Timers dans le microcontrôleur ATtiny2313

I. Généralités sur le Timer/counter

Le timer/counter (temporisateur/compteur) ou tous court un 'timer' est un périphérique matériel important dans le microcontrôleur **fonctionnant indépendamment de l'exécution du programme**. Il peut fonctionner soit comme :

- un temporisateur (timer) : mesure le temps entre deux actions ou évènements et génère des impulsions. Un signal d'horloge interne (clk_I/O) ou dérivé de ce signal est utilisé pour synchroniser le timer.
- un compteur (counter) : compte le nombre d'évènements sur une broche (pin). Un signal externe provenant d'un pin d'un port est utilisé pour le synchroniser.

II. Les timer/counter dans le µC ATtiny 2313

Le µ-controleur ATtiny 2313 possède deux timers avec des résolutions différentes : **Timer/conter0 de 8 bits** et **Timer/conter1 de 16 bits**. La résolution du timer indique sa capacité de comptage, par exemple le timer 8 bits peut compter jusqu'à 2^8=256 avant de passer à 0.

III. Circuit de sélection du signal horloge

Le circuit logique qui permet la sélection du signal horloge pour le timer/counter0 (clk_{T0}) et celui du timer/counter1 (clk_{T1}) est présenté sur la figure 1.

La source du signal horloge est sélectionnée par les trois bits CS00, CS01, CS02 du registre de control du timer/counter0 (TCCR0B) selon le tableau 1. La même logique de sélection est valable pour le timer/counter1 mais en utilisant les bits CS10, CS11, CS12 du registre de control du timer/counter1.

Tableau 1

CS02	CS01	CS00	Description
0	0	0	Arrêter le timer/counter0
0	0	1	clk_{T0} =CK (Pas de division)
0	1	0	clk_{T0} =CK/8 (Division par 8)
0	1	1	clk_{T0} =CK /64 (Division par 64)
1	0	0	clk_{T0} =CK /256 (Division par 256)
1	0	1	clk_{T0} =CK /1024 (Division par 1024)
1	1	0	clk_{T0}=T0 (Le timer0 est synchroniser sur front descendant)
1	1	1	clk_{T0}=T0 (Le timer0 est synchroniser sur front montant)

IV. Timer/counter0 à 8 bits:

Ce timer est caractérisé par :

- Deux unités indépendantes de sortie de comparaison (Output Compare Units) ;
- Deux registres tampons de sortie de comparaison (Double Buffered Output Compare Registers) ;
- Remise à zéro du timer sur Compare Match ;
- Phase Correct Pulse Width Modulator (PWM);
- Période de PWM variable ;
- Générateur de fréquence ;
- Trois sources d'interruptions indépendantes (TOV0, OCF0A, and OCF0B).

Le schéma block simplifié du timer/counter0 est illustré sur la figure 2. Sur la figure 3, on décrit les registres du timer/counter0.

IV.1. Les registres du timer/counter0

IV.1.1. Le registre TCNT0

Valeur de TCNT0 est accessible en lecture ou écriture en permanence. mais si le mode comparaison est actif, le contrôle peut être passé et entraîner une erreur dans votre programme, le compteur ne se bloquera pas.

IV.1.2. Les registres OCR0A et OCR0B

Ces registres sont les registres de sortie de comparaison (Output Compare Registers). Leurs contenues sont constamment comparées avec celui du TCNT0. Le résultat de comparaison est utilisé par le générateur de forme d'onde (Waveform Generator to generate) pour générer un signal PWM ou une sortie à fréquence variable sur les pins OC0A and OC0B (Output Compare pins).

IV.1.3. Les registres TCCR0A, TCCR0B

TCCR0A et TCCR0B (Timer/Counter0 Control Registers) sont les registres de contrôle du timer/counter0.

- Les bits **CS02, CS01 et CS00 (**les bits 2-0 du registre de control TCCR0B**)** permettent la sélection de la source du signal horloge clk_{T0}, voir tableau 1.
- Les bits **WGM02, WGM01 et WGM00** (Le bit 3 du registre de control TCCR0B et les bits 1-0 du registre de control TCCR0A, respectivement) sont les bits qui permettent de déterminer le mode d'opération du timer (c-à-d, la séquence de comptage, la source de la valeur TOP du timer et le type de génération de forme d'onde), tableau 2.

Les modes d'opération que peut supporter le timer/counter0 sont :

- ***Le mode normal ;***
- ***Le mode CTC (Clear Timer on Compare Match) ;***
- ***Le mode Phase correct PWM (pulse width modulation- modulation de largeur d'impultion)***
- ***Le mode fast PWM***

- Les bits **COM0A1-0** (les bits 7-6 du registre TCCR0A) et les bits **COM0B1-0** (les bits 5-4 du registre TCCR0A) sont les bits qui contrôlent le comportement du pin de sortie de comparaison (OCA) et (OCB), respectivement, tableau 3 -4.
- Les bits **FOC0A et FOC0B** (les bits 7-6 du registre de contrôle TCCR0B) sont actifs seulement si les bits WGM02-0 spécifient un mode non-PWM (mode normale ou mode CTC). Dans les modes PWM, ils doivent être mis à zéro. La mise à 1 de ces bits entraine le forçage d'une comparaison immédiate dans l'unité de génération de forme d'ondes. Cela veut dire que la comparaison se fait normalement selon les valeurs des bits COM0A1-0 et COM0B1-0 mais sans la génération d'aucune d'interruption et sans remise à zéro du TCNT sur compare match en mode CTC. Ces bits sont est toujours lu comme zéro.

