# The APL*PLUS® System for the Macintosh: An Overview

**Edward R. Myers**
APL Products Marketing Manager
STSC, Inc.
2115 East Jefferson Street
Rockville, MD USA 20852
(301) 984-5110

## Abstract

The APL*PLUS System for the Macintosh™ is a full-featured APL interpreter that integrates the Macintosh user interface style into a standard APL*PLUS System environment. An overview of these Macintosh-specific features along with a comparison of the capacity and performance of the APL*PLUS Mac System to other APL systems suggests that APL applications that are highly interactive, require large workspaces, and need advanced graphics are best implemented on the Macintosh.

## Background

The recently released APL*PLUS System for the Apple Macintosh (December, 1986) expands the availability of APL on desk-top machines. In addition, some of the features of this system represent an evolution in the way APL interpreters interact with the user. In particular, the design of the keyboard layout and use of the mouse, menus and graphics all combine to make APL easier to use and more interactive.

The APL*PLUS Mac System is based on PortaAPL™ for the Macintosh. Richard Smith, author of PortaAPL, and STSC jointly produced the APL*PLUS Mac System using PortaAPL as the technical base. Many new features were added to the PortaAPL system to make it compatible with STSC's other APL*PLUS Systems. In addition, many supplemental workspaces and greatly expanded documentation were prepared for the product.

Two primary audiences were considered when designing and building the APL*PLUS Mac System:

- Application developers who had APL code on other machines and had a need or desire to move those applications in whole or in part to a Macintosh computer.
- Macintosh users who had a need for an analytical or application development environment.



As expected, many design choices were difficult since these two types of users have opposing needs. For example, the application developers with APL code on IBM PCs want a compatible graphics system on the Macintosh. Macintosh users, however, want to use the powerful (yet incompatible with the APL*PLUS PC System) graphics functions that were available through the Macintosh operating environment.

APL*PLUS System features generally fall into the following two categories:

- APL*PLUS System Standard:
  Those features that behave virtually the same on all implementations of the APL*PLUS System.
- System Dependent:
  Those features that provide similar functionality on each APL*PLUS System, but due to environmental considerations are implemented differently in different environments. Occasionally these features are new or unique to an APL*PLUS System and thus considered experimental until proven successful in meeting customer needs.

In the case of the APL*PLUS Mac System, a particular feature was implemented one way (instead of another) because the preferred implementation strongly supported one of the following product objectives:

- To support the traditional standard APL*PLUS System features.
- To maintain a high level of feature compatibility with system dependant features of other APL systems.
- To integrate standard "user-friendly" Macintosh features into the APL environment so Macintosh users would find APL easy to use.

This paper will examine only the system-dependent features of the APL*PLUS Mac System with these product goals in mind. In addition, this paper will compare the performance, characteristics, and system limits of the APL*PLUS Mac System to other APL*PLUS System implementations. This comparison will allow the APL user to choose the implementation best suited for his particular application.

## System Features

Because of hardware differences, features that perform similar tasks are implemented differently in different APL*PLUS System implementations. This section describes the implementation of the following features in the APL*PLUS Mac System:

- Setting User Preferences
- Keyboard Input and Screen Ouput
- Full-Screen Editing
- Terminal Mode and Communications
- Graphics Generation
- Interface to the Operating System and Non-APL Programs

## Setting User Preferences

Each APL system provides a way to set user preferences. User preferences include such environmental settings as font size or style; keyboard layout; and memory allocation for workspace, printing, communications, or editing.

The APL*PLUS Mac System uses resources to store parameters that might change due to user preference. A suite of utility functions are provided that allow the user to modify these parameters to suit his own tastes. Version 1.0 of the APL*PLUS Mac System allows the user to modify:

- Keyboard type (Unified or APL)
- Font size and style (9, 10, 12, 18, 20, or 24 point in either Roman or Italic typeface)
- Communications parameters (Baud rate, protocol, echo)
- Maximum workspace size
- Heap size

User preferences cannot be modified under program control because of the role resources play in the Macintosh environment. The user must exit APL and restart in order for the new choices to be in effect.

