Informatique embarquée

Fiche de synthèse

Les Timers

1/ Généralités

Un Timer qu'est ce que c'est ?

C'est un registre (8 ou 16 bits) qui s'incrémente (ou se décrémente) au rythme d'une horloge. Cette horloge peut être interne au µc (le quartz) ou être câblée sur une entrée externe !

Mais à quoi ça sert ?

Vu que l'on connaît la fréquence de comptage on peut mesurer le temps qui s'écoule. Mais sans le savoir on s'en est déjà servi à maintes reprises ! En effet, les fonctions *millis()* et *delay()* utilisent un timer !

2/ Les Timers du ATmega328P

Le microcontrôleur AVR ATmega328P d'Atmel qui possède 3 timers :

- Le **timer0**, sur 8 bits, utilisé par les fonctions delay(), millis() et micros(). Il commande également des PWM sur les broches 5 et 6.
- Le **timer1**, sur 16 bits, qui compte de 0 à 65535 (0 à FFFF en hexadécimal) et qui est utilisé par la bibliothèque Servo ou bien pour de la PWM sur les broches 9 et 10.
- Le **timer2**, sur 8 bits, qui est utilisé par la fonction Tone() ou bien pour de la PWM sur les broches 3 et 11.

3/ Les registres utilisés par les Timers

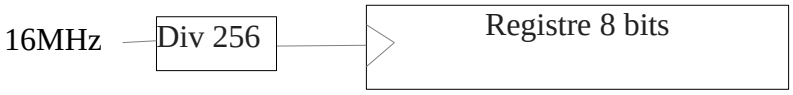
Timer 0	Timer 1	Timer 2	Rôle
TCNT0	TCNT1L	TCNT2	Timer (bit 0 à 7)
-	TCNT1H	-	Timer (bit 8 à 15)
TCCR0A	TCCR1A	TCCR2A	Registre de contrôle
TCCR0B	TCCR1B	TCCR2B	Registre de contrôle
-	TCCR1C	-	Registre de contrôle
OCR0A	OCR1AL	OCR2A	Output Compare (bit 0 à 7)
-	OCR1AH	-	Output Compare (bit 8 à 15)
OCR0B	OCR1BL	OCR2B	Output Compare (bit 0 à 7)
-	OCR1BH	-	Output Compare (bit 8 à 15)
-	ICR1L	-	Input Capture (bit 0 à 7)
-	ICR1H	-	Input Capture (bit 8 à 15)
TIMSK0	TIMSK1	TIMSK2	Interrupt Mask
TIFR0	TIFR1	TIFR2	Interrupt Flag

4/ Prescaler ?

Pour pouvoir avoir une gestion plus fine du comptage du temps qui s'écoule il sera parfois nécessaire d'augmenter ou de diminuer la vitesse de comptage. Sachant que l'on ne peut compter plus vite que l'horloge on peut diminuer cette fréquence de comptage en utilisant un *prescaler* qui signifie en fait diviseur de fréquence.

Soit une horloge de 16MHz combien de temps faut il pour compter de 0 à 0xFF (255) dans un registre 8 bits ?

Idem en utilisant un *prescaler* de 256

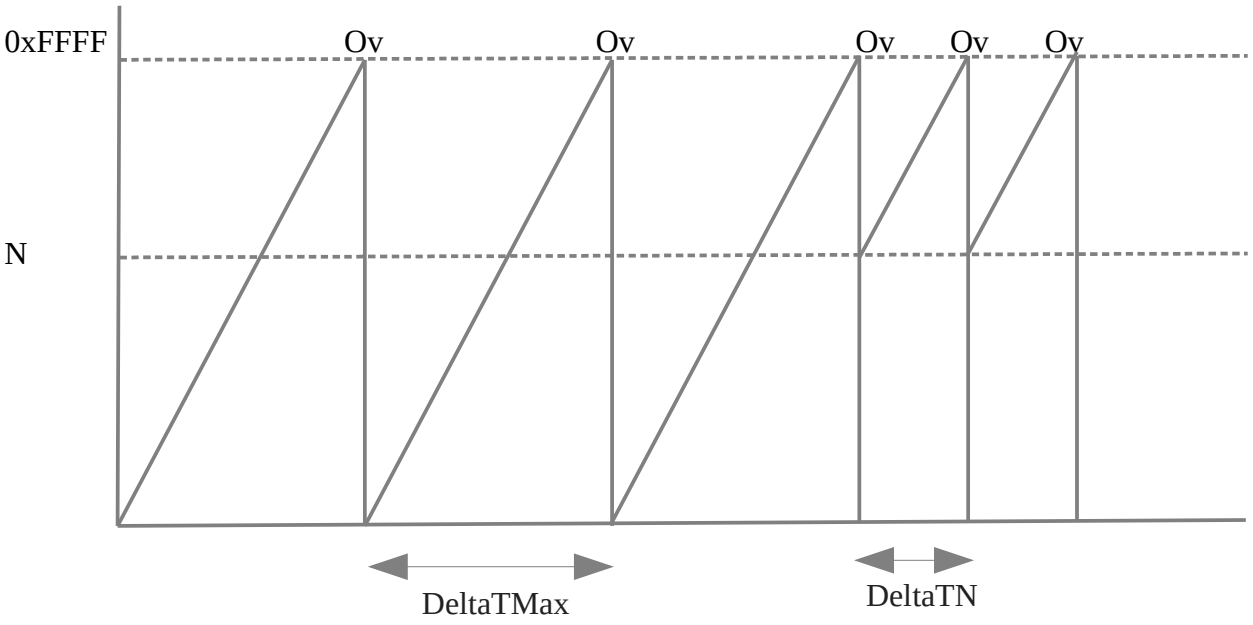


Idem avec un registre 16 bits pour passer de 0 à 0xFFFF (65535) et un *prescaler* de 256

4/ Utilisation en mode « normal »

Dans ce mode, le principe est de détecter le passage du registre de sa valeur maximale à zéro. On appelle ça un overflow.