IV.1.4. Le registre TIFR

Le registre **TIFR** (Timer Interrupt Flag Register) indique l'état des interruptions internes du Timer0 et du Comparateur.

- Le bit **OCF0A** ***Output Compare Flag*** (respectivement **OCF0B**) est mis à 1 quand une égalité se produise entre le contenu du registre TCNT0 et celui du registre OCR0A (respectivement OCR0B).
- Le bit **TOV0** ***Timer/Counter0 Overflow Flag*** est mis à 1 quand le Timer/Counter0 dépasse une certaine valeur. Cette valeur peut être la valeur MAX ou BOTTOM ou bien la valeur stockée dans le registre OCR0x, cela dépend des bits WGM02-1, tableau 2.

Ces bits sont remis à 0 par le matériel lors de l'exécution de l'interruption. Le programmeur peut aussi les mettre à 0.

IV.1.5. Le registre TIMSK

Le registre TIMSK (Timer Interrupt Mask) contient trois bits de masquage d'interruptions associés au timer/counter0 : **OCIE0A, OCIE0B et TOIE0**.

- Lorsque le bit **OCIE0A-*Output Compare Interrupt Enable*-** (respectivement **OCIE0B**) et le *bit I* du registre d'état sont mis à 1 l'interruption de comparaison A (respectivement B) du timer/counter0 est validée.
- Lorsque le bit **TOIE0 –*Timer/counter0 Overflow Interrupt Enable* -** et le *bit I* du registre d'état sont mis à 1 l'interruption de débordement du timer/counter0 est validée.

IV.2. Unité de sortie de comparaison

Le schéma synoptique de l'unité de sortie de comparaison est donné sur la figure 4. Le générateur de forme d'onde emploi les bits COM0A1-0 (respectivement COM0B1-0) pour définir l'état du registre tampon OC0A (respectivement OC0B).
La direction des pins PB2/OC0A et PD5/OC0B est toujours contrôlée par le registre de direction des données (DDRx). Pour sélectionner la sortie de comparaison OC0x provenant du générateur de forme d'onde il faut que l'un des bits COM01-0 soit mis à 1. Avant que la valeur de OC0x soit disponible sur le pin, il faut configurer ce pin en sortie en mettant un 1 dans le bit correspondant du registre de direction de données.
Les registres OCR0x sont à double tampon ('double buffered') seulement en modes PWM. Dans les modes non-PWM la caractéristique du 'double buffering' est désactiver. Le 'double buffering'

synchronise la mise à jour du registre OCR0x ce qui prévient d'obtenir des impulsions PWM de longueur impaire et non-symétrique. Ce fonctionnement est très simple : lorsque le 'double buffering' est validé le CPU a accès au registre tampon de OCR0x et dans le cas contraire le CPU a accès directement au registre OCR0x.

IV.3. Le fonctionnement du Timer/Compteur0

Le compteur TCNT0 est le cœur du timer. Son signal d'horloge clkT0 peut être l'horloge interne, via le pré diviseur (prescalar), une horloge externe sur la broche T0 (PD4) ou inactif quand aucune source d'horloge n'est choisie.
La valeur de ce registre est incrémentée et comparée en permanence à celle du registre OCR0A (respectivement OCR0B). Le résultat de comparaison est utilisé par le générateur de forme d'onde pour produire une sortie PWM ou une sortie à fréquence variable sur le pin OC0A (respectivement OC0B).

IV.4. Les modes d'opération du timer/counter0

Le mode d'opération est le comportement du timer0 et les pins de sortie de comparaison. Il est défini par les bits de mode de génération de forme d'onde WGM02-0 et les bits de mode de sortie de comparaison COM0x1-0. Les bits WGM02-0 affectent la séquence de comptage contrairement aux COM0x1-0. L'inversion ou non de la sortie PWM générée est contrôlé par les bits COM0x1-0. Pour les non-PWM modes ces bits contrôlent si la sortie doit être mise à 1, mise à 0, ou inverser en cas d'égalité (compare match).
Les modes d'opération du timer/counter0 sont :

- Le mode normal ;
- Le mode CTC (Clear Timer on Compare Match) ;
- Le mode Phase correct PWM (pulse width modulation- modulation de largeur d'impultion)
- Le mode fast PWM

V. Programmation du timer/counter0 en mode normal

Le mode normal est le mode le plus simple (WGM01=0 et WGM00=0) où le registre TCNT0 s'incrémente à chaque coup d'horloge jusqu'à sa valeur maximale (Fixed TOP Value=0xFF), puis redémarre à partir du BOTTOM=0x00. Le drapeau de dépassement (TOV0) est mis à 1 dans le même cycle horloge de redémarrage du registre TCNT0. Il peut être lu par le programme (pour générer une interruption) et être remis à 0. La remise à 0 de ce bit se fait en écrivant un 1. A tous moment, une nouvelle valeur peut être écrite dans le registre TCNT0, figure 5. Dans le mode normal, l'unité de sortie de comparaison peut être utilisée pour générer des interruptions à des moments données. La génération des signaux en ce mode est à éviter car elle occupe beaucoup du temps du CPU. Car on ne peut pas forcer le timer à zéro en cas d'égalité entre les registres TCNT0 et OCR0x.

