PIC atout AXE (4)
édition et affichage *en série*

Wouter Spruit
(Pays-Bas)

Grâce à cette série d'articles Elektor.POST vous savez désormais comment programmer un PICAXE et construire des interfaces analogiques et numériques entre des circuits et un PICAXE. Vous avez aussi appris à vous servir du PICAXE pour commuter et commander des dispositifs, (re)découvert le calcul des valeurs des composants d'un circuit, vu diverses utilisations de la modulation de largeur d'impulsion (MLI), positionné des pièces à l'aide de servomoteurs, et même créé une mélodie (aviez-vous reconnu l'air, au fait ?) [1] Dans ce dernier article, nous allons relier un afficheur OLED et un clavier PS/2 à un PICAXE, puis connecter le tout à un PC via une liaison série.

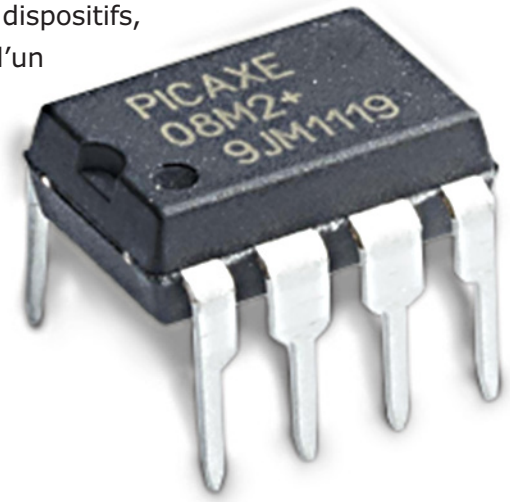


Figure 1.
Brochage du PICAXE 18M2.

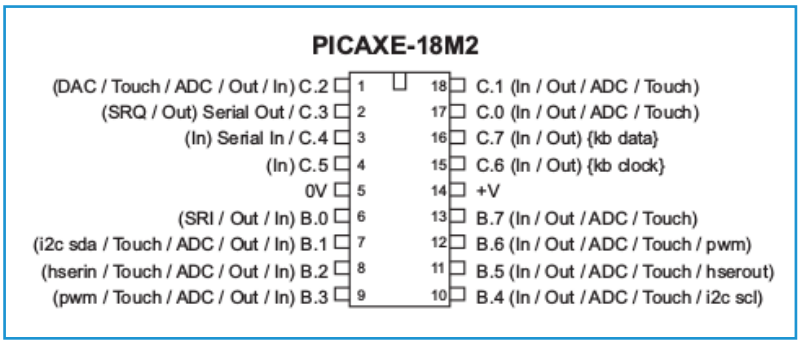
Déjà vu — déjà entendu

En lieu et place du PICAXE 08M2 précédemment utilisé, nous allons nous servir ici du PICAXE 18M2 (cf. **fig. 1** pour son brochage). Nous l'alimentons encore par les 5 V d'une alimentation ATX, et les circuits sont ici aussi assemblés sur des plaques d'essai sans soudure. Notez que l'adaptateur pour carte de prototypage (AXE029) utilise la configuration « 18 » du cavalier, configuration qui aligne les connexions avec les broches de téléchargement des 18M2. Le câble USB-série (**fig. 2**) ne servira pas qu'à programmer la puce, il servira aussi d'interface série entre le PICAXE et un PC.

La puce a été programmée avec le logiciel LinAXEpad, version 1.5.0 pour Linux (Arch). Pour éviter toute confusion, les numéros de broche des schémas se réfèrent à la numérotation physique sur la puce, et dans les listages les numéros de broche font référence aux noms internes des broches (fig. 1). Reportez-vous aux articles précédents [1] ou au manuel PICAXE [2] si vous avez oublié la façon de programmer un PICAXE, et au deuxième article si vous souhaitez revoir les méthodes de calcul des composants. Vous pouvez vous procurer les puces PICAXE et différents périphériques chez *Revolution Education* [3].

