

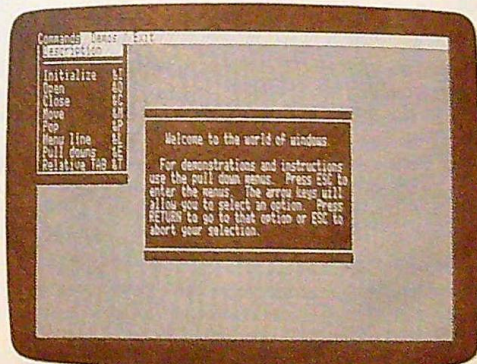
Window Pack

Randy Frank

"Window Pack" adds eight new commands to BASIC, giving Applesoft programmers easy access to windows and pull-down menus. The program runs on an Apple IIc, Apple IIGS, or enhanced Apple IIe with 80-column card. ProDOS required.

By adding windowing commands to Applesoft, "Window Pack" offers desktop operation for all your BASIC programs. With Window Pack's eight new commands, you can open, close, and move text windows, and define and operate pull-down menus, as well.

Window Pack's power is in its simplicity. There are a small number of commands, and each command is straightforward and easy to learn. But before you can use Window Pack, you must type it in.



Window Pack makes your Apple II look like a Macintosh.

Opening the Window

Window Pack consists of three machine language programs and one BASIC program. Programs 1 and 2 contain Window Pack's main routines. Program 3 is required to disable the ProDOS ramdisk so that Window Pack can make use of the 80-column card's extra RAM. The BASIC program, Program 4, helps you design pull-down menus.

To enter Programs 1-3, you'll need to use "Apple MLX," the machine language entry program found elsewhere in this issue. Before loading MLX, you must type **HIMEM:32768**. When you run MLX, answer the first two questions for each program as follows.

For Program 1, answer

STARTING ADDRESS: 3F00
ENDING ADDRESS: 46FF

For Program 2, answer

STARTING ADDRESS: 9900
ENDING ADDRESS: 99EF

For Program 3, answer

STARTING ADDRESS: 0280
ENDING ADDRESS: 02D7

Press E at MLX's Options menu and type the starting address of the program. For example, if this is your first session typing in Program 1, type 3F00. When you've finished, save Program 1 as WPACK, Program 2 as AMPER, and Program 3 as DISCONNECT.RAM.

Finally, type in Program 4. This is a BASIC listing, so you don't need to use Apple MLX. Instead, use "Apple Automatic Proofreader," found elsewhere in this issue, to ensure accurate typing. Use the filename MENU.EDITOR to save Program 4.

Installing the Program

To use Window Pack's commands, you must begin your programs with the following code:

```
10 REM WINDOW PACK STARTUP CODE
20 PRINT CHR$(4)"PR#3"
30 PRINT "### WINDOW PACK ###": PRINT "COPYR
IGHT 1988 COMPUTE! PUBLICATIONS INC.": PR
INT "ALL RIGHTS RESERVED"
40 PRINT : PRINT CHR$(4)"BRUN DISCONNECT.RA
M"
50 PRINT CHR$(4)"BRUN WPACK"
60 PRINT CHR$(4)"BRUN AMPER,A#4000"
70 HIMEM: 30720
80 & I
90 & 0,1,1,78,22,1
100 REM PROGRAM BEGINS HERE
```

This short routine activates 80-column mode, disconnects the ProDOS ramdisk, loads Window Pack's machine language routines, sets HIMEM to make room for pull-down menu data, and initializes the windowing environment with the command &I. An initial output window is set up in line 90. If you like, you may delete this line and set up your own windows later on.

In order for this routine to work, you must have a disk in the drive that contains Window Pack's machine language routines. For this reason, it's best to copy the machine language files onto all of your Window Pack programming disks. You might consider incorporating the installation routine into your STARTUP program. This way, Window Pack will be activated every time you boot your computer.

The Commands

The following describes each of Window Pack's commands. Command names appear in bold, with the parameters (if any) following in italics.

Where coordinates are required, *x* represents the horizontal position while *y* represents the vertical position. The upper left point on the screen is (0,0) and the lower right point is (79,23). *Note: These coordinates reflect character positions, not pixel positions. Window Pack does not support high-resolution graphics.* ▶

Windowing with Machine Language

Window Pack isn't limited to BASIC. Using machine language, you can bypass Applesoft's ampersand commands and access the windowing commands directly.

