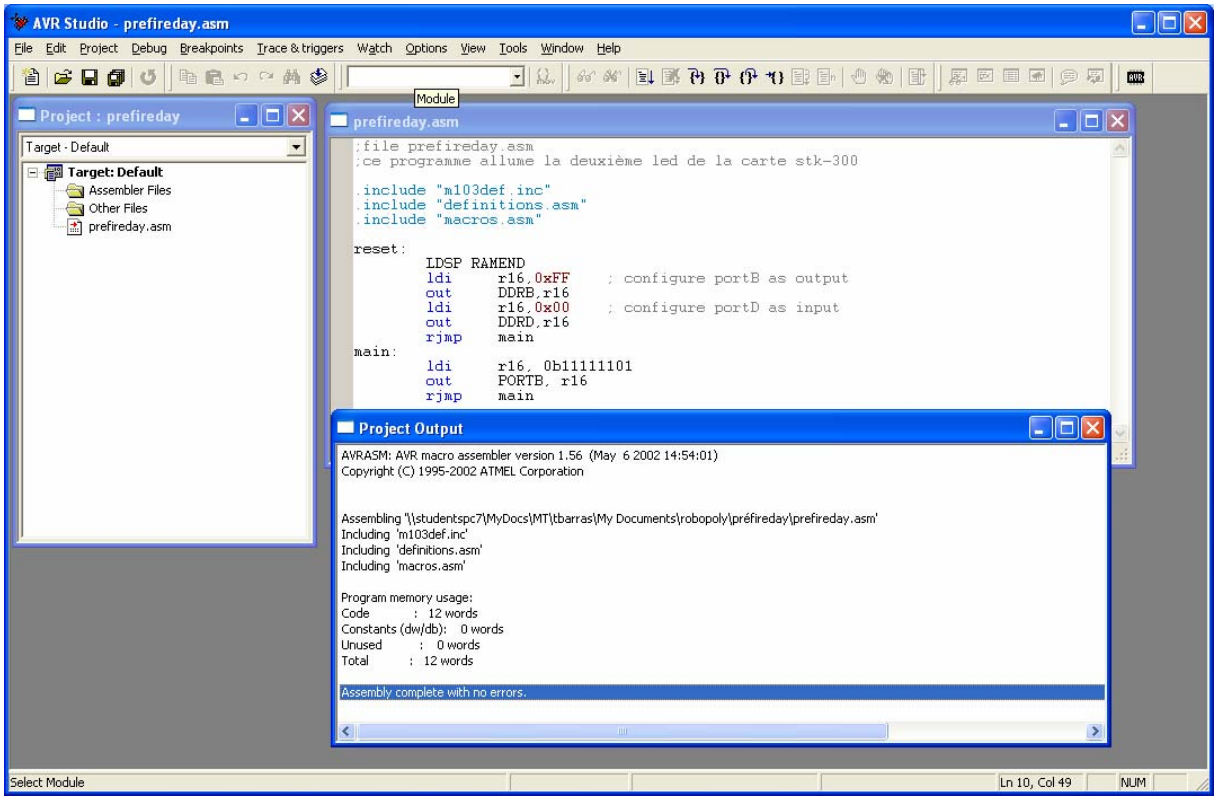
Guide de Programmation

Version 2006-2007



Écrit par Thierry Barras

Ce document explique comment programmer un robot PRisme ou une carte STK-300 avec AVR studio 4 et AVR studio 3.56 puis avec Pony Prog 2000 pour le chargement du programme sur le microcontrôleur.
Ne branchez rien pour l'instant, les explications viendront par la suite…

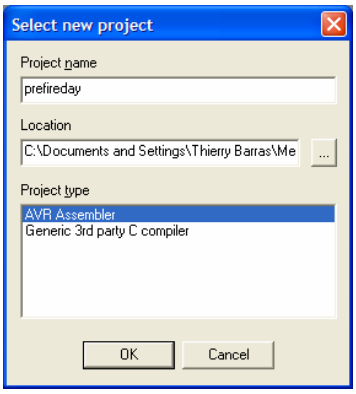
Créer un programme avec AVR

Créer un projet

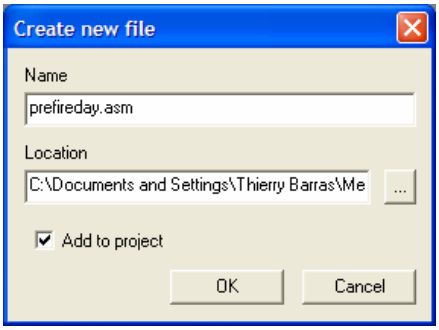
Dans AVR studio 3.56

Démarrez AVR studio 3.56 : *Start->all programs->ATMEL AVR tools->AVR studio 3.56*

Créer votre projet (*project->new*), choisir le dossier de travail et préciser que vous programmez en assembleur (project type : *AVR assembleur*)



Créez un fichier.asm : (*file->new text file*)
Attention : vous devez explicitement ajouter l'extension « .asm » sinon l'assembleur ne reconnait pas le fichier.

