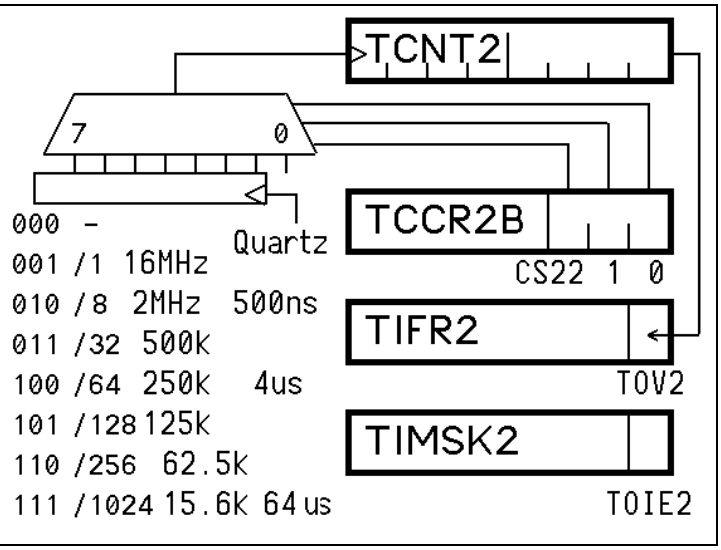
Timer2 –AtMega328

Le but est de comprendre le Timer 2 de l'AtMega328 dans son mode le plus simple de compteur de coups d'horloge du microcontrôleur. Pour utiliser ce mode, il faut extraire des 448 pages de la documentation du circuit les 21 pages qui concernent le timer 2 et sa fonctionnalité commandée par 9 registres 8 bits. Heureusement, les fonctions PWM, input capture, output compare ne sont pas activées au reset, et il nous suffit de travailler avec 4 registres.

Le registre TCNT2 est un compteur 8 bits qui tourne en permanence à une fréquence fixée par un prédiviseur commandé par les 3 bits de poids faible du registre TCCR2B. Le compteur reçoit donc une fréquence entre 16MHz et 15,625 kHz (période 64us), la durée max du comptage est de 16.4ms. Quand le compteur passe de 0xFF à 0, le bit TOV2 dans le registre TIFR2 est activé.
C'est tout ce qu'il nous faut savoir pour l'instant, TIMSK2 interviendra dans les interruptions.



Pour vérifier que l'on a compris, clignotons une led à 1Hz.
C'est une période très basse pour ce timer 8 bits. Le comptage maximum est de 256, le compteur fait un tour en 16.4 ms. Pour avoir 500ms, il nous faudra ajouter un postdiviseur logiciel par 30.5. Pour éviter cette partie décimale, on peut mettre le prédiviseur à la valeur 6 (16 µs) et compter en tout 500ms/16 µs= 31250 = 125x250 (par exemple) soit 125 pour le postdiviseur et 250 pour le timer.
Quand le timer arrive à 256=0 le flag TOV2 est activé. Pour remettre TOV2 à zéro par logiciel il faut écrire un 1!. On compte de 6 à 250, et on remet le timer à 6 à ce moment. On ajoute 1 au postdiviseur, et quand il atteint 125, on inverse la Led., on le remet à zéro pour continuer le cycle.
L'ordre a de l'importance. Si on inverse la Led avec la combinaison digitalWrite/digitalRead, la durée est de ~110 µs! il faudrait ajouter 55 à la valeur 6 tous les 125 cycles si on veut rester précis. En inversant la led avec un ou-exclusif sur le bit, cela dure 0.2 µs et on peut négliger ce temps. Mais il est de toute façon logique de commencer par réinitialiser le compteur et quittancer le flag (mettre à zéro) avant d'exécuter la tâche, que vous pouvez programmer inefficace, puisque que le timer compte sans s'occuper de vos instructions,

Le programme est donc

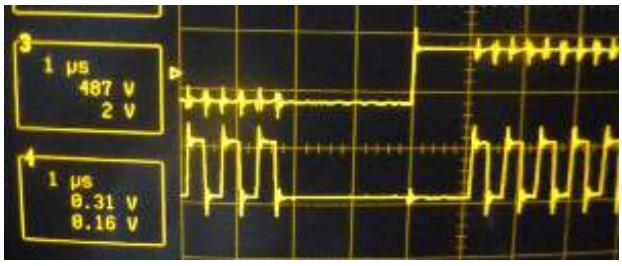
```
// TestTimerSansInt.ino
# define bLed1 5  // PORTD
# define Led1Toggle PORTD ^= 1<<bLed1
(#include "LcDef.h" contient ces deux instructions)
void setup () {
   DDRD |= 1<<bLed1 ;  //(Setup2p21();
  TCCR2B = 0b00000110;  // clk/256 16µs
  TIFR2 = 0b00000001;  // TOIE2
  TCNT2  = 256-250 ;
}
byte postDiv = 0;
void loop () {
  if (TIFR2 & 1<< TOV2) {
     TCNT2 = 256-250 ;
```

```
      bitSet (TIFR2,TOV2); // oui, bitSet
      if (postDiv++ > 125) {
        postDiv = 0 ;
        Led1Toggle;
      }
    }
}
```

Clignoter par interruption

Pour faire la même chose par interruption, il faut l'autoriser (localement et globalement) et dire au processeur ou se trouve la routine d'interruption. Cela doit se faire en assembleur, mais Arduino a prévu une fonction au nom réservé qu'il suffit de compléter. Quand le compteur TCNT2 passe de 0xFF à 0, le flag TOV2 est activé, et si les 2 bits d'interruptions sont activés, la routine d'interruption est activée et TOV2 est automatiquement mis à zéro.
Le programme de test s'écrit simplement

```
//TestTimerInt
# define bLed1 5  // PORTD
# define Led1Toggle PORTD ^= (1<<bLed1)
//(#include "DefC2p2l.h" contient ces deux instructions)
void setup () {
  DDRD |= 1<<bLed1 ;  // (Setup2p2l();
  DDRC = 0xFF;
  cli();   // désactive l'interruption globale
  TCCR2B = 0b00000110;  // clk/256 16us
  TIMSK2 = 0b00000001;  // inter locale par TOIE2
  sei();  // active l'interruption globale
}
volatile byte postDiv ;
ISR (TIMER2_OVF_vect)  {
  TCNT2 = 256-250; // --> 250x16us = 4ms
  if (postDiv++ > 125) {
    postDiv = 0;
    Led1Toggle;
  }
}
void loop () {
    PORTC = ~PORTC ;  //pour mesurer durée inter
}
```



A l'oscilloscope, en observant un bit du PORTC, on voit le temps d'inversion du registre (0.16 us) plus le temps de retour au début de la boucle while (0.16 µs) .
L'interruption coupe le programme principal pendant ~3 µs (~5µs quand le Led1Toggle est exécuté).

Le microcontrôleur AtMega328 a 3 timers utilisés par les librairies Arduino.
Le Timer0 est 8 bits et est utilisé pour les fonctions delay(), millis() et micros().
Il commande le PWM sur les pins 5 et 6, apparemment sans conflits. Sa durée est de 8-10 us.
Le Timer1 est 16 bits et est utilisé pour la librairie Servo
Il commande le PWM sur les pins 9 et 10 si la librairie Servo n'est pas utilisée
Le Timer 2 est 8 bits est utilisé pour la fonction Tone().
Il commande le PWM sur les pins 3 et 11 si Tone n'est pas utilisé. La pin 11 est utilisée par le SPI.
C'est celui que nous utiliserons dans nos exercices et applications.

http://arduino.cc/en/Tutorial/SecretsOfArduinoPWM