/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* The MIT License

\*

\* Copyright (c) 2010, LeafLabs, LLC.

\*

\* Permission is hereby granted, free of charge, to any person

\* obtaining a copy of this software and associated documentation

\* files (the "Software"), to deal in the Software without

\* restriction, including without limitation the rights to use, copy,

\* modify, merge, publish, distribute, sublicense, and/or sell copies

\* of the Software, and to permit persons to whom the Software is

\* furnished to do so, subject to the following conditions:

\*

\* The above copyright notice and this permission notice shall be

\* included in all copies or substantial portions of the Software.

\*

\* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,

\* EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF

\* MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND

\* NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS

\* BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN

\* ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN

\* CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE

\* SOFTWARE.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* @copyright Copyright (c) 2019-2020 Infineon Technologies AG

\*/

#ifndef \_SERVO\_H\_

#define \_SERVO\_H\_

#include <Arduino.h>

#include "wiring\_analog.h"

/\*

\* Note on Arduino compatibility:

\*

\* In the Arduino implementation, PWM is done "by hand" in the sense

\* that timer channels are hijacked in groups and an ISR is set which

\* toggles Servo::attach()ed pins using digitalWrite().

\*

\* While this scheme allows any pin to drive a servo, it chews up

\* cycles and complicates the programmer's notion of when a particular

\* timer channel will be in use.

\*

\* This implementation only allows Servo instances to attach() to pins

\* that already have PWM unit associated with them, which drives the wave.

\*

\* While the Arduino implementation of attach() returns the affected channel,

\* this one returns the index number of the servo or an INVALID\_SERVO = 255 in

\* case of an error.

\* The attach will check if a pin is already in use and if a pin has a PWM unit on

\* the selected XMC board, otherwise it returns an INVALID\_SERVO.

\* This error handling is different than the original one from Arduino.

\*

\* Depending on the XMC type the number of possible PWM channels vary from 4 to 23

\* and may change with future version of the XMC series.

\*/

// Define the MAX\_PWM\_SERVOS number per XMC type and the allowed PWM pins on the selected XMC board

#if defined(XMC1100\_XMC2GO)

#define MAX\_PWM\_SERVOS 4

#define ALLOWED\_PINS {1, 2, 3, 8,}

#elif defined(XMC1100\_Boot\_Kit)

#define MAX\_PWM\_SERVOS 6

#define ALLOWED\_PINS { 3,4,6,9,10,11 }

#elif defined(XMC1300\_Boot\_Kit)

#define MAX\_PWM\_SERVOS 4

#define ALLOWED\_PINS { 26,31,32,33 }

#elif defined(XMC1400\_Arduino\_Kit)

#define MAX\_PWM\_SERVOS 6

#define ALLOWED\_PINS { 3,4,6,9,10,11 }

#elif defined(XMC4200\_Platform2GO)

#define MAX\_PWM\_SERVOS 7

#define ALLOWED\_PINS { 3,5,6,9,22,23,24 }

#elif defined(XMC4400\_Platform2GO)

#define MAX\_PWM\_SERVOS 15

#define ALLOWED\_PINS { 3,5,6,9,10,14,25,26,27,28,29,30,45,48,67 }

#elif defined(XMC4700\_Relax\_Kit)

#define MAX\_PWM\_SERVOS 23

#define ALLOWED\_PINS { 3,5,6,9,10,11,34,36,37,51,61,62,66,70,76,77,79,80,81,88,89,93,94 }

#else

#error "Not a supported XMC Board"

#endif

#define MIN\_ANGLE 0 // the minimal angle in degree

#define MAX\_ANGLE 180 // the maximal angle in degree

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

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

#define MIN\_PULSE\_CHECK 500 // border with below = angle and above = pulse width

#define REFRESH\_FREQUENCY 50u // the refresh frequency on analog pins

#define REFRESH\_TIME 20.0 // the PWM refresh frequency for the servo motor

#define DUTYCYCLE\_STEPS 65536.0 / REFRESH\_TIME // the number of duty cycle steps during one refresh period

#define ADC\_RESOLUTION 16 // the resolution of the adc during analog write

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

/\*\* Class for interfacing with RC servomotors. \*/

class Servo

{

public:

/\*\*

\* @brief Construct a new Servo instance.