Exemple de programmation du timer/counter0 en mode normal pour clignoter une LED.

Pour programmer le timer/counter0 avec les caractéristiques suivantes :

- La source du signal horloge du system : un oscillateur à quartz ;
- Facteur de division du signal horloge provenant d'un oscillateur à quartz : 8 ;
- Facteur de division du signal horloge du timer : 1024

- Mode d'opération : mode normale ;
- Pin de sortie OC0A : inverser en cas d'égalité (toggle on compare match) ;
- Pin de sortie OC0B : déconnecté ;
- Valider les interruptions de dépassement, compare match A et B ;

On doit déterminer la valeur des registres suivants : TCCR0A, TCCR0B, TCNT0, OCR0A, OCR0B, TIMK et CLKPR.

Etape 1 : Configuration du signal horloge

- La configuration du signal horloge source se fait par le registre CLKPR (Clock prescale register), figure 8. Le bit 7 (CLKPCE : Clock Prescaler Change Enable) doit être mis à 1 pour permettre le changement des bits CLKPS. La mise à jour de ce bit n'est possible que si les autres bits du CLKPR sont mis à zéro simultanément. Ce bit est effacé par le matériel 4 cycles après son écriture ou bien lorsque les bits CLKPS sont écris. Les bits 3-0 (CLKPS3-0) définies le facteur de division entre la source du signal horloge et le system horloge interne, tableau 5.
  Si on veut un facteur de division =8 pour l'oscillateur à quartz on doit écrire la portion du programme suivante :

```
#progma optsize-                //Les deux instructions suivantes doit être exécuté en un temps limité c'est //pourquoi il
                                faut accélérer l'exécution par la désactivation temporaire de //l'optimisation de la
                                taille.
CLKPR=0x80 ;                    // mettre à 1 le bit CLKPCE
CLKPR=0x03     ;                //mettre le bit CLKPCE à 0 et choisir CLKPS3-0=3
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+                //Activer l optimisation de la taille du programme
#endif
```

- Pour sélectionner le facteur de division 1024 pour le signal horloge du timer, il faut prendre CS02 CS01 CS00=$(101)_2$ tableau 1

Etape 2 : Sélection du mode normal

Le mode normal correspond à WGM_2=0, WGM_1=0, WGM_0=0.

Etape 3 : Le pin de sortie OC0A doit s'inverser sur compare match

Comme le mode est normale c'est-à-dire un mode non-PWM, alors il faut que COM0A1=0 et COM0A0=1. Pour permettre la validation de la sortie de comparaison, il faut mettre un 1 au bit 2 du registre de direction de données DDRB (le pin OC0A=PB2 doit être configuré en sortie).

Etape 4 : Déconnecter le pin de sortie OC0B

Comme le mode est normale c'est-à-dire un mode non-PWM, alors pour déconnecter le pin de sortie OC0B, il faut que COM0B1=0 et COM0B0=0.
Dans ce cas le pin n'est pas utilisé, donc peu importe la valeur du bit correspondant du registre de direction.

Etape 5 : Manipulation des interruptions

Puisqu'on veut valider les interruptions de dépassement, Compare Match A et B, alors il faut mettre : TOIE0=1 ; OCIE0A=1 ; OCIE0B=1 ; FOC0A=0 ; FOC0B=0.

On peut maintenant calculer les valeurs des registres nécessaires :

Bit	7	6	5	4	3	2	1	0	
	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00	**TCCR0A**
	0	1	0	0	0	0	0	0	=0x40

Bit	7	6	5	4	3	2	1	0	
	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00	**TCCR0B**
	0	0	0	0	0	1	0	1	=0x05

Bit	7	6	5	4	3	2	1	0	
	TOIE1	OCIE1A	OCIE1B	-	ICIE1	OCIE0B	TOIE0	OCIE0A	**TIMSK**
	0	0	0	0	0	1	1	1	=0x07

TCCR0A=$(01000000)_2$=0x40

TCCR0B=$(00000101)_2$=0x05

DDRB=$(11111111)_2$=0xFF

PORTB=0x00

TIMSK=0x07

TCNT0=0x00

OCR0A=0xFA

OCR0B=0x00

```
#include <tiny2313.h>

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
  {/* Place your code here*/ }

// Timer 0 output compare A interrupt service routine
interrupt [TIM0_COMPA] void timer0_compa_isr(void)
{ /* Place your code here*/ }

// Timer 0 output compare B interrupt service routine
interrupt [TIM0_COMPB] void timer0_compb_isr(void)
{/* Place your code here*/}

// Declare your global variables here

void main(void)
{  // Declare your local variables here

// Crystal Oscillator division factor: 8
#pragma optsize-
CLKPR=0x80;     // Enable the clock prescale change
CLKPR=0x03;     // set clock division factor to 8 (8Mz/3=1Mhz)
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif

DDRB=0xFF; // Set all pins as ouput inluding pin PB2 (OC0A pin)
PORTB=0xFF;

//---- Timer/Counter 0 initialization-------
TCCR0B=0x05;  // Clock source, with system clok value 8Mhz/8 = 1MHz and mode setting
TCCR0A=0x40;  // Mode Normal Top=0xFF, OC0A output: Toggle on compaer match
TCNT0=0x00;   // Set an 8-bit value to the timer register
OCR0A=0xFA;   // Set an 8-bit value to Output Compare Register A
OCR0B=0x00;   // Set an 8-bit value to Output Compare Register B
TIMSK=0x07;   // Timer(s)/Counter(s) Interrupt(s) initialization
              // Timer 0 overflow, compare match A and B interrupts are enabled

#asm("sei") // Global enable interrupts

while (1)
      {
      /* Place your code here*/

      }

}
```

