Published: 06/15/67

Identification

Interrupt Mask Procedure
L.J. Lambert, J.H. Saltzer

Purpose

The mask procedure allows hardcore ring modules to set processor interrupt masks without knowledge of system controller interrupt cell assignments. The procedure is master mode, in order to execute the privileged SMCM instruction.

Discussion

The Multics masking strategy views interrupt masks as members of an ordered set, with each member, in order, implying a more restrictive mask than the previous member. A number of mask levels are therefore defined, which describe successively more restrictive interrupt mask settings. A procedure is provided to set the mask to a certain level, and another is provided to get the current mask level. A third procedure briefly sets the mask to a desired level, then restores the previous mask. This third procedure is used to drain certain interrupts or to permit pre-emption only at certain well-defined points of a procedure.

An important constraint in the design of these routines is that the knowledge of which active devices may actually set interrupt cells for this processor be hidden from the caller. By the "rule of 32" (BC.1.04) interrupt cell assignments are the same for all processors, and therefore all processors may use the same set of mask patterns. In particular, if Processor A needs to be masked against interrupts of level equal to and lower than the drum, it is sufficient to set the standard drum-level interrupt mask in A's system controller without asking whether or not Processor A is the receiver of drum interrupts, or any of the lower level interrupts which should also be masked if they can happen to Processor A.

A second important constraint of these mask routines is that the caller not be aware of absolute interrupt cell assignments in the system controller. This constraint is met by providing a separate symbolic in-reference to the system communication segment, scs, for each possible mask level. The caller of set_mask need merely supply a pointer to one of these in-references by writing, for example,

```
     call master_mode_ut$set_mask (scs$drum_level, temp);
```

The reference to scs$drum_level picks out an appropriate constant which set_mask can recognize and interpret to mean "mask all interrupts of drum level and below." The constant is, in fact, the correct bit pattern to be loaded into the AQ register in preparation for the privileged "Set Memory Controller Mask" instruction. The mask constants in <scs> also have the property that more highly restrictive masks are larger in numerical value, so that a program can compare two masks to determine which represents the more restrictive masking level.

Calling sequences

In each of the following calling sequences, mask_pattern is a bit string of length 72.

```
     call master_mode_ut$set_mask (mask_pattern, temp);
```

will set the current interrupt mask to that of the bit string mask-pattern, and return the old mask in the variable temp. (Temp is also bit (72).)

```
     call master_mode_ut$get_mask (mask_pattern);
```

will set the variable mask_pattern to contain the current interrupt mask, as determined by reading the memory controller mask register.

```
     call master_mode_ut$open_mask (mask_pattern);
```

is equivalent to the following sequence:

```
     call master_mode_ut$set_mask (mask_pattern, temp);

     call master_mode_ut$set_mask (temp, temp);
```

with the guarantee that between the two "set mask" operations there will be the opportunity for interrupts allowed by mask_pattern to occur.

Symbolic Mask names

The following is a list of symbolic entry points in the segment "SCS", containing the masks indicated. The masks which end in name "level" are in the order of fewer number of interrupts masked as you read down the page. The four names "sys_level", "drum_level", "gioc_level", and "swap_level" are synonomous with the names they are next to in the chart, and have meanings of mask "all interrupts", "drum and below", "gioc and below", and "process interrupts only", respectively.

Intervention interrupt

The masks in segment <scs> do not mask interrupt cell 32 which is reserved in all system controllers for the operator emergency intervention interrupt.

symbolic mask name	interrupts masked
gioc0_level, sys_level	/* GIOC status channel 0 and lower */
clock_trouble_level	/* clock trouble and lower */
drum_ctl_level, drum_level	/* drum control and lower */
drum_data_level	/* drum data and lower */
drum_pgm_level	/* drum program and lower */
gioc1_level, gioc_level	/* gioc status channel 1 and lower */
gioc2_level	/* gioc status channel 2 and lower */
gioc3_level	/* gioc status channel 3 and lower */
clock_pgm_level	/* clock programmed interrupts and lower */
pre_empt_level, swap_level	/* pre_emption and lower */
time_out_level	/* time_out interrupts and lower */
quit_level	/* quit interrupts and lower */
open_level	/* all interrupts allowed */
drain_prempt	/* special mask to allow pre-emption interrupt only */
drain_timeout	/* same for timer runout */
drain_quit	/* same for quit */

Published: 06/28/67

Identification

Appendix to BK.5.01
L. J. Lambert

Erratum

Page 2, delete last 6 lines;

Insert

will set the interrupt mask to that of the bit string mask-pattern long enough to guarantee that there will be the opportunity for interrupts allowed by mask-pattern.