Cours BASCOM-AVR

(5) Mémoire, interrogation de commutateurs et gestion du temps

Burkhard Kainka

On ne dispose jamais de trop de mémoire et de puissance de calcul. Mais ce sont précisément les ressources des microcontrôleurs qui sont souvent « un peu justes ». Il faut donc en tirer le meilleur parti possible. Cela s'applique particulièrement aux boucles de programme exécutées sans interruption.

Il est toujours important de savoir à quoi le contrôleur passe son temps de calcul. Il suffit par exemple d'une petite boucle discrète d'interrogation de touches ou poussoirs pour dévorer l'intégralité du temps de calcul. Il importe donc de développer un concept basé sur l'utilisation optimale du temps de calcul. Cette partie du cours BASCOM illustre quelques possibilités de programmation optimisée.

RAM et EEPROM

L'ATmega88 possède 1024 octets de RAM et 512 octets d'EEPROM. C'est un bon début ! BASCOM utilise la RAM pour les variables et autres piles. Que reste-t-il ? Testons en enregistrant des données dans un tableau. La variable A(500) est un tableau. Il s'agit donc de 500 octets A(1) à A(500). Attention : A(0) n'existe pas. Le petit programme de test dans **Listage 1** écrit des données ascendantes en mémoire. Elles sont lues dans une seconde boucle et envoyées au PC.

Le fichier Memory.rpt, qui indique le degré d'occupation mémoire, permet de déterminer à quel point on s'approche dangereusement de la limite. Il s'agit d'un fichier texte qui peut être ouvert avec Notepad. Il contient la taille mémoire, la position exacte de toutes les variables et bien plus encore.

Test 2 illustre les possibilités de l'EEPROM qui, contrairement à la RAM, conserve les données lors de la mise hors tension. Les données sont mémorisées de façon permanente par `Writeeeprom, Variable, Adressememoire` et lues par `Readeeprom, Variable, Adressememoire`. Toutes les adresses d'une EEPROM effacée ne contiennent que FF (255). On peut donc lire un octet et déterminer si son contenu a déjà été modifié. Dans Test 2 (**Listage 2**), 512 octets sont écrits et relus.

