phcs_ : <u>Undocumented Entries</u>

get_cur_status
list_change
list_size

} all are gim entries

phcs_$assign
$list_size
$safety
$unassign

Name: gim_assign

This procedure handles the assignment of devices and wired-down storage within the GIOC Interface Module (GIM). The use of the GIM is described in the SWS under the heading of "GIOC Calls". The correspondence between the user callable entry points and the entry points in this procedure is shown below:

User Entry Points	Entry Point in gim_assign
hcs_$assign	gim_assign$assign
hcs_$list_size	gim_assign$list_size
hcs_$safety	gim_assign$safety
hcs_$unassign	gim_assign$unassign

phcs_$get_disk_meters

Name: get_disk_meters

This procedure is responsible for obtaining the data collected by meter_disk in the segment disk_traffic_data, and copying it out into the user ring. It is also responsible for wiring and unwiring the disk_traffic_data segment, and controlling the use of the same as a resource.

As disk traffic may occur during a call to phcs_$ring_0_peek, that procedure is not adequate to observe the data collected by meter_disk. As this data does not represent statistics, but a continuous trace, consistency is important, and this special procedure, called through phcs_$get_disk_meters, must be used.

Since consistent inspection of disk_traffic_data requires that the page tables be locked, the data must be copied out to a wired data base. The segment temp_copyseg1 is used for this purpose, and remains wired only during the call to get_disk_meters.

As the segment temp_copyseg1 is a nonshareable resource, it is protected by a lock. The wait event associated with this lock is the ASCII bit pattern of "dtm_".

Entry: get_disk_meters

This entry returns a consistent copy of disk_traffic_data to the caller. If disk_traffic_data was not wired, zeroes will be returned.

Usage

```
     declare get_disk_meters entry (ptr);

     call get_disk_meters (image_ptr);
```

1) image_ptr   is a pointer to a 1024-word array where the copy of disk_traffic_data should be placed. (Input)

This call can be made by calling phcs_$get_disk_meters, with the same parameter, in other rings.

phcs-$get_status

Internal Interface
Hardcore Ring
02/03/71

Entry: glm3$get_status

This is the status handling entry point for the glm.

Usage

```
declare glm3$get_status entry (fixed bin(12), ptr,
        fixed bin(6), fixed bin(6), fixed bin(1),
        fixed bin);

call glm3$get_status (devx, sap, as, os, w, rcode);
```

1) devx — is the device index of the device for which status is desired. (Input)

2) sap — is a pointer to the array in which status will be returned. This array is declared as follows:

```
declare 1 sa (0:as) based(sap) aligned,
          2 status bit(18),
          2 time bit(52),
          2 listx fixed bin(12),
          2 dcwt fixed bin(12);
```

1) status — is bits 0-5 and 18-29 of the first word of the GIOC status.

2) time — is the time that the status was generated (not currently implemented).

3) listx — is the Data Control Word (DCW) list index of the DCW causing the status to be stored.

4) dcwt — is the DCW tally residue.

Note: — sa (0) contains only the current status (i.e., the current DCW list index).

3) as — is the upper bound of the status array. (Input)

4) os — is the array subscript of the last status array element filled in by glm3$get_status. (Output)

5) w — is non-zero if more status is waiting for this device than could be stored in the status array. (Output)

6) rcode — is an error code. (Output)

phcs_$ initiate

Same calling sequence as hcs_$initiate

This entry sets the calling process's validation level to zero and then calls initiate.

It is used to initiate directories and other data bases which cannot normally be initiated in the user ring.

phcs_$initiate_count.

Same calling sequence as hcs_$initiate_count.

See comments under phcs_$initiate

phcs_$ list_size

See phcs_$ assign

phcs_$ring_0_message

Entry: ring_0_peek$message

This entry will print a message on the 645 operator's console.

Usage

```
     call ring_0_peek$message (string);
```

or

```
     call phcs_$ring_0_message (string);
```

1) string(character(*))	is a message to be printed on the operator's console. (Input)

phcs_$ring_0_peek

Entry: ring_0_peek$ring_0_peek

This entry will move data from any location readable in ring 0 to any location writeable at the user's validation level.

Usage

```
call ring_0_peek (p1, p2, n);
```

or

```
call phcs_$ring_0_peek (p1, p2, n);
```

1) p1(pointer)	is a pointer to first word of data to be read. (Input)
2) p2(pointer)	is a pointer to the first word of the region the data is to be moved to. (Input)
3) n(fixed binary(17))	is the number of contiguous words to be moved (0<n<1024). (Input)

phcs_$safety

See phcs_$assign

phcs_$tdcm_attach

Same as hcs_$tdcm_attach

Not called

phcs_$tdcm_detach

Same as hcs_$tdcm_detach

Not called

phcs_$tdcm_iocall

- Same as hcs_$tdcm_iocall

Not called

phcs_$tdcm_reset_signal

Same as hcs_$tdcm_reset_signal

Not called

phcs_$tdcm_set_signal

Same as hcs_$tdcm_set_signal

Not called

phcs_$unassign

See phcs_$assign