TO: MSPM Distribution
FROM: D. L. Stone
SUBJECT: BF.20.01
DATE: 06/21/68

BF.20.01 describes the interface to the new GIM. It is considerably more primitive than the previous interface and will allow a more efficient implementation of the GIM. The new implementation is now being done by Tom Skinner (MIT x6017) and David Stone (GE x263).

The attached re-issue of BF.20.01 incorporates the addendum of 02/09/68 (hence superseding BF.20.01A), and adds a new call, giminit$safety.

Published: 06/21/68
(Supersedes: BF.20.01, 01/09/68;
BF.20.01, 12/01/67;
BF.20.01, 07/11/67;
BF.20.01, 05/10/67;
BF.20.01, 12/15/66)

Identification

DCM/GIM Interface Specifications
S. D. Dunten and D. L. Stone.

Purpose

This document describes the user interface to the hardcore modules which control Input/output through the GIOC in Multics. It is a functional specification in symbolic terms; for details of the strategy employed and the data bases used, see sections BF.20.02 and BF.20.03, respectively.

Introduction to the GIOC

In order to make use of the GIM interface, the DCM writer must be on intimate terms with the GIOC adapter through which his device is controlled. The GIM is only minimally aware of the relationship which is maintained. Although system security must be assured, the misuse of a particular channel is of no concern to the GIM. Hence, the DCM is allowed almost complete freedom to control its device as it sees fit. In the following discussion, it is assumed that the reader is familiar with Multics document G0050 - the Programmers's reference manual for the GIOC.

The GIM restricts its callers to using a single, contiguous DCW list. Within that list, any of the six DCW types recognized by the GIOC may be used. By judicious use of the GIOC transfer DCW, the DCM may utilize the single list assigned it as if it were several lists; however, it is anticipated that most DCM's will operate their list as a single circular queue of DCW's. The DCW's themselves are specified by the DCM except that absolute addresses in data DCW's are allowed only in the case of privileged users (e.g. -Disc DIM). No user may specify the tally or address fields of a transfer DCW. The address field supplied by the user is interpreted by the GIM as an offset within the DCW list (the first DCW having an offset of 1). The DCW's specified are written directly into the DCW list used by the GIOC.

Once constructed, DCW lists may be activated by means of a single Connect Instruction Word specified by the user. The activity of a list is not monitored by the GIM and consequently, all list activities are performed in the same way by the GIM whether the list is active or not.

Gaining Access to a Channel

The GIM grants access rights to a channel when one of the following two calls is issued by an appropriate process (e.g. a process in a Universal Device Manager Process Group, from ring 1):

```
Call giminit$assign(name,devx,event,type,rcode);
```

Where the arguments are accessed by the GIM through the EPL declaration:

```
dcl  name char(32),          /*symbolic channel name*/
     devx fixed bin(17),     /*device index returned by the
                               GIM for this channel*/
     event bit(70),          /*passed on to the DSTM for device
                               signalling*/
     type char(*),           /*passed to I/O assignment
                               module*/
     rcode fixed bin(17);    /*error return,0=ok*/
```

The assign call, if valid, causes the GIM to initialize certain information pertaining to the caller and his channel. The physical channel and related information is discovered in the Device Configuration Table by locating the symbolic channel name supplied. A device index is constructed and returned to the caller for use in all future calls pertaining to this assignment of this channel. The event id supplied is passed to the Device Signal Table Maintainer along with the device index; it is the device index which is used by the GIM interrupt handler to wake up the caller when interrupts occur on his channel. After this call has been successfully performed, the caller may access and use the assigned channel.

The Channel Copy Table is allocated at this time (see BF.20.03).

An additional use of this call is made by the IØ system when an iopath is pushed down. If the assign call is received for a channel which has been previously assigned, the GIM will simply return the device index.

```
call   giminit$fsassign(name,devx,rcode);
```

This version of the assign call is allowed only to privileged users (ring 0). It causes the GIM to accept absolute addresses in data DCW's and to omit the allocation of a Channel Copy Table for this channel. This call is intended for use by the file system at system initialization.

Defining a List

Before using a DCW-list, the DCM must inform the GIM of the size of the list because the GIOC requires that the list be in contiguous wired down core. For this purpose, the following call is used:

```
call      giminit$list_size(devx,listsize,rcode);

dcl       listsize fixed bin(12);
```

The GIM first looks to see whether a listsize has been previously assigned. If it has, and if that size is different from listsize, then the GIM will reallocate the hardware buffer and reinitialize it. If the sizes are identical, no action is taken. If listsize is zero, then the wired down buffers for the DCW list and associated data buffers are released.