Communication série

La plupart des microcontrôleurs peuvent communiquer par liaison série. Le terme série signifie que les données sont envoyées et reçues par séquences de bits sur un seul fil. La norme RS-232 est communément utilisée pour les communications série (ex. : le port série d'un PC). Les spécifications RS-232



indiquent des tensions de ±15 V, mais la plupart des contrôleurs modernes ignorent ces valeurs et ne suivent le protocole que pour le codage et la synchronisation des données. Le MAX232 est un exemple de CI qui fournit une « vraie » communication RS-232, y compris les tensions. Les commandes série des PICAXE utilisent des modes différents pour les communications série ordinaires (le débit débute par un N) et pour les « vraies » RS-232 (le débit débute par un T).

Une communication série avec un dispositif tel qu'un PC ou un Raspberry Pi permet d'intégrer un projet PICAXE à d'autres projets, le PICAXE servant de tampon pour les circuits d'entée et de sortie qui y sont reliés. Le paragraphe suivant explique comment paramétrer une connexion série, mais vous pouvez passer directement aux exemples pratiques. La théorie est néanmoins utile pour savoir configurer correctement des connexions entre différents éléments d'un projet.

Le PICAXE connaît deux sortes de commandes d'entrée/sortie séries. La première n'est utilisée que pour les broches connectées (habituellement) au câble série : `sertxd` pour la sortie, `serrxd` pour l'entrée. La deuxième série de commandes `serin` et `serout` sert à la communication série avec les broches qui la prennent en charge. Si nous utilisons ici le 18M2, c'est en raison de l'absence de broches `serin/serout` et `kbdata/kbclock` sur le 08M2.

Une connexion série se configure au moyen de paramètres qui décrivent les caractères à recevoir et la façon de les interpréter. Les données sont transmises dans des trames, c'est-à-dire dans des séquences de bit constituées de : 1 bit de départ, 5 à 9 bits de données binaires, 1 bit de parité (éventuel), et 1 (ou plus) bit(s) d'arrêt. Les paramètres d'une connexion série indiquent à la machine le nombre de trames par seconde (le débit en bauds), le nombre de bits de données par trame, si un bit de parité est présent ou non (« n » pour non), et le nombre de bits d'arrêt. La connexion série depuis et vers un PICAXE est définie par : 8 bits de données, aucun bit de parité, et 1 bit d'arrêt par trame. Le débit doit être spécifié pour les commandes `serout/serin`. Pour les communications via

Figure 2.
Dans la série « Le retour » : le câble USB-série.

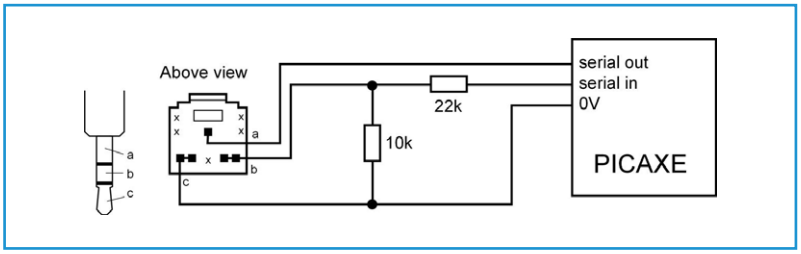


Figure 3.
Pour mémoire : comment connecter le câble de téléchargement.

les broches de téléchargement, le débit est fixé à 4800 bauds (à l'exception des éléments X2, dont le débit est fixé à 9600 bauds).

Connexion au PC

Nous avons déjà expliqué comme brancher le câble de téléchargement dans les articles précédents [1]. Pour rappel, les schémas du manuel PICAXE sont également reproduits sur la **figure 3**. Commencez par établir une connexion série avec votre PC — pour télécharger les programmes dans le PICAXE, mais aussi pour faire communiquer vos programmes avec l'ordinateur. La commande `sertxd` permet d'envoyer des données au PC sans configuration supplémentaire. Pour voir les données envoyées à travers cette connexion, vous avez besoin d'un émulateur de terminal ; l'éditeur LinAXEpad en contient un dans le menu *PICAXE/Terminal...* (F8). Les données enregistrées dans un registre du PICAXE, p. ex. *b0*, sont transmises en tant que données binaires brutes et interprétées en tant que code ASCII [4]. Pour envoyer la valeur stockée dans *b0* non pas en tant que valeur ASCII brute mais en tant que texte (c.-à-d. sous forme d'une séquence de caractères codés en ASCII), utilisez *#b0* (ça ne fonctionne toutefois pas toujours avec les versions obsolètes du PICAXE). Le code du **listage 1** demande au PICAXE d'incrémenter la valeur d'un registre à l'intérieur d'une boucle, puis envoie ladite valeur, d'abord sous forme

