An 'interrupt' is an event which causes control to be passed, at the completion of the current instruction, to an address held in one of the 'interrupt vectors' in memory words 0 to 63. For example:-

* Power failure.
* LKM request.
* Real time clock update.
* Completion of I/O action.
* Attempted Execution of an illegal instruction.

The particular vector used depends upon the type of interrupt. Each vector contains a pointer to an associated 'interrupt handler'.

For example, memory word 0 is the interrupt vector for power failure. When power failure occurs the sequence of instructions currently being executed is interrupted, and control is handed to the instruction pointed to by memory word 0 (the power failure interrupt handler).

When an interrupt occurs the currently executed instruction is always completed before control is passed to the appropriate interrupt routine. All interrupt routines pass control to the dispatcher when their actions are completed. An interrupt is also generated in the event of system stack overflow (the stack is pointed to by A15). This will cause a system halt because processing can not be allowed to continue as the consequences are unpredictable.

As can be seen from figure 12.1, the interrupt handlers in boxes 1, 2, & 4 are self contained; they do not call any subsidiary modules. The handlers in boxes 1 & 2 terminate by halting the machine. The handler in box 4 terminates by branching to the dispatcher (box 38).

The interrupt handler for the real time clock (box 3) activates a special clock task (box 43) every 100 ms. The hardware interrupt occurs every 20ms. The handler terminates by branching to the dispatcher (box 38). The interrupts generated during I/O operations are serviced by the appropriate device driver (eg. box 25).
At completion of the I/O, the driver calls TENDIO (in module TOSSIO) before branching to the dispatcher.

The interrupt handler for LKM requests processes only request type 0. The interrupt handler saves registers A1 to A14 in the task table (TTAB) of the current task and, if necessary, branches to one of several LKM processors in order to process the remaining types of LKM requests.

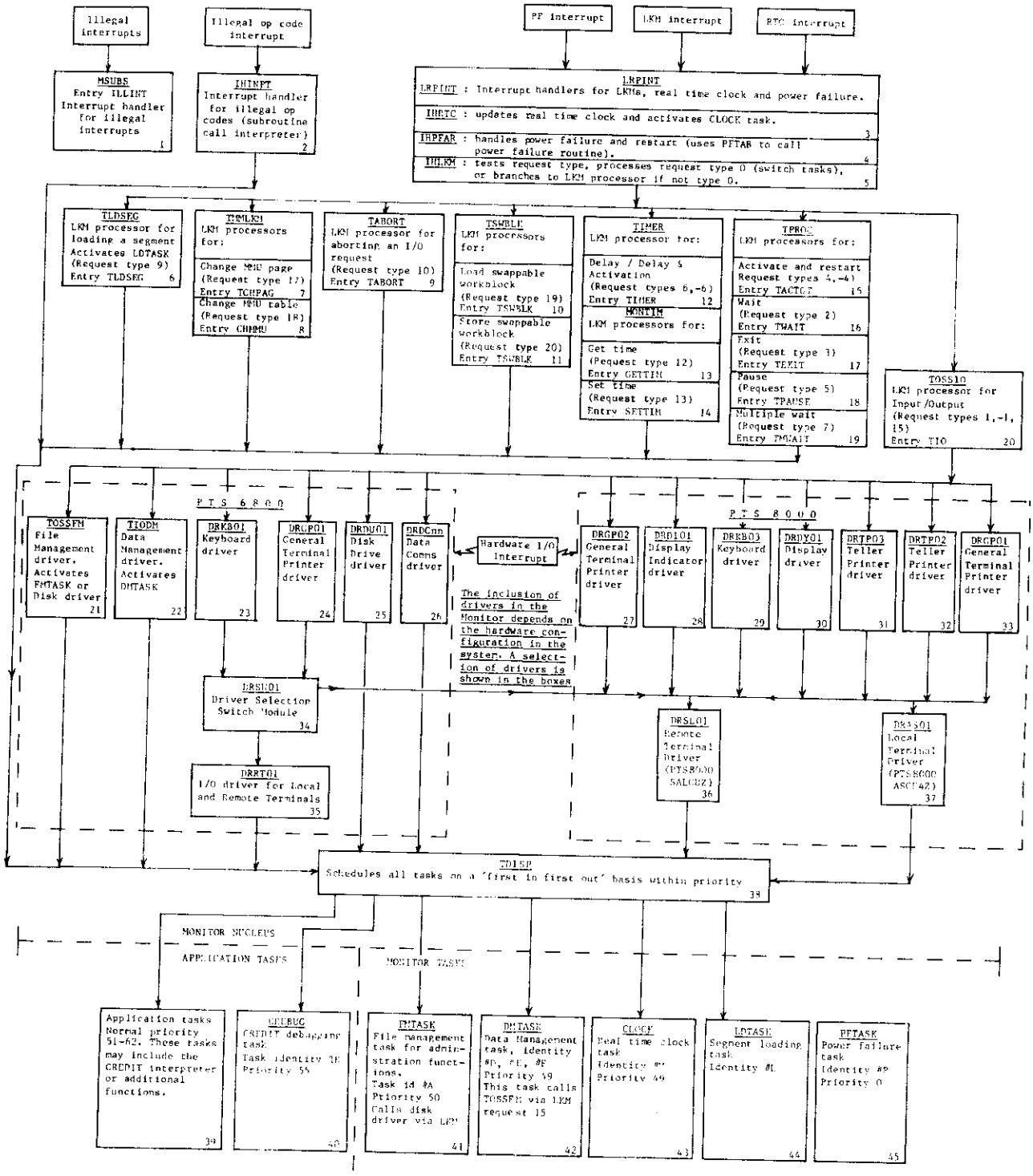

Figure 12.1. TOSS Component Overview.