The Developer Business Report *July 1994*

Apple Directions

Inside This Issue

eWorld Prepares for Launch

eWorld, the new on-line service being prepared for worldwide release by Apple Computer, Inc., will include special posting areas where you can market, support, and distribute your products to the Apple installed base. The service will soon be bundled with Macintosh systems, and its content and unique interface are expected to attract a high percentage of Apple customers. We'll tell you more about eWorld in future issues; for now, more information is available on AppleLink (path—AppleLink Helpdesk:Publishing in eWorld).

Apple News

Apple Ships PowerPC-Upgradable PowerBook Computers

To give its mobile customers access to greater performance, expandibility, and an array of new technologies, Apple Computer, Inc., began shipping four new PowerBook computers and two new PowerBook Duo computers last month.

The PowerBook 520, 520c, 540, and 540c and PowerBook Duo 280 and 280c computers are all driven by the Motorola 68LC040 microprocessor, making them Apple's most powerful mobile computers yet. In addition, they can all be upgraded with a future PowerPC processor.

The PowerBook 500 series opens opportunities for hardware developers to sell PCMCIA and processor-direct slot (PDS) cards to new customers; they're the first PowerBook computers to include an optional PCMCIA Expansion Module and a processor-direct slot. You will, however, have to alter your existing cards or develop brand-new ones if you want them to work with the new computers. Also, you'll have to decide whether to make your

please turn to page 7

Strategy Mosaic

The Best WWDC Ever?

By Gregg Williams, Apple Directions *staff*

According to many developers and Apple employees I talked to, it was the best Apple Worldwide Developers Conference (WWDC) ever. *"Finally,* Apple has a strategy!" I heard a developer exclaim—strong praise from one member of a group (the Apple developer community) whose criticism of Apple has always been driven by the necessity of making good business decisions in a volatile industry.

Certainly Apple's strategies are beginning to converge, to bear fruit. In the rest of this article, I hope to summarize what I see these strategies to be, using information from various WWDC talks to support my points.

You Read It Here First

Some of the most important messages given at this year's WWDC have already appeared in the pages of *Apple Directions,* so I'll restate them briefly and move on.

Time and time again, Apple executives pointed to four key transitions as driving Apple's strategy and computing in the 1990s. These transitions are

- on the hardware level, from CISC computing to RISC computing
- on the application level, from monolithic applications to component software

please turn to page 4

AppleDirections

Volume 2, Number 7

Apple Directions, the monthly developer newsletter of Apple Computer, Inc., communicates Apple's strategic, business, and technical directions to decision makers at development companies to help maximize their development dollar. It is published by the Developer Support Information group within Apple's Developer Press.

Editor
Paul Dreyfus (AppleLink: DREYFUS.P)

Technical Editor
Gregg Williams (GREGGW)

Business & Marketing Editor
Dee Kiamy (KIAMY)

Associate Editor
Anne Szabla

Production Editor
Lisa Ferdinandsen (LISAFERD)

Contributors
Pete Bickford, Alex Dosher, Sally Harris, Ellen Leanse, Kris Newby, Robert Patrick, Jessa Vartanian

Manager, Developer Press
Dennis Matthews

Manager, Developer Support Information
Greg Joswiak

Production Manager
Diane Wilcox

PrePress/Film
Aptos Post

Printer
Wolfer Printing Co., Inc., Los Angeles, CA

Editor's Note

A Place in the Sun

This year's Worldwide Developers Conference (believe it or not) left me thinking about one of my favorite poems, an elegantly simple one in three stanzas called "A Wood," by Richard Wilbur. I can't quote the poem directly without paying Mr. Wilbur's publisher, but I'd briefly like to tell you about it, because the point it makes has a bearing on the software development business.

The first stanza contains lots of impressive words about the trees that tower above the forest floor and how magnificent they are, implying that there's something superior about them. In his next stanza, though, smaller, quieter language praises the little trees—dogwood in particular—that flourish underneath the canopy.

The third and final stanza ties everything together, making a rather obvious point: that it's impossible to say whether the big trees or little shrubs, equally successful in their attempts to thrive, are the better. If that were all the poet were saying, there wouldn't be much to the poem. Turns out, though, that he's really talking about people, not trees. Metaphorically, he's saying that we're all so far away from the perfection of our supreme being that no one's better than anyone else, that we all deserve our place in the sun.

Now back to computing. Lately, everyone is paying lots of attention to the big trees in the personal computer products forest. The dominance of the market by a few companies, and the conglomeration of others, has more than a few smaller concerns thinking they're no good and can't possibly make it. But what I saw at the conference gave me great hope for the little guys and some ideas about how smaller companies can succeed in the shade cast by the big guys.

I saw one company that took on a market leader in a small corner of the computer products market—software piracy protection. Two years after launch, their product is generating $5 million in annual revenues. I saw another company experiencing great success by developing products customized for local markets in Europe.

Most notably, I talked to a developer who, while closing out his tenure at the biggest of the big firms up in Redmond working on OLE 2.0, was starting his own firm to develop and market component software based on OpenDoc.

These and other developers are finding success by specializing—the first two in ways you've probably heard a lot about, the third in a new way that's bound to revolutionize the software industry. The OpenDoc component software model makes the document—not the application—the center of the user's experience; users will be able to choose only the parts they need instead of having to purchase huge applications, parts of which they never use.

This could open unprecedented opportunities for smaller developers. While the big guys are figuring out how to redo their big applications as components, the rest of you will be able to bring to market small software components that accomplish one task, or a very finite group of tasks, extremely well. The key will be to focus on your area of expertise and on just those people who need your component. With the component software model, smaller developers—who can often break the rules and move more quickly and efficiently than more established companies—may even have an advantage.

The lesson here is, sure, there's room for the big guys, but don't be cowed by them. They do a good job at what they do; but even if you're not a Claris or a Novell, you can capitalize on your own talents and succeed on your own terms.

To put different words into the poem's final stanza,

Calls, features, icons, and packaging are to be blended
But no one style, I think, is recommended

Paul Dreyfus
Editor

P.S. Even if you're not a poetry buff, you should check out Richard Wilbur. I think he's one of the best modern American poets.

IndustryWatch: News & Perspective

PowerPC Rolls On

Prepared by the Apple Competitive Analysis staff

[Editor's Note: After writing the IndustryWatch column for its first year, Amanda Hixson has decided to focus her considerable abilities elsewhere. We'd like to thank her for her insight and eloquence these past twelve months, and we wish her luck and success.

No one person can take Amanda's place; instead, we've enlisted an entire department at Apple—Competitive Analysis—to contribute the material that goes into IndustryWatch. The purpose of IndustryWatch is to report key events occurring outside Apple's walls and interpret them for the Apple community at large. Each month, the Apple Competitive Analysis team prepares its Competitive Updates with much the same purpose. Until now, those reports have been confidential, for Apple employees only. Starting this month, IndustryWatch will contain relevant items from the Competitive Updates, with the opinions of the Competitive Analysis staff about their implications for you and the industry.]

Perspective on Lotus 1-2-3

In a move that attracted substantial attention, Lotus announced that it isn't developing a native Power Macintosh version of its 1-2-3 spreadsheet program. The company said it is going ahead with native versions of its Notes groupware and cc:Mail packages.

Implications/Opinions: The 1-2-3 decision is more of a publicity black eye than the loss of a treasured product. Lotus 1-2-3 is gradually losing share in the IBM PC–compatible market, and never took off at all on the Macintosh platform. Although if 1-2-3 had been ported to run in native mode on the Power Macintosh computer, it might have fared better in the marketplace, the fact is that Lotus is betting its future on Notes, which is growing rapidly in popularity. So the news that Lotus will take Notes native is very important.

Also, remember that Apple's Power Macintosh strategy is a multiyear project designed to make Macintosh computers compelling to mainstream PC/DOS/Windows users. As Apple carries out that strategy and succeeds in selling Power Macintosh computers to current PC customers, PC-centric companies like Lotus will see obvious reasons to commit heavily to Apple's platform. But it takes time to change minds. Although the Power Macintosh roll-out was incredibly well received, it may be awhile before core PC users and developers appreciate it. In the meantime, we think current Macintosh customers and "fence sitters"—those trying to decide between Macintosh and PC/DOS/Windows systems—are a large and more easily convinced sales target.

Apple Directions On Line—August

The August issue of *Apple Directions* will be available on AppleLink on July 15. To view the August issue of *Apple Directions* on line, follow the AppleLink path Developer Support:Developer Services: Periodicals:Apple Directions:Apple Directions August 1994.

"I Don't Believe Intel Will Ever Catch Up"

Recently, some of the most credible computer industry leaders have stated that the PowerPC processor will beat Pentium. In its April 1994 issue, *Byte* magazine found that the PowerPC processor delivers a SPECint92 for about $6.45, which is "roughly half the cost of the 66-MHz Pentium." *Byte* continued, "Industry experts and analysts agree: In the microprocessor war, Intel has lost the price/performance battle to RISC."

In addition, perhaps the best-known authority on microprocessors, Michael Slater, told *PC Week* that "The [PowerPC] 604 will put Apple well ahead of anything Intel [Corp.] will have this year, and I don't believe Intel will ever catch up" (April 11, 1994).

Implications/Opinions: These and other quotes about the superiority of the PowerPC chip carry a lot of weight. They support Apple's claim that, over time, PowerPC chips will achieve a large price/performance advantage over Intel's x86 processors.

Faster PowerPC Processors Announced

In early March Intel tried to blunt some of the excitement building up around PowerPC technology by announcing future versions of its Pentium chip that run at 90 MHz and 100 MHz; also, there were hints about future 150 MHz Pentium chips. Now the PowerPC alliance—Motorola, IBM, and Apple—has overshadowed the Intel news by announcing a smaller, faster version of the PowerPC 601 and the high-performance PowerPC 604 chips.

Implications/Opinions: We think the market will interpret these latest PowerPC announcements as further evidence that RISC has set the bar at a level that CISC will not be able to achieve.

A key feature of the new version of the PowerPC 601 chip is its diminutive size. While Intel managed to reduce the size of the original Pentium from 262 square millimeters to about 163 square millimeters, the new PowerPC 601 has shrunk to 74 square millimeters from the original chip's 121 square millimeters. This means that PowerPC processors will maintain a considerable cost advantage over the larger CISC-architecture Pentium.

The PowerPC 604 announcement was important because it shows that the PowerPC alliance is able to deliver additional versions of PowerPC more quickly than Intel can upgrade its x86 architecture. The PowerPC 604 chip is designed for high-end desktop systems, servers, and high-performance graphics workstations. The PowerPC 604 at a clock speed of 100 MHz achieved an estimated SPECint92 rating of 160 and an estimated SPECfp92 rating of 165. The PowerPC 604 is expected to start shipping in volume by the end of 1994, but Intel's next-generation Pentium chip, called the P6, isn't expected to start shipping until the end of 1995.

Not Everything Microsoft Does Makes Headlines

Although Microsoft's public relations momentum continues, sometimes it manages to keep itself out of the news. For example, did you know that the company had to suspend shipments of Windows NT for

at least several weeks? Here's what happened: Microsoft lost on the most important claims in the Stac Electronics suit, forcing Microsoft to remove Doublespace disk compression from MS-DOS 6 and Windows NT. Don't understimate the importance of this: Doublespace was a very popular feature of MS-DOS 6. As a result, shipments of Windows NT were held up so that Doublespace compression could be removed from the product. The holdup in Windows NT shipments went almost completely unnoticed by users or the press.

Here's another story you probably didn't read too much about: For two years Microsoft has offered a scaled-down version of Windows, called Modular Windows, that works on CD-ROM players connected to televisions. Modular Windows was adopted by only one vendor, Tandy, for its Video Information System (VIS), which hasn't attracted a great many customers. Microsoft has now removed Modular Windows from the market, but it is continuing development efforts on other low-end operating-system products. ♣

Strategy Mosaic

The Best WWDC

continued from page 1

- on the user-experience level, from a passive user interface to "active assistance"
- on the networking level, from communication to collaboration

For more details, see "The Macintosh Powers Up," in the April 1994 issue of *Apple Directions.*

In last month's issue, I wrote about "Active Assistance in the Macintosh Interface," which focused on Apple's strategy for implementing the third transition. For you to stay competitive as this transition occurs, you need to implement the following technologies:

- AppleScript (which includes making your application scriptable and recordable)
- Apple Guide
- OpenDoc

Also in last month's issue, Paul Dreyfus's article, "System 7.5: Apple's Unified Operating System for 680x0 and Power Macintosh Computers," talked about this new release of system software, due this fall. WWDC organizers felt so strong about System 7.5 that they devoted an entire session to it and made it the first technical session on the first day of the conference. In it, Apple recommended that every developer should adopt the following System 7.5 technologies:

- Apple Guide
- QuickDraw GX printing model
- Macintosh Drag and Drop
- AppleScript
- PowerTalk mailers (where appropriate)

See page 17 of last month's *Apple Directions* for a list of developer resources related to these and other essential Macintosh technologies.

Transition City

Perhaps the one thing that led to everybody's enthusiasm and sense that things are about to "turn the corner" is the incredible number of transitions that are either "in the works" or "in the cards." Consider the following points:

Between the impressive price cuts on 680x0 Macintosh models and the unequivocal superiority and price/performance of Power Macintosh computers, the feeling is that Apple is making the transition from being "brilliant but not a contender" to "brilliant, with—finally—a shot at the big win." This is, really, a precondition for the success of any of the other transitions.

