

# Using External Interrupts from Digital Lines

Corrado Santoro

**ARSLAB - Autonomous and Robotic Systems Laboratory**

Dipartimento di Matematica e Informatica - Università di Catania, Italy

santoro@dmi.unict.it

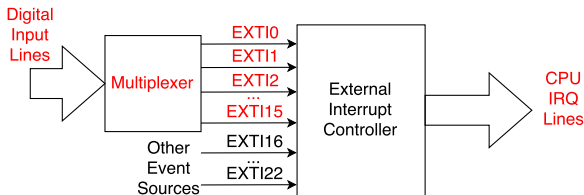


L.A.P. 1 Course

# Digital Lines and Interrupts

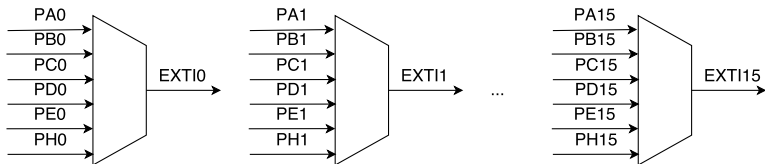
- Each digital input line can trigger an **IRQ**
- The **IRQ** can be configured to be triggered on:
  - **falling edge** (transition “1-to-0”)
  - **rising edge** (transition “0-to-1”)
  - **both edges**
- The circuit that handles the configuration is called **External Interrupt Controller (EXTI)**

# External Interrupt



- The **External Interrupt Controller** handles (internally) **23 interrupt lines**, called **EXTI0, EXTI1, ..., EXTI22**
- Lines **EXTI0, ..., EXTI15** can be connected to GPIO digital inputs through a multiplexer, so **at most** 16 digital inputs can be used, at the same time, as interrupt sources
- Lines **EXTI15, ..., EXTI22** are internally connected to interrupt sources from other peripherals

# The Multiplexer



- The **EXTI multiplexer** configures the **input source** for each EXTI line
- Each **EXTIx** line can be connected to pin **x** of **only one** GPIO

# Using EXTI Interrupts

- 1 Configure the multiplexer in order to connect a **GPIO** pin to a **EXTI** line
- 2 Configure the **EXTI** line to handle the interrupt by specifying the **signal edge**:
  - **falling edge**
  - **rising edge**
  - **both edges**
- 3 Write the **Interrupt Service Routine** relevant to the EXTI line: there you must handle the interrupt and then cancel it by means of a proper function call

- Configure the multiplexer:

```
void GPIO_config_EXTI(GPIO_TypeDef * gpio,  
                      EXTI_line exti);
```

- **gpio** is the port, i.e. **GPIOA**, **GPIOB**, ...
  - **exti** represents the EXTI line, i.e. one of the constants **EXTI0**, **EXTI1**, ..., **EXTI15**
- Enable an EXTI line by configuring the edge:

```
void EXTI_enable_EXTI(EXTI_line exti,  
                     edge_type edge);
```

- **exti** represents the EXTI line, i.e. one of the constants **EXTI0**, **EXTI1**, ..., **EXTI15**
- **edge** represents the edge, i.e. one of the constants **RISING\_EDGE**, **FALLING\_EDGE**, **BOTH\_EDGES**

# EXTI IRQ Handlers

- `void EXTI0_IRQHandler(void)`, line EXTI0
- `void EXTI1_IRQHandler(void)`, line EXTI1
- `void EXTI2_IRQHandler(void)`, line EXTI2
- `void EXTI3_IRQHandler(void)`, line EXTI3
- `void EXTI4_IRQHandler(void)`, line EXTI4
- `void EXTI9_5_IRQHandler(void)`, lines EXTI5 to EXTI9
- `void EXTI15_10_IRQHandler(void)`, lines EXTI10 to EXTI15

# stm32\_unict\_lib functions to check/clear IRQs

- Check the occurrence of IRQ:

```
int EXTI_isset(EXTI_line exti);
```

- **exti** represents the EXTI line, i.e. one of the constants **EXTI0, EXTI1, ..., EXTI15**

- Clear the occurrence of IRQ:

```
void EXTI_clear(EXTI_line exti);
```

- **exti** represents the EXTI line, i.e. one of the constants **EXTI0, EXTI1, ..., EXTI15**



# First Example: LED toggle using EXTI interrupts

```
#include "stm32_unict_lib.h"

int main()
{
    // LED at PC3
    GPIO_init(GPIOC);
    GPIO_config_output(GPIOC, 3);

    // pushbutton Y (PB4)
    GPIO_init(GPIOB);
    GPIO_config_input(GPIOB, 4);

    GPIO_config_EXTI(GPIOB, EXTI4);
    EXTI_enable(EXTI4, FALLING_EDGE);

    for (;;) { } // do nothing
}

void EXTI4_IRQHandler(void)
{
    if (EXTI_isset(EXTI4)) {
        GPIO_toggle(GPIOC, 3);
        EXTI_clear(EXTI4);
    }
}
```

# Using External Interrupts from Digital Lines

Corrado Santoro

**ARSLAB - Autonomous and Robotic Systems Laboratory**

Dipartimento di Matematica e Informatica - Università di Catania, Italy

santoro@dmi.unict.it



L.A.P. 1 Course