To pass command parameters, Window Pack uses memory locations 0-4. The table below explains which memory locations you should use for each parameter:

Parameter	Location
address (low-byte)	0
address (high-byte)	1
x	0
y	1
dx	2
dy	3
id	4

To specify the command you want to execute, you must store a number in memory location 5. The following table gives the numbers required for each command:

Command	Value of Location 5
&I	0
&O	1
&C	2
&P	3
&M	4
&L	5
&E	6
&T	7

After you've stored the appropriate parameters and command number into memory locations 0-5, simply JSR to 768 (\$300) to execute the selected command. In the following example, a machine language routine initializes Window Pack and opens a small window.

```
LDA #0
STA 5
JSR 768
LDA #10
STA 1
STA 2
STA 3
LDA #1
STA 4
STA 5
JSR 768
RTS
```

One note of caution: Always be sure that the 80-column card is active before you call any of Window Pack's commands.

&I

Initialize: This command clears the screen, closes all windows, and removes any pull-down menus. You must execute the initialize command at least once before any other Window Pack command can be used.

&O,x,y,dx,dy,id

Open Window: This command opens a blank window whose upper left corner is at coordinates (x,y). The parameters dx and dy specify the width and height of the window, respectively. If the coordinates specified are off the screen, no window is opened.

To give your window a unique identity, you must pro-

vide an id number from the range 0-255. You'll use this number to access the window later on. Only eight windows can be open at once, so it's best to choose a value 1-8. Be careful not to open more than one window using the same id number. If you do, you will not be able to choose the window you want to work with.

&C,id

Close Window: Use this command to close (remove from the screen) previously opened windows. The id number specifies the window you wish to close. If the window to be closed is obscured by another window, the window will be popped to the surface and then closed.

&P,id

Pop: This command brings the window specified by id to the surface (moves it in front of all other windows) and directs all output to this window. As usual, id must be a number from the range 0-255 and must specify a previously opened window.

&M,x,y,id

Move: This command moves the window specified by id so that the window's upper left corner sits at coordinates (x,y). Move also pops the window to the surface and makes it the current output window.

&T,x,y

Tab: You use this command to position the cursor within the active window. Parameters x and y specify the desired horizontal and vertical position of the cursor. Tab's coordinates are relative to the window's current location. So the command &T,0,0 always moves the cursor to the upper left corner of the window, no matter where the window is located on the screen.

&L,address

Load pull-down menus: This command displays the pull-down menus stored in memory at address, making them accessible to the &E command (below). Before you can use this command, you must BLOAD a menu file that was created using Program 4, the menu editor program (see the "Designing Menus" section, below).

By default, pull-down menus load into memory at 32768. So to activate pull-down menus saved to disk as MYMENUS, you could use the following code:

```
1000 PRINT CHR$(4)"BLOAD MYMENUS"
1010 &L,32768
```

When initialized, Window Pack reserves memory locations above 32768. With almost 8K of available memory, it's possible to load three menu files into this area—each menu file takes up 2K. The first menu can be loaded at 32768, the next at 34816, and the last at 36864. Once the files are loaded, you can use &L to switch between pull-down menus. Here's an example:

```
1000 PRINT CHR$(4)"BLOAD MENU1,A32768"
1010 PRINT CHR$(4)"BLOAD MENU2,A34816"
1020 PRINT CHR$(4)"BLOAD MENU3,A36864"
1030 &L,32768 :&E :REM ACTIVATE MENU1
1040 &L,34816 :&E :REM ACTIVATE MENU2
1050 &L,36864 :&E :REM ACTIVATE MENU3
```

If for any reason the top line on the screen is overwritten, you can redraw the current menu bar with the command &L,0.

&E

Execute Pull-Down Menus: When you use the &E command, the first menu on the menu bar is pulled down. (You must load the pull-down menus with the &L command before using &E.) The up- and down-arrow keys allow you to select the various options. To move from one menu to the next, use the left- and right-arrow keys. As you move from menu to menu, the old menu will close as the new one is pulled down. Any screen information covered by a menu is restored when

the menu is closed. Pressing Return makes the menu selection final. If you decide not to choose any of the options, press Escape.

This command returns two parameters: one in memory location 0 and one in memory location 1. The number PEEKed from these locations specifies which menu and option were selected. PEEK(0) returns a number (0-7) corresponding to the position of the menu on the menu bar (counting left to right). PEEK(1) returns the number of the option (1-15) that was selected. This number corresponds to the position of the option within the menu (counting top to bottom). If the user pressed Escape and did not select an option, both PEEKs return the value 255.

The sample code &E:M=PEEK(0):O=PEEK(1) activates the pull-down menus, gets a selection, returns the menu from which the option was chosen in the variable M, and returns the option selected in the variable O.

Note that if you press Control-Reset while Window Pack is activated, the following commands are automatically executed: &I:&L,0:&O,1,1,78,23,1. This clears the screen, resets the pull-down menus, and opens a window for output.

