```
/*
 Timer Helpers library.

Devised and written by Nick Gammon.
Date: 21 March 2012
Version: 1.0

Licence: Released for public use.

See: http://www.gammon.com.au/forum/?id=11504

 Example:

 // set up Timer 1
 TCNT1 = 0;         // reset counter
 OCR1A =  999;       // compare A register value (1000 * clock speed)

 // Mode 4: CTC, top = OCR1A
 Timer1::setMode (4, Timer1::PRESCALE_1, Timer1::CLEAR_A_ON_COMPARE);

 TIFR1 |= bit (OCF1A);    // clear interrupt flag
 TIMSK1 = bit (OCIE1A);   // interrupt on Compare A Match

*/

#ifndef _TimerHelpers_h
#define _TimerHelpers_h

#if defined(ARDUINO) && ARDUINO >= 100
  #include "Arduino.h"
#else
  #include "WProgram.h"
#endif

/* ------------------------------------------------------------------
 Timer 0 setup
 ------------------------------------------------------------------ */

namespace Timer0
{
  // TCCR0A, TCCR0B
  const byte Modes [8] [2] =
  {

  { 0,                          0 },            // 0: Normal, top = 0xFF
  { bit (WGM00),                0 },            // 1: PWM, Phase-correct, top = 0xFF
  {               bit (WGM01), 0 },            // 2: CTC, top = OCR0A
  { bit (WGM00) | bit (WGM01), 0 },            // 3: Fast PWM, top = 0xFF
  { 0,                          bit (WGM02) },  // 4: Reserved
  { bit (WGM00),                bit (WGM02) },  // 5: PWM, Phase-correct, top = OCR0A
  {               bit (WGM01), bit (WGM02) },  // 6: Reserved
  { bit (WGM00) | bit (WGM01), bit (WGM02) },  // 7: Fast PWM, top = OCR0A

  };  // end of Timer0::Modes

  // Activation
  // Note: T0 is pin 6, Arduino port: D4
  enum { NO_CLOCK, PRESCALE_1, PRESCALE_8, PRESCALE_64, PRESCALE_256, PRESCALE_1024, T0_FALLING,
T0_RISING };

  // what ports to toggle on timer fire
  enum { NO_PORT = 0,

    // pin 12, Arduino port: D6
    TOGGLE_A_ON_COMPARE  = bit (COM0A0),
```

```
    CLEAR_A_ON_COMPARE   = bit (COM0A1),
    SET_A_ON_COMPARE     = bit (COM0A0) | bit (COM0A1),

    // pin 11, Arduino port: D5
    TOGGLE_B_ON_COMPARE  = bit (COM0B0),
    CLEAR_B_ON_COMPARE   = bit (COM0B1),
    SET_B_ON_COMPARE     = bit (COM0B0) | bit (COM0B1),
  };


  // choose a timer mode, set which clock speed, and which port to toggle
  void setMode (const byte mode, const byte clock, const byte port)
  {
  if (mode < 0 || mode > 7)  // sanity check
    return;

  // reset existing flags
  TCCR0A = 0;
  TCCR0B = 0;

  TCCR0A |= (Modes [mode] [0]) | port;
  TCCR0B |= (Modes [mode] [1]) | clock;
  }  // end of Timer0::setMode

}  // end of namespace Timer0

/* ---------------------------------------------------------------
 Timer 1 setup
 --------------------------------------------------------------- */

namespace Timer1
{
  // TCCR1A, TCCR1B
  const byte Modes [16] [2] =
  {

  { 0,                          0 },            // 0: Normal, top = 0xFFFF
  { bit (WGM10),                0 },            // 1: PWM, Phase-correct, 8 bit, top = 0xFF
  {               bit (WGM11), 0 },            // 2: PWM, Phase-correct, 9 bit, top = 0x1FF
  { bit (WGM10) | bit (WGM11), 0 },            // 3: PWM, Phase-correct, 10 bit, top = 0x3FF
  { 0,                          bit (WGM12) },  // 4: CTC, top = OCR1A
  { bit (WGM10),                bit (WGM12) },  // 5: Fast PWM, 8 bit, top = 0xFF
  {               bit (WGM11), bit (WGM12) },  // 6: Fast PWM, 9 bit, top = 0x1FF
  { bit (WGM10) | bit (WGM11), bit (WGM12) },  // 7: Fast PWM, 10 bit, top = 0x3FF
  { 0,                                       bit (WGM13) },  // 8: PWM, phase and frequency correct,
