

# **Real Time Systems – SS2016**

**Prof. Dr. Karsten Weronek**

**Faculty 2**

**Computer Science and Engineering**

Interrupts

Please be aware that Petri nets a mathematical construct. There are a lot of variation and a lot of use cases. There are complete series of lectures at some universities and there are international groups working on this topic only.

However, to model RTS it is **essential**  
to know and understand the fundamentals and  
**to apply the graphical representation of Petri nets!**

In system programming, an **interrupt** is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.

An **interrupt** alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing.

The processor responds by suspending its current activities, saving its state, and executing a function called an **interrupt handler** (or an interrupt service routine, ISR) to deal with the event.

This interruption is temporary, and, after the interrupt handler finishes, the processor resumes normal activities.

There are two types of interrupts:

1. **hardware interrupts** and
2. **software interrupts.**

Hardware interrupts are used by devices to communicate that they require attention from the operating system.

Internally, hardware interrupts are implemented using electronic alerting signals that are sent to the processor from an external device, which is either a part of the computer itself, such as a disk controller, or an external peripheral.

For example, pressing a key on the keyboard or moving the mouse triggers hardware interrupts that cause the processor to read the keystroke or mouse position.

Unlike the software type (described below), **hardware interrupts are asynchronous** and can occur in the middle of instruction execution, requiring additional care in programming.

The act of initiating a hardware interrupt is referred to as an **interrupt request (IRQ)**.

A software interrupt is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which causes an interrupt when it is executed. The former is often called a **trap** or **exception** and is used for errors or events occurring during program execution that are exceptional enough that they cannot be handled within the program itself. For example, if the processor's arithmetic logic unit is commanded to divide a number by zero, this impossible demand will cause a divide-by-zero exception, perhaps causing the computer to abandon the calculation or display an error message.

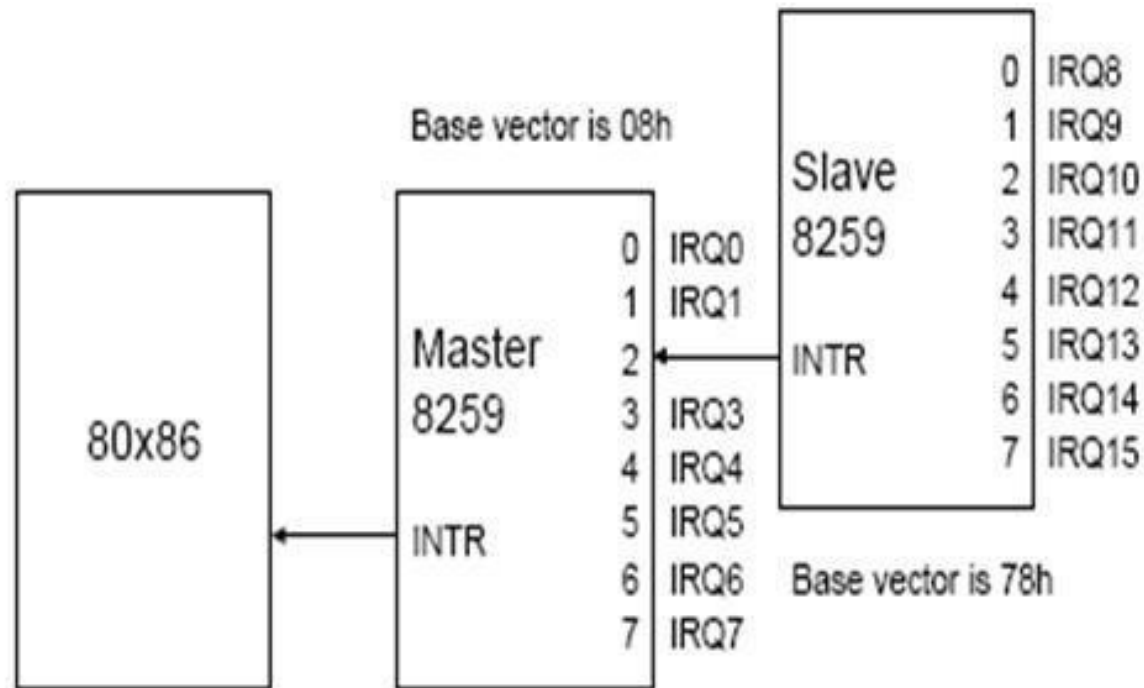
Software interrupt instructions function similarly to subroutine calls and are used for a variety of purposes, such as to request services from low-level system software such as device drivers. For example, computers often use software interrupt instructions to communicate with the disk controller to request reading or writing of data from and to the disk.