Apple is also making the transition from being a "lone wolf" to being "leader of the pack." Alliances abound—with IBM, Motorola, WordPerfect, Novell, Borland, and perhaps a dozen more in various pivotal, niche areas. As Michael Mace, manager of the Apple Competitive Analysis group, said at WWDC, the rules are changing: Now, you set standards by *sharing* your technology. "Apple is setting the standards for the next generation of personal computing," he added. "And we're further down this road than most people think."

Certainly, the juggernaut of all transitions is the one from monolithic applications to component software, as defined by the cross-platform, vendor-neutral OpenDoc compound-document architecture. This is easily the greatest change in personal computing since the acceptance of the graphical user interface.

This transition to component software is inevitable for a number of reasons. First, it's got the backing of IBM, Novell, WordPerfect, Apple, and others, so it's *not* going away. And support for OpenDoc isn't just talk; these companies have shown working OpenDoc parts running on the OS/2, Windows, and Macintosh operating systems.

Second, OpenDoc is also the software "fuel" that will (along with the PowerPC processor) energize computing in the next decade. As David Nagel, senior vice president and general manager of the AppleSoft division of Apple, said, "OpenDoc will mean for software what PowerPC is for hardware." Think about it: Both OpenDoc and PowerPC are new, built-from-the-ground-up technologies that have plenty of room to grow; monolithic applications and CISC processors (both the Intel 80x86 and Motorola 680x0 families) are old technologies that are straining to produce incremental improvements. The PowerPC chip is clearly outperforming Intel's Pentium chip, so you could argue that OpenDoc may well succeed over Microsoft's OLE 2.0 (which is built on top of monolithic applications).

A third argument for OpenDoc's success is an important trend in itself: the transition from complexity to simplicity. Several Apple presenters backed up forecasts that OpenDoc will simplify the madness of ever-more-complex monolithic applications, that Dylan will be a better programming language than C++, and that active assistance will reverse the increasing complexity of the human interface.

The point is well taken: Things are getting so complex that users are beginning to balk. If we can't simplify things, not only will we lose some of today's users, but we will have no chance of expanding the personal computer market to the next tier of new customers. Strunk and White said it best in *The Elements of Style:* "Simplify, simplify, simplify." *[Editor's Note: Pete Bickford amplifies this point in his Human Interface column on page 12.]*

How Important Is OpenDoc?

If you need an indication of how important OpenDoc is, you might note that Apple devoted *fourteen* sessions to it. (Compare this to eight sessions on Power Macintosh

and six on imaging.) Here are some other interesting numbers and their significance for OpenDoc:

• The three top software vendors captured 73 percent of software sales in 1993. (Unless you're one of those three vendors, this is *not* good news.) The simplicity and smaller size of OpenDoc parts make it possible for smaller developers to deliver useful products quickly and with just a few programmers.

• Companies spend $5 billion yearly on "horizontal" software applications, but they spend over $50 billion for customized solutions. This latter figure is in the world of minicomputer and mainframe software, but, as personal computers get faster and cheaper, they will capture parts of that market. The modular design of OpenDoc makes it easier for you to customize an application for an industry's (or even an individual client's) needs. That means that if you support OpenDoc, you have a better chance of capturing part of that $50 billion market than developers who don't.

• WordPerfect is porting OpenDoc to Microsoft Windows (more on that later), and OpenDoc parts will work with OLE 1.0 and 2.0 objects. This means that, by supporting one set of application programming interfaces (APIs), you can reach over 55 million customers—40 million Windows customers, 15 million Macintosh customers, and additional customers—as OS/2 and other operating systems implement OpenDoc.

Another important aspect of OpenDoc is the OpenDoc Parts Framework (OPF), a framework of code that will allow you to write one body of code and, with minimal adjustment, create OpenDoc parts for both the Macintosh and Windows platforms. (OPF is the successor of the Apple/Symantec Bedrock framework; as component software became a key part of Apple's strategy, Apple decided that Bedrock had to evolve to include OpenDoc parts.) For starters, Apple has included a development release of OPF on the technology CD given to all the attendees of this year's WWDC.

For more information on OpenDoc and how to start adopting it, see "Planning for OpenDoc—*Now*" on page 14 of this issue.

OpenDoc for Windows

Mark Ericson of WordPerfect Corporation demonstrated OpenDoc parts working with Windows applications and OLE 2.0 objects. According to Mark, the OpenDoc architecture is a superset of OLE 2.0's architecture. Any OpenDoc part can be used in an OLE document, and any OpenDoc part that supports embedding can accept OLE 1.0 or 2.0 objects, as well as OpenDoc parts.

Mark said that WordPerfect will change OpenDoc for Windows to be compatible with future versions of OLE; this makes OpenDoc for Windows a safe choice despite Microsoft's ability to change OLE itself. He added that, since OpenDoc for Windows protects you from "OLE integration nightmares," it allows you to sell to both the OpenDoc and OLE platforms while supporting only the simpler of the two APIs.

To me, WordPerfect's implementation of OpenDoc for Windows is the most impressive—and important—technology I saw at the WWDC. It looks solid in both implementation and theory. By supporting OpenDoc, you reach a bigger audience, create more compelling software, and never have to learn the OLE 2.0 API.

OpenDoc for Windows proves that OpenDoc will be a commercially supported, cross-platform compound-document architecture. Mark said that WordPerfect would be seeding an alpha version of OpenDoc for Windows in May, a beta in "summer," and the final version in "fall 1994." (To get on the seed list, send Internet mail to opendoc@word perfect.com.) He also said that WordPerfect will not charge a licensing fee for developers to use the DLLs (dynamic linked libraries) that will implement OpenDoc for Windows, and that WordPerfect will donate the source code for OpenDoc for Windows to Component Integration Laboratories. (CILabs is the independent, nonprofit agency dedicated to advancing OpenDoc as a vendor-neutral standard for software integration and component software.)

Copland: The Beginning of the Future

The next 2 1/2 years will bring significant enhancements to the Macintosh OS, but it will be a multipart evolution that will maximize backward compatibility and give you time to change the things that need changing. The next release after this fall's System 7.5, code-named *Copland,* is due in 1995. Copland will improve the performance and human interface of the Macintosh OS, and it will lay the groundwork for later advances in Gershwin (due in 1996) and later system software releases. To prepare for these changes, you can read the "Compatibility Issues for the Future Macintosh OS" paper, located in the Future Mac OS folder of the WWDC Technologies CD.

Copland will deliver, first and foremost, a higher-performance Macintosh OS. An expanded Toolbox will have over 95 percent of its code written in PowerPC code. It will include system-level support that will make development easier—including things like standardized implementations of sliders and floating windows. Copland will also include a major redesign of the human interface that will make the Macintosh easier to use, and it will use Apple Guide to implement even more active-assistance features.

Copland will also deliver a microkernal designed specifically for the needs of a personal computer. This microkernel will include

• both preemptive and cooperative multitasking, which will allow the Macintosh to execute multiple tasks simultaneously

• threading, which will increase the throughput of the processor

• address space management, which will reduce the damage a program can do if it crashes

• new interrupt services, which will speed up Power Macintosh computers and make them more responsive in real-time situations (in video playback, for example)

• a file-based implementation of virtual memory that will allow paging of the system heap

I admit that Copland will not give you everything you want. (That comes later, in Gershwin.) In particular, all Macintosh applications and the Toolbox will cooperatively multitask in the same address space. (Full, preemptive multitasking puts each application in its own address space; if a program crashes, it can't cause any others to crash.) However, in Copland, programs can create protected processes that do run in their own address spaces, so you'll be able to write programs that offer protection against many crashes.

Copland will also include a completely redesigned File Manager that will use concurrent I/O and improved algorithms to run faster and handle larger files and volumes. Programs using the API of today's File Manager will continue to run, but you'll want to convert to the new File Manager,

which is easier to use, has a simpler API, and supports more volume types. The new File Manager will support Unicode filenames.

Finally, Copland will support the Open Transport standard, which will open the Macintosh up to protocols other than AppleTalk. By providing a single API for all transport protocols, it will enable a single application to run without change under different transport protocols. A "multi-homing" feature will allow a Macintosh computer to connect to multiple protocols simultaneously.

Copland will support almost all of today's APIs. However, some software, including external file systems, some hardware drivers, and code that patches low-level calls, will have to change. Copland will support today's Thread Manager, so continue using it; making your programs threaded helps prepare them for Copland and future operating systems.

Gershwin and Beyond

The intent of Gershwin is to extend the preemption and memory-protection features of the Copland operating system. It's hard to say what features the Macintosh OS will have 2 1/2 years from now, but the goal is to have preemptively scheduled Macintosh applications that run in separate, protected memory space. Another goal is to implement symmetric multitasking, which will allow a program to run on multiple processors. This requires threaded software, so be advised that threading is a Good Thing.

Dylan—Thomas or Bob?

Actually, it's neither. Dylan stands for *DYnamic LANguage,* and the name is Apple—Apple Dylan.

What, you may be asking yourself, *is he talking about?* Dylan is an object-oriented dynamic programming language proposed by Apple several years ago, and Apple Dylan is Apple's implementation of the Dylan language (and a development environment) for the Macintosh.

The Dylan presentation at the WWDC produced two big surprises: the language/environment itself (which was very impressive), and developers' enthusiasm for this totally new technology. Both the overview and technical sessions on Dylan—held in large halls—were packed; since then, in less than a week, we've gotten over 80 requests for seeding with Dylan. Hmmm, maybe it had something to do with the fact that Apple engineers who used it reported a two-to-three-times improvement in time-to-completion, reliability, and maintainability.

It's a bit early in the game—Dylan doesn't go into alpha seeding until the third calendar quarter of 1994—but I'd be remiss if I didn't give you some advance notice of it.

Dylan is part of Apple's strategy of making sure you have a choice of development tools. The Dylan group's charter was to start from scratch, create a development environment that is significantly better than existing ones, and—most important—make sure that the result is something that makes sense, given the realities of commercial software development today.

Here are some of the most compelling points about Dylan:

- Dylan is a pure object-oriented language. Apple believes that object-oriented programming helps make a programmer more productive.
- Dylan will also have its own application framework. Experience in object-oriented programming shows that, especially at first, a framework is even more important than the object-oriented language in making a programmer productive.
- Dylan is a dynamic language. This means that the programming experience is more interactive and less broken up into lengthy coding, compiling, and debugging phases. Dylan's dynamic nature makes using it a looser, more fluid process—for example, you can change code while running a program and see the effects of the change immediately. Also, Dylan's incremental compiler lets you see the results of your change without a long wait that interrupts your train of thought.

Because Dylan modules link at run time, you can optimize a module (not changing the argument list, of course) and replace the previous version with it; a program that uses it will continue to work without being recompiled. This solves what is called the *fragile base-class problem,* which is a substantial shortcoming of C++.

- You can use Dylan to create stand-alone, double-clickable commercial applications. Today, in pre-alpha form, Dylan itself contributes an overhead of about 400K, and the Dylan team expects to reduce this significantly in the future.
- The Dylan language is simple but powerful. Not only is Dylan code easier to write than C or C++ code, it's shorter. Early users report that Dylan source code is about one-half to two-thirds the size of that of equivalent C++ programs. Shorter code means shorter development times.
- The Apple Dylan Binder (a combination of a code browser and the Finder) is a dream to use, at least two generations ahead of the browsers associated with object-oriented languages today. It's impossible to do it justice without seeing it in action, but let me summarize by saying that you can manipulate the contents and behavior of browser subpanes directly with the mouse. Further, you can save these customizations to be used, later, in different situations; in other words, the Binder gets better the more you use it. It can also cross-reference your source code in a zillion different ways, and it lets you see different units of code without having a different browser open for each unit.
- Dylan will be able to use existing C and C++ headers and C++ classes. Also, you will be able to use code compiled by Dylan in C and C++ projects. By making C++ code usable with Dylan code and vice versa, Dylan "fits in" with the existing realities of software development, and it's more likely that companies will be willing to use it.
- Dylan will have a user-interface builder, code-named *Woodstock,* that will separate user-interface code from the rest of your code. Because of this, you will be able to use Woodstock to modify the user interface without losing the changes you've made to the non–user-interface body of your program.
- Apple pledges to develop a commercial-quality Windows implementation, but no details are available yet. A Windows version of Dylan is necessary to make Dylan a viable alternative for cross-platform developers.

The Dylan alpha seed will occur during the third calendar quarter of 1994; if you're interested in being a seed site, send an AppleLink message to DYLAN (dylan@applelink.apple.com on the Internet). Please give your name, address, phone number, and an indication of what you want to use Dylan for.

The first seed version of Dylan, called *Early Dylan,* will omit some of the features just mentioned; however, it will be entirely usable. Apple should release Early Dylan 1.0 during the first calendar quarter of 1995. The Dylan team plans on changing Early Dylan based on user feedback and releasing the full Apple Dylan sometime in the second half of 1995. Apple Dylan is already an

exciting environment, but with a difference—this one just may succeed.

1994: A New Hope

People may look back and say, "1994 was the year that things turned around for Apple." Not that Apple did the "turning around" *in* 1994. For the beginning of the turn-around, you must go back to the alliance forged between Apple/IBM/Motorola in 1991, the lowered product prices and difficult financial decisions that Apple has implemented since then, and the technologies (such as OpenDoc) that have been evolving for the past several years.