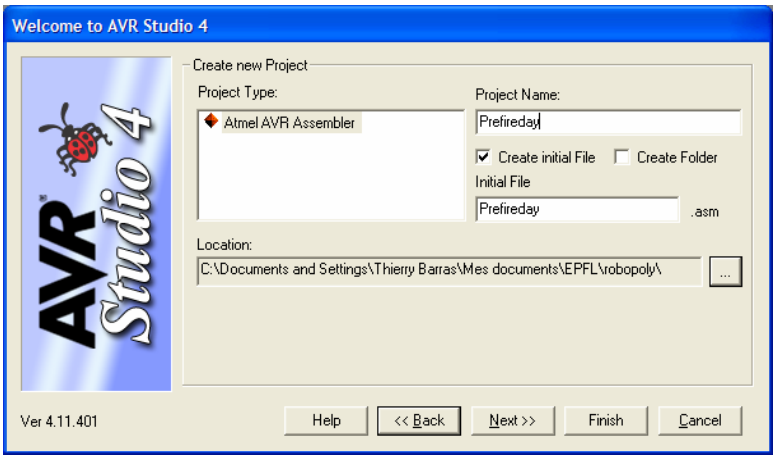


Vérifiez que *Add to project* soit coché

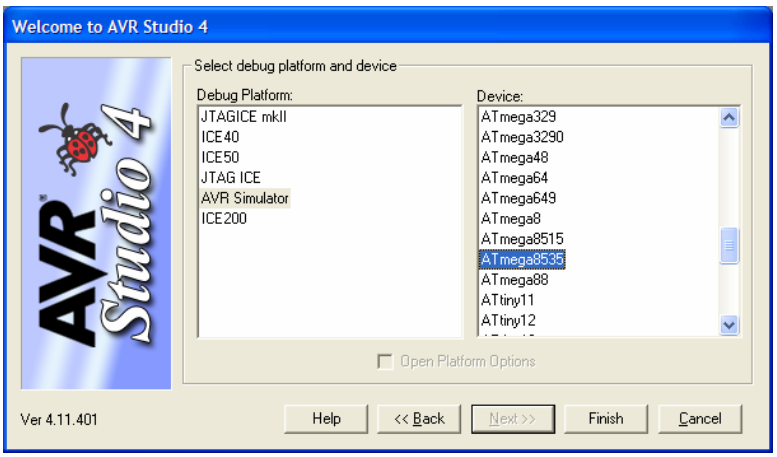
Avec AVR studio 4

Démmarer AVR studio4

Create new project
Entrez un nom de projet et définissez un dossier



Choisissez votre plateforme:
Pour le Prisme : AVR ATmega 8535
Pour la carte STK-300 (prefireday) : AVR ATmega103



Finish.

Votre projet est créé. Un fichier .asm est aussi créé. C'est dans ce dernier qu'il va falloir écrire le code.

Les includes :

Comme nous avons vu pendant les démons, il faut inclure 3 fichiers pour rendre la programmation plus simple :

Copiez les fichiers suivants dans votre dossier de travail

"m8535def.inc"
"definitions.asm"
"macros.asm"

Ces fichiers sont disponibles sur le site de robopoly en téléchargeant :
Rubrique programmation, Prisme.zip ou STK-300.zip

(si vous utilisez la carte STK-300, incluez le fichier "m103def.inc" au lieu de "m8535def.inc")
(si vous voulez utiliser le LCD, vous devez aussi inclure "lcdPRisme.asm")

Débuter la programmation :

Vous devez commencer par écrire en commentaire ce que fait votre programme et son nom.
Vous pouvez aussi mettre le nom de l'auteur, la version du programme ou la date.

Les lignes en gras représentent les lignes de code exactes…

;file prefireday.asm
;ce programme allume la deuxième led de la carte stk-300

Vous devez ensuite lui dire que vous avez inclus des fichiers.
Attention, la suite sera faite pour la carte STK-300. Si vous programmez votre Prisme, pensez à inclure les bons fichiers.

