Labwork 2 – Interrupts and Wiring

H. Cassé <hugues.casse@irit.fr>

This labwork has two goals:

- Programming timer and GPIO using interrupts.
- Connecting new sensors/actuators on STM32.

1 Programming with Interrupts

1.1 Interrupts with the timer

Source: 🗀`src/ex6.c`
Implements the blinking application (with an half-period of 500ms) on the Timer 4 using interrupts.

1.2 Interrupts with the GPIO

Source: 🗀`src/ex7.c`
We want to invert the state of green LED according to the user blue button. Each time is it clicked, it passes from on to off and from of to on.

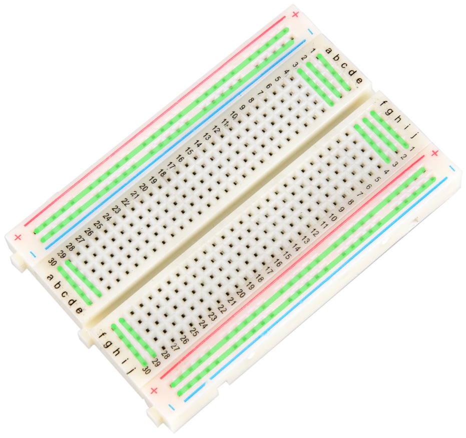
1. Implement the application with an interrupt.
2. What do you observe (to convince yourself the interrupt is raised, put a breakpoint inside it).
3. Are you able to explain the dysfunction? If this explanation is not obvious, put `printf` to scan push and release of the button.
4. How do you fix this problem? Implement this fix.

2 Wiring the STM32F4

2.1 Experimenting with the STM32F4 board

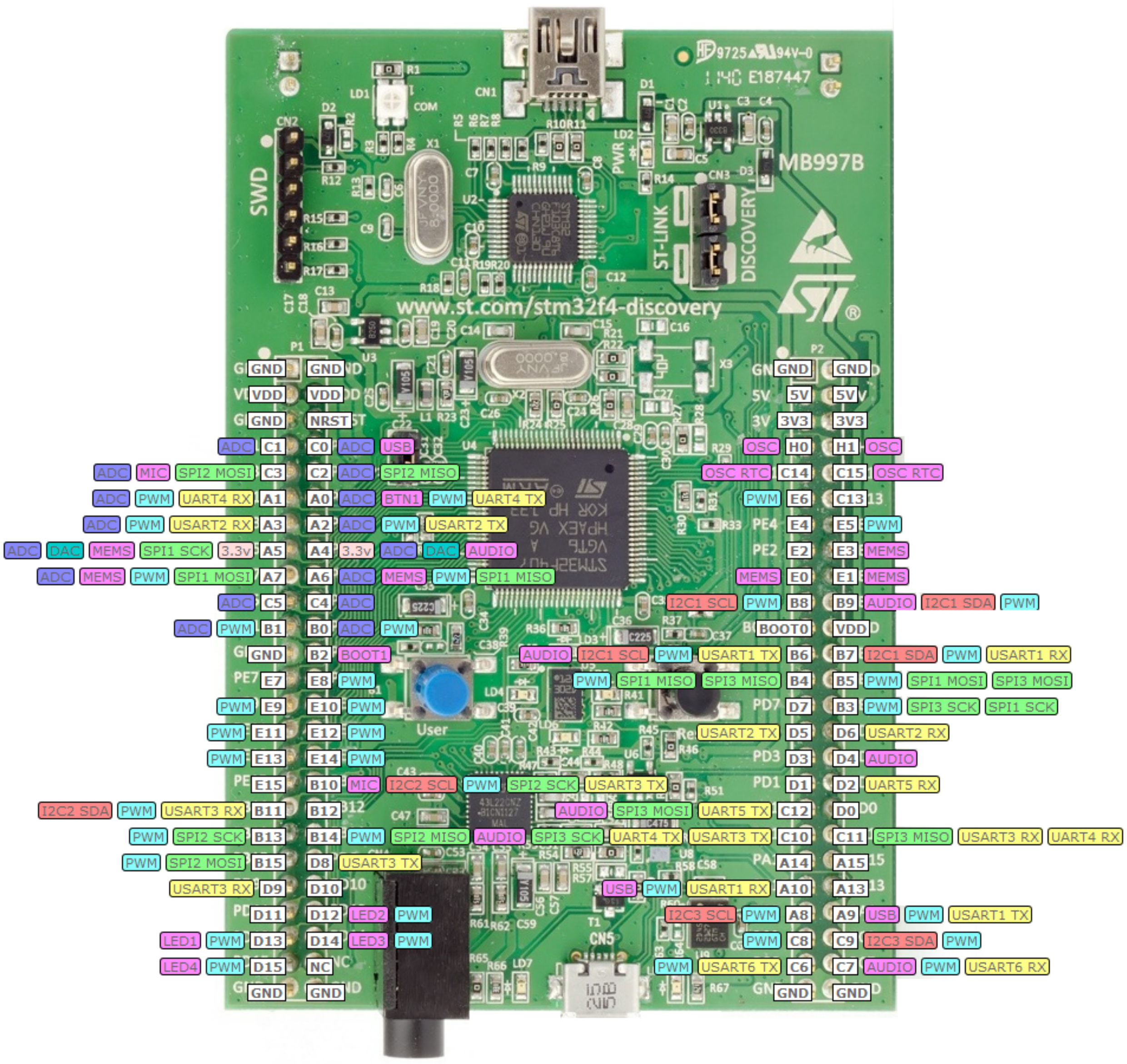
Even though the final use of the board is to solder sensors and actuators, it may first useful to perform experimentation. This means that we want to connect sensors and actuators but not definitively tiying the devices to the board.