Si on veut montrer l'utilité de la procédure de service de l'interruption de sortie de comparaison A, on place par exemple le code suivant :

```
interrupt [TIM0_COMPA] void timer0_compa_isr(void)
{ /* Place your code here*/

m++;
if(m==10)
          {
            m=0;
            PORTB.7=~PORTB.7;
          }
}
```

Avec ce code l'interruption de sortie de comparaison A est utilisée pour faire clignoter la LED reliée au pin7 du port B avec une fréquence inversement proportionnelle à la fréquence de Clk_{T0}.

$f_{PB7}=f_{T0} / M.256=f_{ClkI/O} / N. M.256$ dans ce cas M=10

VI. Programmation du timer/counter0 en mode CTC (Clear Timer on Compare match)

Dans le mode CTC (WGM02-0=2), le registre TCNT0 est mis à zéro lorsque sa valeur est égale à la valeur du registre OCR0x. Ce mode permet un bon contrôle de la fréquence de sortie de comparaison. Il simplifie aussi les opérations de comptage des événements externes. Le schéma bloc du timer/counter0 en mode CTC et son diagramme temporel sont présentés sur la figure 9 et figure 10, respectivement.
Une interruption peut être générer à chaque fois la valeur de TCNT0 atteint la valeur TOP en utilisant le drapeau OCF0x. Si l'interruption est validée, la routine de service peut être utilisée pour changer la valeur TOP. Cependant, le changement de TOP à une valeur proche du BOTTOM lors du comptage du TCNT0 avec un petit facteur de division ou sans division doit être fait avec précaution, car il n y a pas de caractéristique de 'double buffering' dans ce mode. Si une nouvelle valeur écrite dans le registre OCR0x est inférieur à la valeur courante de TCNT0, le compteur va rater le 'Compare Match' et continu son comptage jusqu'à la valeur maximale (0xFF) puis revient à zéro avant que le 'Compare Match' peut se produire.

VI.1. L'application du timer/counter0 en mode CTC dans la génération des signaux

Pour générer un signal en mode CTC, la sortie OC0A est configurer de telle sorte qu'elle s'inverse à chaque 'Compare Match' c'est-à-dire mettre COM0A1-0=1. La fréquence du signal à générer est définie par :

$$f_{OCnx} = \frac{f_{clk\_IO}}{2.N.(1+OCRnx)} \qquad (1)$$

Où f_{OCnx} est la fréquence du signal généré sur le pin OCnx ;
$f_{clk\_I/O}$ est la fréquence du signal horloge utilisé par le timer ;
N est le facteur de division du signale horloge du timer (1,8,64,256,1024)

Lorsque N=1 et OCRnx=0, la fréquence du signal généré atteint sa valeur maximale :

$$f_{OCnx_{max}} = \frac{f_{clk\_IO}}{2.N}$$

VI.2. La sélection du facteur de division du signal horloge à partir de la fréquence du signal à générer.

On suppose que la fréquence du signal horloge du timer clk_I/O est 8Mhz et la fréquence désirée est 200Hz. Pour déterminer la valeur de division de diviseur de fréquence à utiliser, il faut suivre les étapes suivantes :

1) Choisir un facteur de division parmi {1,8,64,256 ou 1024} (256) ;
2) Diviser la fréquence de clk_I/O sur le facteur de division choisis (8000000 / 256 = 31250) ;
3) Diviser le résultat sur la fréquence désirée (31250 / 200Hz = 156,25) ;

4) Comparer le résultat avec la valeur maximale du registre TCNT0. Si elle est inferieur alors garder la valeur de division (256 < 156,25). Sinon il faut choisir une autre valeur de division plus grande.

Exemple

Pour programmer le timer/counter0 pour générer un signal sur le pin B2 une fréquence de 1 KHz et ayant les caractéristiques suivantes :

- La source d'horloge : le signal horloge du system ;
- La fréquence du signale horloge clk_I/O est 8MHz. Donc le facteur de division du signal horloge provenant d'un oscillateur à quartz est 1 ;
- Mode d'opération : mode CTC ;
- Pin de sortie OC0A : inverser en cas d'égalité (toggle on compare match) ;
- Pin de sortie OC0B : déconnecté ;
- Inhiber (bloquer) les interruptions de dépassement, compare match A et B ;

On doit déterminer la valeur des registres suivants : TCCR0A, TCCR0B, TCNT0, OCR0A, OCR0B, TIMK et CLKPR.

Etape 1 : Trouver le facteur de division N et la valeur de OCR0A

- ***Le facteur de division ?***
    1) Si on choisi N=8
    2) 8000000/8= 1000000 ;
    3) 1000000/1000=1000.
    4) 1000>256.

    On doit donc choisir une valeur de N plus grande
    1) Soit N=64
    2) 8000000/64= 125000 ;
    3) 125000/1000=125.
    4) 125<256.

