APPLESOFT WINDOWS

TIPS 'N TECHNIQUES

It's easy to do ***80-column windowing from Applesoft BASIC, on the Apple IIe or IIc, with this useful routine!***

When programmers have wanted to put messages on the screen, they have traditionally cleared lines or a section on the screen for them. However, then they must find a way to replace the original screen information. Reserving a line or two for messages would be a waste of valuable screen space. The solution is Applesoft Windows.

The binary utility Applesoft Windows (**Listing 1**) does the following:

1. Clears a screen area for messages or input
2. Saves the screen characters in a safe memory area
3. Draws a box outline at the window's perimeter
4. Restores the screen to its original state after the message or input

USING APPLESOFT WINDOWS

To better understand how to use Applesoft Windows in your own programs, I've included a short demonstration program (**Listing 2**). **Lines 80-120** set up parameters for the demo program and load the binary program Applesoft Windows. Then the program lists itself on the screen to provide a screenful of characters on which Applesoft Windows is demonstrated.

Clearing the Window

Clearing the screen window is done in **line 140**. This CALL is the only statement you will need in your programs in order to create a window. Its format is:

CALL WINDOW,*VS,VE,HS,HE*,0

where VS is the vertical screen line (1-23) of the window's top edge, VE is the vertical screen line (2-24) of the window's bottom edge, HS is the horizontal screen line (1-78) of the window's left edge, and HE is the horizontal screen line (3-80) of the window's right edge. The zero is the Clear/Restore flag that indicates clearing the screen or restoring the previous information.

ProDOS requires more room for file buffers and that HIMEM be on an even-page boundary.

In **Listing 2**, **line 140** creates a window that contains vertical screen lines 3-10 and horizontal screen lines 10-60. That is, the box is 8 lines tall and 51 characters wide.

In your own program, the next step after clearing a window is to print your screen message or get input. In the demonstration program, this is done in **lines 150-160**.

Restoring the Screen

To restore the original text to the screen (**line 170** of **Listing 2**), simply do the same CALL with one exception. The Clear/Restore flag that was zero for clearing must now be a one. Therefore, the format to restore the screen is:

CALL WINDOW,*VS,VE,HS,HE*,1

where parameters VS, VE, HS and HE are the same as those used to clear.

Syntax error checking is done within the binary program WINDOWS to ensure that VS < VE, VE < 25, HS + 1 < HE, HE < 81, and that your program is in 80-column mode. If all of these conditions are not met, a SYNTAX ERROR message will be displayed.

TABLE 1: Changes for Unenhanced IIe

Address	New Value
$94D5	$AD
$95AB	$AD
$9563	$01
$9564	$CF
$956F	$F2
$9570	$CE
$95CB	$F2
$95CC	$CE
$95D1	$F2
$95D2	$CE
$95EA	$F2
$95EB	$CE

Note that other than doing the CALL to WINDOW to clear the screen text and the CALL to restore the screen text, you must also set HIMEM and load the binary program itself from within your program.

The Inverse Window

Listing 2 also creates a second type of window, an inverse box. The format is the same as that of a "normal" window with one exception — the CALL is to INVRS instead of WINDOW:

CALL INVRS,*VS*,*VE*,*HS*,*HE*,0

INVRS is at decimal memory location 38078, and the zero parameter for the Clear/Restore flag is always used. To restore the original screen after a CALL to INVRS, perform a CALL to WINDOW as before:

CALL WINDOW,*VS*,*VE*,*HS*,*HE*,1

The Memory Buffer

The Applesoft Windows memory buffer in which screen characters are saved is dynamic. That is, it changes from CALL to CALL depending upon the location of your program's variables. The buffer is indexed from zero page locations $01-$02, and is set at each CALL to WINDOW.

TABLE 2: Changes for Enhanced IIe

Address	New Value
$9563	$44
$9564	$CE
$956F	$38
$9570	$CE
$95CB	$38
$95CC	$CE
$95D1	$38
$95D2	$CE
$95EA	$38
$95EB	$CE

The code in **Listing 1** at addresses $9526-$953E sets up address $0 (zero) to hold the number of characters in the window. It also sets up $01-$02 to an even-page boundary centered between the end of the numeric variables memory area (STREND pointer at $6D-$6E) and the start of string variable memory (FRETOP pointer at $6F-$70). Setting up the buffer in this way has three advantages.

1. It is dynamic — no fixed memory location need be dedicated.
2. Its size is only limited by half the memory from the end of the numeric variables to the start of the string variables.
3. There is memory remaining for the creation of new variables while the window is in effect.

