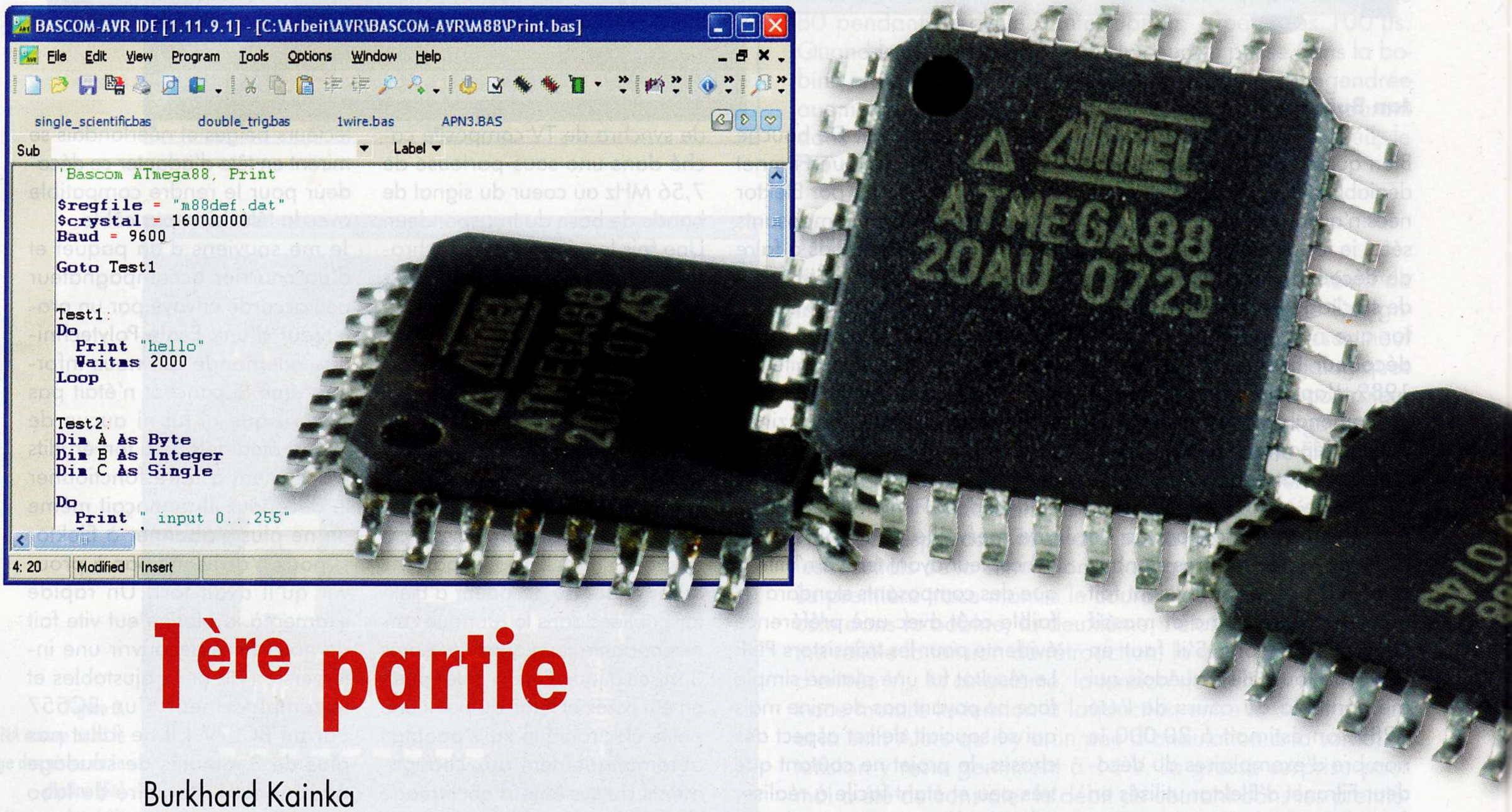
Cours BASCOM AVR

Programmez un contrôleur ATmega



1ère partie

Burkhard Kainka

Les contrôleurs AVR s'enorgueillissent de leur large popularité. Elektor vous a d'ailleurs déjà proposé pas mal de réalisations et de cartes pour ATmega. Nous allons aujourd'hui nous focaliser sur leur programmation. L'objectif de ce bref cours, c'est de vous présenter tout le nécessaire pour attiser votre créativité. Et précisément BASCOM est un outil idéal, rondement maîtrisé, pour vous emmener au plus vite à vos premières réussites.

Le contrôleur ATmega et BASCOM forment une équipe gagnante ! Quel que soit le projet, ATmega a déjà tout ce qu'il faut à bord. Ports, temporisateur, convertisseur A/N, sorties PWM et interface sérielle, RAM, ROM, EEPROM, tout est disponible à volonté. Et la réalisation, BASCOM en fait un jeu d'enfant. Même les périphériques complexes comme écran LCD, RC5 et I²C seront commandés en quelques instructions.

