F817              ;WAIT FOR PLAY SWITCH
```

```
LOC   CODE        LINE

F817                ;
F817  20 2E F8      L938    JSR L1116
F81A  F0 1A                 BEQ L1113
F81C  A0 1B                 LDY #MS7-MS1    ;'PRESS PLAY...'
F81E  20 2F F1      L1020   JSR L1073
F821  20 D0 F8      L1117   JSR L1125        ;TEST STOP KEY
F824  20 2E F8              JSR L1116       ;TEST CASSETTE SWITCHES
F827  D0 F8                 BNE L1117
F829  A0 6A                 LDY #MS18-MS1   ;'OK'
F82B  4C 2F F1              JMP L1073
F82E                ;
F82E                ;TEST CASSETTE SWITCH
F82E                ;
F82E  A9 10         L1116   LDA #$10         ;CHECK PORT
F830  24 01                 BIT $01         ;CLOSED?
F832  D0 02                 BNE L1113       ;NO
F834  24 01                 BIT $01         ;DEBOUNCE
F836  18            L1113   CLC              ;GOOD EXIT
F837  60                    RTS
F838                ;
F838                ;CHECK FOR RECORD & PLAY
F838                ;
F838  20 2E F8      L1114   JSR L1116
F83B  F0 F9                 BEQ L1113
F83D  A0 2E                 LDY #MS8-MS1    ;'PRESS RECORD...'
F83F  D0 DD                 BNE L1020
F841                ;
F841                ;READ HEADER BLOCK
F841                ;
F841  A9 00         L1029   LDA #$00
F843  85 90                 STA STATUS
F845  85 93                 STA VERCK
F847  20 D7 F7              JSR L995
F84A                ;
F84A                ;READ LOAD BLOCK ENTRY
F84A                ;
F84A  20 17 F8      L940    JSR L938         ;'PRESS PLAY...'
F84D  B0 1F                 BCS L1109       ;STOP KEY
F84F  78                    SEI
F850  A9 00                 LDA #$00        ;CLEAR FLAGS
F852  85 AA                 STA RDFLG
F854  85 B4                 STA SNSW1
F856  85 B0                 STA CMP0
F858  85 9E                 STA PTR1
F85A  85 9F                 STA PTR2
F85C  85 9C                 STA DPSW
F85E  A9 90                 LDA #$90        ;ENABLE FOR TAPE IRQ
F860  A2 0E                 LDX #$0E        ;POINT IRQ VECTOR TO READ
F862  D0 11                 BNE L1118       ;ALWAYS
F864                ;
F864                ;WRITE HEADER BLOCK
F864                ;
F864  20 D7 F7      L1069   JSR L995
F867  A9 14         L952    LDA #$14         ;BETWEEN BLOCK SHORTS
F869  85 AB                 STA SHCNH
F86B  20 38 F8      L1089   JSR L1114        ;'PRESS RECORD...'
F86E  B0 6C         L1109   BCS L1115        ;STOP KEY
F870  78                    SEI
F871  A9 82                 LDA #$82        ;ENABLE T2 IRQ
F873  A2 08                 LDX #$08        ;POINT IRQ VECTOR TO WRITE
F875                ;
```

```
LOC   CODE        LINE

F875              ;START TAPE OPERATION ENTRY POINT
F875              ;
F875  A0 7F       L1118  LDY #$7F         ;KILL UNWANTED IRQ
F877  8C 0D DC           STY D1ICR
F87A  8D 0D DC           STA D1ICR        ;ENABLE WANTED
F87D  AD 0E DC           LDA D1CRA
F880  09 19              ORA #$19
F882  8D 0F DC           STA D1CRB
F885  29 91              AND #$91
F887  8D A2 02           STA $02A2
F88A  20 A4 F0           JSR L921         ;WAIT FOR RS-232
F88D  AD 11 D0           LDA VICREG+17    ;BLANK SCREEN
F890  29 EF              AND #$EF
F892  8D 11 D0           STA VICREG+17
F895  AD 14 03           LDA CINV         ;MOVE IRQ TO IRQTEMP
F898  8D 9F 02           STA IRQTMP       ;FOR CASSETTE OPS
F89B  AD 15 03           LDA CINV+1
F89E  8D A0 02           STA IRQTMP+1
F8A1  20 BD FC           JSR L1195        ;CHANGE IRQ VECTOR
F8A4  A9 02              LDA #$02         ;FSBLK STARTS AT 2
F8A6  85 BE              STA FSBLK
F8A8  20 97 FB           JSR L1079        ;PREPARE LOCAL COUNTERS
F8AB  A5 01              LDA $01          ;TURN CASSETTE MOTOR ON
F8AD  29 1F              AND #$1F
F8AF  85 01              STA $01
F8B1  85 C0              STA CAS1         ;FLAG INTERNAL CONTROL
F8B3  A2 FF              LDX #$FF         ;DELAY BETWEEN BLOCKS
F8B5  A0 FF       L1119  LDY #$FF
F8B7  88          L1124  DEY
F8B8  D0 FD              BNE L1124
F8BA  CA                 DEX
F8BB  D0 F8              BNE L1119
F8BD  58                 CLI
F8BE  AD A0 02    L1123  LDA IRQTMP+1     ;CHECK FOR IRQ VECTOR
F8C1  CD 15 03           CMP CINV+1       ;POINTING AT KEY ROUTINE
F8C4  18                 CLC
F8C5  F0 15              BEQ L1115        ;YES, RETURN
F8C7  20 D0 F8           JSR L1125        ;NO CHECK STOP
F8CA  20 BC F6           JSR L1092        ;UPDATE TIME
F8CD  4C BE F8           JMP L1123        ;STAY IN LOOP
F8D0  20 E1 FF    L1125  JSR L309         ;TOP KEY DOWN?
F8D3  18                 CLC              ;ASSUME NOT
F8D4  D0 0B              BNE L1120        ;CORRECT ASSUMPTION
F8D6  20 93 FC           JSR L1192        ;STOP DOWN STOP TAPE
F8D9  38                 SEC              ;FAILED
F8DA  68                 PLA              ;BACK ON RTS
F8DB  68                 PLA
F8DC  A9 00       L1115  LDA #$00         ;DISABLE IRQTMP
F8DE  8D A0 02           STA IRQTMP+1
F8E1  60          L1120  RTS
F8E2              ;
F8E2              ;SET UP TIMEOUT WATCH FOR NEXT DIPOLE
F8E2              ;
F8E2  86 B1       L1126  STX TEMP         ;TIMEOUT CONSTANT
F8E4  A5 B0              LDA CMPO         ;CMPO*5
F8E6  0A                 ASL A
F8E7  0A                 ASL A
F8E8  18                 CLC
F8E9  65 B0              ADC CMPO
F8EB  18                 CLC
F8EC  65 B1              ADC TEMP         ;ADJUST LONG BYTE COUNT
```

```
LOC   CODE        LINE

F8EE  85 B1             STA TEMP
F8F0  A9 00             LDA #$00
F8F2  24 B0             BIT CMPO        ;CHECK CMPO..
F8F4  30 01             BMI L1146       ; MINUS, NO ADJUST
F8F6  2A                ROL A           ; PLUS, ADJUST POS
F8F7  06 B1       L1146 ASL TEMP         ;MULTIPLY CORRECTED
F8F9  2A                ROL A           ; VALUE BY 4
F8FA  06 B1             ASL TEMP
F8FC  2A                ROL A
F8FD  AA                TAX
F8FE  AD 06 DC    L1128 LDA D1TBL        ;WATCH OUT FOR ROLLOVER
F901  C9 16             CMP #$16        ;TIME FOR ROUTINE?
F903  90 F9             BCC L1128       ;TOO CLOSE SO WAIT
F905  65 B1             ADC TEMP        ;CALCULATE AND
F907  8D 04 DC          STA D1TAL       ; STORE ADJUSTED TIME COUNT
F90A  8A                TXA
F90B  6D 07 DC          ADC D1TBH       ;ADJUST FOR HI TIME COUNT
F90E  8D 05 DC          STA D1TAH
F911  AD A2 02          LDA $02A2
F914  8D 0E DC          STA D1CRA
F917  8D A4 02          STA $02A4
F91A  AD 0D DC          LDA D1ICR
F91D  29 10             AND #$10
F91F  F0 09             BEQ L1129
F921  A9 F9             LDA #$F9
F923  48                PHA
F924  A9 2A             LDA #$2A
F926  48                PHA
F927  4C 43 FF          JMP L1041
F92A  58          L1129 CLI
F92B  60                RTS
F92C                    .END
F92C                    .LIB K20
F92C              ;
F92C              ;***************************************
F92C              ;* CASSETTE READ SUBROUTINES
F92C              ;***************************************
F92C              ;
F92C  AE 07 DC    L1130 LDX D1TBH        ;GET TIME SINCE LAST IRQ
F92F  A0 FF             LDY #$FF        ;COMPUTE COUNTER DIFF
F931  98                TYA
F932  ED 06 DC          SBC D1TBL
F935  EC 07 DC          CPX D1TBH       ;TIMER HIGH ROLLOVER?
F938  D0 F2             BNE L1130       ;YES, RECOMPUTE
F93A  86 B1             STX TEMP
F93C  AA                TAX
F93D  8C 06 DC          STY D1TBL       ;RE-LOAD TIMER B
F940  8C 07 DC          STY D1TBH
F943  A9 19             LDA #$19
F945  8D 0F DC          STA D1CRB
F948  AD 0D DC          LDA D1ICR
F94B  8D A3 02          STA $02A3
F94E  98                TYA
F94F  E5 B1             SBC TEMP        ;CALCULATE HIGH
F951  86 B1             STX TEMP
F953  4A                LSR A           ;MOVE 2 BITS FROM
F954  66 B1             ROR TEMP        ; HIGH TO TEMP
F956  4A                LSR A
F957  66 B1             ROR TEMP
F959  A5 B0             LDA CMPO        ;CALC MIN PULSE VALUE
F95B  18                CLC
```

```
LOC   CODE        LINE

F95C  69 3C               ADC #$3C
F95E  C5 B1               CMP TEMP        ;PULSE LESS THAN MIN?
F960  B0 4A               BCS L1141       ;YES, NOISE
F962  A6 9C               LDX DPSW        ;NO, LAST BIT?
F964  F0 03               BEQ L1132       ;NO, CONTINUE
F966  4C 60 FA            JMP L1154       ;YES, FINISH BYTE
F969                ;
F969  A6 A3         L1132 LDX PCNTR        ;9 BITS READ?
F96B  30 1B               BMI L1134       ;YES, GOTO ENDING
F96D  A2 00               LDX #$00        ;SET BIT VAL TO ZERO
F96F  69 30               ADC #$30        ;ADD UP TO HALF WAY BETWEEN
F971  65 B0               ADC CMPO        ; SHIRT PULSE AND SYNC PULSE
F973  C5 B1               CMP TEMP        ;SHORT?
F975  B0 1C               BCS L1139       ;YES
F977  E8                  INX             ;SET BIT VAL TO 1
F978  69 26               ADC #$26        ;MOVE TO MIDDLE OF HIGH
F97A  65 B0               ADC CMPO
F97C  C5 B1               CMP TEMP        ;1?
F97E  B0 17               BCS L1137       ;YES
F980  69 2C               ADC #$2C        ;MOVE TO LONGLONG
F982  65 B0               ADC CMPO
F984  C5 B1               CMP TEMP        ;LONG-LONG?
F986  90 03               BCC L1136       ;GREATER THAN, ERROR
F988  4C 10 FA      L1134 JMP L1145        ;YES
F98B                ;
F98B  A5 B4         L1136 LDA SNSW1        ;NOT SYNCRONIZED?
F98D  F0 1D               BEQ L1141       ;NO, ERROR
F98F  85 A8               STA RER         ;YES, FLAG RER
F991  D0 19               BNE L1141       ;ALWAYS
F993                ;
F993  E6 A9         L1139 INC REZ          ;COUNT REZ UP ON ZEROS
F995  B0 02               BCS L1138       ;ALWAYS
F997                ;
F997  C6 A9         L1137 DEC REZ          ;COUNT REZ DOWN ON ONES
F999  38            L1138 SEC              ;CALC ACTUAL VAL FOR COMPARE
F99A  E9 13               SBC #$13
F99C  E5 B1               SBC TEMP        ;SUBTRACT INPUT VAL
F99E  65 92               ADC SVXT        ; ADD DIFF TO TEMP STORE
F9A0  85 92               STA SVXT        ; USED TO ADJUST SOFT SERVO
F9A2  A5 A4               LDA FIRT        ;FLIP DIPOLE FLAG
F9A4  49 01               EOR #$01
F9A6  85 A4               STA FIRT
F9A8  F0 2B               BEQ L1143       ;SECOND HALF OF DIPOLE
F9AA  86 D7               STX DATA        ;FIRST HALF SO STORE VAL
F9AC                ;
F9AC  A5 B4         L1141 LDA SNSW1        ;NO BYTE START?
F9AE  F0 22               BEQ L1150       ;YES, RETURN
F9B0  AD A3 02            LDA $02A3       ;TIMER 1 IRQ'D?
F9B3  29 01               AND #$01
F9B5  D0 05               BNE L1133       ;YES
F9B7  AD A4 02            LDA $02A4
F9BA  D0 16               BNE L1150       ;NO, EXIT
F9BC  A9 00         L1133 LDA #$00         ;SET DIPOLE FLAG FOR FIRST HALF
F9BE  85 A4               STA FIRT
F9C0  8D A4 02            STA $02A4
F9C3  A5 A3               LDA PCNTR       ;WHERE IN BYTE
F9C5  10 30               BPL L1148       ; STILL DOING DATA
F9C7  30 BF               BMI L1134       ; PROCESS PARITY
F9C9                ;
F9C9  A2 A6         L1144 LDX #$A6         ;SETUP FOR LONGLONG
F9CB  20 E2 F8            JSR L1126
```

