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

\* 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.

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

#if defined(ARDUINO\_ARCH\_XMC)

#include "ServoTimers.h"

uint8\_t \_ServoCount = 1; // internal counter to check if max numbers of servos is reached

static uint8\_t \_allowed[MAX\_PWM\_SERVOS] = ALLOWED\_PINS; // internal array to check allowed pwm pins

static uint8\_t \_servos[MAX\_PWM\_SERVOS]; // static array of used servo pins for checking

/\*\*

\* @brief None blocking wait loop.

\*

\* @param uS microseconds to wait

\*/

static void \_delayUs(unsigned long uS)

{

unsigned long time\_now = micros();

while (micros() < time\_now + uS)

;

}

Servo::Servo()

{

if (\_ServoCount <= MAX\_PWM\_SERVOS )

{

this->servoIndex = \_ServoCount++;

this->\_minAngle = MIN\_ANGLE;

this->\_maxAngle = MAX\_ANGLE;

this->\_minPW = MIN\_PULSE\_WIDTH;

this->\_maxPW = MAX\_PULSE\_WIDTH;

this->\_pin = 0;

this->\_isActive = false;

this->\_pwm = 0;

this->\_deg = 0.0;

}else{

this->servoIndex = INVALID\_SERVO;

}

}

uint8\_t Servo::attach(uint8\_t pin, uint16\_t min, uint16\_t max)

{

if (this->servoIndex <= MAX\_PWM\_SERVOS )

{

// validate selected pin

bool pin\_allowed = false;

for( int i = 0; i < MAX\_PWM\_SERVOS; i++)

{

// check if pin already in use

if ( \_servos[i] == pin)

return INVALID\_SERVO;

// check if selected pin has a pwm unit on the used XMC board

if ( \_allowed[i] == pin)

pin\_allowed = true;

}

// return if pin is not found in allowed pin list

if ( !pin\_allowed )

return INVALID\_SERVO;

// Set min/max values according the input and check for absolute limits

if (min < MIN\_PULSE\_CHECK)

{

this->\_minAngle = constrain(min,MIN\_ANGLE,MAX\_ANGLE);

this->\_minPW = MIN\_PULSE\_WIDTH;

} else {

this->\_minAngle = MIN\_ANGLE; //TODO has to calculated

this->\_minPW = constrain(min,MIN\_PULSE\_WIDTH,MAX\_PULSE\_WIDTH);

}

if (max < MIN\_PULSE\_CHECK)

{

this->\_maxAngle = constrain(max,MIN\_ANGLE,MAX\_ANGLE);

this->\_maxPW = 2 \* MAX\_PULSE\_WIDTH;

} else {

this->\_maxAngle = MAX\_ANGLE; //TODO has to calculated

this->\_maxPW = constrain(max,MIN\_PULSE\_WIDTH,MAX\_PULSE\_WIDTH);

}

this->\_pin = pin;

this->\_isActive = true;

setAnalogWriteFrequency(this->\_pin, REFRESH\_FREQUENCY);

analogWriteResolution(ADC\_RESOLUTION);

}

return this->servoIndex;

}

void Servo::detach()

{

this->servoIndex = \_ServoCount--;

this->\_minAngle = MIN\_ANGLE;

this->\_maxAngle = MAX\_ANGLE;

this->\_minPW = MIN\_PULSE\_WIDTH;

this->\_maxPW = MAX\_PULSE\_WIDTH;

this->\_pin = 0;

this->\_isActive = false;

this->\_pwm = 0;

this->\_deg = 0.0;

}

void Servo::write(int value)

{

if (value < MIN\_PULSE\_CHECK)

{

// angle must be inside the boundaries

double angle = constrain(value, this->\_minAngle, this->\_maxAngle);

double dutyCycle = ( 0.5 + ( angle / MAX\_ANGLE ) \* 2.0 ) \* DUTYCYCLE\_STEPS;

this->\_deg = angle;

this->\_pwm = uint16\_t(dutyCycle);

analogWrite(this->\_pin, uint16\_t(dutyCycle));

\_delayUs(50);

} else {

writeMicroseconds(value);

}

}

void Servo::writeMicroseconds(int value)

{

// value must be inside the boundaries

double pw = constrain(value,this->\_minPW, this->\_maxPW);

double dutyCycle = map(pw, MIN\_PULSE\_WIDTH,MAX\_PULSE\_WIDTH, 0.5 \* DUTYCYCLE\_STEPS, 2.5 \* DUTYCYCLE\_STEPS);

this->\_deg = ( dutyCycle - DUTYCYCLE\_STEPS \* 0.5 ) \* MAX\_ANGLE / ( 2 \* DUTYCYCLE\_STEPS );

this->\_pwm = uint16\_t(dutyCycle);

analogWrite(this->\_pin, uint16\_t(dutyCycle));

\_delayUs(50);

}

#endif