Sur 8 bits 0xFF + 1 =0 et sur 16 bits 0xFFFF + 1 =0



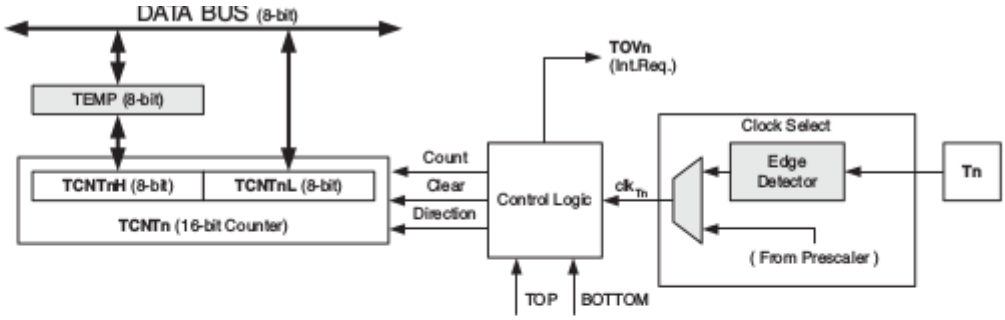
Si après un overflow on recommence à compter à partir de 0 on obtient un intervalle entre les deux oveflow qui est maximal. Cet intervalle de temps peut être réduit en recommençant à compter à partir d'une valeur supérieure à 0 (N dans l'exemple ci-dessus).

En jouant à la fois sur le *prescaler* et sur la valeur de démarrage de comptage on peut gérer très finement ces intervalles de temps qui séparent les *overflow*.

5/ Comment détecter les overflow ?

Il y a plusieurs façon de le faire, le plus simple et le plus efficace étant de générer une interruption à chaque *overflow (*Sinon on peut aussi par *pooling* dans la boucle infinie venir cycliquement tester le bit *d'overflow)*

Configuration du timer1 en mode normal avec IT sur overflow



Signal description (internal signals):

Count	Increment or decrement TCNT1 by 1.
Direction	Select between increment and decrement.
Clear	Clear TCNT1 (set all bits to zero).
clk_{T1}	Timer/Counter clock.
TOP	Signalize that TCNT1 has reached maximum value.
BOTTOM	Signalize that TCNT1 has reached minimum value (zero).

Le mode Normal est sélectionné en positionnant les 4 bits WGM1X à 0. Dans ce mode le registre 16 bits TCNT1 est incrémenté au rythme de l'horloge + *prescaler*. Quand TCNT1 passe de sa valeur maximale à 0 (*overflow*) le bit TOV1 est mis à 1 et doit être remis à 0 par software (en écrivant un 1 dedans!). Dans le cas d'un traitement par interruption, la remise à 0 de TOV1 est automatique et réalisée quand l'interruption associée est exécutée.

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX

TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 16-5. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

Pour autoriser les interruption sur overflow il est nécessaire de positionner le bit TOIE1 à 1.

TIMSK1 – Timer/Counter1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TIFR1 – Timer/Counter1 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

La routine d'interruption s'écrira donc de la manière suivante :

```
// Routine d'interruption
ISR(TIMER1_OVF_vect) {

TCNT1 = …

}
```

6/ Quelques calculs !

Détermination de DeltaT : $DeltaT = Tosc * Prescaler * (2^n - N)$

Avec n=8 ou 16 suivant qu'il s'agit d'un timer 8 ou 16 bits

Attention, il s'agit d'une équation à deux inconnues $Prescaler, N$.

Pour résoudre cette équation, on fixe N=0 et on calcule le prescaler. On prends la valeur immédiatement supérieure parmi celles possibles (1, 8 , 64, 256 ou 1024). Ensuite on calcule la valeur de N.

Exemple : Générer un signal carré de 4Hz.

F= 4Hz => T= ¼ s= 250ms

Il faut inverser un bit de sortie toutes les T/2 c-a-d toutes les 125ms. Par conséquent DeltaT=125ms

Avec N=0 on a $DeltaT = Tosc * Prescaler * 2^n$ donc $Prescaler = DeltaT / (Tosc * 2^n)$

Si Fosc= 16MHz et n= 16 (Timer 1) on a $Prescaler = 30{,}51$

On prends la valeur immédiatement supérieure soit 64.

$N = 2^{16} - DeltaT / (Tosc * Prescaler) = 34285$

C'est cette valeur qu'il faudra recharger dans TCNT 1 à chaque interruption.