DISK

SUBSYSTEMS

SOFTWARE

CONSIDERATIONS

"Foreign trade" enhances the quality of life of any nation. In similar fashion, "foreign devices" attached to a computer's central processor unit (CPU) can often enhance the performance and cost effectiveness of the computer installation.

In both cases, protectionist attitudes can limit the potential benefits. In the world of computers, the first instinct of the user is to protect his system software. The concern is legitimate. Investment in software can easily exceed the value of computer hardware.

System software has been designed, in most cases, to operate within the context of a specific hardware configuration. Even minor alterations in the hardware characteristics can have far-reaching and often unpredictable effects on the operating software.

Yet the fact is that nearly every computer installation is limited not by the capabilities of its CPU, but by the throughput and capacity of the input/output and mass-storage devices attached to the CPU. The performance of nearly every computer installation

can be enhanced, therefore, by taking advantage of state-of-the-art advances in the design of these peripheral elements.

But this is acceptable only if the new equipment is "transparent" to the existing softward. Neither the CPU (for technical reasons) nor the user (for emotional reasons) should be disturbed by the switch to a new type of I/O or mass-storage device.

Software Transparency

These comments apply to any type of equipment attached to a computer--including terminals, printers, tape transports, data-communication lines, and disks.

This paper will concentrate, however, on disk drives--for several reasons. Disks represent the most ecconomical method for providing fast-access mass storage. Moreover, since disks are usually treated as an extension of main memory, they are intimately linked to the CPU and its operating softward. And if these facts were not enough to give pause, disk technology has been progressing at a very rapid rate during the past few years.

The challenge, then, is to realize the potential of the new technology (e.g., the new 3330 and Winchester drives) and still remain "software transparent." Techniques must be found to

attach an advanced (but "foreign") drive, without altering a single instruction in the operating system software or application programs.

Three different methods have been developed to accomplish this objective:

a) Fixed plug-compatibility

b) Dynamic plug-compatibility

c) Virtual transparency

As one of the industry's leading suppliers of disk systems for computer enhancement, CalComp uses all three techniques for achieving software transparency. Each method has its place -- depending on the type of system and the volatility of the hardware and software.

Fixed Plug-Compatibility

The original technique was fixed plug-compatibility--dating back to the 1960's. It was, in fact, the basis for the first effective penetration by independent disk suppliers, such as CalComp, into the monolithic IBM marketplace.

Small but significant changes in the mechanical design of disk drives provided major improvements in reliability and ease of maintenance--plus a modest increase in system throughput.

Despite these advantages, however, the new products could not have found a market unless they appeared (to the CPU) to be identical to equivalent products offered by IBM.

The "foreign" drives were provided with cables that could be plugged directly into the IBM mainframe. They also had electronic logic circuits that could respond, with absolute fidelity, to IBM's disk-drive commands.

But the fixed nature of the interface was a severe handicap. Independent disc suppliers were inhibited from developing performance characteristics beyond those that could be controlled by the IBM disk protocol. They would be, by definition, "transparent"--and of no practical value. Suppliers had to wait for IBM to make the improvements, and were therefore in a constant state of catch-up.

Of much greater concern was the fact that IBM could, at any time, make minor changes in its own operating software. These could render, overnight, all "foreign" disks inoperative. Sometimes the changes could be anticipated and allowances made in the control circuitry. Just as often, the only solution was an emergency retrofit of existing units in the field.

Despite these difficulties, the cost savings and performance benefits were sufficient to create a viable plug-compatible

market for independent disk suppliers. Their sales grew at an accelerating pace and soon extended beyond IBM to systems produced by other mainframe and minicomputer manufacturers. But the vulnerability to change has remained as a constant threat to both users and suppliers.

Dynamic Plug Compatibility

One solution to this problem is what can be referred to as dynamic plug-compatibility. This is the capability of "mapping" or making one type of disk look like another type which is recognized by the host system.

Two factors have contributed to the development of this concept. Of major importance, of course, has been the introduction of LSI and microprocessor circuitry. A microprocessor can be readily adapted to the type of control functions required in a disk interface. Equally important has been the expanding role of mass storage facilities as the principal method for enhancing the efficiency and throughput of computer installations. Disk storage facilities, taking advantage of new, high-capacity drives, have grown to the billion-byte level--as a starting point.

In addition to size, there is also a new emphasis on variety. Disk facilities may include drives with removable or non-removable media with fixed or moving heads. Small-capacity units may be added for private files, while maximum-capacity units serve as

basic storage to provide a minimum average cost per byte. The configuration of the facility may, in fact, change from month to month, or even hour to hour.

With the new emphasis on size and capacity, the additional cost of a dynamic, microprocessor-based disk interface can be easily justified. And as an added bonus, the interface can be easily altered to meet any changes in the disk hardware or operating system software.

