PET Feature
Growing Knowledge Trees
6502 and 6809 Memory Moves
Apple Printer Utilities
LISZT with Strings

by Leonard H. Anderson, Donald Cohen, Richard F. Searle

LISZT turns your Applesoft program listing into an easy-to-understand structured format. The program is designed to be flexible and works with a variety of printers.

LISZT requires:
- Apple II with Applesoft
- Disk Drive
- Printer

Can you understand a program you wrote six months ago? Do you remember some of those special tricks imbedded in a concatenated line of code? The "LISZT" (Logical Interpreter Statement Zeugmatic Tabulator) can help you understand BASIC source code listings by structuring printouts in a clear, orderly form with a minimum of extra characters. Written for the Apple II Plus, it can be modified for other BASIC dialects.

Credit is due Mark Capella for the first listing program. Since then, two others have been published. Not completely satisfied, we decided to start fresh with the following rules:
1. Print results so they are easy to read.
2. Make the program adaptable to various printers.
3. Gather statements in strings for flexibility.
4. Separate REMs from printed code.
5. Omit the concatenation colon and "LET."
6. Split over-long print lines at a logical character.
7. Indent FOR-NEXT loops globally.
8. Indent IF-THEN statements locally.
9. Minimize disk operations.

The main program, LISZTER, was written in linear form to accommodate different printers and to allow easy deletion or addition of special features. This article is both a program description and a partial history of program development.

**Applesoft Source Code Structure**

Source code structure rules the program. One line of Applesoft BASIC is shown in figure 1. Each line contains five overhead bytes: two for a pointer to the next line, two more for the number, and an end-of-line null (binary zero) byte. The last line number source code ends in three null bytes to indicate end-of-program.

All variable names, strings, and punctuation not a function are expressed as 7-bit ASCII with most-significant-bit (MSB) set false or zero. All function words (IF, NEXT, REM, etc.) are stored as one-byte "tokens" with MSB set true or high. There are 107 Applesoft tokens.

**Starting the Program Organization**

Figure 2 is the initial flow chart. Each program byte is examined, beginning with decimal memory location 2049. "Standard" ROM Applesoft code begins here. It can be changed and will vary for other BASICS. String variables hold the line number in NS, statement text in a "gather" string, G$, and the "combination" printout string, C$.

A remarks flag is set if a REM token is encountered. The first decision separates remarks by blank print lines but groups successive remarks without blank lines. Remarks are highlighted without appearing to be part of the main coding.

ASCII characters and token bytes are parsed next with tokens converted to the original function word. This section and the print line formatting section receive the most attention. A prime example is separating concatenated statements and allowing indication of over-long text lines.

**Holding Two BASIC Programs in Memory**

Applesoft reserves two bytes in page zero (first 256 bytes) for the starting address. Start location is normally decimal 2049 for ROM BASIC, stored in locations 103 (low byte) and 104 (high byte). End-of-program in memory is in locations 175 (low) and 176 (high). Either can be changed from the keyboard or program in memory.
Apple's DOS allows the simulation of keyboard commands with an EXEC Text File. An EXEC file loads statements into the keyboard buffer. Each statement is then executed as if it were a keyboard command.

The program to be listed is loaded first. The EXEC file is called next by typing "EXEC LISZT." LISZT then changes normal program start address to the end of program plus two, loads and runs the LISZTER working program. Loading LISZTER will automatically set the new end-of-program address.

Although two programs are now in memory, Applesoft will only execute LISZTER as indicated by the starting address changed by EXEC file LISZT. Original start and end addresses are held in page zero scratchpad locations; LISZTER resets start and end from these scratch locations on completion of printout.

EXEC file LISZT is generated by the short program in listing 1. MAKE LISZT may be deleted after generating LISZT. LISZT EXECution commands are those indicated within quotes in MAKE LISZT line numbers 225 through 265.

LISZTER start location is set slightly higher than normally expected. This and the extra nulls will insure that the listed program can be RUN normally after LISZTER resets start and end addresses on printout completion. Normal source code ending must be three successive null bytes.

**Setting Up LISZTER**

LISZTER begins execution at line number 82 by initializing the variables. Initializing will speed up execution, especially with string variables in Applesoft. Token array T3 contains the 107 function words expressed as literal strings in the DATA statements. Direct expression as strings allows spaces to be added for clarity in gathering and converting the tokens.

