analogWriteResolution()

[Zero, Due & MKR Family]

Description

analogWriteResolution() is an extension of the Analog API for the Arduino Due.

analogWriteResolution() sets the resolution of the analogWrite() function. It defaults to 8 bits (values between 0-255) for backward compatibility with AVR based boards.

The **Due** has the following hardware capabilities:

* 12 pins which default to 8-bit PWM, like the AVR-based boards. These can be changed to 12-bit resolution.
* 2 pins with 12-bit DAC (Digital-to-Analog Converter)

By setting the write resolution to 12, you can use analogWrite() with values between 0 and 4095 to exploit the full DAC resolution or to set the PWM signal without rolling over.

The **Zero** has the following hardware capabilities:

* 10 pins which default to 8-bit PWM, like the AVR-based boards. These can be changed to 12-bit resolution.
* 1 pin with 10-bit DAC (Digital-to-Analog Converter).

By setting the write resolution to 10, you can use analogWrite() with values between 0 and 1023 to exploit the full DAC resolution

The **MKR Family** of boards has the following hardware capabilities:

* 4 pins which default to 8-bit PWM, like the AVR-based boards. These can be changed from 8 (default) to 12-bit resolution.
* 1 pin with 10-bit DAC (Digital-to-Analog Converter)

By setting the write resolution to 12 bits, you can use analogWrite() with values between 0 and 4095 for PWM signals; set 10 bit on the DAC pin to exploit the full DAC resolution of 1024 values.

Syntax

analogWriteResolution(bits)

Parameters

bits: determines the resolution (in bits) of the values used in the analogWrite() function. The value can range from 1 to 32. If you choose a resolution higher or lower than your board’s hardware capabilities, the value used in analogWrite() will be either truncated if it’s too high or padded with zeros if it’s too low. See the note below for details.

Returns

Nothing

Example Code

Explain Code

void setup() {

// open a serial connection

Serial.begin(9600);

// make our digital pin an output

pinMode(11, OUTPUT);

pinMode(12, OUTPUT);

pinMode(13, OUTPUT);

}

void loop() {

// read the input on A0 and map it to a PWM pin

// with an attached LED

int sensorVal = analogRead(A0);

Serial.print("Analog Read) : ");

Serial.print(sensorVal);

// the default PWM resolution

analogWriteResolution(8);

analogWrite(11, map(sensorVal, 0, 1023, 0, 255));

Serial.print(" , 8-bit PWM value : ");

Serial.print(map(sensorVal, 0, 1023, 0, 255));

// change the PWM resolution to 12 bits

// the full 12 bit resolution is only supported

// on the Due

analogWriteResolution(12);

analogWrite(12, map(sensorVal, 0, 1023, 0, 4095));

Serial.print(" , 12-bit PWM value : ");

Serial.print(map(sensorVal, 0, 1023, 0, 4095));

// change the PWM resolution to 4 bits

analogWriteResolution(4);

analogWrite(13, map(sensorVal, 0, 1023, 0, 15));

Serial.print(", 4-bit PWM value : ");

Serial.println(map(sensorVal, 0, 1023, 0, 15));

delay(5);

}

Notes and Warnings

If you set the analogWriteResolution() value to a value higher than your board’s capabilities, the Arduino will discard the extra bits. For example: using the Due with analogWriteResolution(16) on a 12-bit DAC pin, only the first 12 bits of the values passed to analogWrite() will be used and the last 4 bits will be discarded.

If you set the analogWriteResolution() value to a value lower than your board’s capabilities, the missing bits will be **padded with zeros** to fill the hardware required size. For example: using the Due with analogWriteResolution(8) on a 12-bit DAC pin, the Arduino will add 4 zero bits to the 8-bit value used in analogWrite() to obtain the 12 bits required.