top = ICR1
  { bit (WGM10),                             bit (WGM13) },  // 9: PWM, phase and frequency correct,
top = OCR1A
  {               bit (WGM11),               bit (WGM13) },  // 10: PWM, phase correct, top = ICR1A
  { bit (WGM10) | bit (WGM11),               bit (WGM13) },  // 11: PWM, phase correct, top = OCR1A
  { 0,                          bit (WGM12) | bit (WGM13) },  // 12: CTC, top = ICR1
  { bit (WGM10),                bit (WGM12) | bit (WGM13) },  // 13: reserved
  {               bit (WGM11), bit (WGM12) | bit (WGM13) },  // 14: Fast PWM, TOP = ICR1
  { bit (WGM10) | bit (WGM11), bit (WGM12) | bit (WGM13) },  // 15: Fast PWM, TOP = OCR1A

  };  // end of Timer1::Modes

  // Activation
  // Note: T1 is pin 11, Arduino port: D5
  enum { NO_CLOCK, PRESCALE_1, PRESCALE_8, PRESCALE_64, PRESCALE_256, PRESCALE_1024, T1_FALLING,
T1_RISING };

  // what ports to toggle on timer fire
  enum { NO_PORT = 0,

    // pin 15, Arduino port: D9
```

```
    TOGGLE_A_ON_COMPARE  = bit (COM1A0),
    CLEAR_A_ON_COMPARE   = bit (COM1A1),
    SET_A_ON_COMPARE     = bit (COM1A0) | bit (COM1A1),

    // pin 16, Arduino port: D10
    TOGGLE_B_ON_COMPARE  = bit (COM1B0),
    CLEAR_B_ON_COMPARE   = bit (COM1B1),
    SET_B_ON_COMPARE     = bit (COM1B0) | bit (COM1B1),
  };

  // choose a timer mode, set which clock speed, and which port to toggle
  void setMode (const byte mode, const byte clock, const byte port)
  {
  if (mode < 0 || mode > 15)  // sanity check
    return;

  // reset existing flags
  TCCR1A = 0;
  TCCR1B = 0;

  TCCR1A |= (Modes [mode] [0]) | port;
  TCCR1B |= (Modes [mode] [1]) | clock;
  }  // end of Timer1::setMode

}  // end of namespace Timer1

/* ---------------------------------------------------------------
 Timer 2 setup
 --------------------------------------------------------------- */

namespace Timer2
{
  // TCCR2A, TCCR2B
  const byte Modes [8] [2] =
  {

  { 0,                          0 },            // 0: Normal, top = 0xFF
  { bit (WGM20),                0 },            // 1: PWM, Phase-correct, top = 0xFF
  {               bit (WGM21), 0 },            // 2: CTC, top = OCR2A
  { bit (WGM20) | bit (WGM21), 0 },            // 3: Fast PWM, top = 0xFF
  { 0,                          bit (WGM22) },  // 4: Reserved
  { bit (WGM20),                bit (WGM22) },  // 5: PWM, Phase-correct, top = OCR2A
  {               bit (WGM21), bit (WGM22) },  // 6: Reserved
  { bit (WGM20) | bit (WGM21), bit (WGM22) },  // 7: Fast PWM, top = OCR2A

  };  // end of Timer2::Modes

  // Activation
  enum { NO_CLOCK, PRESCALE_1, PRESCALE_8, PRESCALE_32, PRESCALE_64, PRESCALE_128, PRESCALE_256,
PRESCALE_1024 };

  // what ports to toggle on timer fire
  enum { NO_PORT = 0,

    // pin 17, Arduino port: D11
    TOGGLE_A_ON_COMPARE  = bit (COM2A0),
    CLEAR_A_ON_COMPARE   = bit (COM2A1),
    SET_A_ON_COMPARE     = bit (COM2A0) | bit (COM2A1),

    // pin 5, Arduino port: D3
    TOGGLE_B_ON_COMPARE  = bit (COM2B0),
    CLEAR_B_ON_COMPARE   = bit (COM2B1),
    SET_B_ON_COMPARE     = bit (COM2B0) | bit (COM2B1),
  };
```

```
  // choose a timer mode, set which clock speed, and which port to toggle
  void setMode (const byte mode, const byte clock, const byte port)
  {
  if (mode < 0 || mode > 7)  // sanity check
    return;

  // reset existing flags
  TCCR2A = 0;
  TCCR2B = 0;

  TCCR2A |= (Modes [mode] [0]) | port;
  TCCR2B |= (Modes [mode] [1]) | clock;
  }  // end of Timer2::setMode

}  // end of namespace Timer2

#endif
```