```
/*
 * intervalTimer.h
 *
 *  Created on: May 12, 2015
 *      Author: Taylor Cowley
 */

#ifndef INTERVALTIMER_H_
#define INTERVALTIMER_H_


#include <stdint.h>

//these are the status registers for the timers
#define TCSR0   0x00
#define TCSR1   0x10

//these are the load registers, to put numbers into the timer
#define TLR0    0x04
#define TLR1    0x14

//these are the counting register
#define TCR0    0x08
#define TCR1    0x18

//the GO bit
#define ENT0    0x00000080

//the load bits for each number register
#define LOAD0   0x00000020
#define LOAD1   0x00000020

//the cascade bit
#define CASC    0x00000800

//we want to leftshift at one point
#define LEFT_SHIFT_32 32

//for our delay
#define DELAY_COUNT 3

/** For the selected timer, sets the ENT0 bit in TCSR0; starting the timer */
uint32_t intervalTimer_start(uint32_t timerNumber);

/** For the selected timer, clears the ENT0 bit in TCSR0; stopping the timer */
uint32_t intervalTimer_stop(uint32_t timerNumber);

/**This resets the selected timer. It puts 0 in the number register, loads it into the timer,
 * then resets the timer */
uint32_t intervalTimer_reset(uint32_t timerNumber);

/** inits the selected timer
 * -Writes 0 to TCSR0 (clear status)
 * -Writes 0 to TCSR1 (clear status)
 * -Sets the Cascade bit in TCSR0 so they act as one cascaded timer
 *  */
uint32_t intervalTimer_init(uint32_t timerNumber);
```

```c
/** inits all the timers. Simply calls intervalTimer_init for each timer */
uint32_t intervalTimer_initAll();

/** resets all the timers. Simply calls intervalTimer_reset for each timer */
uint32_t intervalTimer_resetAll();

/** This function tests all the timers. It calls intervalTimer_runTest for each one */
uint32_t intervalTimer_testAll();

/** This function tests a timer. There are several printf statements for you to read */
uint32_t intervalTimer_runTest(uint32_t timerNumber);

/** For the selected timer, this gets the total number the timer counted to
 *  and converts it to seconds. Possible problem - the rollover case is not accounted for */
uint32_t intervalTimer_getTotalDurationInSeconds(uint32_t timerNumber, double *seconds);

/** This has the potential to set many bits; as many bits as are 1 in bit
 * ONLY SETS THE BITS THAT ARE 1 */
uint32_t set_bit_in_address(uint32_t address, uint32_t bit);

/** This has the potential to clear many bits; as many bits as are 1 in bit
 * ONLY CLEARS THE BITS THAT ARE 1 */
uint32_t clear_bit_in_address(uint32_t address, uint32_t bit);

/** This receives a timer id and returns that timer's base address */
uint32_t get_timer_base_address(uint32_t timerNumber);

/** This receives a timer id and returns that timer's frequency */
uint32_t get_timer_frequency(uint32_t timerNumber);

#endif /* INTERVALTIMER_H_ */
```