To achieve this, one will use a breadboard as displayed below. A breadboard is basically a metalloc word with holes. According to some wiring map, thes holes are linked together and allow to plug wires, connecting the wires if the holes are linked. The picture below shows the wiring holes:



Remark the series of orthogonal holes on the sides denoted + and - that are useful to connect VCC (power) and ground.

In the opposite, the STM32 comes with a special wiring on its own pins described below:


GND | GND — GND | GND
VDD | VDD — 5V | 5V
GND | NRST — 3V3 | 3V3
ADC C1 | C0 ADC USB — OSC H0 | H1 OSC
ADC MIC SPI2 MOSI C3 | C2 ADC SPI2 MISO — OSC RTC C14 | C15 OSC RTC
ADC PWM UART4 RX A1 | A0 ADC BTN1 PWM UART4 TX — PWM E6 | C13
ADC PWM USART2 RX A3 | A2 ADC PWM USART2 TX — E4 | E5 PWM
ADC DAC MEMS SPI1 SCK 3.3v A5 | A4 3.3v ADC DAC AUDIO — E2 | E3 MEMS
ADC MEMS PWM SPI1 MOSI A7 | A6 ADC MEMS PWM SPI1 MISO — MEMS E0 | E1 MEMS
ADC C5 | C4 ADC — I2C1 SCL PWM B8 | B9 AUDIO I2C1 SDA PWM
ADC PWM B1 | B0 ADC PWM — BOOT0 | VDD
GND | B2 BOOT1 — AUDIO I2C1 SCL PWM USART1 TX B6 | B7 I2C1 SDA PWM USART1 RX
E7 | E8 PWM — PWM SPI1 MISO SPI3 MISO B4 | B5 PWM SPI1 MOSI SPI3 MOSI
PWM E9 | E10 PWM — D7 | B3 PWM SPI3 SCK SPI1 SCK
PWM E11 | E12 PWM — USART2 TX D5 | D6 USART2 RX
PWM E13 | E14 PWM — D3 | D4 AUDIO
E15 | B10 MIC I2C2 SCL PWM SPI2 SCK USART3 TX — D1 | D2 UART5 RX
I2C2 SDA PWM USART3 RX B11 | B12 — AUDIO SPI3 MOSI UART5 TX C12 | D0
PWM SPI2 SCK B13 | B14 PWM SPI2 MISO AUDIO SPI3 SCK UART4 TX USART3 TX C10 | C11 SPI3 MISO USART3 RX UART4 RX
PWM SPI2 MOSI B15 | D8 USART3 TX — A14 | A15
USART3 RX D9 | D10 — USB PWM USART1 RX A10 | A13
D11 | D12 LED2 PWM — I2C3 SCL PWM A8 | A9 USB PWM USART1 TX
LED1 PWM D13 | D14 LED3 PWM — PWM C8 | C9 I2C3 SDA PWM
LED4 PWM D15 | NC — PWM USART6 TX C6 | C7 AUDIO PWM USART6 RX
GND | GND — GND | GND
SWD
ST-LINK
DISCOVERY
MB997B
www.st.com/stm32f4-discovery
User

This pictures represents the external wiring of the board that can be used to connector devices to the board. There are mainly two type sof pins (in white):

- Pins with a letter and number, like $C5$, are connected to GPIO, $GPIOC5$ in our example.
- Power pins named GNG, $5V$ and $3V3$ provides ground and powering.