WWDC '94 was different because it came at the tail end of all this change. In the marketplace, developers have already seen results from some of these changes, and the technologies shown at WWDC reinforced the sense not just of change, but of change with impending results. We have a lot of work ahead of us, but in doing it, we're going to change the very nature of personal computing. ♣

Apple News

PowerBook Computers

continued from page 1

expansion solution available on either PCMCIA or PDS cards, because the new computers will not be able to accomodate both types of cards at the same time. For more information on developing cards for the PowerBook 500 series of computers, see the Technology article "Card Developers Only: What You Need to Know About PowerBook 500 Expansion Cards" on page 17.

The new PowerBook and PowerBook Duo systems offer customers a variety of advanced features. Each of the new computers share a signficant enhancement: They are based on the 68LC040 chip, running at either 66/33 MHz or 50/25 MHz, and, to reiterate an important point, they can all be upgraded to PowerPC technology.

"Why two numbers for clock speed instead of the usual one?" you're probably asking. Earlier this year, Motorola decided to use a new scheme to describe 68LC040 performance. Previously, only the system bus speed (the second number) was used to refer to the chip's performance. Now, performance will be described by both the internal processor bus speed (the first number) and the system bus speed. For example, a 66/33 MHz 68LC040 processor runs at 66 MHz internally and 33 MHz externally on the system bus.

Other enhancements are specific to each product line (PowerBook or PowerBook Duo), so we'll describe them in separate sections on the two types of systems.

PowerBook 500 Series Features

A sign that the PowerBook 500 computers will be winners in the marketplace is that they just took top honors at the Spring COMDEX, held at the end of May in Atlanta. The new systems received *Byte* magazine's "Best of Show" award in the Portable Computer category, coming in ahead IBM's ThinkPad 755 family and the Z-NOTEFLEX from Zenith Data Systems, the other finalists in the category. *Byte* cited the PowerBook 500 computers' upgradability to PowerPC, high performance, and full feature set as its reasons for making the award.

The PowerBook 520 and 540 computers are loaded with new features; perhaps the most noticeable of them is the trackpad they employ. Replacing the trackball, which is used in the new PowerBook Duo computers, the trackpad enables users to control cursor movement by running a finger across the pad's surface. It employs a technology called *coupling capacitance,* the same kind of technology used in elevator buttons that require only that you go near them, not touch them, to select your floor. (For more details, see "How the Apple Trackpad Works" on page 8.)

Additionally, each PowerBook 520 or 540 computer comes with a 9.5-inch diagonal, 640-by-480-pixel display that's somewhat taller and narrower than previous PowerBook displays. That's the difference on the outside; on the inside, the systems employ the following display technologies to produce crisp, clear images:

- PowerBook 520 computer: FSTN passive-matrix LCD that provides 16 grays
- PowerBook 520c computer: dual-scan passive-matrix color LCD that provides 256 colors
- PowerBook 540 computer: active-matrix LCD that displays 64 shades of gray
- PowerBook 540c computer: active-matrix LCD that produces 256 colors, or thousands of colors when it's switched to 640-by-400 resolution, something users can do without having to restart

The PowerBook 520, 520c, and 540 computers can all be upgraded to the color active-matrix LCD that ships standard with the PowerBook 540c system.

Another enhancement that should be very attractive to users is that the new PowerBook models can use not just one, but two batteries; the PowerBook 540 and 540c computers ship with two batteries, while the second battery is available separately for PowerBook 520 and 520c computers. Previous models can be operated only for about two to three hours before requiring recharging; PowerBook 520 models can have up to seven hours of battery life; PowerBook 540 models can have up to six hours.

The computers ship with new PowerBook Intelligent batteries based on nickel-metal-hydride (NiMH) technology, and they include a built-in processor card that communicates directly with the power management system to provide the longest battery life possible. The new PowerBook systems also employ battery management software that lets users monitor and conserve battery power.

Customers will also notice, and appreciate, the complete overhaul Apple industrial designers have given the new systems. Compared to earlier models, the PowerBook 520 and 540 computers employ more curves than edges, which gives them a distinctive appearance and makes them more comfortable to use and easy to carry. They're manufactured from a new thin-wall polycarbonite plastic four times stronger than previous PowerBook plastics and a magnesium subframe, making them more durable than previous models.

The new PowerBook systems measure in at approximately the same size as previous models; because of their new design, even with two batteries they have virtually the same weight as the older systems, which are designed to use only one battery at a time. The heaviest model, the PowerBook 540c computer, weighs 7.3 pounds, just 0.2 pounds heavier than the PowerBook 180c

computer. With one battery, PowerBook 520 and 520c systems weigh only 6.3 and 6.4 pounds, respectively, making them the lightest PowerBook computers yet.

Here are other key features of the PowerBook 520 and 540 computers:

- built-in Ethernet
- expansion slots for optional modem and additional RAM (up to 36 MB)
- 16-bit stereo audio with a built-in microphone and speakers
- support for a wide variety of external monitors
- full-size 19 millimeter-pitch keyboard with function keys
- hard disk drive storage up to 320 MB

In addition, each of the computers ships with the PowerBook Mobility Bundle, software that provides communication capabilities, compatibility, power management, and information management. Included in the bundle is Apple Remote Access Client 2.0, which allows PowerBook users to communicate with a remote Macintosh desktop system; PC Exchange, which reads from and writes to DOS-formatted disks, Dataviz Easy Open translators to automatically open DOS and Windows files; PowerBook File Assistant, which synchronizes PowerBook files with those on any other Macintosh-formatted volume; and the PowerBook Control Strip, which provides power-management services.

Configuration and pricing of the new systems are as follows (all Apple prices are U.S. only):

- PowerBook 520 computer with 4 MB of memory, 160 MB hard disk, and 50/25 MHz (50 MHz is the processor's internal clock speed, 25 MHz the external bus speed) 68LC040 processor—$2,269
- PowerBook 520c computer with same configuration as PowerBook 520 computer—$2,899
- PowerBook 540 computer with 4 MB memory, 240 MB hard disk, and 66/33 MHz 68LC040 processor—$3,159
- PowerBook 540c computer with 4 MB memory, 320 MB hard disk, and 66/33 MHz 68LC040 processor—$4,839

The PowerBook 520 and 540 computers replace all but the PowerBook 145B and 165 computers in the U.S. product line; if you're outside the United States, you can check with your local Apple office to see how the product line has changed in your area.

PowerBook Duo 280 and 280c Features

Dataquest recently determined that the PowerBook Duo computer was the best-selling subnotebook computer in both the United States and Europe in 1993, capturing a 38 percent share of the U.S. and 29 percent share of the European subnotebook markets. To borrow a cliché, the best just got better.

Both the PowerBook Duo 280 and 280c computers employ the 66/33 MHz version of the 68LC040 processor. They can also be expanded from their standard memory configuration of 4 MB to 40 MB.

Both also include active-matrix displays. The PowerBook Duo 280 computer displays 16 levels of gray on a 9-inch (diagonal) 640-by-400-pixel screen. The PowerBook Duo 280c computer is the only subnotebook that supports 16-bit color; it displays 256 colors on an 8.4-inch screen when it's set at 640 by 480 pixels (or thousands of colors at 640 by 400 pixels).

Battery life has been extended two to four hours on the computers through the use of NiMH batteries; the PowerBook Duo 280c computer uses Type III NiMH batteries (which enable PowerBook Duo 210, 230, and 250 systems to operate for up to seven hours), while the PowerBook Duo 280 computer uses Type II NiMH batteries.

Each of these new systems weighs just over four pounds, the same as older PowerBook Duo systems. The PowerBook Duo 280 and 280c computers replace all but the 68030-driven PowerBook Duo 230 computer in Apple's U.S.

How the Apple Trackpad Works

Apple Computer, Inc., has always worked to stay ahead of the curve when it comes to introducing new technology to personal computers, a habit they intend to continue with forthcoming hardware and software releases.

The new PowerBook 520 and 540 computers are the latest examples of Apple's commitment to including new technology at affordable prices. Apple determined that advanced features were important to today's mobile personal computer users, so it built a great many of them into the new PowerBook models. Perhaps the most impressive of these new features is the Apple trackpad.

In designing and building the trackpad, Apple engineers used a technology called *coupling capacitance.* Here's how it works: Built into the trackpad surface are two layers of measurement electrodes, arranged in a checkerboard configuration, with horizontally directed electrodes crossing vertically directed electrodes. As your finger moves over the trackpad's surface, the intersections, or couplings, between the horizontal and vertical electrodes evaluate the change in capacitance.

Horizontal electrodes controlled by special software send out a test-pattern signal that's sensed by the vertical electrodes. As your finger approaches the trackpad surface, its presence lowers the capacitance at the nearest intersection of electrodes and modifies the signal received by the sensing (vertical) electrodes. By monitoring this change, the trackpad first locates and then compares the current location of low capacitance (that is, the spot your finger is touching) with the previous location. Trackpad software then moves the cursor to the new spot.

Special software makes the trackpad's response velocity-sensitive. This allows you to nudge the cursor by stroking the trackpad slowly, or move it across the entire width of the screen with one quick stroke. The cycle of sending out a test pattern, sensing the change in capacitance, and then moving the cursor is carried out repeatedly at very high speed.

The trackpad permits greater control than the trackball; tests with users indicated that the trackpad increased accuracy and efficiency by 20 percent. Also, since it employs no moving parts, it's more reliable than the trackball.

The trackpad is covered by a special polymer that protects it from abrasion and spills. Also, its thin footprint makes room for additional circuitry underneath the trackpad; PowerBook engineers decided to use the extra space for the second battery. It also weighs less and draws less power than the trackball.

product mix. Existing PowerBook Duo 210, 230, 250, and 270c computers can be upgraded to the new 66/33 MHz 68LC040 processor board.

Apple has also released a new version of the Duo Dock, the PowerBook Duo Dock II. When plugged into a Duo Dock, PowerBook Duo systems have all the capabilities of desktop Macintosh computers. The PowerBook Duo Dock II adds to the capabilities of the existing PowerBook Duo Dock with a 68882 math coprocessor, a 32K RAM cache, 1 MB of video RAM for displaying 32,000 colors, and built-in Ethernet.

Configurations and pricing of the new PowerBook Duo systems (all of which ship with the PowerBook Mobility Bundle) and the PowerBook Duo Dock II are as follows (all prices are U.S. only):

- PowerBook Duo 280 computer with 4 MB memory, 240 MB hard disk, and 66/33 MHz 68LC040 processor—$2,639
- PowerBook Duo 280c computer with 4 MB memory, 320 MB hard disk, and 66/33 MHz 68LC040 processor—$3,759
- PowerBook Duo Dock II—$969

All systems are available immediately throughout the world, except the PowerBook PCMCIA Expansion Module, scheduled to start shipping in July. ♣

News From Apple Business Systems

New Version of AppleSearch to Provide Internet Search Capabilities

[Editor's Note: News From Apple Business Systems is a new monthly feature. It's assembled by Robert Patrick (AppleLink: PATRICK.R), an evangelist in Apple Business Systems' Servers and Services group.]

Earlier this year Apple Computer, Inc., announced that it will expand its AppleSearch text search and retrieval software beyond local-area networks (LANs) to help users find the information they need on the Internet. AppleSearch is client-server software designed to give Macintosh computer users personalized, easy access to large text and document databases.

Combining Apple's traditional ease-of-use with sophisticated search-agent technology, the enhanced version of AppleSearch will help users find and prioritize relevant information residing on the Internet by providing seamless access to wide area information servers (WAIS). The WAIS gateway will be available in a new release of AppleSearch this fall.

With the addition of the WAIS gateway feature, AppleSearch allows users to

- provide user-defined "agent-based" updates of developments on the Internet
- allow server administrators to select the specific WAIS servers available to LAN users
- enable unattended, non–prime-time searching to heavily used Internet information sources
- easily integrate information sources based on LANs and wide-area networks (WANs)

In addition to expanding AppleSearch capabilities, Apple intends to continue working closely with commercial and university developers to ensure that a variety of Internet access solutions are available. Currently a beta version of GopherSurfer search engine is available from University of Minnesota that allows Gopher clients to search AppleSearch indexed content.

The AppleSearch Client Developer's Kit is available through APDA. This package contains the AppleSearch Client API software kit, the *AppleSearch Client Developer's Guide,* and the AppleSearch Update File Format Developer's Kit. The kit will be upgraded at the time of the new AppleSearch release to allow developers to capitalize on the new feature.

AppleSearch Background

More and more computer users are looking to the Internet to connect to business and university databases as well as various small networks that are often warehouses of useful information. As this continues, Apple will provide tools to help developers and content providers facilitate users' navigation through the stockpiles of information available on the Internet. It's Apple's intention to create solutions that will enhance the user's experience and increase productivity while providing seamless access to the Internet from a local area network.

AppleSearch features Reporters, which are sophisticated search agents that quickly search through documents on a file server, regardless of their format, to deliver information users need. Reporters can be assigned to keep users updated as new and relevant information appears on the server. Combining features found on high-end bibliographic full-text retrieval software with Apple's traditional ease-of-use and administration, AppleSearch can also tap into information sources such as server-mounted CD-ROM discs, news wire services, or other on-line services that feed into the AppleSearch server. The AppleSearch server software incorporates the Callable Personal Librarian (CPL), developed by Personal Library Software, Inc., as its search engine.