de nombre interprété (c.-à-d. sous la forme d'une séquence de caractères ASCII lisibles), ensuite en tant qu'ASCII brut. Puisque seules certaines parties des caractères ASCII sont du texte lisible, les autres caractères sont interprétés comme caractères spéciaux, non imprimables, ou encore de contrôle. Les valeurs « 13 » et « 10 » envoyées par la commande `sertxd` sont des caractères de contrôle ASCII brut qui indiquent respectivement un retour

Listage 1 : PC_OUT

```
main:
b1=0
do
  b1=b1+1
  sertxd("The value of #b1 is ",#b1,13,10) 'interpret  number as text
  sertxd("The value of b1 is ",b1,13,13,10) 'interpret number as ASCII
  pause 1000
loop
end
```

Listage 2 : Loopback

```
init:
disconnect 'PICAXE no longer scans for program downloads
main:
do
  serrxd [10000,timeout],b0  'wait 10 seconds for input, then goto timeout
  sertxd("character received: ",b0,13,10)
  if b0 = "q" then
    sertxd("q received, type quit to exit",13,10)
    serrxd [5000,timeoutmain],("quit")
    goto quit
  endif
loop

quit:
reconnect
sertxd("quit received. program done." ,13,10)
end

timeout:
reconnect
sertxd("Input timed out",13,10)
'goto main ‘only uncomment this if you know
'how to reprogram the unconnected device!
sertxd("Downloading of programs re-enabled.",13,10)
end

timeoutmain:
sertxd("no quit received within 5 sec. restarting program.",13,10)
goto main
```

chariot et un saut de ligne. Ces deux caractères placent donc le curseur au début d'une nouvelle ligne.
Recevoir des données par la broche de téléchargement exige une configuration supplémentaire, car cette broche est utilisée en permanence par le micrologiciel du PICAXE (elle scrute les nouveaux téléchargements). Nous devons ainsi la « déconnecter » pour qu'elle puisse recevoir des données sérielles. Aucun nouveau programme ne pourra être téléchargé tant qu'elle n'aura pas été reconnectée, ou tant qu'une initialisation matérielle ou logicielle n'aura pas été effectuée. Sur les PICAXE récents, les commandes d'entrée série disposent d'un *timeout* (délai d'attente). Le **listage 2** montre comment activer et désactiver l'entrée des données via le câble de téléchargement, l'utilisation des *timeouts*, et comment recevoir une commande depuis le PC. Ouvrons donc un émulateur de terminal (F8) pour communiquer avec le PICAXE.

Entrées par le clavier

Les claviers PS/2 envoient des données via une connexion série, mais ont besoin d'un signal d'horloge. Les **figures 4 et 5** montrent le brochage d'un connecteur PS/2. Plusieurs PICAXE ont des broches marquées *kb data* et *kb clock* (broches 16 et 15 sur le 18M2) qui servent à connecter un clavier PS/2. La commande `kbin` attend les données d'un clavier (avec *timeout*), tandis que `kbled` permet de commander les LED *Arr déf*, *Verr num* et *Verr maj* du clavier. Notez que `kbled` stoppe l'exécution du programme jusqu'à ce qu'un clavier soit branché. Les données du clavier sont envoyées sous forme de codes *scan*. Ces codes sont décrits dans la deuxième partie du manuel PICAXE, page 131 [5]. La plupart des caractères sont envoyés comme données série codées sur 8 bits.

Branchez votre clavier selon le schéma de la **figure 5**. Les claviers USB ont depuis longtemps remplacés les PS/2, et hormis d'occasion il est difficile d'en trouver. Si vous n'avez pas déniché de connecteur PS/2, vous pouvez toujours en bricoler un, par exemple en coupant le connecteur PS/2 du câble du clavier et en reliant de façon appropriée les quatre fils. Le programme du **listage 3** reçoit une entrée du clavier et l'envoie au PC via le câble de téléchargement. F8 ouvre un émulateur de terminal depuis le logiciel de programmation de PICAXE.