ENTERING THE PROGRAM

To enter Applesoft Windows, you may either use an assembler to assemble the program in **Listing 1**, or type in the hex code directly from the Apple Monitor. Save the program with:

BSAVE WINDOWS,A$94BE,L$142

Next, type in the Applesoft program in **Listing 2**, and save it with:

SAVE WINDOWS.DEMO

For help with entering *Nibble* programs, see the directions in the Program Listings section of this magazine. If you have a IIe, please see the section titled Modifications for the IIe before running the program.

If you are working with ProDOS, change the value of HIMEM in **line 80** to 36864 ($9000). This frees up more room for the ProDOS general purpose buffer and sets HIMEM on an even-page boundary.

Modifications for the IIe

Applesoft Windows uses the Apple IIc's built-in 80-column binary routines. It also makes calls to memory locations in Apple's firmware code, specifically PICK and STORE. To use Applesoft Windows with the IIc, you need not make any changes to the code in **Listing 1**.

To use the program on a IIe, you must make a few changes. If you have an unenhanced IIe, enter **Listing 1** as described above. Then enter the Monitor and make the changes shown in **Table 1**. Save the program again with:

BSAVE WINDOWS,A$94BE,L$142

If you have an enhanced IIe, enter the Monitor and make the changes shown in **Table 2**. Then re-save the program with the same command as shown above.

Happy windowing...and may it be paneless!

Applesoft Windows listings start on page 99

Applesoft Windows

Article on page 46

THIS PROGRAM IS AVAILABLE ON DISK

If you'd rather not type in the listing for this program, you can buy it on disk, complete, free of typos and ready to run. Applesoft Windows, Hi-Res SCRN Command, DOS Device Detective and A Matter of Timing demo programs are available on disk for an introductory price of $17.95 plus $1.50 shipping/handling ($2.50 outside the U.S.) from Nibble, 45 Winthrop St., Concord, MA 01742. Introductory price expires 5/31/87. See the coupon on the last page of the Nibble Software Catalog for ordering information.

Listing 1 for Applesoft Windows

WINDOWS

```
                    1   ************************
                    2   * WINDOWS              *
                    3   * BY LARRY ABRAMS      *
                    4   * COPYRIGHT (C) 1987   *
                    5   * BY MICROSPARC, INC.  *
                    6   * CONCORD, MA  01742   *
                    7   ************************
                    8   *Merlin assembler      *
                    9   ************************
                   10   *
                   11   *       EQUATES
                   12   *
                   13   NUMCHRS  =   $00
                   14   BUFFLO   =   $01
                   15   BUFFHI   =   $02
                   16   TEMPHI   =   $03
                   17   LINNUM   =   $50
                   18   STREND   =   $6E
                   19   FRETOP   =   $70
                   20   VMODE    =   $4FB
                   21   OURCH    =   $57B
                   22   CLRCHR   =   $956A
                   23   BOXTOP   =   $95AB
                   24   BOXSIDE  =   $95C6
                   25   BOXBTM   =   $95E5
                   26   TMPVMODE =   $95F9
                   27   VSBOX    =   $95FA
                   28   CPFLAG   =   $95FB
                   29   HZEND    =   $95FC
                   30   HZSTRT   =   $95FD
                   31   VTEND    =   $95FE
                   32   VTSTRT   =   $95FF
                   33   RD80STOR =   $C018
                   34   STORE    =   $C3C1     ;CEF2 ON IIE, OR CE38 ON ENH. IIE
                   35   PICK     =   $CC1D     ;CF01 ON IIE, OR CE44 ON ENH. IIE
                   36   FRMNUM   =   $DD67
                   37   CHKCOM   =   $DEBE
                   38   SNERR    =   $DEC9
                   39   GETADR   =   $E752
                   40   BASCALC  =   $FBC1
                   41   *
                   42            ORG   $94BE
                   43   *
94BE: A9 20        44   INVRS    LDA   #$20     ;Inverse space.
94C0: 8D 6A 95     45            STA   CLRCHR   ;Clearing char. inverse.
94C3: 8D AB 95     46            STA   BOXTOP   ;Box top inverse space.
94C6: 8D C6 95     47            STA   BOXSIDE  ;Box side inverse space.
94C9: 8D E5 95     48            STA   BOXBTM   ;Box bottom inverse space.
94CC: 20 E4 94     49            JSR   WINDOW   ;Window is inverse box.
94CF: A9 A0        50            LDA   #$A0     ;Normal space character.
94D1: 8D 6A 95     51            STA   CLRCHR   ;Put back original.
94D4: A9 4C        52            LDA   #$4C     ;"Overline" character.
94D6: 8D AB 95     53            STA   BOXTOP   ;Put back original.
94D9: A9 FC        54            LDA   #$FC     ;"Vertical bar" character.
94DB: 8D C6 95     55            STA   BOXSIDE  ;Put back original.
94DE: A9 DF        56            LDA   #$DF     ;"Underline" character.
94E0: 8D E5 95     57            STA   BOXBTM   ;Put back original.
94E3: 60           58            RTS            ;Return to BASIC.
                   59   *
94E4: A2 05        60   WINDOW   LDX   #05      ;Index to get 5 parameters.
94E6: 86 00        61   W1       STX   NUMCHRS  ;Save index.
94E8: 20 BE DE     62            JSR   CHKCOM   ;Process comma.
94EB: 20 67 DD     63            JSR   FRMNUM   ;Evaluate number at TXTPTR.
94EE: 20 52 E7     64            JSR   GETADR   ;Make LINNUM an integer.
94F1: A5 50        65            LDA   LINNUM   ;Get number.
94F3: A6 00        66            LDX   NUMCHRS  ;Get index.
94F5: 9D FA 95     67            STA   VSBOX,X  ;Save VS,VE,HS,HE or CPFLAG.
94F8: CA           68            DEX            ;Decrement index.
94F9: D0 EB        69            BNE   W1       ;Not done yet.
94FB: AD 18 C0     70            LDA   RD80STOR ;Check for 80 column on.
94FE: 29 80        71            AND   #$80     ;Mask off bits 0-6.
9500: F0 21        72            BEQ   ERROR    ;80 column off, syntax error.
9502: AD FF 95     73            LDA   VTSTRT   ;Get VPOS start line.
```