Availability and Pricing

AppleSearch (without the WAIS gateway) is currently available from authorized resellers and dealers. In the United States, a Server/Client 5-Pack is available for $1,799 and a Client 10-Pack is available for $499. An AppleSearch Client Developer's Kit is priced at $299 and is available through APDA, Apple's source for developer tools. Pricing outside the United States may vary. For more information on AppleSearch, contact the Apple Network Information Line at (408) 862-3385. ♣

Seventy-Five Native Power Macintosh Applications Currently Shipping

Seventy-five native Power Macintosh applications were shipping throughout the world by the end of May 1994. Native applications have been recompiled to take advantage of the Power Macintosh computers' two- to six-times performance boost over previous 68040-based Macintosh models. If you haven't begun porting your applications to native PowerPC mode, you'll want to consider doing so, soon. The following is a list of currently shipping native applications.

Note: This list is provided for informational purposes only; it does not imply an endorsement of any of the products by Apple Computer, Inc.

Company	Product
About Software	5PM Term for Macintosh 2.2
	5PM Pro for Macintosh 2.2
ACI	Object Master Universal 2.5
Aetis - Protections Logicielles	Copy Protection
Aldus	Freehand
	PageMaker
ALSOFT	Atlas 1.0.4
	GeoConcept
Artifice, Inc.	DesignWorkshop
Artwork Systems N.V.	ArtPro 1.2
Software B.V.	Drill 1.2
Atlas Software B.V.	Key Software 1.1
	Optipoint 1.1
Autodessys, Inc.	3D Form Synthesizer
Baltic Business Systems	MacHansa Accounting II 2.0
B.E.M.E. R&D	ALIX (colors for printers)
Bungie Software	Pathways Into Darkness
Canto Software GmbH	Cumulus
Casady & Green	Conflict Catcher
	Spaceway 2000
Central Point Software, Inc.	MacTools 3.0
Charles River Analytics, Inc.	Open Sesame!
Claris Corporation	ClarisWorks
DELTAPOINT	DeltaGraph Pro 3
Diehl Graphsoft	MiniCAD+
Domark	Flying Nightmares
Dunaway Systems B.V.	Signalize 2.6
	Spooler 1.2
	PostScript™ Interpreter
	Scanning & Vektorizing 2.3
	Remote Font & Clip Art
FIT Software	Full Contact
Fractal Design	Dabbler 1.0
	Painter 2.0
Frame	FrameMaker
FWB	Hard Disk Toolkit
Gibbs Systems	Virtual Gibbs
GRAFTEK	Ultimage/Pro(Optilab/Pro)
Gryphon Software	Morph!

Company	Product
Hi Resolution Limited	Mac=Bac 1.1
	MacPrefect Remote 1.0.1
	MacVisa 1.1
Insignia	SoftWindows™ 286
Interstudio	flex•plan 1.0
	Nonio C 5.0
ITEDO Software GmbH	IsoDraw 2.6
Jasik Designs	MacNosy
Language Engineering Corp.	LogoVista E to J
MedImage	MedView
Metrowerks	CodeWarrior
MicroMacro, Ltd.	MicroGuard ADB Copy Protection
Orange Micro, Inc.	OrangePC
ORKIS	ImageBasePro 2.5
Pole Position Software GmbH	Mac DCF77
Route 66 Geo Info Systems	AtomicTime
	ROUTE 66 1.2.0
SCITEX America	Full Auto Frame
Segue Software	QA Partner
SOFT Technologies	Simulateur de conduite 1.2
SofTeam Hardware & Software Dist.	MacSign 4.0
	Punto 1.6
Specular	Infini-D
Trillium Research	Remus (ltd version)
Trio Systems Europe	C-Index Pro 1.0
TrueD Software	Live on RISC
UserLand Software	Frontier 3.0.2
usrEZ Software	ultraSecure 3.0
VAMP	MacCAD Trailblazer
Vicom Technology, Ltd.	VICOM MultiTerm
	VICOM Pro SDK 5.0
	VICOM RunTime
VideoFusion	Recorder
	VideoFusion
Wilkensen SCOOP	SCOOP Archive 1.1
WordPerfect	WordPerfect

Technology

Inside This Section

Developer Note Details New PowerBook Systems

New this month from Apple, *Macintosh Developer Note Number 9* (APDA #R0567LL/A) provides relevant technical information on the PowerBook 500 and PowerBook Duo 200 series of computers—the newest, high-performance members of the Macintosh PowerBook family—and the PowerBook Duo Dock II, an enhanced version of the original PowerBook Duo Dock.

The PowerBook 520, 520c, 540, and 540c computers are the first members of a new generation of all-in-one notebook computers featuring the powerful Motorola 68LC040 microprocessor, extended expansion capabilities, "intelligent" long-life batteries, and a unique input device called the *trackpad.* The PowerBook Duo 280 and 280c computers are improved versions of the earlier PowerBook Duo 200 that also include the Motorola 68LC040 microprocessor.

The new developer note focuses on the built-in communication features and extended expansion capabilities of the new PowerBook computers, and provides an in-depth description of the new Power Manager application programming interface (API). This information is invaluable if you are designing new communication solutions, hardware expansion products, or applications that take advantage of the power management software.

This developer note is available in DocViewer format on the June Developer CD—as well as on paper through APDA. (See page 24 for APDA ordering information.) ♣

CD Highlights

System Software Edition

This month's System Software Edition of the Developer CD Series features—along with the cover art of Mark Jasin—localized versions of system software versions 7.1.1 (System 7 Pro) and 7.1.2 (for Power Macintosh computers). We have dedicated almost the entire disc to bringing you all of the localized versions of Apple's latest system software; however, these files take up over 600 MB of space on the disc, leaving little room for anything else.

As we did last quarter, we'll put any additional new system software, and as much of the Printer Drivers and Other Apple Software folders as will fit, on the next disc in the series (the August Tool Chest Edition). System software version 7.1 and the collection of system enablers can be found on the April 1994 disc and will return on the October 1994 System Software Edition.

This state of affairs will be temporary, and we are continuing to examine solutions for delivering new worldwide system software in a timely and easily usable way. Your comments and suggestions are welcome; please send them in an AppleLink message to DEV.CD or in an e-mail message to dev.cd@applelink.apple.com.

Here is a description of some of the other new and revised material included on this CD.

Chinese Language Kit Version 1.1

The Chinese Language Kit contains all the software you need for working in Chinese on your computer. With it, you can use a Chinese application program, or a program that takes advantage of WorldScript, to enter Chinese and other languages in a single document.

The Chinese Language Kit supports the Traditional Chinese and Simplified Chinese scripts. You can install and use one or both of them. For more information, see the documentation accompanying the Japanese Language Kit on this CD.

Japanese Language Kit Version 1.1

The Japanese Language Kit is a System 7.1 extension that adds support for Japanese to any version of the Macintosh operating system, not just KanjiTalk. It is compatible with KanjiTalk 7.1 and Japanese applications, and includes two Kanji TrueType fonts and the Kotoeri input method.

For more information about both the Japanese and Chinese language kits, see "Japanese, Chinese, Korean, Arabic Language Kits" in the August 1993 *Apple Directions.*

European Distributors & Localizers

This folder contains a list of European distributors and localization companies.

Mac Tech Notes Update

Technical notes are a collection of short (and not-so-short) articles dealing with specific development topics. New and updated technical notes for July 1994 include

- *DV 22 - CD-ROM Driver Calls*
- *OS 05 - System Update 3.0*
- *TB 40 - Partial Resources*

Network Software Installer Version 1.4.4

This folder contains the Network Software Installer for AppleTalk version 58.1.3. You can use this Network Software Installer to install the following networking products:

- Update to AppleTalk version 58.1.3
- Network Control Panel version 3.0.2
- AppleTalk Internet Router 2.0 (requires router installation disk)
- EtherTalk version 2.5.6
- Ethernet driver version 1.1.1 for the Apple Ethernet NB Card

please turn to page 18

Human Interface

Defending the Revolution

By Pete Bickford

Once upon a time, computers were pretty hard to use. A mere 15 years ago, people were signing up for night school classes by the droves in order to learn magical incantations like "INIT HELLO" and "CD C:\MISC\PROGSTUF". These words held great power for controlling the silicon beasts and, occasionally, even bending them to a user's will.

Then a machine called Macintosh came along and changed the world. Macintosh was founded on a dream—of bringing the incredible power of computers to "the rest of us." It was a machine that let normal folks do incredible things, a brilliant, democratic revolution of people and technology.

My greatest fear, though, is that we're beginning to lose sight of that dream, letting creeping complexity grow to the point where only the experienced few truly have "the power to be their best."

That Bloated Feeling

We Macintosh folks have long felt pretty smug about our edge in ease of use. One of my favorite old Apple ads made great sport of this complexity gap, comparing the thunderous sound of a stack of DOS tomes hitting a table ("This is what you need to learn in order to use the average personal computer") with the lightweight "flop" that the hundred or so pages of Macintosh manuals made ("And this is what you need to learn to use Macintosh").

If anyone's thinking of doing the same sort of commercial today, I'd suggest they use some really big tables. That way, they'd have a place to drop the manuals—and a place under which "the rest of us" could hide.

It used to be that Macintosh applications shipped in impressive-looking boxes containing little manuals and lots of air. Ten years of upgrades to these programs have yielded some Really Powerful Features™, but at the same time, the little manuals in the big boxes have been replaced by big manuals in even bigger boxes. And the bloat doesn't stop with the manuals. Many columns ago I wrote about how Macintosh word processors once fit (with a system and Finder) on a 400K floppy disk. At the time I bemoaned the fact that a common Macintosh word-processing program took up some 15 MB of disk space when fully installed. That was last year. I've since learned that the new version is slated to occupy some 19.4 MB of disk space, with the core word-processing portion requiring 8 MB of RAM to run comfortably on a Power Macintosh computer.

Along with these ballooning resource requirements, we've seen a huge cluttering of the program's interface, which now features a sea of menus, tool bars, and obscure controls. One wag on the Internet was distributing a picture of what a future version of the application might look like several versions hence: a monstrosity with so many controls that there was only a tiny bit of room left in which to type text. I fear this person wasn't far from the mark. In several word-processing programs on the market today, you can already get this effect when you open their seven or eight icon "ribbons."

In fairness, the designers of the word-processing package are hardly the only ones suffering from Geometric Complexity Bloat (GCB). The last few years have seen Apple release a parade of decimal-numbered system software releases, add-on extensions, and "system enablers." All of these seem quite rational—even necessary—considered individually, but there's little doubt that the net effect has been to add to the growing burden that all of our users must cope with.

We're letting complexity run out of bounds, and it's costing us dearly. According to the Harper's Index, nearly 4 in 10 adults think that if they try to use a computer, they'll actually damage it. Some of the brave ones who take on our current programs are being cheated out of the productivity gains that should be theirs. Many are forced to take special classes or buy third-party books in order to master the arcana that our mainstream "productivity" applications have become. And, for us developers, complexity means lengthened development cycles, higher documentation and support costs, and an ever-increasing burden of maintaining our monolithic applications.

It has to stop, now.

Upgrading Without Overwhelming

The sad thing is that growing complexity is almost a natural phenomenon in this business. Simply by going from one version to the next and tacking on a few more features here and there, we eventually wind up with a program that only an expert can use.

While we want to improve our products, we need to recognize that although hard-disk and RAM sizes may have gone up an order of magnitude in the last ten years, the human ability to understand complexity has not. If we want to add new features, we need to do it without complicating the program. Often, if we work hard enough and use innovation and imagination, we can find ways to make our programs more powerful while actually making them easier to use. This, incidentally, is the same phenomenon that turned the IRS's one-page 1040 form from fifty years ago into the behemoth it is today. Fifty years of "little improvements," and now we hire experts to do our taxes for us.

A case in point is the way control panels, extensions, Apple menu items, and fonts are organized in System 7. Previously, users had to recognize the difference between the various types of items, and carefully put each of them in its proper place. For some of these items—for example, desk accessories and bitmap (but not PostScript) fonts—you had to use a special installer program to embed them into the System file itself. System 7 could have turned the problem into a disaster, introducing as it did a number of new types, the ability to customize your Apple menu, and so on. Instead, the designers stepped back to see the overall configuration problem. They tore down the wall between applications and desk

accessories, put all types of fonts in the same place, and most important, gave the Finder intelligence that let it automatically route the assorted items to their appropriate places whenever the user dropped them into the System Folder.

Tactics for Overcoming Complexity

The battle for ease of use can be won; what we need is a strategy. About two years ago I detailed five "new principles" of interface design for overcoming the challenges of developing modern software. *[Editor's Note: You can find these articles in the June, August, September, October, and November/December 1992 issues of* Apple Direct.] These principles need to be part of each developer's battle plan, and I'll recap them briefly.

First off, our applications need to start providing *active assistance* to users. (Two years ago, I called it *intelligence*.) The idea is to program some rudimentary "smarts" into your application so that the user doesn't need to do all the work to carry out a task. A charting program, for example, should be able to automatically figure out reasonable grid points for a certain set of data. And (my favorite example) a personal organizer should know how to format phone numbers.