The REM token word was changed to an asterisk. It is left as an isolated DATA declaration for those desiring another symbol or word. LET appears as a null string in line 88 to permit completion of the array; token parsing will skip over a LET.

Screen prompts in lines 94 to 100 are optional. Printed page length is normally 60 lines per page including the header. Indent spacing is normally four column spaces, fitting the REM asterisk with three following blanks.

---

**Figure 2: Initial Flow Chart and Sectioning of LISZTER**

![Flow Chart](Image)

**Listing 1: MAKE LISZT "EXEC" file generator printed with the LISZTER program in listing 2.**

<table>
<thead>
<tr>
<th>MAKE LISZT</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEONARD H. ANDERSON</td>
</tr>
<tr>
<td>PRINTOUT ON 20 AUGUST 1981</td>
</tr>
<tr>
<td>200</td>
</tr>
<tr>
<td>205</td>
</tr>
<tr>
<td>210 D4 = Chr$(4)</td>
</tr>
<tr>
<td>215 C6 = Chr$(13)+D4</td>
</tr>
<tr>
<td>220 Print C6&quot;OPEN LISZT&quot;</td>
</tr>
<tr>
<td>225 Print &quot;POKE208, PEEK(103)&quot;</td>
</tr>
<tr>
<td>230 Print &quot;POKE209, PEEK(104)&quot;</td>
</tr>
<tr>
<td>235 Print &quot;POKE210, PEEK(175)&quot;</td>
</tr>
<tr>
<td>240 Print &quot;POKE211, PEEK(176)&quot;</td>
</tr>
<tr>
<td>245 Print &quot;POKE104, PEEK(211)&quot;</td>
</tr>
<tr>
<td>250 Print &quot;IF PEEK(210) &gt; 253 THEN POKE103, (PEEK(210)+1)&quot;</td>
</tr>
<tr>
<td>255 Print &quot;IF PEEK(210) &gt; 253 THEN POKE103, (PEEK(210)+1)&quot;</td>
</tr>
<tr>
<td>260 Print &quot;IF PEEK(210) &gt; 253 THEN POKE103, (PEEK(210)+1)&quot;</td>
</tr>
<tr>
<td>265 Print &quot;POKE104, PEEK(211)&quot;</td>
</tr>
<tr>
<td>270 Print &quot;RUN LISZTER&quot;</td>
</tr>
<tr>
<td>275 Print D4&quot;CLOSE&quot;</td>
</tr>
<tr>
<td>280 End</td>
</tr>
</tbody>
</table>

End of Listing

Program Length = 642 Bytes, Total of 17 Line Numbers
19 Total Non-REM Statements, 2 Total Remarks
Line 104 assumes the printer interface located in peripheral slot 1. The "POKE 33,30" sets screen window width to 30 columns, necessary only for certain Apple I/O interfaces. Line 106 is the main printer control statement.

**Subroutines**

"GET BYTE" simply advances program byte pointer P and fetches a decimal value of the byte in B. "BLANK LINE PRINT" is used mainly in REM separation. String $S$ is a single blank space and required by Centronics 737 printers; most other printers will accept a single PRINT without a variable.

"TEST PAGE" (lines 6 through 14) does several things: line count (LC) advance and test, header printing and a "continued" indicator print on the current page if another page is to be printed. Since concatenated statements are separated, some may be printed on the following page. Lines 13 and 14 place brackets around the line number and print that on the following page for clarity.

**Upper and Lower Case**

The mixture of upper and lower case characters is due primarily to the TS array DATA declarations. We decided that program statements would appear better with the familiar function words in a mixture, non-token statements in normal upper-case only. The choice is up to the user.

Since none of us has direct lowercase control, an available utility program was used to "zap" the desired characters directly on the disk. You should do this on a copy file in disk; strange results occur if a token is accidentally changed.

**First-Byte Decision**

In the section occupying lines 29 through 33 on listing 2 remarks are separated and grouped. A single, first-byte colon in Applesoft is equivalent to a REM; first-byte colons are changed to REMs since the asterisk-equivalent voids the original source code separation intent.