Alors N=64

- ***La valeur de OCR0A ?***

A partir de l'équation (1) on a :

$OCR0A = \frac{f_{clk\_IO}}{2.N.f_{OCn0A}}$ $1 = \frac{8000000}{2.64.1000}$ 1=61.5

Comme la valeur de OCR0A doit être une valeur entier, alors on l'affecte la valeur 61.

Donc OCR0A=61=0x3D.

Etape 2 : Configuration du signal horloge

- La portion du programme qui permet la configuration du signal horloge source s'écris comme suit :

```
#progma optsize-          //Les deux instructions suivantes doit être exécuté en un temps limité c'est //pourquoi il
                          faut accélérer l'exécution par la désactivation temporaire de //l'optimisation de la
                          taille.
CLKPR=0x80 ;              // mettre à 1 le bit CLKPCE
CLKPR=0x00    ;           //mettre le bit CLKPCE à 0 et choisir CLKPS3-0=0
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+          //Activer l optimisation de la taille du programme
#endif
```

- Pour sélectionner le facteur de division 64 pour le signal horloge du timer, il faut prendre CS02 CS01 CS00=$(011)_2$

Etape 3 : Sélection du mode CTC

Le mode normal correspond à WGM_2=0, WGM_1=1, WGM_0=0.

Etape 4 : Le pin de sortie OC0A doit s'inverser sur compare match

Comme le mode est CTC c'est-à-dire un mode non-PWM, alors il faut que COM0A1=0 et COM0A0=1. Pour permettre la validation de la sortie de comparaison, il faut mettre un 1 au bit 2 du registre de direction de données DDRB (le pin OC0A=PB2 doit être configuré en sortie).

Etape 5 : Déconnecter le pin de sortie OC0B

Comme le mode est normale c'est-à-dire un mode non-PWM, alors pour déconnecter le pin de sortie OC0B, il faut que COM0B1=0 et COM0B0=0.
Dans ce cas le pin n'est pas utilisé, donc peu importe la valeur du bit correspondant du registre de direction.

Etape 6 : Manipulation des interruptions

Puisqu'on veut bloquer les interruptions de dépassement, Compare Match A et B, alors il faut mettre : TOIE0=0 ; OCIE0A=0 ; OCIE0B=0 ; FOC0A=0 ; FOC0B=0.

On peut maintenant calculer les valeurs des registres nécessaires :

Bit	7	6	5	4	3	2	1	0	
	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00	**TCCR0A**
	0	1	0	0	0	0	1	0	=0x42

Bit	7	6	5	4	3	2	1	0	
	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00	**TCCR0B**
	0	0	0	0	0	0	1	1	=0x03

Bit	7	6	5	4	3	2	1	0	
	TOIE1	OCIE1A	OCIE1B	-	ICIE1	OCIE0B	TOIE0	OCIE0A	**TIMSK**
	0	0	0	0	0	0	0	0	=0x00

TCCR0A =$(01000010)_2$=0x42
TCCR0B=$(00000011)_2$=0x03
DDRB=$(11111111)_2$=0xFF
PORTB=0x00
TIMSK=0x00

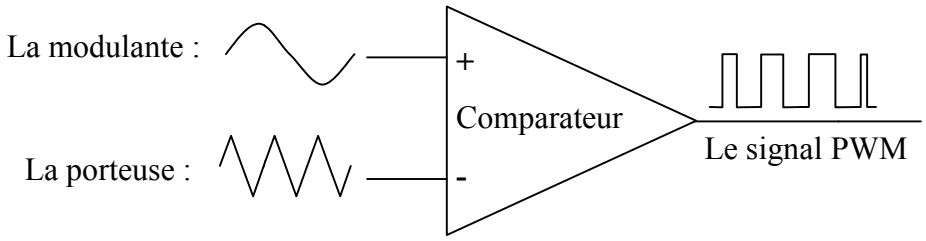
TCNT0=0x00
OCR0A=0x3D
OCR0B=0x00

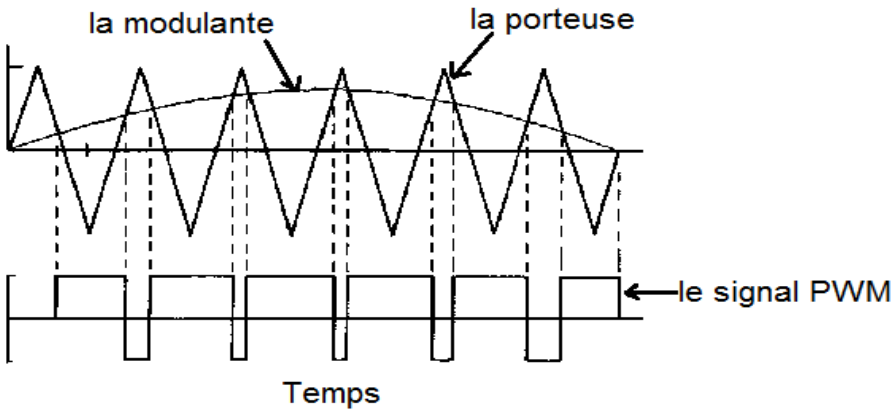
Le programme final qui génère un signal de fréquence 1KHz sur PinB2 en utilisant le timer/counter0 en mode CTC est donné comme suit :

```
#include <tiny2313.h>
void main(void)
{
// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=0x80;
CLKPR=0x00;
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif

// Port B initialization
PORTB=0xFF;
DDRB=0xFF;

// timer 0 initialization
TCCR0B=0x03;  // Clock value: (CLC_I/O/64=125 KHz)
TCCR0A=0x42;  // Mode: CTC top=OCR0A, OC0A output: Toggle on compare match
TCNT0=0x00;
OCR0A=0x3D;
OCR0B=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

while (1)
    {};
}
```