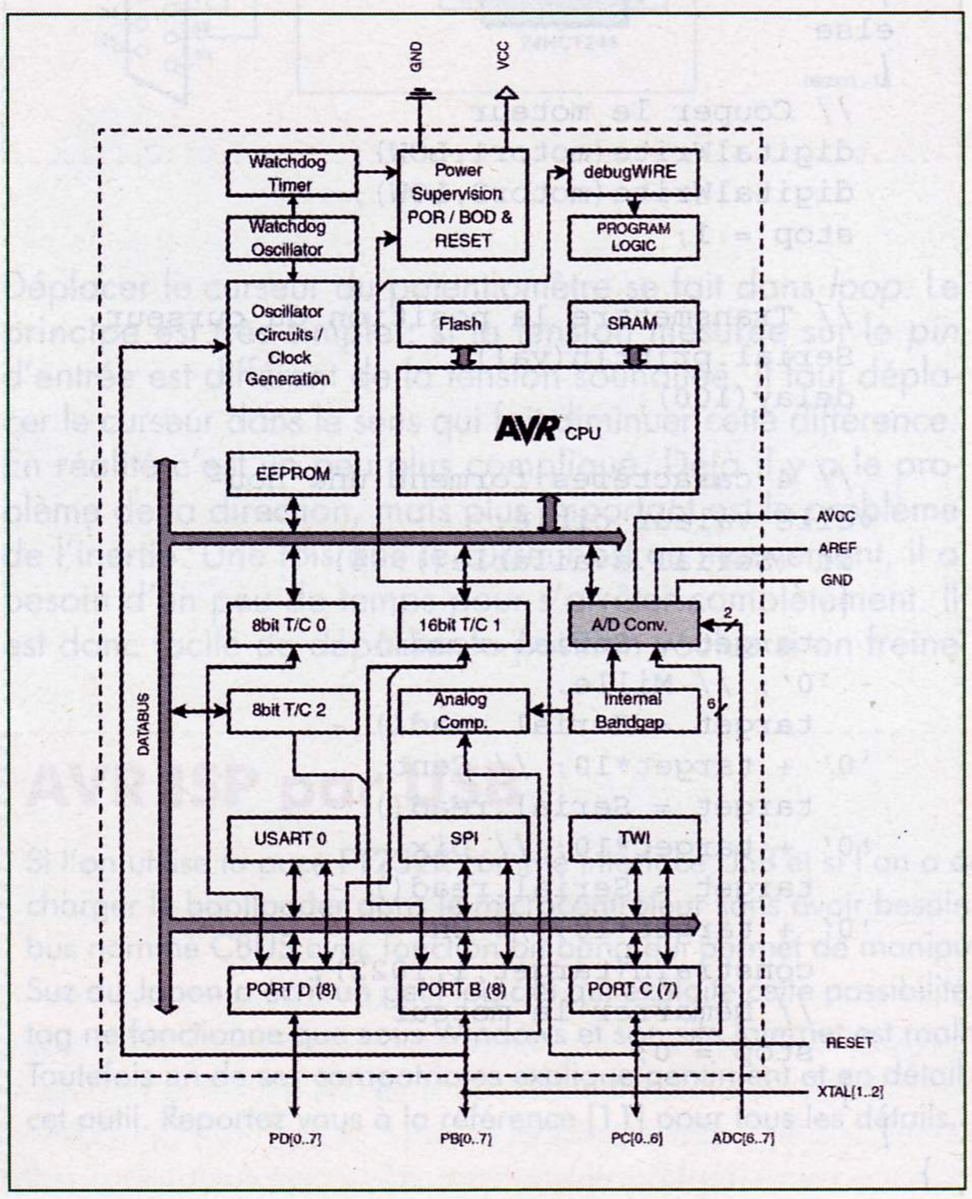

Figure 1. Schéma fonctionnel de l'ATmega88.

Interrogation de touches

Les appareils autonomes sont souvent équipés de touches de commande qui doivent être interrogées et évaluées selon des règles exactement définies. Quoi de plus simple que d'interroger des touches ? Cette tâche, apparemment anodine, recèle toutefois un certain nombre de pièges. L'un d'entre eux résulte de l'ignorance dans laquelle on se trouve quant à la fréquence et à la durée de la pression sur la touche. Il faudrait donc interroger continuellement la touche utilisée. L'introduction de fonctions par touches dans un programme déjà opérationnel ne constitue pas exactement un jeu d'enfant. Il vaut mieux partir des touches pour réaliser pas à pas les fonctions désirées.

Le rebondissement fréquent des touches constitue un autre piège. Les rebondissements d'une touche pressée se chargent de simuler plusieurs événements. L'interrogation des touches sur la base d'une temporisation ne constitue donc

Listage 1 Déposer des données dans la RAM

```
Test1:
Dim A(500) As Byte
Dim N As Word
Do
  For N = 1 To 500
    A(n) = Low(n)
  Next N
  For N = 1 To 500
     Print A(n)
     Waitms 100
  Next N
Loop
```

Listage 2 L'EEPROM

```
Test2:
  For N = 0 To 511
    Writeeeprom N , N
  Next N
Dim D As Byte
Do
  For N = 0 To 511
     Readeeprom D , N
     Print N , D
     Waitms 100
  Next N
Loop
```

Listage 3 Commutation de DEL

```
Test3:
S1 Alias Pind.6
S2 Alias Pind.5
S3 Alias Pind.7
Out1 Alias Portd.2
Out2 Alias Portd.3
Config Portd = &B00001100
Portd.6 = 1
Portd.5 = 1
Portd.7 = 1

Out1 = 1
Do
  If S1 = 0 Then
    Out1 = 1
    Out2 = 0
    Print «1 on»
  End If
  If S2 = 0 Then
    Out1 = 0
    Out2 = 1
    Print «1 off»
  End If
  Waitms 50
Loop
```

Listage 4

Commande élévatrice/abaisseuse d'une sortie PWM

```
Test4:
Dim Pwmold As Integer
Pwma = 0
Do
  If S1 = 0 Then Pwma = Pwma + 1
  If Pwma > 1023 Then Pwma = 1023
  If S2 = 0 Then Pwma = Pwma - 1
  If Pwma < 0 Then Pwma = 0
  If S3 = 0 Then Pwma = 0
  Waitms 20
  Pwm1a = Pwma
  If Pwma <> Pwmold Then
  Print Pwma
  End If
  Pwmold = Pwma
Loop
```

pas une solution. Il faut élaborer une stratégie antirebond efficace. La plus simple consiste à introduire une commande d'attente. Le rebondissement d'une touche n'aura aucun effet si elle n'est interrogée qu'une fois toutes les 10 ms.

L'essai suivant est basé sur trois touches D5 à D7. Les bits correspondants du port sont placés à l'état haut. Les bits de direction des données restent nuls. Cela commute les résistances de charge internes. Une touche non pressée produit 1, une touche pressée 0. Des alias permettent de renommer les bits du port. On peut donc interroger la touche 1 par `If S1 = 0 then` (**Listage 3**).