**Statement Line Parsing**

Line 34 begins a number of decisions. A null program byte indicates end-of-line in Applesoft and jumps to the print line formatting routine at line
54. A decimal value between 1 and 127 is an ASCII character byte; any value above 127 is a token.

The double quote test at line 36 allows colons within quotes or remarks. Any other colons are treated as delimiter characters and tested at line 37. A delimiter forces a new print line but not a new line number as in the case of a null value byte.

Control characters are converted to upper case equivalents. Besides making control characters visible, conversion allows a printer to continue without suddenly switching to a new mode! We enclosed control characters in vertical bars because that print character has little use in normal printing.

Token byte values are changed to allow you to gather them from the T$ array. A token value out of normal range is made into a distinctive word at line 40. A test-true here would indicate an error.

The REM flag set at line 43 is primarily for concatenated remarks. The remarks counter is optional and used only for end-of-listing statistics. REM spacing variable RS is set to one for indenting remarks. While remarks are highlighted, we also wanted their appearance out of the normal program flow.

The FOR flag sets up the start of global FOR-NEXT indenting. The FOR spacing counter is advanced in print line formatting to allow completion of the entire FOR statement. The NEXT test at line 48 removes one FOR indent space. This space is held at zero in case an intermediate (but legal) NEXT is used with the loop.

Conditional tests add an indent space on completion of a THEN. Anything following a THEN, even if only a line number, is considered a separate statement. An IF-GOTO is considered a single statement. The choice was arbitrary to reduce total code.

A LET token is ignored by choice. Omitting line 47 allows you to print a LET.

DATA flag [DF] is used solely in print formatting. When set, it allows splitting an over-long print line only on commas. This is useful when DATA declarations contain strings with spaces as in LISZTER itself.

Listing 2 (Continued)

19 # OPTIONAL STATISTICS
20 Gosub 4
21 Gosub 4
22 Gosub 6
23 Print "Program Length = ", (Peek(211)-Peek(209))#256+Peek(210)-Peek(208) " Bytes, 
24 Total of "ITN" Line Numbers"
25 Gosub 4
26 Gosub 6
27 Print "ITR" Total Non-Rem Statements, "ITR" Total_
28 Remarks"
29 Gosub 4
30 Gosub 6
31 Print "END "
32 Gosub 4
33# TUR Y OFF PRINTER, DISPLAY END PROMPT ON SCREEN
34 Pr# 0
35 Poke 33,40
36 Home
37 VTab 12
38 Mtab 11
39 Inverse
40 Print " END OF LISTING "
41 Normal
42* RESET PAGE 0 POINTERS FOR THE LISTED PROGRAM
43 Poke 105,Peek(210)
44 Poke 106,Peek(211)
45 Poke 107,Peek(210)
46 Poke 108,Peek(211)
47 Poke 109,Peek(210)
48 Poke 110,Peek(211)
49 Poke 111,Peek(115)
50 Poke 112,Peek(116)
51 Poke 103,Peek(208)
52 Poke 104,Peek(209)
53 Poke 175,Peek(210)
54 Poke 176,Peek(211)
55 End
56* MAKE THE LINE NUMBER STRING
57 TN = TN+1
58 Gosub 2
59 D = B
60 Gosub 2
61 K = B+256-D
62 D = Len(Str$(K))
63 N$ = Right$(Len$(Len$(K)+7-D)+Str$(K)+" ",&)B
64* BEGIN LINE PARSING WITH FIRST-BYTE DECISION
65 TB = TB+1
66 D = 0
67 Gosub 2
68 If B = 58 Then
69 B = 178
70 # CONVERT "SIMPLE REM" (A "#" FIRST-BYTE) TO ORDINARY
71 "REM"
72 If B = 178 And Not RF Then
73 Gosub 4
74 B = 34
75 # "REM" FLAGS ARE SET AFTER SEPARATION OF TOKENS;
76 "REM" GROUPS SEPARATED BY BLANK PRINT LINES.
77 If B = 178 And RF Goto 34
78 # BYPASS RF RESET
79 If RF Then
80 RF = 0
81 Gosub 4
82# RE-ENTRY POINT FOR NEXT BYTE IN STATEMENT DECISION FLOW
83 If B = 0 Goto 54
84 # FORCE A NEW LINE ON THE END-OF-LINE NULL MARKER
85 If B>127 Then
86 B = B-127
87 Goto 41
88 # BYTE IS A TOKEN; REMAINDER ARE CHARACTERS (continued)
Listing (Continued)

