TO:	C. T. Clingen F. J. Corbató J. W. Gintell N. I. Morris J. H. Saltzer T. P. Skinner V. L. Voydock S. H. Webber
FROM:	R. J. Feiertag
DATE:	May 21, 1971
SUBJECT:	I/O Workspace Synchronization

Some recent applications on Multics have indicated a need for asynchronous I/O operations. The attempt to implement such asynchronous I/O has brought to light some problems that indicate that the current specification of the I/O system with respect to workspace synchronization requires further scrutiny.

Before indicating the nature of the problem, I will briefly review workspace asynchrony in the Multics I/O system. A device being workspace synchronous implies that if a read or write call returns, the transaction has been logically completed, i.e. the I/O system is finished with the given workspace. A device being workspace asynchronous implies that if a read or write call returns the transaction may not be logically completed, i.e. the workspace is still in use by the I/O system and may be written from or read into subsequently.

Problems arise with the current specification of the I/O system because workspace synchronization is a property of the device. However, workspace synchronization mode implies something about the scope of the workspace, which is a property of the calling program. Programs that are written for devices in workspace synchronous mode will not necessarily work properly if the device is in asynchronous mode. Since most system programs that do I/O are written assuming synchronous mode, they may not be used with devices in asynchronous mode.

There is no reason why workspace synchronization should be an attribute of the device. In fact, workspace synchronization mode is totally unnecessary. Let us assume that devices are always in workspace asynchronous mode. Then a workspace synchronous read or write call can be simulated by a read or write call immediately followed by an iowait call. Using this technique means that whether or not the workspace is to be used synchronously or asynchronously is controlled by the calling program, the entity that also controls the scope of the workspace.

Because of upward compatibility problems it is impossible to simply eliminate workspace synchronous mode. I, therefore, propose that the current read and write calls retain their synchronous qualities, i.e. be equivalent to a read or write call followed by an iowait call in asynchronous mode, and that a new read and a new write call be established that act as read and write calls in asynchronous mode. The worksync call would be eliminated.

call read (~~w~~________ ______ ____ , status)

(include transaction id.)

call ios_$status (stream , status) , evchn)

channel on which change of status will be signalled

returns status of entire stream.

read ahead

write behind

# of outstanding transactions

call ios_$ transaction _status (trans_id , new status)