.include "m103def.inc"
.include "definitions.asm"
.include "macros.asm"

Puis vous devez faire le reset de votre programme :
Cette partie sera parcourue qu'une fois, vous pouvez donc définir ici quels ports vont vous servir d'entrée ou de sortie.

reset:

```
LDSP  RAMEND
ldi   r16,0xFF    ; configure portB as output
out   DDRB,r16
ldi   r16,0x00    ; configure portD as input
out   DDRD,r16
rjmp  main
```

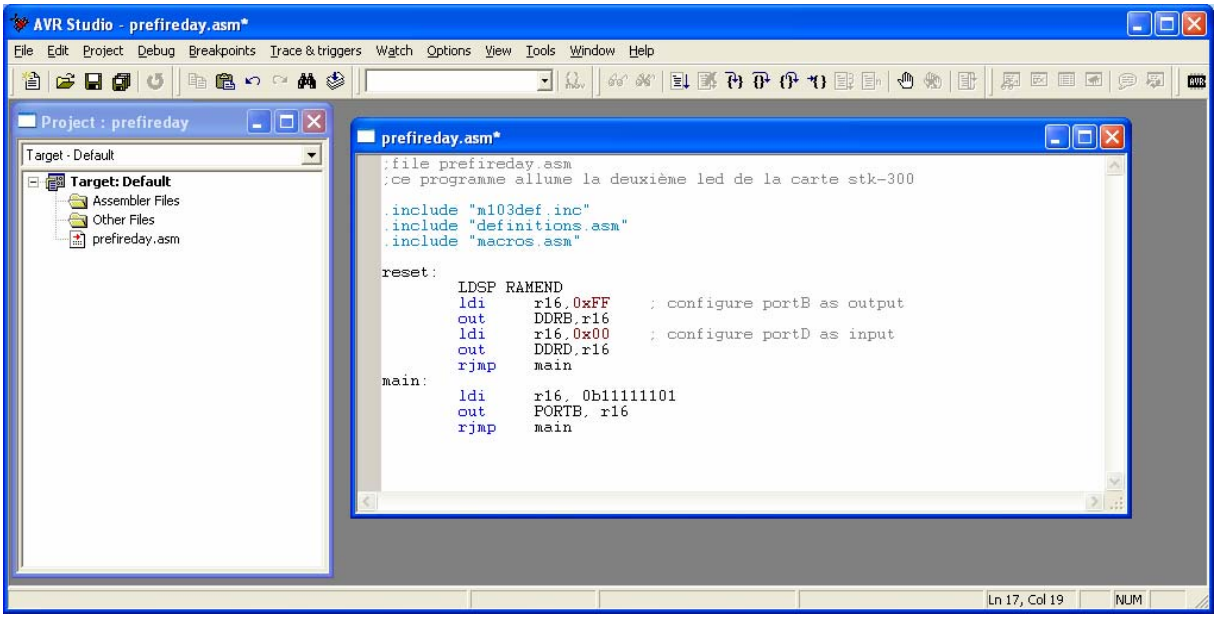
Les leds sont sur le port B, on va donc utiliser le port en sortie. On définit par contre le port D en entrée.(Ici cela ne sert à rien, c'est juste pour l'exemple). Si vous avez des ports non utilisés, vous n'êtes pas obligés de les définir.
Ecrivez toujours **LDSP RAMEND** au début du reset ! Et finissez le reset par rjmp main pour aller dans le programme principal.
(Si vous ne comprenez pas ce bout de code, Une explication concernant le **DDR, PORT et PIN** se trouve en fin de document : fonctionnement des leds et interrupteurs.)

Puis enfin le main.

main:

```
ldi   r16, 0b11111101
out   PORTB, r16
rjmp  main
```

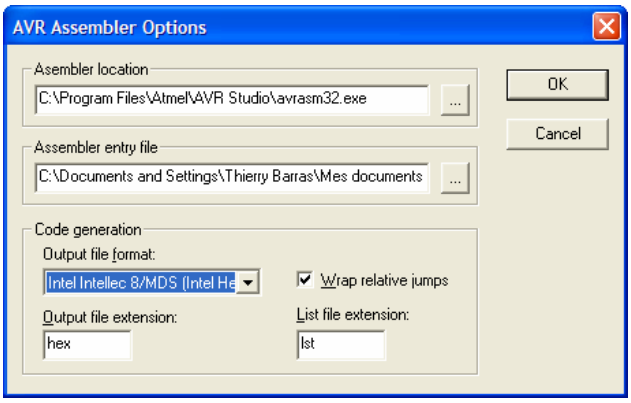
Les leds s'allument lorsque l'on met la valeur du pin à zéro.
A la fin du main, n'oubliez pas de mettre une instruction pour être sûr que le programme restera dans votre boucle, sinon il risque d'exécuter un reste de code non effacé sur le µC.