36 If B = 34 Then
    QF = -QF
    * TOGGLE QUOTE FLAG FOR COLON-PRINT TEST IN NEXT LINE

37 If B = 58 And Not RF And QF<1 Then
    TB = TB+1
    Goto 54
    * OMIT THE CONCATENATION "!" AND FORCE A NEW LINE, ELSE PRINT THE COLON AS A CHARACTER

38 If B=32 Then
    B = B+64
    BS = BS+Chr$(124)+Chr$(B)
    B = 124
    * PRINT CONTROL CHARACTERS AS UPPER-CASE BETWEEN VERTICAL BARS; INDICATOR OF CONTROL CHARACTER OPTIONAL.

39 BS = BS+Chr$(B)
    Gosub 2
    Goto 34

40 * INDICATE UNUSED TOKENS AND CONTINUE

41 If B>107 Then
    BS = BS+" ?! "
    Gosub 2
    Goto 34

42 * ACCEPTABLE TOKENS...

43 If B = 51 Then
    TR = TR+1
    RF = 1
    RB = 1
    * SET BOTH FLAGS AND TOTAL-COUNT ON "REM"

44 If B = 2 Then
    FF = 1
    * A "FOR" IS STARTED

45 If B = 69 Then
    CF = 1
    BS = BS+TB(B)
    Goto 54
    * FORCE A NEW LINE AFTER PRINTING A "THEN"

46 If B = 4 Then
    DF = 1
    * "DATA" STATEMENT BEGIN; WILL AFFECT INDENTING LATER

47 If B = 43 Then
    Gosub 2
    Goto 34
    * IGNORE A "LET" (IT IS A NULL STRING IN DATA STATEMENT LINE #00)

48 If B = 3 Then
    FS = FS-1
    If PBYO Then
        FS = 0
        * "NEXT" TOKEN REMOVES A "FOR" LOOP INDENT

49 BS = BS+TB(B)
    Gosub 2
    Goto 34

50 * ADD EXTRA INDENT FOR EACH SPLIT LINE, LIMITING FOR LINE-UP OF "REM" AND "DATA" PRINT-OUTS

51 BF = 0
    RB = RB+1
    If RB>2 Then
        RB = 2

52 If DF And RB>1 Then
    RB = 1

53 * GET TOTAL INDENT SPACES FOR PRINT LINE PLUS LOW-LIMIT FOR SPLIT-POINT ("E")

54 K = IM$(FS+CS+RB)
    E = K+13
    If K>0 Then
        BS = Left$(BB$,K)+BS

(continued)
number and statement but conflicted with Backus-Naur notation.

Uncompleted split lines jump to line 51 for extra indents. A remark allows one extra indent count to line up the remark second line with first line text. The REM symbol used here takes four columns or one default indent space. The DATA declaration single indent (line 52) seemed to be most readable.

Separate flag and counter variables on FOR and IF statements allow for concatenation in one line number of source code and the global or local indenting in printout. Local indenting of conditionals is reset on a new line number but global indenting of FOR loops is decremented only on a NEXT token at line 45.

A new source code line number is begun only when the program byte contains the end-of-line null.

**Ending it All**

Applesoft indicates the end of a listing by three successive nulls. This would appear as a zero line number—a second zero line number, since LISZTER begins with line number zero. This second zero line number falls through the IF in line 16 to begin optional statistical printouts at lines 18, 20, and 21.

Line 23 disables all Apple peripherals by "PR#0", resets screen width to normal by "POKE 33,40", and indicates a finish on the screen. The print command at line 106 allowed the screen to be active at all times even though lower case characters appear as nonsense on a standard Apple.

The POKEs in line 25 reset the start and end pointers to their original values prior to the EXEC file command. Variable and array space pointers are also reset permitting the user to RUN the program after LISZTING.

**Optional Starting Prompts**

The "RUN 23" notice in line 94 should remain until the user is very familiar with LISZT. It is the only way to restore start and end pointers after a RESET. Address locations in line 95 are optional, useful only with very long programs.

