Published: 11/30/66

Identification

Block
R. L. Rappaport

Purpose

Entry point block in the Process Exchange is the mechanism by which processes, waiting for external events to occur, give away control of the processor on which they are currently running.

Preface

The description of block that follows is divided into two sections. The first section presents the basic outline of the subroutine. This would be an adequate description if it could be assumed that execution of the subroutine will take place while:

1) The processor is completely masked against interrupts.

2) A global interlock is on which denies access to the Process Exchange to all processes except the one in which this subroutine is currently executing.

The second section is a complete specification that describes the steps that must be taken to allow more than one process to be concurrently executing in the Process Exchange.

Basic Outline

Processes calling entry point block are processes which cannot or do not wish to continue executing until some event of interest to the process takes place. Basically block goes through three steps.

1) It determines whether an event of interest has already occurred. It does this by checking the status of the wakeup-waiting switch for the calling process. This switch is one of the data items in the process' entry in the Active Process Table. If the switch is on, block resets it to off and performs an immediate return to its caller instead of giving away control of the processor. If the switch is off, block continues with the next step.

2) It sets the state of the process, as defined by switches in the Active Process Table entry for the process, to the "blocked" state.

3) It calls getwork in order to give away the processor. Return is experienced from getwork only after an event of interest has occurred. Upon return from getwork, block resets the wakeup-waiting switch and returns to its caller.

The calling sequence for block is simply:

```
call block;
```

and the stack used in this call is the calling process' Process Concealed Stack. Calls to block from within the hardcore supervisor use the Process Concealed Stack to make the call directly. Calls to block from outside the hardcore supervisor are directed to a module known as the validator (see Section BD.8) which performs the necessary stack switching.

Figure 1 illustrates the basic outline of block.

Complete Specification of Block

With several processes possibly executing in the Process Exchange concurrently, steps must be taken to coordinate their actions. In particular, two general steps have been taken. First, certain interlocks and switches have been placed in the Active Process Table entry of each process. By observing common rules about the interlocks the various modules are able to guarantee the integrity of the data with which they deal. Secondly, at certain times while some of these interlocks are set, the processor referencing the locked data must be masked against all interrupts. This is to prevent the possibility of putting a processor into an infinite loop. (For a complete discussion of coordination in the Process Exchange see section BJ.6)

Block makes three contributions to this coordination effort:

1. Upon entry block sets on the calling process' intermediate-state switch. This switch resides as a data item in the calling process' Active Process Table entry. If the calling process' wakeup-waiting switch is off, block will change the process' execution state to the "blocked" state before the process stops executing. The intermediate-state switch serves to notify other processes that although

the process is defined to be blocked it may actually be executing. Therefore before the wakeup-waiting switch is interrogated the intermediate-state switch is set on. If the wakeup-waiting switch is on block resets the intermediate-state switch before the return. If the wakeup-waiting switch is off the intermediate-state switch is reset in swap-dbr (see section BJ.5.01) after the process has ceased executing.

2. Block makes use of several data items which other Process Exchange modules might also make use of. In order to prevent fatal mishaps an interlock has been created which controls access to these data items. The interlock is the process-state lock which is another date item in each process' Active Process Table entry. The items to which it governs access are:

    1. The running switch
    2. The ready switch
    3. The wakeup-waiting switch

    The first two items define the process' execution state. The third serves to notify whether or not some other process is trying to "wakeup" this process. The running switch and the ready switch of a process must never be referred to or altered unless the process' process-state lock has previously been set. Whenever a process tests its own wakeup-waiting switch, the process-state lock must be on. However, a process may turn off this switch without first setting this interlock. (A complete discussion of the use of this interlock is given in BJ.6.) Therefore block must lock the process-state lock before testing the wakeup-waiting switch. The lock must remain set until after the process' execution state is redefined to blocked or until it is determined that the wakeup-waiting switch is on.

3. While the process-state lock is set the processor must be masked against all interrupts. This is to prevent an interrupt from being serviced whose handler might encounter this same process-state lock. Therefore in block the processor must be completely masked before the process-state lock is set and cannot be unmasked until this lock is reset.

