Part XIV: MACRO MAGIC

Macros are often misunderstood, and almost always underutilized, but they can make assembly language programming much easier. Scott shows you how!

Assembly language outperforms high-level languages (like BASIC and Pascal) in speed of execution and flexibility of operation, but high-level languages outperform assembly language in efficiency of programming and readability of program code. If only you could magically combine the speed of assembly language programs with the readability of high-level language programs . . . . And you can, with macros.

MACROS

Overview

Macros are single-word or single-line commands that can replace several lines of assembly source code. For example, a common operation in assembly language programs is storing a two-byte (16-bit) value from one variable into another variable:

```
LDA VAR1
STA VAR2
LDA VAR1+1
STA VAR2+1
```

This is equivalent to VAR2 = VAR1 in BASIC or VAR2 := VAR1 in Pascal. In this example, one line of high-level-language code equals four lines of assembly code. However, if you have a macro assembler (The MicroSPARC Assembler, Merlin, ORCA/M, or S-C Macro Assembler), you can define a simple macro so that the single line of code:

```
STOR2 VAR1,VAR2
```

in your assembly source file equals the original four lines.

Unfortunately, macro definition and usage differ from one assembler to another. To define STOR2 with The MicroSPARC Assembler, you would include the following code in your source file:

```
STOR2 MAC
      LDA :A
      STA :B
      LDA :A+1
      STA :B+1
      EMC
```

where the pseudo-opcode (or assembler directive) MAC indicates the start of the definition of the STOR2 macro, :A and :B are parameters passed to the macro, and the pseudo-opcode EMC indicates the end of the macro definition. With the Merlin Pro assembler, you would define the same macro as follows:

```
STOR2  MAC
       LDA ]1
       STA ]2
       LDA ]1+1
       STA ]2+1
       EOM
```

With ORCA/M the definition is:

```
      MACRO
&LAB  STOR2 &NUM1,&NUM2
      LDA &NUM1
      STA &NUM2
      LDA &NUM1+1
      STA &NUM2+1
      MEND
```

And with the S-C Macro Assembler the definition is:

```
 .MA STOR2
 LDA ]1
 STA ]2
 LDA ]1+1
 STA ]2+1
 .EM
```

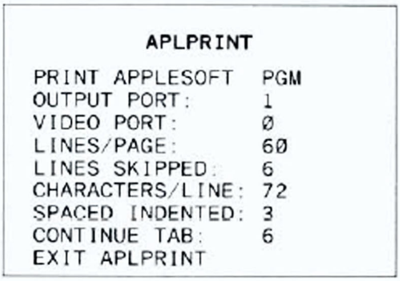
You can see that all the assemblers are slightly different. Carefully read the Macros section in your particular assembler user's manual to learn how macros are defined and used with your system.

Once you define the macro, you can use it just like any other assembler mnemonic. For example, a line of code in APLPRINT (an example program discussed later) looks like this:

```
STOR2 TXTTAB,TXTPTR
```

The assembler translates (or expands) this line into:

FIGURE 1: APLPRINT Menu



```
            APLPRINT

PRINT APPLESOFT    PGM
OUTPUT PORT:       1
VIDEO PORT:        Ø
LINES/PAGE:        6Ø
LINES SKIPPED:     6
CHARACTERS/LINE:   72
SPACED INDENTED:   3
CONTINUE TAB:      6
EXIT APLPRINT
```

```
LDA TXTTAB
STA TXTPTR
LDA TXTTAB+1
LDA TXTPTR+1
```

The program uses STOR2 just like a normal mnemonic opcode, except it represents more than one machine language command.

But macros can perform even more magic. With The MicroSPARC Assembler and others, you can use directives for conditional assembly, local labels, and parameter passing to create a wide variety of useful and powerful macros.

Benefits

Macros help you to:

1. *Save typing.* Unless you are a whiz-bang typist (and what programmer is?) you will appreciate this time-saving feature. Most assembler errors are in fact caused by typos, not actual coding errors. Once you define the macro, you can type just a few characters to represent many lines of code.
2. *Remember important code.* For example, how do you execute a two-byte comparison in 65C02 assembly language? The standard algorithm is:

```
LDA NUM1
CMP NUM2
LDA NUM1+1
SBC NUM2+1
```

followed by BCC (where the branch is taken if NUM1 < NUM2) or BCS (where the branch is taken if NUM1 > = NUM2). Even though it's short, the logic of this code is not easy to remember. But the macro CMP2, defined as:

```
CMP2 MAC
        LDA  :A
        CMP  :B
        LDA  :A+1
        SBC  :B+1
        EMC
```

is easy to remember. Once it's defined, the macro helps avert future errors in writing the two-byte comparison.

3. *Simplify programming.* A library of macros will greatly simplify your programming. For example, one of the hardest pieces of code for the beginning assembly language programmer is the seemingly simple PRINT command. The code:

> *Macros are single-word or single-line commands that can replace several lines of assembly source code.*

