Université Djilali BOUNAAMA, Khemis Miliana
ملیانة خمیس بونعامة جیلالی جامعة
Faculté des Sciences et de la Technologie
والتكنولوجیا العلوم كلیّة

Série d'exercices 4: Timers et Compteurs

Exercice 1 : Comptage simple et débordement (Overflow)

On utilise le Timer0 (8 bits) d'un AVR avec un prescaler de 64. L'horloge système est à 16 MHz.

1. Calculer la période de comptage (temps entre deux ticks).
2. Calculer la durée avant un débordement (passage de 0xFF à 0x00).
3. Écrire une fonction d'initialisation pour que le timer déclenche une interruption à chaque débordement.

Exercice 2 : Mode CTC avec interruption périodique

Avec le Timer1 (16 bits), on veut générer une interruption toutes les **5 ms**. Horloge 16 MHz, prescaler 8.

1. Déterminer la valeur de comparaison OCR1A.
2. Écrire l'initialisation en mode CTC (Clear Timer on Compare Match).
3. Que se passe-t-il si on choisit un prescaler trop petit ?

Exercice 3 : Génération d'un signal PWM rapide (mode Fast PWM)

On utilise le Timer0 en mode Fast PWM (8 bits) sur la broche OC0A (PD6). Fréquence PWM souhaitée = 31,25 kHz. Horloge 16 MHz.

1. Choisir le prescaler.
2. Donner la valeur de TOP et la résolution.
3. Écrire le code pour un rapport cyclique de 30%.

Université Djilali BOUNAAMA, Khemis Miliana
مليانة خميس بونعامة جيلالي جامعة
Faculté des Sciences et de la Technologie
والتكنولوجيا العلوم كلية

Solution Série 4

Exercice 1 :

1. Fréquence du timer :

   $f_{timer} = 16\ \text{MHz}/64 = 250\ \text{kHz}$

   Période d'un tick :

   $T_{tick} = 1/250\ \text{kHz} = 4\ \mu\text{s}$

2. Pour un compteur 8 bits : 256256 ticks.
   Durée avant débordement :

   $T_{overflow} = 256 \times 4\ \mu\text{s} = 1{,}024\ \text{ms}$

3. Code d'initialisation (AVR GCC) :

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

void init_timer0_overflow(void) {
   TCCR0A = 0x00;         // Mode normal
   TCCR0B = (1<<CS01) | (1<<CS00); // Prescaler 64
   TIMSK0 |= (1<<TOIE0);   // Activer interruption overflow
   sei();                 // Activer globalement les interruptions
}

ISR(TIMER0_OVF_vect) {
   // Action à réaliser tous les ~1,024 ms
}
```

Exercice 2 :

1. Fréquence timer : $16\ \text{MHz}/8 = 2\ \text{MHz} \rightarrow T_{tick} = 0{,}5\ \mu\text{s}$
   Nombre de ticks pour 5 ms :

   $N = 5\ \text{ms}/0{,}5\ \mu\text{s} = 10000$

En mode CTC, le compteur repart de 0 à chaque match.
Donc OCR1A = $N-1=9999$ (car on compte de 0 à 9999 = 10000 ticks).

2. Code :

```
void init_timer1_ctc_5ms(void) {
   TCCR1A = 0x00;
   TCCR1B = (1<<WGM12) | (1<<CS11); // CTC, prescaler 8
   OCR1A = 9999;
   TIMSK1 |= (1<<OCIE1A); // Interruption sur match A
   sei();
}

ISR(TIMER1_COMPA_vect) {
   // Toutes les 5 ms
}
```

Si prescaler trop petit (ex: 1), la période tick est plus courte, donc il faudrait une OCR1A très grande (jusqu'à 65535 max). Pour 5 ms avec prescaler 1 :

$N=5\ \text{ms}/(1/16\ \text{MHz})=80000>65535\rightarrow$ Impossible avec un timer 16 bits. Il faut augmenter le prescaler.

Exercice 3 :

1. En Fast PWM 8 bits, TOP = 0xFF, fréquence PWM = horloge timer / 256.
   On veut $f\text{PWM}\approx 31250\ \text{Hz}$
   Horloge timer = $16\ \text{MHz}/N$.
2. $16\cdot 10^6/N\cdot 256=31250 \implies N=16\cdot 10^6/31250\cdot 256=$
3. Donc prescaler = 2 (facteur 2).
   Réel : $16\ \text{MHz}/2=8\ \text{MHz}\rightarrow 8\ \text{MHz}/256=31{,}25\ \text{kHz}$ parfait.
4. Résolution 8 bits (0 à 255). TOP = 255.
5. Rapport cyclique 30% → OCR0A = $0{,}3\times 255\approx 76$ (ou 77 pour arrondir).

```
void init_pwm_timer0(void) {
   DDRD |= (1<<PD6);       // PD6 = OC0A en sortie
   TCCR0A = (1<<COM0A1) | (1<<WGM01) | (1<<WGM00); // Fast PWM, non inversé
   TCCR0B = (1<<CS01);     // Prescaler 8 (mais ici on veut 2 ? Attention)
}
```

Correction : Prescaler 2 n'existe pas sur Timer0 AVR.

Valeurs possibles : 1, 8, 64, 256, 1024.

Donc impossible d'avoir exactement 31,25 kHz. On choisit prescaler 8 :

16 MHz/8=2 MHz → 2 MHz/256=7,8125 kHz

Ou prescaler