DATE: September 24, 1981

TO: R & D Personnel

FROM: Randy Sybel

SUBJECT: Proposal for New Software Disk Interface

REFERENCE: None

KEYWORDS: Disks, Software

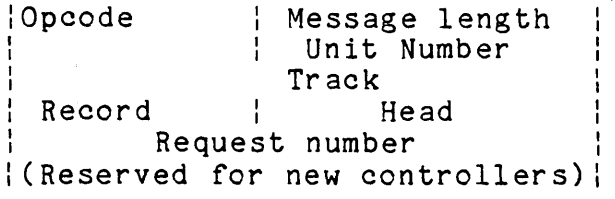
ABSTRACT

This paper is a proposal for a new software disk interface. This proposal is a first step in moving some functionality from Primos to the disk controller.

This paper is a proposal for a new software disk interface. This proposal is a first step in moving some functionality from Primos to the disk controller.

The present implementation in Primos involves 3 channel programs. These are a Pre-seek channel program, a Stall loop, and the Read/Write channel program. Also, Primos orders the requests so as to achieve seek optimization.

The new interface allows much of the low level driver functionality to be moved to the controller. The structure for a Read/Write request is now much simpler and takes up far less space in main memory. It is structured as follows:



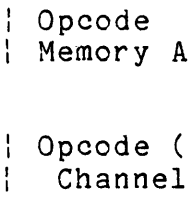
This structure is used for reads, writes, formats, write verifies, and the extended formatting commands. (Several of these are new commands implemented only in the Rabbit disk controller and all future controllers)

The way the new requesting scheme would work is as follows: The disk driver would send the controller an OTA `17 (as is presently done), with a pointer to the above data block. The controller would fetch all words immediately and then go Ready again to accept more requests. As many as 64 requests per disk unit could be made. (More in future disk controllers). The controller orders these requests in a seek optimization scheme (the same as the one Primos uses) for each drive and starts all possible seeks.

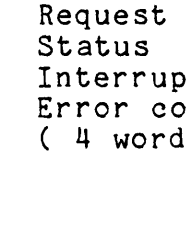
When any seek is completed, the controller will store 2 words in CPU memory and then interrupt. One of the stored words is an interrupt type, and the other is the Request number from the command block. The software must use this number to wire the user`s buffer and I/O window. The disk controller will not transfer data until it receives a command stating which DMA channel and chain to use. The controller would then transfer the data and interrupt on completion, having stored at least 2 more words into CPU memory. These would be an interrupt type (completion or error) and status.

The controller would then interrupt again with another request (if any) as soon as it was ready for data transfer. This would continue until the internal queue was empty. Requests for the disk controller can be sent at any time except during the actual data transfer, when the controller would be Busy.

Several other commands are needed by the controller. The DMT setup and interrupt setup are run only once at initialization time, to give the controller addresses for DMT transfers. The structure of these is as follows:



The second command is the previously mentioned DMA command. This tells the controller which DMA channel to use to transfer data, and how many to chain together. The first command is used for both Interrupts and DMT. For the interrupt, the appropriate opcode is used, and the Memory Address is the Vector for the interrupt. For the DMT command, using the appropriate opcode, the Memory Address is the beginning of a data block, structured as such:



Presently, this interface is being tested on the Rabbit disk controller. The initial controller used the standard disk interface, and performance levels dropped. This interface is an attempt to restore or exceed previous disk performance, and also to take a long needed step to off-load Primos.

Initial tests, done unfortunately with test programs and not a modified Primos version, have shown about a 30 percent improvement in the disk access rate, in a random access test. The resulting performance is about equal to a standard SMD drive. This does not realistically reflect an improvement under Primos, but it also does not show the reduction in CPU time spent in the disk driver. When results of testing under Primos are available, more realistic performance levels can be stated.

This interface is by no means complete. It has been left as flexible as possible for future expansion, but it also is not beyond changes in its present form. It is hoped that the circulation of this paper will produce responses that may help us to develop a better interface.

It should be noted that this is not an "intelligent disk interface". This is a first step towards that. This can be

implemented without a large amount of design and software effort, and can also be used as a learning experiment for how to design an intelligent interface.

Comments on this interface, proposals, critiques, and questions should be sent to Randy Sybel, x 3022, x.mail SYBEL.