Cours BASCOM-AVR

3e partie

Temporisation et interruptions

Burkhard Kainka

La résolution de certains problèmes requiert une temporisation exacte. Les Timer/Counter d'ATmega arrivent à point nommé. Les modèles de Mega8 à Mega32 comportent 3 temporisateurs : Timer 1 avec une capacité de temporisation de 16 bits plus Timer 0 et Timer 2 de 8 bits chacun.

Un coup d'œil au descriptif technique (**figure 1**) donne une idée des multiples possibilités des temporisateurs. Le schéma fonctionnel de Timer 1 semble tout d'abord compliqué. Il ne semble pas évident de farfouiller dans les nombreux réglages quand on programme par exemple en langage assembleur. Avec BASCOM, par contre, l'utilisation d'un temporisateur est un jeu d'enfant.

On dispose de deux possibilités. L'une consiste à appliquer la fréquence interne (directe ou par un prédiviseur) au compteur qui devient alors un Timer. L'autre consiste à utiliser une broche externe (P1 pour Timer 1) comme entrée. Dans ce cas on peut aussi choisir le flanc qui sert au comptage. L'état du compteur peut être lu ou modifié par un accès à TCNT1. Il est possible d'engendrer une interruption lors d'un dépassement de capacité. Pour terminer, le compteur permet d'engendrer un signal modulé en largeur. Voilà pour les applications principales, mais il reste encore bien d'autres modes de fonctionnement spéciaux.

Lecture du temporisateur

Le premier test fait appel au temporisateur 16 bits activé par la fréquence du quartz divisée par 256. Ce réglage ne nécessite qu'une ligne de BASCOM :
Config Timer1 = Timer , Prescale = 256.
Cette ligne active aussi le temporisateur : Start Timer1 n'est plus nécessaire.

Le **listage 1** est celui de ce premier test. Ici aussi, une partie du programme est lancée par « Goto Test1 » pour que les tests suivants puissent être effectués à partir du même texte source. Il suffit de modifier la ligne en « Goto Test2 », etc. et de recompiler.

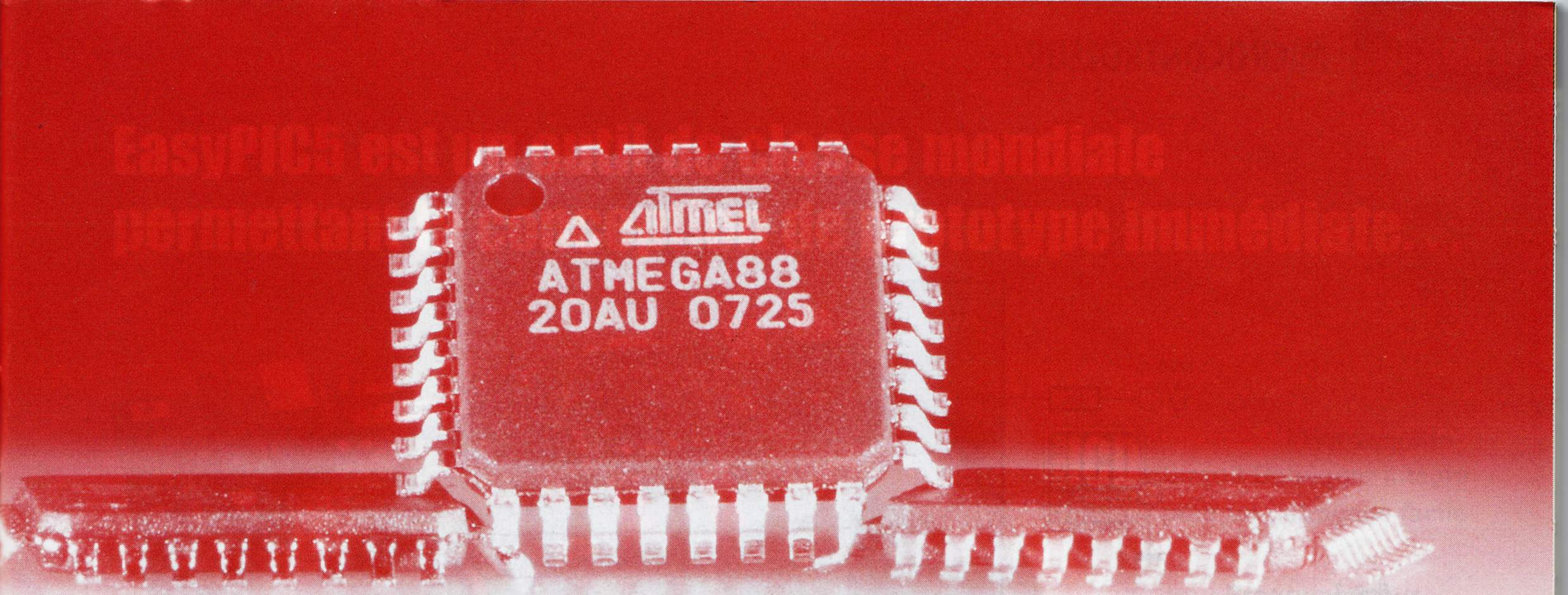
L'exemple Test1 lit l'état du compteur et envoie environ 5 valeurs par seconde au terminal. On voit que les valeurs numériques croissent dans l'intervalle de 0 à 65535 ; un dépassement de capacité se produit environ une fois par seconde :

```
5088
17864
30706
43547
56389
3695
```

