Published: 03/08/68

Identification

Interim disc control and disc initializer
R. K. Rathbun

Purpose

Herein described is the first implementation of the procedures needed to drive the DSU10 disc subsystem under Multics. This version pretends only to function properly and not to be efficient, elegant, or whatever. It exists merely to provide an environment for the development of Multilevel (see BH.1) and to provide an interface for the expansion of the device utility package (see BG.17). The disc could be substituted for the MSU (drum) for Multics-runs, but the lack of speed will exclude this except under emergency conditions.

The implementation consists of three procedures described below:

interimdisc. A very primitive and unimaginative interface to the disc hardware (more precisely, to the GIM). It will read or write (at most) 16 contiguous 64-word blocks as directed, and return when I/O is completed. This is effected by generating a sequence of one cdcw, the required number of ddcw's, and one cdcw with bit 17 set on; the disc is then connected. The dcw's resulting from the call are reissued ten times or until they are processed without error, whichever occurs first. If ten errors are recorded for the dcw's, an error code is returned.

disc_ctl. The driver of interimdisc. The entry disc_ctl$new_io converts pseudo-commands from dim_ command into their corresponding calls to interimdisc, and posts their completion. The error code is returned as zero, since the disc is assumed always to be running correctly. The entry disc_ctl$run simply returns with the error code set off, since the I/O is totally synchronous.

disc_control_init. An initialization procedure which introduces the disc into the system and creates a permanent dcw-list, all via calls to the GIM.

Calling sequences and arguments

The following are the legitimate calling sequences for the procedures described above; items in declares which are not commented are not used by this version:

```
call interimdisc (op, mem, dev, nrecs, rcode);

dcl  op fixed binary (1),        /* operation:  0 read,
                                    1 write */
     mem fixed binary (24),      /* 24 bit memory address */
     dev fixed binary (22),      /* 22 bit disc address */
     nrecs fixed binary (5),     /* number of blocks */
     rcode fixed binary (35);    /* error code */

call desc_ctl$new_io (did, io, memadd, devadd, p1,
     p2, p3, blocksize, rcode);

dcl  did fixed binary (35),
     io (16) fixed binary (2), /* op for each record
                               ... within the hyper-record
                                      0 read
                                      1 write
                                      2 idle
                                      3 write zeroes */
     memadd fixed binary (18), /* starting memory address
                                  ... divided by 64 */
     devadd fixed binary (35), /* starting disc address
                                  ... divided by
                                  (2 * blocksize) */
     (p1, p2, p3) fixed binary (35), /* posting command
                                  ... and arguments */
     blocksize fixed binary (35), /* number of records per
                                  ... hyper-record */
     rcode fixed binary (35);  /* error code,
                                  ... always returned as
                                  zero */

call disc_ctl$run (did, typecall, rcode);

dcl  did fixed binary (35),
     typecall fixed binary (35),
     rcode fixed binary (35); /* error code,
                                  ... always returned as
                                  zero */

     call disc_control_init;
```

There is no error associated with disc_control_init; if an error occurs during initialization, disc_control_init calls panic.