Ajouter un afficheur

Cet exemple utilise l'AXE033Y, un afficheur série/I²C de 16x2 caractères, à relier à la broche *hserial* du PICAXE. Il dispose de 7 bancs de mémoire pour l'affichage de textes pré-programmés, ainsi que d'un module qui lui permet de fonctionner comme une horloge autonome. Nous ne l'utiliserons ici que comme simple afficheur. En théorie, l'exemple fonctionnera avec d'autres modèles d'afficheurs série. Les données sont là encore envoyées à l'afficheur via une connexion série et sous la forme de caractères ASCII. Plusieurs codes ASCII sont non pas des caractères affichables mais des commandes. Vous trouverez un résumé de ces caractères affichables et de ces caractères de commande aux pages 32 et 41 de la partie 3 du manuel PICAXE [6].

Connectez votre afficheur série selon le schéma de la **figure 6**. Avant de programmer le PICAXE avec un code d'affichage relativement complexe, assurez-vous avec le **listage 4** que votre afficheur fonctionne correctement. Le code initialise l'afficheur et envoie le message « Hello Elektor ! »

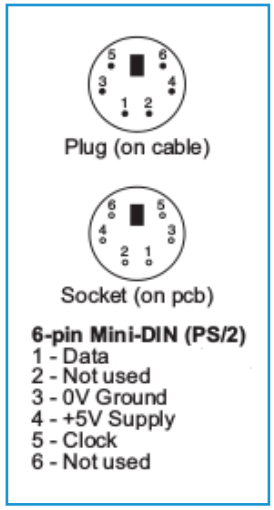


Figure 4.
Brochage d'un connecteur de clavier PS/2.

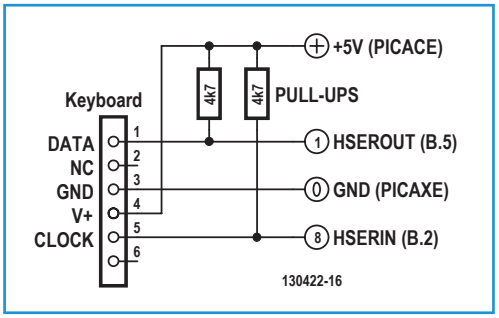


Figure 5.
Comment connecter un clavier.

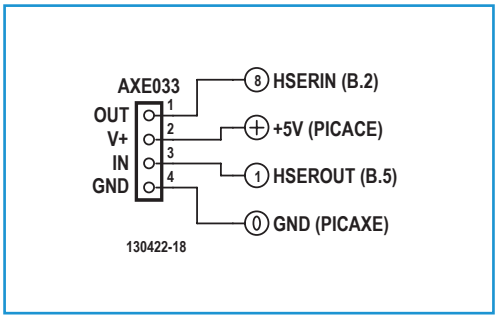


Figure 6.
Branchement de l'afficheur.

Listage 3 : Data from keyboard to PC

```
main:
do
  kbin b1' get keyboard input
  sertxd("Received scancode from PS/2 keyboard: ",#b1,13,10) 'interpret
number as text
loop
end
```

Listing 4 : Hello ~~World~~ Elektor!

```
pause 500         'allow screen to initialize first
serout B.5,N2400,(254,1)   'send clear command to screen
pause 30        'wait for the screen
serout B.5,N2400,(254,128)   'move screen cursor to line 1, char 1
serout B.5,N2400,("Hello Elektor!") 'send text to display
end
```

Le curseur est positionné à l'aide de séquences d'échappement. Chaque fois qu'un caractère est envoyé vers l'afficheur, la position du caractère est automatiquement incrémentée. Certains caractères spéciaux, comme « retour arrière », doivent toutefois être codés différemment afin que l'afficheur réagisse correctement à l'entrée du clavier. Notez que les codes *scan* et ASCII ne sont pas directement compatibles ! Le **listage 5** montre comment convertir une entrée du clavier afin de l'afficher à l'écran ; tous les caractères/touches dont vous avez besoin doivent être convertis (les codes ne sont pas compatibles séquentiellement !) Connectez le clavier et l'afficheur comme décrit précédemment. Les seuls caractères autorisés sont a, b, c, h, i et « espace » ; une fonction backspace (retour arrière) est présente ; les touches Entrée et Echap servent respectivement à redémarrer et arrêter le programme.