TABLE 1: Macro Descriptions

Macros	Functions
STOR2	Two-byte store. The syntax STOR2 NUM1,NUM2 (where NUM1 and NUM2 are 16-bit integers) is the equivalent of NUM2 = NUM1 in Applesoft BASIC.
CMP2	Two-byte compare. The syntax CMP2 NUM1,NUM2 compares the values of the 16-bit numbers NUM1 and NUM2. Use BCC or BCS (or alternatively, the macros BLT or BGE; see below) immediately following CMP2.
INC2	Two-byte increment. The syntax INC2 NUM increments the 16-bit number NUM.
DEC2	Two-byte decrement. The syntax DEC2 NUM decrements the 16-bit number NUM.
SETADR	Sets the address of a label to a two-byte variable. The syntax SETADR LABEL,LBLPTR sets the address of LABEL to the variable (pointer) LBLPTR.
HOME	Clears the text screen and moves the cursor to the upper left corner. (This has the same meaning as in Applesoft BASIC.)
INVERSE, NORMAL, FLASH	Function the same as the corresponding Applesoft BASIC commands.
PR	Function the same as the Applesoft PR# command, for example, to turn on a printer (PR#1) or access the 80-column card (PR#3). Use the syntax PR#1 or PR PRNTPRT.
DECPRNT	Prints the decimal value of a two-byte variable. Use the syntax DECPRNT NUM.
HEXPRNT	Prints the hexadecimal value of a two-byte variable. Use the syntax HEXPRNT NUM.
BELL	Sounds the built-in Apple beeper.
CRETURN	Executes a carriage return.
TABHV	Tabs the cursor to the specified horizontal and vertical positions. The syntax TABHV 10,5 moves the cursor to HTAB 10, VTAB 5. This macro requires the subroutine GOTOXY.
PRINT	Prints a string of characters. The syntax PRINT "Hello, world!" prints the string to the current output device. This macro requires the subroutine MSGOUT.
DA	Defines an address (or pointer) and provides two bytes of memory. The syntax DA LABEL is the same as DFC LABEL,LABEL/.
DECIN	Inputs a decimal number. The syntax DECIN NUM makes the computer wait for the user to type a decimal number from the keyboard and then assigns the input value to NUM. The weakness of this macro is that any input error will produce the message "?SYNTAX ERROR" and exit the assembly language program. You should not use this macro except in simple program utilities (like APLPRINT).
BLT	Functions exactly the same as BCC, but its meaning is easier to remember: BLT means "branch if less than." After a comparison (CMP, CPY, CPX, or the macro CMP2), the syntax BLT LABEL causes a branch to the specified label if the value in the 65C02 register is less than the value specified in the operand of the comparison instruction.
BGE	Functions exactly the same as BCS, but its meaning is easier to remember: BGE means "branch if greater than or equal." After a comparison, the syntax BGE causes a branch to the specified label if the value in the 65C02 register is greater than or equal to the value specified in the operand of the comparison instruction.
ZERO	Zeroes the values of the list of one-byte variables. Use the syntax ZERO NUM1, NUM2, NUM3 (with a space between each of the variables, and with no comment on the same line). It uses the complex, recursive macro STAPARM for storing data in a list of parameters. (Study the Macros section of The Assembler user's manual to understand the various pseudo-opcodes included in this macro.)

```
PRINT "I love Nibble"
```

can't be done in assembly language — unless you use macros. The Applesoft code:

```
INPUT A
```

is easy in BASIC but less straightforward in assembly language — until you apply macros. You'll see how to define these macros later.

4. *Streamline listings.* Macros significantly decrease the number of source code lines, saving disk space and making your programs more manageable.
5. *Make code more readable.* Macros help in program readability — an important feature of well designed programs — by allowing more meaningful mnemonics, with names that express the programming function. For example, the macros INVERSE, NORMAL, PRINT, and HOME are easy to remember and easy to read; any Applesoft BASIC programmer immediately knows what they mean. Other macros are also easy to remember and read once you learn them: for example, TABHV tabs the cursor to a horizontal and vertical location on the screen, DECPRNT prints a decimal number, and SETADR stores an address in a memory location. You will also see these macros later.

TABLE 2: APLPRINT Menu Items

Menu Items	Functions
PRINT APPLESOFT PGM	Prints the formatted program listing.
OUTPUT PORT	This is usually set to 1, the typical port to which a printer is connected. Press Return when the highlight bar is at this item to change the value of the output port. For example, you may wish to output the listing to port 3 (the 80-column card) before sending it to the printer to see what the format looks like.
VIDEO PORT	This is the port to which the system returns after printing the listing. This is usually set to port 0 (40-column screen) or port 3 (80-column screen).
LINES/PAGE	Lets you modify the number of *printed* lines on a page. Standard paper is 11 inches long, and most printers output 6 lines per inch, which means a maximum of 66 lines per page. If you want page breaks between pages or margins at the top and bottom of the page, then the actual number of printed lines should be less than 66. The default is 60 lines per page. If you initialize your printer (prior to running APLPRINT) to 8 lines per inch, you can set the LINES/PAGE as high as 88.
LINES SKIPPED	This is the number of lines skipped between the bottom of one page and the top of another. If you change the LINES/PAGE to 54 (which gives a 1-inch margin at the top and bottom of each page), you should set the LINES SKIPPED to 12, so that the sum of the two always equals 66 (for standard 11-inch paper printed at 6 lines per inch).
CHARACTERS/LINE	This is the approximate number of characters (including indent spaces) printed on each line of output. It's not exact because APLPRINT won't split a word in the middle, but continues printing until it finds a space, period, comma, colon or dash, after which APLPRINT outputs a carriage return. For this reason, you should never set the CHARACTERS/LINE to the maximum platen width of your printer. If you initialize your printer (prior to running APLPRINT) to 12, 15, or 17 characters per inch, you can set the CHARACTERS/LINE to about 90, 120, or 130.
SPACES INDENTED	This is the number of blank spaces in the left margin of the printed page.
CONTINUE TAB	This is the number of blank spaces indented for continuation lines (for an Applesoft command line which takes two or more printed lines). Since APLPRINT automatically right-justifies line numbers (so they always take five spaces), the CONTINUE TAB should usually be set to more than 5. The default is 6.
EXIT APLPRINT	This exits the APLPRINT program. You can also exit (or be forced to) by pressing Control-Reset or by typing an illegal quantity for a new APLPRINT value. Once the Applesoft program begins printing (either to a printer or to the video display), you can pause the listing by pressing the Space bar. Pressing the Space bar a second time causes the output to step through the program one line at a time. Pressing Return restores continuous output. You can halt the printing and return to the APLPRINT menu by pressing Escape.

