Interrupts : *Interrupt vs. Polling*

```
int main()
{
  while(1){
    . . .
   }
}

 OnSwitch_ISR{
  getData()
}
```

Interrupt

```
int main()
{
   while(1){

   if(switch = on ){
   getData(); }
   . . .
   }
  }
```

Polling

Interrupts : *Interrupt vs. Polling*

A single microprocessor can serve several modules by:

- **Interrupt**

  When module needs service, it notifies the CPU by sending an interrupt signal. When the CPU receives the signal the CPU interrupts whatever it is doing and services the module.

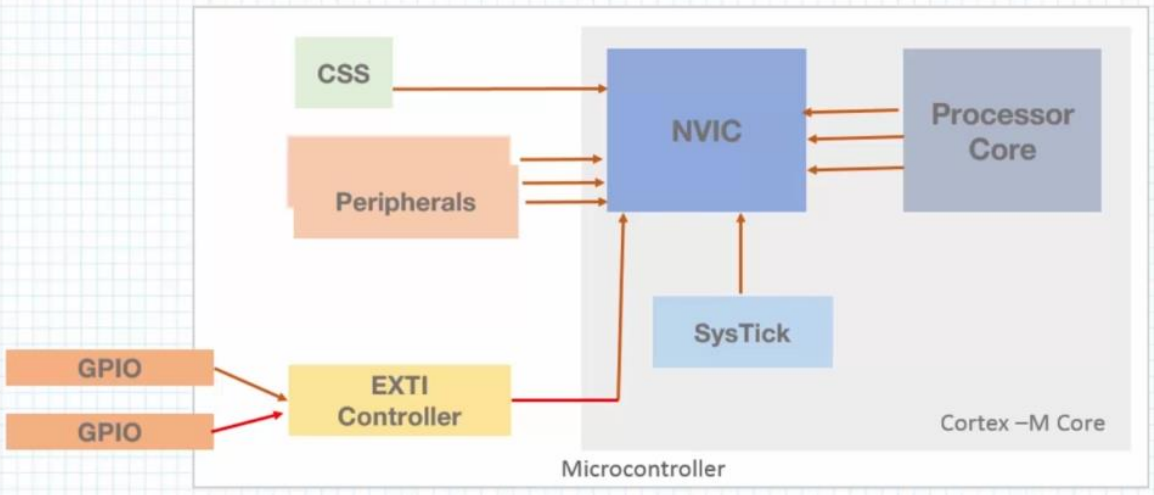
- **Polling**

  The CPU continuously monitors the status of a given module, when a particular status condition is met the CPU then services the module.

Interrupts : *Interrupt Service Routine*

The function that gets executed when an interrupt occurs is called the Interrupt Service Routine(ISR) or the Interrupt Handler

Interrupts : *NVIC*



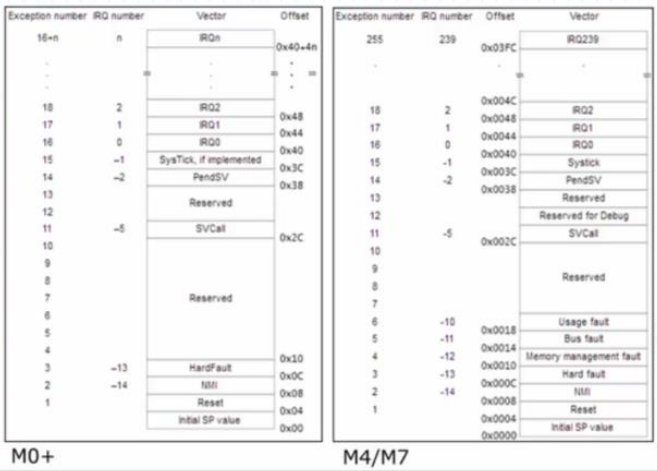
Interrupts : *NVIC*

Nest Vector Interrupt Controller (NVIC)

- A dedicated hardware inside the Cortex-Microcontroller
- It is responsible for handling interrupts.