```
LOC   CODE        LINE

F9CE  A5 9B               LDA PRTY       ;EVEN PARITY?
F9D0  D0 B9               BNE L1136      ;NO, SET ERROR
F9D2  4C BC FE    L1150   JMP L1228       ;RESTORE REGS AND RTI
F9D5              ;
F9D5  A5 92       L1143   LDA SVXT        ;ADJUST SOFT SERVO?
F9D7  F0 07               BEQ L1149      ;NO
F9D9  30 03               BMI L1142      ;YES, MORE BASE TIME
F9DB  C6 B0               DEC CMPO       ;YES, LESS BASE TIME
F9DD  2C                  .BYT $2C       ;SKIP NEXT
F9DE  E6 B0       L1142   INC CMPO
F9E0  A9 00       L1149   LDA #$00        ;CLEAR DIFF FLAG
F9E2  85 92               STA SVXT
F9E4  E4 D7               CPX DATA       ;CONSEC. LIKE VALS IN DIPOLE?
F9E6  D0 0F               BNE L1148      ;NO, PROCESS INFO
F9E8  8A                  TXA            ;YES, CHECK VALS
F9E9  D0 A0               BNE L1136      ;ONES, ERROR
F9EB  A5 A9               LDA REZ        ;HOW MANY ZEROS?
F9ED  30 BD               BMI L1141      ; TOO MANY
F9EF  C9 10               CMP #$10       ; 16?
F9F1  90 B9               BCC L1141      ;NO, CONTINUE
F9F3  85 96               STA SYNO       ;YES, FLAG SYNO
F9F5  B0 B5               BCS L1141      ;ALWAYS
F9F7              ;
F9F7  8A          L1148   TXA             ;MOVE READ DATA TO .A
F9F8  45 9B               EOR PRTY       ;CALC PARITY
F9FA  85 9B               STA PRTY
F9FC  A5 B4               LDA SNSW1      ;REAL DATA?
F9FE  F0 D2               BEQ L1150      ;NO, FORGET
FA00  C6 A3               DEC PCNTR      ;DEC BIT COUNT
FA02  30 C5               BMI L1144      ;NEG, TIME FOR PARITY
FA04  46 D7               LSR DATA       ;SHIFT BIT FROM DATA
FA06  66 BF               ROR MYCH       ; INTO BYTE STORE
FA08  A2 DA               LDX #$DA       ;SETUP FOR NEXT DIPOLE
FA0A  20 E2 F8            JSR L1126
FA0D  4C BC FE            JMP L1228      ;RESTORE REGS AND RTI
FA10              ;
FA10              ;LONGLONG HANDLER
FA10              ;
FA10  A5 96       L1145   LDA SYNO        ;GOT BLOCK SYNC?
FA12  F0 04               BEQ L1140      ;NO
FA14  A5 B4               LDA SNSW1      ;HAD REAL BYTE?
FA16  F0 07               BEQ L1151      ;NO
FA18  A5 A3       L1140   LDA PCNTR       ;END OF BYTE?
FA1A  30 03               BMI L1151      ;YES
FA1C  4C 97 F9            JMP L1137      ;NO, TREAT AS LONG
FA1F              ;
FA1F  46 B1       L1151   LSR TEMP        ;ADJUST TIMEOUT FOR
FA21  A9 93               LDA #$93       ; LONGLONG PULSE VAL
FA23  38                  SEC
FA24  E5 B1               SBC TEMP
FA26  65 B0               ADC CMPO
FA28  0A                  ASL A
FA29  AA                  TAX            ;SET TIMEOUT FOR LAST BIT
FA2A  20 E2 F8            JSR L1126
FA2D  E6 9C               INC DPSW       ;SET BIT THROW AWAY FLAG
FA2F  A5 B4               LDA SNSW1      ;BYTE SYNCRONIZED?
FA31  D0 11               BNE L1152      ;YES, SKIP TO PASS CHAR
FA33  A5 96               LDA SYNO       ;THROW OUT DATA UNTIL SYNC
FA35  F0 26               BEQ L1155      ;NO SYNC
FA37  85 A8               STA RER        ;FLAG DATA AS ERROR
FA39  A9 00               LDA #$00       ;KILL 16 SYNC FLAG
```

```
LOC   CODE        LINE

FA3B  85 96             STA SYNO
FA3D  A9 81             LDA #$81        ;SETUP FOR TIMERB IRQ
FA3F  8D 0D DC          STA D1ICR
FA42  85 B4             STA SNSW1       ;FLAG WE HAVE BYTE SYNC
FA44                ;
FA44  A5 96         L1152  LDA SYNO      ;SAVE SYNO STATUS
FA46  85 B5             STA DIFF
FA48  F0 09             BEQ L1153       ;NO BLOCK SYNC
FA4A  A9 00             LDA #$00        ;TURN OFF BYTE SYNC SWITCH
FA4C  85 B4             STA SNSW1
FA4E  A9 01             LDA #$01        ;DISABLE TIMERB IRQ
FA50  8D 0D DC          STA D1ICR
FA53  A5 BF         L1153  LDA MYCH      ;PASS CHAR TO BYTE ROUTINE
FA55  85 BD             STA OCHAR
FA57  A5 A8             LDA RER         ;COMBINE ERROR VALS
FA59  05 A9             ORA REZ
FA5B  85 B6             STA PRP         ; AND SAVE IN PRP
FA5D  4C BC FE      L1155  JMP L1228     ;GET LAST BYTE
FA60                ;
FA60  20 97 FB      L1154  JSR L1079     ;FINISH BYTE, CLR FLAGS
FA63  85 9C             STA DPSW        ;GET BIT THROW AWAY FLAG
FA65  A2 DA             LDX #$DA        ;INIT FOR NEXT DIPOLE
FA67  20 E2 F8          JSR L1126
FA6A  A5 BE             LDA FSBLK       ;CHECK FOR LAST VAL
FA6C  F0 02             BEQ L1135
FA6E  85 A7             STA SHCNL
FA70                ;
FA70                ;***************************************
FA70                ;* BYTE HANDLER OF CASSETTE READ.
FA70                ;*     RER IS SET IF THE BYTE IS IN
FA70                ;* ERROR. REZ IS SET IF THE INTERRUPT
FA70                ;* PROGRAM IS READING ZEROS. RDFLG TELLS
FA70                ;* US WHAT WE ARE DOING. BIT 7 SAYS TO
FA70                ;* IGNORE BYTES UNTIL REZ IS SET, BIT 6
FA70                ;* SAYS TO LOAD THE BYTE. OTHERWISE
FA70                ;* RDFLG IS A COUNTDOWN AFTER SYNC. IF
FA70                ;* VERCK IS SET WE DO A COMPARE INSTEAD
FA70                ;* OF A STORE AND SET STATUS. FSBLK
FA70                ;* COUNTS THE TWO BLOCKS. PTR1 IS THE
FA70                ;* INDEX TO THE ERROR TABLE FOR PASS1.
FA70                ;* PTR2 IS THE INDEX TO THE CORRECTION
FA70                ;* TABLE FOR PASS2.
FA70                ;***************************************
FA70                ;
FA70  A9 0F         L1135  LDA #$0F
FA72  24 AA             BIT RDFLG       ;TEST FUNCTION MODE
FA74  10 17             BPL L1159       ;NOT WAITING FOR ZEROS
FA76  A5 B5             LDA DIFF        ;ZEROS YET?
FA78  D0 0C             BNE L1156       ;YES, WAIT FOR SYNC
FA7A  A6 BE             LDX FSBLK       ;PASS OVER?
FA7C  CA                DEX             ;ZERO, NO ERROR
FA7D  D0 0B             BNE L1158       ;NO
FA7F  A9 0B             LDA #LBERR
FA81  20 1C FE          JSR L1217       ;YES, LONG BLOCK ERROR
FA84  D0 04             BNE L1158       ;ALWAYS
FA86  A9 00         L1156  LDA #$00
FA88  85 AA             STA RDFLG       ;NEW MODE, WAIT FOR SYNC
FA8A  4C BC FE      L1158  JMP L1228     ;EXIT, DONE
FA8D  70 31         L1159  BVS L1163     ;LOADING
FA8F  D0 18             BNE L1162       ;SYNCING
FA91  A5 B5             LDA DIFF        ;HAVE BLOCK SYNC?
```

```
LOC   CODE        LINE

FA93  D0 F5             BNE L1158       ;YES, EXIT
FA95  A5 B6             LDA FRP         ;FIRST BYTE IN ERROR?
FA97  D0 F1             BNE L1158       ;YES, EXIT
FA99  A5 A7             LDA SHCNL       ;MOVE FSBLK TO CARRY
FA9B  4A                LSR A
FA9C  A5 BD             LDA OCHAR       ; SHOULD BE A HEADER COUNT CHAR
FA9E  30 03             BMI L1157       ;NEG, FIRST BLOCK DATA
FAA0  90 18             BCC L1161       ;EXPECTING FIRST BLOCK DATA
FAA2  18                CLC
FAA3  B0 15       L1157 BCS L1161        ;EXPECTING 2ND BLOCK
FAA5  29 0F             AND #$0F        ;MASK OFF HIGH STORE HEADER
FAA7  85 AA             STA RDFLG       ; COUNT IN MODE FLAG
FAA9  C6 AA       L1162 DEC RDFLG        ;WAIT FOR REAL DATA
FAAB  D0 DD             BNE L1158       ; REAL
FAAD  A9 40             LDA #$40        ;NEXT UP IS REAL DATA
FAAF  85 AA             STA RDFLG       ; SET DATA MODE
FAB1  20 8E FB          JSR L1174       ;SETUP ADDR POINTERS
FAB4  A9 00             LDA #$00
FAB6  85 AB             STA SHCNH
FAB8  F0 D0             BEQ L1158       ;ALWAYS, EXIT
FABA  A9 80       L1161 LDA #$80         ;IGNORE BYTES MODE
FABC  85 AA             STA RDFLG
FABE  D0 CA             BNE L1158       ;ALWAYS
FAC0  A5 B5       L1163 LDA DIFF         ;END OF BLOCK?
FAC2  F0 0A             BEQ L1160       ;YES
FAC4  A9 04             LDA #SBERR      ;SHORT BLOCK ERROR
FAC6  20 1C FE          JSR L1217
FAC9  A9 00             LDA #$00        ;FORCE RDFLG FOR ERROR
FACB  4C 4A FB          JMP L1167
FACE  20 D1 FC    L1160 JSR L1193        ;END OF STORE AREA?
FAD1  90 03             BCC L1164       ;NOT YET
FAD3  4C 48 FB          JMP L1172       ;YES
FAD6  A6 A7       L1164 LDX SHCNL        ;WHICH PASS?
FAD8  CA                DEX
FAD9  F0 2D             BEQ L1169       ;SECOND
FADB  A5 93             LDA VERCK       ;LOAD OR VERIFY?
FADD  F0 0C             BEQ L1166       ;LOADING
FADF  A0 00             LDY #$00        ;VERIFYING
FAE1  A5 BD             LDA OCHAR
FAE3  D1 AC             CMP (SAL),Y     ;COMPARE
FAE5  F0 04             BEQ L1166       ;GOOD, CONTINUE
FAE7  A9 01             LDA #$01        ;BAD, FLAG
FAE9  85 B6             STA FRP         ; AS ERROR
FAEB              ;
FAEB              ;STORE BAD LOCATIONS FOR 2ND PASS RE-TRY
FAEB              ;
FAEB  A5 B6       L1166 LDA FRP          ;CHK FOR ERRORS
FAED  F0 4B             BEQ L1171       ;NONE
FAEF  A2 3D             LDX #$3D        ;MAX OF 30
FAF1  E4 9E             CPX PTR1        ;REACHED MAX?
FAF3  90 3E             BCC L1173       ;YES, FLAG 2ND PASS
FAF5  A6 9E             LDX PTR1        ;INDEX INTO BAD
FAF7  A5 AD             LDA SAH         ; AND STORE BAD LOC
FAF9  9D 01 01          STA BAD+1,X     ; IN BAD TABLE
FAFC  A5 AC             LDA SAL
FAFE  9D 00 01          STA BAD,X
FB01  E8                INX             ;ADVANCE TO NEXT
FB02  E8                INX
FB03  86 9E             STX PTR1
FB05  4C 3A FB          JMP L1171       ;STORE CHAR
FB08              ;
```