Macros versus Subroutines

A macro is similar to a subroutine in that it represents a set of commands usually used more than once in a program. But a subroutine, if it contains many lines of code, reduces the amount of object code (as well as the amount of source code) in the program. A macro, on the other hand, never reduces the amount of object code, just the amount of source code. During assembly, the macro is expanded into multiple lines of assembly code at every occurrence of the macro. Therefore, if a certain macro represents four lines of assembly language code, the assembler inserts four lines into the program everywhere the macro occurs. The code for a subroutine, on the other hand, occurs only once in each program.

Macros can perform even more magic. You can use directives for conditional assembly, local labels, and parameter passing to create a wide variety of useful and powerful macros.

For this reason, macros usually represent short code which is not amenable to being included in a subroutine. Macros usually take the place of 1-6 lines of code; only very rarely would you define a macro that represented more than 10 lines of code. If you examine the macros in this article, you will see that they would not work as subroutines.

EXAMPLE AND OTHER MACROS

Listing 1 is a source file (in The MicroSPARC Assembler format) of common macros. You will see how to use most of these macros in the example program APLPRINT (**Listing 2**). **Table 1** describes each macro.

The best source of example macros is your own assembler system. All of the popular macro assemblers have examples in their documentation and on their system diskettes.

The ultimate use of macros is Macrosoft, a BASIC-to-machine language system published by MicroSPARC and advertised in most issues of *Nibble*. Macrosoft is actually a complete collection of sophisticated assembly language macros along with a set of predefined subroutines used with The MicroSPARC Assembler. The macros have names closely corresponding to Applesoft BASIC commands: DIM, LET, RND, GOSUB, HOME, TEXT, HGR, etc. Therefore, BASIC programmers can start writing machine code almost immediately, with little training in assembly language.

Of course, when you rely completely on published macros without understanding and using assembly language, you pay a price: The programs are not fully optimized for speed or compactness. One approach is to write code which is not speed or space intensive with the built-in macros, and then use customized assembly code for the critical parts of the program.

Whether you write programs in Macrosoft or not, it's a rich source of information for assembly language programs. If you delve into the source code of Macrosoft's macro files, you will discover how to code for HCOLOR, VTAB, DRAW, SQR, and dozens of other commands in assembly language.

ENTERING APLPRINT

The program APLPRINT (**Listing 2**) demonstrates the power of macros. Its source code uses most of the macros given in **Listing 1**. The function of APLPRINT is to print a formatted listing of an Applesoft program. It works with any printer.

> *Macros help you save typing, remember important codes, simplify programming, streamline listings and make code more readable.*

Before trying to assemble the source code in **Listing 2**, you should type in (but not assemble) **Listing 1** and save the source code under the base name MACROS, which will produce the file MACROS.S on disk. This is a macro *library*, which you can use not only with APLPRINT but with any assembly language program you write. (*Note:* APLPRINT does not use all the macros in **Listing 1**.) As you become more proficient with macros, you can add your favorites to the library (and delete others) to optimize your programming proficiency.

If you don't have The MicroSPARC Assembler, you should try to define the macros in the proper format for your system. Check your user's manual for the proper way to define and use macros with your assembler system. (I hope I'm beginning to sound like a broken record: Read your user's manual, read your user's manual, read your user's manual . . . .)

If you don't have a macro assembler, but still want the program APLPRINT, type in the machine code portion of **Listing 2** and save it with:

```
BSAVE APLPRINT,A$ 9200,L$3F1
```

If you do have The Assembler, type in both **Listing 1** and **Listing 2** and then assemble **Listing 2**. If you are using Key Perfect, BLOAD the object file, delete the file on disk, and BSAVE it using the command shown above.

Using APLPRINT

With an Applesoft program in memory and APLPRINT on the disk, type BRUN APLPRINT. If APLPRINT is already in memory, just type CALL 37376. You will see the APLPRINT menu (as shown in **Figure 1**), with a highlight bar over the first menu item. Use the arrow keys to move the highlight bar up and down the menu. Press Return to select or change the value of the highlighted item. **Table 2** describes each menu item.

How APLPRINT Works

The main task of APLPRINT is reading, translating, and outputting the resident Applesoft program. This is not difficult once you understand the structure of Applesoft and the Applesoft LIST routine. The bibliography at the end of this article gives references with this information.

The code in **lines 64-153** is similar to the LIST command ($D6A5), except APLPRINT maintains control of the listing format, putting carriage returns where APLPRINT wants, not where LIST wants. Every time APLPRINT sends a character to the printer, it uses the subroutines COUNTCHR (**lines 289-313**) and PAGECHK (**lines 277-287**), which count the number of characters per line and the number of lines per page, and formats the output accordingly.

Macros in APLPRINT

You should carefully go through **Listing 2** to see how APLPRINT uses macros. In particular, notice the handy use of HOME, INVERSE, NORMAL, TABHV, and PRINT in formatting the display screen. Also, note that TABHV uses the subroutine GOTOXY in **lines 333-337**, and PRINT uses the subroutine MSGOUT in **lines 339-353** of **Listing 2**. Most importantly, notice how macros make programming easier and improve the readability of assembly listings — almost like magic.

