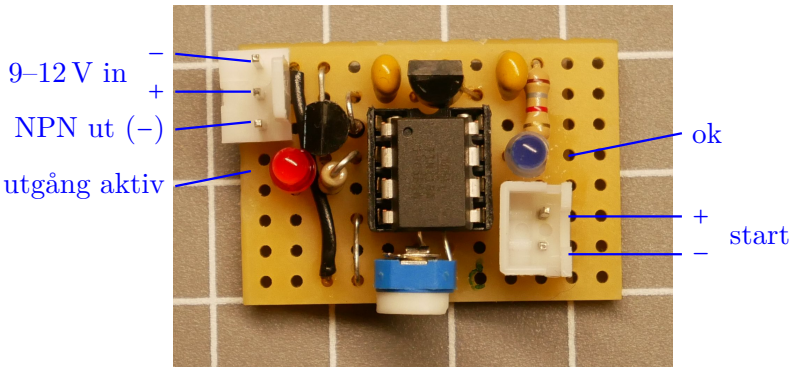
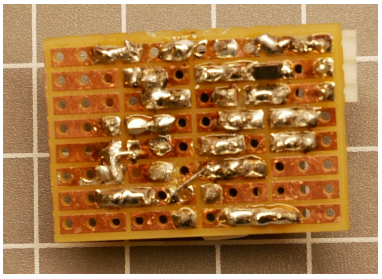
En timerkrets med en ATtiny13

version 15 mars 2021



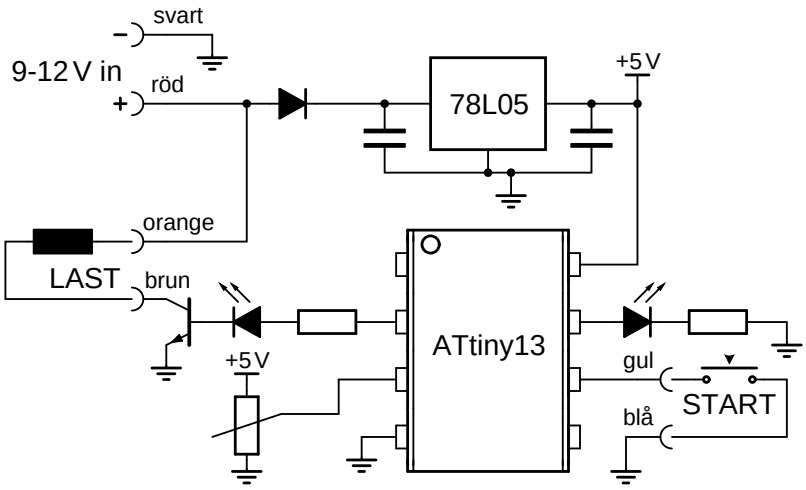
Framsida



Baksida

Det här är en prototyp till en timerkrets baserad på en ATtiny13-microcontroller. Tiden ställs in med hjälp av ett variabelt motstånd som läses av med hjälp av den inbyggda analog-digital-omvandlaren.

Utgången är en NPN-transistor i öppen-kollektor som aktiveras för den inställda tiden. Den röda lysdioden sitter i serie med transistorn och signalerar en aktiverad utgång. Den blå lysdioden signalerar att kretsen får matningsspänningen och att timern är i funktion.




```
/*
 * 20210309_tiny13_timer.c
 *
 * Created: 2021-03-09 21:08:38
 * Author : uwezi
 */

// output on PB3, active high
// trigger on PB1 / INT0 active low

#define F_CPU 1200000UL

```

```
13  #include <avr/io.h>
14  #include <util/delay.h>
15  #include <avr/interrupt.h>
16  #include <util/atomic.h>
17  #include <avr/sleep.h>
18
19  volatile int16_t timeout = 0;
20  volatile int16_t settime = 0;
21  volatile uint8_t do_adc  = 0;
22
23  ISR(INT0_vect)
24  {
25    timeout = settime;
26    PORTB |= (1 << PB3);
27  }
28
29
30  // timer interrupt about 586/s
31  ISR(TIM0_OVF_vect)
32  {
33    int16_t dummy;
34    do_adc ++;
35    if (do_adc > 200)
36    {
37      do_adc = 0;
38      PORTB |= (1 << PB2);
39      ADCSRA |= (1 << ADSC);
40      while (ADCSRA & (1 << ADSC)) { }
41      PORTB &= ~(1 << PB2);
42      dummy = 7*settime + ADC;
43      settime = dummy / 8;
44    }
45    if (timeout > 0)
46    {
47      timeout --;
48    }
49    else
50    {
51      PORTB &= ~(1 << PB3);
52    }
53  }
54
55  // PB0 unused, pull-up
56  // PB1 trigger
57  // PB2 alive blink
58  // PB3 transistor out
59  // PB4 ADC in
60  // PB5 reset
61
62  int main(void)
63  {
64    DDRB   = (1 << PB3) | (1 << PB2);
65    // pull-up on PB1
66    PORTB |= (1 << PB5) | (1 << PB1) | (1 << PB0);
67
68    // disable comparator
69    ACSR = (1 << ACD);
70
71    // prepare sleep
72    set_sleep_mode(SLEEP_MODE_IDLE);
73
74    // ADC channel ADC2
75    // internal ref
76    // normal adjustment
77    // 1:8 prescaler 150 kHz
78    ADMUX  = (0 << REFS0) | (0 << ADLAR) | (1 << MUX1) | (0 << MUX0);
79    ADCSRA = (1 << ADEN) | (0 << ADSC) | (0 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
80    // first conversion
81    ADCSRA |= (1 << ADSC);
82    while (ADCSRA & (1 << ADSC)) { }
83    settime = ADC;
84
```

```
85    // timer 0 normal counting
86    // 1:8 prescaler
87    // 1.2 MHz / 8 / 256 = 585.9375 Hz
88    TCCR0A = 0b00000000;
89    TCCR0B = (0 << CS02) | (1 << CS01) | (0 << CS00);
90    TIMSK0 = (1 << TOIE0);
91
92    // falling edge triggers INT0
93    MCUCR |= (1 << ISC01) | (0 << ISC00);
94    GIMSK |= (1 << INT0);
95    sei();
96
97    while (1)
98    {
99      sleep_enable();
100     sleep_cpu();
101   }
102 }
```