Corrigé du test écrit

1) Initialisation d'une entrée avec interruption

Ecrivez les instructions qui mettent la broche P1.6 en entrée, avec résistance de tirage vers le haut (pull-up) et une interruption sur le front descendant.

```
P1DIR &=~(1<<6); P1REN |= (1<<6); P1OUT |= (1<<6);
P1IES |= (1<<6); P1IE |= (1<<6);
__enable_interrupt();
```

2) Initialisation d'un timer et de son registre de comparaison

Ecrivez les instructions qui initialisent un timer en mode de comptage, avec une horloge externe (INCLK). Une interruption doit être programmée avec un registre de comparaison pour qu'elle survienne après 80'000 cycles de l'horloge externe. Vous n'avez pas besoin d'écrire la routine d'interruption. *Nb : 80'000 est supérieur à 65'536.*

```
TA0CTL = TASSEL_3 | ID_1 | MC_2 | TAIE; // INCLK, prédivision 2
TA0CCR0 = 40000;
```

3) Programme

Écrivez un programme qui mesure la durée d'une impulsion arrivant sur P1.6. Le temps entre le front montant et le front descendant doit être placé dans une variable globale mise à jour à la fin de l'impulsion, exprimé en microseconde. La fréquence du processeur est supposée être de 25 MHz. La boucle principale du programme doit impérativement être laissée vide.

```
volatile uint16_t duree;
int main() {
  // …initialisations habituelles
  P1IES &=~(1<<6); P1IE |= (1<<6); // interruption front montant
  TA0CTL = TASSEL_2 | ID_0 | MC_2; // horloge 25 MHz sans prédivision
  __enable_interrupt();
  while(1) {
  }
}

#pragma vector=PORT1_VECTOR
__interrupt void Port1(void) {
  if (!(P1IES & (1<<6))) { // front montant
    TA0R = 0; // met le timer à 0. Mieux : TA0CTL |= TACLR;
  } else { // front descendant
    temps = TA0R / 25; // la durée mesurée est limitée à (65'535/25)us
  }
  P1IES ^= (1<<6); // inverse le front
  P1IFG &=~(1<<6); // quittance l'interruption
}
```