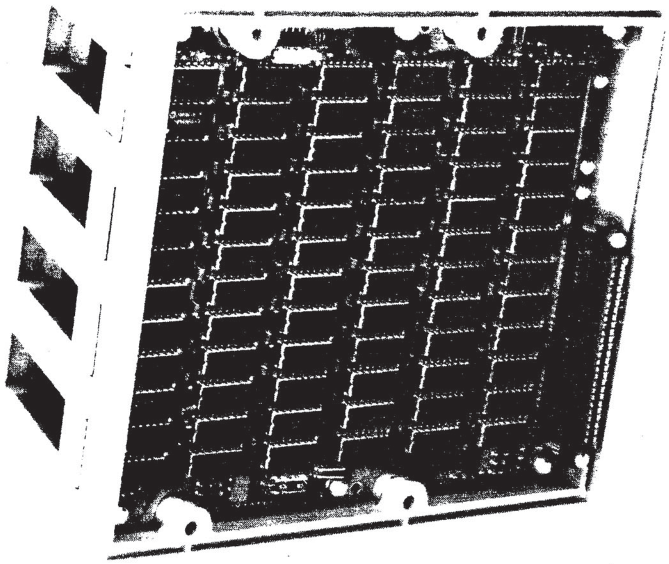
# Tektronix 4050 Series Extended Memory: Working Smarter



*Unobtrusive packaging of the Tektronix Extended Memory File Manager within the 4052/4054 ROM Pack Expander unit was just one of the benefits of thoughtful engineering when satisfying the need for more user memory on the 4052/4054 Desktop Computers.*

**by Brian Diehm**
**Tektronix, Inc.**
**Wilsonville, OR**

*TEKniques Vol. 6, No. 3 reviewed the benefits of the recently introduced Extended Memory File Manager. A glimpse beneath its surface and behind the scenes explains its success.*

Rather than implementing a virtual memory scheme, Tektronix chose to provide additional memory to 4052/4054 Desktop Computer users through a simulated flexible disk interface. In doing so, Tektronix satisfied user needs without burdening them with the expense of a long development cycle and the risk attendant with the "technologically advanced" solution (see sidebar - "An Engineering Approach"). This "electronic disk" also provided users with a unique method for handling overlays and data files.

### A Disk in Disguise

The electronic disk, packaged within the optional 4052/4054 ROM pack expander unit, has no switches or indicators, nor does it take up any of the ROM pack slots in the expander. Its unobtrusiveness fits neatly with its nature of being a strictly internal high speed device. The programmer is aware of the device only by programming it; the ultimate application user needn't be aware of it at all.

To retain the contents of the electronic disk when the 4050 is powered off, power to memory is provided through a small separate unswitched power supply. A battery backup ensures data retention through power failures and while the 4050 is unplugged for short periods, e.g., moving it from location to location. The power supply with its battery backup is placed under the keyboard of the 4050 System, out of sight.

The electronic disk has its own ROM pack which is built into it. This guarantees that the ROM pack will always be plugged in, and plugged into a specific slot in the bank of available slots. This particular electrical location is important for the operating scheme of the electronic disk.

As each 4050 ROM pack is powered up, the functions it supports and addresses for these functions are saved in a table in 4050 memory. Later, when BASIC needs a function supported in a ROM pack, it scans the power-up tables in the same order as they were powered up until that function is found.

The placement of the electronic disk ROM pack ensures that it is powered up last; thus, when a flexible disk ROM pack is present, it will already have been powered up and its entries made in memory. On power up, the electronic disk scans these tables for flexible disk ROM pack entries. If found, the electronic disk "trades places" with the flexible disk. When BASIC scans these ROM pack tables, it finds the disk calls in the electronic disk table before the flexible disk entries. Thus, the electronic disk "intercepts" all disk commands and passes on only those not targeted for itself. Intercepting all disk commands does add some time to those commands ultimately targeted for the flexible disk, but the speed of the electronic disk makes the overhead negligible.

### A Compatible Link

As a convenient interface for 4052/4054 users, the electronic disk behaves like the existing 4907 File Manager. While in theory an electronic disk is a solid state implementation of a hard or flexible disk, the physical differences of the two result in different usage patterns. Consequently, most of the functions in the electronic disk exactly match their counterparts in the 4907, but the 4907 flexible disk has some capabilities that are inappropriate for the electronic disk. Therefore, the electronic disk accepts all 4907 commands and ignores those which are superfluous.

Because it is nonremovable, the electronic disk performs an automatic mount function at power up, and if the medium is not formatted, automatically formats it as well. In fact, the first time the user powers up the electronic disk by turning on the 4050 system, a short (2 to 5 second) delay for formatting may be noticed, a delay that will not be present at the next power up. Because the electronic disk is always "mounted," both mount and dismount commands aren't necessary but will be accepted and ignored. Of course, a user may expressly format the electronic disk.