En fait, le dépassement de capacité ne se produit pas exactement une fois par seconde mais à un rythme un peu plus lent. La valeur de 16 MHz est tout d'abord divisée par 256. Le temporisateur est donc activé au rythme de 62,5 kHz. Un dépassement de capacité se produit après chaque cycle de 65536. Le dépassement de capacité se répète donc toutes les 1,049 s.
Dans cette application, le temporisateur joue simultanément le rôle d'une horloge de haute précision. On pourrait se demander par exemple quel est le temps nécessaire pour exécuter les 2 lignes suivantes : « Print Timer1 » et « Waitms 200 ». La différence entre 2 états du compteur est par exemple 43547 – 30706 = 12841. Une impulsion dure 1 / 62,5 kHz = 15,267 µs. Le temps d'activation vaut donc 12841 * 15,267 µs = 196 ms. Waitms attend donc un peu moins d'une milliseconde et il vaut mieux l'éviter pour des mesures de temps précises.

Interruption du temporisateur

Comment faire pour « battre la seconde » avec précision ? Il serait presque dommage d'utiliser Timer 1 à cet effet. Les 8 bits de résolution de Timer 0 feront l'affaire. Le temporisateur doit engendrer un dépassement de capacité – donc une interruption – toutes les 1000 µs.
Une interruption (*interrupt*) arrête le programme principal et



appelle un sous-programme. On écrit un sous-programme de traitement des interruptions (*Interrupt Service Routine, ISR*) pour chaque interruption. Dans le cas présent, le sous-programme d'interruption se dénomme Tim0. Il ne s'agit pas ici d'un « Sub ». Tim0_isr: est un Label désignant l'adresse à laquelle le contrôleur doit sauter en cas d'interruption. Le sous-programme d'interruption doit se terminer par Return. L'interruption suivante pourra alors être traitée.

Test 2 initialise Timer 0 avec un prédiviseur de 64. Le temporisateur fonctionne donc à la fréquence de 250 kHz. Comme le compteur comporte 8 bits, un dépassement de capacité se produira après 256 impulsions de comptage si on ne prend pas de mesures particulières. Le compteur, positionné à 6 tout au début du sous-programme d'interruption, subira un dépassement de capacité à chaque cycle de 250. Le rythme est donc exactement d'une milliseconde. La variable « Ticks », longue d'un mot, est incrémentée de 1 lors de chaque dépassement de capacité du temporisateur. Une variable des secondes est incrémentée chaque fois que « Ticks » atteint 1000. Les secondes et millisecondes peuvent être évaluées dans le programme principal. Dans le cas présent, le programme envoie au terminal le nombre de secondes écoulées depuis le début.
Pour que le sous-programme d'interruption puisse être appelé, il faut activer l'interruption globale (Enable Interrupts) ainsi que l'interruption de dépassement de capacité de Timer 0 (Enable Timer0). Les secondes croissantes apparaissent alors au terminal :

```
0
1
2
3
```

On peut inversement désactiver toutes les interruptions avec Disable Interrupts.

Moyenne des mesures

Les mesures analogiques sont souvent faussées par la superposition de perturbations à 50 Hz. Le calcul de la moyenne peut améliorer les choses. On annule, ou tout au moins affaiblit considérablement la composante de 50 Hz, en effectuant la moyenne des mesures sur une période de 20 ms exactement.