Test 3 utilise deux touches de commutation. S1 active la première sortie, S2 la seconde. Chaque pression sur une touche s'accompagne d'un message au PC. La touche est interrogée de nouveau après un délai de 50 ms. Une pression prolongée de la touche ne modifie plus les états de sortie, mais envoie des messages supplémentaires par l'interface sérielle. Il faut avouer qu'on est encore loin du but : le microcontrôleur consacre toute sa puissance de calcul à l'interrogation des touches.

Test 4 (**Listage 4**) comporte 2 touches commandant un signal MLP (PWM) à OC1A=PB1. Une touche pour augmenter la valeur MLP, une touche pour la diminuer. Un oscilloscope nous permet d'observer la longueur d'impulsion engendrée. La luminosité d'une DEL raccordée varie en conséquence. Ici aussi, la puissance de calcul du microcontrôleur est mise à rude épreuve. L'antirebond des touches n'est pas un problème dans cette application car le seul facteur qui compte est le temps total de pression sur la touche.

Test 5 (**Listage 5**) n'utilise qu'une seule touche pour allumer/éteindre une DEL. Quand une touche est pressée, le programme attend dans une boucle que la touche soit relâchée. Pendant ce temps, le programme est aveugle et sourd à toute autre pression sur la touche. L'antirebond joue ici un rôle important. Il faut en effet que la pression sur la touche soit interprétée comme un seul événement. Faute d'un délai, la sortie serait commutée plusieurs fois à chaque pression

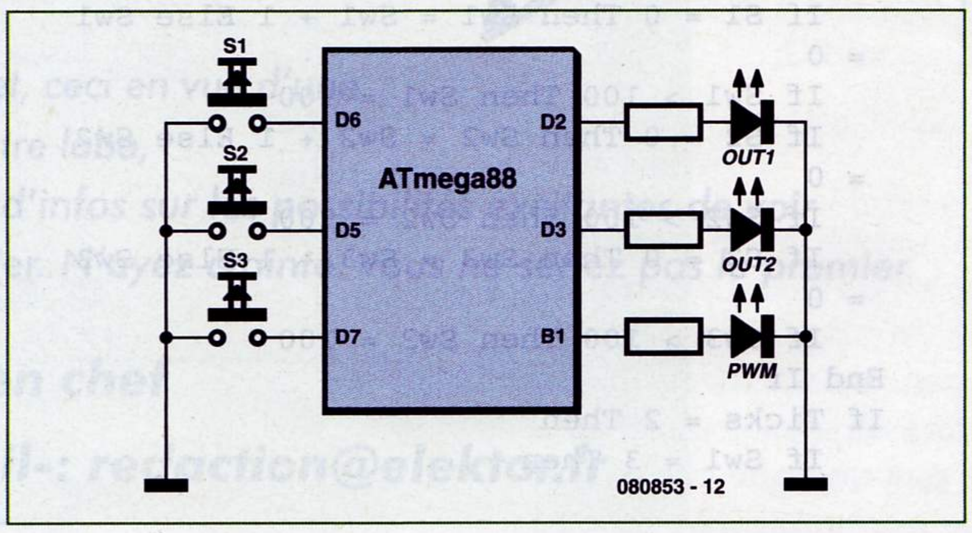

Figure 2. Entrées et sorties.

Listage 5 Une double bascule bistable

```
Test5:
Do
   If S1 = 0 Then
     If Out1 = 0 Then
       Out1 = 1
     Else
       Out1 = 0
     End If
     Waitms 10
   End If
   Do
   Loop Until S1 = 1
   If S2 = 0 Then
     If Out2 = 0 Then
       Out2 = 1
     Else
       Out2 = 0
     End If
     Waitms 10
   End If
   Do
   Loop Until S2 = 1
   Waitms 100
Loop
```

Listage 6

Un double compteur d'événements

```
Test6:
Dim Count1 As Word
Dim Count2 As Word
Do
  If S1 = 0 Then
    Count1 = Count1 + 1
    Print «Count1 «;
    Print Count1
    Waitms 50
    Do
    Loop Until S1 = 1
  End If
  If S2 = 0 Then
    Count2 = Count2 + 1
    Print «Count2 «;
    Print Count2
    Waitms 50
    Do
    Loop Until S2 = 1
  End If
Loop
```

