To: Distribution

From: N. I. Morris

Date: March 27, 1974

Subject: I/O Interfacer Design Specifications

Design Details

The I/O Interfacer has been designed mostly as described in MTB-028. There have been several additions, modifications, and limitations which have come about from the design reviews of MTB-028. These are discussed below:

Hardcore Use of I/O Interfacer

DIM's which connect many processes to a single I/O data path (e.g. the IMP DIM) may wish to reside in the hardcore ring, yet still make use of the features provided by the I/O Interfacer. Special hardcore attachment and allocation entries into the I/O Interfacer will be provided for this purpose. The wired workspace buffer segment must be provided by the caller in this case. The buffer may be of any length, but must start on a 0 mod 512 word boundary and be a multiple of 512 words in length. It may not cross a 256K word boundary. IOM interrupts will not cause a wakeup to be sent over an event channel. Instead, an interrupt-time call to the DIM's interrupt procedure will be made. This will allow fast and efficient processing of interrupts.

Special Connect

T & D programs will wish to be able to connect to the IOM with their own PCW, not one which has been manufactured by the I/O Interfacer. A connect entry will be provided for use by T & D programs. A user-supplied PCW will be used for connecting a channel when this entry is called. The I/O Interfacer will validate certain PCW fields and will reject special connect calls which might result in channel operation which is harmful to system integrity.

Additional Status Information

Certain T & D programs wish to examine more detailed status than required by the average DIM. In particular, they wish to look at the LPW and LPW extension left in the IOM channel mailbox at the completion of an I/O operation. This information will be provided for use by such T & D programs. In addition to the 72-bit hardware status returned by the status entry to the I/O interfacer, the LPW and LPWX will also be returned.

Status Queuing

Status stored by the IOM for devices attached through the I/O Interfacer will be delivered to the I/O Interfacer at interrupt time by the iom_manager. The I/O Interfacer will immediately send status to the interrupt procedure for a device attached via the hardcore attachment entry. IOM status for non-hardcore users of the I/O Interfacer will be placed in a queue and an ipc_ wakeup will be sent to the user. Calls to the status entry of the I/O Interfacer will return IOM status words in the order they were placed in the queue. Information will also be provided on each status call to enable the user to determine if the status queue was empty.

Marker Status Considerations

A user's DCW list may contain more than one Instruction DCW, and each IDCW can generate marker status at the completion of that particular instruction. Although the I/O Interfacer will queue status for a device, the iom_manager currently will not. Therefore, the iom_manager must be able to receive and process a marker interrupt before the next marker or terminate status event occurs for that device. If status events occur too rapidly, status will be overlaid by the IOM and will thus be lost. No estimate can be provided for a "safe" time interval between status events. Therefore, marker status should not be used in cases where the user relies on the information contained in each individual status word (e.g. to compute the length of each tape record when reading a tape containing records of unknown length). Marker status may be used in cases where the user wishes to determine when the IOM passes a particular point in his DCW list. Eventually, the iom_manager will be modified to allow the IOM to queue status. At that time, the danger of losing status will disappear.

Page Control Modifications

A new bit will be added to the active segment table entry (ASTE) for a segment. If this bit is on in a segment's ASTE, page control will be forced to place the pages of the segment in abs-usable memory.* Pages of the segment will be subject to paging as usual; they will be guaranteed to always reside in abs-usable core when paged in. Use of this bit by the I/O Interfacer will avoid the problems associated with the use of I/O buffers in core which can be dynamically removed from the Multics configuration.

Workspace Buffer Allocation

When the I/O Interfacer creates a buffer segment, it will turn on the entry hold switch in the segment's ASTE to prevent deactivation. Then, it will turn on the abs-usable core bit. When the I/O Interfacer is called to connect a channel, it will wire the buffer segment before performing the connect. When the channel terminates, the buffer segment will be unwired. Note that this workspace buffer is used to contain both DCW lists and data. It is the user's responsibility to properly manage the contents of the buffer.

Workspace Buffer Size Limitation

The initial version of the I/O Interfacer will support a maximum buffer size of 1024 words. Later versions will allow larger buffers. (Hardcore users of the I/O Interfacer may supply a larger buffer.) This means that the initial I/O Interfacer will be incapable of supporting Multics Standard Tapes except from a hardcore interface to the I/O Interfacer. The reason for the size limitation is to avoid consideration at this time of the problems of insuring that a multi-paged buffer resides in contiguous core. As described above, the buffer segment will be unwired except when I/O is taking place.

---

* abs-usable memory consists of memory that is used for mailboxes and I/O buffers. Hence, it is not subject to dynamic reconfiguration. It is usually that memory which is connected to the low-order system controller in the Multics configuration.

I/O Interfacer Calling Sequences

Entry: ioi_$assign

This entry is called to make a device known to the I/O Interfacer. iom_manager$iom_assign is called to make the device known to the iom_manager and to assign a device index. This device index is returned to the caller. It will identify the device on all subsequent calls to the I/O Interfacer. The supplied event ID is used to perform an ipc_ wakeup when status is received for the device.

Usage

```
declare ioi_$assign entry (fixed bin(12), char(*),
          fixed bin(71), fixed bin(35));

call ioi_$assign (devx, devname, eventid, rcode);
```

devx — is the assigned device index. (Output)

devname — is the name of the device being assigned. (Input)

eventid — is an ipc_ event channel ID to be used in waking the process upon I/O completion. (Input)

rcode — is a standard error code. (Output)

Entry: ioi_$hardcore_assign

This entry is equivalent to ioi_$assign, except that it can be called only by hardcore procedures. I/O completion interrupts will be passed on to an interrupt procedure instead of generating an ipc_ wakeup.