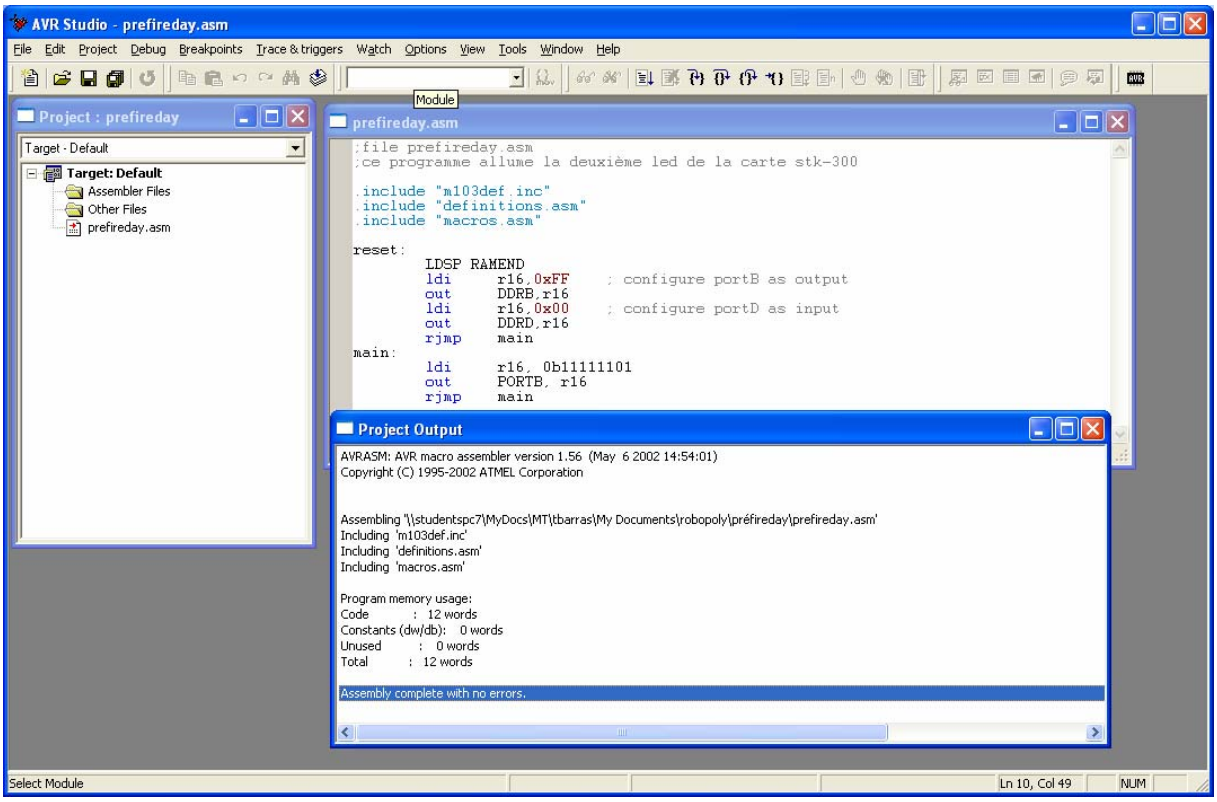
Compiler le programme :

Avant de pouvoir charger un programme vers un système cible, il est nécessaire de générer un fichier objet au format .hex (Intel Hex-extended).

Dans AVR Studio 3.56 :
Sélectionnez cette option dans *Project>Project Settings.*
Dans le *code generation*, choisissez *Intel Intellec 8/MDS (Intel Hex).*



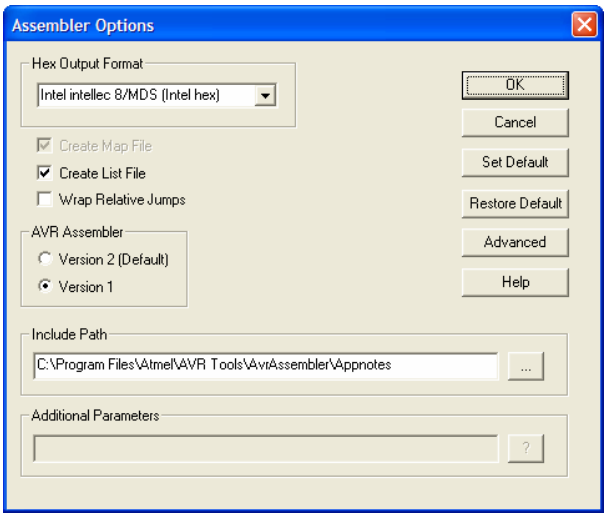
Cliquez *OK*, puis faites *Project->Assemble.*



