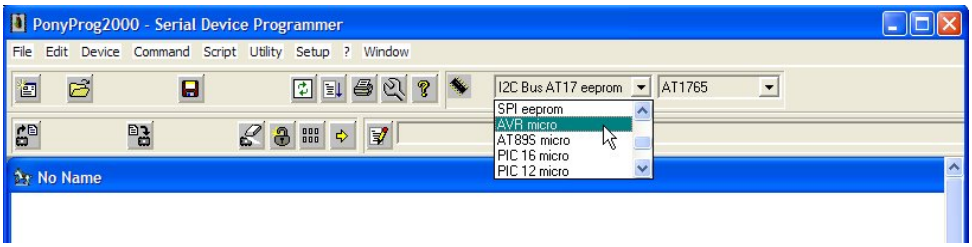
Le soft de programmation est disponible sur http://www.lancos.com/prog.html , puis « download ». La version utilisée ici est la 2.06f Beta.

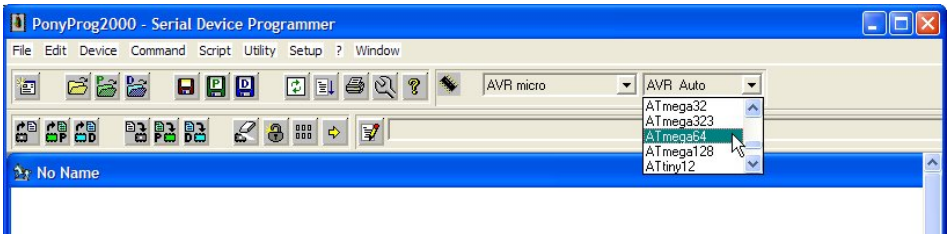
Avant de lancer PonyProg, connecter le programmateur sur le port parallèle de votre PC, **sans relier le programmateur à la platine à programmer.**

Au premier lancement, il va falloir lui configurer quelques trucs si on veut que ça fonctionne…

On va commencer par lui dire sur quel microcontroleur on veut travailler. Pour ça, il suffit d'aller dans le menu déroulant à gauche et de choisir « AVR Micro » :

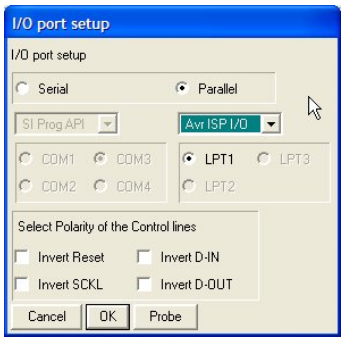


Ensuite, de la même façon, on lui précise quel µC particulier on utilise. C'est le menu déroulant juste à côté, et dans notre cas, c'est « ATMega64 » qui nous convient :



Puis il faut aller lui dire quelle interface de programmation on utilise. On clique sur la petite icône en forme de clé pour accéder à cette option :

et ça nous ouvre une boîte de dialogue :



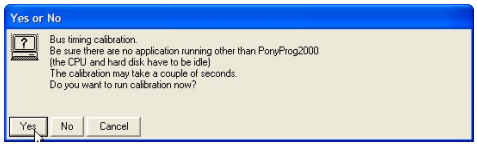
Il faut choisir :
- Parallel,
- AVR ISP I/O, et
- LPT1,

et c'est tout.
Bref, il faut reproduire l'image ☺.

Normalement, ces réglages sont mémorisés, et on ne devrait pas avoir à y retoucher. Cependant, avant de l'utiliser, il va falloir que le soft calibre le timing l'interface pour programmer en toute sécurité.

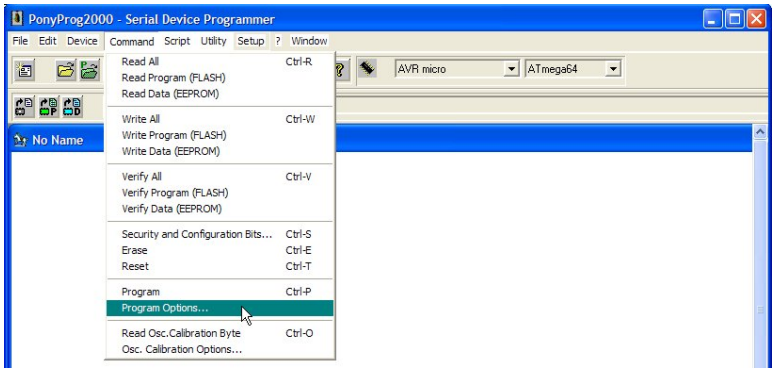
Apapeur, ça se fait tout seul : on va dans le menu « Setup », et « Calibration » :

