An interrupt is a signal that can interrupt the normal execution flow to take care of an immediate event.

In the project we make great use of interrupts. Arduino Leonardo features many different types of interrupts, but we mostly use only 2 types: external interrupts and timed interrupts.

The external interrupts are associated to a specific pin and there are only 5 of them on the Leonardo board, associated with pins 0, 1, 2, 3 and 7. The interrupts can be triggered in one of these different cases:

* When the pin is LOW (LOW)
* Whenever the pin changes value (CHANGE)
* When the pin goes from LOW to HIGH (RISING)
* When the pin goes from HIGH to LOW (FALLING)

We use external interrupts on pin 0, 1, 2, 3 to handle button presses. The two black buttons (pins 0 and 1), used to modify values, are associated with a FALLING interrupt because they are active LOW, the red button (pin 2), on the contrary, uses the RISING interrupt. The DPDT switch (pin 3) uses a CHANGE interrupt, because it is associated with a flag that will be toggled every time it is switched.

EICRA |= (1<<ISC31) | (1<<ISC21) | ((1<<ISC11)|(1<<ISC10)) | (1<<ISC00)

With this command we set the type of interrupt on the register EICRA, which controls interrupts on the aforementioned pins. Writing “10” on ISC3n and ISC2n we are setting the FALLING interrupt, writing “11” on ISC1n we set the RISING interrupt and finally writing “01” on ISC0n we are setting the change interrupt.

EIMSK |= (1<<INT0) | (1<<INT1) | (1<<INT2)|(1<<INT3)

This instruction tells the MCU that we want to enable interrupts on pins 3, 2, 0 and 1 respectively.

We can now write Interrupt Service Routines (ISR) using this syntax (example for interrupt on pin 3):

**ISR(INT0\_vect){...}**

In fact, the Arduino library has its own functions to deal with interrupts, that summarize all the previous instructions:

attachInterrupt(digitalPinToInterrupt(0),BB1Handler,FALLING);

attachInterrupt(digitalPinToInterrupt(1),BB2Handler,FALLING);

attachInterrupt(digitalPinToInterrupt(2),BRHandler,RISING);

attachInterrupt(digitalPinToInterrupt(3),BLHandler,CHANGE);

These instructions are in the setup() function of the source code.

The other type of interrupts are timed interrupts. These interrupts are triggered when the associated Timer fulfills a certain condition:

* The timer has overflown
* The timer reached a specified value
* An external event happened and the timer returns its value

We are using Timer1, a 16 bit timer, to trigger an interrupt every 500ms. We are using a library called TimerOne, but by reading the source we see that the library uses the following register settings:

TCCR1B |= (1<<WGM13) | (1<<CS12)

With this instruction we are setting the operating mode of the Timer as “PWM, Phase and Frequency correct” and the relevant thing of this mode is that we can set programmatically the upper limit of the timer, instead of the default 65535 allowed by the 16bits. The second part of the instruction sets the prescaler to 256, which will slow down the timer ticks.

The register that holds the upper limit is ICR1, which will be set at 31250 (it is a 16bit register), according to the following formula:

Lastly, we set Timer1 to trigger an interrupt on overflow with:

TIMSK1 |= (1<<TOIE1)

We can now write the ISR using this syntax:

ISR(TIMER1\_OVF\_vect){ . . . }

Using the library we only need to call these 2 functions:

Timer1.initialize(500000);

Timer1.attachInterrupt(displayTime);

The first one sets the period of the interrupt at 500000 microseconds, or 0.5 seconds. The second function sets the ISR associated with the interrupt.

<https://www.teachmemicro.com/arduino-timer-interrupt-tutorial/>

<https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/>

<http://www.gammon.com.au/interrupts>

<https://playground.arduino.cc/code/timer1>