Listing 1 for Applesoft Windows

WINDOWS *(continued)*

```
9505: CD FE 95  74            CMP   VTEND       ;Compare it to VPOS end line.
9508: B0 19     75            BCS   ERROR       ;VPOS start >= end, Syntax error.
950A: AD FE 95  76            LDA   VTEND       ;Get VPOS end line.
950D: C9 19     77            CMP   #$19        ;Is vertical end line > 24?
950F: B0 12     78            BCS   ERROR       ;Yes, Syntax error.
9511: 18        79            CLC
9512: AD FD 95  80            LDA   HZSTRT      ;Adjust HPOS start line for box.
9515: 6D 01     81            ADC   #$01
9517: CD FC 95  82            CMP   HZEND       ;Compare it against HPOS end line.
951A: B0 07     83            BCS   ERROR       ;HPOS start >= end, Syntax error.
951C: AD FC 95  84            LDA   HZEND       ;Get HPOS end line.
951F: C9 51     85            CMP   #$51        ;Is horiz line >80?
9521: 90 03     86            BCC   W2          ;No, everything OK.
9523: 4C C9 DE  87  ERROR     JMP   SNERR       ;Do Apple's SYNTAX ERROR and stop.
9526: A9 00     88  W2        LDA   #0
9528: 85 01     89            STA   BUFFLO      ;Set buffer to even page boundary.
952A: AD FB 95  90            LDA   CPFLAG      ;Going to 0=Clear or 1=Put Back?
952D: F0 06     91            BEQ   W3          ;Clearing, so calculate BUFFHI.
952F: A5 03     92            LDA   TEMPHI      ;Get BUFFHI from previous clear.
9531: 85 02     93            STA   BUFFHI      ;Save it.
9533: 10 0A     94            BPL   W4          ;Always.
9535: 18        95  W3        CLC
9536: A5 6E     96            LDA   STREND      ;Get numeric vars. end high byte.
9538: 65 70     97            ADC   FRETOP      ;Add string vars. start high byte.
953A: 4A        98            LSR               ;Divide by two to get BUFFHI.
953B: 85 02     99            STA   BUFFHI      ;Save it.
953D: 85 03     100           STA   TEMPHI      ;Save it for putting back.
953F: CE FD 95  101 W4        DEC   HZSTRT      ;Our HPOS 1 is computer's HPOS 0.
9542: CE FF 95  102           DEC   VTSTRT      ;OUR VPOS 1 is computer's VPOS 0.
9545: AD FF 95  103           LDA   VTSTRT      ;Get our VPOS start line
9548: 8D FA 95  104           STA   VSBOX       ; and keep it for box draw.
954B: AD FF 95  105           LDA   VTSTRT      ;Get VPOS start line.
954E: 20 C1 FB  106 W5        JSR   BASCALC     ;Calc VPOS line address.
9551: AC FD 95  107           LDY   HZSTRT      ;Get HPOS for line.
9554: 8C 7B 05  108 W6        STY   OURCH       ;80 column HPOS.
9557: AD FB 95  109           LDA   CPFLAG      ;Going to 0=Clear or 1=Put Back?
955A: F0 06     110           BEQ   CLEAR       ;We are clearing a box.
955C: A4 00     111           LDY   NUMCHRS     ;Index for buffer.
955E: B1 01     112           LDA   (BUFFLO),Y  ;Get a character from buffer.
9560: D0 09     113           BNE   W7          ;Always.
9562: 20 1D CC  114 CLEAR     JSR   PICK        ;Get a character from screen.
9565: A4 00     115           LDY   NUMCHRS     ;Get buffer index.
9567: 91 01     116           STA   (BUFFLO),Y  ;Save character in buffer.
9569: A9 A0     117           LDA   #$A0        ;Clearing space.
956B: AC 7B 05  118 W7        LDY   OURCH       ;Get HPOS.
956E: 20 C1 C3  119           JSR   STORE       ;Put character on screen.
9571: E6 00     120           INC   NUMCHRS     ;Increment buffer index.
9573: C8        121           INY               ;Increment HPOS.
9574: CC FC 95  122           CPY   HZEND       ;Finished this line?
9577: D0 DB     123           BNE   W6          ;No.
9579: 18        124           CLC
