PDP-11 Interrupts: Variations On A Theme

Bob Supnik, 03-Feb-2002 [revised 20-Feb-2004]

Summary

Despite the presence of documented standards and example implementations, PDP-11 devices showed significant variability in implementing interrupts. While some of these variations were nearly invisible or harmless, others required explicit support or workarounds in device drivers. Consequently, PDP-11 emulators must model device interrupt control with great care.

The "Standard" Implementation

Until the advent of message-oriented devices like the TS11 and the MSCP controllers, all PDP-11 devices contained a "control/status register". The CSR contained a device ready flag in bit <7> and an interrupt enable flag in bit<6> (other common assignments were error summary in bit<15> and go in bit<0>):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR								RDY	IE						GO

The device's interrupt request was implemented with an edge sensitive flip-flop. INTR was set by the rising edge of the logical AND of RDY and IE; it was cleared by device initialization or by interrupt acknowledge; and its output was masked by the logical AND of RDY and IE. The entire circuit required just two AND gates and a JK flip flop:

```
                        +-----+       |--\
          |--\   +5V--|D   1|-----|    |
RDY ---|    |       |     |     |    |--- INTERRUPT REQUEST
       |    |---+---|C     |   +--|    |
IE  ---|    |   |   +--^--+   |  |--/
       |--/     |       |       |
              +------|-----+
                        |
INIT + IAK -----------+
```

Behaviorally,

- A transition of RDY AND IE from 0 to 1 set the interrupt request.
- Granting the request, or initializing the device, cleared the interrupt request.
- Clearing either RDY or IE blocked the interrupt request. This cannot be distinguished from clearing the request.
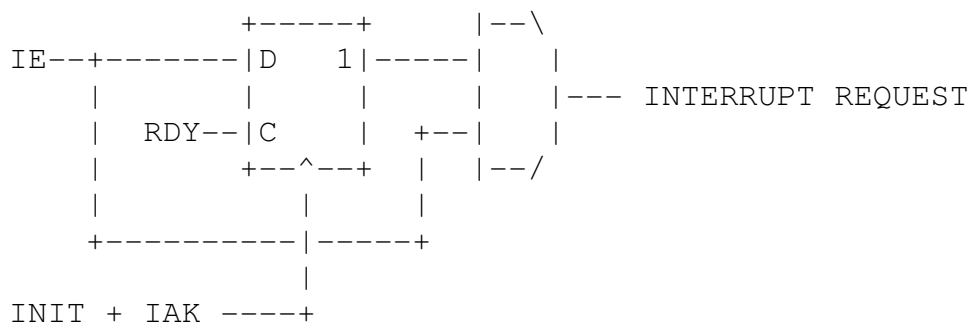
This circuit is presented in all the standard Unibus handbooks and is included in all the early PDP-11 device controllers, such as the PC11 (paper tape), DL11 (serial line), and RK11 (cartridge disk). The circuit was reduced to silicon in the Qbus interrupt logic chip (DC003).

Variations

As the TTL logic family broadened, variations began to appear in the interrupt circuitry. The standard implementation seemed overly general. While it made sense to request an interrupt on the rising edge of RDY, why request an interrupt on the rising edge of IE? The following variation began to appear:

```
              +-----+       |--\
IE--+-------|D   1|-----|     |
    |       |     |     |     |--- INTERRUPT REQUEST
    |  RDY--|C    |  +--|     |
    |       +--^--+  |  |--/
    |          |     |
    +----------|-----+
               |
INIT + IAK ----+
```

If IE was set, the rising edge of RDY requested an interrupt. Once the interrupt was set, clearing IE would block the interrupt request. As before, both device initialization and interrupt acknowledge cleared the interrupt request.

This variation apparently saved a gate with no impact on function. But it had one peculiarity: an interrupt request, once set, could not be cleared by program action. Clearing IE did not actually clear the interrupt request; more importantly, clearing RDY neither cleared nor blocked the interrupt request. The seeds for future confusion were sewn.