Question matériel, l'offre est abondante. Pour le cours, vous pouvez aussi bien vous servir du STK500 d'Atmel ainsi que du système ATM18 du projet CC2 d'Elektor ou d'une des nombreuses cartes d'autres sources. Une construction personnelle sur platine perforée peut également convenir tout en vous inspirant des exemples de programmes de ce cours-ci. Il n'y a pas de grande différence si vous utilisez un Mega8, un Mega88 ou tout autre contrôleur tel que le Mega16 ou le Mega32. Ce qui change se situe au niveau des ports disponibles et de la grandeur de la mémoire interne. Dans la suite de ce premier cours, nous allons nous intéresser à l'UART et au convertisseur A/N du contrôleur.

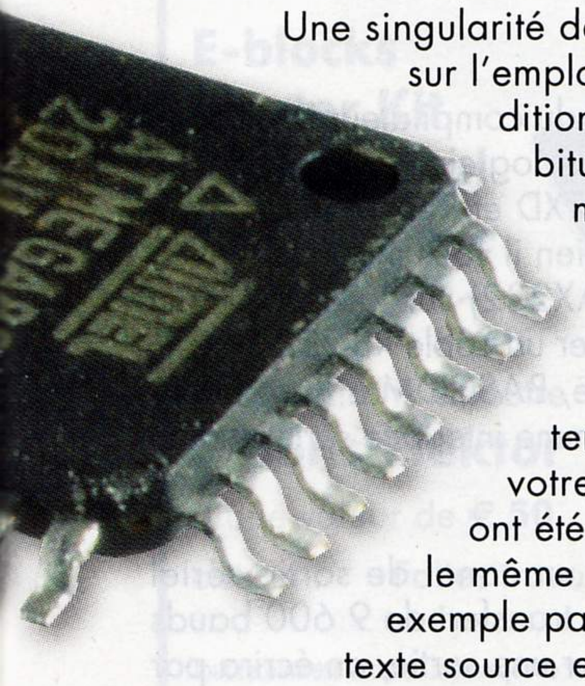
Interface sérielle

Tous les contrôleurs ATmega disposent d'une interface sérielle (UART) avec les lignes RXD (PD0) et TXD (PD1). Les signaux y sont compatibles TTL. On se sert donc d'une

puce de mise au bon niveau comme le MAX232, dont le STK500 est équipé, ou d'un adaptateur USB comme sur la platine de test de l'ATM18. Il suffit d'un programme de terminal correspondant sur le PC et la communication avec l'ATmega peut déjà débuter. C'est ce que nous allons faire en premier, il y a là le moyen de tester des programmes simples et de les corriger.

