14.1 GENERAL

Previous chapters have described the way in which interrupts (LKM requests, completion of I/O action, etc.) are processed by the Monitor. When the processing of an interrupt is complete, control is handed to the Dispatcher, which then determines which application or Monitor tasks are able to proceed. If several tasks are able to proceed, a task is chosen on a 'First-In-First-Out' basis within priority level. Registers A1 to A14 are restored for this task and the task is entered via a RTN instruction.

The RTN instruction automatically enables interrupts to occur. Any interrupt of an equal or lower priority which occurred during the processing of the last interrupt would have been queued. On the RTN instruction being executed this interrupt will take effect immediately and control will again be passed to an interrupt handler.

However, if no interrupt has been queued, the task will begin execution. Execution of the task will continue until another interrupt occurs (which may of course be an LKM instruction executed by the task). When processing of this interrupt has been completed the task will again become a candidate for scheduling, and so on.

A task may exit by issuing an LKM request type 3 (EXIT). The dispatcher will then delete all record of the task and the task will cease to exist. The task can be activated again (from another task) but the register contents will then be undefined except for the contents of registers used for passing parameters.

In order to identify and schedule tasks, each task must have a task identifier and priority level. Application tasks must be assigned a task identifier and priority level during system loading (SYSLOD). Monitor tasks (e.g. data management task, segment loading task) have a predefined task identifier and priority level and need not be specified during system generation. These task identities are preceded by #.

Scheduling of tasks by the LKM processors and Dispatcher is illustrated in figure 14.1.

The following notes refer to the numbers in the diagram:-

1. The dispatcher selects the next task to be dispatched by inspecting the dispatcher queue and taking out the task of the highest priority present which has been the longest time in the queue, i.e. the scheduling is performed in FIFO (First-In-First-Out) basis per level.

2. When a task is activated or restarted it is inserted in the dispatcher queue at its priority level, and within that level, put last in the queue.

3. When a task tries to activate another task which is already active, the request is put in the pending queue (Not the task).

4. When the running task performs an exit, the pending queue is checked, and if there is a pending request for the exiting task, then that task is reactivated (see 2).

5. When the running task issues an I/O LKM (with wait), for example, the task will not be considered for dispatching until the I/O is completed. Meanwhile, the dispatcher selects another task according to 1.

6. When the event that the task was waiting for is completed, the task is inserted in the dispatcher queue according to the principle described in 2.

7. A task may perform the request 'Switch task on same level', which means that the task is put last in the queue within its level, and the next task on the same level is dispatched.

The situation may also arise in 1 that the dispatcher queue is empty, i.e. all tasks are waiting for events to be completed. Then the monitor is 'idling' in the 'idle loop' (priority level 63) waiting for an interrupt to occur. The interrupt handler will finally pass control to the dispatcher, after having queued the requesting task in the dispatcher queue.

In MMU systems, the dispatcher takes care of the loading of the MMU with the contents of the MMU table in TTAB for the task to be dispatched. It also checks that the requested code segment is situated in memory, and if it is not, the dispatcher will request loading of the code segment before dispatching the task. When a task is queued in the dispatcher queue, the contents of the MMU table are saved in the task table (TTAB).