In a similar vein, our interfaces need to be transparent. That is, users should be able to concentrate on their work without having their train of thought (and peace of mind) disturbed by the workings of the program itself. A big part of this consists of limiting the messages the program spits out to those that are actually relevant to users. For instance, connecting to an on-line service doesn't mean that I should have to deal with messages like "Handshake complete." Programmers may need to live in that world, but not users. If you need to speak to users, do it in their language, not yours.

Two other principles, attention to detail and constraints, work to reduce complexity by removing distracting or inappropriate parts of the interface. Attention to detail does this by keeping users from being plagued by small annoyances. The use of constraints is more direct, meaning that you remove or disable controls that don't apply at a given time. For instance, a network set-up dialog box wouldn't ask about your modem's baud rate if you'd previously said that you'd be connecting using AppleTalk ADSP through your printer port.

Finally, there's elegance, which you achieve by seeing the big problems and finding graceful, simple solutions. Elegance is perhaps the hardest principle to master, primarily because it goes directly against the tendency to simply tack on new features. Elegant interfaces integrate features into a coherent whole. Elegance also involves the vigilant use of the 80/20 rule: 80 percent of your program's users employ only 20 percent of its features. The goal is to smooth out the 20 percent of your program that benefits virtually all your users. At the same time, you have to be clever in keeping the complex features demanded by the 20 percent (the "power users") from harming the usability of the core 80 percent.

And Now . . . the Revolution

On top of those other principles, the biggest weapon we have for fighting complexity is the shift from an application-centered view of the world to a document-centered view. In the past, developers had to cram any functionality the user might ever consider into their applications, fueling the growth of today's monolithic applications. Now, with breakthrough technologies such as OpenDoc, we're getting a chance to reinvent the entire model. Instead of cramming the user's work (the document) inside a specific tool (an application), we shift the entire focus to the document, bringing in whatever tools we need to get our job done.

The current way we build documents is mad—we're forced to buy MegaWrite Pro with its 19.4 MB of add-ins because that's the only way we can put the kind of chart we want into our document. It's like trying to build a bench but having to use the same company's screws, wood, and paint because the wood can't be fastened together with another company's screws or painted with a paint that they didn't provide with their kit. If the real world worked like this, you'd see exactly what we have in the computer world: bloated collections of tools that don't fit a given user's needs at all, but that everyone buys anyway because they're scared that nothing else will work together if they buy something else.

By adopting a document-centered model of computing, we have the chance to break this all apart. People could learn a finite set of simple tools, then apply them to any document they wanted to create. And the tools themselves, freed from having to be all things to all people, would have the chance to do their own job simply and effectively. After all, nobody needs a book about how to use a paintbrush in the real world—why should it be so different in the computer world?

OpenDoc is the first step toward realizing this dream. It breaks applications into parts and lets users choose the tools (graphing, text, and so on) of their choice. We could have a revolution in ease of use.

But at the risk of playing Cassandra (which, as Neil Gaiman might have said, I don't have the legs for anyway), we need to recognize that no revolution is a replacement for the hard day-to-day work of forging a better world. Worse yet, the revolution has the potential for being co-opted by those who are too comfortable with business as usual. Dark thought: What if the folks at MegaSoft decided to implement their OpenDoc "parts" by simply relabeling their current modules? You'd then have a "text tool" that takes multiple megabytes on disk, as part of a collection of, oh, say, about 19.4 MB of associated "parts" that slavishly carried the old complexity into the new world. Worse yet, what if other developers did the same?

Reviving the Dream

The power to be your best. It's a slogan that speaks of normal people being able to unleash incredible power through easy-to-use technology. It's a dream that we've let grow clouded over time, so that many people can't tell the difference between ease of use and pretty icons. It's time to get fired up again, revolutionize the revolution. And this time, let's keep it going.

Viva la revolucion!
—Doc

Planning for OpenDoc—*Now*

By Gregg Williams,
Apple Directions *staff*

OpenDoc is *the* major upcoming technology for the Macintosh, and if you attended last month's Apple Worldwide Developers Conference, you should have no doubt about it: Fourteen sessions were devoted to OpenDoc, and numerous others related their strategies to it.

OpenDoc will reinvigorate personal computer software, give you a larger customer base, and offer new opportunities for software development. (For a more detailed description of OpenDoc and its place in the market, see the article "Why 1994 *Will* Be Like 1984: OpenDoc Will Change the Macintosh . . . and More," in the August 1993 issue of *Apple Directions*.)

But what's *inside* OpenDoc? How much will you have to change your current applications to make them OpenDoc-savvy? Though I can't answer this question for you as completely as I'd like, I want to offer you some initial details on the structure of OpenDoc software, so that you can start thinking about how you will approach your conversion to OpenDoc. So, instead of the usual view from 10,000 or even 30,000 feet, think of this article as a view from low satellite orbit (which will necessarily omit a number of details).

Your next step will be to study the OpenDoc documentation on this year's WWDC Technologies CD-ROM, which you should have in your hands by now; the first document to read is the *OpenDoc Technical Summary.* Although you won't receive a beta version of the Macintosh implementation of OpenDoc until later this summer, believe me when I say that deciding what you want to do with OpenDoc will keep you busy until then.

Overview

As you probably know, OpenDoc is a cross-platform compound-document architecture designed to be an open standard. The independent, nonprofit Component Integration Laboratories (CILabs) owns the OpenDoc source code and will make it available to anyone who wants to join their organization. Apple's implementation for the Macintosh platform has already been seeded to WWDC attendees and is scheduled for commercial release this fall. WordPerfect has announced that it will ship its OpenDoc for Windows this fall, and IBM has shown an implementation for OS/2.

OpenDoc has been designed with "Seamless OLE" integration—that is, OLE objects will work in OpenDoc documents and vice versa. (OLE, Object Linking and Embedding, is Microsoft's application-based technology for creating compound documents.) This makes OpenDoc a more attractive development proposition, because with one (OpenDoc) development effort, you can reach both existing Windows applications and OpenDoc-compatible Macintosh products.

OpenDoc makes the user's computing experience document-centered instead of application-centered. A document doesn't "belong" to any one application. Instead, it contains multiple *parts,* each of which is governed by a specific *part handler,* code that handles the user's interaction with the part's content. One part, called the *base part,* serves as a container for all the other parts in the document.

An OpenDoc document might contain text, a spreadsheet, and a graph based on the spreadsheet's data. To the user, each homogeneous area of data is a part, and the boundary around the part (which can be any shape and can overlap other parts) is called the part's *frame.* Clicking in the part allows the user to interact with its content.

The programmer sees the OpenDoc environment differently. To the programmer, a part is the visible portion (or portions) of the document's content within the part's frame (or frames), plus the part handler that manipulates it. The frame is an interpretation of (or view into) the part's data; multiple frames may show different interpretations of one part's data, and they can be physically discontiguous within the document.

In the example just given, the spreadsheet and graph may actually be two frames governed by the same part handler. Drawing, however, is always done into a *facet,* which is owned by a frame. Facets are used to show discontiguous areas of a frame's data. For example, a window with a splitter bar could be implemented as two facets of the same frame.

Sometimes the beauty of OpenDoc as a cross-platform architecture is in what it *doesn't* try to do. For example, it doesn't try to define a set of drawing primitives that every platform, however awkwardly, must implement. Instead, OpenDoc acts as a "traffic cop" between the user, the computer, and the part handlers. It only goes so far as to keep track of the geometry of parts within a document. It knows nothing of drawing, per se, but depends on the part handler to draw what's inside the part. Not only does that keep the OpenDoc architecture platform-independent, but it also increases the likelihood that the programmer converting an application to OpenDoc parts can modify existing drawing code instead of having to start from scratch.

When the user opens an OpenDoc document, system software instantiates (makes an instance of) a subsystem called the *OpenDoc shell,* which is responsible for doing three things:

- creating and initializing the Session object, which serves as the interface between the document and OpenDoc
- opening the document (however that is defined for a given platform)
- accepting human-interface events and passing them to the OpenDoc Dispatcher, which then passes each event to the appropriate part handler

Part Handlers

Obviously, part handlers are what you will spend most of your time thinking about. Part handlers are responsible for handling events, drawing and printing, disk I/O, undo/redo operations, and other functions. The OpenDoc application programming interface (API) includes about 50 methods (routines) that a programmer must write to implement OpenDoc fully. Depending on the complexity of your part handler, though, you may need to implement considerably fewer methods.

OpenDoc is implemented as shared libraries—ASLM (Apple Shared Library Manager) libraries in the alpha version and SOM (System Object Model) libraries in the beta and shipping versions. Currently, you must program in C++, but one of the reasons for moving to SOM is its language neutrality; eventually, you'll be able to do OpenDoc

programming in other languages, including C.

As mentioned earlier, when an OpenDoc document opens, it acquires a Session object, which handles the interaction between OpenDoc and the document. Among the objects made available by the Session object are

- the Arbitrator, which makes access to resources (such as the keystroke stream and the menu bar) available to only one part handler at a time
- the Dispatcher, which accepts every user event and gives it to the right part handler
- the Undo stack, which supports multiple levels of undo operations across different part handlers

Events

For OpenDoc, event-handling code has not changed much; most of it is in a switch statement that handles the different kinds of events. However, unlike a traditional Macintosh application, a part handler receives (through the Dispatcher) only the events that belong to it. A table called the *part registry* associates each part in a document with the right part handler.

Also, since a part handler doesn't "own" the document or the human interface, it has to get permission from the Arbitrator before it can change the menu bar, accept keyboard input, and so on. The process of getting permission to use a resource is called *getting the focus.*

In this environment of necessary cooperation, your part handler has a different relationship to events than a traditional Macintosh application does. Unlike a Macintosh application, which calls WaitNextEvent to "donate" control to other applications while it is waiting, an OpenDoc part handler doesn't retain control of the system. Instead, the Dispatcher gives a part handler control to handle an event that belongs to that part. In this way, OpenDoc is much like MacApp, which retains control but calls your code to perform application-specific functions.

OpenDoc distinguishes two kinds of events—user-interface events and semantic events. User-interface events deal with the graphical display of the document and with actions such as mouse clicks, keystrokes, and menu selections. Semantic events deal with the content model of the document—for example, Apple events from an event suite are semantic events.

OpenDoc separates semantic events from user-interface events for a reason. Doing so aids the process of making a part handler scriptable and recordable; it also isolates the platform-dependent code (the user-interface code) from the code that is more likely to stay the same from platform to platform.

Also, the semantic-event subsystem is actually an extension to OpenDoc. You can be a first-class OpenDoc citizen without it, but scriptable/recordable part handlers will be more useful in the OpenDoc world, and Apple strongly recommends you build scripting support into your OpenDoc parts. If your application is already scriptable, then you've already done most of the work needed for supporting semantic events in your application's OpenDoc equivalent.

Drawing and Layout

As mentioned earlier, OpenDoc does not concern itself directly with drawing; rather, it keeps track of the geometry of all the parts in a document and relies on the part handlers to draw their contents correctly. OpenDoc bases all drawing on three primitive constructs (the implementations of which differ from platform to platform):

- the *canvas,* which provides a drawing context that includes things like a color table and a coordinate space
- the *shape,* which describes an arbitrary geometric area of a canvas
- the *transform,* which describes a geometric transformation available to the drawing system used

OpenDoc depends on part handlers to draw the contents of the frames they handle. However, OpenDoc works with container parts and part handlers to negotiate the layout of the various frames within a container part. For example, a piece of graphics, when dropped into an OpenDoc document, might (if it is designed to do so) resize itself to take up a certain fraction of the document's window size.

Several shape objects assist OpenDoc in the layout process. When the user drags a part into a container part, the container part tells the part (called the *embedded part*) its frame shape—that is, the space the container part has available into which the embedded part can draw itself. The part returns its *active shape,* which is the space it takes up. (This lets the container part know what shape the embedded part is responsible for drawing.) Finally, the *used shape* specifies to the container part the area in which the embedded part will take responsibility for mouse events; often, the active shape and the used shape are the same.

Storage

The *storage system* is the platform-specific mechanism that provides persistent storage for OpenDoc documents. To keep OpenDoc a cross-platform architecture, Apple engineers gave the storage system an abstract design. There are advantages and disadvantages to this. One advantage is that the OpenDoc code that handles document storage and retrieval should run, without modification, on every OpenDoc platform. Another is that the OpenDoc storage code you write will be simpler than what you'd write for non-OpenDoc applications, because the OpenDoc storage model isolates you from numerous implementation details.

The downside of this design occurs when you convert existing applications to part handlers: You must adapt whatever storage model you're using for your documents to the OpenDoc storage model. Still, the news is not all bad. The OpenDoc designers chose a model that defines a document as a system of structured files, each of which contains a stream of data. Since many programs store their documents as multiple streams of data, you can use much existing code when you convert to the OpenDoc storage model.

An OpenDoc document (which is based on Bento, a platform-independent file format) contains one or more *drafts,* each of which represents a snapshot of the document at a point that the user specifies. Each draft contains one or more *storage units.* A storage unit contains a list of uniquely named *properties,* each of which consists of an ordered list of values (see the figure "OpenDoc storage model" on page 16).

Why are there multiple values in a property? This design allows an OpenDoc document, if designed properly, to be more useful to more people. For example, you might decide to store a chunk of text as plain text, Macintosh styled text, and RTF (rich text format) text. That way, people using other OpenDoc-compatible platforms are more likely to be able to open and read the text in your file.