```
LOC   CODE        LINE

FB08              ;CHECK BAD TABLE FOR RE-TRY
FB08              ;
FB08  A6 9F       L1169  LDX PTR2        ;DONE ALL IN TABLE?
FB0A  E4 9E              CPX PTR1
FB0C  F0 35              BEQ L1170      ;YES
FB0E  A5 AC              LDA SAL        ;NEXT IN TABLE?
FB10  DD 00 01           CMP BAD,X
FB13  D0 2E              BNE L1170      ;NO
FB15  A5 AD              LDA SAH
FB17  DD 01 01           CMP BAD+1,X
FB1A  D0 27              BNE L1170      ;NO
FB1C  E6 9F              INC PTR2       ;FOUND NEXT ONE, ADVANCE
FB1E  E6 9F              INC PTR2
FB20  A5 93              LDA VERCK      ;LOAD OR VERIFY?
FB22  F0 0B              BEQ L1168      ;LOADING
FB24  A5 BD              LDA OCHAR      ;VERIFYING
FB26  A0 00              LDY #$00
FB28  D1 AC              CMP (SAL),Y
FB2A  F0 17              BEQ L1170      ;O.K.
FB2C  C8                 INY            ;.Y=1
FB2D  84 B6              STY PRP        ;FLAG IT AS AN ERROR
FB2F  A5 B6       L1168  LDA PRP         ;SECOND PASS ERROR?
FB31  F0 07              BEQ L1171      ;NO
FB33  A9 10       L1173  LDA #SPERR
FB35  20 1C FE           JSR L1217
FB38  D0 09              BNE L1170      ;ALWAYS
FB3A  A5 93       L1171  LDA VERCK       ;LOAD OR VERIFY?
FB3C  D0 05              BNE L1170      ;VERIFY
FB3E  A8                 TAY
FB3F  A5 BD              LDA OCHAR
FB41  91 AC              STA (SAL),Y    ;STORE CHARACTER
FB43  20 DB FC    L1170  JSR L1080       ;NEXT ADDRESS
FB46  D0 43              BNE L1177      ;ALWAYS
FB48              ;
FB48  A9 80       L1172  LDA #$80        ;SET SKIP NEXT DATA
FB4A  85 AA       L1167  STA RDFLG
FB4C  78                 SEI
FB4D  A2 01              LDX #$01
FB4F  8E 0D DC           STX D1ICR
FB52  AE 0D DC           LDX D1ICR
FB55  A6 BE              LDX FSBLK      ;DEC FSBLK FOR NEXT PASS
FB57  CA                 DEX
FB58  30 02              BMI L1165      ;DONE, FSBLK=0
FB5A  86 BE              STX FSBLK      ; ELSE, NEXT
FB5C  C6 A7       L1165  DEC SHCNL       ;DEC PASS CALC
FB5E  F0 08              BEQ L1175      ;ALL DONE
FB60  A5 9E              LDA PTR1       ;FIRST PASS ERRORS?
FB62  D0 27              BNE L1177      ;YES, CONTINUE
FB64  85 BE              STA FSBLK      ;CLEAR FSBLK IF NO ERRORS
FB66  F0 23              BEQ L1177      ;ALWAYS, EXIT
FB68              ;
FB68  20 93 FC    L1175  JSR L1192       ;READ IT ALL, EXIT
FB6B  20 8E FB           JSR L1174      ;RESTORE SAL & SAH
FB6E  A0 00              LDY #$00       ;SHCNH=0
FB70  84 AB              STY SHCNH      ; USED TO CALC PARITY BYTE
FB72              ;
FB72              ;COMPUTE PARITY OVERLOAD
FB72              ;
FB72  B1 AC       L1176  LDA (SAL),Y     ;CALC BLOCK BCC
FB74  45 AB              EOR SHCNH
FB76  85 AB              STA SHCNH
```

```
LOC   CODE        LINE

FB78  20 DB FC          JSR L1080       ;BUMP ADDRESS
FB7B  20 D1 FC          JSR L1193       ;AT END?
FB7E  90 F2             BCC L1176       ;NOT YET
FB80  A5 AB             LDA SHCNH       ;BCC CHAR MATCH?
FB82  45 BD             EOR OCHAR
FB84  F0 05             BEQ L1177       ;YES, EXIT
FB86  A9 20             LDA #CKERR      ;CHKSUM ERROR
FB88  20 1C FE          JSR L1217
FB8B  4C BC FE    L1177 JMP L1228
FB8E                    ;
FB8E  A5 C2       L1174 LDA STAH         ;RESTORE START ADDR
FB90  85 AD             STA SAH         ; TO POINTER SAH & SAL
FB92  A5 C1             LDA STAL
FB94  85 AC             STA SAL
FB96  60                RTS
FB97                    ;
FB97  A9 08       L1079 LDA #$08         ;SETUP FOR 8 BITS+PARITY
FB99  85 A3             STA PCNTR
FB9B  A9 00             LDA #$00        ;INITIALIZE
FB9D  85 A4             STA FIRT        ; DIPOLE COUNTER
FB9F  85 A8             STA RER         ; ERROR FLAG
FBA1  85 9B             STA PRTY        ; PARITY BIT
FBA3  85 A9             STA REZ         ; ZERO COUNT
FBA5  60                RTS             ;.A=0 ON RETURN
FBA6                    .END
FBA6                    .LIB K21
FBA6                ;
FBA6                ;***************************************
FBA6                ;* CASSETTE WRITE SUBROUTINES.
FBA6                ;*      FSBLK IS BLOCK COUNTER FOR RECORD
FBA6                ;*            = 0 SECOND DATA
FBA6                ;*            = 1 FIRST DATA
FBA6                ;*            = 2 FIRST HEADER
FBA6                ;***************************************
FBA6                ;
FBA6                ;TOGGLE WRITE BIT ACCORDING TO LSB
FBA6                ;IN OCHAR
FBA6                ;
FBA6  A5 BD       L1122 LDA OCHAR        ;BIT TO WRITE INTO CARRY
FBA8  4A                LSR A
FBA9  A9 60             LDA #$60        ;ASSUME CARRY CLEAR (SHORT)
FBAB  90 02             BCC L1184       ;CORRECT
FBAD  A9 B0       L1185 LDA #$B0         ;SET LONG
FBAF  A2 00       L1184 LDX #$00         ;SET AND STORE TIME
FBB1  8D 06 DC    L1178 STA D1TBL        ;LO BYTE
FBB4  8E 07 DC          STX D1TBH       ;HI BYTE
FBB7  AD 0D DC          LDA D1ICR       ;CLEAR IRQ
FBBA  A9 19             LDA #$19
FBBC  8D 0F DC          STA D1CRB       ;FORCE LOAD & START TIMER
FBBF  A5 01             LDA $01         ;TOGGLE WRITE BIT
FBC1  49 08             EOR #$08
FBC3  85 01             STA $01
FBC5  29 08             AND #$08        ;LEAVE JUST WRITE BIT
FBC7  60                RTS
FBC8  38          L1181 SEC              ;FLAG PRP FOR END OF BLOCK
FBC9  66 B6             ROR PRP
FBCB  30 3C             BMI L1183       ;ALWAYS
FBCD                    ;
FBCD                ;CALLED AT END OF EACH BYTE TO WRITE
FBCD                ; HHHHHHLLLLLLHHHLLL
FBCD                ;
```

```
LOC   CODE        LINE

FBCD  A5 A8       WRTN   LDA RER          ;CHECK FOR ONE LONG
FBCF  D0 12              BNE L1191
FBD1  A9 10              LDA #$10        ;WRITE LONG BIT
FBD3  A2 01              LDX #$01
FBD5  20 B1 FB           JSR L1178
FBD8  D0 2F              BNE L1183
FBDA  E6 A8              INC RER
FBDC  A5 B6              LDA PRP         ;END OF BLOCK?
FBDE  10 29              BPL L1183       ;NO, CONTINUE
FBE0  4C 57 FC           JMP L1194       ;YES, FINISH OFF
FBE3                     ;
FBE3  A5 A9       L1191  LDA REZ          ;CHECK FOR A ONE BIT
FBE5  D0 09              BNE L1180
FBE7  20 AD FB           JSR L1185
FBEA  D0 1D              BNE L1183
FBEC  E6 A9              INC REZ
FBEE  D0 19              BNE L1183
FBF0                     ;
FBF0  20 A6 FB    L1180  JSR L1122        ;WRITE
FBF3  D0 14              BNE L1183       ;ON BIT LOW, EXIT
FBF5  A5 A4              LDA FIRT        ;FIRST OF DIPOLE?
FBF7  49 01              EOR #$01
FBF9  85 A4              STA FIRT
FBFB  F0 0F              BEQ L1179       ;DIPOLE DONE
FBFD  A5 BD              LDA OCHAR       ;FLIPS BIT FOR COMPLEMENTARY
FBFF  49 01              EOR #$01
FC01  85 BD              STA OCHAR
FC03  29 01              AND #$01        ;TOGGLE PARITY
FC05  45 9B              EOR PRTY
FC07  85 9B              STA PRTY
FC09  4C BC FE    L1183  JMP L1228        ;RESTORE REGS AND RTI
FC0C                     ;
FC0C  46 BD       L1179  LSR OCHAR        ;NEXT BIT
FC0E  C6 A3              DEC PCNTR       ;DEC COUNTER FOR # BITS
FC10  A5 A3              LDA PCNTR       ;8 BITS SENT?
FC12  F0 3A              BEQ L1190       ;YES, DO PARITY
FC14  10 F3              BPL L1183       ;NO, SEND REST
FC16                     ;
FC16  20 97 FB    L1186  JSR L1079        ;CLEAN UP COUNTERS
FC19  58                 CLI             ;ALLOW INTERRUPTS TO NEST
FC1A  A5 A5              LDA CNTDN       ;WRITING HEADER COUNTER?
FC1C  F0 12              BEQ L1189       ;NO
FC1E  A2 00              LDX #$00        ;WRITE HEADER COUNTERS
FC20  86 D7              STX DATA        ;CLEAR BCC
FC22  C6 A5              DEC CNTDN
FC24  A6 BE              LDX FSBLK       ;FIRST BLOCK HEADER?
FC26  E0 02              CPX #$02
FC28  D0 02              BNE L1196       ;NO
FC2A  09 80              ORA #$80        ;YES, MARK 1ST BLOCK HEADER
FC2C  85 BD       L1196  STA OCHAR        ;WRITE CHARS IN HEADER
FC2E  D0 D9              BNE L1183
FC30  20 D1 FC    L1189  JSR L1193        ;ADDR=END?
FC33  90 0A              BCC L1188       ;NOT YET
FC35  D0 91              BNE L1181       ;MARK END
FC37  E6 AD              INC SAH
FC39  A5 D7              LDA DATA        ;WRITE BCC
FC3B  85 BD              STA OCHAR
FC3D  B0 CA              BCS L1183       ;ALWAYS
FC3F                     ;
FC3F  A0 00       L1188  LDY #$00         ;NEXT CHAR
FC41  B1 AC              LDA (SAL),Y
```

