TO: MSPM Distribution
FROM: D. L. Stone
SUBJ: BE.5.05
DATE: January 24, 1968

This revision is part of the updating of GIM documents to reflect the new design. In this area only minor changes have been made in the calling sequences and description.

Published: 01/24/68
(Supersedes: BE.5.05, 05/05/67)

Identification

GIOC Interface Module Debugging Aids in 6.36
D. L. Stone

Purpose

When checking out Device Control Modules and their interface to the GIOC Interface Module (GIM), it frequently becomes necessary to examine the various data bases associated with the I/O device in question. Also, it is handy to have a facility for producing readable messages indicating the nature of errors detected within the GIM. A procedure has been written to facilitate these debugging aspects.

Usage

It is presumed that the reader is familiar with standard GIM terminology as outlined in MSPM BF.20.01 - BF.20.03.

In addition to the standard segments necessary for operation of the GIM, users interested in the display and debugging package should include the following statements in their GECOS control file for the MRGEDT run:

```
LIBE    STOSTAT    SLVPRC    SLVACC
LIBE    PLIST      SLVPRC    SLVACC
```

1. To decode and write error messages derived from GIM return status words, the user may make the following call:

```
call   plist$error(userflag,stop_switch,error_word);
```

where suitable declarations are:

```
userflag char(*)            user_supplied identification
error_word fixed bin(17)    GIM return status
stop_switch fixed bin(17)   Stop on error, see below
```

The error display proceeds as follows. The error word is examined for the presence of error bits. If no bits are set, the routine immediately returns. If one or more error bits are set, the following line is written into the 6.36 error file:

```
(userflag)
gim error translation:
```

The user-supplied item, "userflag", is strictly for user convenience. Following this line, the error word is examined, the appropriate error comment is added to the 6.36 error file. After examining the error word, the stop switch is examined. If it is zero, the routine returns. If it is non-zero, the error routine terminates the run with a divide-check fault.

2. To format and display GIM lists currently defined by the user, the following call is suitable:

```
call   plist$plist(userflag,devx,rtn_stat);
```

where suitable declarations are:

```
userflag char(*)     user-supplied identification
devx fixed bin(17)   ID of caller from assign call
rtn_stat fixed       standard GIM return status word
         bin(17)
```

The display routine commences processing by checking whether bit 3 of the gim external variable "debug_control" is ON. If it is not on, the routine returns immediately. If it is on, the routine writes the following line into the 6.36 error file:

```
(userflag)
```

where the user-supplied item, "userflag", is strictly for user convenience. The list data for the user's list is then formatted and printed as follows:

character position	1-12	13-24	26-37	39-44	46-51	53-56
data	spaces	dcwa	dcwb	segno	offset	buffer length

where:

"dcwa" is the first word of the dcw (in OCTAL) and
"dcwb" is the second word.
"segno" and
"offset" are meaningful only if the dcw they reference is a read data dcw. If so, they indicate the buffer into which the data will be copied. Otherwise they are zero or null.
"buffer length" is printed for all data dcw's and is the length in words of the wired down buffer associated with the dcw.

The only error detected by plist is a bad device index. This error will cause a return with "rtn_stat" set before any list information is printed.

3. To simulate the effect of an interrupt occurring and the storing of a selected status word, the user may include the following statement in his program(s):

```
call   stostat(device_index,status_word,listx,
       intsw,rtn_stat)
```

where suitable declarations are:

```
device_index fixed bin(17)   standard device index
status_word bit(18)          skeleton status word
listx fixed bin(12)          the dcw in the list which
                             is to be thought of as
                             generating this status store
intsw bit(1)                 simulate interrupt
rtn_stat fixed bin(17)       GIM return status word
```

The quantity "device_index" is the argument returned from the assign call. "Status_word" is an 18 bit string which is used to construct a 36 bit status word (for Mod A GIOC) or a 72 bit status word (Mod B GIOC). The left six bits are the left six bits of the status word -- the rest are put in bits 19-30. The LPW tally is set from the "LISTX" argument and the channel number is determined from the device-index.

After storing the status word, stostat examines the interrupt switch, "intsw". It this switch is OFF, the procedure returns. If the switch is ON, an interrupt is simulated (by calling the GIM interrupt handler) and then the procedure returns.

Errors are indicated by the standard GIM return status word. The "stostat" procedure automatically handles GIOC status queue overflow exactly as the hardware. Emergency channel overflows cause the following line to be written into the 6.36 error file:

```
ding ding ding
```

After writing the line, the 6.36 run terminates with a divide-check fault.