Page length, left margin, and indent spacing are useful only if different paper is used. If available, different vertical printer spacing could be added to
the page length prompt at line 98. A variable left margin requires the BB$ string to be slightly longer than onehalf print line width.

We recommend that you retain the inverse video reminder at line 102. Concentration on program development makes us forget the right buttons to push at crucial moments!

Final Thoughts

A "REM-less" version of LISZTER is about 3.9K long and will run in 5.5K of free memory. Disk operations are not required after the initial EXEC LISZT command.

Hesitation in execution occurs only in parsing long character lines. LISZTER's line 76 takes about 20 seconds to gather, split, and begin printing. The 256-byte string maximum has not yet been reached, including one LISZTing over 30 print pages.

Lack of concatenation character does not seem to hamper reading. Those familiar with the interpreter syntax will know it is always there. Statement separation is easier to understand and is improved further with indenting.

Thanks are due to Cliff Bruhn, Dennis Kalo, Sterling Tate, Wes Ten, and Bob Keene of Candid Computers for their trial runs, comments, and suggestions.

References


Listing 2 (Continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>79</td>
<td>SF = 0</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>G6 = &quot;&quot;</td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>IF B = 0 Then</td>
<td>CB = 0</td>
</tr>
<tr>
<td>82</td>
<td>Goto 16</td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>* BET ANOTHER PRINT LINE IF NOT E-O-L NULL, ELSE FALL</td>
<td>THROUGH AND BET ANOTHER LINE NUMBER</td>
</tr>
<tr>
<td>84</td>
<td>Bgosub 2</td>
<td>D = 1</td>
</tr>
<tr>
<td>85</td>
<td>Goto 34</td>
<td></td>
</tr>
<tr>
<td>86</td>
<td># INITIALIZATION OF VARIABLES</td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>Dim T$(107), H$(4)</td>
<td></td>
</tr>
<tr>
<td>88</td>
<td># INITIAL VARIABLE SETTING HAS AN 80-CHARACTER WIDE PRINT</td>
<td>LINE AND 60-LINE PAGE LENGTH (INCLUDING HEADER, EXCLUDING</td>
</tr>
<tr>
<td>89</td>
<td>THE &quot;P=2049&quot; IN LINE #5 ASSUMES A NORMAL APPLESOFT ROM</td>
<td>START AT DECIMAL ADDRESS 2049. CHANGE FOR APPLESOFT IN</td>
</tr>
<tr>
<td>90</td>
<td>P = 2048</td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>B = 0</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>CB = 0</td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>F5 = 0</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>RF = 0</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>CF = 0</td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>FF = 0</td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>BF = 0</td>
<td></td>
</tr>
<tr>
<td>98</td>
<td>DF = -1</td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>LL = 80</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>LP = 60</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>IM = 4</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>E = 0</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>TN = 0</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>TB = 0</td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>TR = 0</td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>S$ = &quot; &quot;</td>
<td></td>
</tr>
<tr>
<td>107</td>
<td>C$ = &quot; &quot;</td>
<td></td>
</tr>
<tr>
<td>108</td>
<td>N$ = &quot; &quot;</td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>G$ = &quot; &quot;</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>M$ = &quot; &quot;</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>H$(0) = &quot; &quot;</td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>LBS$ = &quot; &quot;</td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>BB$ = &quot; &quot;</td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>Data &quot;#&quot; &quot;CHANCE REM TOKEN WORD INDICATOR AS DESIRED&quot;</td>
<td></td>
</tr>
<tr>
<td>119</td>
<td>For K = 1 To 107</td>
<td>Read T$(K)</td>
</tr>
<tr>
<td>120</td>
<td>Next</td>
<td></td>
</tr>
<tr>
<td>121</td>
<td># SCREEN PROMPTS AND ALTERNATE LISTING CONSTANTS</td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>Home</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>VTab 3</td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>Flash</td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>Print &quot; RUN 23 &quot;</td>
<td></td>
</tr>
<tr>
<td>126</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>Print &quot; RESTORES ORIgINAL AFTER RESEt&quot;</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>&quot; RUN 22 &quot; RESTORES POINTERS FOR PROGRAM START AND END TO</td>
<td>ORIGINAL VALUES AND RESETS SCREEN</td>
</tr>
</tbody>
</table>

(continued)
Problems You May Encounter with LISZTER with Strings

1. A colon ending a line causes a stop and 'error at line 76' display. The best solution is to use a line editor program or keyboard to correct the program line to remove the extraneous byte. Usually appears to be a 'forgotten' removal during program editing.