REFERENCES

1. Golding, Val J., "Applesoft From Bottom to Top," in *All About Applesoft*, Call-A.P.P.L.E., Renton, WA, pp. 5-25.

2. Mossberg, Sandy. "Disassembly Lines: LIST and Line Edit," *Nibble*, Vol. 4/No. 1, pp. 161-167.

LISTING 1: MACROS

```
********************************
* MACROS                       *
* BY SCOTT ZIMMERMAN           *
* COPYRIGHT (C) 1987           *
* BY MICROSPARC, INC.          *
* CONCORD, MA 01742            *
********************************
*
STOR2    MAC
         LDA :A
         STA :B
         LDA :A+1
         STA :B+1
         EMC

CMP2     MAC
         LDA :A
         CMP :B
         LDA :A
         SBC :B
         EMC

INC2     MAC
         INC :A
         BNE ]1
         INC :A+1
]1
         EMC

DEC2     MAC
         LDA :A
         BNE ]1
         DEC :A+1
]1       DEC :A
         EMC

SETADR   MAC
         LDA #:A
         STA :B
         LDA #:A/
         STA :B+1
         EMC

HOME     MAC
         JSR $FC58
         EMC

INVERSE  MAC
         JSR $F277
         EMC

NORMAL   MAC
         JSR $F273
         EMC

FLASH    MAC
         JSR $F280
         EMC

PR       MAC
         LDA :A
         JSR $FE95
         EMC

DECPRNT  MAC
         LDA :A+1
         LDX :A
         JSR $ED24
         EMC

HEXPRNT  MAC
         LDA :A+1
         LDX :A
         JSR $F941
         EMC

BELL     MAC
         JSR $FF3A
         EMC

CRETURN  MAC
         JSR $FD8E
         EMC

TABHV    MAC
         LDX #:A
         LDY #:B
         JSR GOTOXY
         EMC

PRINT    MAC
         JSR MSGOUT
         ASC :A
         DFC 0
         EMC

DA       MAC
         DFC :A,:A/
         EMC

DECIN    MAC
         JSR $D52C
         SETADR $200,$B8
         JSR $DD7B
         JSR $E752
         STOR2 $50,:A
         EMC

BLT      MAC
         BCC :A
         EMC

BGE      MAC
         BCS :A
         EMC

STAPARM  MAC
         AIF ":1/
         ALS
         STA :A
         AIF ":0/\
         STAPARM :0/
         AEN
         AEN
         EMC

ZERO     MAC
         LDA #0
         STAPARM :00
         EMC
```

END OF LISTING 1

LISTING 2: APLPRINT

```
                     ;
                     ****************************************************
                     *                                                  *
                     *                    APLPRINT                      *
                     *                                                  *
                     *             By S. Scott Zimmerman                *
                     *              Copyright (c) 1987                  *
                     *              By MicroSPARC, Inc                  *
                     *              Concord, MA  01742                  *
                     *                                                  *
                     *           The MicroSPARC Assembler               *
                     *                                                  *
                     ****************************************************

                              USE MACROS.D2
                              UEN
                              MUL
                              ORG $9200          ;Decimal 37376

                     TEMP     EQU $00            ;Temp data storage
                     ITEMNUM  EQU $02            ;Menu item number
                     TNUM     EQU $06            ;Temporary datum save
                     MENUPTR  EQU $07            ;Menu pointer
                     LINCOUNT EQU $19            ;Line count
                     COLCOUNT EQU $1A            ;Column count
                     CH       EQU $24            ;Cursor horizontal
                     MOD10    EQU $3E            ;Used by DECPRT
                     NUMDIG   EQU $40            ;Number of digits
                     LINNUM   EQU $50            ;Two-byte number
                     TXTTAB   EQU $67            ;Start of Ap program
                     TOKPTR   EQU $9D            ;Ap token pointer
                     TXTPTR   EQU $B8            ;Ap text pointer
                     NEXTLN   EQU $EE            ;Next Ap line pointer
                     APLSOFT  EQU $3D0           ;Ap warm start
                     KEYBD    EQU $C000          ;Keyboard input adrs
                     STROBE   EQU $C010          ;Clear keyboard strobe
                     TOKTBL   EQU $D0D0          ;Ap token table
                     LINPRT   EQU $ED24          ;Decimal number print
                     TABV     EQU $FB5B          ;Vertical tab routine
                     COUT     EQU $FDED          ;Output a character
                     *---------------------------------------------------
                     CR       EQU $8D            ;Carriage return code
                     ESC      EQU $9B            ;Escape code
                     SPACE    EQU $A0            ;Space code
                     LARR     EQU $88            ;Left arrow code
                     DARR     EQU $8A            ;Down arrow code
                     UARR     EQU $8B            ;Up arrow code
                     RARR     EQU $95            ;Right arrow code

                     ****************************************************
                     *  Program beginning:                              *
                     ****************************************************

9200  20 8E FD  START    CRETURN            ;Output a cr
9203  20 FE 92           JSR PRNTMENU       ;Go print the menu
9206  20 47 94           JSR GETMENU        ;Get menu item number

9209  AD EA 95           PR PRNTPORT        ;Set to printer
920C  20 95 FE
920F  A5 67              STOR2 TXTTAB,TXTPTR
9211  85 B8
9213  A5 68
9215  85 B9
9217  A9 00              ZERO LINCOUNT
```