VII. La modulation de largeur d'impulsion (Pulse Width Modulation -PWM)

Elle consiste à comparer la modulante (le signal à moduler) à une porteuse généralement triangulaire. Le signal de sortie vaut 1 si la modulante est plus grande que la porteuse, sinon il vaut 0, voir les deux figures ci-dessous.





La mo1dulation de largeur d'impulsion est une technique très utilisée dans le domaine de télécommunication et le contrôle de puissance (On peut par exemple, contrôler la luminosité des LEDs, mélanger les couleurs en utilisant des LEDs RGB, contrôler la vitesse d'un moteur DC, générer des signaux audio, générer un signal modulé pour par exemple commander une LED infra-rouge d'une télécommande). Elle est principalement utilisée pour contrôler la quantité de l'énergie à fournir au dispositif électrique par fonctionnement tout ou rien. Ainsi, la valeur moyenne de l'énergie reçue par le dispositif électrique dépond du rapport cyclique.

Le **rapport cyclique** pour un signal carré, noté α, est défini comme étant le rapport entre la durée niveau haut (t_H) du signal et sa période (T).

$$\alpha = \frac{t_H}{T}$$

Il varie de 0 à 1, et en pourcentage de 0 % à 100 %.

VIII. Programmation du timer/counter0 en mode fast PWM

Le mode fast PWM (WGM02:0 = 3 or 7) permet la génération de signaux de type PWM de haute fréquences sur la sortie OC0x.

Dans ce mode, le compteur TCNT0 compte du BOTTOM jusqu'à TOP puis recommence à partir du BOTTOM.

$$TOP = \begin{cases} Max & \text{si WGM02=0} \\ OCR0x & \text{si WGM02=1} \end{cases}$$

Avant de passer de la valeur TOP au BOTTOM, l'indicateur TOV0 est mise à 1. Si l'interruption associée est validée, sa routine peut être utilisée pour changer la valeur du registre OCR0x.

Au cours du comptage du BOTTOM au TOP la comparaison entre TCNT0 et OCR0x est effectuée. Si une égalité est trouvée, le pin OC0x est programmé par la valeur de COM0x1 :0 pour se mettre à 1 ou à 0. Plus précisément :

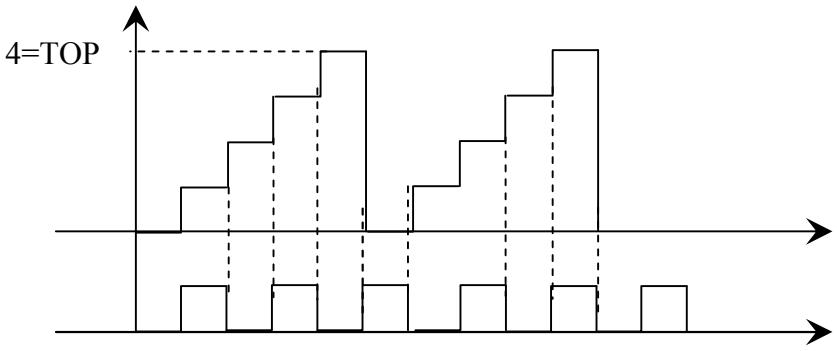
1) Si COM0x1 :0=1 et si WGM02=1, la sortie OC0A est inversé en cas d'égalité entre TCNT0 et OCR0A. Cette fonctionnalité n'est pas disponible pour OC0B.
2) Si COM0x1 :0=2, le mode de sortie de comparaison est sans inversement (mode non-inverted fast PWM) où la sortie OC0x est :
   mise à 0(clear) si TCNT0=OCR0x
   mise à 1(set) si TCNT0=TOP

3) Si COM0x1 :0=3, le mode de sortie de comparaison est avec inversement (mode inverted fast PWM) où la sortie OC0x est :
   mise à 1(set) si TCNT0=OCR0x
   mise à 0(clear) si TCNT0= TOP

Remarque 1 :

Il faut noter que les valeurs extrêmes du registre OCR0x représentent des cas spéciaux dans la génération du signal PWM.

- Si OCR0x =BOTTOM, le signal à la sortie OC0x est formé d'un pic étroit dans chaque (MAX+1) cycle horloge du timer0.
- Si OCR0x = MAX, le signal à la sortie OC0x sera constant de niveau haut dans le mode non-inverted fast PWM et constant de niveau bas dans le mode inverted fast PWM.



VIII.1.1. Fast PWM Mode avec TOP= MAX=0xFF (WGM02=0)

Le schéma bloc du timer/counter0 en Fast PWM mode et son diagramme temporel sont présentés sur la figure 11 et figure 12, respectivement.

La fréquence du signal généré sur OC0x est obtenu par :

$$f_{OC0xPWM} = \frac{f_{T0}}{256} = \frac{\frac{f_{CLK}}{N}}{256}$$

$$f_{OC0xPWM} = \frac{f_{CLK}}{N.256}$$

N représente le facteur de division (1, 8, 64, 256 ou 1024).

