```c
#include <avr/io.h>
#include <avr/interrupt.h>


#include "timer2.h"


static volatile uint32_t clock_ticks;

uint32_t get_clock_ticks(void){
    uint32_t return_value;

  /* Disable interrupts so we can be sure that the interrupt
   * doesn't fire when we've copied just a couple of bytes
   * of the value. Interrupts are re-enabled if they were
   * enabled at the start.
   */
  uint8_t interrupts_were_on = bit_is_set(SREG, SREG_I);
  cli();
  return_value = clock_ticks;
  if(interrupts_were_on) {
    sei();
  }
  return return_value;

}
void mil_delay(int milliseconds){
    int start_time;
    start_time = get_clock_ticks();
    while (get_clock_ticks()<= start_time + milliseconds)
    {
        ;
    }
}
// Motor Initialization routine -- this function must be called
//  before you use any of the above functions
void timer_init()
{
    //sei();
    //Configure TIMER2
     OCR2A = 249;

     TCCR2A |= (1 << WGM21);
     // Set to CTC Mode

     TIMSK2 |= (1 << OCIE2A);
     //Set interrupt on compare match

     TCCR2B |= (1 << CS21)|(1 << CS20);
     // set prescaler to 32 and starts PWM


     // enable interrupts
     TIFR0 &= (1<<OCF0A);


}

ISR (TIMER2_COMPA_vect)
{
    // action to be done every 250 usec
     //TIFR0 &= ~(1<<OCF0A);
    clock_ticks++;
}
```