LISTING 2: APLPRINT *(continued)*

```
9219  85 19
921B  20 8E FD            CRETURN           ;Go down a line
921E  20 8E FD            CRETURN           ;Go down another line
9221  2C 10 C0            BIT STROBE        ;Clear keyboard strobe

9224  AD 00 C0  LISTLOOP  LDA KEYBD         ;Has a key been pressed?
9227  10 13               BPL CONTINUE      ;No, just continue
9229  2C 10 C0            BIT STROBE        ;Yes, clear keyboard
922C  C9 9B               CMP #ESC          ;Is it an <ESC>?
922E  F0 2C               BEQ STOPPRNT      ;Yes, so stop print
9230  AD 00 C0  PAUSLOOP  LDA KEYBD         ;Wait for another key
9233  10 FB               BPL PAUSLOOP
9235  C9 8D               CMP #CR           ;Return pressed?
9237  D0 03               BNE CONTINUE      ;No, don't hit strobe
9239  2C 10 C0            BIT STROBE

923C  A9 8D     CONTINUE  LDA #CR           ;Go down a line
923E  20 F7 94            JSR COUNTCHR
9241  A9 00               ZERO COLCOUNT
9243  85 1A
9245  20 DC 94            JSR PAGECHK       ;Go check if end of page
9248  AE EF 95            LDX INDENT        ;Indent all lines
924B  20 43 95            JSR PRBLANKS      ;Print some blanks
924E  20 37 95            JSR CHARGOT       ;Get 1st char of line
9251  85 EE               STA NEXTLN        ;Save in next-line pntr
9253  20 34 95            JSR CHARGET       ;Get next character
9256  85 EF               STA NEXTLN+1
9258  05 EE               ORA NEXTLN        ;See if program end
925A  D0 0C               BNE LIST1         ;Not end, so continue

925C  20 8E FD  STOPPRNT  CRETURN           ;Do a carriage return
925F  AD EB 95            PR EXITPORT       ;Set back to exit
9262  20 95 FE
9265  4C 00 92            JMP START         ;Exit to BASIC

9268  20 34 95  LIST1     JSR CHARGET       ;Get LOB of line number
926B  85 50               STA LINNUM        ;Set LOB for output
926D  20 34 95            JSR CHARGET
9270  85 51               STA LINNUM+1
9272  20 34 95            JSR CHARGET       ;Get next char
9275  C9 00               CMP #0            ;Is it end of line?
9277  D0 0B               BNE LIST2         ;No, so proceed
9279  A5 EE               STOR2 NEXTLN,TXTPTR
927B  85 B8
927D  A5 EF
927F  85 B9
9281  4C 24 92            JMP LISTLOOP      ;Go for next line

9284  A5 51     LIST2     LDA LINNUM+1      ;Get line number
9286  A4 50               LDY LINNUM        ; and ready it for print
9288  20 76 95            JSR DECPRNT       ;Go print decimal
928B  A9 A0               LDA #SPACE        ;Print a space
928D  20 F7 94            JSR COUNTCHR
9290  20 37 95            JSR CHARGOT       ;Get back next char
9293  4C 99 92            JMP CL0

9296  20 34 95  LINELOOP  JSR CHARGET       ;Get line character
9299  C9 00     CL0       CMP #0            ;End of line?
929B  D0 06               BNE CL1           ;No, continue in line
929D  20 34 95            JSR CHARGET       ;Go to 1st char of next
92A0  4C 24 92            JMP LISTLOOP
                CL1
92A3  C9 0D               CMP #$0D          ;Is it a CR?
92A5  D0 06               BNE CL3           ;No, so proceed
92A7  20 1E 95  CL2       JSR DOWN          ;Yes, go down a line
92AA  4C 96 92            JMP LINELOOP      ;Continue in line
92AD  C9 0A     CL3       CMP #$0A          ;CTRL-J, line feed?
92AF  F0 F6               BEQ CL2           ;Yes, go down a line

92B1  C9 80               CMP #%10000000    ;Is hi bit set?
92B3  B0 0E               BGE TOK           ;Yes, so it's a token
92B5  09 80               ORA #%10000000    ;Set hi bit for print
92B7  C9 A0               CMP #SPACE        ;CTRL character?
92B9  B0 02               BGE CL4           ;No, so proceed
92BB  09 60               ORA #%01100000    ;Yes, make lower case
92BD  20 F7 94  CL4       JSR COUNTCHR      ;Go output the char
92C0  4C 96 92            JMP LINELOOP      ;Go do next character

92C3  38        TOK       SEC               ;Token -$7F is position
92C4  E9 7F               SBC #$7F          ; of keyword in table
92C6  AA                  TAX               ;Make an index
92C7  A9 CF               SETADR TOKTBL-1,TOKPTR
92C9  85 9D
92CB  A9 D0
92CD  85 9E
92CF  A0 00               LDY #0            ;Set dummy index
92D1  CA        TOKLOOP   DEX
92D2  F0 0C               BEQ PT1           ;Go print if it
92D4  E6 9D     NXTTOK    INC2 TOKPTR       ;Go to next token
92D6  D0 02
92D8  E6 9E
92DA  B1 9D               LDA (TOKPTR),Y    ;Get character
92DC  10 F6               BPL NXTTOK        ;Nonfinal char is plus
92DE  30 F1               BMI TOKLOOP       ;Final char is minus
92E0  A9 A0     PT1       LDA #SPACE        ;Put space before token
92E2  20 F7 94            JSR COUNTCHR      ;Output it
92E5  E6 9D     PRTTOK    INC2 TOKPTR
92E7  D0 02
92E9  E6 9E
92EB  B1 9D               LDA (TOKPTR),Y    ;Get the char
92ED  30 08               BMI TOK1          ;Print final char
92EF  09 80               ORA #%10000000    ;Set high bit
92F1  20 F7 94            JSR COUNTCHR      ;Go print it
92F4  4C E5 92            JMP PRTTOK        ;Go to next char
92F7  20 F7 94  TOK1      JSR COUNTCHR      ;Print as is
92FA  A9 A0               LDA #SPACE        ;Put a space there
92FC  D0 BF               BNE CL4           ;(Always)

                *********************************************************
                *  PRNTMENU (Print Menu):                               *
                *********************************************************

92FE  20 58 FC  PRNTMENU  HOME              ;Clear screen
9301  20 77 F2            INVERSE           ;Set inverse
9304  A2 0F               TABHV 15,1
9306  A0 01
9308  20 4F 95
930B  20 57 95            PRINT " APLPRINT "
930E  A0 C1 D0
      CC D0 D2 C9 CE D4
      A0
9318  00
9319  20 73 F2            NORMAL            ;Set back to normal
931C  A2 00               LDX #0            ;Init loop counter
931E  86 06     MENULOOP  STX TNUM          ;Save menu item number
9320  20 40 93            JSR PRITEM        ;Go print item
9323  A6 06               LDX TNUM          ;Restore item number
9325  E8                  INX               ;Go to next menu item
9326  E0 09               CPX #9            ;Past last item?
9328  90 F4               BLT MENULOOP      ;No, loop again

932A  AE D5 95  PRIND     LDX OLDNUM        ;Get old menu num
932D  20 40 93            JSR PRITEM        ;Clear inverse video
9330  20 77 F2            INVERSE           ;Set inverse
9333  AE D6 95            LDX MENUNUM       ;Get current menu
9336  8E D5 95            STX OLDNUM        ;Make it old number