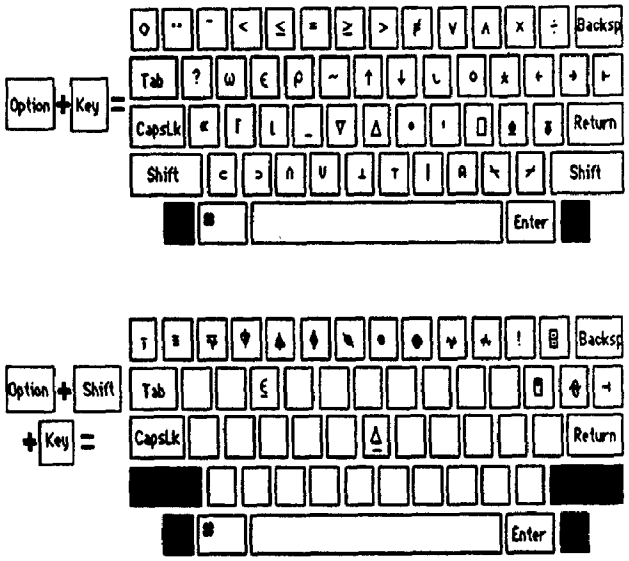
The APL*PLUS Mac System supports two distinct keyboard layouts: APL and unified. The APL keyboard uses the traditional APL terminal keyboard layout with the simple extension that most ASCII characters (e.g., @, #, %, and &) can be entered directly from the keyboard. Users accustomed to using APL on other computers may prefer the APL keyboard.

The default, however, is the unified keyboard. Most users are expected to prefer the unified keyboard layout since it conforms closely to the use of alternate fonts with MacWrite™ or other non-APL software.

The unified keyboard merges the special APL characters into the ASCII keyboard, so keystrokes for characters common to APL and non-APL software will generally match. The keys (pressed by themselves) produce lowercase letters and digits; shifted keys produce uppercase letters and ASCII symbols (such as @, #, and &). The Option key pressed in conjunction with another key produces the APL symbol one would normally expect. For example Option-I produces ⍳, and Option-R ⍴. The Shift-Option combination produces compound characters such as ⍒, ⍋, ⌽, ⍉, ⊖, ⍟, and ⌹. A diagram of the unified keyboard for the Macintosh is provided below.

All APL characters are available as single keystroke combinations, so overstriking to form composite characters is not needed or supported. STSC optionally provides a customization program that allows users to remap the character set to specific keystrokes. This customization option is most useful to international customers who may have unique keyboard arrangements or want to substitute accented vowel characters for unused APL symbols.

**Unified Keyboard Layout**




The APL*PLUS Systems for the PC and UNX use start-up parameters to set user preferences rather than resources (although the parameters that can be changed by the user are different for each system.) In addition, on the PC only, the system function ⎕*POKE* can be used to change many parameters under program control.

## Keyboard Input and Screen Output

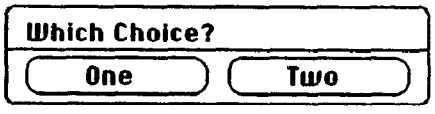
To build highly interactive applications requires that the APL programmer know what events (input) the user or operating system is supplying. For example, on the Macintosh the following events can occur:

- A keyboard key is pressed.
- The mouse button is pressed with mouse at a certain location.
- A menu item is selected from the menu bar.
- The screen is overwritten by a desk accessory.

This list of events is considerably different from the keyboard-only input for mainframe or Unix APL systems. To support these events requires some extensions to existing input functions and the addition of some new input facilities.

In the APL*PLUS Mac System, these events are collected and passed on to the APL application through the system function ⎕*GETKEY*. This event facility combined with the *PTINRECT* function (which determines whether a mouse location falls within a specified rectangular region of the screen) allows one to easily write highly interactive user dialogs. For example, the following function uses ⎕*GETKEY* and *PTINRECT* to determine the user's response to the simple dialog pictured below.

```
10 10 SELECT '/Which Choice?/One/Two'
```