Le **listage 1** présente les éléments nécessaires à tout projet BASCOM. Par $regfile = «m88def.dat», on indique le modèle de contrôleur utilisé, un ATmega88 dans le cas présent. On peut aussi omettre cette ligne et désigner le contrôleur via Options/Compiler/Chip. Toutefois, l'expérience apprend que cette méthode conduit souvent à des erreurs quand on change de système AVR. Mieux vaut placer en tête de chaque programme, de manière bien visible, l'instruction qui supplante alors le contenu du champ Options.

Il ne faut pas oublier non plus d'indiquer la fréquence du quartz ($crystal = 16000000 pour 16 MHz). L'instruction produira ses effets deux lignes plus loin. Par la même occasion, on peut aussi définir la fréquence de la transmission sérielle (BAUD = 9600). Tous les temps de pause se déduisent de la fréquence d'horloge (Waitms 2000 pour 2 s).



Une singularité de ce programme de test repose sur l'emploi de l'instruction de saut inconditionnel Goto Test1. On évite d'habitude le Goto dans une programmation bien structurée. Mais ici, il s'utilise à dessein pour une raison spéciale. Comme ce mini cours contient de nombreux petits exemples, pour éviter de mettre le capharnaüm sur votre disque dur, tous les exemples ont été rassemblés en ribambelle dans le même code source. Pour exécuter un exemple particulier, il suffira de modifier le texte source en Goto Test2, 3, 4 et ainsi de suite puis de recompiler pour exécuter chacune des parties l'une après l'autre.

L'exemple de programme du premier test contient une boucle sans fin. Entre Do et Loop se situent les instructions qui produiront toutes les deux secondes une nouvelle sortie, dont on peut recevoir le texte avec un programme de terminal.

Calcul d'une équation

Le programme suivant (**listage 2**) fournit la superficie d'un cercle quand on lui en donne le rayon : $C=A^2*3.1415$. Pour A, seuls les nombres entiers sont acceptés. On a pris ici une variable octet A dont la valeur est introduite par l'interface sérielle. Le nombre A, dans l'intervalle 0 à 255, est multiplié par lui-même. Le produit B peut donc être contenu dans un mot (entre 0 et 65 535). Pour le résultat final, il faudra un nombre réel. Le produit C est donc défini comme Single, ce qui correspond à un espace mémoire de quatre octets. Attention cependant ! Même si ce calcul peut sembler simple, il ne faut pas oublier une particularité. Dans d'autres dialectes du Basic, on a l'habitude d'écrire tout le calcul sur une seule ligne (C = 3.1415 * A * A, ou même Print 3.1415 * A * A). En BASCOM, ce n'est pas possible, parce que les équations en chaîne ne sont pas permises. Il convient donc de séparer le calcul complet en expressions simples.