2. A double colon starting a line causes LISZTER to think the first colon is a REM, but the second colon causes reversion to gathering tokens and characters in the usual manner. Using an italics set on the printer will make this line look like a REM splat, but has both upper and lower case contents. Best solution is to edit out the extra colons.

3. A statement ending nested FOR loops such as 'NEXT J,K.L' executes in Applesoft as if they were three separate NEXT statements. Since LISZTER will only recognize one NEXT token, all following lines will retain the FOR-NEXT indent(s) for the remainder of printout.

   We don't have a simple solution for this — yet. Changing the program to 'NEXT J:NEXT K:NEXT L' will add only two bytes and bring the left margin back to normal. The two added bytes are the NEXT tokens; concatenation colons take the place of the commas.

4. On any mid-printout deliberate stop, such as RESET, you must key in RUN 23 to restore the program start and end pointers. Failure to do so may attach LISZTER to the program being listed.

Richard F. Searle is manager of software and control systems at an aerospace corporation and a consultant on the application of solar energy. You can write to Mr. Searle at 4511 Adam Road, Simi Valley, California 93063.

Donald Cohen is an electronics engineer specializing in microprocessor control of aircraft emergency power systems. He is also a programming consultant and partner of Donald Cohen Associates. Contact Mr. Cohen at Donald Cohen Associates, 4613 Wolfe Way, Woodland Hills, California 91364.

Leonard H. Anderson is a hardware and systems engineer, who uses his Apple II for number crunching (when not playing games). He is also a contract writer and currently an Associate Editor with Ham Radio Magazine. Address correspondence to 10048 Lanark Street, Sun Valley, California 91352.
"LISZTER"

Listing 2 (Continued)

114 * Working Program to
115 * re-format
116 * APPLESOFT Programs for
117 * Printing
118 * BY
119 * LEONARD H. ANDERSON
120 * Version 4.1.3, 7/24/81
121 * (lower-case version)
122 * MA-80 & "BRAFTRAX"
123 * (ITALICS ON REMS)
124 * DESCRIPTION OF VARIABLES:
125 * B PROGRAM BYTE DECIMAL VALUE
126 * BB 'BIG BLANK' STRING OF 48 SPACES
127 * CF "IF" FLAG: 1 = "IF" STARTED; 0 = NO "IF"
128 * CS "IF" (CONDITIONAL) INDENT SPACE COUNTER
129 * C# CHARACTER AND TOKEN STRING TO BE PRINTED
130 * D 'DIRECTION', A TEMPORARY
131 * DF "DATA" FLAG (ALLows SPLIT ON COMMA ONLY)
132 * E TEMPORARY, FOR SPLIT-LINE LIMITS
133 * FS "FOR" FLAG: 1 = "FOR" STARTED; O = NO "FOR"
134 * Gn 'GATHER' STRING TO BUILD STATEMENT LINE
135 * H# HEADER ARRAY FOR PAGE TITLE
136 * IF INDENT SPACE MULTIPLIER
137 * K TEMPORARY
138 * LB# 'LITTLE BLANK' STRING OF 8 SPACES
139 * LC LINE COUNTER FOR PRAGMATIC TEST
140 * LL LINE-LENGTH (WIDTH) CONSTANT
141 * LP LINES-PER-PAGE CONSTANT
142 * Mn LEFT MARGIN SPACING STRING
143 * M# LINE NUMBER STRING
144 * PO POINTER TO PROGRAM BYTE (DECIMAL VALUE)
145 * Pc PAGE COUNTER FOR HEADER ON EACH PAGE
146 * Gf QUOTE FLAG TO ALLOW/DISALLOW COLON PRINTING
147 * +1 = NO QUOTE OR SECOND QUOTE OF PAIR EXISTS
148 * +1 = FIRST QUOTE OF PAIR EXISTS, ALLOW COLONS
149 * +1 = REM FLAG: 1 = "REM" STARTED; 0 = NO "REM"
150 * Rs "REM" INDENT SPACING COUNTER
151 * Sf SPLIT-LINE FLAG: 1 IF PRINT LINE MUST BE SPLIT
152 * Sd SINGLE SPACE STRING
153 * Lt TOTAL LINE NUMBER COUNTER
154 * Tr TOTAL REMARK-STATEMENT COUNTER
155 * Ts TOTAL STATEMENT COUNTER
156 * AN EXAMPLE OF INDENTS ON NESTED "FOR" LOOPS:
157 For J = 1 To 25
158 For K = J To 26
159 If M(K,J) = 0 Boto 170
160 For L = J To K
161 Next L
162 Next K
163 Next J
164 * THE PRECEDING LINE CONTAINED TWO CONTROL-I CHARACTERS SEPARATED BY A CONTROL-O (BEL).

