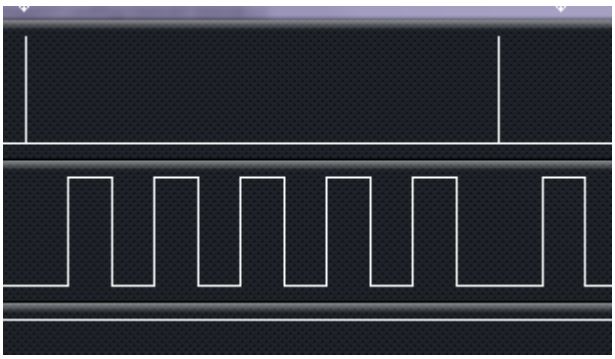
```
uint8_t cntOverflow = 0;

typedef enum BOOLEAN{FALSE, TRUE} boolean;
boolean secondCompMatch = FALSE;

int main(void)
{

    initPorts();

    initTimer1OV();

    while(1) {

    }
}
```

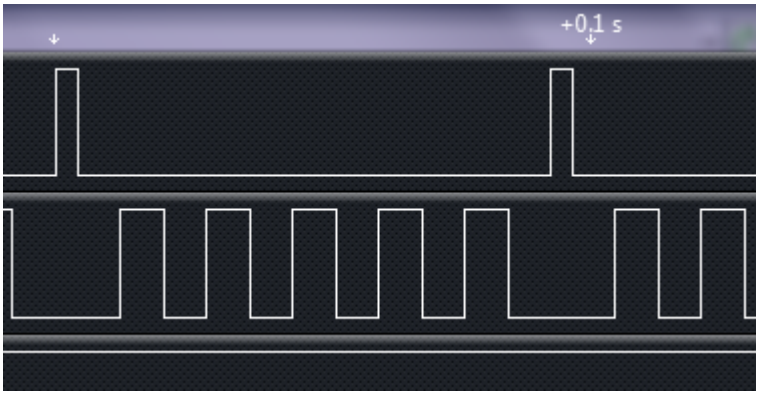
```
void initTimer1OutComp(void)
{
    togglePinC4();
    OCR1A = 0x8000;
    TIMSK = 0x00;
    TIMSK = (1<<OCIE1A); // Interrupts akivieren:  Oupt Compare Match A
    TCCR1B = (1<<WGM12) | (1<<CS10);            // Outputcompare with TOP = OCR1A, kein PreScaler
    sei();
}


ISR (TIMER1_COMPA_vect)
{

    //if (secondCompMatch) // Es wird also erst beim 2. Interrupt auf Overflow umgestellt...
    //{
    //    secondCompMatch = FALSE;
        TCCR1B = 0x00;
        togglePinC4();
        initTimer1OV();
    //}
    //else
    //{
        //secondCompMatch = TRUE;
    //}

}
```


PinC4 oben; PinC5 unten...

Erst beim 2. Compare-Match-Interrupt wird wieder auf Overflow umgestellt...

```
void initTimer1OutComp(void)
{
    togglePinC4();
    OCR1A = 0x8000;
    TIMSK = 0x00;
    TIMSK = (1<<OCIE1A); // Interrupts akivieren:  Oupt Compare Match A
    TCCR1B = (1<<WGM12) | (1<<CS10);            // Outputcompare with TOP = OCR1A, kein PreScaler
    sei();
}


ISR (TIMER1_COMPA_vect)
{

    if (secondCompMatch) // Es wird also erst beim 2. Interrupt auf Overflow umgestellt...
    {
        secondCompMatch = FALSE;
        TCCR1B = 0x00;
        togglePinC4();
        initTimer1OV();
    }
    else
    {
        secondCompMatch = TRUE;
    }

}
```

Nun funktioniert‘s...



```
void initTimer1OV(void) // Timer 1 auf Overflow einstellen
{
    TIMSK = 0x00;
    TIMSK = (1<<TOIE1); // Interrupts akivieren:  Overflow Timer1
    TCCR1B = (1<<CS10); // PreScaler = 1
    sei();

}


ISR (TIMER1_OVF_vect) // Nach dem 10. Overflow, wird Timer 1 auf Output-Compare-A umgestellt
{
    cntOverflow ++;
    if (cntOverflow > 10)
    {
        TCCR1B = 0x00;
        cntOverflow = 0;
        initTimer1OutComp();
    }
    else{
        togglePinC5();
    }

}
```