Identification

The Universal Device Manager Process Groups
S. I. Feldman

Purpose

The universal device manager processes are the processes that normally control I/O devices. This section describes the procedure that initializes and destroys these process groups, under the control of System Control. (See Section BQ.)

Introduction

There is one universal device manager process per universal device manager process group. These groups have user ids of the form ".xx_udmp.yy", where "xx" is the type of device and "yy" is the instance tag. (There may be more than one group handling a particular type of I/O device, and these group will have different instance tags.) For example the first typewriter universal device manager process group will have user id ".typewriter_udmp.aa".

Each device manager process uses a procedure called the Dispatcher (see BQ.2.25). This module is called by the Wait Coordinator when certain events are signaled. The Dispatcher calls the Driver (see BQ.2.24) which in turn makes the appropriate outer call. The Dispatcher’s data base is the Process Dispatching Table (PDT). The section describes the procedure that initializes the PDT of a universal device manager process.

When the system is brought up, System Control creates all of the system processes. For this purpose, System Control has a list of processes and a set of Process Initiation Tables (PITs). (Universal Device Managers will always have such a pre-defined PIT). After the universal device manager process has been created, the following call is made:

```
call udmgr$init(pitptr);
```

The `pitptr` points to a standard PIT (see BQ.1.01). The last entry of the PIT is actually a structure of the following form:

- 4 type char(32),
- 2 pdt_name char(32),
- 2 pit.ndev fixed bin,
2 resource_names(pltptr->plt.ndev) char(32);

Initialization

In response to the call to udmog$init, the following steps are taken:

1. A segment is created in the present process group directory with entry name pltptr->plt.pdt_name. This segment will be the PDT for this process.

2. Store plt.ndev in pdt.nroutes.


4. Set each element of pdt.routes.type equal to plt.type.

5. Set each element of pdt.routes.resource_name equal to the corresponding element of plt.resource_names.

6. Create an event wait channel with priority zero and store its name in plt.shut_down. Give System Control access to this channel.

7. Call ecm$set wait prior, since the above created event is more important than any normal event call event.

8. Make the following call:
   
   call disp$init(pdtptr);
   decl pdtptr ptr; /*point to PDT created above*/

9. Wait for the shut_down event to be signaled.

Destruction

After all of the normal user processes have been destroyed or saved, System Control signals the event whose name was stored in plt.shut_down above. The Wait Coordinator then returns from step 9, and udmog then does the following:

10. Signal the event with name plt.shut_down_complete for the process with ld plt.sys_control. It is assumed that all I/O has been shut down by the time System Control wishes to destroy the universal DMPs.

11. Return to the caller.
The following is the declaration of the PDT:

dcl 1 pdt based(p), /*Process Dispatching Table*/
  2 init_proc char(32), /*name of procedure to be
    " called for initialization.
    Equal to "disp$init"*/
  2 dmp_proc_id bit(36), /*Id of this Device Manager
   " Process*/
  2 reassign_event bit(70), /*event channel to be signaled
     when device is assigned or
     unassigned to this process*/
  2 creator_id bit(36), /*Id of process that created this
   " Device Manager*/
  2 init_done_event bit(70), /*event channel to be signaled when
     Initialization of this process is
     complete.*/
  2 current_ptr, /*pointer to element of routes
   " for device for which work
   is being done at present*/
  2 pdt_name char(32), /*name used by other processes to
   " find PDT*/
  2 dtabp ptr, /*pointer to Driver's driving
   " table*/
  2 disp_ptr, /*pointers to entry points of
   " the Dispatcher*/

  3 reassign_ptr,
  3 locall_ptr,
  3 reenable_ptr,
  3 restart_ptr,
  3 quit_ptr,
  3 hardware_ptr,
  2 nroutes fixed bin(17), /*number of entries in routes array*/
  2 routes(n), /*an entry for each device which
    " may be assigned to this process.
    n = pdt.nroutes*/

  3 type char(32), /*type of resource*/
  3 resource_name char(32), /*resource_name for this device*/
  3 user_id char(50), /*user to whom device is assigned*/
  3 loname char(15), /*DCM loname, a unique character string*/
  3 plbp ptr, /*pointer to PIB for this DSM*/
  3 lcbp ptr, /*pointer to ICB for DSM*/
  3 tbsp ptr, /*pointer to Transaction Block
    segment in user's group
    directory*/

  3 att_stack ptr, /*pointer to entry in attach_stack
    area for pushed-down DCM*/
  3 locall_event bit(70), /*event to be signaled by DSM
    for localling, resetting,
    inverting, and diverting*/
  3 restart_event bit(70), /*signaled to restart a path
    in external quit condition*/
  3 hardware_event bit(70), /*event channel signaled when
"Interrupt received from device/"
3 quit_event blt(70),  /*event to be signaled to stop
   device and prepare for a divert*/
3 reenable_event blt(70), /*signaled when auxiliary
   chain or TBS is unlocked*/
3 device_absent bit(1), /*1 if device not present*/
3 assigned bit(1), /*1 if device assigned to this
   process*/
3 attached bit(1), /*1 if attach call has been
   issued*/
3 ext_quit bit(1), /*1 if device in external quit
   condition*/
3 int_quit bit(1), /*1 if device in internal (hardware)
   quit condition*/
2 attach_stack area((10000)); /*area into which blocks are
   allocated for diverted paths*/
/
*/
dcl 1 att_thread based(p), /*declaration of block to be
   allocated into att_stack
   area for pushing down of
   DCMs*/
   "
2 loname char(15), /*DCM loname*/
2 local_event bit(70), /*event channel name*/
2 reenable_event bit(70), /*event channel name*/
2 plbp ptr,
2 lcbp ptr,
2 status,
3 attached bit(1), /*1 if device not present*/
3 ext_quit bit(1), /*1 if device in external quit
   condition*/
2 next ptr; /*points to next block in thread
   of pushed-down DCMs*/