Le rapport cyclique est déterminé par la valeur à mettre dans le registre OCR0x. Cette valeur varie de 0 à 255 et peut être obtenue a partir de la formule suivante :

Rapport cyclique=$\frac{OCR0x+1}{256}$ 100 mode Non-Inversé

Rapport cyclique =$\frac{256-OCR0x}{256}$ 100 mode Inversé

Exemple : en mode Non inversé

Rapport cyclique(%)	OCR0x	
	En décimale	En hexadécimale
30	76	0x4C
64	163	0xA3
80	2044	0xCC

Remarque 2 :
Le Timer0 peut générer des signaux PWM sur deux pins différents. Ces signaux possèdent la même fréquence mais diffèrent dans le rapport cyclique.

La génération d'un signal PWM dans ce mode se fait en deux étapes :

- Programmation du Timer0 pour générer un signal de fréquence spécifiée.
- Détermination de la valeur fixe à mettre dans le registre de comparaison OCR0x qui permet de définir le rapport cyclique.

VIII.1.2. Fast PWM Mode avec TOP=OCR0x (WGM02=1)

Ce mode est utilisé pour avoir plus de contrôle sur la fréquence du signal PWM généré que le mode précédent.

Dans ce mode, le compteur TCNT0 compte du BOTTOM jusqu'à TOP=**OCR0A** puis recommence à partir du BOTTOM. Selon le cas spécial (OCR0x=TOP) qu'on a cité dans la remarque 1, on a
Si COM0A1 :0=2, le signal généré sur OC0A est constant de niveau haut (ce n'est pas signal PWM).

Si COM0A1 :0=3, le signal généré sur OC0A est constant de niveau bas (ce n'est pas signal PWM).
La seule combinaison utilisable est :
COM0A1 :0=1 où le signal généré sur OC0A s'inverse chaque Compare Match.
Dans ce cas la fréquence du signal généré sur OC0A est :

$$f_{OC0A} = \frac{f_{clk\_IO}}{2.N.(1 + OCR0A)}$$

Et le rapport cyclique est toujours fixé à 50%

Cela s'explique par le fait que la valeur du registre OCR0A ne peut pas être utilisée comme la valeur maximale de comptage et la valeur de comparaison à la fois.
En fait, c'est le canal B qui est utilisé pour générer un signal PWM avec une fréquence et un rapport cyclique contrôlable.
La fréquence du signal est donnée par :

$$f_{OC0B} = \frac{f_{clk\_IO}}{N.(1 + OCR0A)}$$

Et le rapport cyclique est donné par :

Rapport cyclique$=\frac{OCR0B+1}{OCR0A+1}$ 100 mode Non-Inversé

Rapport cyclique $=\frac{(OCR0A+1)-OCR0B}{OCR0A+1}$ 100 mode Inversé

La génération d'un signal PWM dans ce mode se fait en 2 étapes :

- Programmation du Timer0 pour générer un signal de fréquence spécifiée en même temps que la détermination de la valeur fixe à mettre dans le registre de comparaison OCR0A.
- Détermination de la valeur fixe à mettre dans le registre de comparaison OCR0B qui permet de définir le rapport cyclique.

IX. Programmation du timer/counter0 en mode Phase Correct PWM:

Dans le mode phase correct PMW (WGM02:0 = 1 or 5), le compteur commence par compter de BOTTOM à TOP puis décompte de TOP jusqu'à BOTTOM. Notez bien que le compteur est = à TOP juste pendant un seul cycle horloge du timer0.

$$TOP = \begin{cases} MAX & \text{si WGM02 :0=1} \\ OCR0x & \text{si WGM02 :0=5} \end{cases}$$

Quand le compteur atteint BOTTOM, l'indicateur TOV0 est mise à 1.

Au cours du comptage et du décomptage la comparaison entre TCNT0 et OCR0x est effectuée.

Si une égalité est trouvée, le pin OC0x est programmé par la valeur de COM0x1 :0 pour se mettre à 1 ou à 0. Plus précisément :

1) Si COM0A1 :0=1 et si WGM02=1, la sortie OC0A est inversé en cas d'égalité entre TCNT0 et OCR0A. Cette fonctionnalité n'est pas disponible pour OC0B.
2) Si COM0x1 :0=2, le mode de sortie de comparaison est sans inversement (mode non-inverted fast PWM) où la sortie OC0x est :
   mise à 0(clear) si TCNT0=OCR0x lors du comptage
   mise à 1(set) si TCNT0=OCR0x lors du décomptage

3) Si COM0x1 :0=3, le mode de sortie de comparaison est avec inversement (mode inverted fast PWM) où la sortie OC0x est :
   mise à 1(set) si TCNT0=OCR0x lors du comptage
   mise à 0(clear) si TCNT0=OCR0x lors du décomptage

Remarque 3 :

1) Il faut noter que les valeurs extrêmes du registre OCR0x représentent des cas spéciaux dans la génération du signal PWM.
   - Si OCR0x = BOTTOM, le signal à la sortie OC0x est constamment au niveau bas.
   - Si OCR0x = MAX, le signal à la sortie OC0x sera constant de niveau haut dans le mode non-inverted PWM et constant de niveau bas dans le mode inverted PWM.