Listing 5 : KEYBOARD_DISPLAY

```
init:
pause 500  'allow screen to initialize first
gosub initscreen
serout B.5,N2400,("Enter text:")
gosub setcursor
gosub showcursor
do
  kbin b1
  if b1 = $1C then 'a
    b2=97   'convert to lower case ASCII a
    gosub printcharacter
  elseif b1 = $32 then 'b
    b2=98
    gosub printcharacter
  elseif b1 = $21 then 'c
```

```
    b2=99
    gosub printcharacter
  elseif b1 = $33 then 'h
    b2=104
    gosub printcharacter
  elseif b1 = $43 then 'i
    b2=105
    gosub printcharacter
  elseif b1 =  $29 then 'space
    b2=32
    gosub printcharacter
  elseif b1 = $66 then 'backspace
    gosub backspace
  elseif b1= $76 then 'escape quits the loop
    goto quit:
  elseif b1= $5A then 'enter resets the program
    goto init:
  endif
loop

quit:
gosub initscreen
gosub hidecursor
serout B.5,N2400,("It is now safe to turn off the PICAXE")
do
  serout B.5,N2400,(254,24) ' move the window left
  pause 100
loop
end

initscreen:
serout B.5,N2400,(254,1) 'clear screen
pause 30
serout B.5,N2400,(254,128) 'move to start of first line
symbol CURSOR_POS = b3
CURSOR_POS = 0
return

setcursor:
b4=CURSOR_POS+192   '192: start of second line
serout B.5,N2400,(254,b4)
return

showcursor:
serout B.5,N2400,(254,14)  ‘turn on cursor
return

hidecursor:
serout B.5,N2400,(254,12)  'turn off cursor
```

```
return

printcharacter:
if CURSOR_POS > 15 then return endif  'only use visible character space
serout B.5,N2400,(b2)
pause 200
CURSOR_POS=CURSOR_POS+1
return

backspace:
if CURSOR_POS = 0 then return endif 'already start of line
CURSOR_POS=CURSOR_POS-1
gosub setcursor 'move back one space
serout B.5,N2400,(32) 'overwrite with space character
gosub setcursor 'set cursor to correct position
pause 100
return
```

La mémoire de l'afficheur comprend deux lignes de 40 caractères, mais seuls les 16 premiers caractères de chaque ligne sont affichés. L'effet de défilement peut être obtenu par déplacement de la fenêtre de lecture relativement au texte contenu en mémoire, comme le fait la fonction `quit` du listage 5. Il m'est bien sûr impossible ici d'explorer avec vous toutes les fonctions et options dont sont dotés les afficheurs. Les possibilités offertes par l'AXE033 sont décrites dans le manuel [7].