Usage

```
declare ioi_$hardcore_assign entry (fixed bin(12),
          char(*), ptr, fixed bin(35));

call ioi_$hardcore_assign (devx, devname, intp, rcode);
```

intp — is a pointer to the procedure to be called at interrupt time. The calling sequence to the interrupt procedure is described below under "Interrupt Procedure Call". (Input)

Entry: ioi_$unassign

Tnis entry is called when a user is finished with a device. Any pending I/O is stopped and all buffer space is released. The iom_manager is called to unassign the device.

Usage

```
declare ioi_$unassign entry (fixed bin(12), fixed
        bin(35));

call ioi_$unassign (devx, rcode);
```

devx is the previously-assigned device index. (Input)

Entry: ioi_$workspace

Workspace for DCW lists and data is provided as a result of this call. Initially, a limitation of 1024 words will be placed on the size of the workspace buffer.

Usage

```
declare ioi_$workspace entry (fixed bin(12), ptr, fixed
        bin(18), fixed bin(35));

call ioi_$workspace (devx, segp, seglen, rcode);
```

segp is a pointer to the buffer segment to be used to contain DCW lists and data. (Output)

seglen is the desired length of the buffer segment in words. (Input)

Entry: ioi_$hardcore_workspace

This entry can be called only by hardcore ring procedures. It is used to identify a wired-down buffer space to the I/O Interfacer. The segment must be supplied by the caller and begin on a 0 mod 512 word boundary. It must be a multiple of 512 word in length and must not span a 256K word boundary.

Usage

```
declare ioi_$hardcore_workspace entry (fixed bin(12),
        ptr, fixed bin(18), fixed bin(35));

call ioi_$hardcore_workspace (devx, segp, seglen,
        rcode);
```

segp is a pointer to the buffer segment. (Input)

Entry: ioi_$set_timeout

This entry establishes a time-out interval for a device. The clock is read each time the device is connected. If a terminate does not occur within the time-out interval, termination will be forced and the user notified. The set_timeout entry may be called as many times as desired for a device. If it is never called, a default interval of 30 seconds will be used.

Usage

```
declare ioi_$set_timeout entry (fixed bin(12), fixed
        bin(52), fixed bin(35));

call ioi_$set_timeout (devx, time, rcode);
```

time — is the time (in microseconds) to be given to allow a device to terminate after a connect. (Input)

Entry: ioi_$connect

The I/O Interfacer will construct a PCW in the hardcore ring to be used in connecting the channel. The PCW will contain a reset status command, and the continue bit will be set. This PCW will be essentially ignored when connecting PSIA channels, and it will act as a null operation when connecting common peripheral channels. The buffer segment will be wired-down (unless the device was attached through the hardcore_assign entry) and a call will be made to the iom_manager to connect the channel in relative mode at the place specified by the supplied offset.

Usage

```
declare ioi_$connect entry (fixed bin(12), fixed
        bin(18), fixed bin(35));

call ioi_$connect (devx, offset, rcode);
```

offset — is the offset into the buffer segment where the channel is to be connected to a DCW list. (Input)

Entry: ioi_$connect_pcw

This entry is identical to the connect entry except that word A of the PCW is supplied by the caller. It is copied into the hardcore ring and used to connect the channel. No modifications are made to the PCW fields, but the call will be rejected if invalid information is found in the PCW. When the buffer segment is wired, the absolute address of the base of the segment will be returned to the caller. This entry is intended

specifically for use by T & D routines. It can also be used by channels which require a PCW to be supplied (e.g. the Console Channel Adapter).

Usage

```
declare ioi_$connect_pcw entry (fixed bin(12), fixed
          bin(18), bit(36), fixed bin(24), fixed
          bin(35));

call ioi_$connect_pcw (devx, offset, pcwa, absaddr,
          rcode);
```

pcwa — is word A of the PCW to be used in connecting the channel. (Input)

absaddr — is the absolute address of the base of the buffer segment. (Output)

Entry: ioi_$get_status

This entry is called to pick up status from the status queue for a device.

Usage

```
declare ioi_$get_status entry (fixed bin(12), ptr,
          fixed bin(35));

call ioi_$get_status (devx, sp, rcode);

declare 1 ioi_stat based (sp) aligned,
          2 completion fixed bin(17),
          2 level fixed bin(3),
          2 offset fixed bin(18),
          2 padding fixed bin,
          2 iom_stat bit(72),
          2 lpw bit(72);
```

sp — is a pointer to the ioi_stat structure. (Input)

completion — is a code indicating the state of the channel and whether or not status has been returned from the status queue. Possible values are as follows:
0 = No status in queue; channel not in operation.
1 = No status in queue; channel in operation.
2 = Status returned.
3 = Time-out occurred.

level — is the interrupt level which caused the IOM status to be stored.

1 = system fault status.
3 = terminate status.
5 = marker status.
7 = special status.

offset — is the offset into the buffer segment of the last DCW processed. If completion is equal to 1, offset will be the location of the DCW currently being processed.

iom_stat — is the IOM hardware status.

lpw — is the LPW and LPWX from the channel's mailbox.

Interrupt Procedure Call

The interrupt procedure which is called using the intp supplied in the hardcore_assign entry is described below:

```
declare interrupt_procedure entry (fixed bin(12), ptr);

call interrupt_procedure (devx, sp);
```

sp — is a pointer to the ioi_stat structure described above. (Input)

(END)