/\*

Copyright (c) 2016 Arduino. 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

\*/

#if defined(ARDUINO\_ARCH\_NRF52)

#include <Arduino.h>

#include <Servo.h>

static servo\_t servos[MAX\_SERVOS]; // static array of servo structures

uint8\_t ServoCount = 0; // the total number of attached servos

uint32\_t group\_pins[3][NRF\_PWM\_CHANNEL\_COUNT]={{NRF\_PWM\_PIN\_NOT\_CONNECTED, NRF\_PWM\_PIN\_NOT\_CONNECTED, NRF\_PWM\_PIN\_NOT\_CONNECTED, NRF\_PWM\_PIN\_NOT\_CONNECTED}, {NRF\_PWM\_PIN\_NOT\_CONNECTED, NRF\_PWM\_PIN\_NOT\_CONNECTED, NRF\_PWM\_PIN\_NOT\_CONNECTED, NRF\_PWM\_PIN\_NOT\_CONNECTED}, {NRF\_PWM\_PIN\_NOT\_CONNECTED, NRF\_PWM\_PIN\_NOT\_CONNECTED, NRF\_PWM\_PIN\_NOT\_CONNECTED, NRF\_PWM\_PIN\_NOT\_CONNECTED}};

static uint16\_t seq\_values[3][NRF\_PWM\_CHANNEL\_COUNT]={{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}};

Servo::Servo()

{

if (ServoCount < MAX\_SERVOS) {

this->servoIndex = ServoCount++; // assign a servo index to this instance

} else {

this->servoIndex = INVALID\_SERVO; // too many servos

}

}

uint8\_t Servo::attach(int pin)

{

return this->attach(pin, 0, 2500);

}

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

{

int servo\_min, servo\_max;

if (this->servoIndex < MAX\_SERVOS) {

pinMode(pin, OUTPUT); // set servo pin to output

servos[this->servoIndex].Pin.nbr = pin;

if(min < servo\_min) min = servo\_min;

if (max > servo\_max) max = servo\_max;

this->min = min;

this->max = max;

servos[this->servoIndex].Pin.isActive = true;

}

return this->servoIndex;

}

void Servo::detach()

{

servos[this->servoIndex].Pin.isActive = false;

}

void Servo::write(int value)

{

if (value < 0)

value = 0;

else if (value > 180)

value = 180;

value = map(value, 0, 180, MIN\_PULSE, MAX\_PULSE);

writeMicroseconds(value);

}

void Servo::writeMicroseconds(int value)

{

uint8\_t channel, instance;

uint8\_t pin = servos[this->servoIndex].Pin.nbr;

//instance of PWM module is MSB - look at VWariant.h

instance=(g\_APinDescription[pin].ulPWMChannel & 0xF0)/16;

//index of PWM channel is LSB - look at VWariant.h

channel=g\_APinDescription[pin].ulPWMChannel & 0x0F;

group\_pins[instance][channel]=g\_APinDescription[pin].ulPin;

NRF\_PWM\_Type \* PWMInstance = instance == 0 ? NRF\_PWM0 : (instance == 1 ? NRF\_PWM1 : NRF\_PWM2);

//configure PWM instance and enable it

seq\_values[instance][channel]= value | 0x8000;

nrf\_pwm\_sequence\_t const seq={

seq\_values[instance],

NRF\_PWM\_VALUES\_LENGTH(seq\_values),

0,

0

};

nrf\_pwm\_pins\_set(PWMInstance, group\_pins[instance]);

nrf\_pwm\_enable(PWMInstance);

nrf\_pwm\_configure(PWMInstance, NRF\_PWM\_CLK\_125kHz, NRF\_PWM\_MODE\_UP, 2500); // 20ms - 50Hz

nrf\_pwm\_decoder\_set(PWMInstance, NRF\_PWM\_LOAD\_INDIVIDUAL, NRF\_PWM\_STEP\_AUTO);

nrf\_pwm\_sequence\_set(PWMInstance, 0, &seq);

nrf\_pwm\_loop\_set(PWMInstance, 0UL);

nrf\_pwm\_task\_trigger(PWMInstance, NRF\_PWM\_TASK\_SEQSTART0);

}

int Servo::read() // return the value as degrees

{

return map(readMicroseconds(), MIN\_PULSE, MAX\_PULSE, 0, 180);

}

int Servo::readMicroseconds()

{

uint8\_t channel, instance;

uint8\_t pin=servos[this->servoIndex].Pin.nbr;

instance=(g\_APinDescription[pin].ulPWMChannel & 0xF0)/16;

channel=g\_APinDescription[pin].ulPWMChannel & 0x0F;

// remove the 16th bit we added before

return seq\_values[instance][channel] & 0x7FFF;

}

bool Servo::attached()

{

return servos[this->servoIndex].Pin.isActive;

}

#endif // ARDUINO\_ARCH\_NRF52