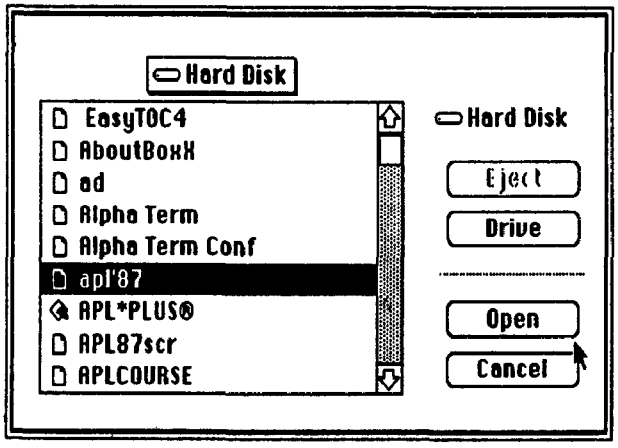
The key part of the *SELECT* program (which is included in unlocked form with the APL*PLUS Mac System) looks like this:

```
[14] Sel_Rect←2 4⍴10 20 30  80 10 90 30 150
...
[16] Until: Input←⎕GETKEY
[17]        →(Until,Char,Event)[⎕io+⍳⍴Input]
...
[19] Char:      ⍝ Handle Keystroke entry
...
[23] Event:
[24]→(Until,Mouse,Menu,Refresh)[⎕IO+1↑Input]
[25]  Mouse: loc←1↓Input ⍝ Which region?
[26]  selected←(loc PTINRECT Sel_Rect)⍳1
[27] ⍝ Perform action based on region
...
[31] Menu:     ⍝ Handle Menu Choice
...
[35] Refresh: ⍝ Refresh screen
...
```

The dialog box itself was generated using the QuickDraw graphics functions.

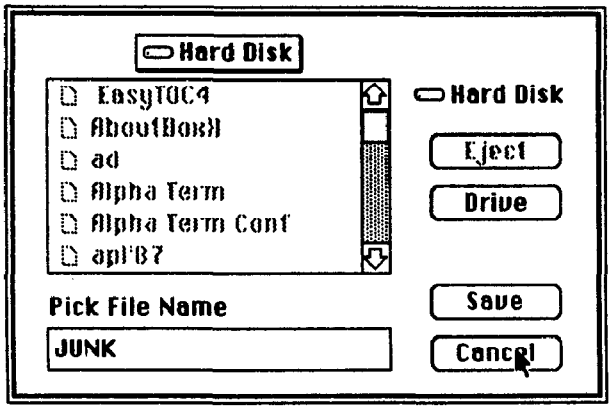
In addition to building custom dialog boxes, two standard dialog boxes are supplied as system functions. *⎕SFOPEN* provides an open dialog

```
⎕SFOPEN ''
```




and *⎕SFSAVE* provides a save dialog

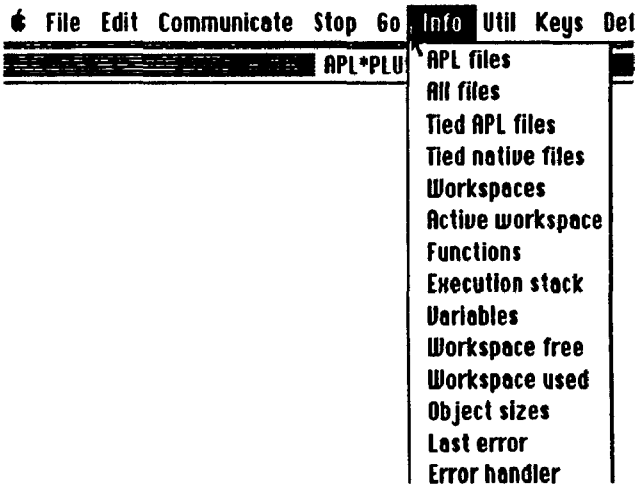
```
'Pick File Name' ⎕SFSAVE ''
```




The APL*PLUS Mac System provides a set of default menu selections for interactively specifying system commands like *)LOAD* and *)SAVE*; or to enter terminal mode and set communications parameters like baud rate and flow control. The menu facility also allows the APL application developer to define his own menus on the system menubar. The menus, like function keys on the PC or PF keys on a mainframe application, allow the user to assign macros or other useful keystrokes to a single key combination. In addition, the menubar, in conjunction with the mouse, can become a help facility.

The menu below shows the default menus (File, Edit, Communicate, Stop) and illustrates one of several optional menus provided with the APL*PLUS Mac System. The menus are created by specifying both the menu items and the keystrokes to be simulated. For example, selecting "Tied APL files" on the info menu simulates entering the following keystrokes:

{clear line} *⎕FNAMES*,'*I*11' *⎕FMT ⎕FNUMS* {cr}



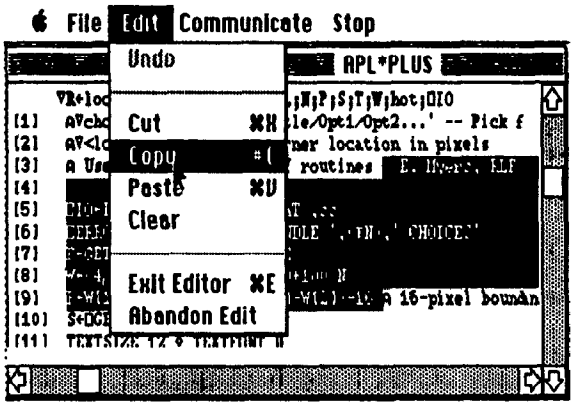

Other screen management functions for the APL*PLUS Mac system are very similar to those in the PC implementation. *⎕WGET*, *⎕WPUT*, and *⎕WINDOW*, have the same syntax and function in both implementations. The only difference is that certain attributes (e.g., color and highlighting) are not supported on the Macintosh as they are on the PC. If future Macintosh machines support a color screen, these system functions can easily be extended to provide color support in a manner similar to the PC.

**Full-Screen Editing**

All APL*PLUS Systems implementations generally include a full-screen editor for use in editing functions and character text.

The APL*PLUS Mac editor is invoked using *⎕ED* for a character vector or matrix, or del (∇) for a function. The full-screen editor adopts many of the cut-and-paste features found in most Macintosh applications. Below is a sample edit session. Note the horizontal and vertical scrolling as well as the edit commands.

Version 1.0 of the APL*PLUS Mac System editor does not include a search and replace facility. However, utility functions (*WSSHOW* and *FNREPL*) are provided to perform syntactic global "search and replace" operations on individual functions or a whole workspace.




### Terminal Mode and Communications

Terminal mode is suprisingly similar in all APL*PLUS implementations (with the exception of the mainframe products, which don't have a terminal emulator.) Each implementation has a single keystroke or menu choice to switch into terminal mode from within an APL session. In addition, each implementation has a facility to read and write character data to/from the communications port. This gives the user the ability to write special programs to upload and download APL programs and data between computers. A standard upload/download program (SERXFER) is available for all APL*PLUS System implementations to facilitate moving data and APL functions between APL systems.

### Graphics Generation

Since the Macintosh has such rich graphics functions built into the machine, the APL*PLUS Mac System was designed to interface with these routines rather than competing with them. The Macintosh graphics routines are accessed through APL cover functions that call a hidden system function (*⎕TOOLBOX*). The system function is not documented since it cannot easily validate the arguments and if the user provides incorrect parameters often the operating system will crash. The cover functions perform a limited amount of error checking and provide more reliable usage.

The QuickDraw cover routines provide information on current global settings or allow the settings to be changed:

- *GETPEN* Returns the position of the pen.
- *GETPENMODE* Returns the pen mode.
- *GETPENPAT* Returns the pen pattern.
- *GETPENSIZE* Returns the size of the pen.
- *GETPENVIS* Tells if the pen is visible.
- *GETTEXTFACE* Returns the text style
- *GETTEXTFONT* Returns the ID of the text font.
- *GETTEXTMODE* Returns the mode for drawing text
- *GETTEXTSIZE* Returns the size of the text in points.
- *TEXTFACE* Specifies the text style.
- *TEXTFONT* Specifies the text font.
- *TEXTSIZE* Specifies the size of the text in points.
- *TEXTWIDTH* Returns width of a string in pixels.
- *TEXTMODE* Sets the text mode.

Line drawing functions allow users to move the pen and draw lines from specific or relative points, or to place text on the screen:

- *LINE* Draws a line relative to the current position.
- *LINETO* Draws a line to a specified point.
- *MOVE* Moves the pen along a relative path.
- *MOVETO* Moves the pen to a specific location.
- *DRAWLINE* Draws a line from one location to another.
- *DRAWTEXT* Draws text using the current attributes.

Several functions allow users to draw and manipulate various shapes, including rectangles, rounded rectangles, ovals, arcs, and polygons:

- Draw the outline of the specifed shape with *FRAMERECT*, *FRAMEROUNDRECT*, *FRAMEOVAL*, *FRAMEARC*, and *FRAMEPOLY*.
- Fill the inside of the specified shape with the current pen pattern with *PAINTRECT*, *PAINTROUNDRECT*, *PAINTOVAL*, *PAINTARC*, and *PAINTPOLY*.
- Invert (interchange black and white pixels) a portion of the specified shape with *INVERTRECT*, *INVERTROUNDRECT*, *INVERTOVAL*, *INVERTARC*, and *INVERTPOLY*.
- Erase (fill with the background pattern) a portion of the specified shape with *ERASERECT*, *ERASEROUNDRECT*, *ERASEOVAL*, *ERASERARC*, and *ERASEPOLY*.
- Fill the inside of the specified shape with the specified pattern with *FILLRECT*, *FILLROUNDRECT*, *FILLOVAL*, *FILLARC*, and *FILLPOLY*.