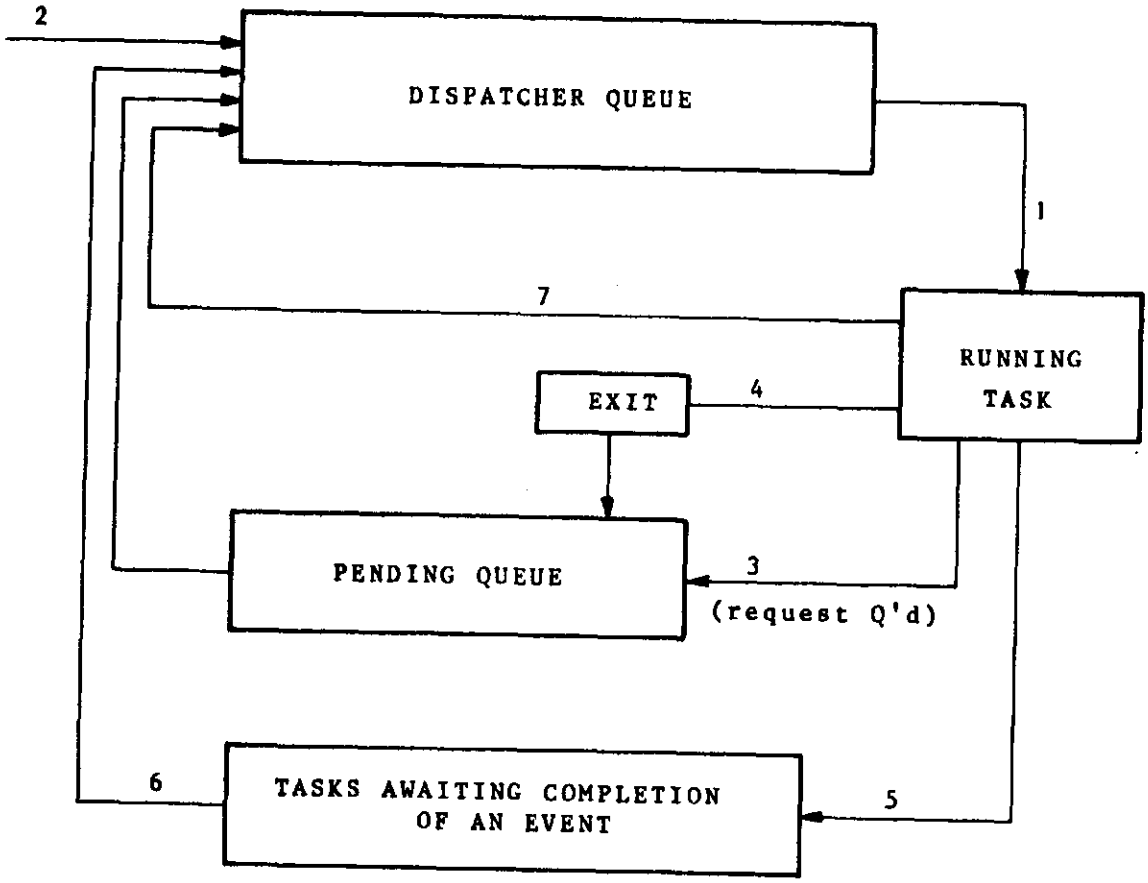
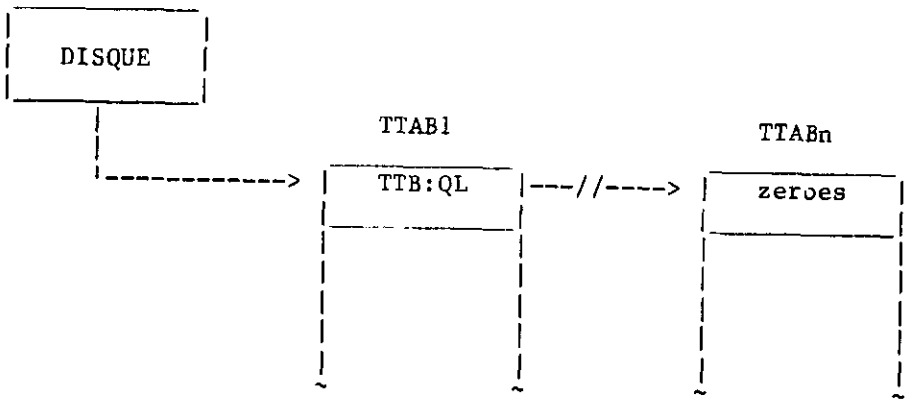


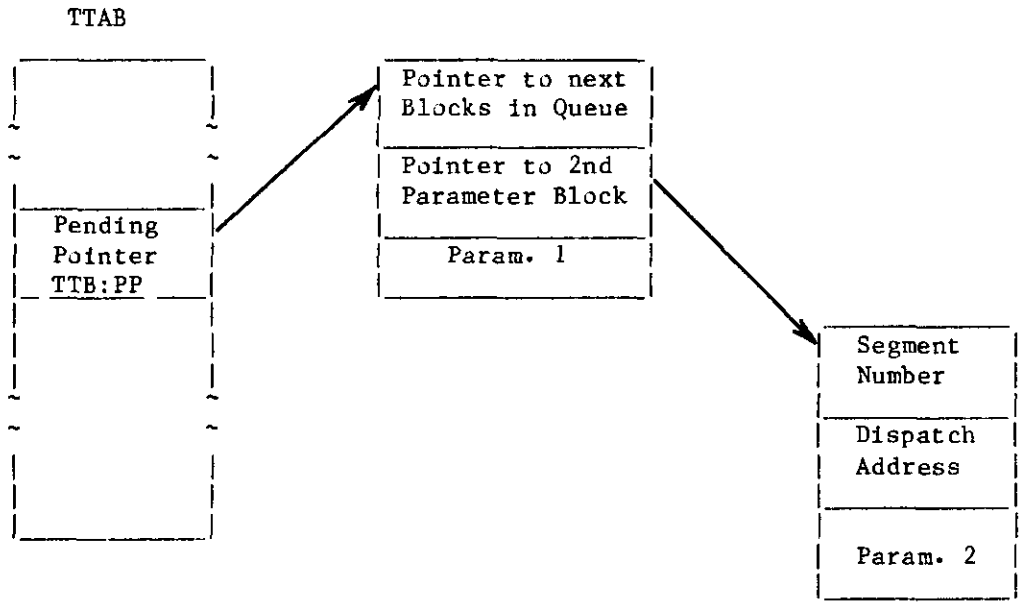
Figure 14.1. Task Scheduling.

