Published: 11/03/67
(Supersedes: BJ.1.04, 03/15/67)

Identification

The Process Data Block
J. H. Saltzer, R. L. Rappaport, A. Evans

Purpose

Each process in Multics has its own block of storage, maintained by the hardcore supervisor, known as the Process Data Block. The process data block of a process is in wired-down core whenever the process is in the loaded state, but it can be paged onto secondary storage if the process is unloaded. The process data block of a process is contained in the Process Data Segment (see Section BJ.1.03).

Contents of the Process Data Block

Each process data block has an identical structure. The process data block is only accessed by one process, so it needs no interlock. The following items are contained in each process data block. PL/I declarations are shown at the end of this section.

1. Process Identification. This item allows the Basic File System, the Traffic Controller and others to ascertain the identification of the process with a single core reference. The format of a process id is given in BJ.7.03.

2. Stack Pointer Value. This item contains the value of base register sp the last time this process was in execution. It is used to reload sp in swap_dbr. See BJ.5.01.

3. Process Segment Table Entry Pointer. This is a relative pointer which points to this process' entry in the Process Segment Table. It is returned by actproc when the process is activated, and it identifies the process to the file system. See BG.3.03. (This item is also kept in the APT - see BJ.1.01.)

4. Current Ring Number. This item is the simulated ring register. Its value is the ring in which the process is currently executing. See BG.3.05.

5. Wired Ring Number. This item contains the number of a ring whose descriptor segment is currently wired down. This ring is either the current ring or a ring in which a fault or interrupt was encountered which caused the process to leave the ring temporarily. See BG.3.05.

6. Wired_DBR. This item is the dbr value of the descriptor segment for the ring specified by item number 5. Its format is compatible with the 645 opcodes ldbr and sdbr. See BG.3.05.

7. Hardcore_DBR. This is the dbr value of the hardcore descriptor segment, The format is as in item 6 above. See BG.3.05.

8. Block Lock Count. This item records the number of hardcore ring "block type" interlocks that this process currently has set. The maintenance of this count makes it possible to determine when the process may be "quit" without leaving critical system data bases locked and when it is entitled to special priority from the scheduler. See BG.15.02 and BJ.4.00.

9. Stack Switch Temporary. This item provides a small amount of wired down temporary storage needed by the Stack Switching Module. See BK.5.04.

10. Create type. When a new process is being created, special attention is required the first time it is switched to. The fact of this item's being non-zero indicates such a need to swap_dbr, and the specific value indicates what type of processing to use. This item is almost always zero.

PL/I Declarations

All the above items are entries in the process data block. The item names and their respective PL/I declaration are given below.

```
/* Declarations for the Process Data Block */

dcl (

     pds$processid bit(36),     /* Process Identification */

     pds$last_sp bit(36),       /* save sp while blocked */

     pds$pstep bit(18),         /* entry in Process Segment Table */

     pds$cur_ring fixed,        /* current ring number */

     pds$wired_ring fixed,      /* ring whose descriptor is
                                   wired down */

     pds$wired_dbr bit(36),     /* descriptor for previous
                                   item */

     pds$hardcore_dbr bit(36),/* address of hard core ring
                                   descriptor */

     pds$block_lock_count       /* count of block locks set */
     fixed,

     pds$sstemp bit(72),        /* wired down temp for stack
                                   switcher */

     pds$create_type fixed      /* process-being-created switch */

     ) external;
```