\*

\* The new instance will not be attached to any pin, but only PWM capable pins will run.

\* see pin list above.

\*/

Servo();

/\*\*

\* @brief Associate this instance with a servomotor whose input is

\* connected to pin.

\*

\* If this instance is already attached to a pin, it will be

\* detached before being attached to the new pin.

\* If the pin is not allowed for running PWM or the max number of

\* PWM channels on the XMC board is reached it will return

\* with an INVALID\_SERVO, otherwise with the servoIndex number.

\*

\* @param pin Pin connected to the servo pulse wave input. This

\* pin must be capable of PWM output.

\*

\* @param min If this value is below MIN\_PULSE\_CHECK it will be associated

\* with an angle in degree. Otherwise it will be the minimum

\* pulse width.

\* min as an angle must be between MIN\_ANGLE < angle < MAX\_ANGLE

\* with default as MIN\_ANGLE

\* min as a pulse width must be between MIN\_PULSE\_WIDTH < pwm < MAX\_PULSE\_WIDTH

\* with a default as MIN\_PULSE\_WIDTH

\*

\* @param max If this value is below MIN\_PULSE\_CHECK it will be associated

\* with an angle in degree. Otherwise it will be the maximum

\* pulse width.

\* max as an angle must be between MIN\_ANGLE < angle < MAX\_ANGLE

\* with default as MAX\_ANGLE

\* max as a pulse width must be between MIN\_PULSE\_WIDTH < pwm < MAX\_PULSE\_WIDTH

\* with a default as MAX\_PULSE\_WIDTH

\*

\* @return servoIndex number or INVALID\_SERVO = 255 in case of an error

\*/

uint8\_t attach(uint8\_t pin, uint16\_t min = MIN\_ANGLE, uint16\_t max = MAX\_ANGLE);

/\*\*

\* @brief Stop driving the servo pulse train.

\*

\* If not currently attached to a motor, this function has no effect.

\*

\* @return true if this call did anything, false otherwise.

\*/

void detach();

/\*\*

\* @brief Set the servomotor target angle by recalculating the duty cycle

\* for XMC PWM settings.

\*

\* @param value Target angle, in degrees. If the target angle is

\* outside the range specified at attach(), it

\* will be clamped to lie in that range.

\*

\* @see Servo::attach()

\*/

void write(int value);

/\*\*

\* @brief Set the pulse width, in microseconds by recalculating it for the

\* XMC PWM settings. It also calculates the angle from the pwm value.

\*

\* @param value Pulse width to send to the servomotor, in

\* microseconds. If outside of the range

\* specified at attach() time, it is clamped to

\* lie in that range.

\*

\* @see Servo::attach()

\*/

void writeMicroseconds(int value);

/\*\*

\* returns the current value in degree as an angle between 0 and 189 degrees

\*

\* @see Servo::attach()

\*/

int read() const { return uint16\_t(this->\_deg); }

/\*\*

\* returns the current pwm value in microseconds.

\*

\* @see Servo::attach()

\*/

int readMicroseconds() const { return uint16\_t(this->\_pwm); }

/\*\*

\* @brief Check if this instance is attached to a servo.

\* @return true if this instance is attached to a servo, false otherwise.

\* @see Servo::attachedPin()

\*/

bool attached() const { return this->\_isActive; }

private:

uint16\_t \_minPW; // the initial minPulseWidth, if not set than MIN\_PULSE\_WIDTH

uint16\_t \_maxPW; // the initial maxPulseWidth, if not set than MAX\_PULSE\_WIDTH

int16\_t \_minAngle; // the initial minAngle, if not set than MIN\_ANGLE

int16\_t \_maxAngle; // the initial maxAngle, if not set than MAX\_ANGLE

int16\_t \_pin; // attached arduino pin number

double \_deg; // actual angle in degree

double \_pwm; // actual pwm signal in microseconds

uint8\_t \_isActive; // true if this pin is active, otherwise false

uint8\_t servoIndex; // the actual number of Servos attached to this library

};

#endif /\* \_SERVO\_H\_ \*/