```
LOC   CODE        LINE

FC43  85 BD               STA OCHAR       ;STORE IN OUTPUT CHAR
FC45  45 D7               EOR DATA        ;UPDATE BCC
FC47  85 D7               STA DATA
FC49  20 DB FC            JSR L1080       ;BUMP ADDRESS
FC4C  D0 BB               BNE L1183       ;ALWAYS
FC4E                ;
FC4E  A5 9B         L1190 LDA PRTY         ;PARITY INTO OCHAR
FC50  49 01               EOR #$01
FC52  85 BD               STA OCHAR       ; FOR NEXT BIT
FC54  4C BC FE      L1187 JMP L1228        ;RESTORE REGS AND RTI
FC57                ;
FC57  C6 BE         L1194 DEC FSBLK        ;END?
FC59  D0 03               BNE L1182       ;BLOCK ONLY
FC5B  20 CA FC            JSR L1121       ;WRITE SO TURN OFF MOTOR
FC5E  A9 50         L1182 LDA #$50         ;PUT 80 CASSETTE
FC60  85 A7               STA SHCNL       ; SYNCS AT END
FC62  A2 08               LDX #$08
FC64  78                  SEI
FC65  20 BD FC            JSR L1195       ;SET VECTOR TO WRITE ZEROS
FC68  D0 EA               BNE L1187       ;ALWAYS
FC6A                ;
FC6A  A9 78         WRTZ  LDA #$78         ;WRITE LEADING ZEROS
FC6C  20 AF FB            JSR L1184       ; FOR SYNC
FC6F  D0 E3               BNE L1187
FC71  C6 A7               DEC SHCNL       ;DONE WITH LOW SYNC?
FC73  D0 DF               BNE L1187       ;NO
FC75  20 97 FB            JSR L1079       ;YES, CLEAN UP COUNTERS
FC78  C6 AB               DEC SHCNH       ;DONE WITH SYNC?
FC7A  10 D8               BPL L1187       ;NO
FC7C  A2 0A               LDX #$0A        ;YES, VECTOR FOR DATA
FC7E  20 BD FC            JSR L1195
FC81  58                  CLI
FC82  E6 AB               INC SHCNH       ;ZERO SHCNH
FC84  A5 BE               LDA FSBLK       ;DONE?
FC86  F0 30               BEQ L1198       ;YES, SYSTEM RESTORE
FC88  20 8E FB            JSR L1174
FC8B  A2 09               LDX #$09        ;SETUP FOR HEADER COUNT
FC8D  86 A5               STX CNTDN
FC8F  86 B6               STX PRP
FC91  D0 83               BNE L1186       ;ALWAYS
FC93                ;
FC93  08            L1192 PHP              ;CLEAN UP IRQ AND
FC94  78                  SEI             ; RESTORE PIA'S
FC95  AD 11 D0            LDA VICREG+17   ;RESTORE SCREEN
FC98  09 10               ORA #$10
FC9A  8D 11 D0            STA VICREG+17
FC9D  20 CA FC            JSR L1121       ;TURN OFF MOTOR
FCA0  A9 7F               LDA #$7F        ;CLEAR INTERRUPTS
FCA2  8D 0D DC            STA D1ICR
FCA5  20 DD FD            JSR L1202       ;RESTORE KEYBOARD IRQ
FCA8  AD A0 02            LDA IRQTMP+1    ;RESTORE KEYBOARD INTERRUPT VECTOR
FCAB  F0 09               BEQ L1127       ;NO IRQ
FCAD  8D 15 03            STA CINV+1
FCB0  AD 9F 02            LDA IRQTMP
FCB3  8D 14 03            STA CINV
FCB6  28            L1127 PLP
FCB7  60                  RTS
FCB8                ;
FCB8  20 93 FC      L1198 JSR L1192        ;RESTORE SYSTEM IRQ
FCBB  F0 97               BEQ L1187       ;CAME FOR TAPE IRQ SO RTI
FCBD                ;
```

```
LOC   CODE        LINE

FCBD              ;*************************************
FCBD              ;* SUBROUTINE TO CHANGE IRQ VECTORS.
FCBD              ;* ON ENTRY, .X = 8 WRITE ZEROS TO TAPE
FCBD              ;*              = 10 WRITE DATA TO TAPE
FCBD              ;*              = 12 RESTORE TO KEYSCAN
FCBD              ;*              = 14 READ DATA FROM TAPE
FCBD              ;*************************************
FCBD              ;
FCBD  BD 93 FD    L1195  LDA $FD93,X     ;MOVE IRQ VECTORS
FCC0  8D 14 03           STA CINV        ; TO VECTOR TABLE
FCC3  BD 94 FD           LDA $FD94,X
FCC6  8D 15 03           STA CINV+1
FCC9  60                 RTS
FCCA              ;
FCCA  A5 01       L1121  LDA $01         ;TURN OFF CASSETTE MOTOR
FCCC  09 20              ORA #$20
FCCE  85 01              STA $01
FCD0  60                 RTS
FCD1              ;
FCD1              ;*************************************
FCD1              ;* COMPARE START AND END OF LOAD/SAVE
FCD1              ;* ADDRESSES. SUBROUTINE CALLED BY
FCD1              ;* TAPE READ, SAVE, TAPE WRITE
FCD1              ;*************************************
FCD1              ;
FCD1  38          L1193  SEC
FCD2  A5 AC              LDA SAL
FCD4  E5 AE              SBC EAL
FCD6  A5 AD              LDA SAH
FCD8  E5 AF              SBC EAH
FCDA  60                 RTS
FCDB              ;
FCDB              ;BUMP ADDRESS POINTER SAL
FCDB              ;
FCDB  E6 AC       L1080  INC SAL
FCDD  D0 02              BNE L1083
FCDF  E6 AD              INC SAH
FCE1  60          L1083  RTS
FCE2                     .END
FCE2                     .LIB K22
FCE2              ;
FCE2              ;*************************************
FCE2              ;* START -- SYSTEM RESET.
FCE2              ;*      THIS ROUTINE IS USED WHEN THE
FCE2              ;* MACHINE IS SWITCHED ON. IT TESTS FOR
FCE2              ;* A CARTRIDGE IN PLACE, IF SO JUMPS
FCE2              ;* TO ($8000) OTHERWISE INITIALISES
FCE2              ;* BASIC ($A000).
FCE2              ;* FOR CARTRIDGE TO BE IN PLACE, LOCS
FCE2              ;* $8004-$8008 MUST BE 'CBM80' AND CBM
FCE2              ;* HAVE THE HIGH BITS SET.
FCE2              ;*************************************
FCE2              ;
FCE2  A2 FF       START  LDX #$FF
FCE4  78                 SEI             ;DISABLE IRQS
FCE5  9A                 TXS             ;RESET STACK
FCE6  D8                 CLD
FCE7  20 02 FD           JSR L1201       ;CHECK FOR 8ROM
FCEA  D0 03              BNE L1199       ;NO
FCEC  6C 00 80           JMP ($8000)     ;YES, INITIALISE
FCEF  8E 16 D0    L1199  STX VICREG+22    ;SHRINK SCREEN
```

```
LOC   CODE        LINE

FCF2  20 A3 FD            JSR L1211       ;INIT I/O
FCF5  20 50 FD            JSR L1206       ;SET TOP MEMORY
FCF8  20 15 FD            JSR L1206       ;INIT KERNAL VECTORS
FCFB  20 5B FF            JSR L1271
FCFE  58                  CLI             ;ENABLE IRQS
FCFF  6C 00 A0            JMP ($A000)     ;INIT BASIC
FD02                ;
FD02                ;TEST FOR CARTRIDGE
FD02                ;
FD02  A2 05         L1201 LDX #$05         ;LENGTH OF COMPARE
FD04  BD 0F FD      L1200 LDA TBL80-1,X    ;GET MASK
FD07  DD 03 80            CMP $8003,X     ;COMPARE TO BROM
FD0A  D0 03               BNE L1207       ;NO MATCH
FD0C  CA                  DEX             ;MATCH, DO NEXT
FD0D  D0 F5               BNE L1200
FD0F  60            L1207 RTS             ;DONE
FD10  C3            TBL80 .BYT $C3,$C2,$CD,'80';..CBM80..
FD11  C2
FD12  CD
FD13  38 30
FD15                ;
FD15                ;SET KERNAL INDIRECTS & VECTORS
FD15                ;
FD15  A2 30         L1206 LDX #<VECTSS
FD17  A0 FD               LDY #>VECTSS
FD19  18                  CLC
FD1A                ;
FD1A  86 C3         L1204 STX TMP2
FD1C  84 C4               STY TMP2+1
FD1E  A0 1F               LDY #$1F        ;LENGTH OF TABLE
FD20  B9 14 03      L1233 LDA CINV,Y      ;GET FROM STORE
FD23  B0 02               BCS L1209       ;STORAGE TO USER
FD25  B1 C3               LDA (TMP2),Y    ;USER TO STORAGE
FD27  91 C3         L1209 STA (TMP2),Y    ;PUT IN USER
FD29  99 14 03            STA CINV,Y      ;PUT IN STORAGE
FD2C  88                  DEY
FD2D  10 F1               BPL L1233
FD2F  60                  RTS
FD30                ;
FD30                ;****************************************
FD30                ;* TABLE OF KERNAL INDIRECTS
FD30                ;****************************************
FD30                ;
FD30  31 EA         VECTSS .WOR KEY
FD32  66 FE                .WOR TIMB
FD34  47 FE                .WOR NNMI
FD36  4A F3                .WOR NOPEN
FD38  91 F2                .WOR NCLOSE
FD3A  0E F2                .WOR NCHKIN
FD3C  50 F2                .WOR NCKOUT
FD3E  33 F3                .WOR NCLRCH
FD40  57 F1                .WOR NBASIN
FD42  CA F1                .WOR NBSOUT
FD44  ED F6                .WOR NSTOP
FD46  3E F1                .WOR NGETIN
FD48  2F F3                .WOR NCLALL
FD4A  66 FE                .WOR TIMB
FD4C  A5 F4                .WOR NLOAD
FD4E  ED F5                .WOR NSAVE
FD50                ;
FD50                ;MEMORY SIZE CHECK & SET
```

```
LOC   CODE        LINE

FD50              ;
FD50  A9 00       L1208  LDA #$00         ;ZERO LO MEM
FD52  A8                 TAY             ;START AT 00
FD53  99 02 00    L1203  STA $0002,Y      ;ZERO PAGE
FD56  99 00 02           STA BUF,Y        ;USER BUFFERS AND VARS
FD59  99 00 03           STA $0300,Y      ;SYSTEM SPACE & USER SPACE
FD5C  C8                 INY
FD5D  D0 F4              BNE L1203
FD5F              ;
FD5F              ;ALLOCATE TAPE BUFFER
FD5F              ;
FD5F  A2 3C              LDX #<TBUFFR
FD61  A0 03              LDY #>TBUFFR
FD63  86 B2              STX TAPE1
FD65  84 B3              STY TAPE1+1
FD67              ;
FD67              ;SET TOP OF MEMORY
FD67              ;
FD67  A8                 TAY
FD68  A9 03              LDA #$03
FD6A  85 C2              STA STAH
FD6C  E6 C2       L1210  INC STAH         ;START AT $0400
FD6E  B1 C1       L1213  LDA (STAL),Y     ;STORE LOC
FD70  AA                 TAX
FD71  A9 55              LDA #$55         ;TEST %01010101
FD73  91 C1              STA (STAL),Y
FD75  D1 C1              CMP (STAL),Y
FD77  D0 0F              BNE L1212        ;NOT RAM
FD79  2A                 ROL A            ;TEST %10101010
FD7A  91 C1              STA (STAL),Y
FD7C  D1 C1              CMP (STAL),Y
FD7E  D0 08              BNE L1212        ;NOT RAM
FD80  8A                 TXA              ;RESTORE LOC
FD81  91 C1              STA (STAL),Y
FD83  C8                 INY
FD84  D0 E8              BNE L1213
FD86  F0 E4              BEQ L1210        ;AND AGAIN
FD88  98          L1212  TYA              ;FOUND TOP OF MEM
FD89  AA                 TAX
FD8A  A4 C2              LDY STAH
FD8C  18                 CLC
FD8D  20 2D FE           JSR L991         ;SET TOP OF MEM
FD90  A9 08              LDA #$08
FD92  8D 82 02           STA MEMSTR+1     ;SET BOTTOM
FD95  A9 04              LDA #$04
FD97  8D 88 02           STA HIBASE       ;SET SCREEN
FD9A  60                 RTS
FD9B  6A FC              .WOR WRTZ
FD9D  CD FB              .WOR WRTN
FD9F  31 EA              .WOR KEY
FDA1  2C F9              .WOR L1130
FDA3              ;
FDA3              ;INITIALISE I/O DEVICES
FDA3              ;
FDA3  A9 7F       L1211  LDA #$7F         ;KILL IERS
FDA5  8D 0D DC           STA D1ICR
FDA8  8D 0D DD           STA D2ICR
FDAB  8D 00 DC           STA D1DPA
FDAE  A9 08              LDA #$08
FDB0  8D 0E DC           STA D1CRA
FDB3  8D 0E DD           STA D2CRA
```

