Hallo,

ATmega328P Manual Version DS40002061B.

Nun denn. Die Frequenz habe ich jetzt mal mit 100Hz festgelegt.

Die Einstellungen sehen dann wie unten aus. Wie kommt man darauf?

Timer 1

Tabelle auf Seite 141-142. Bsp. Fast-Mode gibt es mehrere. Modi 5, 6, und 7 sind auf feste **TOP** Werte festgelegt. Das heißt die Timerfrequenz ist in diesen drei Modi nur noch mittels Prescaler einstellbar. Fein einstellbar wäre Fast-Mode im Modi 14 oder 15, weil es hiermit Register gibt um einen TOP Wert einzustellen.

Der flexibelste Modi ist immer der, aus meiner Sicht, wenn man einen verwenden kann wo man den **TOP** Wert in einem **OCR1A** Register einstellen kann. Das ist nämlich gepuffert, daß heißt man kann den Wert live in seinem Programm ändern ohne den Timer stoppen/nullen zu müssen, wenn alles korrekt laufen soll mit dem Compare Match.

Wenn die Frequenz konstant bleibt, kann man für die **TOP** Wert Einstellung auch einen Modi nehmen wo ein **ICR1** Register für TOP zuständig ist. Man könnte auch hier die Frequenz später im Programm ändern, hat nur im Code mehr Aufwand, weil ICRn ungepuffert ist, dass am Rande.

In dem Bsp. verwende ich Mode 15. Damit sind schon 2 Dinge festgelegt laut Tabelle Seite 142. Das Frequenz bestimmende Register ist **OCR1A** und das Duty Cycle Register ist **OCR1B**. Der Timer 1 beim ATmega328P hat nur zwei **OCR1x** Register. **A** und **B**.
Einen verwenden wir als TOP, nämlich OCR1A, damit bleibt nur OCR1B übrig für den Hardware Timerausgangspin oder ISR.

Die LED Frequenz ist auf 100Hz festgelegt.

Jetzt benötigen wir den TOP Wert zusammen mit dem Prescaler der die 100Hz Frequenz erzeugt.
Dazu suchen wir uns die passende Formel raus.
Abschnitt Fast-PWM, Seite 132.
Die Formel stellen wir nach TOP um.
TOP = (CPU Takt / Prescaler / Takt) - 1

Mit Prescaler 64 erhalten wir: 2499 … etwas wenig für einen 16 Bit Counter
Mit Prescaler 8 erhalten wir: 19999 … schon besser
Mit Prescaler 1 erhalten wir: 159999 … zu viel, 16 Bit geht nur bis 65535

Also nehmen wir den Prescaler 8 und TOP Wert 19999.
Damit taktet der Timer schon einmal mit 100Hz.

Jetzt wollen wir aber den Pin bspw. bei 25% der Periodendauer abschalten.
Das heißt, wir brauchen noch einen zweiten Vergleichswert für OCR1B unser Compare Match Register.

25% von unseren TOP 19999 sind 5000. Das ist der Wert für unser OCR1B Register.

Damit der Timer zugehörige Pin auch schaltet, müssen wir das noch aktivieren.
Und zwar das **COM1B1** Bit, Seite 141, Tabelle 16-2.

In welchen Registern befinden sich nun die vielen Bits die wir setzen müssen?

Nun, dazu müssen wir uns im Datenblatt die Registerbelegungen mit ihren Bits anschauen.
Das beginnt ab Seite 140.

Für Fast Mode 15, Tabelle 16-4, müssen wir **alle 4 WGM1x** Bits einschalten.
Die verstreuen sich auf 2 Register. Seite 140 und 142.
TCCR1A und **TCCR1B**.

Prescaler soll 8 sein, dafür benötigen wir Bit **CS11**. Siehe Tabelle 16-5. Wird ebenfalls im Register **TCCR1B** eingestellt, Seite 142. Das sind die CS Bits.

Mehr benötigen wir nicht, alles in Code umgesetzt sieht es dann so wie unten aus. Wichtig wäre noch das man in der IDE alle Register vorher nullt, weil die intern vorgeladen werden für analogWrite usw. Sonst macht der Timer nicht das was man möchte.

Ich hoffe es ist verständlich formuliert.

```
/*
  Arduino UNO/Mega2560

  LED mit 25% Duty Cycle bei 100Hz
*/
// Mega
// const byte Takt_Pin {12};  // OC1B bzw. PB6, nicht invertiert

// Uno
const byte Takt_Pin {10};  // OC1B bzw. PB2, nicht invertiert

void setup(void)
{
  pinMode(Takt_Pin, OUTPUT);
  set_Timer1();
}

void loop(void)
{

}

// ****** Funktionen ******* //
void set_Timer1()   //  Fast-PWM, Mode 15
{
  TCCR1A = 0;    // Reset
  TCCR1B = 0;    // Reset
  TIMSK1 = 0;    // Reset (disable Timer Compare Interrupts)
  TCNT1  = 0;    // Reset bzw. Start 0
  OCR1A = 19999; // TOP Wert bestimmt Auflösung und mit Prescaler den PWM Takt
  OCR1B = 5000;  // Pulsweite, OCR1B <= OCR1A
  TCCR1A = (1<<COM1B1) | (1<<WGM11) | (1<<WGM10);  // nicht invertiert
  TCCR1B = (1<<WGM13) | (1<<WGM12) | (1<<CS11);    // Prescaler 8
}
```