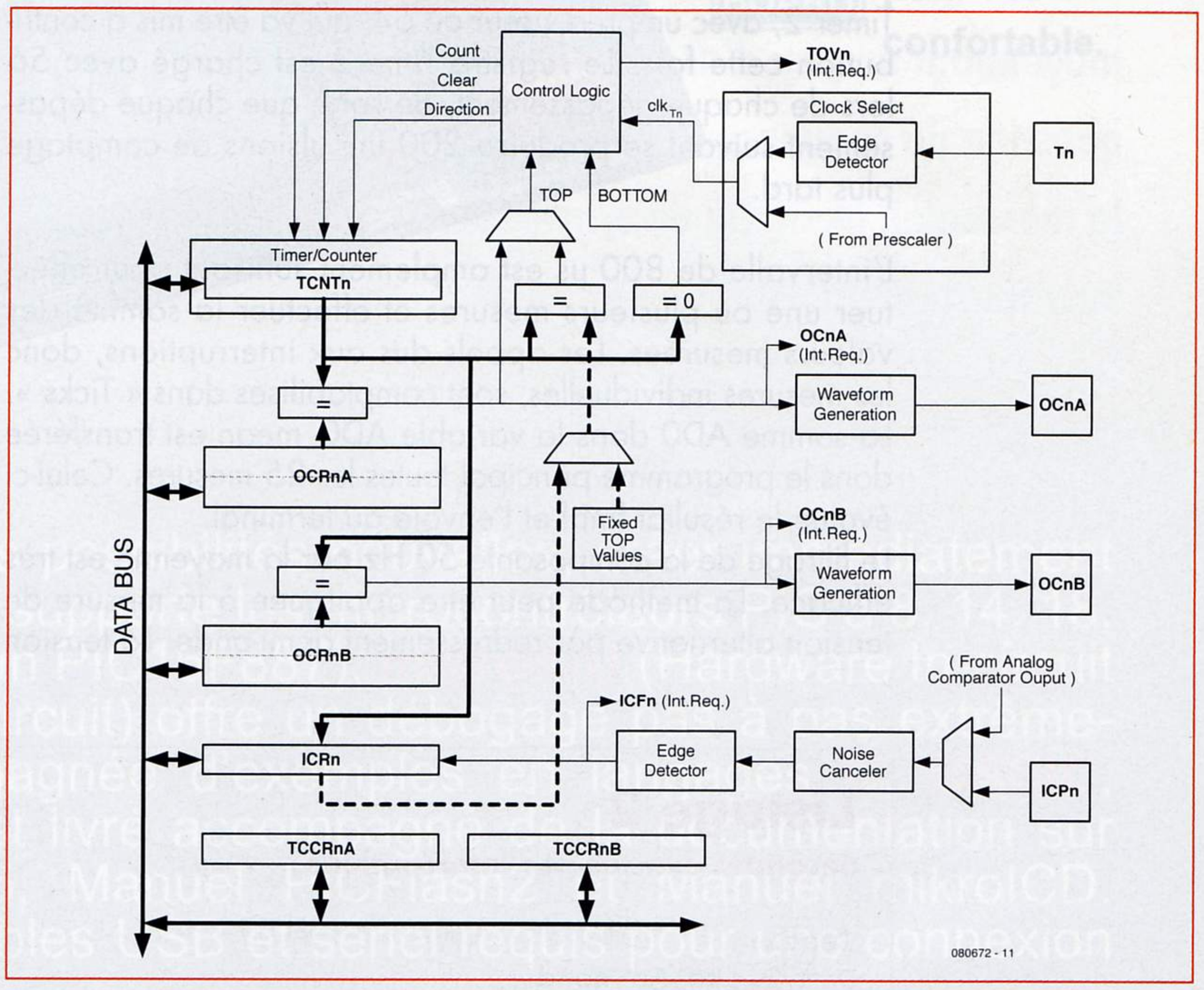


Figure 1. Schéma fonctionnel du temporisateur.

Listage 1

Lecture du registre de temporisation

```
,Bascom ATmega88, Timer
$regfile = „m88def.dat“
$crystal = 16000000
Baud = 9600
Goto Test1

Test1:
Config Timer1 = Timer , Prescale = 256
,Start Timer1
Do
  Print Timer1
  Waitms 200
Loop
```