```
LOC   CODE        LINE

FDB6  8D 0F DC            STA D1CRB
FDB9  8D 0F DD            STA D2CRB
FDBC  A2 00               LDX #$00
FDBE  8E 03 DC            STX D1DDRB      ;KEYBOARD COLS OUTPUT
FDC1  8E 03 DD            STX D2DDRB      ;NO RS-232
FDC4  8E 18 D4            STX SIDREG+24   ;NO VOLUME
FDC7  CA                  DEX
FDC8  8E 02 DC            STX D1DDRA      ;KEYBOARD ROWS INPUT
FDCB  A9 07               LDA #$07
FDCD  8D 00 DD            STA D2DPA
FDD0  A9 3F               LDA #$3F
FDD2  8D 02 DD            STA D2DDRA
FDD5  A9 E7               LDA #$E7
FDD7  85 01               STA $01
FDD9  A9 2F               LDA #$2F
FDDB  85 00               STA $00
FDDD  AD A6 02    L1202   LDA $02A6
FDE0  F0 0A               BEQ L1197
FDE2  A9 25               LDA #$25
FDE4  8D 04 DC            STA D1TAL
FDE7  A9 40               LDA #$40
FDE9  4C F3 FD            JMP L1214
FDEC  A9 95       L1197   LDA #$95
FDEE  8D 04 DC            STA D1TAL
FDF1  A9 42               LDA #$42
FDF3  8D 05 DC    L1214   STA D1TAH
FDF6  4C 6E FF            JMP L1232
FDF9                ;
FDF9                ;***************************************
FDF9                ;* SETNAM.
FDF9                ;*      THIS ROUTINE STORES THE FILENAME
FDF9                ;* LENGTH AND THE POINTER TO THE NAME FROM
FDF9                ;* .A(NAME LENGTH) AND .X, .Y FILE ADDRESS
FDF9                ;* (LOW/HIGH).
FDF9                ;***************************************
FDF9                ;
FDF9  85 B7       L1215   STA FNLEN
FDFB  86 BB               STX FNADR
FDFD  84 BC               STY FNADR+1
FDFF  60                  RTS
FE00                ;
FE00                ;***************************************
FE00                ;* SETLFS.
FE00                ;*      STORE DEVICE NUMBER (.X), LOGICAL
FE00                ;* FILE NUMBER (.A), AND SECONDARY
FE00                ;* ADDRESS (.Y) FOR OPEN, LOAD, AND SAVE.
FE00                ;***************************************
FE00                ;
FE00  85 B8       L1240   STA LA
FE02  86 BA               STX FA
FE04  84 B9               STY SA
FE06  60                  RTS
FE07                ;
FE07  A5 BA       L1239   LDA FA           ;WHICH DEVICES TO READ
FE09  C9 02               CMP #$02        ;RS-232?
FE0B  D0 0D               BNE L1234       ;NO
FE0D  AD 97 02            LDA RSSTAT      ;YES, RS-232 UP
FE10  48                  PHA
FE11  A9 00               LDA #$00        ;CLEAR RS-232 STATUS
FE13  8D 97 02            STA RSSTAT
FE16  68                  PLA
```

```
LOC   CODE        LINE


FE17  60                RTS
FE18  85 9D       L1238  STA MSGFLG
FE1A  A5 90       L1234  LDA STATUS
FE1C  05 90       L1217  ORA STATUS
FE1E  85 90              STA STATUS
FE20  60                RTS
FE21              ;
FE21  8D 85 02    L857   STA TIMOUT
FE24  60                RTS
FE25              ;
FE25              ;****************************************
FE25              ;* READ/SET TOP OF MEMORY.
FE25              ;*     THIS ROUTINE IS USED TO READ OR SET
FE25              ;* THE TOP OF MEMORY POINTERS. TO READ,
FE25              ;* THE CARRY SHOULD BE SET AND THE TOP
FE25              ;* OF MEMORY IS RETURNED IN .X(LO),
FE25              ;* .Y(HI). TO SET, CARRY SHOULD BE CLEAR
FE25              ;* AND .X HOLDS LO AND .Y HOLDS HI.
FE25              ;****************************************
FE25              ;
FE25  90 06       L1237  BCC L991         ;C CLEAR, SET
FE27  AE 83 02    L1235  LDX MEMSIZ       ;ELSE READ
FE2A  AC 84 02           LDY MEMSIZ+1
FE2D  8E 83 02    L991   STX MEMSIZ       ;SET
FE30  8C 84 02           STY MEMSIZ+1
FE33  60                RTS
FE34              ;
FE34              ;****************************************
FE34              ;* READ/SET BOTTOM OF MEMORY.
FE34              ;*    THIS ROUTINE READS OR SETS THE
FE34              ;* BOTTOM OF MEMORY IN THE SAME WAY AS
FE34              ;* THE READ/SET TOP OF MEMORY.
FE34              ;****************************************
FE34              ;
FE34  90 06       L1044  BCC L1236        ;C CLEAR SET
FE36  AE 81 02           LDX MEMSTR     ;ELSE READ
FE39  AC 82 02           LDY MEMSTR+1
FE3C  8E 81 02    L1236  STX MEMSTR       ;SET
FE3F  8C 82 02           STY MEMSTR+1
FE42  60                RTS
FE43                     .END
FE43                     .LIB K23
FE43              ;
FE43              ;****************************************
FE43              ;*NMI HANDLER ROUTINE
FE43              ;****************************************
FE43              ;
FE43  78          NMI    SEI              ;DISABLE IRQ
FE44  6C 18 03           JMP (NMINV)
FE47  48          NNMI   PHA              ;SAVE 6502 REGS
FE48  8A                 TXA
FE49  48                 PHA
FE4A  98                 TYA
FE4B  48                 PHA
FE4C  A9 7F              LDA #$7F
FE4E  8D 0D DD           STA D2ICR
FE51  AC 0D DD           LDY D2ICR       ;REAL NMI?
FE54  30 1C              BMI L1220       ;NO
FE56  20 02 FD           JSR L1201       ;CHECK FOR 8ROM
FE59  D0 03              BNE L1218       ;NO
FE5B  6C 02 80           JMP ($8002)     ;YES
```

```
LOC   CODE        LINE

FE5E              ;
FE5E              ;STOP KEY DOWN?
FE5E              ;
FE5E  20 BC F6    L1218  JSR L1092        ;BUMP CLOCK
FE61  20 E1 FF           JSR L309        ;STOP KEY?
FE64  D0 0C              BNE L1220       ;NO STOP KEY
FE66              ;
FE66              ;WHERE SYSTEM GOES ON BRK
FE66              ;
FE66  20 15 FD    TIMB   JSR L1206        ;RESTORE SYSTEM INDIRECTS
FE69  20 A3 FD           JSR L1211       ;RESTORE BASIC I/O
FE6C  20 18 E5           JSR L678        ;RESTORE BASIC SCREEN
FE6F  6C 02 A0           JMP ($A002)     ;BASIC WARM RESTART
FE72              ;
FE72              ;HANDLE RS-232 NMI
FE72              ;
FE72  98          L1220  TYA
FE73  2D A1 02           AND $02A1       ;SHOW ONLY ENABLES
FE76  AA                 TAX             ;SAVE IN .X FOR LATER
FE77  29 01              AND #$01
FE79  F0 28              BEQ L1223
FE7B  AD 00 DD           LDA D2DPA
FE7E  29 FB              AND #$FB
FE80  05 B5              ORA NXTBIT
FE82  8D 00 DD           STA D2DPA
FE85  AD A1 02           LDA $02A1
FE88  8C 0D DD           STA D2ICR
FE8B  8A                 TYA
FE8C  29 12              AND #$12
FE8E  F0 0D              BEQ L1224
FE90  29 02              AND #$02
FE92  F0 06              BEQ L1219
FE94  20 D6 FE           JSR L1147       ;RS-232 IN
FE97  4C 9D FE           JMP L1224
FE9A  20 07 FF    L1219  JSR L1225        ;RS-232 OUT
FE9D  20 BB EE    L1224  JSR L877         ;RS-232 SEND
FEA0  4C B6 FE           JMP L1229       ;EXIT NMI
FEA3              ;
FEA3  8A          L1223  TXA
FEA4  29 02              AND #$02
FEA6  F0 06              BEQ L1222
FEA8  20 D6 FE           JSR L1147       ;RS-232 IN
FEAB  4C B6 FE           JMP L1229       ;EXIT NMI
FEAE  8A          L1222  TXA
FEAF  29 10              AND #$10
FEB1  F0 03              BEQ L1229
FEB3  20 07 FF           JSR L1225       ;RS-232 OUT
FEB6  AD A1 02    L1229  LDA $02A1        ;EXIT NMI, CLEAR
FEB9  8D 0D DD           STA D2ICR       ; NMI FLAG
FEBC  68          L1228  PLA             ;RESTORE REGISTERS
FEBD  A8                 TAY
FEBE  68                 PLA
FEBF  AA                 TAX
FEC0  68                 PLA
FEC1  40                 RTI
FEC2              ;
FEC2              ;BAUD RATE TABLE
FEC2              ;
FEC2  C1 27              .WOR $27C1
FEC4  3E 1A              .WOR $1A3E
FEC6  C5 11              .WOR $11C5
```

```
LOC   CODE        LINE

FEC8  74 0E               .WOR $0E74
FECA  ED 0C               .WOR $0CED
FECC  45 06               .WOR $0645
FECE  F0 02               .WOR $02F0
FED0  46 01               .WOR $0146
FED2  B8 00               .WOR $00B8
FED4  71 00               .WOR $0071
FED6                ;
FED6                ;RS-232 IN
FED6                ;
FED6  AD 01 DD      L1147  LDA D2DPB
FED9  29 01                AND #$01
FEDB  85 A7                STA INBIT
FEDD  AD 06 DD             LDA D2TBL
FEE0  E9 1C                SBC #$1C
FEE2  6D 99 02             ADC BAUDOF
FEE5  8D 06 DD             STA D2TBL
FEE8  AD 07 DD             LDA D2TBH
FEEB  6D 9A 02             ADC BAUDOF+1
FEEE  8D 07 DD             STA D2TBH
FEF1  A9 11                LDA #$11
FEF3  8D 0F DD             STA D2CRB
FEF6  AD A1 02             LDA $02A1
FEF9  8D 0D DD             STA D2ICR
FEFC  A9 FF                LDA #$FF
FEFE  8D 06 DD             STA D2TBL
FF01  8D 07 DD             STA D2TBH
FF04  4C 59 EF             JMP L893        ;RECEIVE RS-232
FF07                ;
FF07                ;RS-232 OUT
FF07                ;
FF07  AD 95 02      L1225  LDA M26AJB
FF0A  8D 06 DD             STA D2TBL
FF0D  AD 96 02             LDA M26AJB+1
FF10  8D 07 DD             STA D2TBH
FF13  A9 11                LDA #$11
FF15  8D 0F DD             STA D2CRB
FF18  A9 12                LDA #$12
FF1A  4D A1 02             EOR $02A1
FF1D  8D A1 02             STA $02A1
FF20  A9 FF                LDA #$FF
FF22  8D 06 DD             STA D2TBL
FF25  8D 07 DD             STA D2TBH
FF28  AE 98 02             LDX BITNUM
FF2B  86 A8                STX BITCI
FF2D  60                   RTS
FF2E                ;
FF2E  AA            L1226  TAX
FF2F  AD 96 02             LDA M26AJB+1
FF32  2A                   ROL A
FF33  A8                   TAY
FF34  8A                   TXA
FF35  69 C8                ADC #$C8
FF37  8D 99 02             STA BAUDOF
FF3A  98                   TYA
FF3B  69 00                ADC #$00
FF3D  8D 9A 02             STA BAUDOF+1
FF40  60                   RTS
FF41  EA                   NOP
FF42  EA                   NOP
FF43                       .END
```