To reduce the number of methods you need to know, individual storage-system methods (such as Remove) perform differently depending on the context in

which they are called. A method called Focus limits the context of future operations to a specific property or value. Then the Remove method, for example, can be used to remove a value, a property (and all its values), or even an entire storage unit, depending on the focus that was previously set up. In addition, methods such as InsertValue, DeleteValue, GetValue, and SetValue allow you to manipulate data streams within the value that is currently the object of the focus.

Undo and Redo

The OpenDoc session object creates a session-wide *undo object* that contains both an Undo and a Redo stack. When your code has made a change that you want to be "undoable," you create a record containing all the information needed for restoring your part to its former state. You then pass this record to the undo object using a method called AddActionToHistory. You must also implement Undo and Redo methods. When an action needs to be undone or redone, OpenDoc system software sees that these methods get the data they need (stored as a record on the Undo or Redo Stack).

Cooperation

The success of the Macintosh platform has hinged on every piece of Macintosh software "being a good citizen." The same is true in OpenDoc. To work efficiently, OpenDoc requires a certain amount of cooperation from part handlers—for example, in the areas of relinquishing focus and freeing memory.

When a part acquires a focus from the Arbitrator, OpenDoc system software may send it a request to relinquish the focus. In some cases, your part handler will be in the middle of an operation that shouldn't be interrupted (for example, your part handler shouldn't give up ownership of the serial port when it's in the middle of a file transfer) and should ignore the request. However, if at all possible, it should give up a focus when requested. The methods BeginRelinquishFocus, CommitRelinquishFocus, and AbortRelinquishFocus govern this process. In difficult situations, OpenDoc may unilaterally give or take away the focus; you must implement the FocusAcquired and FocusLost methods to handle these situations.

Similarly, your part handler should, if at all possible, release whatever memory it can when it receives the Purge method. You might free cached data or resources, for example. However, this is a voluntary action.

The OpenDoc Parts Framework

Earlier this year, Apple announced it was modifying the Bedrock application framework (until recently, a joint project with Symantec) to support OpenDoc. The result is the OpenDoc Parts Framework (OPF), which will allow you to create OpenDoc parts for both the Macintosh and Windows platforms with one development effort. A development version of the OpenDoc Parts Framework is on the WWDC Technology CD that attendees of this year's Apple Worldwide Developers Conference received; beta and final versions will be released after the commercial release of OpenDoc.

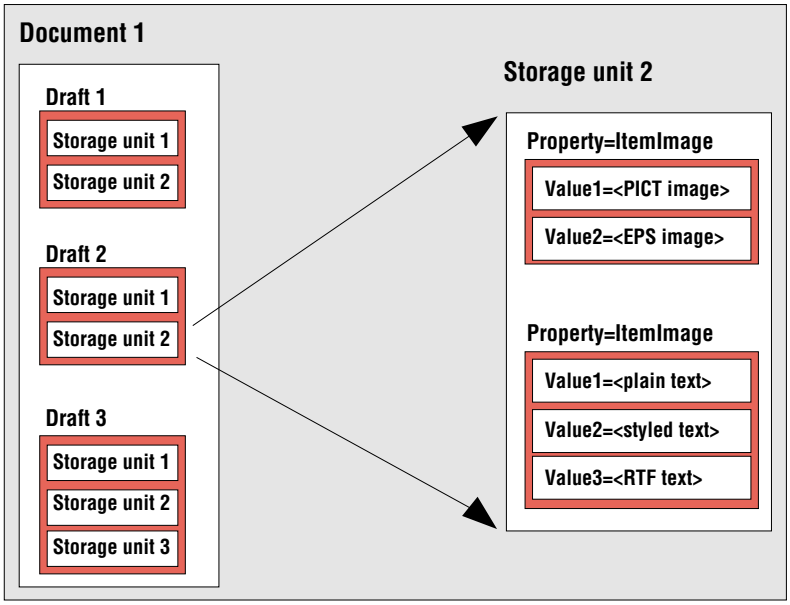
Your Next Steps

You'll need to begin by doing some research: in particular, reading the *OpenDoc Technical Summary* mentioned at the beginning of this article. Chapter 3, "Programmer Model," gives an overview of what you need to do to create a part handler from scratch, adapt an existing application, make your application's documents into OpenDoc container parts, or implement OpenDoc itself on a new platform. Also, issue 19 of *develop* (September 1994—not yet published) will contain an article on building OpenDoc part handlers, written by Kurt Piersol, the chief architect of OpenDoc.

Here are some thoughts on further steps you might take:

- The simplest thing you can do is to convert your application documents to be an OpenDoc *container application;* that way, your application will work with the first OpenDoc parts that become available. (A container application is still an application, but users can drag other parts into documents created by the container application.) According to the *OpenDoc Technical Summary,* "The task is not trivial but is in fact relatively simple. It is more difficult than the QuickTime revision but can probably be accomplished with a week or two of programmer time." Once you convert your application to be a container application, you should consider implementing new features or major upgrades of existing features as OpenDoc parts, all the while working on converting the rest of your application to OpenDoc.

(An interesting aside: By converting your application in this way, it can automatically partici-



OpenDoc storage model. An OpenDoc document can have multiple drafts. A draft contains a variable number of storage units, each of which contains one or more properties. Values are arbitrary streams of data, and a property can have more than one value.

directing applications that are scriptable—in scripting. Someone will inevitably make an OpenDoc button part that, when pressed, executes an AppleScript script. People can then drag such buttons into your documents, instantly making them into "smart documents" that users can interact with.)

• You may want to create your own OpenDoc parts. Since a part needs to do only one thing well, you can create one much more quickly than you can an application—and with fewer programmers, which is attractive if you're in a small company or want to start one. Since OpenDoc is a completely new technology, it gives you a chance to become an innovator—and you don't have to have venture capital to do it! If your OpenDoc part is the "better mousetrap" for some task, you may not only sell it to customers directly, but you may also find it profitable to license it to others for inclusion in their products.

• A more ambitious undertaking is to convert your entire application to a set of OpenDoc parts. Granted, doing this is a large undertaking, but it would put you at the hub of the OpenDoc universe—and possibly get you a helping hand from Apple. People will buy your product (gulp—we can't call them *applications* anymore!) as the foundation for their own OpenDoc documents. You can then start selling add-on parts to your customers.

Other advantages: By the time you get your Macintosh product ready to go, you'll be able to start converting it to make a Windows product. Also, by mixing and matching parts, you'll be able to create different versions of your product—with less than a proportional amount of extra work.

• If your application is scriptable and, optionally, recordable, you'll have an easier time converting it to OpenDoc parts than developers whose applications are not. Such products will be the most desirable ones in the OpenDoc universe, ready to reap all the benefits of OpenDoc. And since important future technologies depend upon your product being scriptable and recordable, scriptable/recordable OpenDoc parts will be the most prepared for the future.

Getting In on the Ground Floor

It's not every day that you get a chance to participate in an entirely new technology that has the potential to change everyday computing as much as the graphical user interface of the Macintosh did a decade ago. Even better, this new technology increases the chance that a small company with good ideas can take the market by storm.

Of course, the biggest rewards will go to the earliest innovators, and some developers are already on their way. But it's a whole new frontier, and there's plenty of room for you to stake your claim. It won't happen automatically, so now is the time for you to educate yourself and decide what OpenDoc means for your company's future. ♣

Card Developers Only: What You Need to Know About PowerBook 500 Expansion Cards

For the first time, you'll be able to sell PCMCIA cards and processor-direct slot (PDS) cards to Apple's PowerBook customers, because the new PowerBook 500 series now includes an optional PCMCIA Expansion Module and a processor-direct slot. This article briefly tells you what you need to know to begin taking advantage of this opportunity. For more product details about the new PowerBook computers, see "Apple Ships PowerPC-Upgradable PowerBook Computers" on page 1.

The Processor-Direct Slot

To make the new computers more expandable than earlier PowerBook models, each PowerBook 500 computer ships standard with a 68030 processor-direct slot—the same kind of slot used by Macintosh LC models—in the left expansion bay (inside the left battery compartment). Electrically, it's not a "true" PDS, because it connects to the input/output sytem's 68030 bus and not directly to the 68040 processor bus. In other words, the PDS is independent of the processor bus, which means that customers can upgrade computers in the PowerBook 500 series to a new microprocessor (such as the PowerPC 603) without having to buy new PDS cards.

The PCMCIA Expansion Module, available for purchase separately, plugs into the PDS connector. This means that customers who opt for PCMCIA expansion will not be able to use PDS cards at the same time, and vice versa.

PCMCIA card development tends to be less expensive and comparatively easier than PDS card development, and the market appears to be moving in the direction of PCMCIA expansion. As a result, many of you will probably opt, in the long run, to develop PCMCIA cards for the PowerBook market. That said, the rest of this article will give you some specifics about each type of card.

PCMCIA Cards

PCMCIA cards derive their name from the industry committee that determines the standards for these credit-card-sized cards, the Personal Computer Memory Card International Association. The cards—which are increasingly being called "PC cards"—can contain memory, input/output hardware, or both. The types of memory that can be put on the cards are static RAM (SRAM), ROM, and flash memory; input/output possibilities are virtually unlimited, but currently the cards are used to provide a variety of services, including modem, network connection, hard disk, cellular phone, and paging.

PCMCIA cards are currently a delivery medium for Newton applications. Also, many Intel-based notebook computers currently include slots for PCMCIA cards. With the PowerBook 500 series, PowerBook customers will also be able to expand their systems with PCMCIA cards. The PCMCIA Expansion Module accepts two Type II PCMCIA cards or one Type III card.

Special software is required for PCMCIA cards to work with the PowerBook 500 computers. As an interim solution, Apple will first ship basic driver software that enables PowerBook 500

computers to accommodate cards that provide modem, hard drive and flash memory, and paging services.

Later this year, Apple is expected to release the PCMCIA Manager, which is a complete implementation of Card Services and Socket Services, industry-standard software that registers PCMCIA cards with their corresponding drivers and mediates between multiple cards sending data to the processor. The PCMCIA Manager will enable you to adapt your PCMCIA drivers to work in a PowerBook system. Once it's available, Apple will be encouraging you to help provide a wide range of PCMCIA-based solutions—which some of you might do by modifying existing cards—for PowerBook 500–series customers.

We'll provide specifics about the PCMCIA Manager when they're available. Until then, see *Macintosh Developer Note Number 9,* which is available from APDA. (You can read more about this new developer note on page 11 of this issue.)

PDS Cards

The PowerBook 500 computer PDS is designed to enable users to install or remove PDS cards on their own. Although it's similar to the slot found in Macintosh LC computers, the physical interface is different, requiring 90 pins in two segments. As a result, if you want to sell your PDS cards to PowerBook customers, you'll have to adapt your cards' designs to make them fit the expansion bay.

Additionally, you'll have to change the power requirements of your PDS cards to work with the battery power available to PowerBook systems. PowerBook computers with PDS cards installed give up one battery and the associated battery life, which substantially reduces the length of time a user can work without replacing the batteries. To help this situation, your card needs to include circuitry that puts it into a low-power mode when the computer is in Sleep mode and turns the card off when the computer shuts down. Furthermore, the card isn't used all the time even when the computer is running, so you'll want to consider including a power control circuit that reduces your card's power consumption when it's not being used.

If you're interested in developing PDS cards, some services you might want to build into the cards include network connections, wireless networking (both for connections to local area networks and to modems), video capture, and NTSC, PAL, or SECAM video output.

For technical details on developing PDS cards for the PowerBook 500 series of computers, see the developer note we just referred to and *Designing Cards and Drivers for the Macintosh Family,* third edition (#M7075/C), available from APDA or your local bookseller. (See page 24 for APDA ordering information.) ♣

CD Highlights

continued from page 11

- Ethernet driver version 6.0.5 for the Apple EtherTalk NB Card
- Ethernet driver version 1.1.1 for the Apple Ethernet NB Twisted-Pair Card
- Ethernet driver version 1.1.1 for the Apple Ethernet LC Card
- Ethernet driver version 1.1.1 for the Macintosh Quadra 700, 900, 950, and 800 computers, and Macintosh Centris 610 and 650 computers, with built-in Ethernet ports
- Ethernet driver version 1.0.1 for the Macintosh Quadra 840AV and Macintosh Centris 660AV
- Ethernet driver version 1.1.1 for the Apple Ethernet LC Twisted-Pair Card
- Ethernet driver version 1.1.1 for the Apple Ethernet CS Twisted-Pair/Coaxial/AAUI Card
- TokenTalk version 2.5.6
- Token Ring Control Panel version 1.0.1
- Token Ring driver version 2.5.2 for the TokenTalk NB and Token Ring 4/16 Cards

In addition, this Installer installs A/ROSE 1.2.1. This installer works on Macintosh computers running System 6.0.5 or later and on System 7 Macintosh computers.

On Macintosh computers with a 68040 microprocessor (Macintosh Quadra, Centris, and Performa 470 series), some programs may quit unexpectedly or cause the computer to "freeze" when you attempt to open them over a network. You can solve this problem by using the Network Launch Fix available on AppleLink or the Internet (ftp.apple.com).