For the moment, we are not concerned by colored names.
So to connect and use the device to the board, we have to:

1. Plug the device on the breadboard (avoid to connect different pins on the same connected holes).
2. Select which pins of the STM32F4 board will be used to control the device.
3. Using straps (represented below), connect the STM32F4 pins on the breadboard.
4. Add jumper wires to power (and ground) the device.
5. Develop and test the code to drive the device.

Warning 1: ensure, when you connect a device to the STM32F4 board, that the pin is compatible with the characteristics of the pin: an overvoltage may break the board! For example, do not connect a device producing a voltage of 5V to a pine supporting only 3.3V.

Warning 2: the STM32F4 board is able to power on devices with a limited current requirement (like LED or push buttons). For more demaning actuators (typically motors or servomotors), an external power will be required. Connecting a device demanding too much power may also break the board.

2.2 Exercises on LED

Sources: 🗀 `src/ex8.c`

Connect a LED to the board according to the map below (picture from Arduino but works in the same way with STM32F4) on Fig. 1. This wiring is designed for an Arduinon but it is easy to adapt to your STM32F4 board (you have to choose the output pin).

Observe that:

- the LED requires a resistor of 330Ω (otherwise you will break the LED).

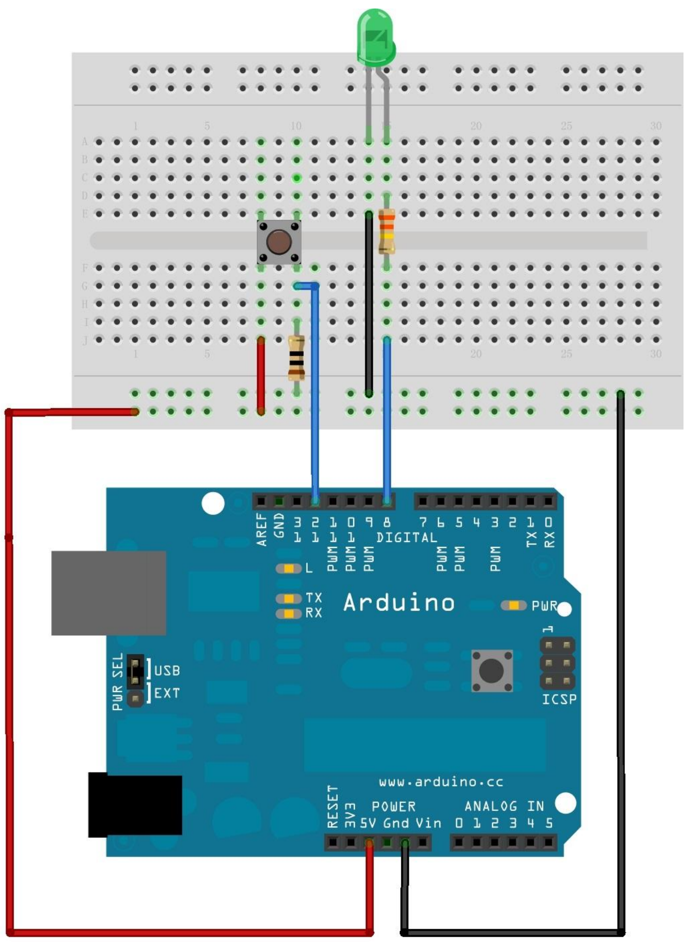
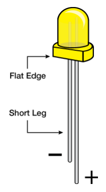
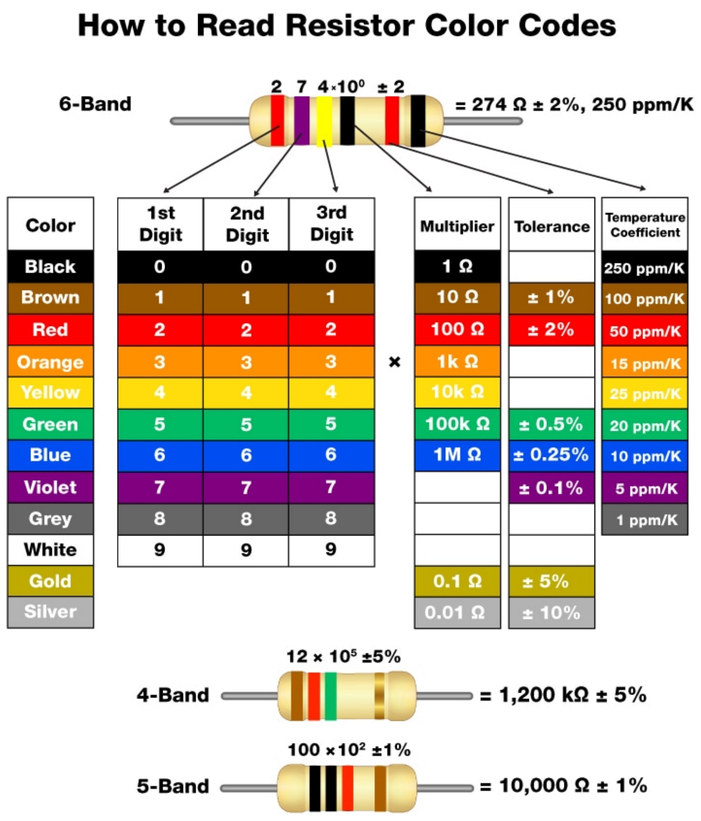