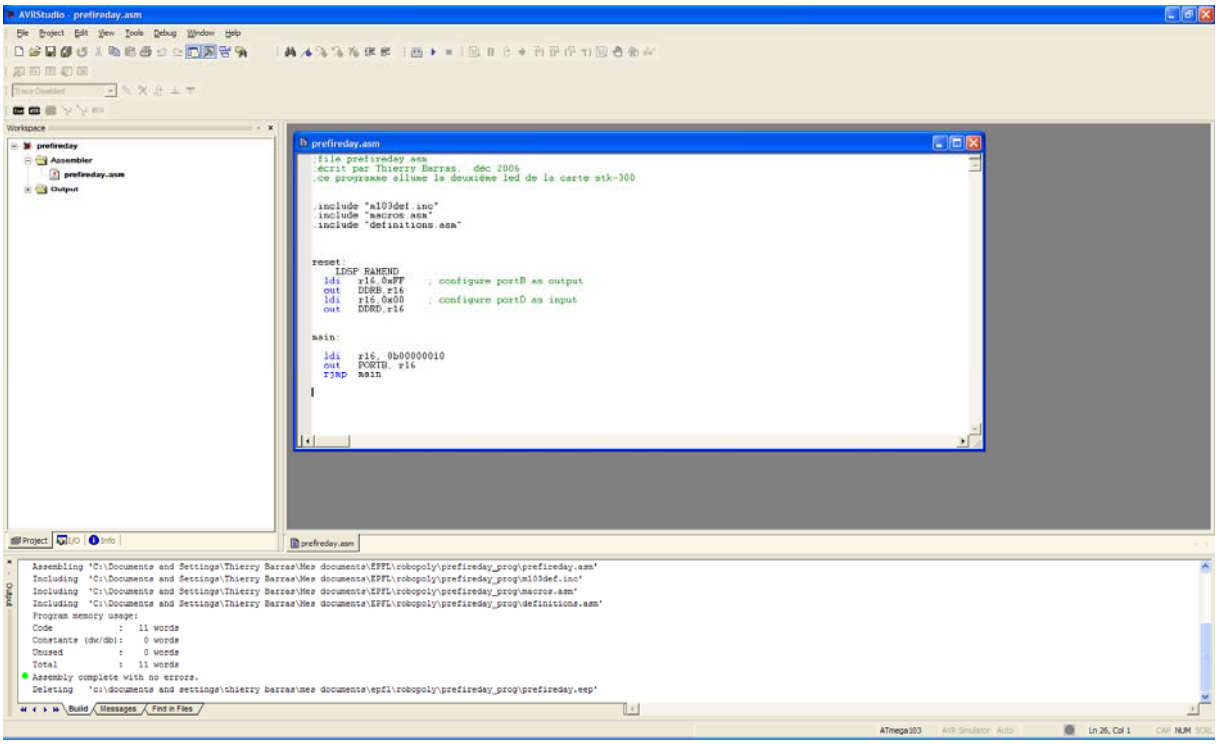
Vous voyez dans le rapport si vous avez fait des erreurs ou non.
Votre fichier .hex a été créé, il se trouve dans votre dossier de travail.

Dans AVR studio 4 :

Il faut tout d'abord aller dans *Project->Assembler Options* puis cocher *Version 1* et choisir *Intel Intellec 8/MDS (Intel Hex).*
Puis cliquez *Set Default.*



Vous pouvez ensuite effectuer *Project-> Build*



Vous voyez dans le rapport si vous avez fait des erreurs ou non.
Votre fichier .hex a été créé, il se trouve dans votre dossier de travail.

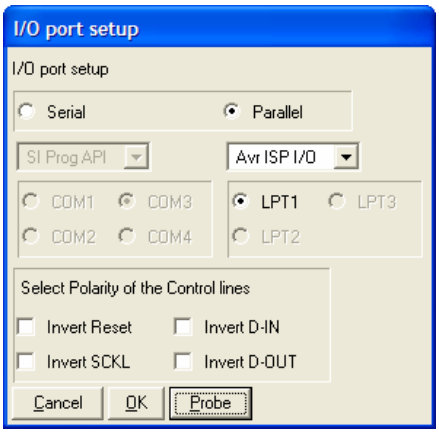
Transfert du programme dans le microcontrôleur avec Pony Prog 2000:

Branchements :

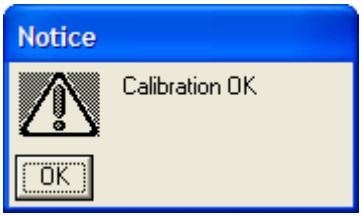
Branchez le dongle dans le port parallèle du PC
Reliez le dongle à la carte STK-300 au moyen du câble gris plat
Connectez le bloc d'alimentation 9V au réseau 220V
Enclenchez la carte au moyen de l'interrupteur noir situé à gauche. La LED ON doit être allumée.

Réglages :

Lancer le programme Pony Prog 2000
Start->All Programs->Pony Prog->Pony Prog
Cliquez *OK.*
Dans le menu *Setup->Interface Setup* choisissez les options suivantes : *Parrallel, Avr ISP I/O et LPT1*. Contrôlez que les 4 cases au dessous ne soient pas cochées. Si vous cliquez sur *Probe*, vous devez recevoir un message «Test Ok ».