If no list currently exists, the GIM allocates a new one which is one DCW larger than that specified by the DCM. Every DCW slot is filled in with a transfer to a system-wide "safety" pair of DCW's which are guaranteed to stop any channel. This apparatus protects the system from faulty DCW lists which might cause the GIOC to run off of the list.

It is expected that the low speed common peripheral adapter users will make only one listsize call per assignment, whereas communication adapter users may make two calls -- one for the DCW list required to wait for dial-ins, and one for the IØ associated with a dialed-in channel.

Changing a DCW list

Having gained access to a channel and defined a DCW-list length, the DCM may add DCW's to the list by making the following call:

```
call gim$list_change(devx,dcwp,datap,listx,count,rcode);

dcl  (dcwp,                  /*pointer to "dcw_array"*/
      datap) ptr,            /*pointer to "data_array"*/
      (listx,                /*index within DCW_list,(start-
                               ing with 1) of first DCW to
                               be changed*/
       count)fixed bin(17);  /*number of elements in
                               "dcw_array" which are to be
                               used*/
```

where the data referred to is accessed as:

```
dcl  dcw_array(count)bit(72)based(dcwp),
                          /*a list of real DCW's*/
1 data_array(count)based(datap),
     2 p ptr,             /*pointer to user workspace if nth
                            DCW is data DCW*/
     2 rw bit(2);         /*"10"b = readDDCW
                            "01"b = writeDDCW*/
```

(Note: for privileged users, datap should be null).

In response to this call, the GIM performs the following actions for each DCW, beginning at the countth one and working backwards: (the current DCW index is referred to as "j").

1. If the jth DCW transfers data then:

   a) if the number of words required is the same as the old jth DCW, insert that address in a copy of the new DCW.

   b) if the number is not the same, then allocate a new buffer and fill in the address accordingly.

   c) if the new DCW is a write transfer, then copy the data from the user's area into the buffer.

2. If the new DCW is a transfer DCW, then in a copy of it adjust the tally field and translate the address field from an index within the DCW list to an absolute address.

3. In the copy of the new DCW, set all non-zero status pointers to the status channel being used by the GIM. Move the copy of the new DCW into the list at the jth place.

4. Sample the address of the DCW in the mailbox to see if the old DCW is now being processed by the GIOC. If so, and if the old DCW is a data transfer, then shut down the channel unless the data buffer is being reused by step 1a above. Give an error return.

5. Free the data area associated with the old DCW, if any, unless it is to be reused.

The above algorithm allows the clever DCM to minimize the number of reallocations which must be done for its data DCW's. By making its data buffers a constant size when practical, the DCM can cause reuse of the hardware buffer associated with each data DCW. On output, this strategy would require the use of a false tally field

and would therefore be practiced only on devices which can terminate from a character in the transmitted data (PRT202); on input, it can be applied more easily.

Note that for privileged users (i.e., those who have made the fsassign call), the absolute address in the DCW is used and the datap pointer is not used.

Activating a DCW LIST

In order to perform certain tasks -- notably to activate a DCW-list, the DCM must provide a Connect Instruction Word (CIW) to be processed by a GIOC connect channel. The following call allows the DCM to connect one CIW.

```
call gim$list_connect(devx,ciw,listx,rcode);

dcl  ciw fixed bin(18);  /*the CIW is treated as an
                           18 bit string which is the
                           right half of a CIW*/
```

The GIM actions are:

1. if listx = 0 then go to 3

2. if a DCW list has not been allocated, there is an error.

3. create the proper CIW and connect it.

Stopping a DCW List

A running (indirect) channel may be stopped in mid flight or an inactive channel safetied by the following call:

```
call giminit$safety(devx,rcode);

dcl  (devx,rcode) fixed bin(17);
```

devx is the device index of the channel and rcode is the standard GIM error return.

Upon receipt of this call the GIM will jam the safety DCW in the channel mailbox and the address of this DCW in the LPW mailbox. The safety DCW is a command DCW (type 4) with bit seventeen on. All four status channel pointers will be set so the user will be able to detect all channel activity from this point on, e.g., the special interrupt caused by a tape going into ready status.