This software should not be installed with AppleTalk Internet Router and an AppleShare client on a Macintosh with only 1 MB of memory. If this release of the Network Software Installer is being used in conjunction with an Apple Internet Router version 3.0 Installer, AppleTalk Internet Router 3.0 must be installed first. This will ensure that the AppleTalk Internet Router installation does not overwrite the newer AppleTalk software contained on Network Software Installer 1.4.4.

System Software 7.1.1 (System 7 Pro)

This folder contains U.S. and localized versions of Macintosh system software version 7.1.1 (System 7 Pro).

System Software 7.1.2 (Power Macintosh)

This folder contains U.S. and localized versions of Macintosh system software version 7.1.2 for Power Macintosh systems.

System Update 3.0

System Update 3.0 is a set of software enhancements that improves the performance and reliability of Macintosh computers running system software version 7.1, 7.1.1 (System 7 Pro), or 7.1.2 (for Power Macintosh).

System Update 3.0 provides all of the enhancements of System Update 2.0.1, Hardware System Update 2.0, and Hardware System Update 1.0, plus additional enhancements. See the document System Update 3.0 Read Me for details.

Thread Manager Extension 2.0.1

The Thread Manager is a System 7 extension that allows an application to make use of multithreading within its application context on all Macintosh platforms. Version 2.0.1 supports both 68K and PowerPC-processor applications and contains several bug fixes; see the file What's New In This Package for details.

There is a minimal once-a-year licensing fee for unlimited redistribution of the Thread Manager extension with your software product. Read the file Software Licensing Info for information on contacting the Apple Computer, Inc., Software Licensing Group.

Coming Next Month

Next month's CD will include as much sample code as I can extort from the DTS engineers, more *Inside Macintosh,* and the usual motley crew of hacks 'n' tools.

Alex Dosher, Developer CD Leader

Business & Marketing

Market Research Monthly

Home Learning: The Fastest Growing U.S. Software Market

One formula for success is to introduce a quality product into a rapidly growing market. This month, we tell you about the fastest growing U.S. market for software: home learning.

In the United States (as well as throughout the world), consumers are increasingly buying personal computers for home use; price points have decreased enough to make many consumers think of personal computers as must-have appliances. Today, consumers in the home buy more than half of all personal computers, according to *PC Week* magazine.

The market research firm Inteco reports that more than 25 million American households—approximately 25 percent—owned computers as of late 1993, and that another 25–30 percent of the remaining households are expected to purchase computer systems in the near future.

Consequently, every software category is selling more quickly into the U.S. home market, but none as quickly as educational software for children. Last year, the Software Publisher's Association (SPA) reported that sales of educational software to consumers in the home rose 66 percent, faster than any other software segment the SPA tracks. Sales of home education Macintosh software increased more than 80 percent in 1993, according to the SPA.

What should all these numbers mean to you? If you have a K–12 educational product, now is the time to take it to the home market. If you've been thinking about entering the educational market, this rapid growth means that there's never been a better time to do so.

This month we present data from Apple Computer, Inc., on how today's Macintosh customers in the home and K–12 markets configure their systems. This data should help

Inside This Section

A Few Tips (and More Resources) About the Home Learning Market

While you're planning your home learning product strategy, you may want to keep in mind the following points, which are provided by Sally Harris, home learning evangelist for Apple Computer, Inc. First off, your product needs to appeal to both children and their parents; parents are increasingly concerned about education, and want to do everything they can to enhance what they perceive as a faltering U.S. school system. Also, parents want to be involved with their children; if parents work, they're aware of their children's need to spend time with them. One way fill this need is by using home learning software together. Those are strong motives for parents to buy home learning products.

But they won't buy just any product. Parents are aware of what Sally calls "junk food software," violent games and other arcade-like software with little or no educational value. They want products that teach their children—whether they teach good values, how to read, write, think, do math, and so on.

Finally, teacher referrals lead to many sales of learning software to customers in the home. Parents ask teachers to recommend good software: More often than in other market segments, Macintosh products are recommended because of the dominant Macintosh share—well over 50 percent, according to most estimates—of the K–12 personal computer market.

For more information about the market for educational software, you'll want to look at *Apple Directions'* June Marketing Feature ("The ABC's of the U.S. Preschool Software Market" on page 23) and Developer Outlook ("Strategies for Success in the Early Learning Market" on page 29). In addition, the July and August 1992 issues of *Apple Direct* include a two-part article on cross-marketing educational products, moving products originally intended for other markets into education markets, and vice versa. All back issues of *Apple Directions* and *Apple Direct* can be found on the Reference Library Edition of the Developer CD (path—Dev.CD:Reference Library:Periodicals).

Finally, for a glimpse at what consumers are looking for in educational products, the April 1994 issue of *Atlantic Monthly* (available in most libraries) contains an excellent article called "Software for Kids."

you figure out how to build products that will best fit these customers' system requirements.

You'll be glad to know that the two market segments—on average—have very similarly configured systems, making it easy to sell one product to both markets. Currently, both markets primarily employ 68030-based systems. The majority of customers in both markets use System 7 or later, 256-color displays, 4 MB of memory, and 80 MB hard disk drives.

Also, both markets can be expected to rely increasingly on CD-ROM drives. Apple projects that at least three-fourths of current and future home consumer and K–12 school customers will buy CD drives with their computer systems.

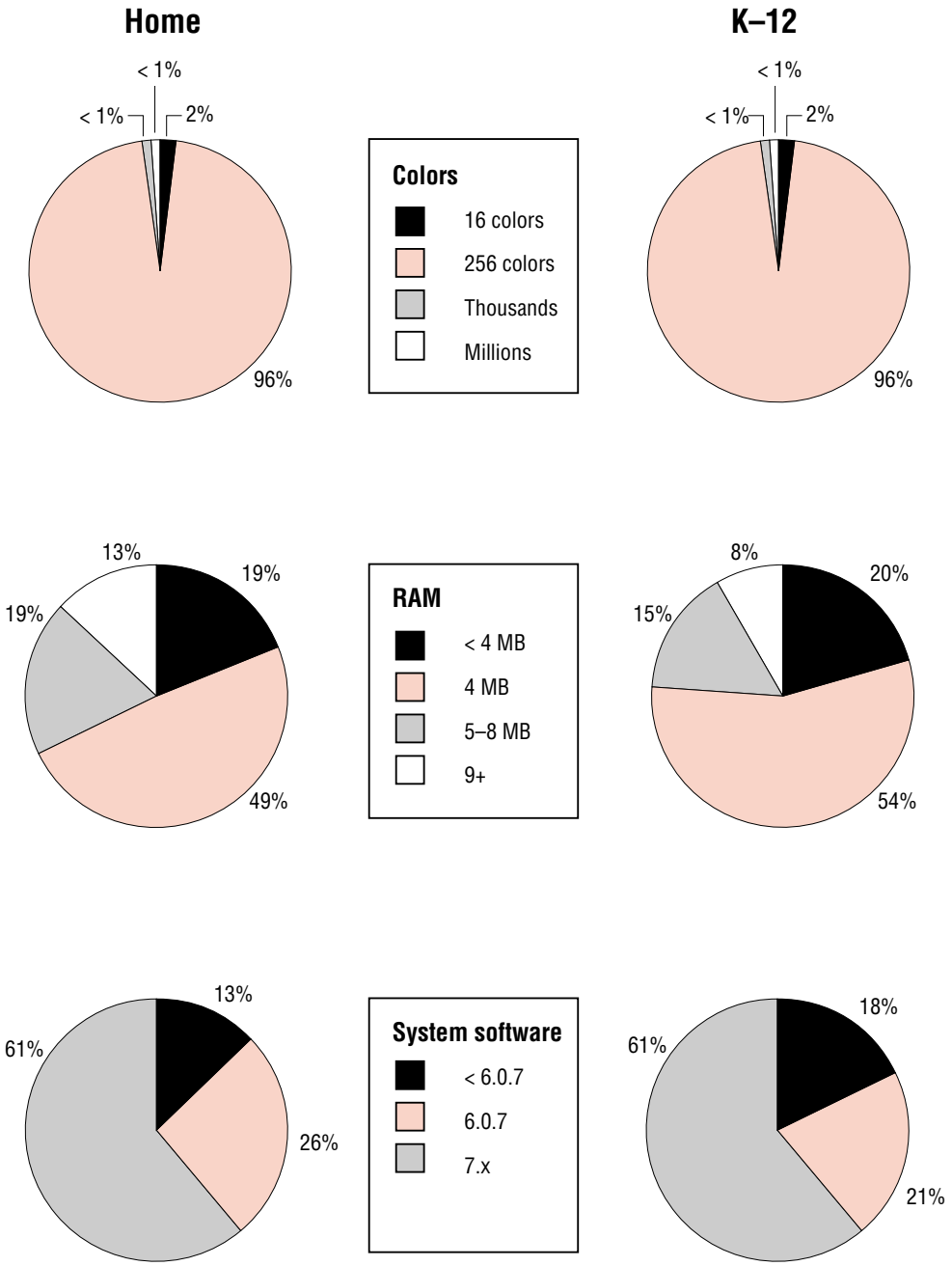
For more data, see the figure "U.S. Home and K–12 Markets: Configuration Data." Note that this data doesn't tell you what customers are looking for in home learning products; you can, however, find that information in the text box on page 19, "A Few Tips (and More Resources) About the Home Learning Market." ♣

Correct Phone Numbers

"Resources for Early Childhood Software Development" in last month's issue included two incorrect telephone numbers. We regret any inconvenience this may have caused. The correct numbers are as follows:

Children's Software Newsletter: *(713) 467-8686*
KidSoft: *(800) 354-6150*

U.S. Home and K–12 Markets: Configuration Data



Note: Data in some graphs does not add up to 100 percent because of rounding.

Marketing Feature

The Top Ten Channel Marketing Mistakes

By Frank Catalano, Catalano Consulting

You sweat. You toil. You create a wonderful product that will advance the cause of the personal computer by a decade—or at least a few weeks. It's beta-tested, documented, packaged, and shrink-wrapped.

Now all you have to do is sell it.

If you decide to work with resellers, anyone who's worked with them will tell you your challenges are just beginning. It doesn't matter if the channel is mail order, superstore, mass merchant, or software-only reseller: All have been raising the bar on the types and number of products they'll carry. That puts an increasing burden on you to make the approach that will land your products with the best possible resellers. (A quick aside: If you sell only directly to customers, resellers naturally become your competitors in your product category.)

Approaching resellers is a true marketing art. If you don't get it just right, you can inadvertently shoot yourself in the foot. It happens to all companies at one time or another—big and small, public and venture-capital, new and old.

How can you best succeed in this delicate area? Based on my experiences at Egghead and as a marketing consultant, I've put together a reseller reality check: a list of the ten most common mistakes developers make when approaching resellers. These mistakes are based on the real-life exploits of a variety of companies. (All names have been changed for the usual reasons.) But don't be embarrassed if you find your story somewhere in here: You're in good company.

Even if you read no further, just memorize these two words: *planning* and *communication.* Far too often, the channel marketing process is left solely to the sales force and isn't integrated into a developer's overall company strategy and product marketing plans. Channel marketing strategies should be planned as extensively and painstakingly as any other aspect of marketing, and communication between all parties—product managers and marketers, sales reps, and resellers—should be ongoing.

Some of these mistakes and the ways to avoid them may seem pretty basic. (For a summary, see the chart "Mistake-Avoidance Checklist" on page 23.) But as competition in our industry heats up and time to market shrinks, companies naturally look for shortcuts. Unfortunately, we sometimes shortchange ourselves on the very basics that could take companies from being marginally successful to having a huge hit.

Mistake #1: Don't Tell Them It's Coming

Understandably, you don't want to let your competitors know the exact ship dates of new products. But sometimes developers forget that resellers are their partners, not their competitors. Frequently, a reseller doesn't hear about a new product until (a) it's been listed in the "shipping products" section of a trade publication or (b) the reseller has been given a 30-day advance notice so it can place an order.

If you want reseller assistance with product marketing, let the reseller know your projected ship date in advance: the earlier, the better. Resellers generally must have three to four months notice before your product ships. That's the lead time they usually need for publishing catalogs and for making decisions about in-store promotion and point-of-purchase (POP) displays. If you view resellers simply as a mechanism for fulfilling demand and aren't interested in having them promote your product, then you don't necessarily have to give them much advance notice of your product ship date.

The ship date you initially provide doesn't necessarily have to be rock solid, but it must be realistic and well within the ballpark. The best tack is to give resellers advance notice and then keep them in the loop in case the ship date changes. Also, make sure you're aware of any of your resellers' drop-dead dates for pulling the plug on any marketing materials they are producing on your behalf—catalogs, direct mail pieces, and in-store signage.

Mistake #2: Focus on the Demo

One major developer used to start meetings with resellers by doing product demonstrations. These demos—even for a single product—would consume 50 minutes of an hour-long meeting. Then, after the lights were turned back on and the reseller woke up, the developer wanted to talk about marketing plans and business terms in the remaining ten minutes.

These meetings rarely accomplished anything other than showing off a new product, and they didn't allow enough time to discuss other important issues, such as marketing plans. By the time subsequent meetings were scheduled with the reseller to discuss important business basics, the deadline to start a promotion had passed.

If you choose to do a demo in reseller meetings, make it a top-line one that hits the highlights in not more than five minutes. Then get right to your business, marketing, and alliance issues. If resellers want to see more of a product, they'll ask. But you're not doing yourself or the reseller any good if you treat the meeting as you would a demo at a trade show exhibit. Instead, focus on the true purpose of your meeting: business negotiation.