```
LOC   CODE        LINE

FF43                    .LIB K24
FF43                ;
FF43                ;INTERRUPT FROM TAPE
FF43                ;
FF43  08            L1041  PHP
FF44  68                   PLA
FF45  29 EF                AND #$EF        ;CLEAR BREAK FLAG
FF47  48                   PHA
FF48                ;
FF48                ;IRQ INTERRUPT
FF48                ;
FF48  48            PULS   PHA              ;SAVE REGISTERS
FF49  8A                   TXA
FF4A  48                   PHA
FF4B  98                   TYA
FF4C  48                   PHA
FF4D  BA                   TSX
FF4E  BD 04 01             LDA BAD+4,X     ;GET BREAK FLAG
FF51  29 10                AND #$10        ;SET?
FF53  F0 03                BEQ L1131       ;NO
FF55  6C 16 03             JMP (CBINV)     ;YES
FF58  6C 14 03      L1131  JMP (CINV)
FF5B                ;
FF5B                ;INITIALIZE VIC CHIP
FF5B                ;
FF5B  20 18 E5      L1231  JSR L678
FF5E  AD 12 D0      L1205  LDA VICREG+18
FF61  D0 FB                BNE L1205
FF63  AD 19 D0             LDA VICREG+25
FF66  29 01                AND #$01
FF68  8D A6 02             STA $02A6
FF6B  4C DD FD             JMP L1202
FF6E                ;
FF6E  A9 81         L1232  LDA #$81
FF70  8D 0D DC             STA D1ICR
FF73  AD 0E DC             LDA D1CRA
FF76  29 80                AND #$80
FF78  09 11                ORA #$11
FF7A  8D 0E DC             STA D1CRA
FF7D  4C 8E EE             JMP L843
FF80  00                   BRK
FF81                       .END
FF81                       .LIB K25
FF81                ;
FF81                ;JUMP TABLE FOR OPERATING SYSTEM ROUTINES
FF81                ;
FF81  4C 5B FF             JMP L1231
FF84  4C A3 FD             JMP L1211       ;INITIALIZE I/O
FF87  4C 50 FD             JMP L1208       ;INIT SYSTEM CONSTANTS
FF8A  4C 15 FD             JMP L1206       ;KERNAL RESET
FF8D  4C 1A FD             JMP L1204       ;KERNAL MOVE
FF90  4C 18 FE      L1216  JMP L1238        ;FLAG STATUS
FF93  4C B9 ED             JMP L871        ;SEND LISTEN SA
FF96  4C C7 ED             JMP L860        ;SEND TALK SA
FF99  4C 25 FE      L25    JMP L1237        ;READ/SET TOP MEMORY
FF9C  4C 34 FE      L670   JMP L1044        ;READ/SET BOTTOM MEM
FF9F  4C 87 EA             JMP L814        ;READ KEYBOARD
FFA2  4C 21 FE             JMP L857        ;SET TIMEOUT
FFA5  4C 13 EE             JMP L865        ;RECEIVE FROM SERIAL
FFA8  4C DD ED             JMP L861        ;SEND TO SERIAL
FFAB  4C EF ED             JMP L863        ;SEND 'UNTALK'
```

```
LOC   CODE        LINE

FFAE  4C FE ED            JMP L1006       ;SEND 'UNLISTEN'
FFB1  4C 0C ED            JMP L966        ;SEND 'LISTEN'
FFB4  4C 09 ED            JMP L836        ;SEND 'TALK'
FFB7  4C 07 FE      L669  JMP L1239        ;GET STATUS
FFBA  4C 00 FE      L217  JMP L1240        ;SET LA,FA,SA
FFBD  4C F9 FD      L641  JMP L1215        ;SET FNLEN,FNADR
FFC0  6C 1A 03      L640  JMP (IOPEN)      ;OPEN LOGICAL FILE
FFC3  6C 1C 03      L638  JMP (ICLOSE)     ;CLOSE LOGICAL FILE
FFC6  6C 1E 03      L639  JMP (ICHKIN)     ;OPEN CHANNEL IN
FFC9  6C 20 03      L624  JMP (ICKOUT)     ;OPEN CHANNEL OUT
FFCC  6C 22 03      L674  JMP (ICLRCH)     ;CLOSE I/O CHANNEL
FFCF  6C 24 03      L16   JMP (IBASIN)     ;INPUT FROM CHANNEL
FFD2  6C 26 03      L622  JMP (IBSOUT)     ;OUTPUT TO CHANNEL
FFD5  4C 9E F4      L620  JMP L990         ;LOAD A FILE
FFD8  4C DD F5      L628  JMP L1072        ;SAVE A FILE
FFDB  4C E4 F6      L627  JMP L1244        ;SET TIME
FFDE  4C DD F6      L165  JMP L1096        ;READ TIME
FFE1  6C 28 03      L309  JMP (ISTOP)      ;SCAN STOP KEY
FFE4  6C 2A 03      L115  JMP (IGETIN)     ;GET CHAR FROM Q
FFE7  6C 2C 03      L625  JMP (ICLALL)     ;CLOSE ALL FILES
FFEA  4C 9B F6      L71   JMP L1078        ;INCREMENT CLOCK
FFED  4C 05 E5            JMP L1248       ;GET SCREEN SIZE
FFF0  4C 0A E5      L808  JMP L1246        ;READ/SET ROW/COLUMN
FFF3  4C 00 E5      L192  JMP L676         ;GET I/O BASE
FFF6  52                  .BYT $52,$52,$42,$59
FFF7  52
FFF8  42
FFF9  59
FFFA  43 FE               .WOR NMI
FFFC  E2 FC               .WOR START      ;COLD START INDIRECT
FFFE  48 FF               .WOR PULS       ;INTERRUPT HANDLER
0000                      .END
```

Symbol Table

This section is provided by the Commodore Assembler and lists all variable and label names followed by their address in memory or values for constants.

```
SYMBOL VALUE
 ABS      BC58    ADRAY1   0003    ADRAY2   0005    ARGEXP   0069
 ARGHO    006A    ARGSGN   006E    ARISGN   006F    ARYTAB   002F
 ASC      B78B    ATN      E30E    AUTODN   0292    BAD      0100
 BASTMP   004B    BASZPT   00FF    BAUDOF   0299    BDF      0002
 BDFH     0004    BITCI    00A8    BITNUM   0298    BITS     0068
 BITTS    00B4    BLF      0001    BLNCT    00CD    BLNON    00CF
 BLNSW    00CC    BLUE     0006    BSOUR    0095    BSOUR1   00A4
 BUF      0200    BUFPT    00A6    BUFSZ    00C0    C3PO     0094
 CAS1     00C0    CBINV    0316    CBMCOL   D800    CBMSCN   0400
 CHARAC   0007    CHR      B6EC    CHRGET   0073    CHRGOT   0079
 CINV     0314    CKERR    0020    CLOSE    E1C7    CLR      A65E
 CMPO     00B0    CNTDN    00A5    COLD     E394    COLM     DC00
 COLOR    0286    COLTAB   E8DA    COMP     B016    CONT     A857
 CONTRL   EC78    COS      E264    COUNT    00A5    COUNTB   000B
 CR       000D    CRSW     00D0    CURLIN   0039    D1CRA    DC0E
 D1CRB    DC0F    D1DDRA   DC02    D1DDRB   DC03    D1DPA    DC00
 D1DPB    DC01    D1ICR    DC0D    D1IODB   DC0C    D1TAH    DC05
 D1TAL    DC04    D1TBH    DC07    D1TBL    DC06    D1TOD1   DC08
 D1TOD2   DC09    D1TOD3   DC0A    D1TOD4   DC0B    D2CRA    DD0E
 D2CRB    DD0F    D2DDRA   DD02    D2DDRB   DD03    D2DPA    DD00
 D2DPB    DD01    D2ICR    DD0D    D2IODB   DD0C    D2TAH    DD05
 D2TAL    DD04    D2TBH    DD07    D2TBL    DD06    D2TOD1   DD08
 D2TOD2   DD09    D2TOD3   DD0A    D2TOD4   DD0B    DATA     00D7
 DATLIN   003F    DATPTR   0041    DATSCN   000F    DEF      B3B3
 DELAY    028C    DFLTN    0099    DFLTO    009A    DIFF     00B5
 DIM      B081    DIMFLG   000C    DPSW     009C    EAH      00AF
 EAL      00AE    EML      E388    ENDCHR   0008    EOT      0005
```

SYMBOL	VALUE						
FA	00BA	FACEXP	0061	FACHO	0062	FACOV	0070
FACSGN	0066	FAND	AFE9	FAT	0263	FBUFPT	0071
FIRT	00A4	FNADR	00BB	FNLEN	00B7	FOR	A742
FORPNT	0049	FRE	B37D	FREKZP	00FB	FREPSC	0035
FRETOP	0033	FSBLK	00BE	FTOI	B1AA	GARBFL	000F
GDBLN	00CE	GDCOL	0287	GET	AB7B	GOSUB	A883
HIBASE	0288	IBASIN	0324	IBSOUT	0326	ICHKIN	031E
ICKOUT	0320	ICLALL	032C	ICLOSE	031C	ICLRCH	0322
IF	A928	IGETIN	032A	ILOAD	0330	INBIT	00A7
INDEX	0022	INDX	00C8	INITNM	E5A8	INPFLG	0011
INPPTR	0043	INPRMT	0013	INPUT	ABBF	INPUTH	ABA5
INSRT	00D8	INTFLG	000E	IOPEN	031A	IRQTMP	029F
ISAVE	0332	ISTOP	0328	KEY	EA31	KEYD	0277
KEYLOG	028F	KEYTAB	00F5	KOUNT	028B	L0	A38A
L1	A3B8	L10	A437	L100	BC2F	L1000	F30F
L1001	F33C	L1002	F31F	L1003	F343	L1004	EE03
L1005	F351	L1006	EDFE	L1007	F359	L1008	F362
L1009	F701	L101	AE43	L1010	F384	L1011	F6FE
L1012	F3D4	L1013	F38B	L1014	F3F6	L1015	F393
L1016	F40F	L1017	F3AC	L1018	F729	L1019	F3C2
L102	A7CE	L1020	F81E	L1021	F3D5	L1022	F5C1
L1023	F3B8	L1024	F7F7	L1025	F3D1	L1026	F707
L1027	F74B	L1028	F3AF	L1029	F841	L103	A80B
L1030	F3D3	L1031	F3FC	L1032	F409	L1033	F406
L1034	F657	L1035	F43A	L1036	F41D	L1037	EF54
L1038	F45C	L1039	F440	L104	A7E1	L1040	F446
L1041	FF43	L1042	F474	L1043	F47D	L1044	FE34
L1045	F4BF	L1046	F4B2	L1047	F539	L1048	F4F0
L1049	F713	L105	A854	L1050	F533	L1051	F4F3
L1052	F5DA	L1053	F51C	L1054	F63A	L1055	F501
L1056	F51E	L1057	F524	L1058	F530	L1059	F5AE
L106	A7EF	L1060	F541	L1061	F549	L1062	F5AF
L1063	F55D	L1064	F56C	L1065	F57D	L1066	F556
L1067	F5A9	L1068	F579	L1069	F864	L107	A7BE
L1070	F5D2	L1071	F5D1	L1072	F5DD	L1073	F12F
L1074	F605	L1075	F5F4	L1076	F65F	L1077	F624
L1078	F69B	L1079	FB97	L108	A82C	L1080	FCDB
L1081	F642	L1082	F63F	L1083	FCE1	L1084	F633
L1085	F659	L1086	F676	L1087	F68F	L1088	F68D
L1089	F86B	L109	A807	L1090	F68E	L1091	F5C7
L1092	F6BC	L1093	F6CC	L1094	F6DC	L1095	F6DA
L1096	F6DD	L1097	F6FB	L1098	F72C	L1099	F76A
L11	A435	L110	A81D	L1100	F757	L1101	F769
L1102	F767	L1103	E4E2	L1104	F7D0	L1105	F7A5
L1106	F7CF	L1107	F7B7	L1108	F7EA	L1109	F86E
L111	AF0D	L1110	F80D	L1111	F80C	L1112	F80B
L1113	F836	L1114	F838	L1115	F8DC	L1116	F82E
L1117	F821	L1118	F875	L1119	F8B5	L112	A80E
L1120	F8E1	L1121	FCCA	L1122	FBA6	L1123	F8BE
L1124	F8B7	L1125	F8D0	L1126	F8E2	L1127	FCB6
L1128	F8FE	L1129	F92A	L113	A8BC	L1130	F92C
L1131	FF58	L1132	F969	L1133	F9BC	L1134	F988
L1135	FA70	L1136	F98B	L1137	F997	L1138	F999
L1139	F993	L114	A82B	L1140	FA18	L1141	F9AC
L1142	F9DE	L1143	F9D5	L1144	F9C9	L1145	FA10
L1146	F8F7	L1147	FED6	L1148	F9F7	L1149	F9E0
L115	FFE4	L1150	F9D2	L1151	FA1F	L1152	FA44
L1153	FA53	L1154	FA60	L1155	FA5D	L1156	FA86
L1157	FAA3	L1158	FA8A	L1159	FA8D	L116	A849
L1160	FACE	L1161	FABA	L1162	FAA9	L1163	FAC0
L1164	FAD6	L1165	FB5C	L1166	FAEB	L1167	FB4A

