Teensy_sPWM Reference

Version 0.1 (release date TBD)

Overview

This is a library for emulating pulse-width modulation on Teensy 2.0 and similar devices.

How to use

Using this library requires including *Teensy_sPWM.h* in your source file(s). Before doing anything else, you must call **PWM_init(int Hz)**, specifying a value for **Hz**. This will set up the library and all necessary subsystems, with pulse-width set by the frequency you specify in **Hz**.

You must have a program loop, in which **PWM_loop()** must be called regularly to generate the pulses. It should be called as often as possible, such as in an idle loop.

Functions

int PWM_init(unsigned int Hz, char options)

Initializes Arduino library so functions like **millis()** and **micros()** can be used. Initializes variables that associate pin letters and numbers with their respective internal representations. Sets all pins to 0% PWM. Sets the pulse width to the corresponding length as specified by the **Hz** parameter. The parameter **options** is used to set various internal operating modes. Valid options are:

MAKE_UP_LOST_TIME – When enabled the system will attempt to make up for delays to power a pin from the last cycle during the next one, and will subtract excess time spent in a previous powered cycle from the next one.

void PWM_loop(void)

This function is responsible for updating pin statuses and pulse regulation. It should be called very often, ideally from a loop that runs infinitely (unless you wish to disable PWM).

int set_pin_PWM(char port, char pin, uint8_t pwmPercent)

Sets the PWM level for a given pin. **port** should be a letter between A and F representing the port letters on the Teensy. **pin** should be an integer between 0 and 7 representing the respective pin number on a given port. **pwmPercent** is the level of PWM and should be an integer between 0 and 100. This function returns 1 on success and 0 on failure. The following line would set pin B3 to PWM level of 75 percent:

```
set_pin_PWM('B', 3, 75);
```

int set_pin_PWM_normalized(char port, char pin, float normPwm)

This function is identical to set_pin_PWM, except for the parameter **normPwm**. You can pass a floating point value from 0.0 to 1.0 in **normPwm**, representing values from 0 to 100 percent. This function returns 1 on success and 0 on failure.

int set_abstract_pin_PWM(uint8_t pin, uint8_t pwmPercent)

Sets the PWM level for a given abstractly numbered pin. **pin** should be an integer between 0 and the number of pins on your given device, representing the respective pin number of this functions numbering convention (0 starts at upper left and goes down, continues at bottom right upwards, and then inward to the left. TODO: link or include a diagram showing pin numbering for different devices). **pwmPercent** is the level of PWM and should be an integer between 0 and 100. This function returns 1 on success and 0 on failure. The following line would set pin B7 to PWM level of 75 percent, as pin B7 is abstract pin number 4:

```
set_abstract_pin_PWM(4, 75);
```

int set_abstract_pin_PWM_normalized(uint8_t pin, float normPwm)

This function is the same as set_abstract_pin_PWM, except for the parameter **normPwm**. You can pass a floating point value from 0.0 to 1.0 in **normPwm**, representing values from 0 to 100 percent. This function returns 1 on success and 0 on failure.

int set_all_abstract_pins_PWM(uint8_t pwmPercent)

Sets all pins to the PWM percentage passed in **pwmPercent**. **pwmPercent** is the level of PWM and should be an integer between 0 and 100. This function returns 1 on success and 0 on failure.

int set_all_abstract_pins_PWM_normalized(float normPwm)

Sets all pins to the PWM level passed in **normPwm**. You can pass a floating point value from 0.0 to 1.0 in **normPwm**, representing values from 0 to 100 percent. This function returns 1 on success and 0 on failure.

int set_abstract_pin_range_PWM(int start, int end, uint8_t pwmPercent)

Sets all abstract pins in the range starting at **start** and ending at **end** to the PWM level specified in **pwmPercent**. **pwmPercent** is the level of PWM and should be an integer between 0 and 100. This function returns 1 on success and 0 on failure.

int set_abstract_pin_range_PWM_normalized(int start, int end, float normPwm)

Sets all abstract pins in the range starting at **start** and ending at **end** to the PWM level specified in **normPwm**. You can pass a floating point value from 0.0 to 1.0 in **normPwm**, representing values from 0 to 100 percent. This function returns 1 on success and 0 on failure.