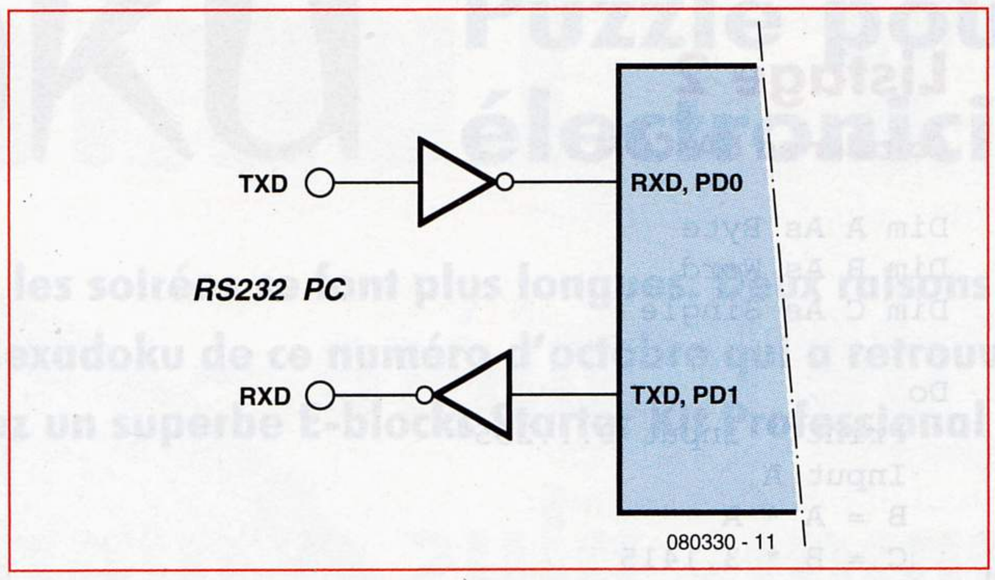


Figure 1. L'interface sérielle.

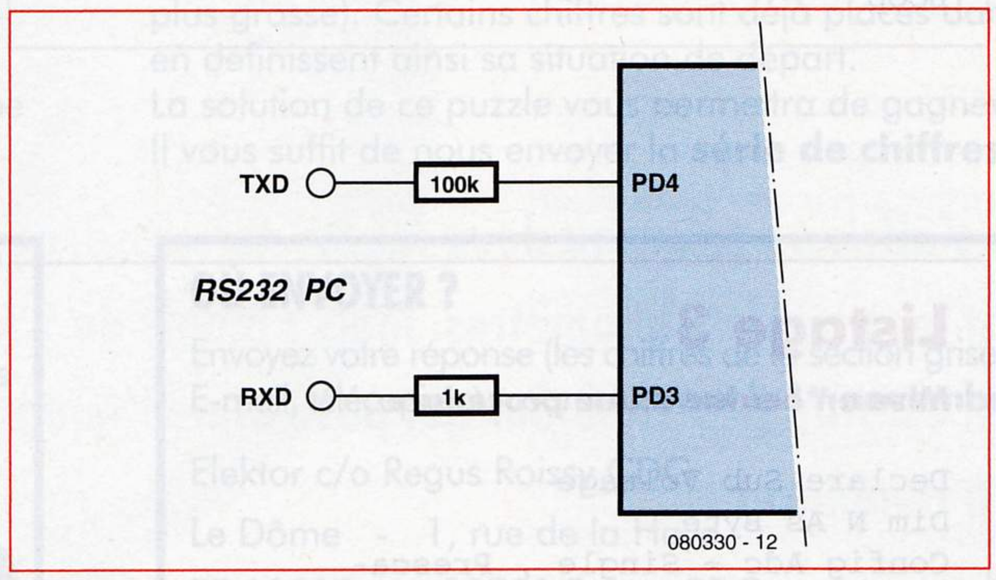


Figure 2. L'interface sérielle la plus simple sans inversion.

Procédures

Le convertisseur A/N de l'ATmega dispose d'une résolution de 10 bits. Comme l'indique le **listage 3**, on doit l'initialiser avant de s'en servir :

Config Adc = Single , Prescaler = 64 , Reference = Off.

Ici, le convertisseur est scandé à 1/64e de la fréquence d'horloge, donc à 250 kHz en partant de 16 MHz. La référence interne est débranchée, on utilise alors la tension présente sur la broche AREF. Dans la plupart des cas, il s'agit de la tension d'alimentation stabilisée de 5 V.
On a utilisé dans cet exemple une procédure pour la mesure

Listage 1

Print „hallo"

```
,Bascom ATmega88, Print
$regfile = „m88def.dat"
$crystal = 16000000
Baud = 9600

Goto Test1

Test1:
Do
  Print „hello"
  Waitms 2000
Loop

Test2:
…
Test3:
…
End
```

Listage 2

Calculer en BASCOM

```
Dim A As Byte
Dim B As Word
Dim C As Single

Do
  Print " input 0...255"
  Input A
  B = A * A
  C = B * 3.1415
  'not allowed: C = 3.1415 * A * A
  Print C
Loop
```

