Interrupts

Kate Bruns, Andrew Calhoun, Maddie Horvat, and Nick Nusgart
Friday, September 21, 2018 1:00 PM
Physics 398 DLP

What is an interrupt?

An interrupt is a signal to the processor that there is something important to pay attention to.

They were originally used in hardware to replace polling loops so that the CPU could be used more efficiently.

Interrupts are used to signal the processor

Many kinds of hardware will signal interrupts upon completion of some task (eg: hard drive finishes writing data, tape finishes rewinding).

Interrupts can also be used to execute some code at regular intervals.

Arduino Examples:

- Monitor some pin
- Monitor SPI or I2C transactions
- Monitor USART transmissions (have characters / ready to send)

How it works

attachInterrupt(#, ISR, mode)

- # → Use digitalPinToInterrupt in most cases
- ISR: the handler - void and takes no parameters: *must be **fast***
- mode: when should interrupt trigger

noInterrupts() and interrupts()

Most processors (including Arduino):

1) Complete current instruction
2) Clear Interrupt Flag
3) Call applicable handler
4) Handler executes
5) Set Interrupt flag
6) Return to next instruction

Types of Interrupts (for Arduino)

Triggered by some piece of hardware

Can be used to be “notified” to changes

Digital Pins 2, 3, 18, 19, 20, 21

Timer

SPI/I2C

No software interrupts for Arduino

14.1 Interrupt Vectors in ATmega640/1280/1281/2560/2561

Table 14-1. Reset and Interrupt Vectors

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	PCINT0	Pin Change Interrupt Request 0
11	$0014	PCINT1	Pin Change Interrupt Request 1
12	$0016(3)	PCINT2	Pin Change Interrupt Request 2
13	$0018	WDT	Watchdog Time-out Interrupt
14	$001A	TIMER2 COMPA	Timer/Counter2 Compare Match A
15	$001C	TIMER2 COMPB	Timer/Counter2 Compare Match B
16	$001E	TIMER2 OVF	Timer/Counter2 Overflow
17	$0020	TIMER1 CAPT	Timer/Counter1 Capture Event
18	$0022	TIMER1 COMPA	Timer/Counter1 Compare Match A
19	$0024	TIMER1 COMPB	Timer/Counter1 Compare Match B
20	$0026	TIMER1 COMPC	Timer/Counter1 Compare Match C
21	$0028	TIMER1 OVF	Timer/Counter1 Overflow
22	$002A	TIMER0 COMPA	Timer/Counter0 Compare Match A
23	$002C	TIMER0 COMPB	Timer/Counter0 Compare match B
24	$002E	TIMER0 OVF	Timer/Counter0 Overflow
25	$0030	SPI, STC	SPI Serial Transfer Complete
26	$0032	USART0 RX	USART0 Rx Complete
27	$0034	USART0 UDRE	USART0 Data Register Empty
28	$0036	USART0 TX	USART0 Tx Complete
29	$0038	ANALOG COMP	Analog Comparator
30	$003A	ADC	ADC Conversion Complete

Troubleshooting

If it doesn’t run quickly enough, you may encounter problems, as interrupts are very time sensitive. Target the handler

While an interrupt is running, no other interrupt can run.

Interrupts can cause race conditions -- situations where updates to a variable can be lost by other code being interrupted, or where you start to wait for something after it happens.

Summary

An interrupt is a signal to the processor that there is something important to pay attention to.

Interrupts allow your system to free up the loop() function and perform time-sensitive tasks.

The kinds of interrupts we'll be dealing with will involve hardware.

Most processors respond similarly to interrupts.