constants

[Constants]

Description

Constants are predefined expressions in the Arduino language. They are used to make the programs easier to read. We classify constants in groups:

Defining Logical Levels: true and false (Boolean Constants)

There are two constants used to represent truth and falsity in the Arduino language: true, and false.

false

false is the easier of the two to define. false is defined as 0 (zero).

true

true is often said to be defined as 1, which is correct, but true has a wider definition. Any integer which is non-zero is true, in a Boolean sense. So -1, 2 and -200 are all defined as true, too, in a Boolean sense.

Note that the true and false constants are typed in lowercase unlike HIGH, LOW, INPUT, and OUTPUT.

Defining Pin Levels: HIGH and LOW

When reading or writing to a digital pin there are only two possible values a pin can take/be-set-to: HIGH and LOW.

HIGH

The meaning of HIGH (in reference to a pin) is somewhat different depending on whether a pin is set to an INPUT or OUTPUT. When a pin is configured as an INPUT with [pinMode()](https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode), and read with [digitalRead()](https://www.arduino.cc/reference/en/language/functions/digital-io/digitalread), the Arduino (ATmega) will report HIGH if:

* a voltage greater than 3.0V is present at the pin (5V boards)
* a voltage greater than 2.0V volts is present at the pin (3.3V boards)

A pin may also be configured as an INPUT with [pinMode()](https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode), and subsequently made HIGH with [digitalWrite()](https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite). This will enable the internal 20K pullup resistors, which will *pull up* the input pin to a HIGH reading unless it is pulled LOW by external circuitry. This can done alternatively by passing INPUT\_PULLUP as argument to the [pinMode()](https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode) funtion, as explained in more detail in the section "Defining Digital Pins modes: INPUT, INPUT\_PULLUP, and OUTPUT" further below.

When a pin is configured to OUTPUT with [pinMode()](https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode), and set to HIGH with [digitalWrite()](https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite), the pin is at:

* 5 volts (5V boards)
* 3.3 volts (3.3V boards)

In this state it can source current, e.g. light an LED that is connected through a series resistor to ground.

LOW

The meaning of LOW also has a different meaning depending on whether a pin is set to INPUT or OUTPUT. When a pin is configured as an INPUT with [pinMode()](https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode), and read with [digitalRead()](https://www.arduino.cc/reference/en/language/functions/digital-io/digitalread), the Arduino (ATmega) will report LOW if:

* a voltage less than 1.5V is present at the pin (5V boards)
* a voltage less than 1.0V (Approx) is present at the pin (3.3V boards)

When a pin is configured to OUTPUT with [pinMode()](https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode), and set to LOW with [digitalWrite()](https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite), the pin is at 0 volts (both 5V and 3.3V boards). In this state it can sink current, e.g. light an LED that is connected through a series resistor to +5 volts (or +3.3 volts).

Defining Digital Pins modes: INPUT, INPUT\_PULLUP, and OUTPUT

Digital pins can be used as INPUT, INPUT\_PULLUP, or OUTPUT. Changing a pin with [pinMode()](https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode) changes the electrical behavior of the pin.

Pins Configured as INPUT

Arduino (ATmega) pins configured as INPUT with [pinMode()](https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode) are said to be in a *high-impedance* state. Pins configured as INPUT make extremely small demands on the circuit that they are sampling, equivalent to a series resistor of 100 Megohms in front of the pin. This makes them useful for reading a sensor.

If you have your pin configured as an INPUT, and are reading a switch, when the switch is in the open state the input pin will be "floating", resulting in unpredictable results. In order to assure a proper reading when the switch is open, a pull-up or pull-down resistor must be used. The purpose of this resistor is to pull the pin to a known state when the switch is open. A 10 K ohm resistor is usually chosen, as it is a low enough value to reliably prevent a floating input, and at the same time a high enough value to not draw too much current when the switch is closed. See the [Digital Read Serial](http://arduino.cc/en/Tutorial/DigitalReadSerial) tutorial for more information.

If a pull-down resistor is used, the input pin will be LOW when the switch is open and HIGH when the switch is closed.

If a pull-up resistor is used, the input pin will be HIGH when the switch is open and LOW when the switch is closed.

Pins Configured as INPUT\_PULLUP

The ATmega microcontroller on the Arduino has internal pull-up resistors (resistors that connect to power internally) that you can access. If you prefer to use these instead of external pull-up resistors, you can use the INPUT\_PULLUP argument in [pinMode()](https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode).

See the [Input Pullup Serial](http://arduino.cc/en/Tutorial/InputPullupSerial) tutorial for an example of this in use.

Pins configured as inputs with either INPUT or INPUT\_PULLUP can be damaged or destroyed if they are connected to voltages below ground (negative voltages) or above the positive power rail (5V or 3V).

Pins Configured as OUTPUT

Pins configured as OUTPUT with [pinMode()](https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode) are said to be in a *low-impedance* state. This means that they can provide a substantial amount of current to other circuits. ATmega pins can source (provide current) or sink (absorb current) up to 40 mA (milliamps) of current to other devices/circuits. This makes them useful for powering LEDs because LEDs typically use less than 40 mA. Loads greater than 40 mA (e.g. motors) will require a transistor or other interface circuitry.

Pins configured as outputs can be damaged or destroyed if they are connected to either the ground or positive power rails.

Defining built-ins: LED\_BUILTIN

Most Arduino boards have a pin connected to an on-board LED in series with a resistor. The constant LED\_BUILTIN is the number of the pin to which the on-board LED is connected. Most boards have this LED connected to digital pin 13.