Apple Lisa Computer
Technical Information

Apple Lisa Computer: MacWorks Plus Developement Information

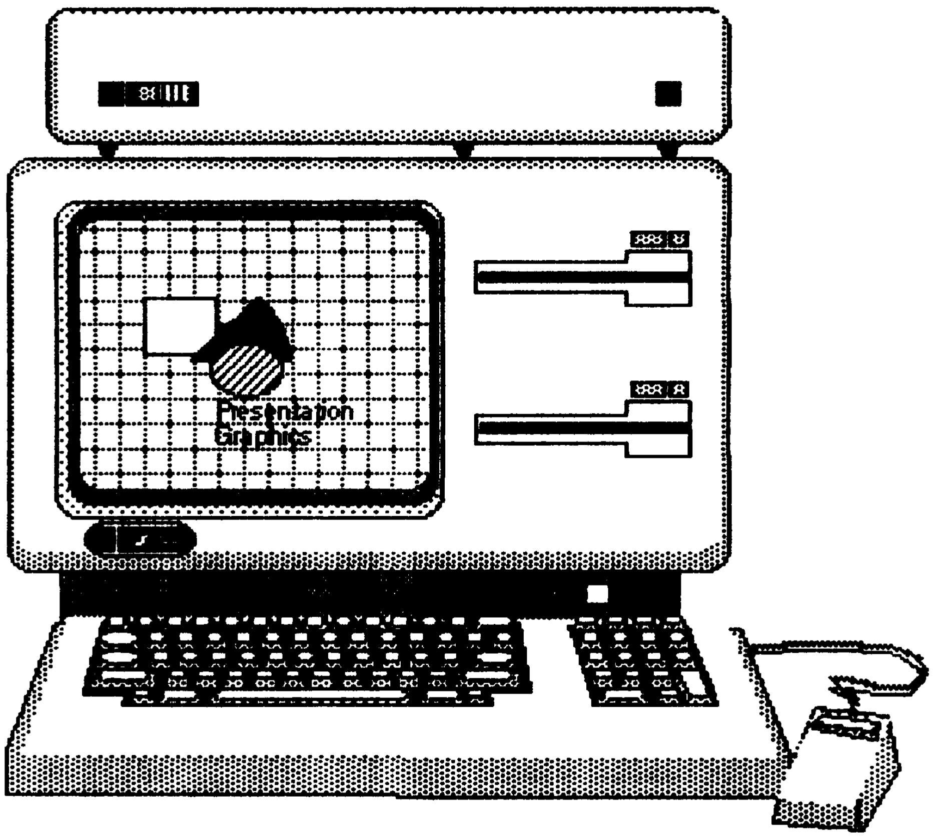
Lisa Computer:
1983 - 1985

Printed by: Macintosh Picture Printer 0.0.5 1999-01-11 Printed: 1999-02-01 18:54:45

Apple Lisa Personal Computer

1983 to 1985

MacWorks Plus Development Info



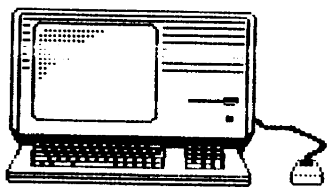
David T. Craig - 736 Edgewater, Wichita, Kansas 67230 - (316) 733-0914

"LisaMacWorksPlusDevInfo 0.PICT" 246 KB 1999-02-01 dpi: 360h x 363v pix: 2539h x 3236v

MacWorks Plus: **Making A Lisa Speak Macintosh**

4 pages

Charles T. Lukaszewski
Sun Remarketing, Inc.



Apple's Lisa computer never had a chance. Its windows and mice were ahead of their time. A nice sports car cost less than a minimally-configured Lisa system. And the most significant obstacle, which shaped Lisa's decline and recent return, was that Steve Jobs didn't want it, and for a very good reason. Jobs' pet project, code-named Macintosh, was introduced one year after the Lisa at one-third the price.

But despite the Lisa's dismal marketing history, it has persevered through the efforts of Sun Remarketing in Logan, Utah, and a handful of other organizations, the Lisa continues to be sold, serviced and supported. My role in this effort has been to develop for Sun Remarketing a new Lisa operating system called MacWorks Plus. MacWorks Plus allows the Lisa to perfectly emulate a Macintosh Plus. It is the successor to a program called MacWorks, which was an imperfect emulation of a 64K ROM Macintosh. As this is a topic of natural interest to the Mac technical community, this article will analyze how the old MacWorks functioned, and then address some of the obstacles encountered in developing a new one.