End of Listing

Program Length = 10061 Bytes, Total of 175 Line Numbers
271 Total Non-Ren Statements, 119 Total Remarks

END

APPLESCOPE

DIGITAL STORAGE OSCILLOSCOPE
Interface for the Apple II Computer

The APPLESCOPE system combines two high speed analog to digital converters and a digital control board with the high resolution graphics capabilities of the Apple II computer to create a digital storage oscilloscope. Signal trace parameters are entered through the keyboard to operational software provided in PROM on the DI control board.

* DC to 15 MHz sample rate with 1024 byte buffer memory
* Petrieger Viewing
* Programmable Scale Select
* Continuous and Single Sweep Modes
* Single or Dual Channel Trace

Price for the two board Applescope system is $595

EXTERNAL TRIGGER ADDAERTER $29

APPLESCOPE ACCESSORIES

APPLESCOPE-H12: High resolution 12 bit analog to digital converter with sample rates to 100 kHz. Requires 48K Apple II with disk drive. Software provides on floppy disk includes basic SCOPE DRIVER package.

Price per channel $695

APPLESCOPE-H18: High Resolution And High Speed. Circuit combines two 8 bit flash analog to digital converters to give a 10 bit dynamic range. The 10 bit converter resolution is maintained at sampling rates up to the 7 Mhz maximum for signal slew rates less than 5 volts per microsecond. Larger input slew rates will reduce the converter resolution to 6 bits until the signal stabilizes within the 5 volt per microsecond limit. Requires 48K Apple II and disk drive. Software provided on disk includes basic SCOPE DRIVER package.

Price per channel $695

APPLESCOPE-EXT: External trigger adapter has a switch selectable external trigger input to a BNC connector mounted in a rear slot of the Apple II computer.

Price $29.00

APPLESCOPE-BNC BNC connector adds the Bergstick connectors on the A1 circuit card to make BNC plugs mounted in a rear slot of the Apple II.

Price $14.95

BUS EXTENDER: Allows easy access to Apple II peripheral card slots.

Price $19.95

SCOPE PROBES: Oscilloscope probes for use with the APPLESCOPE - BNC adapter.

Price $29.95

SCOPE DRIVER: Advanced software for the Applescope system provided on 5 1/4" floppy disk. Available options include:

* Signal Averaging - Acquires 1 to 255 signal sweeps and displays the averaged result.
* Digital Volt Meter - Allows use as a real time DVM or use to measure points on an acquired sweep.
* Hard Copy - Uses graphics printer to produce hard copy output of displayed traces.
* Disk Storage - Allows automatic storage and recovery of acquired data on floppy disks.

Spectrum Analyzer - Calculates and displays frequency spectrum of acquired data.

BUS RIDER

LOGIC ANALYZER for the APPLE II

The BUS RIDER circuit card silently rides the Apple II peripheral bus and allows real time tracking of program flow. Software provided on EPROM allows set up of trace parameters from the keyboard and read back of disassembled code after a program has been tracked.

* 32 bit by 1024 sample memory buffer
* Monitors Data and Address bus plus 8 external inputs
* Trigger on any 32 bit word or external trigger
* Petrieger viewing

The BUS RIDER is an invaluable development tool for anyone working with Apple II or Apple III computers.

Price $295

RC ELECTRONICS INC.
7265 Tuolumne St., Goleta, CA 93117
VISA (805) 968-6614
MA/C

No. 48 - May 1982
MICRO - The 6502/6809 Journal

45