Several functions allow users to manipulate the graphics cursor and pen:

- *HIDECURSOR* Makes the cursor invisible.
- *SHOWCURSOR* Makes the cursor visible.
- *INITCURSOR* Sets the cursor to the standard arrow.
- *OBSCURECURSOR* Hides the cursor until it is moved.
- *SETCURSOR* Changes mouse cursor appearance.
- *PENMODE* Sets the mode of the pen.
- *PENNORMAL* Resets the pen to its default settings.
- *PENPAT* Sets or changes the pattern of the pen.
- *PENSIZE* Varies the thickness of the pen stroke.

In addition, other specialized functions make dialogs or animation easier and let users manipulate QuickDraw pictures, background, and drawing setup:

- *EMPTYRECT* Determines if a rectangle is empty.
- *EQUALRECT* Determines if two rectangles are the same.
- *PTINRECT* Determines if a point is in a rectangle.
- *BACKPAT* Sets the background pattern.
- *GETBACKPAT* Returns the background pattern.
- *CLIPRECT* Sets a boundary for drawing.
- *CUTPICTURE* Selects a picture from part of the window.
- *PICTFRAME* Returns the coordinates of the picture.
- *SCROLLRECT* Scrolls a rectangular area on the screen.

### Interface to the Operating System and Non-APL Programs

The native file facility does much to allow data to flow easily between APL programs and other programs on a particular machine. On the Macintosh, the clipboard is also available to move text or pictures from one application to another. In addition to copying highlighted portions of text from the editor or APL

session log to the clipboard, the utility function *PUTCLIP* can also be used to put an APL variable on the clipboard. A *GETCLIP* function also exists to pull things off the clipboard, perhaps stored there by another program.

The system documentation also describes the layout of the APL workspace and objects within the workspace so that Assembler functions can interface directly to APL objects. A system function, *⎕MLFX*, makes it easy to create an APL-like function out of an Assembler program. Although implemented differently because of the nature of the machine architecture, this function provides the same flexibility and power that *⎕CALL* and *⎕XP1* do on the APL*PLUS PC and UNX Systems.

### Comparison to Other APL*PLUS System Implementations

Experience with the various APL implementations suggests that if one is given the luxury of choosing a machine as well as the APL language for a particular application, that the choice will depend on the answer to several key questions:

How interactive must the application be?
Both the PC and the Mac Systems have much easier to use and faster facilities for interactive applications than the UNX and Mainframe systems.

What are the capacity limits of the APL system?
The PC system is the only system with a practical workspace limit. Its 500K workspace is small in comparison to the Macintosh and other APL*PLUS System implementations which can all support workspaces up to 4 Megabytes or more. More detail on capacity limits is provided below.

What graphics facilities are available?
Graphics functions are built-in for both the PC and Macintosh implementations. Without a doubt, the APL*PLUS Mac System has better graphics simply because the speed and power of accessing the supplied graphics routines in the Macintosh ROM (as is done in APL*PLUS Mac) is superior to providing special APL graphics routines (as is the case with PC).

What operating performance is expected?
The benchmark data provided in the next section indicates that the actual performance depends on the hardware selected.

Using the answers to these questions and the data below,users can more easily select the right combination of hardware and APL*PLUS System implementation for their application.

### Performance Benchmarks

The table below compares several computers for various operations with different data types. The "Weighted Rating" row was derived by applying the weights in the "Weight" column to the individual machine timings. The "Weighted Rating" column attempts to give an overall measure of the APL performance on that machine.

Although these benchmarks are standard STSC benchmarks that try to yield a statistical model of a typical application, they do not show how well a given application will actually perform in a given environment. A benchmark of the actual application is needed for that information. However, we can safely say that most Macintosh APL applications are expected to out-perform similar applications on the IBM PC XT but not on the IBM PC AT.

Macintosh computers modified to use the 68020 CPU (see the Levco benchmark numbers below) will provide APL performance that exceeds that of an IBM AT. It is also expected that modifying the APL*PLUS Mac System to use the math coprocessor directly (as is done for the PC implementation) rather than through standard software calls would greatly improve the floating point benchmark number and thus the overall performance rating for the Levco computer.