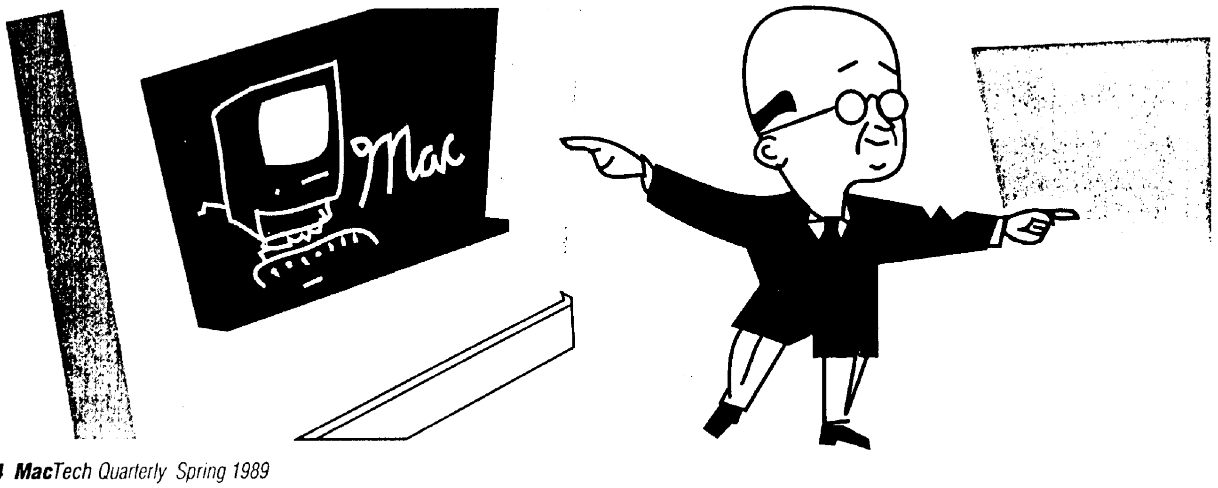
Historical Background ∆ It is asserted in both the Lisa user community and some offices in Cupertino that Apple wants to kill the Lisa. Though hindsight clearly provides the empirical basis for such a claim, it is no longer true. Apple's new willingness to permit the development of MacWorks Plus is a welcome confirmation of John Sculley's changing of the guard.

The survival of the Lisa has always hinged on the Macintosh. Early in 1984 it was recognized that the Lisa Office System alone couldn't finish off Apple's inventory of Lisas: the Lisa would have to speak Macintosh. And so the spare time project of an Apple employee named Rich Castro was given official status as MacWorks. This was an implementation of the 64K ROM and was sold in conjunction with an upgrade kit which converted the Lisa's rectangular pixels to square ones. The Lisa managed to remain attractive because you could put up to two megabytes of memory in it as well as a 10MB hard disk This was at the time that the Mac sported a meagre 128K of memory, hard disks were nowhere in sight, and the "Please Insert The Disk" message was a good friend to us all.

With the introduction of the Mac Plus, however, the Lisa lost virtually all of its remaining competitive edge. MacWorks was not capable of supporting HFS, Apple Share or the increasing number of programs which counted on facilities in the 128K ROM and new System Software. These problems were felt most acutely at Sun Remarketing. Sun is the only company in the world authorized by Apple to sell and service its out-of-production equipment. For some time it had also been a sounding board for all of the complaints about the Lisa. Something had to be done.

MacWorks ∆ The original MacWorks was the first port of the Macintosh Operating System to another computer. The architectures of the Lisa and the Macintosh are radically different: the Lisa had a memory management unit that supported context switching and logical memory mapping, but the Macintosh used hard-mapped address lines; the Lisa had three expansion slots with DMA support, while the Macintosh had none; the Lisa had an intelligent keyboard processor with programmable power-on and power-off, but the Mac's keyboard was about as smart as an aardvark. To make the Lisa speak Macintosh, it proved necessary to throw away the Macintosh hardware interface and provide a new one.