Each interrupt has its own interrupt handler.

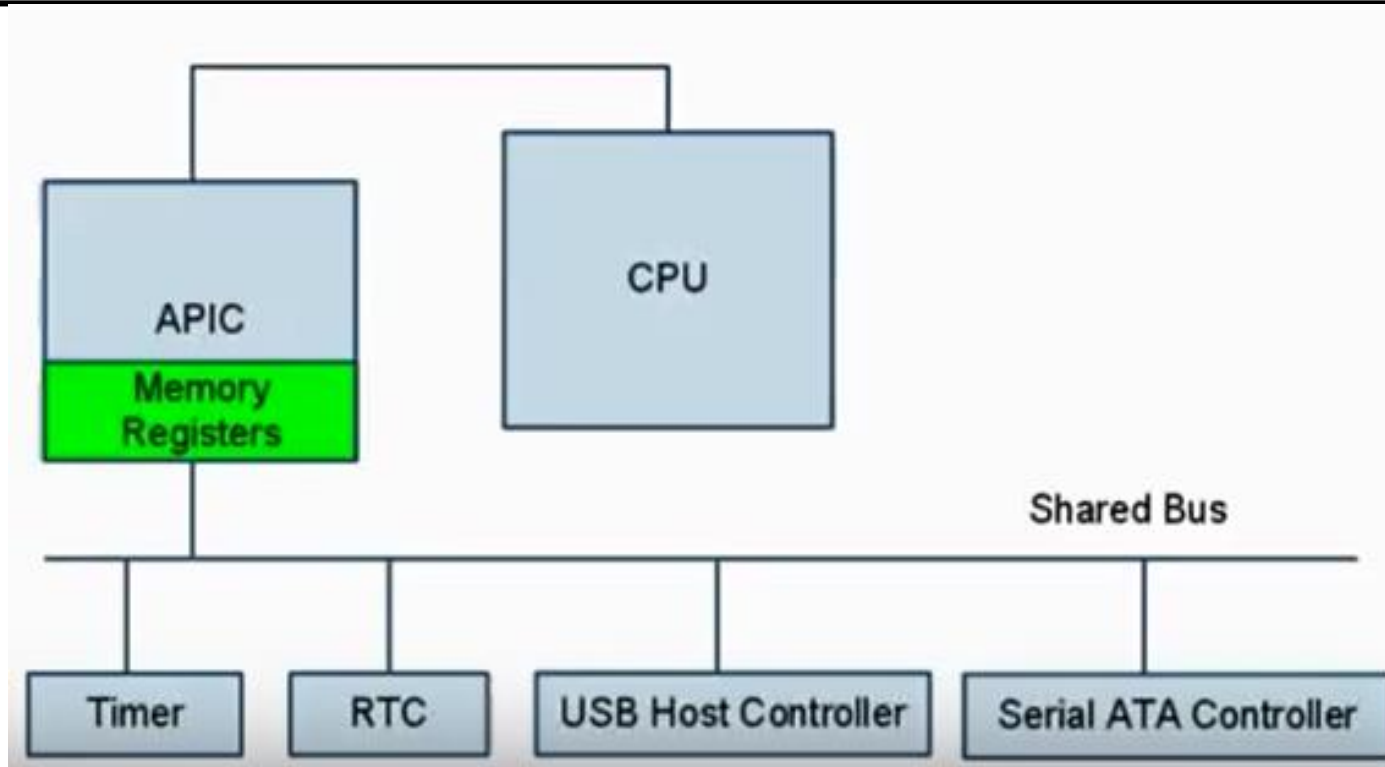
The number of hardware interrupts is limited by the number of interrupt request (IRQ) lines to the processor, but there may be hundreds of different software interrupts.