Ca nous ouvre une petite fenêtre :

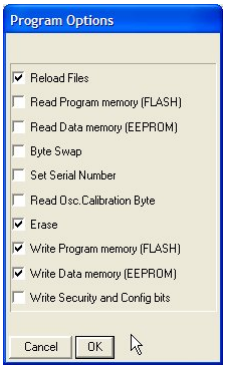


Pour continuer, il est préférable de fermer toutes les autres applications Windows qui mangent du CPU ou des accès disques, et on peut cliquer sur « Yes »… Un court instant plus tard, il doit fièrement vous annoncer que la calibration est OK.

Une dernière chose avant de pouvoir commencer, plus par confort et paresse que pour sa réelle utilité, de façon à s'automatiser la programmation des µC. On va dans le menu « Command » puis « Program Options » :



Ca nous ouvre un petit popup avec des cases à cocher. Cocher ce qu'il faut pour que ça ressemble exactement à ça :

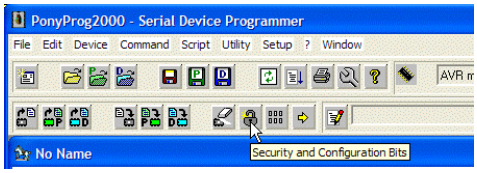


Eh ben on est prêt à programmer… On y va ?

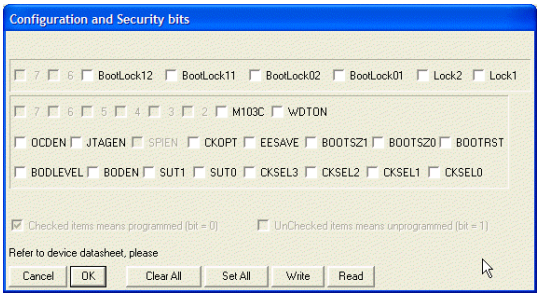
Le programmateur est alimenté en tension seulement par le circuit qu'on veut programmer. Alors on met le jus sur ce circuit-là, et si rien ne fume, on relie le circuit au programmateur via un câble en nappe de 10 conducteurs. Attention au repérage de la pin 1 sur les connecteurs. Il faut que la pin 1 du programmateur et celle du circuit à programmer coïncident, mais si on a bien monté les connecteurs HE10 à

chaque bout du câble (les triangles sur les HE10 femelles et le trait repère sur le câble en nappe) , tout doit bien se passer. En théorie, en cas d'inversion, le précieux circuit à programmer ne risque rien, vu que c'est lui qui alimente, mais le programmateur risque d'y passer, lui.

Bon, alors tout est connecté ? Alors le premier test du programmateur. On va lui demander d'aller lire quelques informations sur le circuit à programmer, plus précisément les bits de configuration. Pour cela, on clique sur l'icône adaptée, le cadenas :

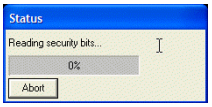


et cela nous ouvre un popup :



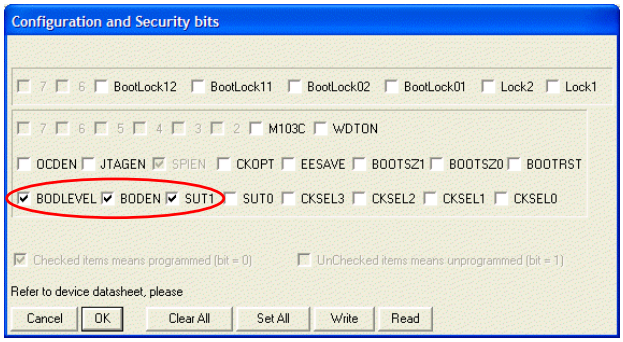
Bon d'accord, c'est un peu hermétique, mais on va y arriver. On va courageusement appuyer sur le bouton « Read ». Le soft va demander au µC sur la carte quels sont les bits de configuration qu'il possède actuellement.