Of course, Apple already had a hardware interface for the Lisa. In fact, they had two to choose from: the Lisa Office System/Lisa Workshop interface and the Monitor interface. The Monitor Operating System was the Lisa's first development environment. It was essentially a port of the Apple /// Pascal Workshop, and as such it inherited the same text-only output with single-letter command lines that we all knew and loved eight years ago. Monitor OS became Apple's choice for the MacWorks hardware interface for a very good reason: the structure of its device drivers and their I/O command blocks was almost identical to that used in the Mac.



So it was that the Monitor device drivers, some freshly-written interrupt handlers and the source code to the Mac Toolbox were united into a product called MacWorks. At this point Apple also made some design changes to the Lisa and began to sell it as the Macintosh XL. It is important to note here that these decisions about the design of MacWorks established specific limits on its performance and its compatibility with Mac applications, and set the stage for the development of MacWorks Plus.

The Monitor Operating System ∆ In order to get any type of Macintosh emulator running, a system must go through some type of bootstrap and load procedure to install the emulator and protect it from being detected and/or overwritten. Also, the software environment must be made to look as much as possible like that on a Mac before starting the emulator.

MacWorks did not address this problem very well. Because the Monitor OS ran underneath MacWorks, certain memory regions had to remain intact across all soft restarts. Several of these regions were in low memory where the Mac normally locates some of its own information, including the trap dispatch tables. Furthermore, Monitor wasted memory because its file management kernel was required during bootup but then sat idle while MacWorks used its own. This difficulty can be understood best by analyzing the bootstrap technique used by Monitor.

The Lisa is the last Apple computer that did not include an operating system in the ROM. When a Lisa is powered on, there is only enough code in the ROM to perform diagnostic tests and boot device selection. Of course, this is the ideal machine on which to port the Mac OS, especially given the ease with which the Lisa's MMU can map the emulator's physical memory to the $00400000 range where you would expect it on a real Macintosh. Once a boot device is selected, the ROM completes stage zero of the boot process by loading sector zero from that device to address $00020000 and beginning execution.

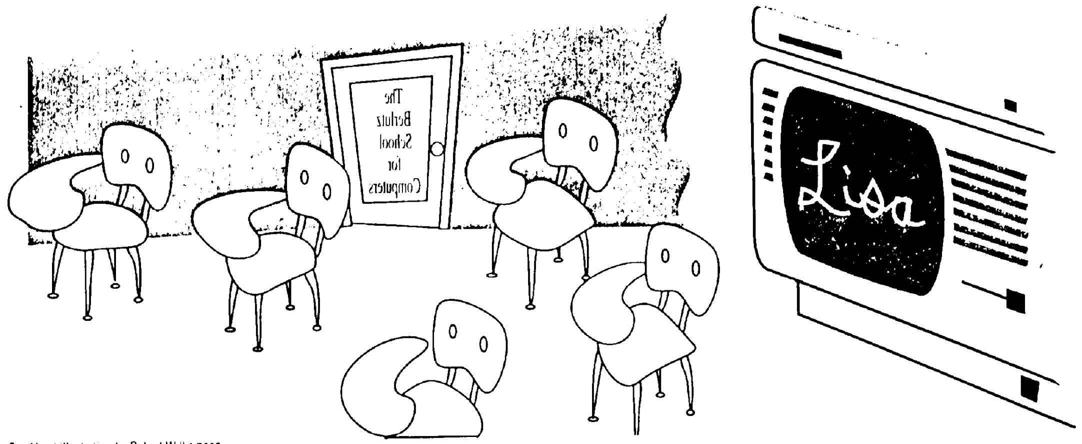
Under Monitor & MacWorks, the 512 bytes of code in sector zero must in turn load the initial operating system kernel. This kernel may be located anywhere on the disk, so it is necessary to put a pointer somewhere that the stage one bootcode can find. This is accomplished by putting an offset into the second word of the second sector of the disk. The stage one bootcode retrieves this value, subtracts two, and positions the device at this sector. It proceeds to load eight sectors to address $00020800 and transfers control.