    At this point we are in a position to completely specify block:

    1. The intermediate-state switch of the calling process is set on.

2. The current processor interrupt mask is saved and the processor is masked against all interrupts.

3. The process-state lock is locked.

4. If the wakeup-waiting switch is off, go to step 5. Otherwise:

    a) unlock the process-state lock

    b) restore the previous processor interrupt mask

    c) turn off the intermediate-state switch

    d) go to step 9

5. Set the process' execution state to "blocked".

6. Unlock the process-state lock.

7. Restore the previous processor interrupt mask.

8. Call getwork.

9. Reset the wakeup-waiting switch.

10. Return.

Figure 2 is a complete flow diagram of block.

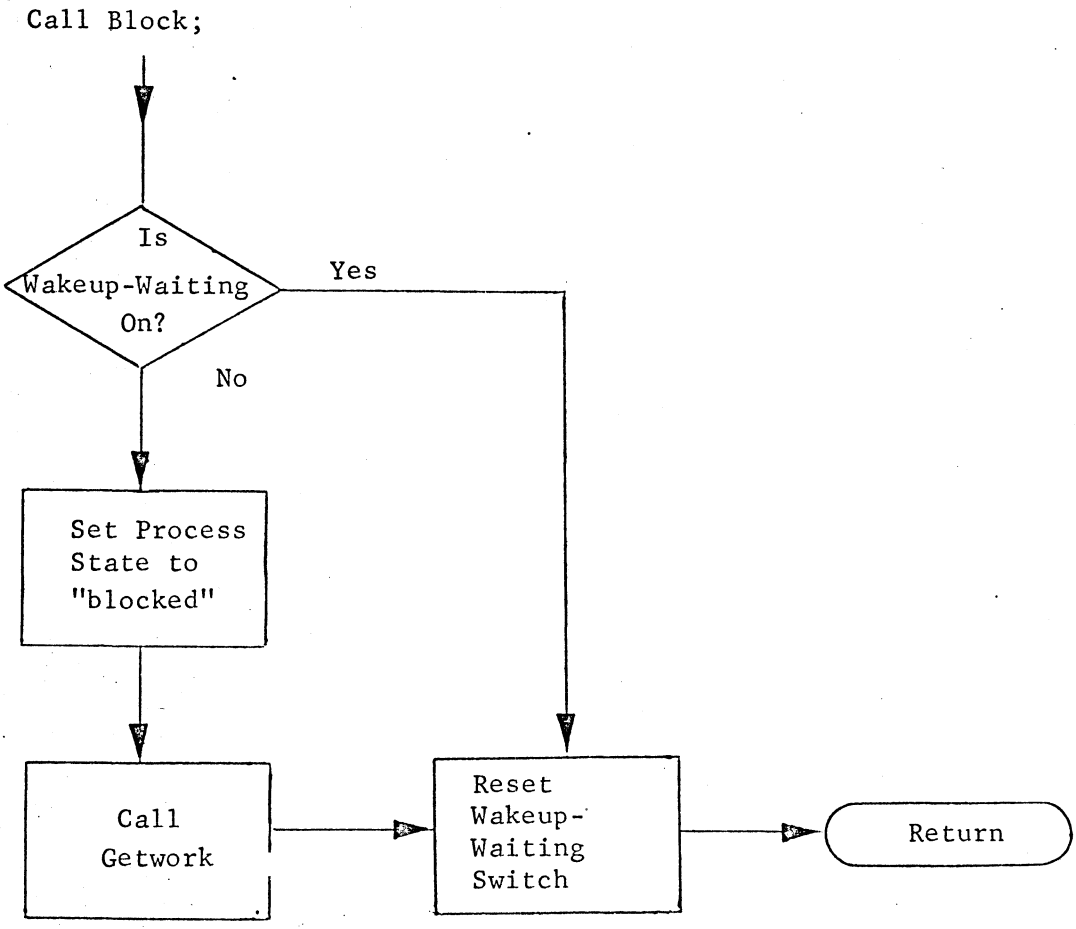

Figure 1. Basic Outline of Block

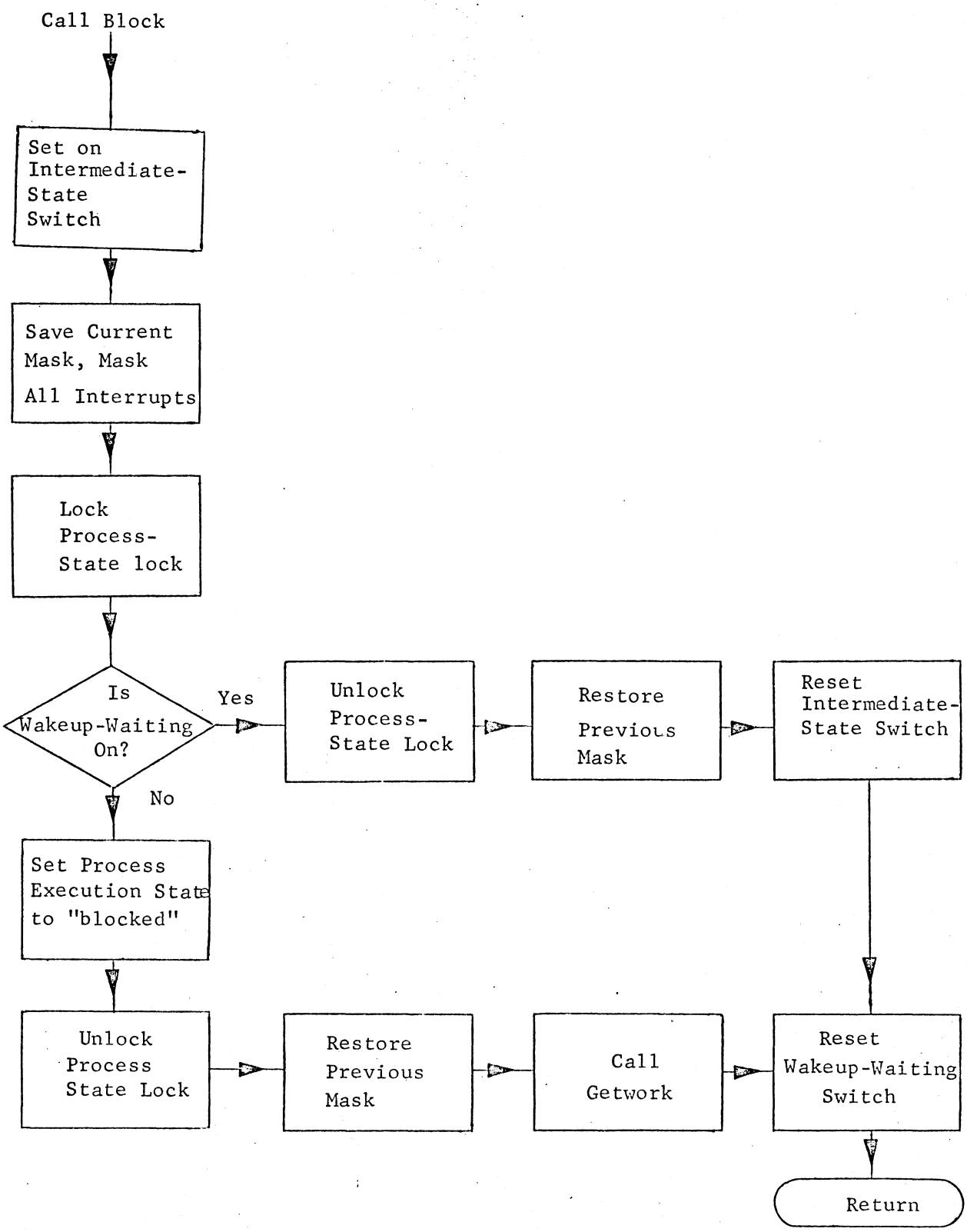

Figure 2. Complete Flow diagram of block