957A: A5 01     125           LDA   BUFFLO      ;Update buffer location to
957C: 65 00     126           ADC   NUMCHRS     ; handle page crossing.
957E: 85 01     127           STA   BUFFLO
9580: 90 02     128           BCC   W8
9582: E6 02     129           INC   BUFFHI
9584: A9 00     130 W8        LDA   #0
9586: 85 00     131           STA   NUMCHRS     ;Clear NUMCHRS.
9588: EE FF 95  132           INC   VTSTRT      ;Increment screen line.
958B: AD FF 95  133           LDA   VTSTRT      ;Get new screen line.
958E: CD FE 95  134           CMP   VTEND       ;Are we finished?
9591: D0 BB     135           BNE   W5          ;No.
9593: AD FB 95  136           LDA   CPFLAG      ;Get Clear/Put-Back Flag.
9596: F0 01     137           BEQ   BOX         ;Clearing, so draw box.
9598: 60        138           RTS               ;Return. (Putting back.)
                139 *
9599: AD FB 04  140 BOX       LDA   VMODE       ;Get existing VMODE.
959C: 8D F9 95  141           STA   TMPVMODE    ;Save it.
959F: 29 FE     142           AND   #$FE        ;Clear zero bit.
95A1: 8D FB 04  143           STA   VMODE       ;Set alternate character set.
95A4: AD FA 95  144           LDA   VSBOX       ;Get VTSTRT for box.
95A7: 20 C1 FB  145           JSR   BASCALC     ;Calc VPOS line address.
95AA: A9 4C     146           LDA   #$4C        ;"Overline" character.
95AC: 20 E6 95  147           JSR   DRAW        ;Draw vertical line.
95AF: AD FB 04  148           LDA   VMODE
95B2: 09 01     149           ORA   #$01        ;Set zero bit.
95B4: 8D FB 04  150           STA   VMODE       ;Set primary character set.
95B7: CE FC 95  151           DEC   HZEND       ;Our HPOS 1 is computer's HPOS 0.
95BA: AD FA 95  152 SIDES     LDA   VSBOX       ;Get VPOS.
95BD: CD FE 95  153           CMP   VTEND       ;Finished drawing sides?
95C0: F0 16     154           BEQ   BOTTOM      ;Yes, go draw bottom.
95C2: 20 C1 FB  155           JSR   BASCALC     ;Calc VPOS line address.
95C5: A9 FC     156           LDA   #$FC        ;"Vertical bar" character.
95C7: AC FD 95  157           LDY   HZSTRT      ;Box left edge.
95CA: 20 C1 C3  158           JSR   STORE       ;Put on screen.
95CD: AC FC 95  159           LDY   HZEND       ;Box right edge.
95D0: 20 C1 C3  160           JSR   STORE       ;Put on screen.
95D3: EE FA 95  161           INC   VSBOX       ;Increment VPOS.
95D6: D0 E2     162           BNE   SIDES       ;Always.
95D8: CE FE 95  163 BOTTOM    DEC   VTEND       ;Our VPOS 1 is computer's VPOS 0.
95DB: EE FD 95  164           INC   HZSTRT      ;Put back original HZSTRT.
95DE: AD FE 95  165           LDA   VTEND       ;Get VPOS for bottom of box.
95E1: 20 C1 FB  166           JSR   BASCALC     ;Calc VPOS line address.
95E4: A9 DF     167           LDA   #$DF        ;"Underline" character.
95E6: AC FD 95  168 DRAW      LDY   HZSTRT      ;Box left edge.
95E9: 20 C1 C3  169 D1        JSR   STORE       ;Put on screen.
95EC: C8        170           INY               ;Increment HPOS.
95ED: CC FC 95  171           CPY   HZEND       ;Finished drawing bottom?
95F0: D0 F7     172           BNE   D1          ;No.
95F2: AD F9 95  173           LDA   TMPVMODE    ;Get original VMODE back.
95F5: 8D FB 04  174           STA   VMODE       ;Save it.
95F8: 60        175           RTS               ;Return. Done with box.
                176 *
95F9: 00        177           HEX   00          ;TMPVMODE.
95FA: 00        178           HEX   00          ;VSBOX.
95FB: 00        179           HEX   00          ;CPFLAG.
95FC: 00 00 00  180           HEX   00000000    ;HZEND,HZSTRT,VTEND,VTSTRT.
95FF: 00

--End assembly, 322 bytes, Errors: 0
```