Monitor's boot process is extremely flexible, but also extremely dependent on specific files. Every bootable Monitor disk contains at least four files: MON.LOADER, MONITOR.OBJ, CONFIG.DATA and BOOTFILES.DATA. The MON.LOADER file contains those eight sectors that are loaded by the stage one bootcode, and the use of the offset in sector two is an inexpensive way to avoid a directory search. MON.LOADER, however, does contain directory search code, and uses it to pull in a sequence of files. First, the CONFIG.DATA file is read into memory page one. This defines several of the initial low-memory globals required by Monitor. Second, MONITOR.OBJ is read much higher in memory. Finally, BOOTFILES.DATA is read. In this clear-text file are a sequence of filenames which are to be read to complete the boot process. An entry in this file consists of a one-byte file type field and a fifteen character name. These files are loaded by their order of appearance. When this is complete, control transfers to MONITOR.OBJ and the Monitor OS starts.

While this approach works, it imposes unacceptable resource limitations on a MacOS port. The most obvious difficulty is that you end up with two file-oriented operating systems, one of which is never used after startup. In addition to the memory wasted by the unused code, a great deal of disk space is idled for overhead like directories for that second operating system. This is of particular concern on a hard disk, since some type of partitioning scheme is going to be required to separate the operating systems. But what does MacWorks do? Why, it just copies a complete image of its 400K disk onto the hard disk, and then takes an extra 100K for good measure. As MacWorks only requires about 100K to begin with, the waste is enormous.

This approach also creates performance limitations because every Mac application has to go through two levels of calls to do hardware dependent tasks. This includes drawing to the screen as well as sound generation and storage device I/O.

The Macintosh Operating System ∆ If you've ever seriously perused the list of low-memory system globals in *Inside Macintosh*, it becomes obvious that the Mac Operating System is really portable. "Just look at all of those hardware dependent parameters, Beaver!" "Gosh, Wally, there's screen size and pixel resolution and device addresses and interrupt vectors and memory layout and everything. This port'll be easier than eating brussels sprouts!"



FreeHand illustration by Robert Williamson

In all seriousness, Apple has done a fantastic job of developing an operating system which is consistent across an entire line of computer systems, and which has enormous potential for modification and expansion. Indeed, these qualities have been exploited by Apple from the first. Apple has chosen a relatively expensive method by which to achieve this level of consistency: a centralized configuration database in the low-memory of each Macintosh. It is expensive because it costs cycles to look up information that isn't hard-coded. But who cares about expense when it also makes a reconfiguration of that operating system on a different architecture extremely simple?

This is not to say that Apple has taken all of the work out of doing an operating system port. It is a time consuming and often frustrating task. However, they have made it possible to cleanly separate a hardware dependent kernel from the rest of the MacOS. If one replaces this kernel with one for a different computer and sets the low-memory globals appropriately, the rest needs no modification. This is what Apple did for the original MacWorks.

MacWorks Plus! Δ Late in 1987 it was clear that something had to be done about MacWorks. In addition to the problems already described, the fact that it was based on the 64K ROM from the original Macintosh meant that it could neither run System Software beyond the 5.3/3.2 release, nor could it run many applications which depended on later versions of the ROM and System Software. Also, Sun Remarketing had developed an 800K floppy disk drive for the Lisa as well as HFS, but the modifications took up additional memory and approximately doubled the system overhead on a hard disk. Worse yet, MacWorks would only use 32MB of Sun's new 40MB internal hard disk.

Today, MacWorks Plus has eliminated every one of the performance and compatibility problems that plagued its predecessor. The vital statistics speak for themselves. Overall operating system speed has been improved 25% and Quickdraw™ operations run 30% to 40% faster. System Software 6.0.2 runs perfectly, including MultiFinder™. Applications like MPW 2.0.2, HyperCard 1.2.1, Tops 2.0.1, Lightspeed C 3.0, TMon 2.81, FullWrite 1.0, Excel 1.5, Crystal Quest 2.2 and many more also work perfectly. In addition, MacWorks Plus supports a SCSI interface and disk drives of any size.

The enhancements in MacWorks Plus can be divided into three categories: bootcode, hardware interface and operating system. The bootcode was completely rewritten, and as a result we were able to eliminate wasted code in memory and simultaneously cut disk consumption down to 200K. The multi-file approach that the Monitor OS used was discarded in favor of a single stage two loader which handles memory management, machine initialization and loading of the MacWorks Plus ROM Image. There are additional advantages to this solution. First, operating system globals do not need to be preserved, so soft restarts are free to clear all of the available memory. Aside from being good programming practice to erase all signs of a previous operating system, this is crucial to compatibility for some programs. Second, and even more important, we could put every system global in its proper location. For example, the trap dispatch tables, which were located at $00411000 under MacWorks, are now where they should be.

