Lights in C

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

#define F_CPU 4000000
#define tickA F_CPU/64
#define tickB tickA/2


static volatile uint8_t counter_A;
static volatile uint8_t counter_B;

static void initTimers(void);
static void initPort(void);

static void initPort(void)
{
    DDRA = 0xFF; // Init PORTA as output
    DDRB = 0xFF; // Init PORTB as output
    PORTA = 0x0;
    PORTB = 0x0;
    DDRC = ~0x01; // Init PORTC as output, bit 0 is input
    PORTC = 0x00;
};

static void initTimers(void)
{
    //Timer 1
    TCNT1 = 0x0;
    // Prescaler 64 in control register for timer 1, WGM12: Set CTC
    // (Clear timer on compare match)
    TCCR1B = (1 << CS10)|(1 << CS11)|(1 << WGM12);
    OCR1A = tickA;
    // Interrupt (corresponding vector, TIMER1_COMPA_vect) on timer match
    // with tickA (62500)
    TIMSK = (1 << OCIE1A);

    //Timer 3
    TCNT3 = 0x0;
    // Prescaler 64 in control register for timer 3, WGM12: Set CTC
    // (Clear timer on compare match)
    TCCR3B = (1 << CS10)|(1 << CS11)|(1 << WGM12);
    OCR3A = tickB;
```

```
    // Interrupt (corresponding vector, TIMER3_COMPA_vect) on timer match
    // with tickB (31250)
    ETIMSK = (1 << OCR3A);

    sei(); // Enable global interrupts
};

ISR(TIMER1_COMPA_vect)
{
    counter_A++;
};

ISR(TIMER3_COMPA_vect)
{
    counter_B++;
};

int main(void)
{
    initPort();
    initTimers();

    static update = 0;

    while(1)
    {
        // Run while bit 0 in PINC is 0 (Invert PINC then AND with 1, this will
        // return 1 if PINC0 was originally 0) (Check if button is pressed)
        while (~PINC & 1)
        {
            if (!update) {
                cli(); // Disable global interrupts
                PORTA = 0xFF;
                PORTB = 0xFF;
                update = 1; // Has been updated
            }
        }
        // If button is not pressed, continue blinking lights
        if (update)
        {
            update = 0;
            sei(); // Enable global interrupts
        }

        if (counter_A > 6) counter_A = 0;
        // Bitshifting shenanigans, shifts one up. Ex: 0b00000010 -> 0b00000100
```

```
        PORTA = (1 << counter_A++);

        if (counter_B > 6) counter_B = 0;
        // Bitshifting shenanigans, shifts one up. Ex: 0b00000010 -> 0b00000100
        PORTB = (1 << counter_B++);
    };

    return 0;
}
```