The RH70 and RH11

The Massbus controllers took the circuit variant described above, added an additional "feature", and in doing so created something unique. Disk controllers have always had an issue in handling overlapped seeks on multiple units. Software would like to have an interrupt for each distinct operation; but to multiplex all the seek complete interrupts, and the controller complete interrupt, onto a single interrupt request line requires complex mechanisms like unit polling, as in the RK11. Without this mechanism, software must time overlapped seeks, as in the RL11.
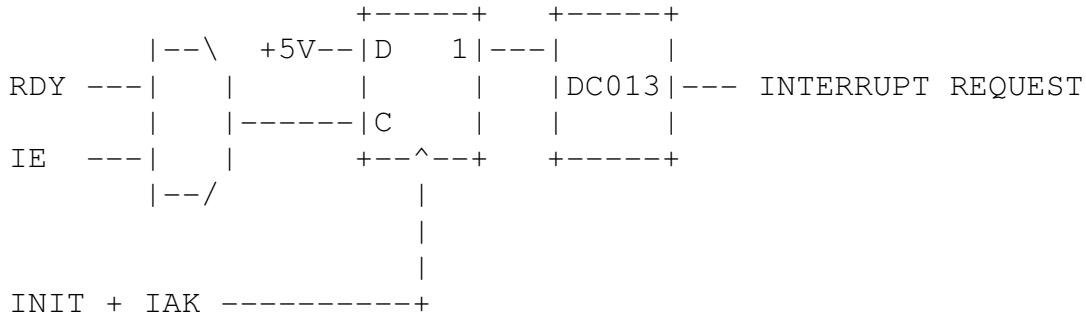
The Massbus designers proposed a simple hardware-software combination to handle this problem. Each disk drive would have an "attention" (ATA) flag.

Provided that the controller was enabled for interrupts and was not performing a data transfer, the controller would request an interrupt if any ATA flag was set. The software driver would have to clear the ATA flag of the drive requesting attention. Thus, ATA interrupts behaved like the level-sensitive interrupts of the PDP-8 and PDP-15.

To implement this additional class of interrupt, the Massbus controller simply OR'd the ATA interrupt requests with the output of the RDY interrupt circuit. But because the ATA request already included IE, and the RDY circuit gated IE, it omitted the final AND with IE:

```
                +-----+        |--\
IE-----------|D    1|----- \   \
                |        |       |  >--- INTERRUPT REQUEST
RDY---------|C       |  +-- /   /
                +--^--+  |   |--/
                    |       |
ATA + RDY + IE-------+
                    |
INIT + IAK ----+
```

Now here was a circuit with truly peculiar properties. A transition of RDY from 0 to 1 with IE set latched an interrupt request. This request not only couldn't be cleared by program action; it couldn't be blocked by software action. That is, once the interrupt request was latched, clearing IE did not prevent the interrupt from happening! This inexcusable mistake didn't even save gates; the correct implementation required only a 3-input AND gate in place of the 3-input OR gate:

```
        +--------------------------+
        |  +----------------------+ |
        |  |                          | |   |--\
        |  |     +-----+        |--\  | +--|    |
IE--+--|----|D    1|----- \   \  +----|    |-- INTR REQUEST
        |  |     |        |       |  >------|    |
RDY----+----|C       |  +-- /   /        |--/
                +--^--+  |   |--/
                    |       |
ATA ----------------+
                    |
INIT + IAK ----+
```

With this implementation, RDY interrupt would have behaved like "classical" PDP-11 edge-sensitive interrupts, while ATA would be a level-sensitive interrupt conditioned on RDY and IE.

DEC drivers didn't utilize the peculiarities of the RH70 and RH11, but UNIX variants, such as Ultrix-11, did. Simulators attempting to model the RH series

controllers cannot running Ultrix-11 without mimicing the behavior of its interrupt logic.

The DEUNA

The DEUNA shows that the advent of purpose-built IC's did not fix the problem. The DEUNA replaced the last AND gate of the classic implementation with the DC013 Unibus grant controller chip:

```
                      +-----+    +-----+
          |--\  +5V--|D   1|---|     |
RDY ---|    |        |     |   |DC013|--- INTERRUPT REQUEST
          |    |------|C    |   |     |
IE  ---|    |        +--^--+   +-----+
          |--/           |
                          |
                          |
INIT + IAK ----------+
```

Unfortunately, the DC013 had no enable input, only a request input. Thus, once the DEUNA requested an interrupt, there was no programmatic way to remove it. As with the RH11/RH70, simulators attempting to model the DEUNA must model its incorrect interrupt behavior.

Acknowledgements

Joseph Young first diagnosed problems in the RH simulators running BSD 2.9 and 2.11 and pointed to the interrupt logic as the source. Tom Evans and Alan Baldwin brought the DEUNA example to my attention.