END OF LISTING 1

```
KEY PERFECT 5.0
RUN ON
WINDOWS

=====================================
  CODE-5.0    ADDR# - ADDR#    CODE-4.0
  --------    -------------    --------
  97252F5B     94BE - 950D       27E0
  BF3BD3BA     950E - 955D       253F
  4B1F9129     955E - 95AD       292A
  0BAC9385     95AE - 95FD       2884
  00ED26BE     95FE - 95FF         00
  CBF8D2FC = PROGRAM TOTAL =     0142
```

Listing 2 for Applesoft Windows

WINDOWS.DEMO

```
10  REM  ************************
20  REM  * WINDOWS.DEMO         *
30  REM  * BY LARRY ABRAMS      *
40  REM  * COPYRIGHT (C) 1987   *
50  REM  * BY MICROSPARC, INC.  *
60  REM  * CONCORD, MA  01742   *
70  REM  ************************
80  PRINT  CHR$ (4)"PR#3"
90  PRINT "WINDOWS DEMONSTRATION": PRINT "BY
    LARRY ABRAMS": PRINT "COPYRIGHT 1987 BY
    MICROSPARC, INC.": PRINT
100  PRINT : ONERR  GOTO 240
110  PRINT  CHR$ (4)"BLOAD WINDOWS": HIMEM: 3
    6864
120  POKE 216,0:WINDOW = 38116:INVRS = 38078:
     LIST
130  REM  WINDOW DEMO
140 VS = 3:VE = 10:HS = 10:HE = 60: CALL WIND
    OW,VS,VE,HS,HE,0
150  VTAB 4: POKE 1403,15: PRINT "This window
     was invoked from BASIC by:": VTAB 6: POKE
     1403,20: PRINT "CALL WINDOW,VS,VE,HS,HE,
    0"
160  VTAB 9: POKE 1403,13: PRINT "Press any k
    ey for INVERSE WINDOW or ";: INVERSE : PRINT
    "Q";: NORMAL : PRINT "uit: ";: GET A$
170  CALL WINDOW,VS,VE,HS,HE,1
180  IF A$ =  CHR$ (81) OR A$ =  CHR$ (113) THEN
     VTAB 23: END
190  REM  INVERSE DEMO
200 VS = 12:VE = 18:HS = 1:HE = 75: CALL INVR
    S,VS,VE,HS,HE,0
210  VTAB 14: POKE 1403,15: INVERSE : PRINT "
    This INVERSE window was invoked from BAS
    IC by:": VTAB 16: POKE 1403,24: PRINT "C
    ALL INVRS,VS,VE,HS,HE,0": NORMAL
220  VTAB 24: POKE 1403,0: GET A$: CALL WINDO
    W,VS,VE,HS,HE,1
230  GOTO 130
240  HOME : PRINT "WINDOWS OBJECT FILE NOT FO
    UND,": END
```

END OF LISTING 2

Ultra Fast Pix

Article on page 52

Listing 1 for Ultra Fast Pix

ULTRA.FAST.DEMO

```
100  REM ***********************
110  REM *  ULTRA.FAST.DEMO     *
120  REM * BY CHARLES PUTNEY   *
130  REM * COPYRIGHT (C) 1987  *
140  REM * BY MICROSPARC, INC  *
150  REM * CONCORD, MA  01742  *
160  REM ***********************
```