Paramétrage du terminal

Pour une pleine intégration du PICAXE dans un système existant (ou nouveau), p. ex. une application domotique, le PC et l'utilisateur doivent pouvoir communiquer avec le PICAXE via une liaison série, sans avoir à passer par le terminal de l'éditeur de programmation. L'émulateur *minicom* [8] tourne sous Linux et autres systèmes POSIX (comme Mac OSX) ; cela dit tout le monde n'a pas Linux (ou un Mac). Pour l'utiliser, vous devez vous assurer que le (module pour le) câble de téléchargement série est en service. LinAXEpad vous aidera à activer le bon module pour le câble USB-série (*view → options → port → AXE027 modprobe*) et vous indiquera l'adresse du périphérique (probablement */dev/ttyUSB0*). Pour installer *minicom* sur une distribution de type Debian (y compris sur le SE Raspbian du Raspberry Pi), entrez `sudo apt-get install minicom` (ou `sudo pacman -S minicom` si vous êtes sous Arch). Pour configurer *mimicom*, entrez : `sudo minicom -s`. Sélectionnez l'option *Serial port setup* (touches de direction, entrée), et tapez « A » pour entrer le chemin d'accès du câble USB de téléchargement (*/dev/ttyUSB0* dans mon cas). Tapez « E » pour modifier le débit, et servez-vous des touches A et B pour sélectionner 4800 bauds. Assurez-vous que *minicom* est paramétré avec 8 bits de données, aucune parité, et 1 bit d'arrêt (8N1). Revenez au menu principal avec Entrée, et sélectionnez l'option *dfl* (*default*) comme paramètre d'enregistrement. Après avoir sélectionné *Exit* (PAS *Exit from minicom*), vous pouvez entrez `minicom` pour communiquer avec le PICAXE (ou le lancer avec `sudo minicom`). Pour quitter *minicom* : Ctrl-A, puis X, puis YES (et Entrée). Montrons comment utiliser la communication avec un PICAXE dans un programme ou un script PC. Nous allons supposer ici que vous êtes sous l'interpréteur bash de Linux, et que la connexion série avec le PICAXE fonctionne. Toutes les commandes exigent un droit d'administrateur (*root*). Commençons par paramétrer la connexion. Entrez `stty -F /dev/ttyUSB0 speed 4800 cs8 -cstopb -parenb`. L'utilitaire `cat` permet de voir les données : `cat /dev/ttyUSB0`. Pour lire une seule ligne contenue dans une variable : `read variable < /dev/ttyUSB0`. Pour voir son contenu : `echo $variable`. Enfin, pour envoyer des données via la connexion : `echo q > /dev/`

ttyUSB0 (bien que la commande quit puisse être envoyée et reçue via *minicom*, elle ne semble pas fonctionner avec echo). Le script bash du **listage 6** reprend ces commandes pour communiquer avec le PICAXE (programmé avec le code du listage 2 dont la ligne goto main de la sous-routine timeout a été décommentée). N'oubliez pas de rendre le script exécutable (avec chmod +x nom_du_script.sh).

Et maintenant : over 2 U !

Voilà, vous savez désormais comment faire dialoguer un PICAXE avec différents périphériques via une connexion série, et surtout savez comment visualiser des données sur un afficheur OLED et interpréter les entrées d'un clavier PS/2. Disposer d'une communication bidirectionnelle avec un PC grâce au câble USB-série permet de déboguer des programmes PICAXE depuis l'éditeur de programmation PICAXE (ou logiciel équivalent). Avec une connexion série et un PICAXE, vous pouvez contrôler vos interfaces électroniques depuis un PC, un Raspberry Pi, ou même une interface web. Le système PICAXE permet de construire rapidement et à moindre coût des systèmes complexes, domotiques ou encore robotiques. À l'exception évidente du code et des fonctions propres au logiciel PICAXE, les principes exposés dans cette série peuvent être appliqués à d'autres microcontrôleurs. La page *PICAXE Support Site* de l'Elektor. LABS [9] contient entre autres ressources l'ensemble des listages des projets. Vous avez l'atout PIC en main, à vous de jouer !

(130422 – version française : Hervé Moreau)

Liens

[1] PIC atout AXE (1, 2 et 3), projets Elektor. POST no 8, 16 et 19. www.elektor-magazine.com/extra/post

[2] www.picaxe.com/Getting-Started/PICAXE-Manuals

[3] www.techsupplies.co.uk/PICAXE

[4] www.asciitable.com

[5] www.picaxe.com/docs/picaxe_manual2.pdf

[6] www.picaxe.com/docs/picaxe_manual3.pdf

[7] www.picaxe.com/docs/axe033.pdf

[8] http://linux.die.net/man/1/minicom

[9] www.elektor-labs.com/PICAXE

Listing 6 : BASH_SERIAL

```
#!/bin/bash
#run as root.
export DEVICE='/dev/ttyUSB0'  #the serial device to use
#initialize
echo setting up device $DEVICE
stty -F $DEVICE speed 4800 cs8 -cstopb -parenb
echo waiting for input...
read INPUT < $DEVICE
echo received: $INPUT
sleep 1
echo sending a "q" character
echo q > $DEVICE
echo waiting for response...
read INPUT < $DEVICE
sleep 1
echo response: $INPUT
echo done
```