// ---------------------------------------------------------------------------

// Created by Tim Eckel - teckel@leethost.com

// Copyright 2013 License: GNU GPL v3 http://www.gnu.org/licenses/gpl-3.0.html

//

// See "NewTone.h" for purpose, syntax, version history, links, and more.

// ---------------------------------------------------------------------------

#include "NewTone.h"

unsigned long \_nt\_time; // Time note should end.

uint8\_t \_pinMask = 0; // Pin bitmask.

volatile uint8\_t \*\_pinOutput; // Output port register

void NewTone(uint8\_t pin, unsigned long frequency, unsigned long length) {

uint8\_t prescaler = \_BV(CS10); // Try using prescaler 1 first.

unsigned long top = F\_CPU / frequency / 4 - 1; // Calculate the top.

if (top > 65535) { // If not in the range for prescaler 1, use prescaler 256 (61 Hz and lower @ 16 MHz).

prescaler = \_BV(CS12); // Set the 256 prescaler bit.

top = top / 256 - 1; // Calculate the top using prescaler 256.

}

if (length > 0) \_nt\_time = millis() + length; else \_nt\_time = 0xFFFFFFFF; // Set when the note should end, or play "forever".

if (\_pinMask == 0) {

\_pinMask = digitalPinToBitMask(pin); // Get the port register bitmask for the pin.

\_pinOutput = portOutputRegister(digitalPinToPort(pin)); // Get the output port register for the pin.

uint8\_t \*\_pinMode = (uint8\_t \*) portModeRegister(digitalPinToPort(pin)); // Get the port mode register for the pin.

\*\_pinMode |= \_pinMask; // Set the pin to Output mode.

}

ICR1 = top; // Set the top.

if (TCNT1 > top) TCNT1 = top; // Counter over the top, put within range.

TCCR1B = \_BV(WGM13) | prescaler; // Set PWM, phase and frequency corrected (ICR1) and prescaler.

TCCR1A = \_BV(COM1B0);

TIMSK1 |= \_BV(OCIE1A); // Activate the timer interrupt.

}

void noNewTone(uint8\_t pin) {

TIMSK1 &= ~\_BV(OCIE1A); // Remove the timer interrupt.

TCCR1B = \_BV(CS11); // Default clock prescaler of 8.

TCCR1A = \_BV(WGM10); // Set to defaults so PWM can work like normal (PWM, phase corrected, 8bit).

\*\_pinOutput &= ~\_pinMask; // Set pin to LOW.

\_pinMask = 0; // Flag so we know note is no longer playing.

}

ISR(TIMER1\_COMPA\_vect) { // Timer interrupt vector.

if (millis() >= \_nt\_time) noNewTone(); // Check to see if it's time for the note to end.

\*\_pinOutput ^= \_pinMask; // Toggle the pin state.

}