```
SYMBOL  VALUE
L1168   FB2F    L1169   FB08    L117    A87D    L1170   FB43
L1171   FB3A    L1172   FB48    L1173   FB33    L1174   FB8E
L1175   FB68    L1176   FB72    L1177   FB8B    L1178   FBB1
L1179   FC0C    L118    A84B    L1180   FBF0    L1181   FBC8
L1182   FC5E    L1183   FC09    L1184   FBAF    L1185   FBAD
L1186   FC16    L1187   FC54    L1188   FC3F    L1189   FC30
L119    A862    L1190   FC4E    L1191   FBE3    L1192   FC93
L1193   FCD1    L1194   FC57    L1195   FCBD    L1196   FC2C
L1197   FDEC    L1198   FCB8    L1199   FCEF    L12     A416
L120    A474    L1200   FD04    L1201   FD02    L1202   FDDD
L1203   FD53    L1204   FD1A    L1205   FF5E    L1206   FD15
L1207   FD0F    L1208   FD50    L1209   FD27    L121    A870
L1210   FD6C    L1211   FDA3    L1212   FD88    L1213   FD6E
L1214   FDF3    L1215   FDF9    L1216   FF90    L1217   FE1C
L1218   FE5E    L1219   FE9A    L122    A897    L1220   FE72
L1221   E544    L1222   FEAE    L1223   FEA3    L1224   FE9D
L1225   FF07    L1226   FF2E    L1227   EEC8    L1228   FEBC
L1229   FEB6    L123    A663    L1230   EF6D    L1231   FF5B
L1232   FF6E    L1233   FD20    L1234   FE1A    L1235   FE27
L1236   FE3C    L1237   FE25    L1238   FE18    L1239   FE07
L124    A8A0    L1240   FE00    L1241   F4AF    L1242   F5F1
L1243   F6FA    L1244   F6E4    L1245   F6A7    L1246   E50A
L1247   E513    L1248   E505    L125    A911    L126    A8C0
L127    A8D1    L128    A8E8    L129    A8E3    L13     A421
L130    A8F8    L131    A906    L132    A937    L133    A919
L134    ADA4    L135    A940    L136    A948    L137    A905
L138    A953    L139    B7A1    L14     B52A    L140    A95F
L141    A8EB    L142    A96A    L143    A804    L144    A96B
L145    A957    L146    A9A5    L147    A99F    L148    B090
L149    AD96    L15     A434    L150    A9DA    L151    A9D9
L152    BC23    L153    B1CC    L154    BBD4    L155    AA3D
L156    B6AA    L157    AA27    L158    AA24    L159    BAED
L16     FFCF    L160    BC0F    L161    AA1D    L162    BAF8
L163    AA07    L164    BCAF    L165    FFDE    L166    AA2C
L167    B24A    L168    BD91    L169    AA52    L17     AAE5
L170    AA4B    L171    AA68    L172    AA86    L173    B47D
L174    B688    L175    B6EB    L176    AA90    L177    ABB7
L178    AA9A    L179    E11E    L18     AB47    L180    AAA2
L181    AB28    L182    AAE8    L183    AB0E    L184    AAEE
L185    AB19    L186    AA9D    L187    BDDF    L188    B48D
L189    AB42    L19     AB4D    L190    AAA0    L191    AAE7
L192    FFF3    L193    AAF8    L194    AB0F    L195    B79E
L196    AB62    L197    AB10    L198    AB1E    L199    AACA
L2      A3B0    L20     A469    L200    AB13    L201    AB3B
L202    AB45    L203    E112    L204    AB6B    L205    AB5B
L206    AB5F    L207    AB92    L208    B3AE    L209    ABB5
L21     A68D    L210    E124    L211    AC15    L212    ABCE
L213    ABD6    L214    AEC6    L215    AC03    L216    ABF9
L217    FFBA    L218    A8FB    L219    AC0F    L22     AB21
L220    ABEA    L221    AC0D    L222    AC45    L223    AC4A
L224    E194    L225    AE51    L226    ACD1    L227    AC4D
L228    AC91    L229    AC71    L23     A480    L230    AC72
L231    AC7D    L232    AC89    L233    B497    L234    B7EB
L235    A9ED    L236    AC9D    L237    BCF7    L238    A9D6
L239    ACB8    L24     BDCD    L240    AB57    L241    ACEA
L242    AEFF    L243    AC41    L244    ACDF    L245    AD35
L246    ACFB    L247    AD24    L248    AD27    L249    AD32
L25     FF99    L250    AD75    L251    B86A    L252    BC92
L253    AD8A    L254    AD78    L255    AD99    L256    AD9E
L257    AD97    L258    ADA9    L259    AE8A    L26     A562
L260    ADE8    L261    AE33    L262    ADD7    L263    AE11
```

SYMBOL	VALUE						
L264	AE5B	L265	ADF0	L266	B65D	L267	AE64
L268	AE30	L269	AE20	L27	A49C	L270	AE5D
L271	AE80	L272	AE19	L273	ADF9	L274	ADFA
L275	AE38	L276	ADB8	L277	AE83	L278	AE66
L279	AE9A	L28	A4D7	L280	B11C	L281	AEAD
L282	AF5C	L283	AEBD	L284	AE92	L285	AF0F
L286	AE8F	L287	AEE3	L288	AECC	L289	AEEA
L29	A582	L290	AF14	L291	B3A2	L292	AEF1
L293	B418	L294	AEF7	L295	AFD1	L296	AEFD
L297	AE07	L298	AF28	L299	AF6E	L3	A3B7
L30	A7ED	L300	AF27	L301	AF5D	L302	AF92
L303	BE6A	L304	B475	L305	AF84	L306	AFA7
L307	AFA0	L308	BC5B	L309	FFE1	L31	A971
L310	BC44	L311	AFD6	L312	AD90	L313	B02E
L314	BBFE	L315	B056	L316	BC5D	L317	B066
L318	B6D5	L319	B05B	L32	A617	L320	B072
L321	B07B	L322	B061	L323	B07E	L324	B092
L325	B08B	L326	B0AF	L327	B0B0	L328	B0C4
L329	B0BA	L33	A508	L330	B0D4	L331	B0DB
L332	B0E7	L333	B09F	L334	B0EF	L335	B1DB
L336	B109	L337	B123	L338	B113	L339	B18F
L34	A4DF	L340	B0FB	L341	B0F1	L342	B11D
L343	B138	L344	B143	L345	B128	L346	B159
L347	B13B	L348	B185	L349	B194	L35	A4ED
L350	B1B2	L351	B1CE	L352	B1D1	L353	B1B8
L354	B21C	L355	AEFA	L356	B237	L357	B274
L358	B245	L359	B261	L36	A660	L360	B228
L361	B24D	L362	B1A0	L363	B240	L364	B2F2
L365	B27D	L366	B286	L367	B2B9	L368	B355
L369	B296	L37	A53C	L370	B30E	L371	B2C8
L372	B2EA	L373	B2CD	L374	B34C	L375	B30F
L376	B30B	L377	B320	L378	B331	L379	B308
L38	A522	L380	B337	L381	B34B	L382	B35F
L383	B384	L384	B378	L385	B391	L386	BC49
L387	B3F4	L388	B46F	L389	B09C	L39	A3BF
L390	B3E1	L391	B449	L392	BBFC	L393	B44F
L394	BDE7	L395	B4F6	L396	B4A9	L397	B4A8
L398	B4A4	L399	B4B5	L4	A3A4	L40	A52A
L400	B4BF	L401	B4CA	L402	B4D2	L403	B68C
L404	B4F4	L405	B50B	L406	B526	L407	B516
L408	B4D5	L409	B501	L41	A560	L410	B559
L411	B5DC	L412	B54D	L413	B566	L414	B56E
L415	B5C7	L416	B561	L417	B5AE	L418	B63D
L419	B572	L42	A55F	L420	B5B0	L421	B5BD
L422	B57D	L423	B5B8	L424	B601	L425	B5E6
L426	B5F6	L427	B606	L428	A3DC	L429	B544
L43	A544	L430	B67A	L431	B690	L432	ADBB
L433	B6A2	L434	B699	L435	B6A3	L436	B6DB
L437	B6D6	L438	B706	L439	B7B5	L44	E118
L440	B487	L441	B782	L442	B70D	L443	B748
L444	B70C	L445	B761	L446	B79B	L447	B70E
L448	B725	L449	B798	L45	A579	L450	B3A6
L451	B6A6	L452	B1BF	L453	B7CD	L454	B8F9
L455	B7E2	L456	B83C	L457	B7F1	L458	B840
L459	B7F7	L46	A576	L460	B848	L461	BAB7
L462	B86F	L463	B9A6	L464	B8AF	L465	B877
L466	B849	L467	B897	L468	B8A3	L469	B867
L47	A43A	L470	B9BA	L471	B91D	L472	B8D2
L473	B8DB	L474	B94D	L475	B936	L476	B8F7
L477	B938	L478	B929	L479	B947	L48	AAD7
L480	B983	L481	B97E	L482	B999	L483	B9EA

SYMBOL	VALUE						
L484	B9B0	L485	B9AC	L486	B9F4	L487	BA28
L488	BB12	L489	B853	L49	A5A4	L490	E059
L491	BA59	L492	BA8C	L493	BAB9	L494	BA5E
L495	BA61	L496	BBA2	L497	B985	L498	BA9B
L499	BA7D	L5	A412	L50	A5DC	L500	BADF
L501	BACF	L502	BAE2	L503	BAD4	L504	B8FE
L505	BAFE	L506	B893	L507	BB29	L508	BB8F
L509	BB4C	L51	A58E	L510	BB4F	L511	BB7E
L512	BB8A	L513	BB7A	L514	BB3F	L515	BB5D
L516	BC0C	L517	BC1A	L518	BC1B	L519	B97D
L52	A5F5	L520	B946	L521	BC3C	L522	B8D7
L523	BC31	L524	BC98	L525	BCBB	L526	BC9B
L527	BC38	L528	BCF2	L529	BCBA	L53	A5AC
L530	B96F	L531	BCCC	L532	BCF3	L533	BD06
L534	BD0F	L535	BD0A	L536	BD0D	L537	BD2E
L538	BD71	L539	BD47	L54	A5B6	L540	BD49
L541	BD35	L542	BD30	L543	BD33	L544	BD41
L545	BDA0	L546	BD52	L547	BD67	L548	BD62
L549	BB07	L55	A5B8	L550	BD5B	L551	BD6A
L552	BFBE	L553	BD7E	L554	BDAE	L555	BDC2
L556	BDDD	L557	BC4F	L558	BDF8	L559	BE00
L56	A5F9	L560	BF07	L561	BE09	L562	BE0B
L563	BA30	L564	BE47	L565	BE2F	L566	BE28
L567	BE32	L568	BE21	L569	BE16	L57	A613
L570	B850	L571	BE48	L572	BE53	L573	BE64
L574	BE68	L575	BE66	L576	BE90	L577	BE8E
L578	BE97	L579	BEB2	L58	A5DE	L580	BEC4
L581	BEC6	L582	BED3	L583	BEE3	L584	BF0C
L585	BEEF	L586	BF04	L587	BF84	L588	BFFD
L589	BF9E	L59	A5E5	L590	B8FB	L591	BFB4
L592	BCE9	L593	BC02	L594	B9F1	L595	BFED
L596	E000	L597	BC2B	L598	E00B	L599	BC11
L6	A3FB	L60	A5EE	L600	E01E	L601	BADA
L602	E00E	L603	E043	L604	B862	L605	E05D
L606	BAC4	L607	BBD0	L608	E06C	L609	BBCA
L61	A609	L610	E070	L611	E0BE	L612	E07D
L613	E0E3	L614	E0D3	L616	E0F6	L617	E109
L618	A677	L619	E10C	L62	A5C7	L620	FFD5
L621	E104	L622	FFD2	L623	E4B6	L624	FFC9
L625	FFE7	L626	E200	L627	FFDB	L628	FFD8
L629	E1D4	L63	A5C9	L630	E19E	L631	E1A1
L632	E195	L633	E1B5	L634	E1D1	L635	A533
L636	A67A	L637	E23F	L638	FFC3	L639	FFC6
L64	A641	L640	FFC0	L641	FFBD	L642	E20D
L643	E26B	L644	E206	L645	E211	L646	E20E
L647	E219	L648	E257	L649	BB0F	L65	A644
L650	E2A0	L651	E2AD	L652	E2DC	L653	E29D
L654	E0F9	L655	E316	L656	E324	L657	E337
L658	E33D	L659	E386	L66	A637	L660	E3A2
L661	A456	L662	E455	L663	E3E2	L664	E453
L665	E3B9	L666	E3BF	L667	E3A8	L668	E421
L669	FFB7	L67	A640	L670	FF9C	L671	E422
L672	A659	L673	E4AD	L674	FFCC	L675	E4DA
L676	E500	L677	E4EB	L678	E518	L679	E570
L68	A62E	L680	E5AA	L681	E560	L682	E555
L683	EA07	L684	E566	L685	E58D	L686	E57C
L687	E5A0	L688	E597	L689	E56C	L69	A6A4
L690	E5B4	L691	E5CA	L692	E72A	L693	E5E7
L694	E5F3	L695	EA1C	L696	E5B9	L697	E606
L698	E5FE	L699	E5CD	L7	A3E8	L70	A68E
L700	E632	L701	E60F	L702	E64A	L703	E6F4

SYMBOL	VALUE						
L704	E66F	L705	E650	L706	E654	L707	E65D
L708	E690	L709	E682	L71	FFEA	L710	E672
L711	E674	L712	E684	L713	E691	L714	E69F
L715	E6A8	L716	E6CD	L717	E6B6	L718	E8B7
L719	E701	L72	A827	L720	E700	L721	E6DA
L722	E981	L723	E6ED	L724	E8F6	L725	E6F7
L726	E9FF	L727	E880	L728	E716	L729	E6B0
L73	A6BB	L730	E731	L731	E7DC	L732	E73D
L733	E8A1	L734	E74C	L735	E73F	L736	E745
L737	E697	L738	E759	L739	E699	L74	A6C9
L740	E785	L741	E762	L742	E70B	L743	E77E
L744	E8A5	L745	EA5C	L746	E773	L747	E7CE
L748	E78B	L749	E792	L75	A6E6	L750	E7A8
L751	E7C0	L752	E7AD	L753	E7D4	L754	E7AA
L755	E7CB	L756	E7C8	L757	E8CD	L758	EC4F
L759	E7E3	L76	A717	L760	E7EA	L761	E693
L762	E7FE	L763	E832	L764	E82D	L765	E805
L766	E80A	L767	E829	L768	E967	L769	E826
L77	A832	L770	E847	L771	E854	L772	E874
L773	E84C	L774	E864	L775	E871	L776	E86A
L777	E87C	L778	E54D	L779	EC58	L78	A6E8
L780	E891	L781	E888	L782	E8B3	L783	E8B0
L784	E8CA	L785	E8C2	L786	E8CB	L787	E8EA
L788	E8D6	L789	E918	L79	A6EF	L790	E9D4
L791	E913	L792	E94D	L793	E922	L794	E8FF
L795	E958	L796	E956	L797	E98F	L798	E9AB
L799	E9A6	L8	A3F3	L80	BDDA	L800	E9C8
L801	E9BF	L802	E9BA	L803	E965	L804	E9F0
L805	E9E0	L806	E4E0	L807	EA13	L808	FFF0
L809	EA71	L81	A714	L810	EA61	L811	EA24
L812	EA79	L813	EA7B	L814	EA87	L815	EAA8
L816	EB0D	L817	EAB3	L818	EADC	L819	EACB
L82	A72C	L820	EACC	L821	EAE0	L822	EAC9
L823	EAAB	L824	EAFB	L825	EB42	L826	EB17
L827	EB64	L828	EB26	L829	EB6B	L83	E391
L830	EC44	L831	EB76	L832	EAF0	L833	EC5B
L834	EC69	L835	EC72	L836	ED09	L837	EC5E
L838	F0AA	L839	ED2E	L84	A700	L840	ED50
L841	EEA0	L842	ED36	L843	EE8E	L844	EE97
L845	EEB6	L846	EEB3	L847	EDB0	L848	ED66
L849	ED55	L85	A753	L850	ED5A	L851	ED7A
L852	EDB2	L853	ED7D	L854	EEA9	L855	ED9F
L856	EDAD	L857	FE21	L858	EE06	L859	ED40
L86	A737	L860	EDC7	L861	EDDD	L862	EDEB
L863	EDEF	L864	ED20	L865	EE13	L866	EE20
L867	EE47	L868	EE3E	L869	EE5A	L87	A72F
L870	EE56	L871	EDB9	L872	EE30	L873	EE67
L874	EE80	L875	EE85	L876	EE09	L877	EEBB
L878	EF13	L879	EF06	L88	A6F3	L880	EED1
L881	EEE6	L882	EEF6	L883	EF00	L884	EEFC
L885	EEF2	L886	EED7	L887	EEE7	L888	EF2E
L889	EF31	L89	A9C2	L890	EF39	L891	EF3B
L892	EF58	L893	EF59	L894	EF97	L895	EFA9
L896	EF7E	L897	EFE1	L898	EF6E	L899	EF4A
L9	A3EC	L90	A38F	L900	EF90	L901	EFCD
L902	EFC5	L903	EFB1	L904	EF70	L905	EFCA
L906	EFDB	L907	EFD0	L908	F014	L909	F012
L91	A79F	L910	EFF9	L911	F006	L912	F00D
L913	F04C	L914	F017	L915	F04D	L916	F084
L917	F086	L918	F070	L919	F077	L92	A408
L920	F07D	L921	F0A4	L922	F12B	L923	F0BB

SYMBOL VALUE

L924	F14A	L925	F13C	L926	F14E	L927	F166
L928	F173	L929	F09C	L93	A909	L930	E63A
L931	F18D	L932	F1B1	L933	F1D5	L934	F1A9
L935	F199	L936	F196	L937	F193	L938	F817
L939	F1AD	L94	AF08	L940	F84A	L941	F1B5
L942	F1B8	L943	EE1B	L944	F155	L945	F1B4
L946	F1B3	L947	F1DB	L948	F1DD	L949	EDE6
L95	AD8F	L950	F216	L951	F1FC	L952	F867
L953	F207	L954	F208	L955	F028	L956	F1FD
L957	F314	L958	F22A	L959	F704	L96	AD8D
L960	F32E	L961	F237	L962	F245	L963	F233
L964	F062	L965	F70D	L966	ED0C	L967	F248
L968	EDD6	L969	F258	L97	AE58	L970	EDCC
L971	F70A	L972	F25F	L973	F26F	L974	F710
L975	F279	L976	F286	L977	F275	L978	EFF2
L979	F262	L98	BBC7	L980	ED11	L981	F289
L982	F298	L983	EDBE	L984	F316	L985	F2BA
L986	F2F2	L987	F2F1	L988	F2E0	L989	F30D
L99	A7AE	L990	F49E	L991	FE2D	L992	F2BF
L993	F2C8	L994	F483	L995	F7D7	L996	F1F8
L997	F2EE	L998	F781	L999	F654	LA	00B8
LASTPT	0017	LAT	0259	LBERR	0008	LDTB1	00D9
LDTND	0098	LEFT	B700	LEN	B77C	LINNUM	0014
LINTMP	00F2	LIST	A69C	LLEN	0028	LLEN2	0050
LNMX	00D5	LOAD	E168	LSTP	00CA	LSTSHF	028E
LSTX	00C5	LSXP	00C9	LTBLUE	000E	M26AJB	0295
M26CDR	0294	M26CTR	0293	MAXCHR	0050	MEMSIZ	0283
MEMSTR	0281	MEMTOP	0037	MEMUSS	00C3	MID	B737
MODE	0291	MODE1	EB81	MODE2	EBC2	MODE3	EC03
MS1	F0BD	MS10	F106	MS11	F10E	MS17	F120
MS18	F127	MS21	F116	MS5	F0C9	MS6	F0D4
MS7	F0D8	MS8	F0EB	MSGFLG	009D	MYCH	00BF
NBASIN	F157	NBSOUT	F1CA	NCHKIN	F20E	NCKOUT	F250
NCLALL	F32F	NCLOSE	F291	NCLRCH	F333	NDX	00C6
NEW	A642	NEXT	AD1E	NGETIN	F13E	NLINES	0019
NLOAD	F4A5	NMI	FE43	NMINV	0318	NMIPAN	E59A
NNMI	FE47	NOPEN	F34A	NOT	AED4	NSAVE	F5ED
NSTOP	F6ED	NWRAP	0002	NXTBIT	00B5	OCHAR	00BD
OLDI IN	003B	OLDTXT	003D	ON	A94B	OPEN	E1BE
OR	AFE6	PCNTR	00A3	PEFK	B80D	PLF	0003
PNT	00D1	PNTR	00D3	POKE	B824	POS	B39E
POWER	BF78	PRINTH	AA80	PRP	00B6	PRTY	009B
PTR1	009E	PTR2	009F	PULS	FF48	QTSW	00D4
R2D2	00A3	RDFLG	00AA	READ	AC06	REM	A93B
RER	00A8	RESHO	0026	RETURN	A8D2	REZ	00A9
RIBUF	00F7	RIDATA	00AA	RIDBE	029B	RIDBS	029C
RIGHT	B72C	RINONE	00A9	RIPRTY	00AB	RND	E097
RNDX	008B	ROBUF	00F9	RODATA	00B6	RODBE	029E
RODBS	029D	ROPRTY	00BD	ROWS	DC01	RPTFLG	028A
RSSTAT	0297	RUN	A871	RVS	00C7	SA	00B9
SAH	00AD	SAL	00AC	SAT	026D	SAVE	E156
SBERR	0004	SFDX	00CB	SGN	BC39	SGNFLG	0067
SHCNH	00AB	SHCNL	00A7	SHFLAG	028D	SHFLOG	EB48
SIDREG	D400	SNSW1	00B4	SPERR	0010	SQR	BF71
STAH	00C2	STAL	00C1	START	FCE2	STATUS	0090
STKEY	0091	STR	B465	STREND	0031	SUBFLG	0010
SVXT	0092	SYNO	0096	SYS	E12A	SYSRTN	E147
T1	009E	T2	009F	TAN	E2B4	TANSGN	0012
TAPE1	00B2	TBL80	FD10	TBLX	00D6	TBUFFR	033C
TEMP	00B1	TEMPPT	0016	TEMPST	0019	TIMB	FE66
TIME	00A0	TIMOUT	0285	TMP2	00C3	TMPC	009F