Designing Menus

Program 4 makes designing menus easy. You will use it to create all your pull-down menus. When you tell the program how many menus and options you want, along with the menu and option names, the program creates a pull-down menu file that can be BLOADED into memory and accessed via the &L and &E commands.

The first question asked is how many menus you'd like. Enter a number from the range 1-8 and press Return. Now enter the menu name. Menu names are limited to eight characters in length.

Options come next. Enter the number of options that your menu will offer. You may have as many as 15 options. If you want a thin line to divide any of the option names, count that line as an option. For example, if you want a FILE menu to offer the options LOAD, SAVE, and PRINT, with a thin line separating SAVE and PRINT, enter 4 as your number of menu options.

On the right of the screen, an input window prompts you for the option names. Using our previous example—a FILE menu with four options—you'd enter the following:

- 1) LOAD
- 2) SAVE
- 3) DIVIDER
- 4) PRINT

pressing Return after each word. Option names may be 15 characters in length. If you make a mistake, use the cursor-left key to delete characters.

Wherever the word DIVIDER appears—as in example option number 3—the menu editor replaces it with a thin line in the menu. This line will not be selectable; it simply serves as a separator, to make your menus more readable.

This cycle of entering menu name, number of options, and option names continues until all the menus have been entered.

After you've entered everything, the program activates your recently entered menu list so you can test it. When you've finished looking at your menus, simply press Return and the menu editor will ask if you'd like to save the menu file. If you answer yes, you must enter a filename. Answering no exits the program.

Using the instructions given for the &L command, you can load your file and activate the menus using &E.

Besides allowing you to design pull-down menus, the menu editor provides a good example of how to use Window Pack's windowing commands—it uses these commands ex-

tensively. Because it's a Window Pack application, you should include Window Pack's machine language commands on the same disk. Otherwise, the program will not run.

Hints and Tips

Although text output is normally restricted to the active window, you can move the cursor above or below the window's normal boundaries using Applesoft's VTAB command. This can be useful for printing window titles. For example, say you opened a window with the program lines

```
100 &I
110 &O,10,5,20,20,1
```

To give this window a title, you'd simply add the following code:

```
120 PRINT:VTAB 5:INVERSE:PRINT " MY WINDOW
":NORMAL
```

The PRINT statement moves the cursor to the left-most position of the window, and VTAB moves the cursor into the window's border—something that you normally can't do.

One of the most common things for a program to do is wait for the user to press a key—usually Return—before continuing. Here's a short subroutine that does just that, but with impressive results:

```
500 REM OPEN WAITING WINDOW
510 &O,25,20,30,3,8
520 PRINT:VTAB 21:INVERSE:PRINT "WAITING
WINDOW...":NORMAL
530 SS="PRESS RETURN TO CONTINUE..."
   SS=SS+SS
540 I=1
550 &T,0,0:PRINT MIDS(SS,I,30);
560 I=I+1:IF I=28 THEN I=1
570 FOR D=1 TO 20:NEXT D
580 IF PEEK(49152)<>141 THEN 550
590 POKE 49168,0:&C,8:RETURN
```

As the program waits for the Return key, the message PRESS RETURN TO CONTINUE scrolls patiently through a convenient waiting window.

If you're a machine language programmer, check out the accompanying sidebar, "Windowing with Machine Language."



On Disk Only

If you purchase this issue's COMPUTE!'s Apple Applications Disk, you'll find a "Window Pack" demo program. This program, saved as WPACK.DEMO, demonstrates Window Pack's various capabilities while providing excellent examples of how to use both the menu and windowing commands.

Look for the "On Disk Only" box in all of Apple Applications' articles. If a program or article can be enhanced by additional disk files, we'll explain them here and provide them on disk. For more information on ordering COMPUTE!'s Apple Applications Disk, see page 32.

Program 1: WPACK

For mistake-proof entry, use "Apple MLX," found elsewhere in this issue, to type in this program.

```
3F00: A0 00 B9 00 40 8D 05 C0 09
3F08: 99 00 10 8D 04 C0 C8 D0 B3
3F10: F1 EE 04 3F EE 0A 3F AD B3
3F18: 0A 3F C9 17 D0 E2 A0 B0 E9
3F20: B9 80 3F 99 00 03 8D 05 49
3F28: C0 99 00 03 8D 04 C0 88 24
3F30: 10 EE 60 00 00 00 00 00 7E
3F38: 00 00 00 00 00 00 00 00 B6
3F40: 00 00 00 00 00 00 00 00 BE
3F48: 00 00 00 00 00 00 00 00 C6
```