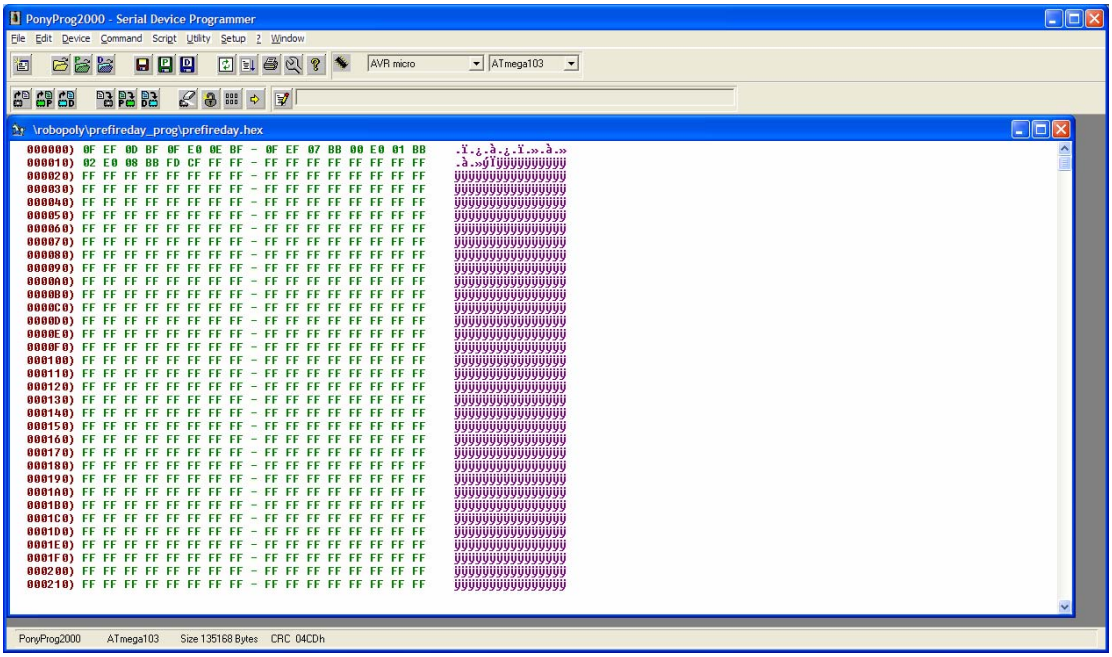
Retournez dans le menu *Setup* et sélectionnez *Calibration*. Appuyer *OK* et attendez que la calibration soit effectuée.



Il faut encore spécifier quel est le processeur cible.
Allez dans *Device->AVR micro->ATmega103.*

Ouvrir et charger le programme FLASH :

Accédez au menu File->Open Program (FLASH) File et séléctionnez le fichier .hex que vous désirez télécharger.
Executez *Command->Erase* pour effacer le contenu actuel de la carte
Exécutez *Command->Write Program (FLASH)*



Votre programme se charge sur la carte. Une fois la vérification terminée, votre programme s'exécute sur la carte. Si vous éteignez la carte, en la rallumant le programme recommencera au début !

Pour modifier un programme :

Il vous suffit de modifier le code dans le fichier.asm dans AVR Studio puis d'assembler pour modifier l'extension .hex.
Avec Pony Prog devez charger la nouvelle version du .hex, donc vous l'ouvrez une nouvelle fois avant de la charger sur la carte.

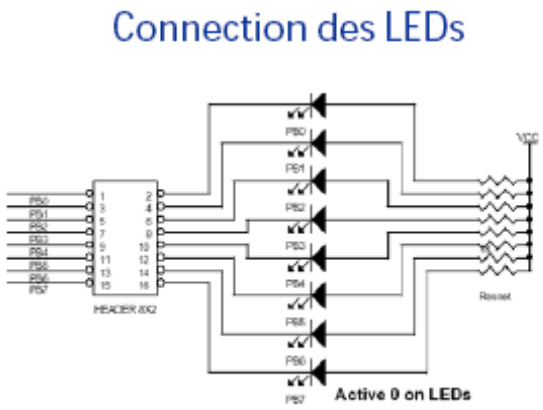
Bonne Chance et amusez vous bien...
Si vous avez des questions, le comité est là pour vous donner un coup de pouce.

Exercices de programmation :

Une liste des instructions du langage assembleur est disponible sur le site de robopoly.
Dans le module programmation : Instructions.pdf

Les solutions de ces quelques exemples se trouvent en fin de document…

Allumez des autres leds
(Mettez une valeur dans un registre et sortez ce registre sur le port B, les leds s'allument lorsque la valeur du pin est à 0.)