9339  20 40 93            JSR PRITEM        ;Print item in inverse
933C  20 73 F2            NORMAL            ;Set back to normal
933F  60                  RTS

9340  86 02     PRITEM    STX ITEMNUM       ;Save item number
9342  18                  CLC               ;Prepare to add
9343  8A                  TXA               ;Put here for add
9344  69 03               ADC #3            ;Calculate VTAB
9346  A8                  TAY               ;Make it Y position
9347  A2 0A               LDX #10           ;HTAB
9349  20 4F 95            JSR GOTOXY        ;Move cursor there
934C  A6 02               LDX ITEMNUM       ;Restore item number
934E  8A                  TXA               ;Put item number in A
934F  0A                  ASL               ;Mult by two
9350  A8                  TAY               ;Make it the index
9351  B9 D7 95            LDA MENUADR,Y     ;Get proper address
9354  85 07               STA MENUPTR       ;Put in pointer
9356  B9 D8 95            LDA MENUADR+1,Y
9359  85 08               STA MENUPTR+1
935B  6C 07 00            JMP (MENUPTR)     ;Go there

935E  20 57 95  PRMEN     PRINT " PRINT APPLESOFT PGM "
9361  A0 D0 D2
      C9 CE D4 A0 C1 D0
      D0 CC C5 D3 CF C6
      D4 A0 D0 C7 CD A0
9376  00
9377  60                  RTS
9378  20 57 95  PPMEN     PRINT " OUTPUT PORT:      "
937B  A0 CF D5
      D4 D0 D5 D4 A0 D0
      CF D2 D4 BA A0 A0
      A0 A0 A0
938D  00
938E  A4 02     MENUCONT  LDY ITEMNUM       ;Get item number
9390  BE E9 95            LDX DEFVAL,Y      ;Get current value
9393  A9 00               LDA #0            ;Zero high byte
9395  20 24 ED            JSR LINPRT        ;Go print value
9398  A9 A0               LDA #SPACE        ;Print a space
939A  4C ED FD            JMP COUT
939D  20 57 95  EPMEN     PRINT " VIDEO PORT:       "
93A0  A0 D6 C9
      C4 C5 CF A0 D0 CF
      D2 D4 BA A0 A0 A0
      A0 A0 A0
93B2  00
93B3  4C 8E 93            JMP MENUCONT      ;Menu continue
93B6  20 57 95  LPMEN     PRINT " LINES/PAGE:       "
93B9  A0 CC C9
      CE C5 D3 AF D0 C1
      C7 C5 BA A0 A0 A0
      A0 A0 A0
93CB  00
93CC  4C 8E 93            JMP MENUCONT      ;Menu continue
93CF  20 57 95  SKMEN     PRINT " LINES SKIPPED:    "
93D2  A0 CC C9
      CE C5 D3 A0 D3 CB
      C9 D0 D0 C5 C4 BA
      A0 A0 A0
93E4  00
93E5  4C 8E 93            JMP MENUCONT      ;Menu continue
93E8  20 57 95  CLMEN     PRINT " CHARACTERS/LINE: "
93EB  A0 C3 C8
      C1 D2 C1 C3 D4 C5
      D2 D3 AF CC C9 CE
      C5 BA A0
93FD  00
93FE  4C 8E 93            JMP MENUCONT      ;Menu continue
9401  20 57 95  INMEN     PRINT " SPACES INDENTED: "
9404  A0 D3 D0
      C1 C3 C5 D3 A0 C9
      CE C4 C5 CE D4 C5
      C4 BA A0
9416  00
9417  4C 8E 93            JMP MENUCONT      ;Menu continue
941A  20 57 95  TBMEN     PRINT " CONTINUE TAB:     "
941D  A0 C3 CF
```

```
      CE D4 C9 CE D5 C5
      A0 D4 C1 C2 BA A0
      A0 A0 A0
942F  00
9430  4C 8E 93            JMP MENUCONT      ;Menu continue
9433  20 57 95  EXMEN     PRINT " EXIT APLPRINT "
9436  A0 C5 D8
      C9 D4 A0 C1 D0 CC
      D0 D2 C9 CE D4 A0
9445  00
9446  60                  RTS

                *****************************************************
                * GETMENU (Get menu item):                          *
                *****************************************************

9447  AD 00 C0  GETMENU   LDA KEYBD         ;Check keyboard
944A  10 FB               BPL GETMENU       ;Not pressed
944C  2C 10 C0            BIT STROBE        ;Pressed, clear strobe
944F  C9 8D               CMP #CR           ;Carriage return?
9451  D0 1E               BNE GM1           ;No, check next
9453  AD D6 95            LDA MENUNUM       ;Is it ready to print
9456  F0 12               BEQ PRINTIT       ;It's zero, so print
9458  C9 08               CMP #8            ;Is it exit?
945A  D0 0F               BNE GOSET         ;No, set new value
945C  AD E8 95            PR EXITPORT
945F  20 95 FE
9462  20 58 FC            HOME
9465  4C D0 03            JMP APLSOFT       ;Exit to BASIC
9468  D0 01               BNE GOSET         ;No, so set number
946A  60        PRINTIT   RTS               ;Yes, return to print
946B  20 A2 94  GOSET     JSR GETVAL        ;Go get value
946E  4C 47 94            JMP GETMENU       ;Get next key
9471  C9 95     GM1       CMP #RARR         ;Right arrow?
9473  D0 13               BNE GM2           ;No, check next
9475  AE D6 95  DNIND     LDX MENUNUM       ;Get current menu num
9478  E8                  INX               ;Go to next
9479  E0 09               CPX #9            ;Past last?
947B  90 02               BLT SETNEW        ;No, it's okay
947D  A2 00               LDX #0            ;Yes, set back to zero
947F  8E D6 95  SETNEW    STX MENUNUM       ;Save new menu num
9482  20 2A 93            JSR PRIND         ;Print next indic
9485  4C 47 94            JMP GETMENU       ;Get next key
9488  C9 8A     GM2       CMP #DARR         ;Down arrow?
948A  F0 E9               BEQ DNIND         ;Yes, go down one
948C  C9 88               CMP #LARR         ;Left arrow?
948E  D0 0B               BNE GM4           ;No, proceed
9490  AE D6 95  UPIND     LDX MENUNUM       ;Get current menu num
9493  CA                  DEX               ;Go to previous
9494  10 02               BPL GM3           ;Value okay ; proceed
9496  A2 08               LDX #8            ;Value neg ; set to end
9498  4C 7F 94  GM3       JMP SETNEW        ;Set new menu item
949B  C9 8B     GM4       CMP #UARR         ;Up arrow?
949D  F0 F1               BEQ UPIND         ;Yes, move menu up
949F  4C 47 94            JMP GETMENU       ;No selection

94A2  A2 0B     GETVAL    TABHV 11,13
94A4  A0 0D
94A6  20 4F 95
94A9  20 57 95            PRINT "NEW VALUE: "
94AC  CE C5 D7
      A0 D6 C1 CC D5 C5
      BA A0
94B7  00
94B8  20 2C D5            DECIN TEMP
94BB  A9 00
94BD  85 B8
94BF  A9 02
94C1  85 B9
94C3  20 7B DD
94C6  20 52 E7
94C9  A5 50
94CB  85 00
94CD  A5 51
94CF  85 01
94D1  AC D6 95            LDY MENUNUM       ;Make menu num an index
94D4  A5 00               LDA TEMP          ;Get new value
94D6  99 E9 95            STA DEFVAL,Y      ;Save new value
94D9  4C FE 92            JMP PRNTMENU      ;Print menu again

                *****************************************************
                * Short subroutines:                                *
                *****************************************************

94DC  E6 19     PAGECHK   INC LINCOUNT      ;Go to next line
94DE  A5 19               LDA LINCOUNT      ;Get the num lines