Listage 7 Interrogation des touches par interruption

```
Test7:
Dim Ticks As Byte
Dim Sw1 As Byte
Dim Sw2 As Byte
Dim Sw3 As Byte
Dim Sw4 As Byte
Dim Pwm1 As Integer
Dim Pwm1old As Integer
Dim Ledtimer As Byte
Dim Ledblink As Byte

Ledblink = 1
Enable Timer0
Enable Interrupts
Cls
Lcd 0

Do
  If Ticks = 1 Then Out1 = 1
  If Ticks = 5 Then Out1 = 0
Loop

Timer0isr:
  Ticks = Ticks + 1
  If Ticks = 1 Then
     If S1 = 0 Then Sw1 = Sw1 + 1 Else Sw1
   = 0
     If Sw1 > 100 Then Sw1 = 100
     If S2 = 0 Then Sw2 = Sw2 + 1 Else Sw2
   = 0
     If Sw2 > 100 Then Sw2 = 100
     If S3 = 0 Then Sw3 = Sw3 + 1 Else Sw3
   = 0
     If Sw3 > 100 Then Sw3 = 100
  End If
  If Ticks = 2 Then
     If Sw1 = 3 Then
       Pwm1 = Pwm1 + 1
       If Pwm1 > 1023 Then Pwm1 = 1023
     End If
     If Sw1 = 100 Then
       Pwm1 = Pwm1 + 1
       If Pwm1 > 1023 Then Pwm1 = 1023
     End If
     If Sw2 = 3 Then
       Pwm1 = Pwm1 - 1
       If Pwm1 < 0 Then Pwm1 = 0
     End If
     If Sw2 = 100 Then
       Pwm1 = Pwm1 - 1
       If Pwm1 < 0 Then Pwm1 = 0
     End If
     If Pwm1 <> Pwm1old Then
       Print Pwm1
     End If
     Pwm1a = Pwm1
     Pwm1old = Pwm1
  End If
  If Ticks = 3 Then
     If Sw3 = 3 Then
        If Ledblink = 1 Then
          Ledblink = 0
        Else
          Ledblink = 1
        End If
     End If
  End If
  If Ticks = 4 Then
     Ledtimer = Ledtimer + 1
     If Ledtimer > 100 Then Ledtimer = 0
     If Ledtimer = 1 Then
       If Ledblink = 1 Then Out2 = 1
     End If
     If Ledtimer = 50 Then Out2 = 0
  End If
  If Ticks = 10 Then Ticks = 0
Return
```

sur la touche pour aboutir finalement à un état aléatoire.
Le déroulement du Test 5 permet aussi de commander 2 compteurs. Les pressions sur les touches S1 et S2 sont comptées. Le PC est informé de chaque modification d'un compteur. Le problème de la charge à 100 % du contrôleur n'est toujours pas résolu.

Interrogation des touches par interruption du temporisateur

Tous les exemples d'interrogation de touches présentés jusqu'ici, s'ils en illustrent bien le principe, ne gèrent pas efficacement le temps de calcul. Quand il faudra gérer plusieurs touches et d'autres fonctions, il sera grand temps de trouver une méthode de gestion du temps plus performante. Test 7 (**Listage 7**) illustre une solution plus efficace. Les touches sont interrogées à l'arrière-plan par un sous-programme d'interruption de temporisation. Le programme principal peut se consacrer accessoirement à d'autres tâches.
Une variable Sw1 à Sw3 est associée à chaque touche S1 à S3. La variable associée à une touche est nulle tant que cette touche n'est pas pressée. La valeur de la variable croît continuellement jusqu'à 100 quand la touche est pressée. On peut donc déterminer à tout moment si une touche a été pressée et, si tel est le cas, pendant combien de temps. Pour évaluer une pression particulière sur une touche on teste si sa valeur, par exemple 3, est associée au déclenchement d'un événement. La valeur 100 correspond à une pression prolongée.
Le sous-programme de temporisation contient un second compteur prévu pour les petites unités de temps : les « Ticks ». La variable Ticks varie de 1 à 10. Les touches ne sont donc interrogées qu'à chaque dixième appel (lorsque Ticks=1). Les problèmes de rebond sont donc accessoirement résolus et le programme est encore assez rapide pour ne rater aucun événement.
Les événements peuvent être évalués à des instants différents. Dans l'exemple, une valeur PWM est modifiée lorsque le temps Ticks = 2 et le clignotement d'une DEL est activé ou désactivé lorsque le temps Ticks = 3. Ticks = 4 gère le clignotement proprement dit. Ce partage du temps de calcul donne une impression de simultanéité à l'utilisateur, mais le contrôleur dispose encore de grandes réserves.
Le programme principal commande la sortie 1. Il l'active pendant 5 Ticks et la désactive pendant 5 Ticks. Une DEL raccordée clignote si rapidement qu'on n'observe qu'une diminution de la luminosité. La luminosité de la DEL ne varie pas, indiquant que le rythme de fonctionnement du programme est constant.
La touche de commande de la sortie MLP possède deux modes dans cet exemple : une courte pression et une pression prolongée. Une courte pression permet de modifier la valeur d'une unité. Une pression prolongée provoque un changement continu du compteur. Il est donc plus facile d'atteindre une valeur particulière.

(080853-I, trad. Softcraft)

Téléchargements :

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