Figure 1: Connecting LEDs and buttons

- the LED, as any diod, is oriented: the long leg must be connected on $+$ and the short leg on $-$.
- LED short leg is connected to GND,
- LED long leg is connected to the resistor that is connected to a GPIO pin.



The code color on the resistor can be read with:


How to Read Resistor Color Codes
6-Band
2 7 4×10⁰ ± 2
= 274 Ω ± 2%, 250 ppm/K
×
4-Band
12 × 10⁵ ±5%
= 1,200 kΩ ± 5%
5-Band
100 ×10² ±1%
= 10,000 Ω ± 1%

Color	1st Digit	2nd Digit	3rd Digit	Multiplier	Tolerance	Temperature Coefficient
Black	0	0	0	1 Ω		250 ppm/K
Brown	1	1	1	10 Ω	± 1%	100 ppm/K
Red	2	2	2	100 Ω	± 2%	50 ppm/K
Orange	3	3	3	1k Ω		15 ppm/K
Yellow	4	4	4	10k Ω		25 ppm/K
Green	5	5	5	100k Ω	± 0.5%	20 ppm/K
Blue	6	6	6	1M Ω	± 0.25%	10 ppm/K
Violet	7	7	7		± 0.1%	5 ppm/K
Grey	8	8	8			1 ppm/K
White	9	9	9			
Gold				0.1 Ω	± 5%	
Silver				0.01 Ω	± 10%	

Exercice: connect a LED to the board and make it blink (source 🗀 `src/ex8.c`).

Exercice: connect 6 LEDs to the board and perform an animation: in turn (period 500ms), switch on more and more LEDs. When all LEDs are lighted on, invert the animation: more and more LEDs are swicthed off (source 🗀 `src/ex9.c`).

2.3 Exercices on Push Button

The wiring of the push button is displayed in Fig 1. Notice the use of the resistor of $10K\Omega$. Such a wiring will make thez button to work in inverse logic:

- When the button is release, the input level is high.
- When the button is pushed, the input level is low.

Connect also the LED according to Fig 1.

Exercise: write an application that, when the button is pushed, lights on the LED (source: 🗀`src/ex10.c`).

3 Synthesis Exercise (optional)

Source: 🗀`src/ex11.c`

Wire 5 LEDs one next to each other and two buttons: one for left move and the other for right move. When the application starts, only the LED at the center is lighted on.

When the left button is clicked, the lighted LED moves to the left. When the right button is clicked, the lighted LED moves to the right. When the left, respectively the left, border is reached, a left, rescpectivelty a right, click has no effect on the lighted LED that remains the same.

Warning: do not forget the resistors!