Although the electronic disk doesn't compress files in place (restructure and compact the data on the disk), it still accepts the compress command.

Like the 4907 flexible disk, files on the electronic disk may be named with up to 10 alphanumeric characters followed by an optional extension of up to four alphanumeric characters. Unlike the flexible disk, library levels and passwords weren't seen as adding any real utility to a local dedicated device.

# The 4050 Series Electronic Disk
# An Engineering Approach

When confronted with a design challenge, engineers often think in terms of technological solutions and "greater horsepower." The intuitive approach is to apply the latest technology - faster more powerful chips, exotic algorithms - perhaps bypassing an easily applied method that solves the problem by working smarter, and yet uses available technology. An example of a design challenge was extended memory for the Tektronix 4050 Series Desktop Computers.

The problem was lack of user memory space. The 4050 Series Desktop Computer had reached the 64K memory address limit. Since it featured a BASIC interpreter, any scheme for more memory had to comply with the *modus operandi* of BASIC programmers; the user interface was critical. In addition, adding more memory was being done in conjunction with other major updates to the product which meant very definite time limits for a solution.

The engineering team identified several approaches and fleshed out the characteristics of each:

1) A virtual memory approach (bank-switching the address space) provided a transparent user interface, but updating the BASIC interpreter would take extensive effort and might not be completed within the time limits.
2) A monolithic array of characters provided a gigantic string for storing alphanumeric strings and numeric values. While the engineering effort would be trivial, it placed an unacceptable interface burden on the user.
3) An internal magnetic tape look-alike matched the integral tape drive of the system, and was familiar to the user. Although faster (no tape seek time), its functionality was limited by sequential-only access and other tape-related restrictions. It would, however, be easy and quick to implement.
4) A Pascal-like heap provided additions to the BASIC language to support structured data types, and an area in which the user would define and store them. This method would be fairly low risk except in defining the BASIC interface.
5) A nonremovable disk provided the user with a device whose operation was nearly identical to the existing flexible disk peripheral to the 4050 Series, the Tektronix 4907 File Manager. Since the nonremovable disk was internal, however, the absence of seek and head loading time would make it faster. There would be some inherent schedule risk and some forced overhead upon the user for memory management.

Looking at all the alternatives, management and engineering settled on the electronic disk. It was the best suited to overlay management, an important consideration, for no matter how much direct memory is provided, at some point a large program needs to be divided into overlays. The user interface of the electronic disk would be familiar to users of the 4907 File Manager, and for users not familiar with the 4907, the interface had been proven to be easily learned. If the scheduling became unacceptable, the internal magnetic tape look-alike could be substituted.

It is important to note that the realization of the ideal user interface did not occur until after all the alternatives were considered. Without the engineering investigation, the intuitive assumption of the appropriateness of the bank switching virtual memory scheme would never have been questioned. The result would have been a late product, one more costly, and most important, one less suited for the users' needs.

The electronic disk satisfies the original need for more memory that led to its inception, and goes beyond. But the engineering effort succeeded not because of the application of the latest and greatest in technology. In fact, it uses only tried and tested methods. Its success lies in accurately stating and defining the need to be fulfilled and thoroughly investigating all alternatives instead of leaping to the intuitive approach. The result is a product carefully aligned with user needs, not engineering preconceptions. Like the product itself, that's an example of working smarter!

Therefore, while the electronic disk firmware will accept complete filename specifications appropriate to the flexible disk, all but the filename and extension are ignored.

The electronic disk assigns different meanings to the values returned from hard error status calls. And when calls to set time and return time are passed to the flexible disk, if no flexible disk is on the system, set time is a no-op and time returns a null string.

A tricky command to implement was the status command CUSTAT which returns the status of all disk controllers on the system. The electronic disk first passes control to the flexible disk, if present, then intercepts the data string returned to add information about itself.

Several commands are unique to the electronic disk. One is TESTEM which performs either destructive or nondestructive tests of the random access memory of the electronic disk. Primarily a service tool, TESTEM often can pinpoint failures to the individual integrated circuit. Other calls also assist service personnel and include means for reading, writing, or dumping any portion of the memory to the screen.

**Strategy in Storage**

The electronic disk does not naturally have any "blocking" or partitioning. But it would be difficult to extend files or recover space left by deleted or shortened files without blocking; therefore, some method of logically structuring the monolithic block of memory is necessary. The firmware implementation treats the device as if it were divided into blocks of 256 bytes each (analogous to the sectors of a conventional external disk).

