

## LANGUAGE

### FUNCTIONS

### VARIABLES

### STRUCTURE

## LIBRARIES

## GLOSSARY

The Arduino Reference text is licensed under a [Creative Commons Attribution-Share Alike 3.0 License](#).

Find anything that can be improved? [Suggest corrections and new documentation via GitHub](#).

Doubts on how to use Github? Learn everything you need to know in [this tutorial](#).

Last Update: 11/15/2017

[EDIT THIS PAGE](#)

Reference > Language > Functions > External interrupts > Attachinterrupt

# attachInterrupt()

[External Interrupts]

## Description

### Digital Pins With Interrupts

The first parameter to `attachInterrupt` is an interrupt number. Normally you should use `digitalPinToInterrupt(pin)` to translate the actual digital pin to the specific interrupt number. For example, if you connect to pin 3, use `digitalPinToInterrupt(3)` as the first parameter to `attachInterrupt`.

| BOARD                             | DIGITAL PINS USABLE FOR INTERRUPTS   |
|-----------------------------------|--|
| Uno, Nano, Mini, other 328-based  | 2, 3   |
| Mega, Mega2560, MegaADK           | 2, 3, 18, 19, 20, 21   |
| Micro, Leonardo, other 32u4-based | 0, 1, 2, 3, 7  |
| Zero                              | all digital pins, except 4   |
| MKR1000 Rev.1                     | 0, 1, 4, 5, 6, 7, 8, 9, A1, A2   |
| Due                               | all digital pins   |
| 101                               | all digital pins (Only pins 2, 5, 7, 8, 10, 11, 12, 13 work with <b>CHANGE</b> ) |

## Notes and Warnings

### Note

Inside the attached function, `delay()` won't work and the value returned by `millis()` will not increment. Serial data received while in the function may be lost. You should declare as volatile any variables that you modify within the attached function. See the section on ISRs below for more information.

## Using Interrupts

[SIGN IN](#) [HOME](#)[EDUCATION](#)

program to do anything else, because the program would need to constantly poll the sensor lines for the encoder, in order to catch pulses when they occurred. Other sensors have a similar interface dynamic too, such as trying to read a sound sensor that is trying to catch a click, or an infrared slot sensor (photo-interrupter) trying to catch a coin drop. In all of these situations, using an interrupt can free the microcontroller to get some other work done while not missing the input.

## About Interrupt Service Routines

ISRs are special kinds of functions that have some unique limitations most other functions do not have. An ISR cannot have any parameters, and they shouldn't return anything.

Generally, an ISR should be as short and fast as possible. If your sketch uses multiple ISRs, only one can run at a time, other interrupts will be executed after the current one finishes in an order that depends on the priority they have. `millis()` relies on interrupts to count, so it will never increment inside an ISR. Since `delay()` requires interrupts to work, it will not work if called inside an ISR. `micros()` works initially, but will start behaving erratically after 1-2 ms. `delayMicroseconds()` does not use any counter, so it will work as normal.

Typically global variables are used to pass data between an ISR and the main program. To make sure variables shared between an ISR and the main program are updated correctly, declare them as **`volatile`**.

For more information on interrupts, see [Nick Gammon's notes](#).

### Syntax

```
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);  
(recommended)  
attachInterrupt(interrupt, ISR, mode); (not recommended)  
attachInterrupt(pin, ISR, mode); (not recommended Arduino Due, Zero,  
MKR1000, 101 only)
```

### Parameters

**`interrupt`**: the number of the interrupt (**`int`**)  
**`pin`**: the pin number (*Arduino Due, Zero, MKR1000 only*)  
**`ISR`**: the ISR to call when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.  
**`mode`**: defines when the interrupt should be triggered. Four constants are predefined as valid values:

- **`LOW`** to trigger the interrupt whenever the pin is low,

[SIGN IN](#)[HOME](#)[EDUCATION](#)

- **HIGH** to trigger the interrupt whenever the pin is high.

## Returns

Nothing

## Example Code

```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
  digitalWrite(ledPin, state);
}

void blink() {
  state = !state;
}
```

## Interrupt Numbers

Normally you should use `digitalPinToInterrupt(pin)`, rather than place an interrupt number directly into your sketch. The specific pins with interrupts, and their mapping to interrupt number varies on each type of board. Direct use of interrupt numbers may seem simple, but it can cause compatibility trouble when your sketch is run on a different board.

However, older sketches often have direct interrupt numbers. Often number 0 (for digital pin 2) or number 1 (for digital pin 3) were used. The table below shows the available interrupt pins on various boards.

Note that in the table below, the interrupt numbers refer to the number to be passed to `attachInterrupt()`. For historical reasons, this numbering does not always correspond directly to the interrupt numbering on the atmega chip (e.g. `int.0` corresponds to `INT4` on the Atmega2560 chip).

| BOARD            | INT.0 | INT.1 | INT.2 | INT.3 | INT.4 | INT.5 |
|------------------|-------|-------|-------|-------|-------|-------|
| Uno,<br>Ethernet | 2     | 3     |       |       |       |       |
| Mega2560         | 2     | 3     | 21    | 20    | 19    | 18    |

[SIGN IN](#)[HOME](#)[EDUCATION](#)[MICRO](#)

For Due, Zero, MKR1000 and 101 boards the **interrupt number** = pin number.

undefined

LANGUAGE [detachInterrupt\(\)](#)

## NEWSLETTER

ENTER YOUR EMAIL TO SIGN UP

SUBSCRIBE

[Copyright Notice](#)

[Contact Us](#)

[About Us](#)

[Careers](#)

© 2018 Arduino