This page is also available in **2 other languages** Change language English

LANGUAGE

FUNCTIONS

VARIABLES

STRUCTURE

LIBRARIES

IOT CLOUD API

GLOSSARY

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 Revision: 2023/11/20

Last Build: 2024/05/01

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

BOARD	DIGITAL PINS USABLE FOR INTERRUPTS	NOTES
Uno Rev3, Nano, Mini, other 328-based	2, 3	
UNO R4 Minima, UNO R4 WiFi	2, 3	
Uno WiFi Rev2, Nano Every	All digital pins	
Mega, Mega2560, MegaADK	2, 3, 18, 19, 20, 21	(**pins 20 & 21** are nc available to use for interrupts while the are used for I2C communication; the also have external pull-ups that cannot b

Help

Micro, Leonardo	0, 1, 2, 3, 7	
Zero	0-3, 5-13, A0-A5	Pin 4 cannot be use as an interrupt.
MKR Family boards	0, 1, 4, 5, 6, 7, 8, 9, A1, A2	
Nano 33 IoT	2, 3, 9, 10, 11, 13, A1, A5, A7	
Nano 33 BLE, Nano 33 BLE Sense (rev 1 & 2)	all pins	
Nano RP2040 Connect	0-13, A0-A5	
Nano ESP32	all pins	
GIGA R1 WiFi	all pins	
Due	all digital pins	
101	all digital pins	(Only pins 2, 5, 7, 8, 10, 11, 12, 13 work with **CHANGE**))

Notes and Warnings

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

Using Interrupts

Interrupts are useful for making things happen automatically in microcontroller programs and can help solve timing problems. Good tas for using an interrupt may include reading a rotary encoder, or monitori user input.

If you wanted to ensure that a program always caugh rom a rotary encoder, so that it never misses a pulse, it woul ery tric

Help

when they occurred. Other sensors have a similar interface dynamic toc such as trying to read a sound sensor that is trying to catch a click, or ar infrared slot sensor (photo-interrupter) trying to catch a coin drop. In all these situations, using an interrupt can free the microcontroller to get s other work done while not missing the input.

About Interrupt Service Routines

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

Generally, an ISR should be as short and fast as possible. If your sketch u multiple ISRs, only one can run at a time, other interrupts will be execut after the current one finishes in an order that depends on the priority th have. `millis()` relies on interrupts to count, so it will never increment in an ISR. Since `delay()` requires interrupts to work, it will not work if callec inside an ISR. `micros()` works initially but will start behaving erratically a 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 m 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. Additionally, this syntax only works on Arduino SAMD Boards, Uno WiFi Rev2, Due, and 1(

Parameters

`interrupt`: the number of the interrupt. Allowed data types: `int`.
`pin`: the Arduino pin number.
`ISR`: the ISR to call when the interrupt occurs; this function must take no

Help

`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,
- **CHANGE** to trigger the interrupt whenever the pin changes value
- **RISING** to trigger when the pin goes from low to high,
- **FALLING** for when the pin goes from high to low.

The Due, Zero and MKR1000 boards allow also:

- **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 a interrupt number directly into your sketch. The specific pins with interru and their mapping to interrupt number varies for each type of board. Di use of interrupt numbers may seem simple, but it can cause compatibili trouble when your sketch runs on a different board.

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

Help

not always correspond directly to the interrupt numbering on the ATme chip (e.g. int.0 corresponds to INT4 on the ATmega2560 chip).

BOARD	INT.0	INT.1	INT.2	INT.3	INT.4	INT
Uno, Ethernet	2	3				
Mega2560	2	3	21	20	19	18
32u4 based (e.g Leonardo, Micro)	3	2	0	1	7	

For Uno WiFi Rev2, Due, Zero, MKR Family and 101 boards the **interrup number = pin number**.

See also