Interrupts are a commonly used technique for computer multitasking, especially in real-time computing.

Such a system is said to be interrupt-driven.



Hardware interrupts were introduced as an optimization, eliminating unproductive waiting time in polling loops, waiting for external events. They may be implemented in hardware as a distinct system with control lines, or they may be integrated into the memory subsystem. If implemented in hardware, an interrupt controller circuit such as the IBM PC's Programmable Interrupt Controller (PIC) may be connected between the interrupting device and the processor's interrupt pin to multiplex several sources of interrupt onto the one or two CPU lines typically available. The number of hardware interrupts are very limited!



- MSI: Message Signal Interrupts: By writing into the associated memory each device can produce Device sends a message to the Advanced Programmable Interrupt Controller via a shared bus
  - Message indicates which device is requesting attention
  - no data transmitted
  - registers on APIC store messages until handled by the OS
- only interrupt mechanism available on PCIe/express bus
- Legacy PCI/ISA support requires APIC to support physical interrupt lines as well as MSI registers
- MSI supports up to 32 IRQs per device, MSI-X up to 2,048



**Maskable interrupt (IRQ):** a hardware interrupt that may be ignored by setting a bit in an interrupt mask register's (IMR) bit-mask.

**Non-maskable interrupt (NMI):** a hardware interrupt that lacks an associated bit-mask, so that it can never be ignored. NMIs are used for the highest priority tasks such as timers, especially watchdog timers.

**Inter-processor interrupt (IPI):** a special case of interrupt that is generated by one processor to interrupt another processor in a multiprocessor system.

**Software interrupt:** an interrupt generated within a processor by executing an instruction. Software interrupts are often used to implement system calls because they result in a subroutine call with a CPU ring level change.

**Spurious interrupt:** a hardware interrupt that is unwanted. They are typically generated by system conditions such as electrical interference on an interrupt line or through incorrectly designed hardware.

Processors typically have an **internal interrupt mask** which allows software **to ignore all external hardware interrupts** while it is set.

Setting or clearing this mask may be faster than accessing an interrupt mask register (IMR) in a PIC or disabling interrupts in the device itself.