Mistake #3: Be Unaware of Reseller Goals and Plans

To effectively work within the channel, you must know as much about a reseller's goals, plans, and competitive situation as possible. Do your homework, and incorporate what you learn in your marketing plan.

For example, you can't appropriately develop marketing activities for use at a particular reseller unless you know who shops there. One example, based on personal experience, is the number of software marketers who don't know that Egghead is more than a collection of retail storefronts. It also has a significant presence in corporate, government, and education channels, and a moderate presence in mail order.

Also, it's crucial to learn which product categories and customers a reseller prefers to emphasize, and to build your

channel marketing strategy to dovetail with the reseller's goals. Most resellers don't cut their audience broadly (no more than you would say that your target audience is "anyone with a computer"); resellers usually have specific kinds of customers they cater to. Also make sure you know if key resellers have partnered with any of your competitors for special promotions—perhaps for one that would run at the same time as your planned promotion.

There are a variety of sources of reseller information. Check resellers' annual reports, look closely at their catalogs and ads, read trade publications such as *Computer Reseller News* and *Computer Retail Week*—or simply ask resellers whom they perceive as their typical customers and where your product category falls within the resellers' priorities. Some resellers may even have clues in their product submission kits, if they have them. Those kinds of clues will tell you a lot about which resellers to approach and how to get their attention.

And don't ignore an obvious but often overlooked source of information: A trip to a retail reseller's store. Note the kinds and price ranges of products the reseller carries, how it organizes its wares, the kinds and quantities of POP materials it displays, and any other important information that might be difficult to get without actually making a store visit. Pick a product from the shelf and speak with a sales rep about it. What's the pitch? How much does the rep know?

If you can uncover this information, you won't waste your time or marketing dollars on a promotion that was handicapped from the start by being done in the wrong time frame or with the wrong reseller. Don't try to sell shoes to snakes—or to sell networking software in a shopping mall store.

Mistake #4: Create a One-Size-Fits-All Channel Promotion

Listen in on this (only slightly) paraphrased real-life developer-reseller exchange.

The developer: "Here's our channel marketing plan: In-store displays, shelf-talkers, a really great promotion with a contest, and everything's already printed and ready to ship to you."

"So we'll have exactly the same offer, POP, and contest as all of our competitors do?" responds the reseller.

"Well, consistency is important to us. And the POP really makes a statement: It's 8 feet tall, 4 feet wide, and holds 12 dozen units of the product."

At this point the reseller is thinking of a statement of his own he'd really like to make.

The developer took only its own needs into account and ignored the reseller's needs. What incentive does the reseller have for running—or more important, rallying its sales staff behind—a promotion that's not a unique point of difference for that reseller? Why give up valuable in-store real estate for a carbon copy of what a customer can see anywhere?

While in these budget-strained times a one-size-fits-all approach seems to be a cost-effective way of creating promotions, in reality I don't think this serves your needs. With the fragmentation of the traditional retail storefront channel into software specialty stores, computer superstores, office superstores, consumer electronics stores, warehouse clubs, and traditional department stores, the target audience and the sales messages it responds to will differ greatly from channel to channel, based on customers' experience and comfort level with personal computers.

A better idea is to develop a core promotion that contains elements that can be customized for each reseller you partner with. Also, a promotion doesn't have to run with every reseller. Select your key partners and customize the promotion's details, such as the contest, POP materials, or other elements, to give that reseller some differentiation with its competition—something that will appeal more to that particular reseller's customers. This approach also demonstrates to your reseller partner that you really want to help it be successful, as well.

And I can't emphasize this enough: Never, ever meet with a reseller to discuss upcoming promotions with everything already set in concrete. Your plans will sink accordingly. I suggest approaching a reseller with a concept, not a finished program. Treating the reseller as an afterthought in the process may be read as a lack of flexibility on your part and result in no promotion taking place.

Mistake #5: Make It Complicated and Don't Think It Through

Here's yet another True-Life Adventure, based on a real promotion with a contest: Coupons are attached to a monster POP. Customers tear off the coupons, which instruct customers to proceed to the demonstration computer in the store and do a product demo. Based on what they see in the demo, customers answer some product questions on the coupon and then take the coupon home and mail it to the developer. That enters them in a drawing to win a free copy of the product.

On the surface, it seems like a great idea: Customers can win something; you get them to see a product demo; and you capture a name to add to your direct marketing list.

What's wrong with this scenario? Three things.

One: There are so many steps to go through that many customers aren't likely to take part—unless they're already so motivated by the product they'd buy it without a complicated contest. Contests should have the fewest possible steps to fulfillment. Remember the old acronym KISS: Keep It Simple, Stupid.

Two: There's no motivation for customers to actually buy anything; there's nothing to motivate closing the sale. After all, if they have a chance to win the product, why should customers buy it now? You've deferred a purchase decision.

Three: None of the steps of this process involve the reseller itself, other than providing the venue for the demo. Because customers mail the coupons to you and you handle awarding the prize, the reseller has no incentive for being involved.

But wait, I hear some of you say: The customer really is the developer's. The reseller is just the middleman.

Sure. And customers are the ones who show up on your doorstep to buy products every morning at 10 A.M. Right.

If you're doing the marketing through the channel, it's a partnership. Ideally (and in the reseller's eyes), that customer is owned jointly by both parties.

To avoid making mistakes like these, keep contests simple and focused, think them all the way through to fulfillment, and make sure you ask yourself what will be in it for the reseller.

Mistake #6: Jump on the Tactical Bandwagon

A sure way to get drowned in the noise of the marketplace is to adopt a "me-too" approach to promotions. Just because everyone else is using a certain tactic doesn't mean it's right for your product.

Recently, I watched the progress of a company (let's call it SoftWare) that offered a good product in a category that had no

clear dominant player. As representatives of SoftWare ran down their list of possible marketing tactics, they mentioned that, of course, they would be doing a competitive upgrade.

The problem here is that none of SoftWare's competitors were offering a competitive upgrade, there was no dominant player to steal market share from, and SoftWare's product wasn't priced so high that a competitive upgrade would appeal to a whole new class of customers and thus draw in new business.

Instead, by offering a competitive upgrade, the perceived value of the regular product is lowered to the price of the competitive upgrade. This tempts resellers to pick up *only* the competitive upgrade. Many resellers can't carry multiple variations of a product because of limited shelf or inventory space. They'll often choose to stock only the version that will sell the most units—and if there's a cheap competitive upgrade with few qualifications that must be met for customers to buy it, resellers are more likely to carry the competitive upgrade than your regular product.

Keep throwing your tactics against your goals and see what sticks. Don't let the fact that a particular tactic worked well for a fellow developer push you onto the same bandwagon. Know your specific objectives and carefully choose the marketing tactics that will get you where you need to go.

Mistake #7: Don't Calculate ROI

It's easy to get caught up in the excitement of doing a promotion with a reseller, especially if the reseller has contacted you about a big promotion it's initiating. However, before you get swept away, make sure you do the math. What is it going to cost you? What do you need to get out of a promotion with a reseller to make it worth your efforts and money? Specifically, what levels of sell-in, sell-through, and awareness will give you adequate return on your investment (ROI)?

Have a specific target in mind for what you expect the promotion to accomplish (your goal) and what you would be willing to pay to get those results. Then, compare the goal to the results. Also, see how the budget fits in with other plans and programs you may have on tap.

It's important to thoroughly plan your budget for any promotional tactic, and clearly define how you propose to pay the reseller for it—whether it's co-op accrual, one-time market development funds, or a per-box marketing dollar allotment, or something else. Otherwise, you'll end up writing the reseller a blank check without knowing exactly what you're buying.

Without the math, you may be able to afford a first promotion, but not a second.

Mistake #8: Don't Meet Deadlines

You've worked your way through the goals, tactics, and the costs, and you've reached an agreement with a key reseller on a promotion.

Don't hang up the phone or leave the meeting without asking exactly what materials the reseller needs from you, and when. A developer once provided a major reseller with what looked like a crayon sketch, expecting it to be suitable for a catalog photograph that required a product package. What was needed was a finished or color comp box.

Agreeing on deadlines and needed materials sounds simple, but it's a continual nightmare for resellers: Developers commit without any knowledge of what materials are needed and when they're needed, or whether they

Mistake-Avoidance Checklist

Mistake	Avoidance
Giving reseller less than three months notice prior to product promotion (for example, a 30-day reseller notification policy)	Giving four or more months notice prior to product promotion (vague date is better than none)
Doing highly complicated promotion (and not thinking through to fulfillment)	Using the KISS method (fewest steps possible to fulfillment)
Creating one-size-fits-all promotion for channel (POP, ads, details), with no consideration of target audience differences	Customizing channel promotions (same core promotion with customized details)
Having no knowledge of resellers or their customers (for example, selling network software in a shopping mall)	Learning about reseller objectives and target audience
Adopting a marketing tactic because everyone else is (for example, competitive upgrades)	Knowing your specific objectives and how the marketing tactic gets you there
Offering no assistance to the retailer	Doing training, demos, and detailing
In channel meetings, just doing the demo	In channel meetings, talking about business and alliance issues and giving a top-line demo
Having no idea of budget or ROI (return on investment)	Knowing budget range, its structure (MDF, co-op accrual, and so on), and ROI target
Not caring about other vendors' activities in time frame	Knowing what the competition is doing in all of its marketing, including channel alliances
Commiting without knowing channel deadlines for materials	Knowing exact dates when materials will be available
Spending all your marketing dollars on a channel partner	Considering the channel part of an overall marketing strategy
Using marketing tactics without a marketing plan	Using a clear marketing plan that dovetails with your company's business plan

can deliver on those dates. This can lead to the charming experience of having a catalog page run with the picture of your old product box, or having another developer's product substituted for yours—and you still must pay the bill because you blew the contracted deadline.

On the flip side, it's also important to know what the drop-dead date is for pulling out of a promotion or substituting a different product. That way, you can ensure that if your product ship date slips, the marketing doesn't start before the product is ready.

Before proposing or agreeing to a channel promotion, know the exact date your advertising materials will be available and the format required (line art, color comps, color photographs). Stick to it; when you can't, be sure to communicate any time slips to the reseller on a timely basis.

Mistake #9: Leave Implementation to the Reseller

Now that the promotion is set and the contracts are signed, you can't walk away and wait for the next big order. Communication with the reseller is an ongoing process. And it's a multilevel process that involves many people at the reseller company, from its corporate headquarters to its field sales staff.

Don't expect that a reseller's sales staff will know anything about your product. Don't expect that they've read the copy on your packaging or in the trade publications. Don't offer to send a thick packet of reviews from the *Skokie Software News*—or, as some developers do, send an empty box to each salesperson to have them read it for product information. Also, don't expect reseller sales staffers to load and view demos and send you filled-in questionnaires.

Do prepare simple, straightforward sell sheets that focus on how you compare to the competition, how to qualify the customer, and what the top three benefits (not technical features) are. If you have the resources, coordinate in-person training and demos at reseller sites. Either hire detailers, or have your staff make spot checks of resellers to make sure the right product is stocked in the right place when the promotion is supposed to run.

One recent estimate in a reseller trade publication is that 50 percent of stores are, in some way, not in compliance with promotions arranged by their head office; either the wrong signs are put up, or the product isn't in stock, or some other problem crops up. Continued communication and follow-up at all levels is one way to keep that percentage low.

Mistake #10: Dance Only With One Partner

The most important thing about choosing key reseller partners is the *s* after *partner.* Spending all your marketing dollars and making all your plans with a single reseller can limit your ability to reach your target audience: You won't generate demand from customers who shop at other resellers.

Carefully choose several partners that appeal to different segments of your market and work with them closely. Let them see copies of your product launch plans so they know how their role fits in with the overall launch. This will allow each reseller to maximize its own sales impact.

The Big Picture

Consider channel marketing part of an overall marketing strategy. You wouldn't start executing marketing tactics without having in hand a marketing plan that clearly dovetails with your company's overall business plan. Similarly, you shouldn't execute channel marketing tactics without knowing how they support your overall marketing goals.

Successful channel marketing comes down to the two things mentioned at the beginning of this article: planning and communication. Take the time up front to determine your goals and how your channel marketing fits into and supports your overall marketing goals. Then, communicate those goals regularly with your key channel partners. Make that communication a two-way street.

Resellers really do want to sell more product, and with good channel marketing, it could be *your* product. ♣

[Author's note: My special thanks to Carolyn Bickford of Edmark (who was a long-time marketing colleague at Eggbead) for reminding me of some of the more amazing channel marketing proposals we encountered.]

Frank Catalano is the principal of Catalano Consulting, a marketing strategy and market analysis firm for software and interactive technology companies located in Sumner, Washington. Frank is also a former marketing manager for Eggbead Software and an Apple developer tools organization.

More Catalano Expertise Available

If you found this marketing article useful, you can read more from this author in the book *The High-Tech Marketing Companion: Expert Advice on Marketing to Macintosh and Other PC Users.*

This book, by Dee Kiamy and the editors of *Apple Directions,* is packed with practical, proven techniques from a variety of experts for solving your toughest marketing problems. To order, call Computer Outfitters at (800) 260-0009 (U.S. and Canada) or (406) 758-8000 (other countries).