The second category of enhancements involved rewriting the hardware interface. This was also the most time consuming portion of the project. Wherever possible, we built on code from MacWorks, adding control calls and functionality that were missing. Unfortunately, this avenue was only open for the hard disk driver, AppleTalk drivers, and small pieces of the others. The Sony, Sound and Serial drivers were written from scratch. Every driver in MacWorks Plus now conforms to the specifications in the five volumes of *Inside Macintosh*.

One very prominent difference between the 64K and 128K ROMs — the handling of device drivers in the ROM — also had a major impact on our implementation of a new hardware interface. In the 64K ROM, drivers are simply code resources that have hard vectors installed in the unit table. Fonts and cursors are stored in the same fashion, as well as other system information. However, in the 128K ROM Apple decided that since these things are resources anyway, they should be stored as resources. Hence, the memory from $004176F8 to $0041FFFF on a Mac Plus mimics an open resource file, and special hook flags were set up at $00000B9E and $00000B9F to access it.

Obviously, all of this information, including the ROM resource file map, is hardcoded. The longword at $0040001E points to the start of the ROM resource file map, which has a format that differs greatly from standard maps. Then the resource data begins and continues until the end of the ROM. We decided that the simplest way to implement our device drivers was just to generate our own complete ROM resource file and append it to the MacWorks Plus emulator. We included all of the generic resource types in the 128K ROM like CRSR, FONT, MDEF and WDEF, and then linked in our own. For example, the Sun Remarketing logo which appears during bootup is stored as a PICT in the ROM resource file. A highly problematic piece of code was then developed to render this standard resource file into a ROM image file and append it to the MacWorks Plus ROM image. We were highly successful with this technique. Again, we chose to do a little extra work to perfect the emulation rather than to kludge it and risk incompatibilities.

Performance improvements are largely the province of the operating system, the third category of enhancements. There are two types: new features and upgraded features. New features include the HFS File Manager, SCSI Manager, List Manager, and all of the other system trap calls introduced in *Inside Macintosh Volume IV*. Upgraded features include rewrites of several slow subroutines, including Quickdraw calls. While these operating system enhancements are the most visible of the differences between MacWorks Plus and its predecessor, they also took the least amount of time to complete. Since all of the MacOS that the user sees is not hardware dependent, it can be moved between machines with little effort. For example, SCSI support is one of the most dramatic improvements for the Lisa, but it took perhaps the least effort of all from a software standpoint. So long as the SCSI address lines are mapped into precisely the same locations in the Lisa as they are in the Mac, then almost no code needs to be rewritten to support it.

Issues for Developers Δ Any application you write for the Macintosh should work under MacWorks Plus subject to one proviso: *Follow the rules*. There are many software packages which do not adhere to the guidelines set forth in *Inside Macintosh*, and thanks to a great deal of determined effort and some clever system patches they will now run on a Lisa. In becoming something of an expert in tracking down such violations, it is my experience that they generally fall into two categories: environment checks and hardware interfaces. Those of you who read comp.sys.mac.programmer have already seen my tirade about non-standard environment checks, so I will be brief. There are many applications which don't bother to use _SysEnvirons to determine the runtime environment, either because it is quicker to read the low-memory global, or because _SysEnvirons is incomplete. A good example is the direct check for a floating-point coprocessor at $00500000 that new Microsoft applications perform.

Hardware interfaces are the second problem area. There are applications, for example, which directly reference the VIA chip in the Mac. On a Lisa, the VIA is in a different location, and so the low-memory global which points to it is also different. However, the address of the VIA in a Mac Plus, $00EFFFFE, is the same as the address of the screenbase register on a Lisa. As you might imagine, this generates some very pretty crashes.

I cannot stress enough that this is a Macintosh issue as well as a Lisa issue. Given the impending release of System Software 7.0 when context switching may become reality and references to low-memory globals may cause privilege violations, it is smart programming practice to depend as little as possible on the physical environment. It is true that Apple Developer Technical Services has not been consistent in its positions. But that in-and-of-itself is not a reason to disregard their current recommendations. Besides, I'd rather have the inconsistent positions and volumes of accurate technical information that you get from Apple than the volumes of inaccurate technical information and total absence of developer support that you get from IBM.