The functions of a dynamic plug-compatible interface are dictated by the microprocessor program--stored in easily interchanged PROMs or floppy disks attached directly to the interface controller. Or the operating system itself can define its own plug-compatibility by downloading a suitable interface program at the time of system generation. Response to changes can be, for all practical purposes, instantaneous.

Virtual Transparency

Fixed plug-compatibility is still the most direct method for achieving software transparency in applications where the system software is static and little advantage can be gained by changing the mass storage facilities. Dynamic plug-compatibility can be justified when the storage facilities are large, or when there is a high degree of volatility in the system software and physical makeup of the storage facility.

In both of these cases, an assumption is made that the "plug-compatible" device is recognized by the existing system software. But this leaves a third situation in which neither technique is truly applicable. For example, the system software may have been written without any provision for the newer types of disk drives (e.g., 300 MB drives with a 1.2 MB/second transfer rate). There is, therefore, no "plug" to be compatible with. Situations could also exist in which the hardware and software are evolving at a rapid rate, yet the scope of the mass storage facility cannot justify the use of a dynamic, microprocessor-based "mapping" approach.

In both of these instances, a technique which can be referred to as "virtual transparency" can be an effective solution. CalComp is using this method to interface a variety of different capacity Trident disk drives to CPU's produced by many different minicomputer manufacturers, some offering many distinctly different operating systems. All of this is accomplished, moreover, with a microprocessor-based disk-controller design--produced in volume and thoroughly tested by hundreds of successful applications in the field.

A Logical Answer

Virtual transparency can best be understood in terms of its origin: Virtual memory. Originated by IBM and now adopted by

most minicomputer and mainframe operating systems, virtual memory was developed as a way to free programmers from any need to allocate and keep track of the computer's memory resources. Application programs could be written in abstract terms. The computer's operating system would translate "logical" virtual-memory addresses into physical addresses--taking advantage of any memory space available.

As programs grew in size and larger volumes of information were processed, much of the data (including application programs and portions of the operating system itself) were transferred to mass-storage devices like magnetic tapes and disks. But these were treated as I/O peripherals, and soon the computers were spending a majority of their time on the transfer of files between mass-storage and memory.

The virtual-memory concept again came to the rescue. Just as the use of logical addresses, independent of physical locations, simplified the life of the programmer, an extension of the virtual memory technique to mass storage devices served to relieve the main operating software from an equivalent concern for the physical configuration of the system. Subsidiary modules could accomplish the necessary mapping and address translations.

The benefits are manifold. The programmer and his application program can ask for data stored at a logical location. The

executive portion of the operating system passes the request along to the appropriate memory-management module. The subsidiary software determines whether the requested data is in main memory--immediately accessible to the application program--or is remotely stored on disk or tape. If the latter is the case, the transfer can be initiated by the lower-level software while the operating-system executive moves on to other, more demanding tasks.

Device Handler

The first step in the transfer operation would be for the memory-management module to pass the physical address along to an even more subsidiary software unit: the device handler for a specific disk-drive controller. Only now, three steps removed from the application program and two steps removed from the main body of the operating-system software, would the address request take a form that relates to a specific, physical device.

Virtual transparency takes advantage of the fact that there are, in truth, two interfaces between the disk storage and the system software. One is the physical interface at the plug connecting the disk-drive controller with the CPU hardware. The second is the software transition between the virtual addresses of the system software and the physical location of the stored data.

Either one of the interfaces can be used to maintain "software transparency." The plug-compatible techniques use the physical interface. CalComp's virtual-transparency method takes advantage of the modular structure of nearly all operating systems. When the memory-management module "calls" for a specific device-handler module, it can just as easily invoke a CalComp-supplied segment of software as one supplied by the computer manufacturer. In neither case is the main body of the system software affected. Not a single line of the existing application programming must be changed. Yet the user has the advantage of the latest technology disks, with capacities that far exceed those of the largest disks anticipated by the designers of the original software.

System Generation

The simplicity of the virtual-transparency concept is evident each time the operator "generates" the computer system for a specific group of application programs. The complete set of operating-system modules is rarely used. Instead, to conserve main-memory space, the operator invokes a system-generation program that allows him to specify only the modules required for the particular applications.

A CalComp-supplied module is included in his choice of options. The software itself is supplied on tape or disk, depending on the system configuration. The operator also has a sheet

of load instructions, written in exactly the same format as that used by the computer manufacturer. At an appropriate time in the system-generation procedure, the operator loads the CalComp device-handler module--or leaves it out. Moreover, if there is ever a problem with the CalComp disk hardware or software, the disk-controller cable can be simply unplugged and the system regenerated without the CalComp module.

Summary

The true test of "transparency" is whether a foreign device can be added to a computer system to enhance its performance and capabilities without affecting, in any way, the user's investment and confidence in his operating software and application programming.

At least three different methods can be used to achieve this result. CalComp has used all three techniques to enhance both existing and new computer installations.