94E0  CD EC 95            CMP LINESPP       ;Is it end of page?
94E3  90 11               BLT EXIT          ;No, so print line
94E5  AE ED 95            LDX SKIP          ;Skip down some lines
94E8  F0 0C               BEQ EXIT          ;If zero, don't do creturn
94EA  20 8E FD  PGLOOP    CRETURN           ;Output a carriage return
94ED  CA                  DEX               ;End of spaces?
94EE  D0 FA               BNE PGLOOP        ;No, go do another
94F0  A9 00               ZERO LINCOUNT, COLCOUNT
94F2  85 19
94F4  85 1A
94F6  60        EXIT      RTS               ;Return from PAGECHK
                *---------------------------------------------------*
94F7  8D F1 95  COUNTCHR  STA SAVECHAR      ;Save output char
94FA  20 ED FD            JSR COUT          ;Output the character
94FD  E6 1A               INC COLCOUNT      ;Go to next character
94FF  A5 1A               LDA COLCOUNT      ;Get column count?
9501  CD EE 95            CMP CHRSPL        ;Is it end of line?
9504  B0 01               BGE CC1           ;Yes, see if break
9506  60        CC0       RTS               ;No, just return
9507  AD F1 95  CC1       LDA SAVECHAR      ;Restore character
950A  C9 A0               CMP #SPACE        ;Check if it is a
950C  F0 10               BEQ DOWN          ; character at which
950E  C9 AC               CMP #",           ; a break can be made
9510  F0 0C               BEQ DOWN
9512  C9 BA               CMP #":
9514  F0 08               BEQ DOWN
9516  C9 AE               CMP #".
9518  F0 04               BEQ DOWN
951A  C9 AD               CMP #"-
951C  D0 E8               BNE CC0
951E  A9 00     DOWN      ZERO COLCOUNT
9520  85 1A
9522  20 8E FD            CRETURN           ;Ouput a carriage return
9525  20 DC 94            JSR PAGECHK       ;Go check page
9528  AE EF 95            LDX INDENT        ;Indent all lines
952B  20 43 95            JSR PRBLANKS      ;Print some blanks
952E  AE F0 95            LDX TAB           ;Tab continuation lines
9531  4C 43 95            JMP PRBLANKS      ;Print some blanks
                *---------------------------------------------------*
9534  20 3C 95  CHARGET   JSR INCTXTP       ;Increment text pointer
9537  A2 00     CHARGOT   LDX #0
9539  A1 B8               LDA (TXTPTR,X)
953B  60                  RTS
                *---------------------------------------------------*
953C  E6 B8     INCTXTP   INC TXTPTR
953E  D0 02               BNE FIN
9540  E6 B9               INC TXTPTR+1
9542  60        FIN       RTS
                *---------------------------------------------------*
9543  CA        PRBLANKS  DEX               ;Was it zero?
9544  30 08               BMI ENDBLNK       ;Yes, no blanks
9546  A9 A0     BLNKLOOP  LDA #SPACE        ;Get a "blank" char
9548  20 F7 94            JSR COUNTCHR      ;Go print with COUT
954B  CA                  DEX               ;End of blanks?
954C  10 F8               BPL BLNKLOOP      ;No, loop again
954E  60        ENDBLNK   RTS
                *---------------------------------------------------*
954F  CA        GOTOXY    DEX               ;Set range start at 0
9550  86 24               STX CH            ;Set horizontal tab
9552  88                  DEY               ;Set range start at 0
9553  98                  TYA               ;Put vertical tab here
9554  4C 5B FB            JMP TABV          ;Go tab there
                *---------------------------------------------------*
9557  68        MSGOUT    PLA               ;Pull LOB return address
9558  85 00               STA TEMP          ;Save temporarily
955A  68                  PLA               ;Pull HOB return address
955B  85 01               STA TEMP+1        ;Save it temporarily
955D  A0 00               LDY #0            ;Init string index
955F  E6 00     MSGLOOP   INC2 TEMP         ;Incr RTS adr for char
9561  D0 02
9563  E6 01
9565  B1 00               LDA (TEMP),Y      ;Get character
9567  F0 06               BEQ MSGRTS        ;If zero, end of string
9569  20 ED FD            JSR COUT          ;Output it
956C  4C 5F 95            JMP MSGLOOP       ;Get next char
956F  A5 01     MSGRTS    LDA TEMP+1        ;Get HOB of RTS
9571  48                  PHA               ;Push back onto stack
9572  A5 00               LDA TEMP          ;Get LOB of RTS
9574  48                  PHA               ;Push it onto stack
9575  60                  RTS               ;Return to there

                *****************************************************
                * DECPRT (decimal printer):                         *
                *****************************************************

9576  A9 00     DECPRNT   ZERO NUMDIG
9578  85 40
957A  85 40               STA NUMDIG

957C  A9 00     CONVERT   ZERO MOD10, MOD10+1
957E  85 3E
9580  85 3F
9582  A2 10               LDX #16           ;16-bits to divide by 10
9584  18                  CLC
9585  26 50     DIVLOOP   ROL LINNUM        ;Do division by 10
9587  26 51               ROL LINNUM+1
9589  26 3E               ROL MOD10         ;Keep track of remainder
958B  26 3F               ROL MOD10+1
958D  38                  SEC               ;Prepare to subtract
958E  A5 3E               LDA MOD10
9590  E9 0A               SBC #10
9592  A8                  TAY               ;Save LOB
9593  A5 3F               LDA MOD10+1
9595  E9 00               SBC #0
9597  90 04               BLT DECCNT
9599  84 3E               STY MOD10
959B  85 3F               STA MOD10+1
959D  CA        DECCNT    DEX               ;Go to next bit
959E  D0 E5               BNE DIVLOOP       ;Not done, so continue

95A0  26 50               ROL LINNUM        ;Shift in last carry
95A2  26 51               ROL LINNUM+1

95A4  E6 40               INC NUMDIG
95A6  A5 3E               LDA MOD10
95A8  18                  CLC
95A9  69 B0               ADC #"0           ;Add ASCII zero
95AB  A4 40               LDY NUMDIG
95AD  99 F2 95            STA DIGBUFF,Y     ;Save ASCII digit
95B0  A5 50               LDA LINNUM        ;See if value now zero
95B2  05 51               ORA LINNUM+1
95B4  D0 C6               BNE CONVERT       ;No, so do next digit

                * Print leading blanks:

95B6  38                  SEC               ;Calc number blanks to
95B7  A9 05               LDA #5            ; right justify number
95B9  E5 40               SBC NUMDIG        ;Subtract number digits
95BB  F0 09               BEQ PRDEC         ;None, so don't pad
95BD  A8                  TAY               ;Make # blanks a counter
95BE  A9 A0     BLLOOP    LDA #SPACE        ;Get ASCII for blank
```