14.2 THE DISPATCHER QUEUE



DISQUE is the queue anchor of the dispatcher queue. The contents of this word refer to the first task table (TTAB) in the dispatcher queue. The first word in TTAB is designated TTB:QL and, when it contains zeroes, indicates the end of the dispatcher queue. When the dispatcher queue is empty, word DISQUE contains zero.

14.3 THE PENDING QUEUE



When a task is activated, and the task is already active, i.e. already in the dispatcher queue, then this activation request is put into the pending queue. Six words (2 times three blocks) are obtained from the monitor block pool and pointed to by the word TTB:PP in the task table. These words contain information required to activate the task after an EXIT is performed by that task. When the word TTB:PP contains zero, the pending queue does not have to be checked. Insertions in the pending queue are done on a FIFO basis, except for data management tasks, for which the basis is LIFO.

The meanings of the words are as follows:-

* Pointer to next blocks in queue.
  This word contains a pointer to the next pending request in the queue; end of queue when zero.

* Pointer to 2nd parameter block.
  Pointer to the second block, which contains more information about the pending request.

* Segment number.
  This word contains the segment number in which the activation address (dispatch address) is held.

* Dispatch address.
  This word contains the dispatch address within a segment.

* Parameters 1 and 2.
  Two parameters available to the TOSS monitor.

When an EXIT is performed the pending queue is checked and the blocks released.

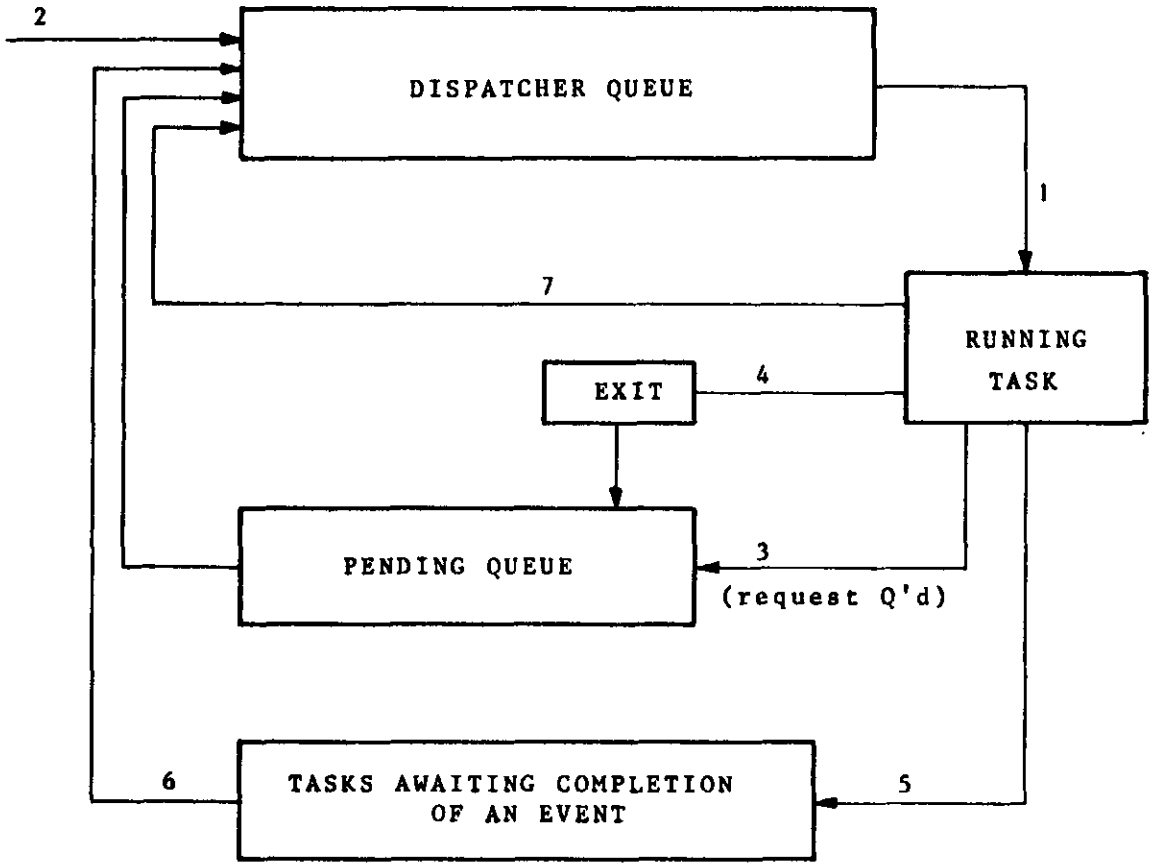
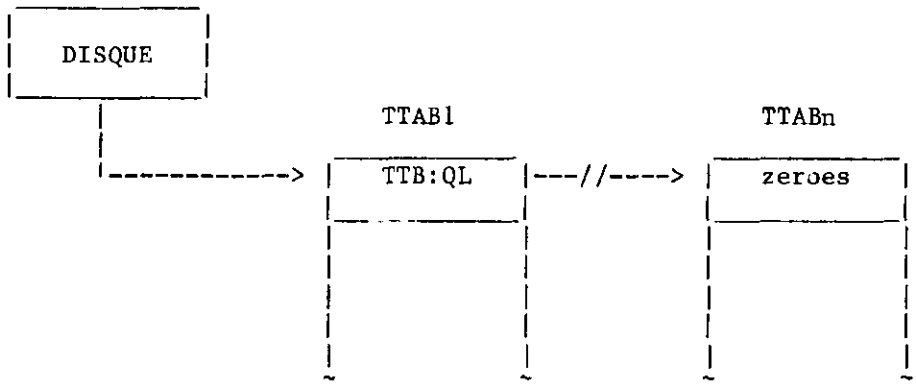