**Performance of Selected Machines with the APL*PLUS System**

| Benchmark | Weight | XT | Mac | AT | Levco | Sun |
|---|---|---|---|---|---|---|
| Small Integer | 10% | 35.0 | 9.2 | 11.3 | 1.4 | 1.2 |
| Large Integer | 10 | 197.6 | 10.7 | 53.8 | 1.8 | 1.2 |
| Floating Point | 15 | 196.3 | 227.8 | 54.0 | 54.9 | 6.2 |
| Mixed Functions | 20 | 143.3 | 52.2 | 51.7 | 9.2 | 2.9 |
| Scaler Operators | 10 | 129.4 | 17.8 | 46.7 | 3.0 | 1.6 |
| Function Execution | 35 | 61.7 | 62.3 | 21.2 | 12.2 | 4.0 |
| Files* | N/A | 67.5 | 59.0 | 32.1 | 3.8 | 12.0 |
| Weighted Rating | | 115.9 | 70.2 | 37.0 | 15.0 | 3.3 |

Notes:

XT: IBM PC XT with 8088 CPU at 4.77 Mhz running APL*PLUS PC System Version 6.0
Mac: Apple Macintosh Plus with 68000 CPU at 8 Mhz and 2 floppy drives running APL*PLUS Mac System Version 1.0
AT: IBM PC AT with 80286 CPU at 6 Mhz and 80287 coprocessor at 5 Mhz running APL*PLUS PC Version 6.0
Levco: Apple Macintosh with Levco modification (68020 CPU at 12 Mhz, 68881 math coprocessor, and hard disk) running APL*PLUS Mac System Version 1.0
Sun: Sun Workstation (Model 260) with 68020 CPU at 24 Mhz and 68881 coprocessor at 20 Mhz running APL*PLUS UNX System Version 3.0
* Not included in the weighted rating because of the variety of environmental factors

### System Characteristics and Limits

The following comparison of system characteristics and limits helps portray the capacity of each APL system.

| | Mac | PC | UNX§ | MF |
|---|---|---|---|---|
| Size of Data Elements (bits) | | | | |
| • Characters | 8 | 8 | 8 | 8 |
| • Booleans | 1 | 16 | 1 | 1 |
| • Integers | 32 | 16 | 32 | 32 |
| • Floating Point | 64 | 64 | 64 | 64 |
| Nested Arrays | No | No | Yes | Yes |
| Maximum rank of an array | 63 | 63 | 127 | 63 |
| Maximum length of a symbol | 76 | 77 | 100 | 77 |
| Max.workspace size (M) | 4 | .5 | # | 15 |
| Execute length limit | 1K | 32K | 32K | 32K |
| Full print precision | 17 | 17 | 17 | 16 |
| Function line length limit | 512 | 32K | 32K | 32K |
| Maximum lines per function | 9999 | 9999 | 32K | 2047 |
| Input line limit | 80* | 511 | 1024 | 255 |
| Symbol limit | † | 2048 | 32K | 6000 |
| Floating point numbers | IEEE | IEEE | IEEE | 370FP |
| Interpreter Size (in K) | 220 | 140 | 340 | 550 |

§ Implemented on a variety of hardware architectures and hence has a few unique hardware limitations.
# Varies from 1M to over 32M depending on operating system's maximum available memory.
* Depends on font size.
† Symbol table is dynamically allocated and thus does not a have a practical limit

## Conclusion

Hardware, unfortunately, is often selected for reasons other than the software that will run on it. If, however, an application needs to be highly interactive, requires a large workspace, and includes graphics as an integral part of the application, the APL*PLUS Mac System is an ideal APL system. The integation of the Macintosh style of editing, graphics, dialog boxes, and menus into the APL*PLUS System provides an environment that encourages building APL applications that are easy and fun to use.

If raw speed (particularly floating-point computation) is desired from inexpensive desktop machines, then an IBM AT-type machine running APL*PLUS PC is a better choice than the Macintosh. However, future Macintosh machines with 68020 processors (now available through machine modifications from companies like Levco and rumored to soon be available directly from Apple) provide impressive performance that rivals even mainframe APL systems. Although machines like the Compaq 386 can provide equivalent or better APL speed performance compared to a 68020 Macintosh, it cannot provide the large workspaces that the Macintosh can--typically needed in bringing mainframe applications to the desktop computer environment.

Over the past several years the Macintosh has grown into a serious desktop computer. With the APL*PLUS System now available for it, it is also a serious APL machine.

## Acknowledgments

The benchmark programs were provided courtesy of James G. Wheeler, Director of Core Technology, STSC, Inc. The Levco benchmarks provided courtesy of William Dumouchel of Belmont, MA.