LISTING 2: APLPRINT *(continued)*

```
95C0  20 F7 94            JSR COUNTCHR     ;Output it and count
95C3  88                  DEY              ;Go to next blank
95C4  D0 F8               BNE BLLOOP

                 *  Print the digits:

95C6  A4 40      PRDEC    LDY NUMDIG
95C8  A2 01               LDX #1
95CA  B9 F2 95   DECLOOP  LDA DIGBUFF,Y    ;Get ASCII digit char
95CD  20 F7 94            JSR COUNTCHR     ;Print to screen
95D0  E8                  INX              ;Point to next buff loc
95D1  88                  DEY              ;End of string?
95D2  D0 F6               BNE DECLOOP      ;No, continue
95D4  60                  RTS              ;Done!

                 ***************************************************
                 *  Data and variables:                            *
                 ***************************************************

95D5  00         OLDNUM   DFC 0            ;Old menu number
95D6  00         MENUNUM  DFC 0            ;Menu item number
                 *-------------------------------------------------*
95D7  5E 93      MENUADR  DA PRMEN
95D9  78 93               DA PPMEN
95DB  9D 93               DA EPMEN
95DD  B6 93               DA LPMEN
95DF  CF 93               DA SKMEN
95E1  E8 93               DA CLMEN
95E3  01 94               DA INMEN
95E5  1A 94               DA TBMEN
95E7  33 94               DA EXMEN
                 *-------------------------------------------------*
95E9  01         DEFVAL   DFC 1            ;Start of list data
95EA  01         PRNTPORT DFC 1            ;Printer port
95EB  00         EXITPORT DFC 0            ;Exit port
95EC  3C         LINESPP  DFC 60           ;Lines printed/page
95ED  06         SKIP     DFC 6            ;Lines skip at page bot
95EE  48         CHRSPL   DFC 72           ;Characters per line
95EF  03         INDENT   DFC 3            ;Amnt indent every line
95F0  06         TAB      DFC 6            ;Continuation line tab
                 *-------------------------------------------------*
                 SAVECHAR DFS 1            ;Save output character
                 DIGBUFF  DFS 6            ;Up to 5 dec digits

000  Errors

9200  Hex Start of Object
95F7  Hex end of Object
03F8  Hex Length of Object
73D9  Hex end of Symbols
```

END OF LISTING 2

```
               KEY PERFECT 5.0
                    RUN ON
                   APLPRINT
============================================
  CODE-5.0    ADDR# - ADDR#     CODE-4.0
  --------    -------------     --------
  13DDF062     9200 - 924F        2AC4
  865C75F2     9250 - 929F        2584
  E1CF2443     92A0 - 92EF        2687
  FDB5F644     92F0 - 933F        2BA3
  A19091F6     9340 - 938F        2694
  20D243C5     9390 - 93DF        26C2
  5E084D7B     93E0 - 942F        25EA
  645ECAD9     9430 - 947F        2736
  0459B770     9480 - 94CF        2775
  D80EF0C2     94D0 - 951F        290E
  5AC9AA25     9520 - 956F        2752
  D12EB208     9570 - 95BF        2A43
  68ED82AA     95C0 - 95F0        1A9A
  7DEAAA36 = PROGRAM TOTAL =      03F1
```