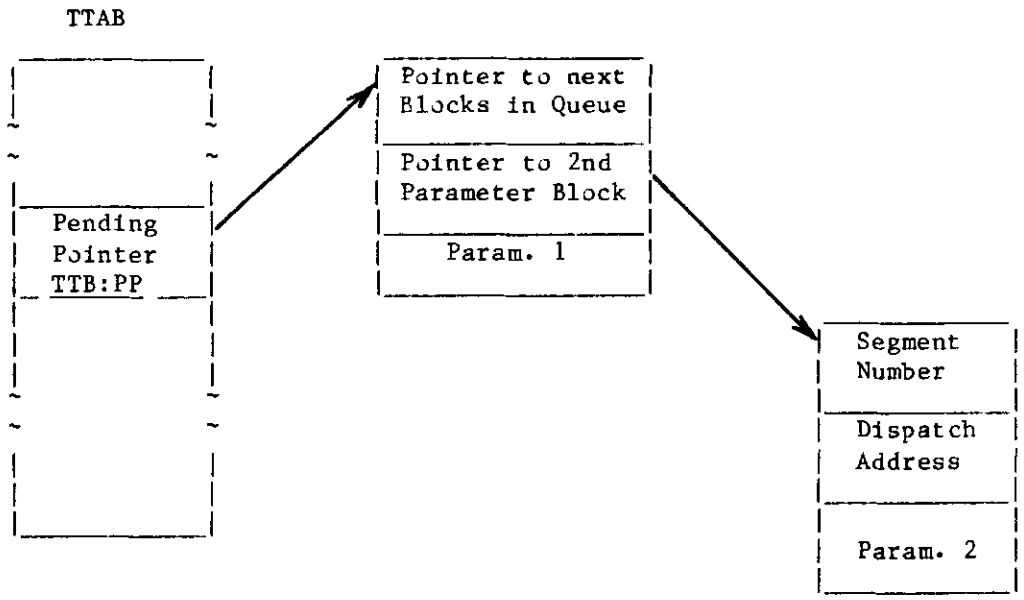
Figure 14.1. Task Scheduling.

14.2 THE DISPATCHER QUEUE



DISQUE is the queue anchor of the dispatcher queue. The contents of this word refer to the first task table (TTAB) in the dispatcher queue. The first word in TTAB is designated TTB:QL and, when it contains zeroes, indicates the end of the dispatcher queue. When the dispatcher queue is empty, word DISQUE contains zero.

14.3 THE PENDING QUEUE



When a task is activated, and the task is already active, i.e. already in the dispatcher queue, then this activation request is put into the pending queue. Six words (2 times three blocks) are obtained from the monitor block pool and pointed to by the word TTB:PP in the task table. These words contain information required to activate the task after an EXIT is performed by that task. When the word TTB:PP contains zero, the pending queue does not have to be checked. Insertions in the pending queue are done on a FIFO basis, except for data management tasks, for which the basis is LIFO.

The meanings of the words are as follows:-

* Pointer to next blocks in queue.
  This word contains a pointer to the next pending request in the queue; end of queue when zero.

* Pointer to 2nd parameter block.
  Pointer to the second block, which contains more information about the pending request.

* Segment number.
  This word contains the segment number in which the activation address (dispatch address) is held.

* Dispatch address.
  This word contains the dispatch address within a segment.

* Parameters 1 and 2.
  Two parameters available to the TOSS monitor.

When an EXIT is performed the pending queue is checked and the blocks released.