2) On peut avoir des transitions sans qu'une égalité ne se soit produite. Il ya deux cas :
   - Lorsque OCR0x=MAX et si la valeur de OCR0x change au moment où TCNT0= MAX, figure 14. La sortie OC0x est programmée selon le cas d'une égalité lors de décomptage.
   - Lorsque le timer0 commence le comptage à partir d'une valeur supérieur à la valeur de registre OCR0x. le signal à la sortie OC0x est programmée selon ce qui a due ce passé.

IX.1.1. Phase corret PWM Mode avec TOP= MAX=0xFF (WGM02=0)

Le diagramme temporel du timer/counter0 en phase correct PWM mode est présenté sur la figure 13.

Le rapport cyclique du signal généré sur OCR0x reste le même que celui du mode Fast PWM mais la fréquence est divisé par deux.

La fréquence du signal généré sur OCR0x est obtenu par :

$$f_{OCR0xPWM} = \frac{f_{T0}}{510} = \frac{\frac{f_{CLK}}{N}}{510}$$

$$f_{OCR0xPWM} = \frac{f_{CLK}}{N.510}$$

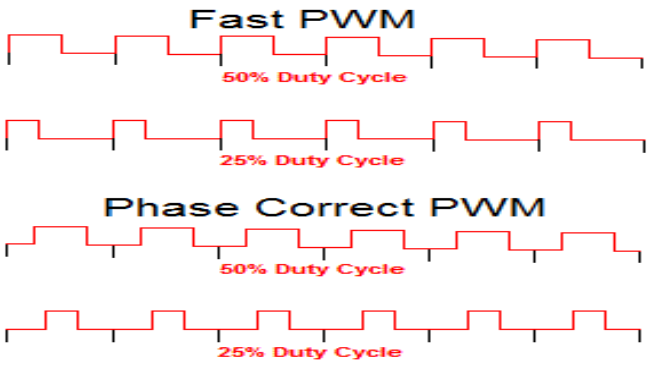
N représente le facteur de division (1, 8, 64, 256 ou 1024).

IX.1.1. Phase correct PWM Mode avec TOP=OCR0x (WGM02=1)

Le mode de Fast PWM contre le mode phase correct PWM

L'inconvénient du mode Fast PWM :

Ce mode ne peut pas être utilisé pour la commande des moteurs parce que sa phase n'est pas correcte.



Dans le mode fast PWM, le niveau haut du signal commence toujours dans les même instants (quelque soit le rapport cyclique), mais il se termine à des instants différents. Donc le centre du niveau haut n'est pas toujours fixe.

The fast PWM HIGH part of the wave always start in the same place as you can see in the picture but the LOW part will end in an arbitrary place, that is defined by the duty cycle, this means that the center of the HIGH part of the wave will not be constant and that's the main and key difference between the two modes and that's why you shouldn't use it in motor control.

Fast PWM mode are used for digital to analog converters and where phase does not matter but higher frequency is needed.

but it also as its short-comings, for example it shouldn't be used as a PWM for motor control because it is phase incorrect, this image will enlighten you about the phase correctness problem:

phase correct mode à pour avantage d'augmenter la résolution du timer, au détriment de la fréquence maximale, qui est divisée par deux.

What to write in the interrupt service routine:

Interrupts can be generated when wrap around occurs, allowing you to count these resets or initiate a timed event.

when that point arrives, you can use the interrupt as an alert, run different code, or change a pin output.

Using normal code techniques, you'd have to set a variable with the next time the LED should blink, then check constantly to see if that time had arrived. With a timer interrupt, you can set up the interrupt, then turn on the timer

Timer resolution:

La résolution du timer (granularité) est définie par la plus petite unité de temps mesurable. Tous simplement, c'est la période du signal horloge du timer. Elle est déterminée par le facteur de division du prescaler.

Résolution du timer = $\frac{1}{f_{ClkT0}} = N \frac{1}{f_{Clk}}$

Le facteur de division du prescaler	La résolution du timer0 pour $f_{clk} = 1MHz$
1	1 µs
8	8 µs
64	64 µs
256	256 µs
1024	1024 µs

Timer Resolution = (1 / Input Frequency)[sec]
the smallest measurable unit of time will be the period of this clock

Each timer can be configured with a different source (internal or external) and a prescaler. The prescaler determines the timer granularity (resolution). A timer with a prescaler of 1 increments its counter every 4 clock cycles - 1,000,000 times a second if using a 4 MHz clock. A timer with a prescaler of 8 increments its counter every 32 clock cycles. It is recommended to use the highest prescaler possible with your application.

Calculating Time Passed
The equation to determine the time passed after counting the number of ticks would be:

delay (in ms) = (# ticks) * 4 * prescaler * 1000 / (clock frequency)

for example . . .

Assume that Timer1 is set up with a prescaler of 8 on a MCU clocked at 20 MHz. Assume that a total of 6250 clicks were counted.

then . . .

delay (in ms) = (# ticks) * 4 * 8 * 1000 / (20000000)

delay (in ms) = (6250) / 625 = 10 ms

Pseudocode
Set up LED hardware
Set up timer in CTC mode
Enable CTC interrupt
Enable global interrupts
Set timer compare value to one second
WHILE forever
END WHILE
ISR Timer Compare
Toggle LED
END ISR