La diode est illuminée si un courant la traverse …
donc si la valeur de PBx et à '0'

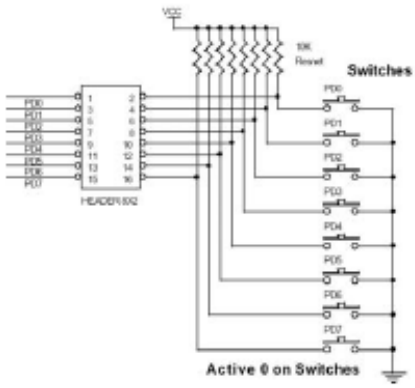
Faite clignotez toute les leds
(utilisez les macros **WAIT_MS** ou **WAIT_C**)

Faitee clignoter une seule led, sans modifier les autres.
(Faites **sbi, cbi** sur un seul pin)

Allumez successivement les leds de la carte
(faire un **rol** du registre avant de le mettre dans le PORTB)

Allumez les leds en fonction des interrupteurs
(les leds sont sur le port B et les interrupteurs sur le port D. Il faut donc définir les leds en sortie et les interrupteurs en entrée. Puis vous récupérez la valeur des boutons dans un registre que vous affectez finalement au port B, donc aux leds. On remarque ici que les interrupteurs sont à 0 quand ils sont appuyés, et à 1 sinon.)

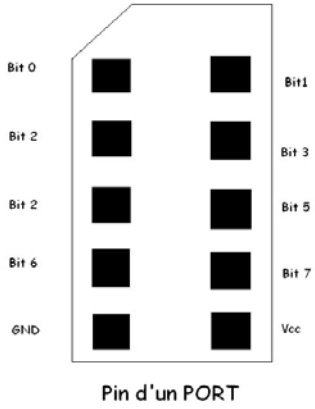
Eteignez les leds en appuyant sur les boutons
(inverser la valeur du registre (**com**) avant de mettre la valeur dans le registre leds)



Lorsqu'on n'appuie pas sur le bouton-poussoir, la résistance tire la ligne vers Vcc
résistance de pull-up

Utilisez vos capteurs IR à la place des boutons
(la carte STK-300 possède des alimentations 4.5V sur chaque ports et une masse. Connectez-y vos capteurs et définissez le port utilisé en entrée puis allez chercher la valeur du pin pour allumer ou éteindre des leds.
Regardez bien le schéma pour ne pas faire griller vos capteurs.
Vcc= 4.5V et GND = masse = 0V

Pour le sens, vous devez repérer l'angle coupé sur le dessin autour du port…



Pin d'un PORT

Faites tourner vos moteurs
(Branchez l'alimentation de vos ponts-H sur une alimentation de la carte, et les fils input des capteurs sur un port. Vous avez donc deux valeurs à donner au pont-H pour qu'il fasse tourner votre moteur. Définissez le port utilisé en sortie et affectez les valeurs **10** ou **01** sur le port pour faire tourner votre moteur. Pour l'arrêter, affectez **00**.)

Faites tourner vos moteurs et arrêter les lorsqu'un capteur détecte quelque chose.
Vous allez faire un petit programme pour vous arrêter devant un obstacle. Ces petits bouts de code sont très pratiques pour les concours. Ils permettent de gagner beaucoup de temps.

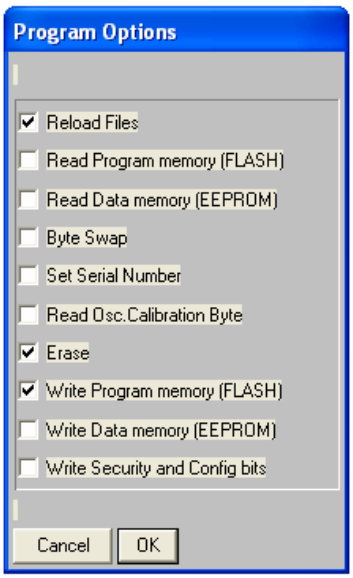
Pour les programmes suivants, ayez de l'imagination et amusez vous bien...

Voici quelques idées pour vous préparer au concours.

-Suivez des lignes noires sur fond blanc (largeur des lignes env. 10mm)
-Arrêter vous devant des obstacles.
-Réagissez aux obstacles, donc reculer, tourner et reprenez votre route.
-Contournez des obstacles.
-Faites demi tour (pour utiliser le temps de rotation par exemple)
-Suivez des murs avec des capteurs IR sur le coté du robot.
-Inventez toute sorte de programmes...

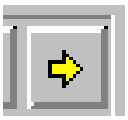
Trucs et astuces :

Utilisation rapide de Pony Prog :



Vous avez remarqué que vous devez à chaque fois effacer le contenu de votre carte, charger le nouveau .hex dans Pony Prog et enfin le télécharger sur votre carte.