```
SYMBOL VALUE
 TMPO     00C1    TRMPOS   0009    TXTPTR   007A    TXTTAB   002B
 USER     00F3    USRCMD   032E    VAL      B7AD    VALTYP   000D
 VARNAM   0045    VARPNT   0047    VARTAB   002D    VECTSS   FD30
 VERCK    0093    VERCKB   000A    VERIFY   E165    VICREG   D000
 WAIT     B82D    WRST     E37B    WRTN     FBCD    WRTZ     FC6A
 XMAX     0289    XSAV     0097

END OF ASSEMBLY
```

A knowledge of the internal workings of the 64 ROMs is essential for anyone wishing to do serious programming of the Commodore 64, especially when programming in machine code. Never before has there been published a full annotated assembly listing of the 64 ROM software. Each routine is shown in detail and its function and potential use by other programs examined in detail with full notes on how to use it.

Armed with this book, any serious programmer of the Commodore 64 will be able to make the fullest use of the machine's potential as well as gaining a fascinating insight into the way the machine works.

The Authors

Nick Hampshire is a well-known author and microcomputer expert who has specialised in Commodore computer equipment. He started the first hobby microcomputer magazine, later absorbed into *Practical Computing*, of which he was technical editor for several years. He was the co-founder of *Popular Computing Weekly* and founder and managing editor of *Commodore Computing International* magazine. He is also the author of over a dozen books on popular computing, including the very successful and widely acclaimed *PET Revealed* and *VIC Revealed*.

Richard Franklin and Carl Graham are programmers with Zifra Software Ltd and together with Nick Hampshire have written some of the software included in this book.

Also by Nick Hampshire

ADVANCED COMMODORE 64 BASIC REVEALED
0 00 383088 8

ADVANCED COMMODORE 64 GRAPHICS AND SOUND
0 00 383089 6

THE COMMODORE 64 KERNAL AND HARDWARE REVEALED
0 00 383090 X

THE COMMODORE 64 DISK DRIVE REVEALED
0 00 383091 8

COLLINS
Printed in Great Britain
0 00 383087 X

£8.95 net