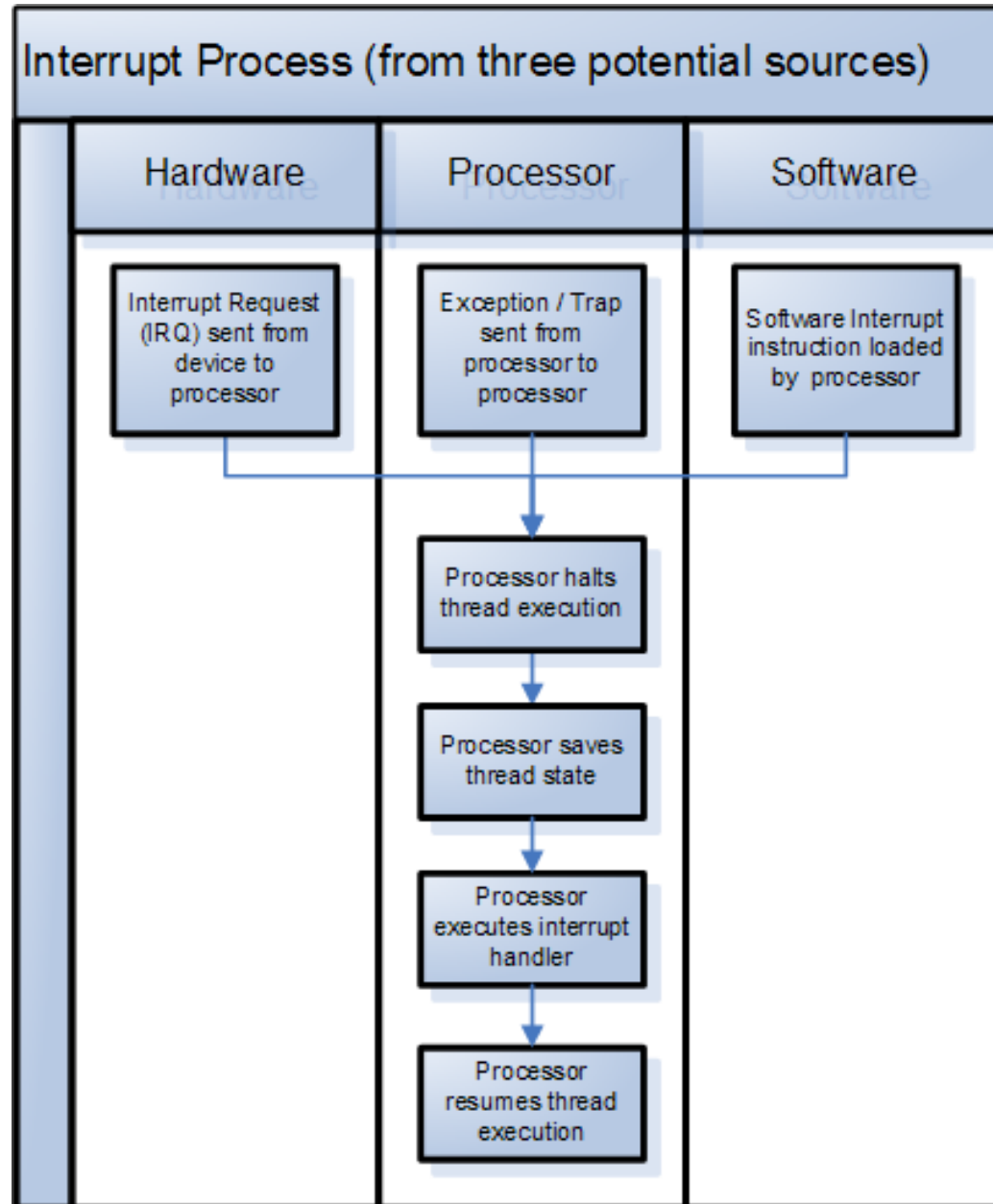
This internal interrupt mask is sometimes very suitable for RTS.

An interrupt that leaves the machine in a well-defined state is called a **precise interrupt**. Such an interrupt has four properties:

1. The Program Counter (PC) is saved in a known place.
2. All instructions before the one pointed to by the PC have fully executed.
3. No instruction beyond the one pointed to by the PC has been executed (that is no prohibition on instruction beyond that in PC, it is just that any changes they make to registers or memory must be undone before the interrupt happens).
4. The execution state of the instruction pointed to by the PC is known.

An interrupt that does not meet these requirements is called an **imprecise interrupt**

1. **Level-triggered interrupts:**  
is an interrupt signalled by maintaining the interrupt lin at high or low level
2. **Edge-triggered interrupt**  
is an interrupt signalled by a level transistion on the interrupt line, either a falling edge (high to low) or a rising edge (low to high).
3. **Hybrid:**  
the hardware not only looks for an edge, but it also verifies that the interrupt signal stays active for a certain period of time
4. **Message signaled**  
the device signals its request by sending a short message over some communication medium



[https://en.wikipedia.org/wiki/File:Interrupt\\_Process.PNG](https://en.wikipedia.org/wiki/File:Interrupt_Process.PNG)