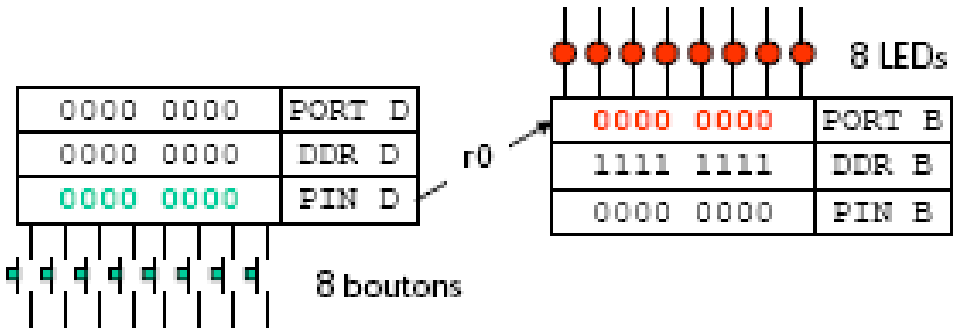
Pour effectuer cela avec une seule touche, aller dans Command->Program Options
et cocher les actions que vous désirez faire :
(Reload file, Erase, Write program memory FLASH)



La flèche jaune (icone)
effectue toute les fonctions cochées. Vous pouvez aussi utiliser (ctrl P)

De plus, lors du chargement, vous pouvez arrêter la vérification en cliquant *Abort*. Cela fonctionne quand même. (malgré qu'il marque failed.)

Fonctionnement des leds et interrupteurs :



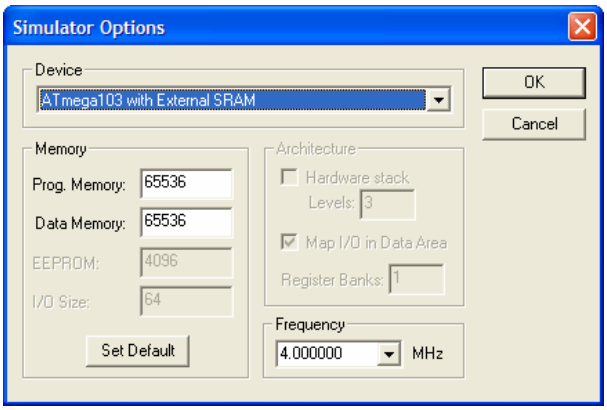
Rappel :

Chaque port contient les registres suivants PORT, DDR, PIN.
Le DDR sert à définir le port en entrée (0) ou en sortie (1) (out DDRB, r16)
Le PORT sert pour envoyer une valeur en sortie (out PORTB, r16)
Le PIN sert à aller chercher une valeur en entrée (in r16, PINB)

Simuler votre programme :

Au lieu de toujours testez votre programme sur la carte, vous pouvez le simuler dans AVR.
Au lieu de faire *Build*, cliquez *Build and Run*

La première fois que vous lancez le simulateur (build and run) dans un nouveau projet, il faut indiquer de quel processeur il s'agit. Choisissez ici *ATmega103 with External SRAM* et choisissez la fréquence de *4 MHz*.
Pour le prisme, choisissez la *ATmega8535*. Fréquence *8MHZ*



Vous pouvez ensuite aller dans l'onglet *Degub* et choisir comment vous debuggez.

Solutions de programmes:

Allume différentes leds.

Valeur du register pour allumer les leds 0, 1, 2, 4, 6:
0b10101000 = 0xA8

Clignoter tout le port :

```
main:
        ldi     r16, 0b01010101
        out     PORTB, r16
        WAIT_MS 100

        ldi     r16, 0b10101010
        out     PORTB, r16

        WAIT_MS     100
        rjmp    main
```

Faire clignoter une seule led, sans modifier les autres.

```
main:
        ldi     r16, 0xA8
        out     PORTB, r16

blink_led2:
        sbi     PORTB, 2
        WAIT_MS 100
        cbi     PORTB, 2
        WAIT_MS     100
        rjmp    blink_led2
```

Allumez successivement les leds de la carte

```
main:
        ldi     r20, 0b11111110
        SEC     ;pour mettre le carry à 1 car il va tourner avec le registre

move_led:

        rol     r20
        out     PORTB, r20
        WAIT_MS 100

        rjmp move_led
```

Allume les leds en appuyant sur les boutons.

```
main:
        in      r16, PIND
        out     PORTB, r16
        rjmp    main
```

Eteindre les leds en appuyant sur les boutons.

```
main:
        in      r16, PIND
        com     r16
        out     PORTB, r16
        rjmp    main
```