Status

The GIM provides two types of status information for the caller: where is the GIOC in the DCW list and what hardware status words have been stored for the GIOC processing of the list. These two types of information are used at different points in the driving of a device. If the DCM is attempting to keep its DCW-list active, a typical strategy would be to inquire about the GIOC's current position in the list, change the list accordingly and only then request that hardware status be relayed. This strategy delays time consuming processing of status words until after the list has been patched, thus increasing the probability that the list will remain active. If, upon examination of the status words, the DCM discovers that the channel has terminated, then it can issue a connect; otherwise, it need simply update its tables and return the status to its caller.

```
call gim$get_cur_status(devx,listx,dcwt,rcode);

dcl  (listx,                /*current index of dcw being
                              processed*/
dcwt)fixed bin(12);         /*current dcw tally*/
```

Upon receipt of this call, the GIM samples the mailbox for the channel to determine the index and tally. Note that for direct channels, the dcwt will in general be wrong. Next, the GIM copies any read data which has not been copied into the caller's work space up to and including the element described by the dcwt (except for direct channels). The word in which the last (dcwtth) element is contained will be transmitted as a word, so the remainder of that word may not be input data. For direct channels, the DCW on which the GIOC is currently working will not cause any copying to be done.

```
call gim$get_status(devx,status_array_ptr,array_size,
     outsize,waiting,rcode);

dcl  status_array_ptr ptr,  /*pointer to "status_array"*/
     (arraysize,            /*maximum number of status
                              elements to be returned*/
     outsize)fixed bin(17), /*actual number returned*/
     waiting fixed bin(17);

dcl  1 status_array(array_size)based(status_array_ptr),
     2 status,
          3 type bit(4),      /*bits 0-3 of (modB) status
                                word*/
          3 int_sig bit(2),   /*bits 4-5*/
          3 adapter bit(12),  /* bits 18-29*/
```

```
        2 time bit(52),          /*time at which stored
                                   interrupt was process d by
                                   handler*/
        2 listx fixed bin(12),   /*which DCW caused status
                                   store?*/
        2 dcwt fixed bin(12);    /*dcw tally at interrupt, if
                                   applicable*/
```

The GIM returns two types of information from the get_status entry - both in the array pointed to by "status_array_ptr". The first element of the array has zero status and time; the "listx" and "dcwt" entries are set to the current value as determined from the mailbox of the channel (a call to get_cur_status is made to determine this information). The remaining entries in the array are filled from as many status words as are currently in the hardware status queue for the channel. "outsize" is set to the number of status words put into the array from the status queue. If there are no status words waiting, the outsize is set to zero. No more than "array_size" -1 status words are entered into the array. If more status words are waiting, then "waiting" is to be set to the number waiting. If none are waiting then "waiting" is set to zero.

Releasing Lists and Channels

The DCM can take advantage of lulls in activity on its device by releasing its wired down buffers during such periods. After releasing, the DCM must make a list size call in order to have a wired down buffer allocated.

```
giminit$list_size(devx,0,rcode);
```

Finally, when the DCM wishes to relinquish the channel and have all information associated with the assignment freed, it issues:

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

Strategy for using the new GIM interface

1. Low Speed Common Peripheral Adapter

   Terminates are a single IDCW. Consider the list as circular with a transfer from the last DCW to the first. At any given time there should be exactly one terminate in the list. Upon gaining control (assuming that previous list_changes and connects have been made) call gim$get_cur_status. This call is quite fast for write DCW's, (causes copying for read DCW's), and will uncover how many free dcw's there are in the list after the

terminate. Call gim$list_change to fill in the empty slots with queued data adding a new terminate and overwriting the old one. Finally call get_status to determine whether a call to gim$list_connect is necessary (if the channel has terminated). This call may take some time to execute since status is moved, and it should therefore be postponed until after the active patching is performed, in an attempt to keep the channel active.

2. Communications Adapter

   Terminates are sequences of n > 1 dcw's. Keep all such sequences beyond the extent of the circular list described above and use a transfer to the sequence as the "logical terminate" to be overwritten in the strategy above.

3. As indicated in the list_change description, keep data buffers a constant size.

Restrictions Imposed by the GIM

1. Patches of active DCW lists will cause the channel to be terminated by the GIM if data dcw's are patched while the GIOC is processing them (i.e., they are in the mailbox) and the length of the wired down buffer is changed.

2. A corollary to (1) suggests that active (or possibly active) patching should be restricted to a single dcw of type "instruction" or "transfer" - hence terminates should be caused by a single IDCW with the terminate bit on or by a transfer to a longer sequence (for communication channels).

3. Data read in by the GIOC is not copied into the user's area until he has made a status call to the GIM. The user is not guaranteed that more data than is specified by the LPW-DCW tallies is available in his workspace.

4. Data areas specified by the user as the recipients of read data should begin and end on a word boundary. Since the GIM copies data by word, bits to the left of the first bit specified by the DDCW and bits to the right of the last bit will be overwritten by the GIM. This restriction will be particularly useful when the GIM is recoded in EPLBSA.

5. Adapter-wide and GIOC-wide errors are <u>not</u> recognized by the GIM.

6. Status not called for will eventually be discarded.

7. Status channel pointers supplied by user are interpreted as zero or non-zero -- particular channel is chosen by GIM.