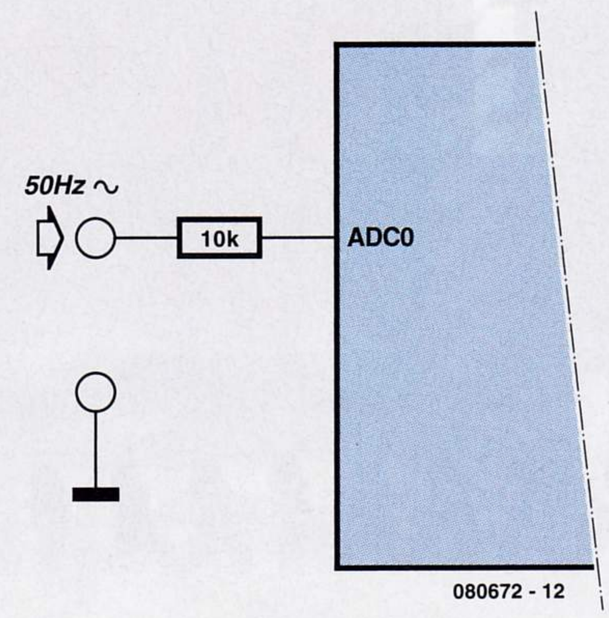


Figure 2. Mesure d'une tension alternative.

Dans ce cas aussi, une interruption du temporisateur assurera le déroulement exact des opérations dans le temps. Il faut effectuer la moyenne de 25 mesures réparties sur 20 ms. La mesure est donc répétée toutes les 800 µs. C'est Timer 2, avec un prédiviseur de 64, qui va être mis à contribution cette fois. Le registre Timer2 est chargé avec 56 lors de chaque dépassement, de sorte que chaque dépassement suivant se produira 200 impulsions de comptage plus tard.

L'intervalle de 800 µs est amplement suffisant pour effectuer une ou plusieurs mesures et effectuer la somme des valeurs mesurées. Les appels dus aux interruptions, donc les mesures individuelles, sont comptabilisés dans « Ticks ». La somme AD0 dans la variable AD0_mean est transférée dans le programme principal toutes les 25 mesures. Celui-ci évalue le résultat final et l'envoie au terminal.
Le filtrage de la composante 50 Hz par la moyenne est très efficace. La méthode peut être appliquée à la mesure de tension alternative par redressement demi-onde. La tension alternative est appliquée directement à l'entrée de mesure à travers une résistance de protection de 10 k (**figure 2**). Le programme détermine la moyenne des demi-ondes positives qui correspond à la moitié de la moyenne des valeurs absolues. On trouve par exemple à la sortie :

```
226
227
226
226
226
```

La valeur est égale à 226, hormis de légères fluctuations. Une valeur de 226 correspond à une tension moyenne de 5 V * 226 / 1023 = 1,10 V. La moyenne absolue de la tension alternative mesurée est donc de 2,20 V. La valeur efficace est de 2,44 V si la tension est sinusoïdale. Cela équivaut à une valeur de crête de 3,46 V. Le rapport entre la tension de crête et la tension efficace est de $\sqrt{2} = 1{,}414$ pour une tension alternative sinusoïdale. Le rapport entre la tension de crête et la tension moyenne est de $\pi/2 = 1{,}571$ quand la moyenne arithmétique est utilisée. Autrement dit, la moyenne affichée est égale à 90,03 % de la valeur efficace.

(080672-1e)

Listage 2

Secondes exactes par interruptions

```
Test2:
Dim Ticks As Word
Dim Seconds As Word
Dim Seconds_old As Word
Config Timer0 = Timer , Prescale = 64
On Ovf0 Tim0_isr
Enable Timer0
Enable Interrupts

Do
  If Seconds <> Seconds_old Then
    Print Seconds
    Seconds_old = Seconds
  End If
Loop

Tim0_isr:
  '1000 µs
  Timer0 = 6
  Ticks = Ticks + 1
  If Ticks = 1000 Then
    Ticks = 0
    Seconds = Seconds + 1
  End If
Return
```

Listage 3

Calcul de la moyenne

```
Test3:
Dim Ad0 As Word
Dim Ad0_mean As Word
Config Adc = Single , Prescaler = 64 , Re-
   ference = Off
Config Timer2 = Timer , Prescale = 64
On Ovf2 Tim2_isr
Enable Timer2
Enable Interrupts

Do
  Ad0_mean = Ad0_mean / 25
  Print Ad0_mean
  Waitms 100
Loop

Tim2_isr:
  '800 µs
  Timer2 = 56
  Ticks = Ticks + 1
  Ad0 = Ad0 + Getadc(0)
  If Ticks > 24 Then
    Ticks = 0
    Ad0_mean = Ad0
    Ad0 = 0
  End If
Return
```

Téléchargements et infos :

Vous trouverez sur *www.elektor.fr* une page web se référant à chaque série de cours et permettant de ***télécharger le logiciel*** nécessaire.