The opinions just presented were not paid for by Apple Computer.

Environment Checking and Debuggers Δ If for some reason you do need to determine if you are on a Lisa, _SysEnvirons will no longer do it. _SysEnvirons will return a Mac Plus. If the following comparison is true, then you are running on a Lisa:

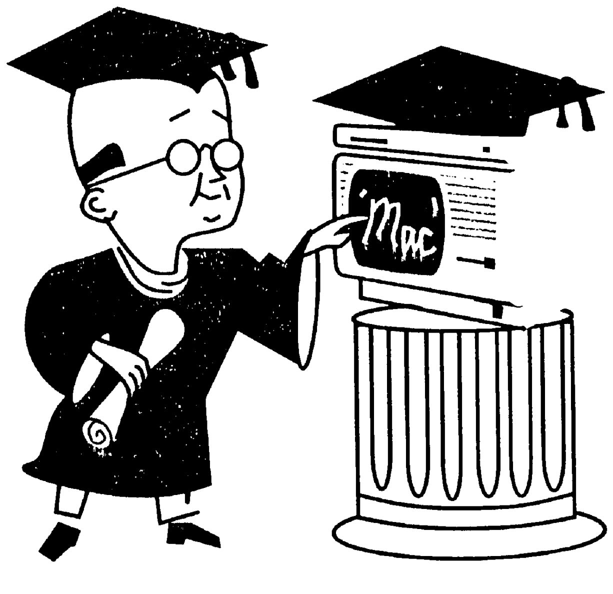
```
CMPI.L     #'MACW',$00400040;        Lisa check
```

Finally, a few words need to be said about the use of debuggers under MacWorks Plus. The popular debuggers MacsBug V5.5, Nosy Debugger, and TMon V2.xx all work off-the-shelf. Nosy and TMon need no modification to run, and MacsBug requires only a replacement of the keyboard input routine. This done to the memory image during system bootup to avoid requiring a special version. And of course, since MacWorks Plus is a robust implementation of the 128K ROM, the ROM debugger is also available by pressing the NMI button on the back of the Lisa.

Why MacWorks Plus! Matters to Developers Δ Software development is a very I/O bound activity. As programmers, our productivity increases at a rate inversely proportional to the compile-link cycle time. Those who use our applications also benefit when we make sure that they are robust and non-hardware dependent. The Lisa with MacWorks Plus can make important contributions in both of these areas. As a heavy user of both Lightspeed C and MPW, I have been won over by the speed difference between my Lisa and my Macintosh SE. On compile-link cycles of 30 seconds or more, the Lisa with a Sun internal 20MB or 40MB hard disk is always faster than the SE with any hard disk whose access rate is over 30ms. To compile MacWorks Plus from scratch under MPW takes 20 minutes on the SE and 17 on the Lisa. The I/O difference is even more pronounced given that the Lisa is running 3Mhz slower than the Mac.

A Lisa with MacWorks Plus is also valuable as a sort of ToolBox Verification Suite, and is far cheaper than a Mac II with A/UX™. Its 12-inch screen is a godsend for using multi-window development systems effectively, especially if a large monitor and a video card are outside your budget.

Acknowledgements Δ Though I performed all of the development work for MacWorks Plus, I am deeply indebted to a number of individuals, without whom it would not have happened: Bob Cook and Dale Gilbreath of Sun Remarketing; Dave Ramsey, Dan Allen, Steve Schwartz, Rich Castro and the Mac ROM team at Apple; Dave Heinen, Steve Jasik, Sam Neulinger, and Bruce Auerbach. Finally, a very special thank you goes to Harold Stuart for code suggestions, Lisa hints, shared Lisa frustrations, people pointers and for finding something new to show me each day in Logan, Utah for a month.



Chuck Lukaszewski is the president of Open Systems Architects, Inc.™, a Minneapolis-based firm that plans, installs and manages computer data networks for corporate clients. In addition to his Lisa and Macintosh work with Sun Remarketing, he has developed applications software for and managed a Gould high-speed image processing system at the Minnesota Supercomputer Institute, and authored software for a variety of other organizations around the Twin Cities. His opinions about Intel microprocessors versus Motorola microprocessors are, well, his own.