Section BF.20.04 is updated to correspond to the new GIM.
Identification
GIM INNER CALLS
T. P. Skinner, D. L. Stone

Purpose
This section lists all of the procedures which constitute the GIM. For each entry point of each procedure a list of information is given:

calling sequence and argument declarations,
purpose
procedures and data bases referenced and
procedures referencing.

Data bases are distinguished by enclosing parentheses.

GIM Procedures

ABSADR:

\begin{verbatim}
fb24 = absadr(p, errtn);
dcl p ptr, errtn label, fb24 fixed bin (24),
    absadr ext entry fixed bin (24);
\end{verbatim}

Returns the absolute address of the word to which the pointer points.

references: (descriptor segment and page tables)
called by: allo$dcw

ALLO:

\begin{verbatim}
call allo$dcw (n, p, add);
dcl n fixed bin (17), p ptr, add bit (24);
\end{verbatim}

Allocates storage in "gim_abs_seg" and returns a pointer and the absolute address.

references: absadr, ilock$looplock, ilock$loopunlock,
    switch_stack,
    (debug_control, gim_abs_seg)
called by: giminit$1ist_size, gim$1ist_change
call allo$free_dcw (n, offset);
dcl n fixed bin (17), offset bit (18);

Frees storage in "gim_abs_seg".

references: ilock$looplock, ilock$loopunlock, switch_stack
(debug_control, gim_abs_seg)
called by: giminit$list_size, gim1$list_change

CHANNEL:
call channel$safe (giocno, chan, err);
dcl (giocno, chan, err) fixed bin (17);

Ensures termination of a channel by storing data in that channel's mailbox.

references: double$load, double$store
(cat_seg, gim_abs_seg, mailbox)
called by: giminit$assign, giminit$list_size,
gim1$list_change

CHECK:
call check$device_index (devx, cctp, rcode);
dcl (devx, rcode) fixed bin (17), cctp ptr;

Verifies existence of CCT and returns pointer to it.

references: (cat_seg)
called by: giminit$list_size, giminit$unassign,
gim2$list_change, gim2$list_connect,
gim3$get_status, gim4$get_cur_status

- call check$device_name (devnam, dctx, dvx, nrtn);
dcl devnam char(*), (dctx, dvx, nrtn)
fixed bin (17);

Verifies that devnam is in DCT, returns device index.

references: (DCT-SEG)
called by: giminit$assign, giminit$fsassign.
DCWSIZE: 
call dcwsize (dcwp, size);

dcl dcwp ptr, size fixed bin (17);

Returns number of words which would be referenced if a given dcw were executed by the GIOC.

references: none;
called by: gim1$1ist_change, gim1$get_cur_status

DOUBLE: 
\[ \text{b72} = \text{double} \text{load} \text{ (b);} \]

dcl \( b72, b \) bit (72), double$load ext entry bit (72);

Uses "LDAQ" instruction to obtain 72 bits.

references: none

called by: channel$safe, gim1$1ist_change, gim2$1ist_connect, gim4$get_cur_status

- 
call double$store (b1, b2);

dcl \( b1, b2 \) bit (72);

Stores \( b_1 \) in \( b_2 \) using "STAQ" instruction.

references: none

called by: channel$safe, gim1$1ist_change, gim2$1ist_connect

FAKE72: 
call fake72 (giocn, asw, bsw);

dcl giocn fixed bin (17), asw bit (36), bsw bit (72);

Converts model A gioc status words to equivalent model B words.

references: (cat_seg, mailbox)
called by: gim3$get_status
GIM:
The gim is simply a transfer vector which dispatches calls to gim1, gim2, gim3 and gim4. It can be called only by a ring 0 user.

call gim1$list_change (devx, dcwp, datap, listx, count, rcode);

dcl (devx, rcode) fixed bin (17), (dcwp, datap) ptr, (listx, count) fixed bin (12);

See BF.20.01.

references: allo$dcw, allo$free_dcw, channel$safe, check$device_index, dcwsize, double$load, double$store

(cat_seg, CCT, gim_abs_seg, mailbox)

called by: gim$list_change.

GIM2:
call gim2$list_connect (devx, CIW, listx, rcode);

dcl (devx, rcode) fixed bin (17), listx fixed bin (12), CIW fixed bin (18);

See BF.20.01.

references: check$device_index, double$load, double$store, dummy$connect, ilock$looplock, ilock$loopunlock, master_mode_ut$CIOC, Switch_stack

(cat_seg, debug_control)

called by: gim$list_connect

GIM3:
call gim3$get_status (devx, sap, as, os, w, rcode);

dcl (devx, as, os, w, rcode) fixed bin (17), sap ptr;

See BF.20.01.

references: check$device_index, fake72, gim4$get_cur_status, ilock$looplock, ilock$loopunlock

(cat_seg, CCT, debug_control, status_seg)

called by: gim$get_status
GIM4:

call gim4\$get_cur_status (devx, lprwt, dcwt, rcode);

dcl (devx, rcode) fixed bin (17),
   (lprwt, dcwt) fixed bin (12);

See BF.20.01.

references: check\$device_index, dcwsize, double\$load
   (cat_seg, CCT, gim_abs_seg, mailbox)

called by:  gim3\$get_status, gim4\$get_cur_status

GIMINIT:

call giminit\$assign (devnam, devx, event,
   typename, rcode);

dcl devnam char (32), (devx, rcode) fixed bin (17),
   event bit (70), typename char (*);

See BF.20.01.

references: appendb, bin_dec, channel\$safe,
   check\$device_name, delentry,
   dstm\$set_auth, estblseg, get_debug_devnam
   (cat_seg, CCT, dct_seg, debug_control, 
    gim_abs_seg)

called by:  none

- call giminit\$fsassign (devnam, devx, rcode);

dcl devnam char (32), (devx, rcode) fixed
   bin (17);

See BF.20.01.

references: channel\$safe, check\$device_name,
   get_debug_devnam
   (cat_seg, dct_seg, debug_control, 
    gim_abs_seg)

called by:  none
call giminit$unassign (devx);
dcl devx fixed bin (17);
See BF.20.01.
references: check$device_index, delentry, giminit$list_size
            (cat_seg, CCT)
called by: none

- call giminit$list_size (devx, listsize, rcode);
dcl (devx, rcode) fixed bin (17), listsize
     fixed bin (12);
See BF.20.01.
references: allo$dcw, allo$free_dcw, channel$safe,
            (cat_seg, CCT, gim_abs_seg)
called by: giminit$unassign

PLIST
call plist$error (flag, stop, error);
dcl flag char(*), (stop, error) fixed bin (17);
See BE.5.05.
references: messag
called by: none

- call plist$plist (flag, devx, error);
dcl flag char(*), (devx, error) fixed bin (17);
See BE.5.05.
references: bin_dec, bin_oct, check$device_name, messag
            (cat_seg, CCT, debug_control, gim_abs_seg)