Interrupts : *The Vector Table*

- The vector table contains the addresses of the Interrupt Handlers and Exception Handlers.



(Page3 203 in reference manual)

Interrupts : *External Interrupt (EXTI) lines*

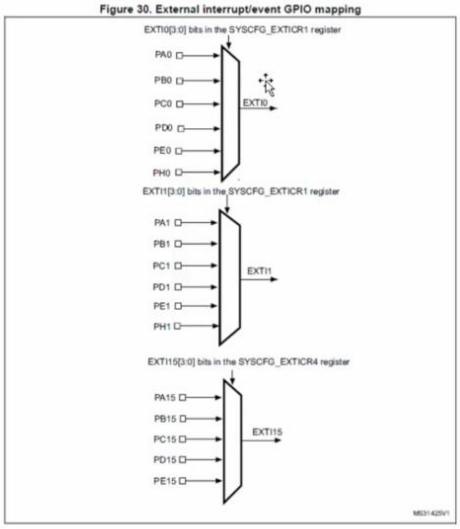
- GPIO pins are connected to EXTI lines
- It possible to enable interrupt for any GPIO pin
- Multiple pins share the same EXTI line

- **Pin 0 of every Port** *is connected* ***EXTI0_IRQ***
- **Pin 1 of every Port** *is connected* ***EXTI1_IRQ***
- **Pin 2 of every Port** *is connected* ***EXTI2_IRQ***
- **Pin 3 of every Port** *is connected* ***EXTI3_IRQ***

. . .

This means we cannot have PB0 and PA0 as input interrupt pins at the same time since they are connected to the same multiplexer i.e. EXTI0

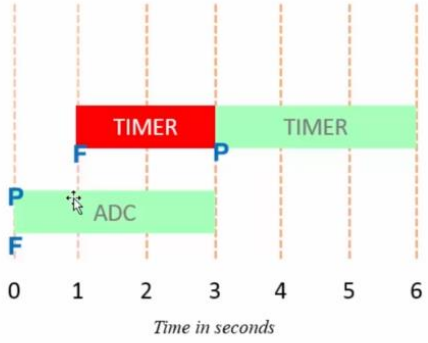
Same for PC4 and PB4 at the same time, etc.



(Page 208 in reference manual)

Interrupt priority

Interrupts : *States- Pending vs. Active*



Legend

- **Active State**
- **Pending State**
- **P Pending State Cleared**
- **F Interrupt fired**

Let's assume
ADC Interrupt has a higher priority than TIMER interrupt

ADC Interrupt fires at time t = 0.
This is indicated by F

Since there is no other interrupt, the pending state is cleared and the interrupt becomes active.
This is indicated by P

At time t=1 TIMER interrupt fires
This is indicated by F

Since it has a lower priority than the ADC *interrupt it remains in the pending state*

At time t=3 ADC interrupt completes its execution

Since there is no other interrupt with a higher priority , the pending state of the TIMER interrupt is cleared and the interrupt becomes active.
This is indicated by P

2.3.1 Exception states

Each exception is in one of the following states:

Inactive	The exception is not active and not pending.
Pending	The exception is waiting to be serviced by the processor. An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.
Active	An exception that is being serviced by the processor but has not completed. *Note: An exception handler can interrupt the execution of another exception handler. In this case both exceptions are in the active state.*
Active and pending	The exception is being serviced by the processor and there is a pending exception from the same source.

Interrupt preemption

Programing

Hardware interrupt selection

To configure the 23 lines as interrupt sources, use the following procedure:

- Configure the mask bits of the 23 interrupt lines (EXTI_IMR)
- Configure the Trigger selection bits of the interrupt lines (EXTI_RTSR and EXTI_FTSR)
- Configure the enable and mask bits that control the NVIC IRQ channel mapped to the external interrupt controller (EXTI) so that an interrupt coming from one of the 23 lines can be correctly acknowledged.

(Page 207 in reference manual)