/\*

Servo.h - Interrupt driven Servo library for Arduino using 16 bit timers - Version 2

Copyright (c) 2009 Michael Margolis. All right reserved.

This library is free software; you can redistribute it and/or

modify it under the terms of the GNU Lesser General Public

License as published by the Free Software Foundation; either

version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,

but WITHOUT ANY WARRANTY; without even the implied warranty of

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public

License along with this library; if not, write to the Free Software

Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

\*/

/\*

A servo is activated by creating an instance of the Servo class passing

the desired pin to the attach() method.

The servos are pulsed in the background using the value most recently

written using the write() method.

Note that analogWrite of PWM on pins associated with the timer are

disabled when the first servo is attached.

Timers are seized as needed in groups of 12 servos - 24 servos use two

timers, 48 servos will use four.

The sequence used to seize timers is defined in timers.h

The methods are:

Servo - Class for manipulating servo motors connected to Arduino pins.

attach(pin ) - Attaches a servo motor to an I/O pin.

attach(pin, min, max ) - Attaches to a pin setting min and max values in microseconds

default min is 544, max is 2400

write() - Sets the servo angle in degrees. (invalid angle that is valid as pulse in microseconds is treated as microseconds)

writeMicroseconds() - Sets the servo pulse width in microseconds

read() - Gets the last written servo pulse width as an angle between 0 and 180.

readMicroseconds() - Gets the last written servo pulse width in microseconds. (was read\_us() in first release)

attached() - Returns true if there is a servo attached.

detach() - Stops an attached servos from pulsing its I/O pin.

\*/

#ifndef Servo\_h

#define Servo\_h

#include <inttypes.h>

/\*

\* Defines for 16 bit timers used with Servo library

\*

\* If \_useTimerX is defined then TimerX is a 16 bit timer on the current board

\* timer16\_Sequence\_t enumerates the sequence that the timers should be allocated

\* \_Nbr\_16timers indicates how many 16 bit timers are available.

\*/

// Architecture specific include

#if defined(ARDUINO\_ARCH\_AVR)

#include "avr/ServoTimers.h"

#elif defined(ARDUINO\_ARCH\_SAM)

#include "sam/ServoTimers.h"

#elif defined(ARDUINO\_ARCH\_SAMD)

#include "samd/ServoTimers.h"

#elif defined(ARDUINO\_ARCH\_STM32F4)

#include "stm32f4/ServoTimers.h"

#elif defined(ARDUINO\_ARCH\_NRF52)

#include "nrf52/ServoTimers.h"

#elif defined(ARDUINO\_ARCH\_MEGAAVR)

#include "megaavr/ServoTimers.h"

#elif defined(ARDUINO\_ARCH\_MBED)

#include "mbed/ServoTimers.h"

#elif defined(ARDUINO\_ARCH\_RENESAS)

#include "renesas/ServoTimers.h"

#elif defined(ARDUINO\_ARCH\_XMC)

#include "xmc/ServoTimers.h"

#else

#error "This library only supports boards with an AVR, SAM, SAMD, NRF52, STM32F4, Renesas or XMC processor."

#endif

#define Servo\_VERSION 2 // software version of this library

#define MIN\_PULSE\_WIDTH 544 // the shortest pulse sent to a servo

#define MAX\_PULSE\_WIDTH 2400 // the longest pulse sent to a servo

#define DEFAULT\_PULSE\_WIDTH 1500 // default pulse width when servo is attached

#define REFRESH\_INTERVAL 20000 // minimum time to refresh servos in microseconds

#define SERVOS\_PER\_TIMER 12 // the maximum number of servos controlled by one timer

#define MAX\_SERVOS (\_Nbr\_16timers \* SERVOS\_PER\_TIMER)

#define INVALID\_SERVO 255 // flag indicating an invalid servo index

#if !defined(ARDUINO\_ARCH\_STM32F4) && !defined(ARDUINO\_ARCH\_XMC)

typedef struct {

uint8\_t nbr :6 ; // a pin number from 0 to 63

uint8\_t isActive :1 ; // true if this channel is enabled, pin not pulsed if false

} ServoPin\_t ;

typedef struct {

ServoPin\_t Pin;

volatile unsigned int ticks;

} servo\_t;

class Servo

{

public:

Servo();

uint8\_t attach(int pin); // attach the given pin to the next free channel, sets pinMode, returns channel number or INVALID\_SERVO if failure

uint8\_t attach(int pin, int min, int max); // as above but also sets min and max values for writes.

void detach();

void write(int value); // if value is < 200 it's treated as an angle, otherwise as pulse width in microseconds

void writeMicroseconds(int value); // Write pulse width in microseconds

int read(); // returns current pulse width as an angle between 0 and 180 degrees

int readMicroseconds(); // returns current pulse width in microseconds for this servo (was read\_us() in first release)

bool attached(); // return true if this servo is attached, otherwise false

private:

uint8\_t servoIndex; // index into the channel data for this servo

int8\_t min; // minimum is this value times 4 added to MIN\_PULSE\_WIDTH

int8\_t max; // maximum is this value times 4 added to MAX\_PULSE\_WIDTH

};

#endif

#endif