These blocks in no way restrict the operation of the electronic disk. The firmware can read or write a twenty byte structure in the middle of a block without having the artificial need to read or write entire blocks. Similarly, the firmware can read or write contiguous data files (files where the blocks are assigned in sequential order) in a single operation that ignores block boundaries. On the other hand, it can carefully align scatter files (files where the logical sequence of assigned blocks is in no particular physical order) with block boundaries and transfer data accordingly.

Because the electronic disk firmware recognizes the mode of access and efficiently maximizes the disk's unique properties, it is not viewed as critical that a user have control or even be aware of file scattering.

Unlike scatter files on a flexible disk which can incur much overhead, especially if the blocks are widely scattered and force long seeks between blocks, electronic disk scatter files incur overhead but the penalty is not nearly so severe. Space overhead is one penalty and is typically less than one percent, but in rare cases may range as high as 50%. Time overhead is another penalty but only because of the requisite blocking of data; the seek penalty is nonexistent.

However, when adding to or creating a file in the electronic disk, the space allocation routines always search for a contiguous space large enough for the entire file, and make the file contiguous if possible. When a user expands a file through a SPACE command, or by simply writing off the allocated end of the file, if the file is contiguous, every attempt is made to preserve its contiguity. If it can be expanded in place, that is done. If not, but another area is large enough to hold the entire file, the whole file is copied into the new area. Only if contiguity cannot be maintained will a file be scattered.

Users concerned about the penalties being paid for scattering of files can avoid them by not depending upon the feature of dynamic expansion of files. If all files are always created to full length before being written, no use of scatter files will ever be required. Once a file is scattered on the electronic disk, it can't be made contiguous again.

To provide large gains in performance, array storage in the electronic disk departs from the standard of the flexible disk. Flexible disk arrays are placed in binary files element by element, with an item header byte in front of each array element. The electronic disk writes an array as a collection of bytes from system memory, with only one item header in front of the entire array. While this saves minimal data space on the disk, it provides data transfer rates up to ten times faster than if the standard had been maintained and up to 100 times faster than the flexible disk.

The penalty is that such an array must be read into an array of the same size; the user is unable to read only a portion of the array at a time. For those who have come to depend on this unsupported aspect of flexible disk usage, the electronic disk offers an alternate file type (specified at file creation) that provides for element-by-element storage of arrays.

### An Added Benefit

The electronic disk was developed in conjunction with the A-Series of improvements for the 4050 Desktop Computers, and, therefore, was optimized for use with the upgraded systems. A pair of microcode instructions quickly reads data from 4050 system memory and transfers it byte by byte to the electronic disk data register, and back again. This increases the raw transfer rate of bytes to and from the electronic disk by a factor of twenty, and provides about an eightfold speed increase when used with the 4050A Series.

### A Variety of Uses

One of the most useful aspects of the electronic disk is overlay management. For a typical performance comparison, consider an overlay that takes 32K of memory and is written in 670 lines of BASIC statements. This overlay takes 19 seconds to load from an ASCII file on the integral tape 4050 tape drive, 8 seconds from a binary file on tape, 6 seconds from a binary file on the flexible disk, and 2.3 seconds from a binary file on the electronic disk. This increases performance almost an order of magnitude, but even more important, it moves performance into the realm of interactivity.

An operator will object to responses of more than a couple of seconds, and in time will learn to restructure his approach to avoid overlays that are major roadblocks. Responses in the two to three second range won't cause any interruption of thought processes. Its overlay management coupled with its memory retention (battery backup) makes the electronic disk a powerful system for highly interactive, but extensive program tasks.

If the electronic disk performance in program and overlay storage is a quantum improvement, the gains in manipulating data files are truly revealing. While the maximum data throughput of the flexible disk is in the 3K byte-per-second range, the electronic disk can, under optimum conditions, move data at 400K bytes per second, and under less than ideal conditions even increase the rate of the flexible disk by an order of magnitude. In some cases this is due to design improvements not available to the flexible disk, such as microcode assistance of a direct data link as opposed to the funneling of all data through a GPIB. In other cases, such as in streamlining data arrays instead of transferring elements one by one, the improvements are due to new approaches. In still other cases, such as ignoring block structuring when transferring data to and from contiguous files, the physical characteristics of the device are recognized and used to yield the most efficient performance.

The utility of this new performance is self-evident. Consider a program that performs a sort upon a large disk file. The speed of the electronic disk makes it feasible to copy the file to it, perform the sort, and transfer the sorted data back to the flexible disk, the entire operation taking much less time than before. Or consider a program which operates upon local data while interacting with the operator - a mailing list is a typical example. With the electronic disk, local 4050 memory is fully available for the program; no portion of the data must be kept locally to increase speed. Further, all the editing can take place in the electronic disk at high speed, and only at the end of the session transfer the data to a permanent storage device. In fact, the electronic disk could even be that permanent storage device.