Listage 3

Mise en oeuvre d'une procédure

```
Declare Sub Voltage
Dim N As Byte
Config Adc = Single , Presca-
   ler = 64 , Reference = Off
Start Adc

Do
  N = 0 : Voltage
  Print „ADC(0) = „ ; U ; „ V" „
  N = 1 : Voltage
  Print „ADC(1) = „ ; U ; „ V" „
  Print
  Waitms 500
Loop

Sub Voltage
  D = Getadc(n)
  D = D * 5
  U = D / 1023
End Sub
```

Listage 4

Utilisation de l'UART logiciel

```
Baud = 9600
,Open „comd.3:9600,8,n,1" For Output As #2
Open „comd.3:9600,8,n,1,INVER
   TED" For Output As #2
Config Adc = Single , Presca-
   ler = 64 , Reference = Off

Do
  N = 0 : Voltage
  Print #2 , „ADC(0) = „ ; U ; „ V" „
  N = 1 : Voltage
  Print #2 , „ADC(1) = „ ; U ; „ V" „
  Print #2 ,
  Waitms 500
Loop
```

et la conversion de la tension. Comme chacun sait, une procédure s'indique tout particulièrement quand la même opération se répète à différents endroits d'un programme. Dans ce cas-ci, on s'occupera de deux tensions. La nouvelle procédure se nomme Sub Voltage. Avant de pouvoir y faire appel, il faut la déclarer dans BASCOM : Declare Sub Voltage.

Le style utilisé ici s'écarte volontairement de l'orthodoxie. En Visual Basic, on aurait vraisemblablement passé le canal N par un appel à la procédure : Voltage(N). Ou encore aurait-on construit une fonction pour ensuite écrire lors de l'appel : U = Voltage(N). Ici, au contraire, nous ne travaillons qu'avec des variables globales. D, N et U sont valables dans tout le programme, il est donc parfaitement possible qu'une autre procédure y fasse appel. Ce n'est peut-être pas un style de programmation particulièrement heureux, mais il décharge considérablement le contrôleur et le compilateur. L'expérience nous enseigne que même les programmes les plus longs qui utilisent jusqu'à 100% de la Flash sur un Mega32 tournent de manière absolument stable et sans difficulté avec quantité de variables globales. Mais dès qu'il faut transmettre des variables à des procédures ou des fonctions, il peut se produire de petites erreurs qui seront bien difficiles à dépister après coup.

L'interface sérielle logicielle

L'une des nombreuses prouesses du compilateur BASCOM réside dans son interface sérielle logicielle. Vous utilisez déjà l'interface matérielle avec TXD et RXD et il vous faudrait un autre port COM ? Ou bien il vous manque l'inverseur d'interface (comme un MAX232) sur la carte, mais vous voulez malgré tout y brancher un câble RS-232 ? Dans une situation comme dans l'autre, BASCOM a la solution. Vous pouvez en effet déclarer comme interface sérielle n'importe quelle ligne de port.

L'exemple du **listage 4** utilise un canal de sortie sériel sur PD3. Il dispose d'un débit de transfert de 9 600 bauds sous le numéro d'interface 2. Pour une sortie, on écrira par exemple Print #2, « salut ».

On peut aussi, à l'aide du paramètre supplémentaire INVERTED, se passer de l'adaptateur d'interface. BASCOM inverse alors la polarité du signal, si bien que le niveau de repos est bas. Du coup, le MAX232 est au chômage, il suffit d'une liaison directe avec la ligne RXD de l'interface du PC. C'est une option utilisable par exemple avec la carte de test ATM18 qui a renoncé au MAX232 au profit de l'adaptateur USB.

(080330-I)

Téléchargements et infos :

En complément de ce cours, vous trouverez sur le site Internet www.elektor.fr les fichiers à télécharger du logiciel et d'autres informations. N'hésitez pas non plus à communiquer vos réactions sur redaction@elektor.fr.