On va alors voir apparaître une boîte de dialogue du genre :



Si tout se passe bien (si le programmateur marche), on va voir la barre de progression brièvement arriver à 100%, et la fenêtre de bits de configuration va se réafficher avec des cases cochées et d'autres non. Ce qui est coché n'est pas important, c'était juste pour vérifier que le programmateur marchait ☺

On va maintenant faire un « Clear All » pour tout remettre à 0, et on va manuellement cocher 3 cases – et pas une de plus – comme sur la figure ci-dessous :



Ne rien cocher d'autre, sinon il y a des chances pour que ça ne fonctionne pas. Une fois que les bonne cases ont été cochées, on va aller écrire cette nouvelle configuration dans le µC du circuit. Il suffit pour cela d'appuyer sur le bouton

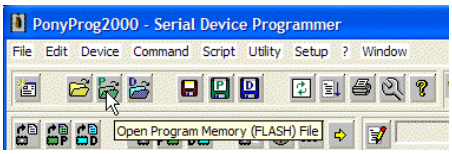
« Write », et d'attendre un peu. Tout doit bien se passer, et pour vérifier, on va aller lire ces bits (bouton « Read ») et normalement les bits cochés ne doivent pas changer.

Un fois ceci terminé, on peut fermer la fenêtre de bits de configuration (« OK ») et passer à la suite.

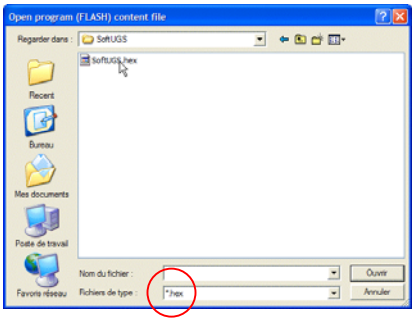
Cette étape d'écriture des différents bits de configuration n'est normalement à faire qu'une seule fois, et si on veut mettre à jour le programme du µC, pas besoin de repasser par là.

On va donc maintenant mettre le fameux programme dans la puce.

Pour cela, il faut charger le code dans PonyProg. On commence par le programme en lui-même, et on appuie donc sur l'icône correspondante :



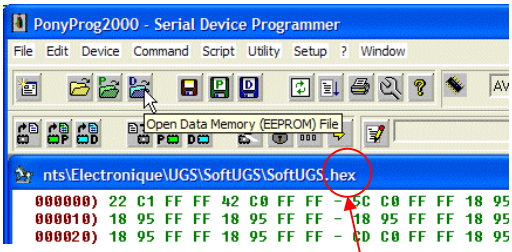
Ca nous ouvre une fenêtre classique de sélection de fichier, à l'aide de laquelle on va aller chercher le fichier :



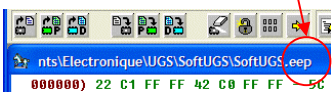
Il faut choisir le fichier avec l'extension **.hex**, c'est celui qui contient le programme. Si jamais aucun fichier n'apparaît, pensez à vérifier le type de fichier, on ne sait jamais…

Une fois le fichier chargé, on voit apparaître un sabir incompréhensible dans la fenêtre du soft. Pas d'inquiétude, c'est normal ☺

On refait la même chose, mais pour le contenu de l'EEPROM du µC, qui se trouve dans le fichier avec l'extension **.eep**. On le charge de la même façon, en utilisant le bon bouton :

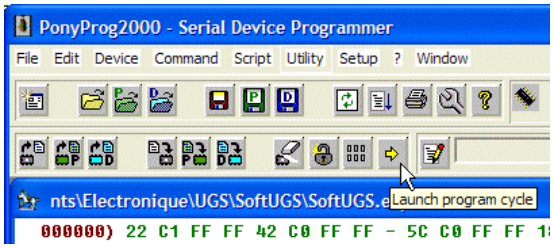


Même boîte de choix de fichier, même procédure.
On ne s'inquiète pas : normalement rien ne change à l'affichage, hormis le titre de fenêtre qui reflète le changement :



En réalité le fichier hexadécimal qu'on vient de charger s'est mis à la suite de l'autre, donc tout au fond du fond de la fenêtre.

Bon, ben on est prêt à lancer la programmation. Suffit d'appuyer sur le bouton…



Et c'est parti mon kiki.

Normalement, plusieurs petites barres de progression vont s'afficher (Erasing, Writing, Verifying…) jusqu'à ce qu'un victorieux « Success » s'affiche.

Après y'a pus qu'à… ☺